ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Υλοποίηση μοντέλων Βαθιάς Μάθησης σε περιβάλλον MATLAB – εφαρμογή στην πρόβλεψη χρονοσειρών και στην ταξινόμηση»

ΑΛΕΞΑΝΔΡΙΝΑ – ΙΟΥΛΙΑΝΑ ΙΛΙΕΣ

ΑΜ: 711171007

Εισηγητής: ΠΑΡΙΣ ΜΑΣΤΟΡΟΚΩΣΤΑΣ ΚΑΘΗΓΗΤΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Υλοποίηση μοντέλων Βαθιάς Μάθησης σε περιβάλλον MATLAB – εφαρμογή στην πρόβλεψη χρονοσειρών και στην ταξινόμηση»

ΑΛΕΞΑΝΔΡΙΝΑ – ΙΟΥΛΙΑΝΑ ΙΛΙΕΣ

ΑΜ: 711171007

Εισηγητής:

ΠΑΡΙΣ ΜΑΣΤΟΡΟΚΩΣΤΑΣ ΚΑΘΗΓΗΤΗΣ

Εξεταστική Επιτροπή:

ΧΡΗΣΤΟΣ ΤΡΟΥΣΣΑΣ, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΠΑΝΑΓΙΩΤΑ ΤΣΕΛΕΝΤΗ, ΕΔΙΠ

Ημερομηνία εξέτασης 22/7/2024

Εγκρίθηκε από την τριμελή επιτροπή την 22η Ιουλίου 2024		
Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2024.07.24 07:47:20 +03'00' Πάρις Μαστοροκώστας Καθηγητής	Christos Troussas Digitally signed by Christos Troussas Date: 2024.07.24 08:39:57 +03'00' Χρήστος Τρούσσας Επίκουρος Καθηγητής	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2024.07.26 21:14:48 +03'00' Παναγιώτα Τσελέντη ΕΔΙΠ

Αθήνα - Αιγάλεω Ιούλιος 2024

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη ΑΛΕΞΑΝΔΡΙΝΑ – ΙΟΥΛΙΑΝΑ ΙΛΙΕΣ του ΑΛΕΞΑΝΔΡΟΥ, με αριθμό μητρώου 711171007, φοιτήτρια του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς, είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

«Βεβαιώνω ότι είμαι συγγραφέας της παρούσας διπλωματικής εργασίας και ότι έχω αναφέρει ή παραπέμψει σε αυτή, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη διπλωματική εργασία»

Η Δηλούσα,

Ilies

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ το Θεό, την Πανάγια, τον π. Ιωάννη Δρογγίτη, τη γλυκιά μου την κυρία Σοφία, τον Πολύκαρπο Βενέτη, τον Αθανάσιο Ουζουνίδη, την αδελφή μου την Έλενα και τον σύζυγο της Ματέι, τον επιβλέποντα καθηγητή μου κ. Πάρι Μαστοροκώστα και όλους τους ανθρώπους που ήταν δίπλα μου όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Ο σκοπός της παρούσας διπλωματικής εργασίας αφορά την υλοποίηση μοντέλων Βαθιάς Μάθησης και την εφαρμογή τους στην πρόβλεψη και στην ταξινόμηση χρονοσειρών με τη βοήθεια των αναδρομικών και των συνελικτικών τεχνητών νευρωνικών αρχιτεκτονικών δομών, οι οποίες εντάσσονται στις δυνατότητες του λογισμικού MATLAB.

ABSTRACT

This thesis investigates the application of deep learning models for two common types of machine learning problems: time series prediction and image classification. We focus on utilizing recursive and respectively, convolutional artificial neural network architectures available within the MATLAB software environment.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: ΒΑΘΙΑ ΜΑΘΗΣΗ

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Deep Learning, Νευρωνικά δίκτυα, ταξινόμηση, πρόβλεψη

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

PreLU - Parametric ReLU

GD - Gradient Descent

SLP - Single layer Perceptron

MLP- Multi - Layers Perceptrons

CNN - Convolutional Neural Networks

RNN - Simple Recurrent Networks

LSTM - Long - Short Term Memory

GRU - Gated Recurrent Units

ΚΕΦΑΛΑΙΟ 1

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της διπλωματικής εργασίας και γίνεται μια ιστορική αναδρομή καθώς και μια σύντομη περιγραφή των αρχιτεκτονικών νευρωνικών δικτυών και μεθόδων της συγκεκριμένης περιοχής.

1.1. Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Το αντικείμενο της παρούσας διπλωματικής εργασίας αφορά την υλοποίηση μοντέλων Βαθιάς Μάθησης και την εφαρμογή τους στην πρόβλεψη και στην ταξινόμηση χρονοσειρών με τη βοήθεια των αναδρομικών και των συνελικτικών τεχνητών νευρωνικών αρχιτεκτονικών δομών, οι οποίες εντάσσονται στις δυνατότητες του λογισμικού MATLAB.

Στα πλαίσια της θεωρητικής μελέτης καταγράφονται συνοπτικά ορισμένες αρχιτεκτονικές τεχνητών νευρωνικών δικτύων, όπως τα δίκτυα Perceptron μονής και πολλαπλής στρώσης, τα Συνελικτικά Δίκτυα (CNNs), τα Επαναλαμβανόμενα Δίκτυα (RNNs), οι αλγόριθμοι που χρησιμοποιούν, καθώς και οι τεχνικές που υλοποιούνται για τη βελτίωση της απόδοσής τους.

Στο πρακτικό μέρος της έρευνας υλοποιούνται και εκπαιδεύονται ένα Συνελικτικό Δίκτυο (CNN) και δυο Επαναλαμβανόμενα Δίκτυα (LSTM και GRU), χρησιμοποιώντας δυο ξεχωριστά σύνολα δεδομένων. Ένα σύνολο δεδομένων για το CNN, το οποίο περιλαμβάνει λουλούδια και θα χρησιμοποιηθεί για την εργασία της ταξινόμησης εικόνων (Image Classification) και ένα δεύτερο σύνολο, το οποίο περιλαμβάνει ιστορικά δεδομένα τιμών για τρία διαφορετικά κρυπτονομίσματα, που θα χρησιμοποιηθεί στην πρόβλεψη χρονοσειρών. Στη συνέχεια τα υλοποιημένα μοντέλα δοκιμάζονται ως προς την απόδοση τους, χρησιμοποιώντας μετρήσεις απόδοσης και εφαρμόζονται μεθοδολογίες που οδηγούν στη βελτίωσή τους.

1.2. Ιστορική Αναδρομή

Κατά τις δεκαετίες 1940-1950 εισήχθη για πρώτη φορά η έννοια των τεχνητών νευρωνικών δικτύων (ANNs) από τους Warren McCulloch και Walter Pitts. Οι McCulloch & Pitts πρότειναν ένα μαθηματικό μοντέλο που μιμούνταν ορισμένες πτυχές λειτουργίας του ανθρωπίνου εγκέφαλου. Το συγκεκριμένο μοντέλο που αναπτύχθηκε ήταν ένα μονοεπίπεδο νευρωνικό δίκτυο (Perceptron single layer), ικανό να μαθαίνει και να κάνει δυαδικές ταξινομήσεις.

Το 1969 ο Minsky με τον Seymour Papert συγγράψανε το βιβλίο Perceptrons, το οποίο περιλαμβάνει την ανάλυση των τεχνιτών νευρωνικών δικτύων, η οποία βρισκόταν σε αρχικό στάδιο μελέτης. Η έρευνα των νευρωνικών δικτύων τους οδήγησε στο συμπέρασμα ότι οι μεμονωμένοι νευρώνες καθώς και τα νευρωνικά δίκτυα μονής στρώσης δεν μπορούσαν να εφαρμόσουν κάποιες λογικές λειτουργίες όπως η πράξη «Αποκλειστικό Η» (XOR). Αργότερα εξακριβώθηκε ότι τα δίκτυα πολλαπλών

επίπεδων υλοποιούσαν τις συγκεκριμένες λογικές πράξεις. Ωστόσο το βιβλίο συνεισέφερε στη συμβολική τεχνητή νοημοσύνη, που επιχειρεί να κατανοήσει τις νοητικές διεργασίες του ανθρωπίνου εγκεφάλου και την προσομοίωση της ανθρώπινης νοημοσύνης αλγοριθμικά, χρησιμοποιώντας σύμβολα και λογικούς κανόνες υψηλού επιπέδου.

Τα πολυεπίπεδα νευρωνικά δίκτυα αναλυθήκαν το 1960 αλλά μελετήθηκαν αργότερα στη δεκαετία του 1980, όταν ο Peter Jackson παρουσίασε τα συστήματα βασισμένα στη γνώση ή τα λεγόμενα «έμπειρα» συστήματα με τη βοήθεια των χαρτών αυτό-οργάνωσης με χρήση την ανταγωνιστική μάθηση, που εισήχθησαν στη δεκαετία του 1970. Ο ίδιος ο Jackson περιγράφει στο βιβλίο του το έμπειρο σύστημα ως εξής: «Ένα έμπειρο σύστημα είναι ένα πρόγραμμα υπολογιστή, που αναπαριστά και αιτιολογεί με γνώση κάποιο εξειδικευμένο αντικείμενο, με σκοπό την επίλυση προβλημάτων ή την παροχή συμβουλών».

Επίσης στη δεκαετία του 1980 κυκλοφόρησε και ο πρώτος αλγόριθμος μηχανικής μάθησης backpropagation, ξεκίνησαν οι έρευνες του ανθρωπίνου εγκέφαλου και οι ερευνητές της τεχνητής νοημοσύνης εφάρμοσαν μαθηματική και στατιστική ανάλυση για να αναπτυχθούν κατάλληλοι αλγόριθμοι για τα τεχνητά αυτά νευρωνικά δίκτυα.

Στη δεκαετία του 1990 ο Vladimir Vapnic και οι συνεργάτες του ανέπτυξαν τις μηχανές διανύσματος υποστήριξης (Support Vector Machine). Στις συγκεκριμένες μηχανές εφαρμόζεται ένα σύνολο μεθόδων για τη λύση προβλημάτων, που σχετίζονται με την αναγνώριση προτύπων, με την ταξινόμηση, με την παλινδρόμηση καθώς και άλλες λειτουργίες επεξεργασίας δεδομένων.

Στα μέσα της δεκαετίας του 2000 οι ερευνητές ξεκίνησαν να χρησιμοποιούν τα νευρωνικά δίκτυα πολλαπλών επίπεδων (deep architectures) και ανακάλυψαν ότι οι αρχιτεκτονικές σε βάθος θα μπορούσαν να εξαγάγουν σημαντικά χαρακτηριστικά από ακατέργαστα δεδομένα. Η εργασία του Geoffrey Hinton για την εκπαίδευση δικτύων βαθιάς πεποίθησης και η ανάπτυξη τεχνικών προκατάρτισης χωρίς επίβλεψη συνέβαλαν στην αναζωπύρωση του ενδιαφέροντος για τη βαθιά μάθηση.

Το έργο του Ian Goodfellow πάνω στην ανάπτυξη δικτύων γεννητικής αντιπαράθεσής (GAN), τα οποία έφεραν την επανάσταση στο πλαίσιο της δημιουργίας συνθετικών δεδομένων και εικόνων, άσκησε μεγάλη επιρροή στην προώθηση του πεδίου της μάθησης χωρίς επίβλεψη.

Ο Yoshua Bengio, γνωστός για την πρωτοποριακή του ερευνά στη βαθιά μάθηση και στα νευρωνικά δίκτυα, με την εργασία του πάνω στις αρχιτεκτονικές νευρωνικών δικτύων και στους αλγορίθμους βαθιάς μάθησης, έβαλε τα θεμέλια για την προώθηση της κατανόησης και των εφαρμογών της τεχνητής νοημοσύνης.

Ο Aaron Courville αναγνωρίστηκε για τη συμβολή του στη θεωρία βαθιάς μάθησης και τις εφαρμογές της. Το έργο του συχνά επικεντρώνεται στην κατανόηση των βασικών αρχών των νευρωνικών δικτύων και των εφαρμογών τους σε διάφορους τομείς.

Ο Ian Goodfellow, ο Yoshua Bengio και ο Aaron Courville συνέγραψαν το βιβλίο με τίτλο «Deep Learning», το οποίο αποτέλεσε βασική πηγή για πολλούς ερευνητές,

φοιτητές και επαγγελματίες στον τομέα. Η συνεισφορά τους έχει διαμορφώσει σημαντικά την κατανόηση και την ανάπτυξη μεθοδολογιών βαθιάς μάθησης και νευρωνικών δικτύων.

1.3. Νευρωνικά Δίκτυα

Το νευρικό μας σύστημα αποτελείται από το Περιφερικό Νευρικό Σύστημα και το Κεντρικό Νευρικό Σύστημα. Μεταξύ των άλλων δομικών στοιχείων περιλαμβάνει και τα νευρικά κύτταρα, δηλαδή τους νευρώνες. Οι βιολογικοί νευρώνες είναι κύτταρα που σχηματίζουν ένα πολύπλοκο και εξαιρετικά διασυνδεδεμένο δίκτυο. Στέλνουν ηλεκτρικά σήματα ο ένας στον άλλον, εξειδικεύονται στο να διακινούν, να δέχονται, να επεξεργάζονται μηνύματα και να μεταβιβάζουν ερεθίσματα σε περιοχές του εγκέφαλου που ευθύνονται για τον έλεγχο, το συντονισμό, την ερμηνεία και τη δημιουργία των γενικών αισθήσεων και των συναισθημάτων (αφή, πίεση, πόνος, χαρά, όραση, ακοή), για τις εκούσιες κινήσεις, για τη χρήση λέξεων και κατανόηση του λογού, για την έκφραση σκέψεων και συναισθημάτων, για την εκτίμηση των αποτελεσμάτων της συμπεριφοράς, καθώς και για τη συγκέντρωση, την αντίληψη, το σχεδιασμό και τη λύση προβλημάτων. Τα μηνύματα αυτά συλλέγονται από τους δεντρίτες, που σαν βασική λειτουργία έχουν τη λήψη των ερεθισμάτων μέσω των συνάψεων με άλλους νευρώνες και ονομάζονται νευρικές ώσεις (σήματα). Οι βιολογικές νευρικές δομές μεταβάλλονται κατά τη διάρκεια ζωής του ανθρώπου, αναπτύσσονται νέοι νευρώνες στον ιππόκαμπο του εγκέφαλου, ο οποίος σχετίζεται άμεσα με τη μνήμη και τη μάθηση και έχουν θετική επιρροή όσον αφορά το στρες, τα συναισθήματα, τα κίνητρα, την προσοχή, τη μνήμη, τη μάθηση. Αφού αναπτυχθούν οι καινούργιοι νευρώνες στη συνέχεια ενώνονται με το νευρικό δίκτυο. Η νευρογένεση στην πραγματικότητα δίνει τη δυνατότητα στον άνθρωπο, καθ' όλη τη διάρκεια της ζωής του, να αποκτήσει γνώση, να εκπαιδεύσει τον εαυτό του, να αποκτά καινούριες δεξιότητες και ικανότητες.

Από τις μελέτες που πραγματοποιήθηκαν στο Κεντρικό Νευρικό Σύστημα και συγκεκριμένα πάνω στον ανθρώπινο εγκέφαλο προέκυψε ότι οι λειτουργίες του εγκεφάλου είναι πολύ σημαντικές και ότι μέσω των λειτουργιών αυτών ο άνθρωπος, αντιλαμβάνεται τις μεταβολές του περιβάλλοντος, συλλέγει και επεξεργάζεται πληροφορίες και ανάλογα δίνει και τις κατάλληλες εντολές. Τον τελευταίο αιώνα έχει παρουσιαστεί σημαντική πρόοδος, καθώς με τη βοήθεια των ηλεκτρονικών υπολογιστών ξεκίνησε η προσπάθεια προσομοίωσης των λειτουργιών του εγκεφάλου και προέκυψαν τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ), τα οποία επεξεργάζονται πληροφορίες χρησιμοποιώντας τη συνδετική προσέγγιση (connectionist approach) στους υπολογισμούς που πραγματοποιούν.

1.4. Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ)

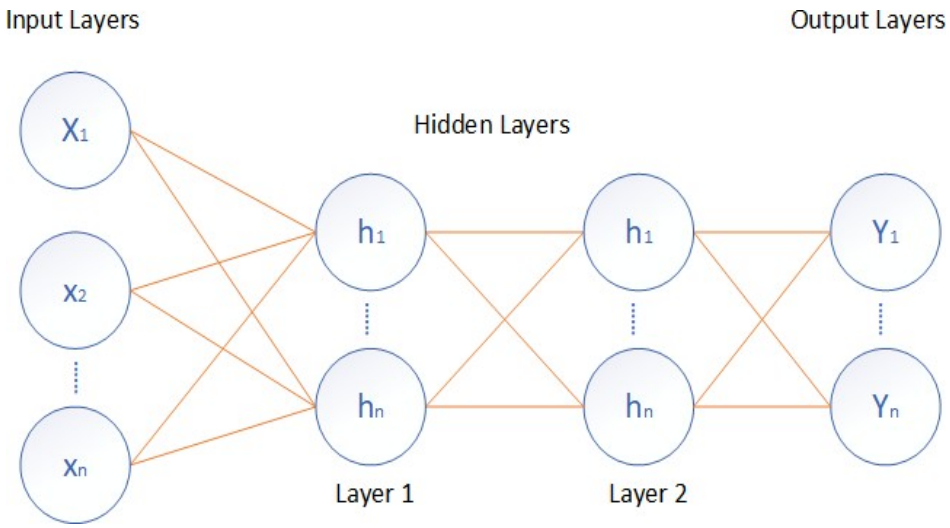
Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από τους τεχνητούς νευρώνες που συνεργάζονται μεταξύ τους και εκτελούν περίπλοκες εργασίες. Οι τεχνητοί νευρώνες είναι μονάδες λογισμικού που ονομάζονται κόμβοι και τα τεχνητά νευρωνικά δίκτυα είναι προγράμματα λογισμικού ή αλγόριθμοι, που στον πυρήνα τους χρησιμοποιούν υπολογιστικά συστήματα για την επίλυση μαθηματικών υπολογισμών.

Ένα βασικό νευρωνικό δίκτυο έχει διασυνδεδεμένους τεχνητούς νευρώνες σε τρία επίπεδα:

- Επίπεδο εισόδου (Input Layer)
- Κρυφό επίπεδο (Hidden Layer)
- Επίπεδο εξόδου (Output Layers)

Οι πληροφορίες εισέρχονται στο νευρωνικό δίκτυο, στο επίπεδο εισόδου. Στη συνέχεια οι κόμβοι εισόδου μεταβιβάζουν τα δεδομένα στο κρυφό επίπεδο. Τα τεχνητά δίκτυα μπορούν να έχουν μεγάλο αριθμό κρυφών επιπέδων. Σε κάθε κρυφό στρώμα αναλύονται, επεξεργάζονται και εκτελούνται όλοι οι υπολογισμοί και στη συνέχεια τα δεδομένα είτε τροφοδοτούνται στην είσοδο κάποιου άλλου κρυφού στρώματος είτε κατευθείαν στο επίπεδο εξόδου. Όπως και τα προηγούμενα επίπεδα το επίπεδο εξόδου μπορεί να έχει από έναν έως και πολλούς κόμβους. Σε αυτό το επίπεδο μετά τις απαραίτητες εργασίες που έχουν πραγματοποιηθεί στα προηγούμενα επίπεδά παράγεται το τελικό αποτέλεσμα.

Στο **Σχήμα 1.4.α** παρουσιάζεται η βασική αρχιτεκτονική ενός Τεχνητού Νευρωνικού Δικτύου με κρυφά στρώματα:



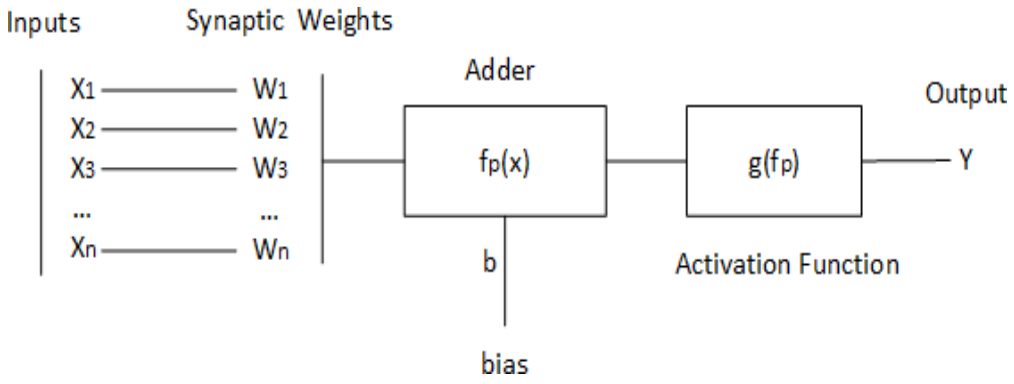
Σχήμα 1.4.α: Νευρωνικό δίκτυο με κρυφά στρώματα.

Η μονάδα λειτουργίας ενός βασικού μοντέλου τεχνητού νευρωνικού δικτύου, μπορεί εύκολα να κατανοηθεί ως μια μαθηματική συνάρτηση $f_p(x)$ με είσοδο ένα διάνυσμα n διαστάσεων x όπου $x=x_0,x_1,x_2,x_3,\ldots,x_n$ σταθμισμένο από ένα άλλο διάνυσμά n διαστάσεων με βάρη w όπου $w=w_1,w_2,w_3,\ldots,w_n$, στα οποία προστίθεται μια νευρωνική πόλωση (bias) b, που ενεργοποιείται από μια συνάρτηση μεταφοράς $g(f_p)$, η οποία τροφοδοτεί την έξοδο y, όπως φαίνεται στο **Σχήμα 1.4.β**, και δίνεται από τις παρακάτω μαθηματικές σχέσεις:

$$y = f_p(x) = f_p(x_1w_1 + x_2w_2 + x_3w_3 + \cdots + x_nw_n + b) \qquad (\alpha)$$

$$y = f_p(x) = \sum_{i=1}^{n} x_i \; w_i + b \qquad (\beta)$$

$$\text{και} \quad y(x) = g(f_p) \qquad (\gamma)$$



Σχήμα 1.4.β: Μονάδα Λειτουργίας Βασικού Νευρωνικού Δικτύου.

Οι συνδέσεις μεταξύ των νευρώνων αντιπροσωπεύονται από βάρη, τα οποία αποτελούν παραμέτρους που μπορούν να αλλάξουν και να προσαρμοστούν κατά τη διάρκεια της εκπαιδευτικής διαδικασίας για τη βελτιστοποίηση της απόδοσης του δικτύου, αρά ουσιαστικά κάθε σύναψη μεταξύ των νευρώνων έχει ένα σχετικό βάρος που καθορίζει την ισχύ της σύνδεσης.

Οι σταθεροί όροι (biases) είναι πρόσθετες παράμετροι, που προστίθενται στο σταθμισμένο άθροισμα πριν από την εφαρμογή της συνάρτησης ενεργοποίησης, όπως και τα βάρη ενημερώνονται και αυτά κατά τη διάρκεια της εκπαίδευσης για να ελαχιστοποιηθεί το σφάλμα πρόβλεψης του δικτύου.

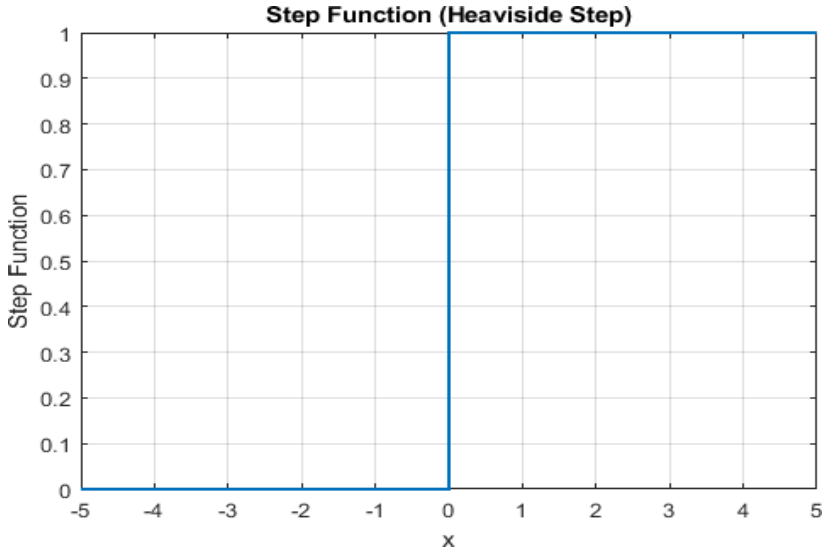
Οι τεχνητοί νευρώνες επεξεργάζονται ταυτόχρονα τις εισόδους τους, επιτρέποντας στα νευρωνικά δίκτυα να εκτελούν παράλληλους κατανεμημένους υπολογισμούς, να μοντελοποιήσουν πολύπλοκες σχέσεις στα δεδομένα και να μαθαίνουν από παραδείγματα.

Οι συναρτήσεις ενεργοποίησης εισάγουν τη μη γραμμικότητα στο δίκτυο επιτρέποντάς του να μαθαίνει πολύπλοκα μοτίβα καθώς επίσης καθορίζουν αν ο νευρώνας πρέπει να «πυροδοτεί» ή όχι με βάση την είσοδο του. Οι πιο γνωστές συναρτήσεις ενεργοποίησης είναι οι ακόλουθες:

- **Η βηματική συνάρτηση (Heaviside or Step function**) τροφοδοτείται με μια τιμή που εξαρτάται από την τιμή κατωφλίου. Αν η τιμή της εισόδου είναι μεγαλύτερή από το όριο, τότε ο νευρώνας ενεργοποιείται και χρησιμοποιείται για τη λύση προβλημάτων που σχετίζονται με τη δυαδική ταξινόμηση (binary classification). Ωστόσο δεν παρέχει εξόδους πολλαπλών τιμών και δεν είναι αποδοτική σε προβλήματα ταξινόμησης πολλών κλάσεων. Λόγω του ότι είναι μη παραγωγήσιμη στο σημείο που μεταβαίνει από το 0 στο 1 (στο x=0), που δείχνει ότι υπάρχει μια απότομη αλλαγή στην τιμή της χωρίς μια συνεχή μετάβαση, με αποτέλεσμα μια «άπειρη» κλήση της εφαπτομένης σε εκείνο το σημείο που προκαλεί εμπόδια στη διαδικασία οπισθοδιάδοσης (backpropagation). Η βηματική συνάρτηση ορίζεται από την παρακάτω μαθηματική σχέση:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

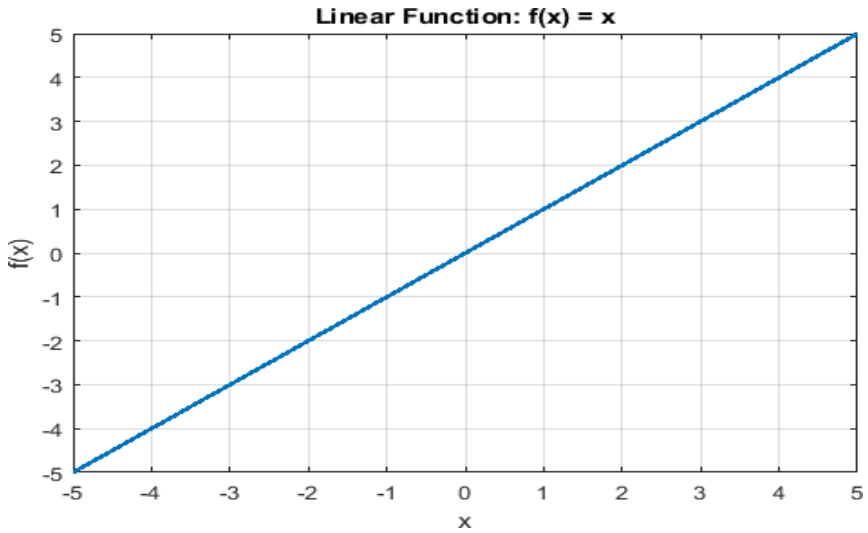
και απεικονίζεται στο **Σχήμα 1.4.γ.**



Σχήμα 1.4.γ: Η Βηματική Συνάρτηση.

- **Η γραμμική συνάρτηση (Linear function)** είναι παραγωγήσιμη σε όλα τα σημεία ακόμα και στο σημείο x=0. Η παράγωγος της γραμμικής συνάρτησης ισούται με 1 για όλα τα x, που σημαίνει ότι σε κρυφά επίπεδα η συνάρτηση περιορίζει σοβαρά την ικανότητα του δικτύου να μαθαίνει πολύπλοκες μη γραμμικές σχέσεις και αυτό μειώνει την αναπαραστατική ισχύ του μοντέλου, μετατρέποντας το σε ένα νευρωνικό δίκτυο γραμμικής παλινδρόμησης. Η συνάρτηση δίνεται από τον παρακάτω τύπο:

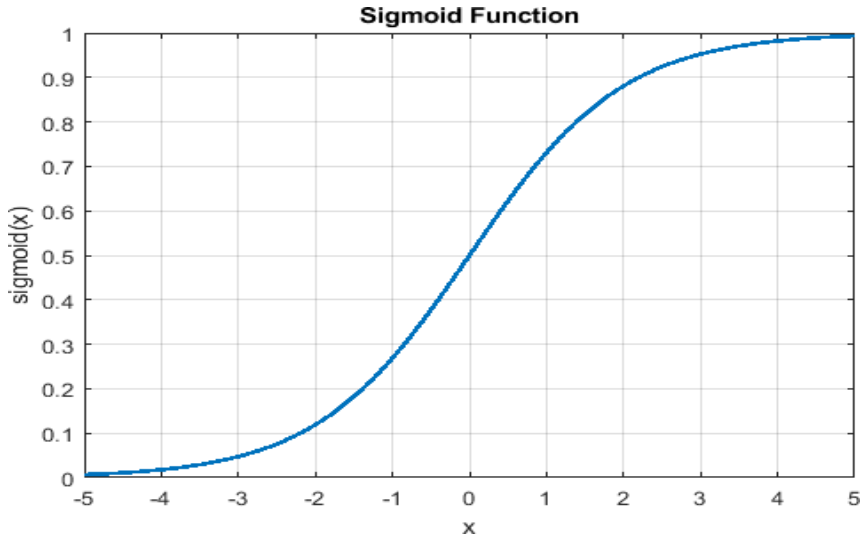
$$f(x) = x$$



Σχήμα 1.4.δ: Η Γραμμική Συνάρτηση f(x)=x.

- **Η σιγμοειδής συνάρτηση (Sigmoid Function)**, γνωστή και ως σιγμοειδής καμπύλη, η οποία φράζει της εξόδους των νευρώνων στις τιμές 0 και 1, είναι ιδανική για προβλήματα δυαδικής κατηγοριοποίησης (binary classification) και λογιστικής παλινδρόμησης (logistic regression). Ωστόσο η έξοδος της σιγμοειδούς συνάρτησης δεν είναι συμμετρική γύρω από το μηδέν. Αυτό έχει σαν αποτέλεσμα η έξοδος όλων των νευρώνων να είναι του ιδίου πρόσημου και να δημιουργεί αστάθεια κατά την εκπαίδευση του νευρωνικού δικτύου. Η συνάρτηση ορίζεται από την ακόλουθη μαθηματική έκφραση και η γραφική παράσταση φαίνεται στο **Σχήμα 1.4.ε**.

$$f(x) = \frac{1}{1 + e^{-x}}$$

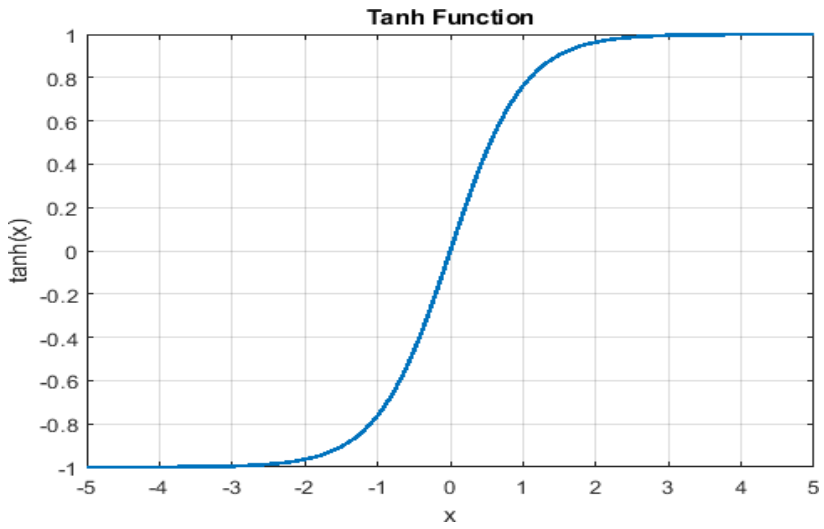


Σχήμα 1.4.ε: Η Σιγμοειδής Συνάρτηση.

- **Η τριγωνομετρική συνάρτηση της υπερβολικής εφαπτομένης (Tanh function)** είναι μια παραλλαγή της σιγμοειδούς συνάρτησης. Χρησιμοποιείται σε κρυφά επίπεδα. Ο μέσος ορός του κρυφού στρώματος ισούται με το μηδέν ή είναι πολύ κοντά στο μηδέν λόγω του ότι η περιοχή εξόδου της συνάρτησης κυμαίνεται στο διάστημα [-1 1]. Η Tanh ειδικεύεται στο κεντράρισμα των δεδομένων, διευκολύνει την εκμάθηση του επόμενου επιπέδου και ισχύει:

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

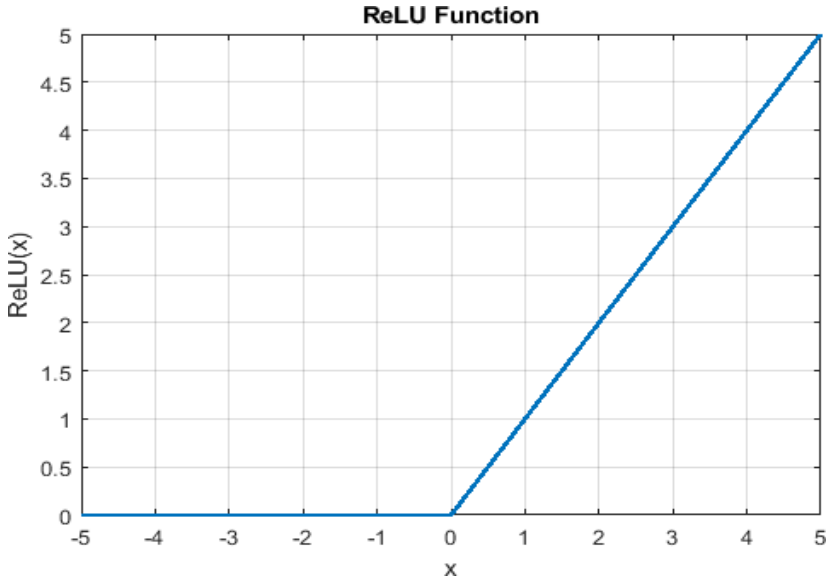
Στο Σχήμα 1.4.στ δίνεται η γραφική παράσταση της συνάρτησής Tanh.



Σχήμα 1.4.στ: Η Συνάρτηση Tanh.

- **Η συνάρτηση ράμπας ReLU (Rectified Linear Unit)** χρησιμοποιείται ιδίως στα συνελικτικά νευρωνικά δίκτυα. Είναι υπολογιστικά αποδοτική και επιταχύνει την εκπαίδευση του δικτύου. Η παράγωγος της ReLU είναι 0 για όλα τα αρνητικά x, ($f'(x) = 0$, if x<0) και στην περίπτωση που η τιμή της μεταβλητής που εκφράζει τον ρυθμό εκμάθησης είναι πολύ υψηλή, τότε κατά τη διαδικασία της οπίσθιας διάδοσης (backpropagation) δεν ενημερώνονται τα βάρη και οι σταθεροί όροι (biases) για ορισμένους νευρώνες δημιουργώντας νεκρούς νευρώνες που δεν ενεργοποιούνται ποτέ. Ο συγκεκριμένος περιορισμός της συνάρτησης είναι γνωστός ως Dying ReLU. Ο μαθηματικός τύπος της συνάρτησης δίνεται παρακάτω:

$$f(x) = \max(0, x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

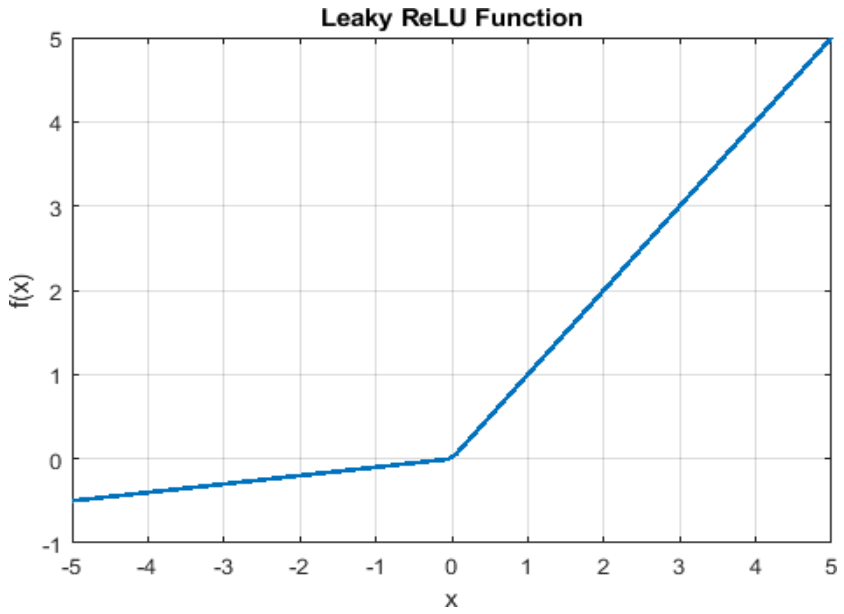


Σχήμα 1.4.ζ: Η Συνάρτηση ράμπας ReLU.

- **Η συνάρτηση Leaky ReLU (Rectified Linear Unit)** είναι μια τροποποιημένη έκδοση της συνάρτησης ReLU, που επιτρέπει μια μη μηδενική κλίση στην αρνητική περιοχή λύνοντας το πρόβλημα Dying ReLU. Ωστόσο καθιστά χρονοβόρα την εκμάθηση των παραμέτρων του μοντέλου. Η συνάρτηση ορίζεται από τον τύπο:

$$f(x) = \begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases}$$

οπού η τιμή της μεταβλητής α είναι μια μικρή θετική σταθερά που καθορίζει την κλίση για τις αρνητικές τιμές.

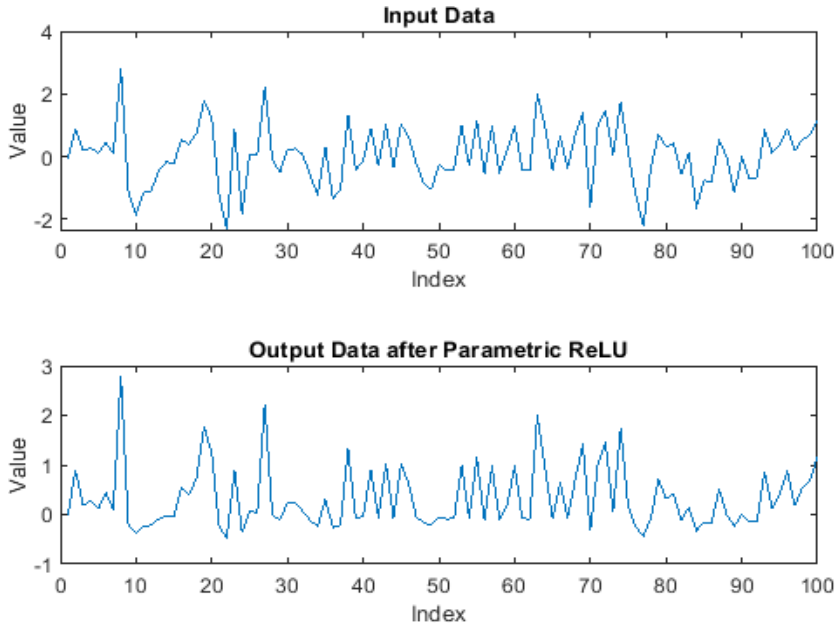


Σχήμα 1.4.η: Η Συνάρτηση Leaky ReLU.

- **Η Παραμετρική συνάρτηση ReLU (Parametric ReLU)** είναι επίσης μια τροποποιημένη έκδοση της συνάρτησης ReLU. Ο στόχος της είναι να λύσει το πρόβλημα του Dying ReLU, όταν η συνάρτηση Leaky ReLU αποτυγχάνει κατά την εκτέλεση του backpropagation η παράμετρος α καθορίζει την κλήση του αρνητικού μέρους της συνάρτησης. Η παράμετρος α μαθαίνεται κατά τη διαδικασία εκπαίδευσης.
  Ο τύπος της Παραμετρικής ReLU είναι ο εξής:

$$f(x) = \max(ax, x)$$

Όπως φαίνεται στο Σχήμα 9 σε ένα δείγμα 100 τυχαίων πραγματικών τιμών εφαρμόζοντας τη συνάρτηση PreLU μειώνεται η επίδραση των αρνητικών τιμών στο δείγμα. Ο τρόπος που η PreLU επηρεάζει το δείγμα εξαρτάται από την κατανομή των τιμών στο δείγμα και την τιμή του παραμετρικού συντελεστή α.



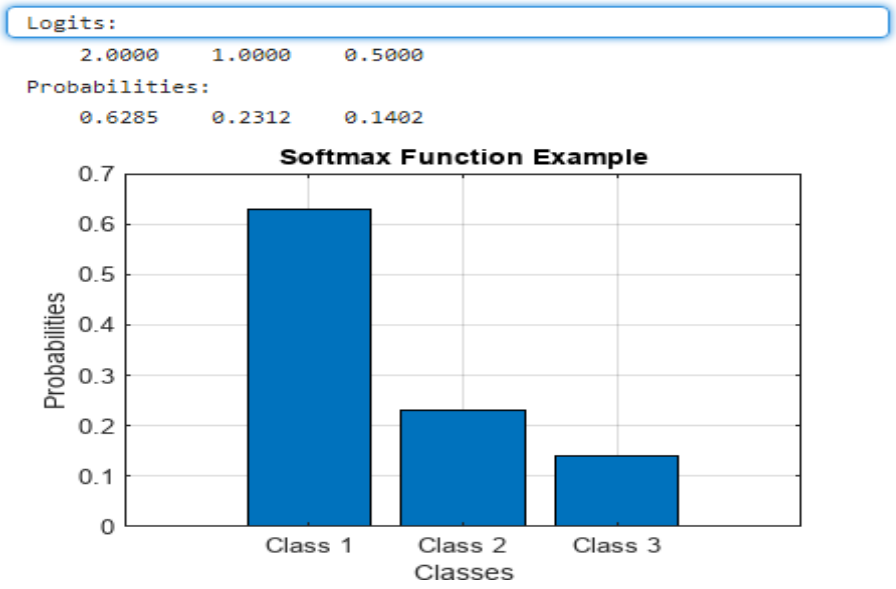
Σχήμα 1.4.θ: Παραμετρική Συνάρτηση ReLU

- **Η συνάρτηση Softmax** χρησιμοποιείται ειδικά στο επίπεδο εξόδου όταν υπάρχουν προβλήματα ταξινόμησης πολλαπλών κλάσεων. Παίρνει ένα διάνυσμα Κ πραγματικών αριθμών (logits) και τα μετατρέπει σε κατανομή πιθανότητας Κ πιθανών αποτελεσμάτων. Η συνάρτηση Softmax δίνεται από τον παρακάτω μαθηματικό τύπο:

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

για $i,j = 1,\ldots, k$ και $x = (x_1, \ldots , x_k) \varepsilon \mathbb{R}^k$

Στο παρακάτω σχήμα παρουσιάζονται τα αποτελέσματα της συνάρτησής Softmax που πήρε στην είσοδο τα Logits και επέστρεψε την κατανομή πιθανότητας.



Σχήμα 1.4.ι.: Η Συνάρτηση Softmax.

Έχουν αναπτυχθεί διάφορες αρχιτεκτονικές τεχνιτών νευρωνικών δικτύων Βαθιάς Εκμάθησης. Μερικές από αυτές είναι οι ακόλουθες:

- Νευρωνικά Δίκτυα Perceptron:
  - Το μονοεπίπεδο νευρωνικό δίκτυο (Single Layer Perceptron)
  - Το πολυεπίπεδο δίκτυο τροφοδοσίας (Multi- Layers Perceptron)

- Τα νευρωνικά δίκτυα συνέλιξης (Convolutional Neural Networks)
- Τα ανατροφοδοτούμενα δίκτυα (Recurrent Neural Networks)
  - Απλά ανατροφοδοτούμενα δίκτυα (Vanilla Recurrent Neural Networks)
  - Δίκτυα μακράς βραχυπρόθεσμης μνήμης (Long-Short Term Memory)
  - Περιφραγμένη Επαναλαμβανομένη μονάδα (Gated Recurrent Unit)
- Παραγωγικό Ανταγωνιστικό Δίκτυο (Generative Adversarial Networks)
- Δίκτυα Βαθιάς Πεποίθησης (Deep Belief Networks)

Τα τεχνητά νευρωνικά δίκτυα δεν προσεγγίζουν την πολυπλοκότητα του εγκέφαλου. Ωστόσο υπάρχουν δυο ομοιότητες μεταξύ του βιολογικού και του τεχνητού δικτύου. Πρώτον τα δομικά στοιχεία τόσο του τεχνητού δικτύου όσο και του βιολογικού λύνουν ιδιαίτερα προβλήματα και δεύτερον οι συνδέσεις μεταξύ των νευρώνων καθορίζουν τη λειτουργία του δικτύου. Αξίζει να σημειωθεί ότι παρόλο που οι βιολογικοί νευρώνες είναι πολύ αργοί σε σύγκριση με τα ηλεκτρικά κυκλώματα (10^{-3} το δευτερόλεπτο σε σχέση με 10^{-10} το δευτερόλεπτο) ο εγκέφαλος μπορεί να εκτελέσει πολλές εργασίες πολύ πιο γρήγορα από οποιονδήποτε συμβατικό υπολογιστή. Αυτό οφείλεται εν μέρει στη μαζική παράλληλη δομή των βιολογικών ουδέτερων δικτύων, όπου όλοι οι νευρώνες λειτουργούν ταυτόχρονα, ενώ στα τεχνητά νευρωνικά δίκτυα οι νευρώνες μοιράζονται την παράλληλη δομή.

1.5. Αλγόριθμοι Μάθησης

Οι Αλγόριθμοι Μάθησης των νευρωνικών δικτύων ανήκουν κυρίως στις ακόλουθες κατηγορίες:

- Εποπτευόμενη Μάθηση (Supervised Learning)
- Μη Εποπτευόμενη Μάθηση (Unsupervised Learning)
- Ημι-εποπτευόμενη Μάθηση (Semi-Supervised Learning)

1.5.1. Εποπτευόμενη Μάθηση (Supervised Learning)

Κατά τη διαδικασία της εποπτευόμενης μάθησης το μοντέλο εκπαιδεύεται και μαθαίνει να κάνει προβλέψεις ή ταξινομήσεις με βάση τα παρεχόμενα παραδείγματα ζευγών εισόδου εξόδου. Τα βήματα που ακολουθεί η εποπτευόμενη μάθηση είναι τα ακόλουθα:

- **Συλλογή και επισήμανση δεδομένων**
  Συλλέγονται τα δεδομένα εισόδου και σε αυτά προστίθενται οι αντίστοιχες σωστές ετικέτες εξόδου.
- **Επεξεργασία δεδομένων**
  Το σύνολο των δεδομένων που συλλέγεται ενδέχεται να πρέπει να υποβληθεί σε προεπεξεργασία για να διασφαλιστεί ότι είναι στην κατάλληλη μορφή. Αυτό το βήμα περιλαμβάνει τον καθαρισμό των δεδομένων, το χειρισμό τιμών που λείπουν, την κανονικοποίηση των αριθμητικών χαρακτηριστικών, τη μετατροπή κατηγορικών δεδομένων σε χρησιμοποιήσιμη μορφή.
- **Εξαγωγή δυνατοτήτων**
  Σε αυτό το βήμα επιλέγονται ή μετατρέπονται τα χαρακτηριστικά εκείνα από τη συλλογή των δεδομένων που μπορούν να βελτιώσουν τη διαδικασία μάθησής και την απόδοση του μοντέλου.
- **Επιλογή Μοντέλου**
  Ανάλογα με τη φύση των δεδομένων και το πρόβλημα που καλούμαστε να λύσουμε επιλέγεται η αρχιτεκτονική που είναι κατάλληλη για την εκάστοτε εργασία, είτε πρόκειται για παλινδρόμηση (πρόβλεψη συνεχών τιμών) είτε για ταξινόμηση (ανάθεση ετικετών σε κατηγορίες).
- **Εκπαίδευση μοντέλου**
  Κατά το βήμα της εκπαίδευσης το επιλεγμένο μοντέλο τροφοδοτείται με τα δεδομένα εισόδου μαζί με τις αντίστοιχες ετικέτες εξόδου. Το μοντέλο προσαρμόζει τις εσωτερικές παραμέτρους επαναληπτικά για να ελαχιστοποιήσει τη διαφορά μεταξύ των προβλέψεων και των πραγματικών ετικετών. Αυτό το βήμα περιλαμβάνει έναν αλγόριθμο βελτιστοποίησης που στοχεύει να βρει τις καλύτερες τιμές παραμέτρων για το μοντέλο.
- **Αξιολόγηση**
  Αφού εκπαιδευτεί το μοντέλο αξιολογείται ως προς την απόδοση του χρησιμοποιώντας ένα ξεχωριστό σύνολο δεδομένων επικύρωσης ή δοκιμής.
- **Συντονισμός μοντέλου**
  Εάν η απόδοση του μοντέλου δεν είναι ικανοποιητική χρειάζεται επανεξέταση των βημάτων επιλογής, επεξεργασία χαρακτηριστικών ή να ρυθμιστούν με ακρίβεια οι παράμετροι που ελέγχουν πτυχές της διαδικασίας εκμάθησης.

- **Συμπεράσματα**
  Μετά την εκπαίδευση και την αξιολόγηση, το μοντέλο είναι έτοιμο να χρησιμοποιηθεί για να δώσει λύση σε προβλήματα που περιλαμβάνουν άγνωστα σύνολα δεδομένων, τα οποία δεν σχετίζονται με το σύνολο δεδομένων δοκιμής η επικύρωσης.

Η εποπτευόμενη μάθηση έχει ένα ευρύ φάσμα εφαρμογών, όπως ταξινόμηση εικόνων, ανάλυση συναισθημάτων, ανίχνευση ανεπιθύμητων μηνυμάτων, ιατρική διάγνωση, μετάφραση γλώσσας και πολλά άλλα.

1.5.2. Μη Εποπτευόμενη Μάθηση (Unsupervised Learning)

Η μη εποπτευομένη μάθηση στοχεύει στην ανακάλυψη μοτίβων, δομών και σχέσεων μέσα σε ένα σύνολο δεδομένων, που δεν έχει προκαθορισμένες σωστές απαντήσεις ή ετικέτες. Ο αλγόριθμος προσπαθεί να βρει εγγενείς δομές ή ομαδοποιήσεις μέσα στα δεδομένα. Η μη εποπτευομένη μάθηση ακολουθεί τις παρακάτω φάσεις:

- **Συλλογή δεδομένων**
  Το σύνολο των δεδομένων που συλλέγονται περιέχουν μόνο τα δεδομένα εισόδου. Αυτά τα δεδομένα εισόδου περιλαμβάνουν διάφορους τύπους πληροφοριών, όπως αριθμητικές τιμές, έγγραφα κειμένου, εικόνες κ.λπ.
- **Προεπεξεργασία δεδομένων**
  Κατά τη φάση αυτή τα δεδομένα μπορεί να χρειαστεί να υποβληθούν σε επεξεργασία για το χειρισμό τιμών που λείπουν, να περάσουν από τη διαδικασία της κανονικοποίησης ή της μετατροπής των δεδομένων σε κατάλληλη μορφή για ανάλυση.
- **Εξαγωγή χαρακτηριστικών**
  Ανάλογα με τη φύση των δεδομένων μπορεί να εφαρμοστούν τεχνικές εξαγωγής χαρακτηριστικών ή μείωσης διαστάσεων για την απλοποίηση των δεδομένων διατηρώντας παράλληλα σημαντικές πληροφορίες. Αυτό μπορεί να βοηθήσει στη μείωση του θορύβου και της υπολογιστικής πολυπλοκότητας.
- **Ομαδοποίηση**
  Ένα από τα κυρία καθήκοντα στη μάθηση χωρίς επίβλεψη είναι η ομαδοποίηση. Οι αλγόριθμοι ομαδοποίησης ομαδοποιούν παρόμοια σημεία δεδομένων βάσει ορισμένων μέτρων ομοιότητας. Ο στόχος είναι να εντοπιστούν φυσικές συστάδες μέσα στα δεδομένα χωρίς προηγουμένη γνώση αυτών καθώς και να «καταλάβουν» το τί αντιπροσωπεύουν αυτές οι συστάδες.
- **Μείωση διαστάσεων**
  Περιλαμβάνει τη μείωση των χαρακτηριστικών ή διαστάσεων στο σύνολο των δεδομένων, διατηρώντας ωστόσο τις περισσότερες σχετικές πληροφορίες. Συνήθως γι’ αυτό τον σκοπό χρησιμοποιούνται τεχνικές όπως η Ανάλυση Κύριου Στοιχείου (Principal Component Analysis) και η τ-Κατανεμημένη Ενσωμάτωση Στοχαστικού Γείτονα (t-Distributed Stochastic Neighbor Embedding).
- **Ανίχνευση ανωμαλιών**
  Αφού μάθει τα κανονικά μοτίβα μέσα στα δεδομένα ο αλγόριθμος μπορεί να εντοπίσει περιπτώσεις που αποκλίνουν σημαντικά από τον κανόνα, όπως ανίχνευση ανωμαλιών, σφαλμάτων η ακραίων τιμών.
- **Μοντελοποίηση θεμάτων**

Σε μια συλλογή κειμένου κατά την επεξεργασία φυσικής γλώσσας χρησιμοποιούνται κάποιες τεχνικές μάθησης για την ανακάλυψη θεμάτων χωρίς να απαιτούνται σαφείς ετικέτες θεμάτων. Μια τέτοια τεχνική είναι η Λανθάνουσα Κατανομή Dirichlet (Latent Dirichlet Allocation).

- **Οπτικοποίηση**
  Για την καλύτερη κατανόηση και ερμηνεία των προτύπων που ανακαλύπτονται χρησιμοποιούνται τεχνικές οπτικοποίησης. Αυτές οι τεχνικές βοηθάνε στην απόκτηση γνώσεων που σχετίζονται με την υποκείμενη δομή των δεδομένων.

Η μάθηση χωρίς επίβλεψη εφαρμόζεται και είναι ιδιαίτερα χρήσιμη σε μεγάλα και πολύπλοκα σύνολα δεδομένων, όπου η χειροκίνητη επισήμανση μπορεί να μην είναι πρακτική ή εφικτή. Ορισμένες εφαρμογές περιλαμβάνουν την τμηματοποίηση προϊόντων ή πελατών, συμπίεση εικόνας, των εντοπισμό ανωμαλιών στον κυβερνοχώρο και όχι μόνο.

1.5.3. Ημι-εποπτευόμενη Μάθηση (Semi-Supervised Learning)

Η ημι-εποπτευόμενη Μάθηση είναι μια προσέγγιση που συνδυάζει στοιχεία τόσο της εποπτευόμενης όσο και της μη εποπτευόμενης μάθησης. Στοχεύει στην αξιοποίηση των πληροφοριών που υπάρχουν τόσο σε δεδομένα με ετικέτα όσο και σε δεδομένα χωρίς ετικέτα για να δημιουργήσει πιο ισχυρά και ακριβή μοντέλα. Η λειτουργία της Ημι-εποπτευόμενης Μάθησης περιλαμβάνει τα ακόλουθα:

Μικτά δεδομένα

Το σύνολο των δεδομένων περιλαμβάνει δεδομένα χωρίς ετικέτες εξόδου και δεδομένα με αντίστοιχες ετικέτες εξόδου.

Συνδυασμός δεδομένων

Στο σύνολο των δεδομένων με ετικέτα εξόδου εφαρμόζονται τυπικές τεχνικές εποπτευόμενης μάθησης για την εκπαίδευση του μοντέλου, ενώ στο σύνολο των δεδομένων χωρίς τις ετικέτες εξόδου οι τεχνικές αυτές ενσωματώνονται κατά το βήμα της εκπαίδευσης, βελτιώνοντας την απόδοση του μοντέλου και επιτυγχάνοντας τη διαδικασία της εκπαίδευσης.

Αυτό-εκπαίδευση

Το συγκεκριμένο βήμα ξεκινάει με το να εκπαιδεύεται το μοντέλο πάνω στα δεδομένα με ετικέτα. Στη συνέχεια ενσωματώνονται τα δεδομένα χωρίς ετικέτα και γίνονται προβλέψεις πάνω σε αυτά τα δεδομένα, οι οποίες αντιμετωπίζονται ως ψευδό-ετικέτες. Το μοντέλο ύστερα εκπαιδεύεται εκ νέου χρησιμοποιώντας τα επισημασμένα δεδομένα μαζί με τα ψευδό-επισημασμένα δεδομένα.

Συνεκπαίδευση

Μια άλλη προσέγγιση είναι η συνεκπαίδευση, η οποία περιλαμβάνει την εκπαίδευση πολλαπλών μοντέλων σε διαφορετικά υποσύνολα δεδομένων. Κάθε μοντέλο μαθαίνει από τα δεδομένα με ετικέτα και στη συνέχεια χρησιμοποιεί τις προβλέψεις του για να βελτιώσει τη διαδικασία εκμάθησης των άλλων μοντέλων. Η συγκεκριμένη προσέγγιση λειτουργεί καλά, εφόσον τα δεδομένα μπορούν να χωριστούν σε διακριτά υποσύνολα.

Τεχνικές τακτοποίησης

Η ημι-εποπτευόμενη Μάθηση σε κάποιες περιπτώσεις ενσωματώνει τεχνικές τακτοποίησης, που βοηθάνε το μοντέλο να γενικεύει καλυτέρα και να παράγει συνεπείς προβλέψεις τόσο σε επισημασμένα όσο και σε μη επισημασμένα δεδομένα.

Η ημι-εποπτευόμενη Μάθηση εφαρμόζεται σε εφαρμογές που περιλαμβάνουν την αναγνώριση ομιλίας, την επεξεργασία φυσικής γλώσσας, την ανάλυση ιατρικής εικόνας και όχι μόνο.

1.6. Τεχνικές Βελτιστοποίησης εκπαίδευσης Νευρωνικών Δικτύων

Τα Νευρωνικά δίκτυα χρησιμοποιούν διάφορους αλγορίθμους βελτιστοποίησης για να εκπαιδεύσουν και να ενημερώσουν τις παραμέτρους τους κατά τη διαδικασία της εκμάθησης. Ο στόχος των αλγορίθμων βελτιστοποίησης είναι η ελαχιστοποίηση της συνάρτησης σφάλματος / απώλειας. Προσαρμόζοντας τις παραμέτρους του δικτύου τα βάρη και τους σταθερούς όρους, η συνάρτηση μετρά πόσο απέχει η προβλεπόμενη έξοδος από την πραγματική έξοδο στόχο του νευρωνικού δικτύου. Η επιλογή της συνάρτησης εξαρτάται από τη φύση του προβλήματος αλλά και από τη λειτουργία ενεργοποίησης που χρησιμοποιείται στο επίπεδο εισόδου. Μετά από την εκπαίδευση του δικτύου, χρησιμοποιώντας ένα ξεχωριστό σύνολο επικύρωσης ή σύνολο δοκιμής το μοντέλο αξιολογείται ως προς την απόδοσή του. Η τιμή της συνάρτησης απώλειας παίζει καθοριστικό ρόλο καθώς βοηθά στη μέτρηση του ποσό καλά το μοντέλο γενικεύεται σε αόρατα δεδομένα. Οι πιο γνωστές συναρτήσεις κόστους / απώλειας είναι το Μέσο Τετραγωνικό Σφάλμα (MSE), η Δυαδική Απώλεια Διασταυρούμενης Εντροπίας, η Κατηγορική Απώλεια Διασταυρούμενης Εντροπίας, η Απόκλιση Kullback-Leibler, η Απώλεια Huber και προσαρμοσμένες συναρτήσεις απώλειας για την αντιμετώπιση μοναδικών απαιτήσεων.

Οι πιο συχνά χρησιμοποιούμενοι αλγόριθμοι βελτιστοποίησης, που εφαρμόζονται κατά την εκπαίδευση των νευρωνικών δικτυών είναι οι ακόλουθοι:

Gradient Descent (GD)

Είναι ένας από τους βασικούς αλγόριθμους βελτιστοποίησης, που χρησιμοποιείται στην εκπαίδευση των νευρωνικών μοντέλων αλλά και σε διαφορά προβλήματα βελτιστοποίησης και στοχεύει στην εύρεση της ελάχιστης τιμής της συνάρτησης απώλειας / κόστους.

Ο αλγόριθμος GD ακολουθεί τα παρακάτω βήματα:

1. Τυχαία αρχικοποίηση των παραμέτρων (βάρη και προκαταλήψεις) του μοντέλου.
2. Υπολογισμός της συνάρτησης κόστους (loss function), η οποία ποσοτικοποιεί πόσο απέχουν οι προβλεπόμενες τιμές του μοντέλου από τις πραγματικές τιμές, χρησιμοποιώντας τα δεδομένα εκπαίδευσης και τις τρέχουσες παραμέτρους.
3. Υπολογισμός της διαβάθμισης (gradient) της συνάρτησης κόστους σε σχέση με κάθε παράμετρο του νευρωνικού δικτύου, χρησιμοποιώντας τεχνικές όπως π.χ. η οπισθοδρόμηση (backpropagation).
4. Ενημέρωση των παραμέτρων. Στο συγκεκριμένο βήμα προσαρμόζονται οι παράμετροι αφαιρώντας ένα κλάσμα της διαβάθμισης, που κλιμακώνεται με

τον ρυθμό εκμάθησης από τις τρέχουσες τιμές των παραμέτρων. Η εξίσωση δίνεται παρακάτω:

New_Parameter = Old_Parameter - (Learning_Rate * Gradient)

Όπου ο όρος "Gradient" αναφέρεται στη μερική παράγωγο της συνάρτησης απώλειας σε σχέση με την παράμετρο που ενημερώνεται.

5. Σε αυτό το βήμα ελέγχεται αν πληρούται το κριτήριο σύγκλησης, το οποίο μπορεί να είναι η επίτευξη ενός συγκεκριμένου αριθμού επαναλήψεων (εποχές), ή ένα όριο για το μέγεθος της κλίσης.
6. Τα βήματα 2-5 επαναλαμβάνονται έως ότου ικανοποιηθεί το κριτήριο διακοπής του μοντέλου.

Τύποι αλγορίθμων Gradient Descent (GD):

Batch Gradient Descent (BGD) - ενημερώνει τις παραμέτρους χρησιμοποιώντας τη μέση τιμή της διαβάθμισης (average gradient) του συνόλου των δεδομένων εκπαίδευσης.

New_Parameters = Old_Parameters - (Learning_Rate * Average_Gradient)

Όπου:

Old_Parameters - τρέχουσες τιμές των παραμέτρων

Learning_Rate - ο ρυθμός εκμάθησης

Average_Gradient - ο μέσος όρος των μερικών παραγώγων, που υπολογίζεται για μια παρτίδα εκπαίδευσης.

Stochastic Gradient Descent (SGD) - ενημερώνει τις παραμέτρους χρησιμοποιώντας τη διαβάθμιση της απώλειας ενός μόνο παραδείγματος προπόνησης κάθε φορά. Για ένα παράδειγμα εκπαίδευσης ο τρόπος που ενημερώνονται οι παράμετροι δίνεται από τον παρακάτω μαθηματικό τύπο:

New_Parameters = Old_Parameters - (Learning_Rate * Gradient)

Όπου:

Old_Parameters - είναι οι τρέχουσες τιμές των παραμέτρων προς ενημέρωση

New_Parameters - είναι οι ενημερωμένοι παράμετροι ενός εκπαιδευτικού κύκλου

Learning_Rate - είναι ο ρυθμός εκμάθησης

Gradient - είναι οι μερικοί παράγωγοι της συνάρτησης κόστους/απώλειας σε σχέση με τις παραμέτρους που ενημερώνονται. Υπολογίζεται χρησιμοποιώντας ολόκληρο το σύνολο δεδομένων.

Mini-batch Gradient Descent - ενημερώνει τις παραμέτρους χρησιμοποιώντας μικρά σύνολα (μικρές παρτίδες) των δεδομένων εκπαίδευσης.

Momentum

Αποτελεί μια βελτιωμένη εκδοχή του GD, που βοηθάει στην επιτάχυνση της σύγκλισης, συσσωρεύοντας έναν κινητό μέσο ορό (running average) παλαιοτέρων κλίσεων, μειώνοντας τις ταλαντώσεις (oscillations) και τις υπερβάσεις (overshooting) στις ενημερώσεις των παραμέτρων. Δίνεται από το παρακάτω τύπο:

$$V=\beta * v + (Learning\_Rate * Gradient)$$

Όπου

v - είναι ο όρος ταχύτητας/όρος momentum (velocity/ momentum term), που λειτουργεί ως ένας κινητός μέσος όρος των παλαιοτέρων κλίσεων της συνάρτησης κόστους/απώλειας.

β - συντελεστής ταχύτητας και συνήθως παίρνει τιμές μεταξύ 0 και 1.

Learning_Rate - ο ρυθμός εκμάθησης.

Gradient **-** είναι οι μερικοί παράγωγοι της συνάρτησης κόστους/απώλειας.

Nesterov accelerated Gradient (NAG)

Είναι μια βελτιωμένη μέθοδος του αλγόριθμου Momentum, που επιτρέπει την πιο ακριβή εκτιμήση των συσσωρευμένων προηγουμένων διαβαθμίσεων, χρησιμοποιώντας τον όρο ταχύτητας/momentum, προσαρμόζοντας ανάλογα τον υπολογισμό της διαβάθμισης της συνάρτησης κόστους/απώλειας σε σχέση με τις παραμέτρους που αξιολογούνται στην κατά προσέγγιση μελλοντική τους θέση και δίνεται από τον ακόλουθο τύπο:

$$V = \beta * v - Learning\_Rate * Gradient(Old\_Parameters + b * v)$$

Όπου:

V - είναι ο όρος ταχύτητας/όρος momentum (velocity/ momentum term)

B - ο συντελεστής ταχύτητας και συνηθώς παίρνει τιμές μεταξύ 0 και 1

Learning_Rate - ο ρυθμός εκμάθησης

Gradient (Old_Parameters+b*v) - η διαβάθμιση της συνάρτησης κόστους/απώλειας σε σχέση με τις παραμέτρους που αξιολογούνται στην κατά προσέγγιση μελλοντική τους θέση.

Adagrad

Ο συγκεκριμένος αλγόριθμος προσαρμόζει τον ρυθμό εκμάθησής κάθε παραμέτρου με βάση ένα ιστορικό, που περιλαμβάνει παλιές κλίσεις, συσσωρεύοντας τις τετραγωνικές διαβαθμίσεις και διατηρώντας το ρυθμό εκμάθησης με την τετραγωνική ρίζα αυτής της συσσώρευσης. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη σε αραιά δεδομένα και δίνεται από τον τύπο:

$$New\_Parameters = Old\_Parameters - \frac{Learning\_Rate}{\sqrt{Cache + e}} \times Gradient$$

Όπου

Old_Parameters - είναι οι τρέχουσες τιμές των παραμέτρων προς ενημέρωση

New_Parameters - είναι οι ενημερωμένες παράμετροι ενός εκπαιδευτικού κύκλου

Learning_Rate - είναι ο ρυθμός εκμάθησης

Gradient - είναι οι μερικοί παράγωγοι της συνάρτησης κόστους/απώλειας

e- μικρή σταθερά για αριθμητική σταθερότητα.

Cache - είναι η προσωρινή μνήμη, που συσσωρεύει τις τετραγωνικές διαβαθμίσεις της συνάρτησης κόστους/απώλειας και υπολογίζεται με τη βοήθεια της παρακάτω εξίσωσης.

$$\text{Cache} = \text{Cache} + \text{Gradient}^2$$

Root Mean Square Propagation (RMSprop)

Είναι επέκταση του Adagrad. Διατηρεί ένα κινητό μέσο όρο τετραγωνικών κλίσεων αλλά με συντελεστή αποσύνθεσης που βοηθά στο να αποτραπεί το να γίνει πολύ μικρό το ποσοστό εκμάθησης πολύ γρήγορα. Δίνεται από το παρακάτω τύπο:

$$\text{New\_Parameters} = \text{Old\_Parameters} - \frac{\text{Learning\_Rate}}{\sqrt{\text{Cache} + \text{e}}} \times \text{Gradient}$$

Όπου

Old_Parameters - είναι οι τρέχουσες τιμές των παραμέτρων προς ενημέρωση

New_Parameters - είναι οι ενημερωμένες παράμετροι ενός εκπαιδευτικού κύκλου

Learning_Rate - είναι ο ρυθμός εκμάθησης

Gradient - είναι οι μερικοί παράγωγοι της συναρτήσεις κόστους/απώλειας

e - μικρή σταθερά

Chache - είναι ο κινητός μέσος όρος των τετραγωνικών διαβαθμίσεων της συνάρτησης κόστους/απώλειας και υπολογίζεται με τον ακόλουθο τρόπο:

$$\text{Cache} = \beta \times \text{Cache} + (1 - \beta) \times \text{Gradient}^2$$

όπου το β είναι το ποσοστό αποσύνθεσης

Adaptive Moment Estimation (Adam)

Ένας συνδυασμός μεταξύ του Momentum και του RMSprop. Διατηρεί δύο κινητούς μέσους όρους, έναν παλιότερων κλίσεων και έναν τετραγωνικών κλίσεων με τον συντελεστή αποσύνθεσης (decay factor) ενημερωμένο για κάθε παράμετρο και προσαρμόζει ανάλογα τους ρυθμούς εκμάθησης. Η ενημέρωση των παραμέτρων γίνεται με τον ακόλουθο τρόπο:

$$\text{New\_Parameters} = \text{Old\_Parameters} - \frac{\text{Learning\_Rate}}{\sqrt{\hat{\text{v}} + \text{e}}} \times \hat{\text{m}}$$

Όπου

m̂– είναι η εκτιμήση της πρώτης στιγμής με ενημερωμένη μεροληψία δηλ. ένας τρέχων μέσος όρος των κλίσεων της συνάρτησης κόστους/απώλειας.

v̂ - είναι η εκτιμήση της δεύτερης στιγμής με ενημερωμένη μεροληψία δηλ. ένας τρέχων μέσος όρος των κλίσεων της συνάρτησης κόστους/απώλειας.

Learning_Rate - ρυθμός εκμάθησης.

e - μικρή σταθερά για αριθμητική σταθερότητα.

Old_Parameters - είναι οι τρέχουσες τιμές των παραμέτρων προς ενημέρωση.

Adadelta

Αποτελεί παραλλαγή του Adagrad. Προσαρμόζει δυναμικά τους ρυθμούς εκμάθησης με βάση έναν τρέχοντα μέσο όρο των ενημερώσεων των τετραγωνικών παραμέτρων και δίνεται από τον ακόλουθο τύπο:

$$\text{New\_Parameters} = \text{Old\_Parameters} + \text{Update}$$

Όπου

Old_Parameters - είναι οι τρέχουσες τιμές των παραμέτρων προς ενημέρωση

New_Parameters - είναι οι ενημερωμένες παράμετροι

Update - συσσωρευμένες τετραγωνικές ενημερώσεις των παραμέτρων και υπολογίζονται με τον ακόλουθο τρόπο:

$$\text{Update} = -\frac{\sqrt{\text{Cache\_Update} + \text{e}}}{\sqrt{\text{Cache\_Grad} + \text{e}}} \times \text{Gradient}$$

Όπου

$\text{Chache\_Update} = \text{p} \times \text{Chache}_{\text{Grad}} + (1 - \text{p}) \times \text{Gradient}^2$- περιλαμβάνει τις τιμές των τρεχουσών παραμέτρων.

Chache_Grad - συσσωρευμένες τετραγωνικές διαβαθμίσεις

P - το ποσοστό αποσύνθεσης

Gradient - Μερικοί παράγωγοι

E - μικρή σταθερά

Nadam

Είναι μια επέκταση του NAG, που συνδυάζει την ορμή του Nesterov με προσαρμοστικούς ρυθμούς εκμάθησης τύπου Adam. Η βασική διαφορά από το NAG έγκειται στον τρόπο με τον οποίο χρησιμοποιείται η κλίση της συνάρτησης απώλειας/κόστους όχι μόνο στην τρέχουσα θέση άλλα και σε μια πιθανή μελλοντική θέση των παραμέτρων. Δίνεται από τον τύπο:

$$\text{New\_Parameters} = \text{Old\_Parameters} - \frac{\text{Learning\_Rate}}{\sqrt{\hat{\text{v}} + \text{e}}} \times \hat{\text{m}}$$

Όπου

New_Parameters - είναι οι ενημερωμένες παράμετροι

Old_Parameters **-** είναι οι τρέχουσες τιμές των παραμέτρων προς ενημέρωση

Learning_Rate - ρυθμός εκμάθησης.

e - μικρή σταθερά

$\hat{m}$– η εκτιμήση των κλίσεων της συνάρτησης κόστους/απώλειας στην τρέχουσα θέση.

$\hat{v}$ - είναι η εκτιμήση των κλίσεων της συνάρτησης κόστους/απώλειας σε μια μελλοντική θέση.

Limited-memory Broyden-Fletcer-Goldfarb-Shanno (L-BFGS)

Είναι ένας αλγόριθμος βελτιστοποίησης βασισμένος στη μέθοδο quasi-Newton, που προσεγγίζει τον πίνακα Hessian (παραγωγό δεύτερης τάξης της συνάρτησης απώλειας), χρησιμοποιώντας περιορισμένη μνήμη, κατάλληλος για σύνολα δεδομένων μικρού η μεσαίου μεγέθους.

1.7. Νευρώνες Perceptron

1.7.1. Μονοεπίπεδο Νευρωνικό Δίκτυο (Single layer Perceptron - SLP)

Το μονοεπίπεδο νευρωνικό δίκτυο Perceptron εφευρέθηκε το 1957 από τον Frank Rosenblatt. Αποτελεί τροποποιημένη έκδοση του νευρώνα McCulloch and Pitts (MP Neuron), που είναι σχετικά το πιο απλό δίκτυο, έχει μόνο ένα στρώμα υπολογιστικών μονάδων (κρυφό στρώμα), είναι κατάλληλο για δυαδική ταξινόμηση (binary classification) ακόμα και όταν τα δεδομένα είναι γραμμικά διαχωρισμένα (linearly separable).

1.7.2. Ο αλγόριθμος πίσω από τη λειτουργία του μονοεπίπεδου νευρωνικού δικτύου Perceptron

Ο κανόνας εκμάθησης της μονής στρώσης Perceptron, που συχνά αποδίδεται στον Frank Rosenblatt, προσαρμόζει τα βάρη w με βάση το σφάλμα μεταξύ της προβλεπόμενης και της επιθυμητής εξόδου, στοχεύοντας στην ελαχιστοποίηση του σφάλματος ταξινόμησης πάνω σε ένα σύνολο δεδομένων εκπαίδευσης.

Τα βήματα που ακολουθεί ο Αλγόριθμος μονής στρώσης Perceptron είναι τα ακόλουθα:

1. **Αρχικοποίησή των βαρών w και της προκατάληψης b (bias)**

Στο μονοεπίπεδο Perceptron τα βάρη $w=w_1,w_2,\ldots..,w_n$ εκχωρούνται τυχαία σε κάθε είσοδο $x=x_1$, x_2 , … , x_n καθώς δεν υπάρχει εκ των προτέρων γνώση που να σχετίζεται με τις εισόδους. Τυχαία εκχωρείται και η τιμή της προκατάληψης (bias).

2. **Υπολογισμός ανατροφοδότησης**

Το μονοεπίπεδο Perceptron αθροίζει όλες τις σταθμισμένες εισόδους βάζοντας στο τέλος και την προκατάληψη (bias). Αν το άθροισμα είναι πάνω από την προκαθορισμένη τιμή που παράγει η συνάρτηση κατωφλίου $f_p(x)$ τότε το δίκτυο θα ενεργοποιηθεί από τη συνάρτηση μεταφοράς $g(f_p)$.

$$f_p(x) = \sum_{i=1}^{n} (x_i \ w_i \ ) + b \quad (\alpha)$$

$$y(x) = g(f_p) \quad (\beta)$$

3. Ενημέρωση βαρών

Κατά το βήμα της ενημέρωσης συγκρίνονται η τρέχουσα τιμή της εξόδου με την τιμή εξόδου που πρέπει να πετύχει για να θεωρηθεί ότι το μοντέλο Perceptron είναι κατάλληλο και αποδοτικό. Αν η τρέχουσα τιμή της εξόδου Y δεν είναι η επιθυμητή τιμή, τότε, επειδή δεν υπάρχει τεχνική οπισθοδιάδοσης, θα πρέπει να ενημερώνονται τα βάρη w για να μειωθεί το σφάλμα του μοντέλου .

Το τρέχον σφάλμα του μοντέλου δίνεται από τον τύπο

$$\text{Error} = y - y_{\text{actual}}$$

Οπού y η επιθυμητή έξοδος

Και y_{actual} η τρέχουσα έξοδος

Η ενημέρωση των βαρών w δίνεται από τον τύπο:

$w_{new} = w + a \times \text{Error} \times x$

όπου w_{new} είναι το διάνυσμα με τα ενημερωμένα βάρη και w το διάνυσμα με τα αρχικά βάρη, η σταθερά a είναι ο ρυθμός εκμάθησης, που κυμαίνεται στο διάστημα $0< \boldsymbol{\alpha} <1$ και ελέγχει το ποσό γρήγορα αλλάζει η διαδικασία ενημέρωσης ως απόκριση σε νέα.

Ο αλγόριθμος συνεχίζει να ενημερώνει τα βάρη έως ότου ελαχιστοποιηθεί το σφάλμα ταξινόμησης στα δεδομένα εκπαίδευσης ή έως ότου επιτευχθεί ένας προκαθορισμένος αριθμός επαναλήψεως.

4. Επανάλαβε τον Υπολογισμό ανατροφοδότησης

Αφού ενημερώθηκαν τα βάρη ο αλγόριθμος επιστρέφει και εκτελεί εκ νέου το δεύτερο βήμα (Υπολογισμός ανατροφοδότησης) υπολογίζοντας εκ νέου τη νέα σταθμισμένη είσοδο αθροίζοντας το bias και ελέγχει εκ νέου αν η τιμή της εξόδου Y είναι επιθυμητή.

5. Σύγκλιση και όριο απόφασης

Εάν τα δεδομένα εκπαίδευσης είναι γραμμικά διαχωρισμένα, ο αλγόριθμος θα συγκλίνει τελικά σε ένα σύνολο βαρών, που ταξινομούν σωστά τα δείγματα εκπαίδευσης. Τα βάρη και οι σταθεροί όροι ορίζουν ένα όριο απόφασης που χωρίζει τις δυο κατηγορίες στο χώρο εισόδου.

1.7.3. Η συνάρτηση ενεργοποίησης του SLP

Η συνάρτηση ενεργοποίησης δεν είναι κάτι άλλο πάρα μονάδα λήψης αποφάσεων νευρωνικών δικτύων, που τροφοδοτούν την έξοδο ενός νευρωνικού κόμβου. Η λειτουργία ενεργοποίησης ενός νευρώνα υπαγορεύει αν ο νευρώνας πρέπει να ενεργοποιηθεί η να τεθεί σε αδράνεια. Στα μονής στρώσης Perceptron εφαρμοζόταν η βηματική συνάρτηση (Heaviside), η οποία παράγει μια δυαδική τιμή με βάση το αν το σταθμισμένο άθροισμα υπερβαίνει ένα συγκεκριμένο όριο. Δίνεται από τον ακόλουθο μαθηματικό τύπο:

$$g(f_p) = \begin{cases} 1 & f_p \geq 0 \\ 0 & f_p < 0 \end{cases}$$

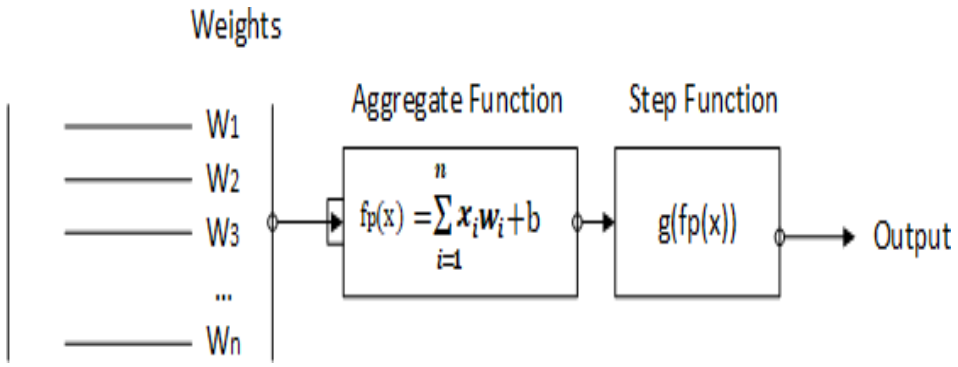
Όπου το fp είναι το σταθμισμένο άθροισμα, στο οποίο προστίθεται η νευρωνική πόλωση b (bias) και δίνεται από τη μαθηματική έκφραση:

$$f_p(x) = \sum_{i=1}^{n} (x_i \ \ w_i \ \ ) + b$$

1.7.4. Περιορισμοί και εφαρμογές του μονοεπίπεδου Perceptron

Το Perceptron μονής στρώσης μπορεί να μοντελοποιήσει μόνο γραμμικά διαχωρίσιμα δεδομένα. Αυτό σημαίνει ότι μπορεί να ταξινομήσει με επιτυχία σημεία δεδομένων που μπορούν να διαχωριστούν με μια ευθεία γραμμή. Δεν μπορεί να εκπαιδευτεί σε δεδομένα που είναι διαχωρισμένα με κυκλικό όριο. Είναι ευαίσθητο στο θόρυβο δεδομένων και δύσκολα εκπαιδεύεται σε δεδομένα με πολλές παραμέτρους.

Στο **Σχήμα 1.7.4.α** απεικονίζεται η λειτουργία της μονής στρώσης Perceptron

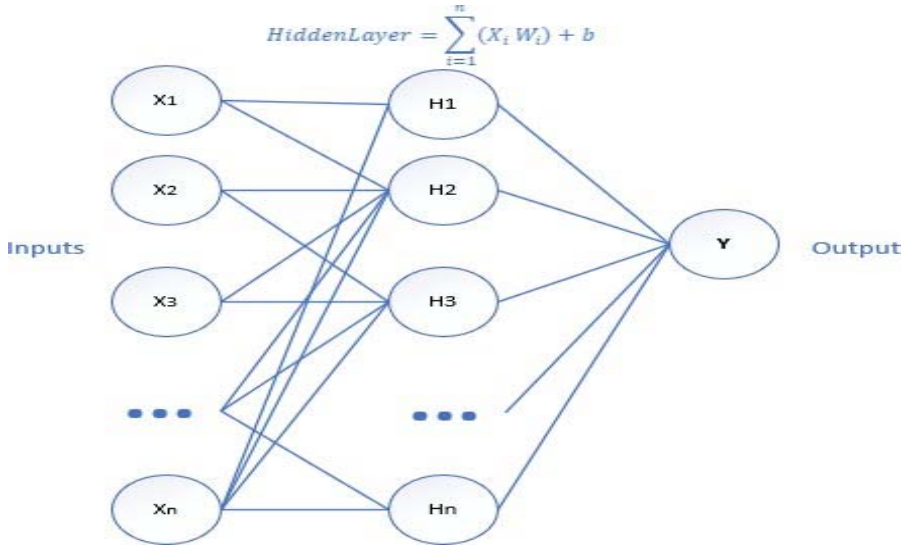


Σχήμα 1.7.4.α: Μονάδα Λειτουργίας Μονής Στρώσης Perceptron.

1.7.5. Τα πολυεπίπεδα δίκτυα τροφοδοσίας (Multi- Layers Perceptrons - MLP)

Τα δίκτυα Perceptron πολλαπλών στρώσεων αποτελούν προηγμένες αρχιτεκτονικές νευρωνικών δικτύων και αποτελούνται από πολλαπλά επίπεδα κρυφών νευρώνων, ένα επίπεδο εισόδου και ένα στρώμα εξόδου. Τα πολλαπλά κρυφά επίπεδα επιτρέπουν στο πολυεπίπεδο Perceptron να συλλαμβάνει πολύπλοκα μοτίβα και σχέσεις πάνω στα δεδομένα εισόδου, καθιστώντας το κατάλληλο για την εκτέλεση ενός ευρέος φάσματος εργασιών, και την επίλυση τόσο γραμμικά διαχωρίσιμων όσο και μη γραμμικά διαχωρίσιμων προβλημάτων.

Τα πολυεπίπεδα Perceptron χρησιμοποιούν αλγορίθμους βελτιστοποίησης, που βασίζονται σε κλίση κατάβασης (gradient descent) και τον αλγόριθμο backpropagation, που είναι βασικό συστατικό στην εκπαίδευση των MLP. Στο **Σχήμα 1.7.5.α** απεικονίζεται ένα Πολυεπίπεδο δίκτυο Perceptron με ένα κρυφό στρώμα.



Σχήμα 1.7.5.α: Πολυεπίπεδο δίκτυο Perceptron με ένα κρυφό στρώμα.

Ο μαθηματικός τύπος ενός μεμονωμένου νευρώνα του πολυεπιπέδου Perceptron

Έστω ότι εξετάζουμε ένα νευρώνα στο στρώμα (j), που συνδέεται με νευρώνες στο προηγούμενο στρώμα (j-1) και x_i είναι το διάνυσμα εισόδου στο νευρώνα που βρίσκεται στο στρώμα j, w_i είναι το βάρος που σχετίζεται με αυτόν το νευρώνα και b_i είναι η μεροληψία (bias) γι αυτόν το νευρώνα, τότε το σταθμισμένο άθροισμα των εισροών δίνεται από την ακόλουθη μαθηματική έκφραση:

$z^{(j)} = \sum_{i=1}^{N^{(j-1)}} (x_i^{(j-1)} w_i^{(j)}) + b_i^{(j)}$ όπου $N^{(j-1)}$ είναι ο αριθμός των νευρώνων στο προηγούμενο στρώμα.

Η έξοδος $y^{(j)}$ του νευρώνα που βρίσκεται στο στρώμα j δίνεται από τον εξής τύπο:

$y^{(j)} = g^{(j)}(z^{(j)})$ όπου το $g^{(j)}$ είναι η συνάρτηση ενεργοποίησης του συγκεκριμένου νευρώνα του στρώματος j.

Αυτή η διαδικασία επαναλαμβάνεται για κάθε νευρώνα του στρώματος j και η έξοδος του στρώματος j ($y^{(j)}$) γίνεται η είσοδος στο επόμενο στρώμα j+1,j+2….. έως ότου το δίκτυο συγκλίνει και πέτυχει το επιθυμητό επίπεδο ακρίβειας.

1.7.6. Αλγόριθμος εκμάθησης

Ο αλγόριθμος εκπαίδευσης των MLP ακολουθεί τα παρακάτω βήματα:

1. **Forward Pass**

   Τα δεδομένα εισόδου τροφοδοτούνται μέσω των επίπεδων του δικτύου, κάθε νευρώνας υπολογίζει το σταθμισμένο άθροισμα προσθέτοντας την νευρωνική πόλωση. Πάνω στο συγκεκριμένο αποτέλεσμα εφαρμόζεται μια συνάρτηση ενεργοποίησης και το αποτέλεσμα της συνάρτησης τροφοδοτείται στην είσοδο του κρυφού στρώματος. Η ίδια διαδικασία επαναλαμβάνεται έως ότου δεν υπάρχουν άλλα κρυφά στρώματα και η έξοδος του κρυφού στρώματος τροφοδοτείται στην είσοδο του επιπέδου εξόδου.

2. **Υπολογισμός απώλειας**
   Ο υπολογισμός της απώλειας του δικτύου δηλ. η διαφορά της προβλεπόμενης εξόδου και της πραγματικής εξόδου γίνεται χρησιμοποιώντας μια συνάρτηση απώλειας, η οποία μετρά πόσο απέχουν οι προβλέψεις του δικτύου από τους πραγματικούς στόχους. Συνήθως στα προβλήματα παλινδρόμησης χρησιμοποιείται το μέσο τετραγωνικό σφάλμα και στα προβλήματα ταξινόμησης η διασταυρούμενη εντροπία.

3. **Backpropagation**
   Για να ελαχιστοποιηθεί η απώλεια του μοντέλου η διαδικασία οπίσθιας διάδοσης ακολουθεί τα παρακάτω βήματα:

   - **Υπολογισμός κλίσης της συνάρτησης απώλειας**

     Υπολογίζεται η διαβάθμιση της συνάρτησης απώλειας σε σχέση με τις ενεργοποιήσεις του επιπέδου εξόδου, δηλ. οι κλήσεις της συνάρτησης σε σχέση με τα βάρη. Αυτές οι κλήσεις δείχνουν πόσο συνέβαλε κάθε κόμβος εξόδου στο συνολικό σφάλμα.

   - **Πίσω διάδοση των διαβαθμίσεων**
     Η διαβάθμιση πηγαίνει προς τα πίσω μέσα από τα επίπεδα του δικτύου και για κάθε επίπεδο υπολογίζονται οι διαβαθμίσεις της απώλειας σε σχέση με το σταθμισμένο άθροισμα των εισροών του επιπέδου.

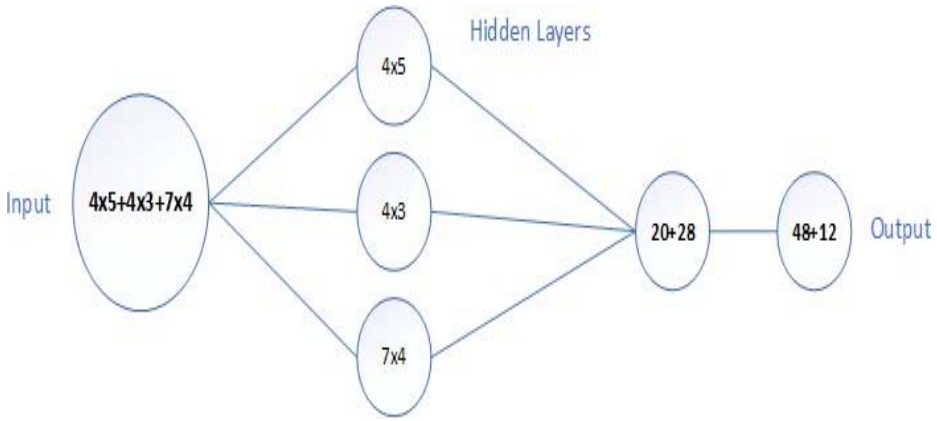
   - **Κανόνας της Αλυσίδας**
     Οι διαβαθμίσεις υπολογίζονται χρησιμοποιώντας τον κανόνα της αλυσίδας. Πολλαπλασιάζεται η διαβάθμιση απώλειας σε σχέση με την έξοδο του νευρώνα με τη διαβάθμιση της εξόδου του νευρώνα σε σχέση με το σταθμισμένο άθροισμα.

4. **Ενημέρωση βαρών**
   Τα βάρη του δικτύου ενημερώνονται χρησιμοποιώντας τον αλγόριθμο βελτιστοποίησης κατάβασης κλίσης (gradient descent), που ωθεί τα βάρη προς την κατεύθυνση που μειώνει την απώλεια.

5. **Πολλαπλή επανάληψη**
   Όλα τα παραπάνω βήματα εφαρμόζονται και επαναλαμβάνονται πολλές φορές πάνω στο σύνολο των δεδομένων εκπαίδευσης. Ο στόχος της επαναληπτικής διαδικασίας είναι η βελτίωση της απόδοσης του δικτύου προσαρμόζοντας τα βάρη για να ελαχιστοποιηθεί η απώλεια. Η συγκεκριμένη διαδικασία περιλαμβάνει τη λεπτομερή ρύθμιση των παραμέτρων του μοντέλου ώστε να συγκλίνει σε μια κατάσταση οπού το δίκτυο είναι αποδοτικό σε αόρατα δεδομένα.

Τα MLP δίκτυα έχουν σχεδιαστεί να λύσουν περίπλοκα προβλήματα μέσω τις συνεργασίας των πολλαπλών στρωμάτων διασυνδεδεμένων νευρώνων, οπού κάθε στρώμα καλείται να λύσει ένα συγκεκριμένο μέρος του προβλήματος. Μια

αναπαράσταση του τρόπου λειτουργίας των MLP φαίνεται στο **Σχήμα 1.7.6.α**, όπου στην είσοδο του μοντέλου δίνεται μια μαθηματική έκφραση που στη συνέχεια διασπάται σε μικρά προβλήματα (επιμέρους υπολογισμούς της αρχικής μαθηματικής έκφρασης).



Σχήμα 1.7.6.α: Λειτουργία υπολογισμού μιας μαθηματικής έκφρασης του πολυεπιπέδου Perceptron(MLP).

1.7.7. Η συνάρτηση ενεργοποίησης

Τα MLP (Multi-Layer Perceptrons) χρησιμοποιούν διάφορες συναρτήσεις ενεργοποίησης, όπου κάθε λειτουργία ενεργοποίησης έχει τα χαρακτηριστικά της και τις περιπτώσεις χρήσης. Στα MLP εφαρμόζονται συχνά οι συναρτήσεις σιγμοειδούς, Tanh, ReLU, Leaky ReLU, Παραμετρική ReLU, Εκθετική Γραμμική Μονάδα (ELU), Swish , GLU (Gated Linear Unit).

1.7.8. Περιορισμοί των δικτύων Perceptrons πολλών στρώσεων

Τα Perceptrons πολλών στρώσεων είναι ένα θεμελιώδες μοντέλο και χρησιμοποιούνται σε διάφορες εφαρμογές. Ωστόσο παρουσιάζουν αρκετούς περιορισμούς, δεν μπορούν να χειριστούν φυσικά διαδοχικά δεδομένα όπως είναι οι χρονοσειρές ή κείμενα, και δεν έχουν την ικανότητα να αποτυπώνουν.

ΚΕΦΑΛΑΙΟ 2

2. ΒΑΘΙΑ ΜΑΘΗΣΗ

2.1. Ορισμός της Βαθιάς Μάθησης

Η Βαθιά Μάθηση είναι περιοχή της τεχνητής νοημοσύνης και αποτελεί εξέλιξη της Μηχανικής Μάθησης. Ασχολείται με μοντέλα που βασίζονται σε πολυεπίπεδα νευρωνικά δίκτυα, καθώς οι επιστήμονες έχουν εμπνευστεί τη δομή των μοντέλων της Βαθιάς Μάθησης από τον ανθρώπινο εγκέφαλο, κατασκευάζοντας με μη γραμμικούς συνδυασμούς και με μαθηματικούς κανόνες τον τρόπο που οι βιολογικοί νευρώνες σηματοδοτούν ο ένας τον άλλον στην επίλυση κάποιου προβλήματος ή το πώς οι νευρώνες μεταδίδουν τα σήματα αναμεταξύ τους και αναγνωρίζουν ή αντιδρούν σε κάποιο εξωτερικό ερέθισμα, όπως για παράδειγμα στο φως, στα χρώματα, στα αντικείμενα, στη συμπεριφορά.

Τα συστήματα Βαθιάς Μάθησης εκπαιδεύονται και μαθαίνουν απευθείας από τα δεδομένα, εφόσον αποτελούνται από πολλά επίπεδα επεξεργασίας, τα οποία χρησιμοποιούνται για την κατανόηση της αναπαράστασης δεδομένων με πολλά επίπεδα αφαίρεσης. Δηλαδή τα συστήματα αυτά, αφού κατασκευαστούν και εκπαιδευτούν από άκρο σε άκρο αποτελούν πλέον αυτοδίδακτα εργαλεία, που μαθαίνουν φιλτράροντας τα δεδομένα, τα οποία ωθούνται στην είσοδο τους, καθώς αυτά επεξεργάζονται στα κρυφά στρώματα του νευρωνικού δικτύου με παρόμοιο τρόπο όπως το κάνει ο άνθρωπος. Τα συστήματα βαθιάς μάθησης, καθώς έχουν τη δυνατότητα να λειτουργούν πάνω σε ογκώδεις συλλογές δεδομένων, δίνουν λύσεις σε περίπλοκα προβλήματα, που είναι δύσκολο να επιλυθούν από τον άνθρωπο.

Η χρήση μονάδων επεξεργασίας γραφικών (GPU) για την εκπαίδευση νευρωνικών δικτύων επιτάχυνε σημαντικά τους χρόνους εκπαίδευσής τους, καθιστώντας την βαθιά εκμάθηση πιο πρακτική. Οι μέθοδοι βαθιάς μάθησης άρχισαν να παρουσιάζουν αποτελέσματα αιχμής σε διάφορους τομείς.

2.2. Εφαρμογές της Βαθιάς Μάθησης

Υπάρχει πολύ μεγάλη ποικιλία εφαρμογών που χρησιμοποιούν αλγόριθμους (τεχνικές) Βαθιάς Μάθησης. Μερικές από αυτές μας είναι πολύ γνώριμες και αλληλοεπιδρούμε μαζί τους στην καθημερινότητα μας. Τα Alexa της Amazon, Cortana της Microsoft, Siri της Apple είναι εικονικοί βοηθοί (Virtual Assistants) που βασίζονται σε Cloud υπηρεσίες, δέχονται φωνητικές εντολές και ολοκληρώνουν εργασίες για τον χρήστη προσφέροντας κάθε φορά μια βελτιωμένη εμπειρία χρήστη. Τα chatbots, που λύνουν προβλήματα πελατών, παρέχοντας αλληλεπίδραση με τους πελάτες και αυτοματοποιημένες απαντήσεις. Οι εφαρμογές στον κλάδο της υγείας, που μπορούν να ενοποιήσουν μέσω της ιατρικής απεικόνισης και να κάνουν διάγνωση σοβαρών ασθενειών καθώς και οι εφαρμογές που χρησιμοποιούνται για ιατρική έρευνα και εύρεση νέων φαρμάκων. Στην ψυχαγωγία συναντάμε τις τεχνικές βαθιάς μάθησης που, με βάση το ιστορικό περιήγησης και τα ενδιαφέροντα των χρηστών, προωθούν προτάσεις για να βοηθήσουν τους χρήστες να επιλέξουν προϊόντα και υπηρεσίες. Στη ρομποτική κατασκευάζονται ρομπότ που τροφοδοτούνται από τους αλγόριθμους βαθιάς μάθησης και εκτελούν εργασίες μιμούμενα τον άνθρωπο, καθώς υπάρχει

δυνατότητα ενημέρωσης τους σε πραγματικό χρόνο, όπως είναι τα ρομπότ της Boston Dynamics, που αντιδρούν όταν κάποιος τα σπρώχνει, ή τα ρομπότ που μελετούν, κατανοούν και επεξεργάζονται την ανθρώπινη γλώσσα, τα αυτοοδηγούμενα αυτοκίνητα και τα drones που χρησιμοποιούνται για την παράδοση φαγητού και παραγγελίας, τα οποία μαθαίνουν να ανταποκρίνονται και να ενεργούν σε διάφορα σενάρια της καθημερινότητας, χρησιμοποιώντας εκατομμύρια σύνολα δεδομένων για την εκπαίδευση τους. Επίσης τα μοντέλα σύνθεσης μουσικής που βασίζονται σε DL, τα μοντέλα χρωματισμού εικόνας, που ταξινομούν τα χρώματα και μαθαίνουν να χρωματίζουν τις εικόνες εφαρμόζοντας αντιληπτική και σημασιολογική κατανόηση των χρωμάτων, τα συστήματα οπτικής αναγνώρισης, τα μοντέλα που ανιχνεύουν την απατή, τα συστήματα που ασχολούνται με τις εφαρμογές δημογραφικών και εκλογικών προβλέψεων.

Η Βαθιά Μάθηση έχει αναπτυχθεί πάρα πολύ τα τελευταία χρόνια προχωρώντας με ταχείς ρυθμούς, με τους ερευνητές να ερευνούν νέες αρχιτεκτονικές, νέες τεχνικές βελτιστοποίησης και μεταφοράς μάθησης (Transfer Learning), που οδηγούν στη μελλοντική ανάπτυξη καινούργιων εφαρμογών, επιλύοντας δύσκολα προβλήματα και κάνοντας ευκολότερες τις εργασίες μας.

Το συνολικό αποτέλεσμα των προσπαθειών και ερευνών οδήγησε σε δίκτυα βαθιάς μάθησης και στην εφαρμογή τους στη μηχανική μάθηση. Αναπτύχθηκε η Εξόρυξη δεδομένων, η οποία ήταν η πρώτη μορφή της μηχανικής μάθησης. Χρησιμοποιήθηκε το κρυφό μοντέλο Markov για αναγνώριση ομιλίας σε συνδυασμό με βάσεις δεδομένων, οι οποίες περιείχαν τεράστιες συλλογές δεδομένων.

Τα τελευταία χρονιά η βαθιά μάθηση σημείωσε σημαντική πρόοδο, αναπτύχθηκαν νέα εργαλεία, που εφαρμόζονται σε προβλήματα όπως η αναγνώριση και η σύνθεση ομιλίας, αναγνώριση προτύπων και αντικειμένων, μετάφραση κειμένων από εικόνες, πρόβλεψη, ταξινόμηση, παλινδρόμηση. Έχουν σχεδιαστεί εφαρμογές που συνέβαλαν στη βελτίωση σε τομείς της ανθρώπινης προσπάθειας, όπως στην αλληλεπίδραση ανθρώπου υπολογιστή, στην υγειονομική περίθαλψη, στην ιατρική ερευνά, στην επικοινωνία και σε πολλούς άλλους τομείς.

2.3. Νευρωνικά Δίκτυα Συνέλιξης (Convolutional Neural Networks - CNN)

Τα νευρωνικά δίκτυα συνέλιξης (CNN) αποτελούν μια κατηγορία βαθιών νευρωνικών δικτύων και έχουν σχεδιαστεί ειδικά για να λύσουν περιπλοκά προβλήματα, που σχετίζονται με την ταξινόμηση εικόνων, ανίχνευση αντικειμένων, τμηματοποίηση εικόνας. Είναι αποτελεσματικά στο χειρισμό δεδομένων που μοιάζουν με πλέγμα όπως εικόνες, δεδομένα αισθητήρων, χρονοσειρών και 2D πλέγματος.

2.3.1. Η αρχιτεκτονική των δικτύων CNN (Convolutional Neural Networks)

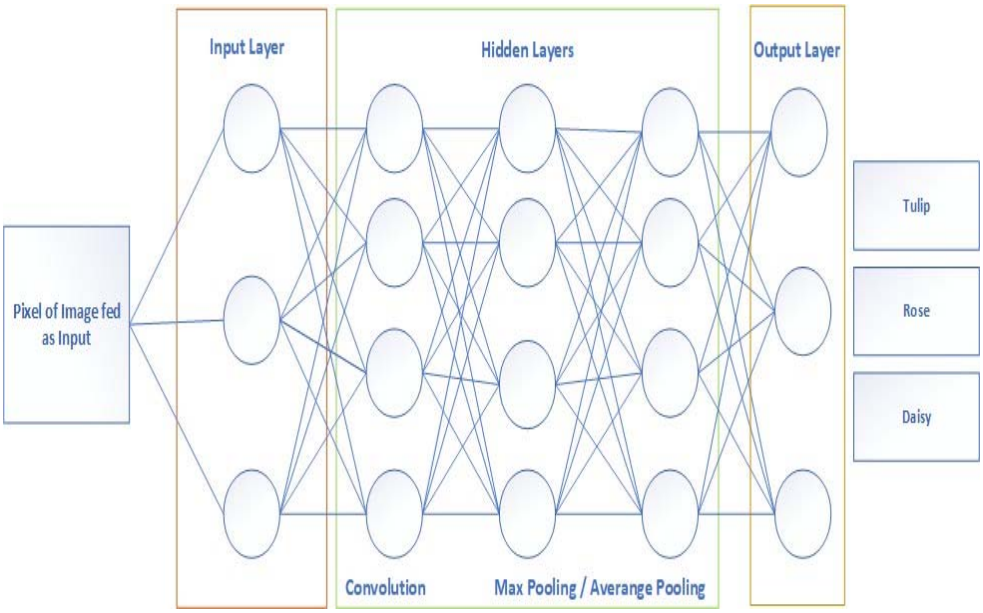
Τα νευρωνικά δίκτυα συνέλιξης αποτελούνται από τα ακόλουθα βασικά δομικά στοιχεία:

- **Το επίπεδο εισόδου (Input Layer)** δέχεται τα δεδομένα εισόδου και τα προωθεί προς το επόμενο επίπεδο
- **Τα Συνελικτικά επίπεδα (Convolutional Layers),** τα οποία αποτελούνται από ένα σύνολο φίλτρων γνωστών και ως πυρήνες, που χρησιμοποιούν

λειτουργίες συνέλιξης, σαρώνοντας τα δεδομένα εισόδου και εντοπίζουν χαρακτηριστικά σε διαφορετικές χωρικές ιεραρχίες.

- **Τα επίπεδα σμίκρυνσης (Pooling Layers)** εφαρμόζουν λειτουργίες σμίκρυνσης όπως το μέγιστο (Max Pooling) και ο μέσος όρος (Average Pooling). Χρησιμοποιούνται στη μείωση των διαστάσεων των επιπέδων, των παραμέτρων που πρέπει να υπολογιστούν και της υπολογιστικής πολυπλοκότητας, βοηθώντας στην αποτύπωση των πιο σημαντικών χαρακτηριστικών.
- **Τα πλήρως συνδεδεμένα επίπεδα (Fully Connected Layers)** συνδέουν όλους τους νευρώνες του προηγουμένου στρώματος με κάθε νευρώνα του τρέχοντος στρώματος και είναι υπεύθυνα για την εκμάθηση μοτίβων και την πραγματοποίηση προβλέψεων με βάση τα χαρακτηριστικά που έχουν μάθει τα προηγούμενα επίπεδα.
- **Το επίπεδο εξόδου (Output Layer)** παράγει τις τελικές προβλέψεις ή εξόδους. Η δομή και τα χαρακτηριστικά αυτού του επιπέδου εξαρτώνται από τη φύση της εργασίας που καλείται να πραγματοποιήσει το μοντέλο και από την επιθυμητή μορφή εξόδου.
  Παρακάτω ακολουθούν κάποια παραδείγματα επιπέδων εξόδου CNN που λύνουν διαφορά προβλήματα.
  1. Σε κάποιες περιπτώσεις ταξινόμησης το επίπεδο εξόδου είναι ένα πλήρως συνδεδεμένο στρώμα γνωστό και ως πυκνό στρώμα, όπου κάθε νευρώνας αντιστοιχεί σε μια συγκεκριμένη κατηγορία η κλάση.
  2. Το επίπεδο εξόδου μπορεί να είναι ένα συνελικτικό επίπεδο. Το συναντάμε στην τμηματοποίηση εικόνας, όπου η έξοδος είναι μια εικόνα με προβλέψεις ή μάσκες που προέκυψαν από τα εικονοστοιχεία. Η έξοδος του μοντέλου παράγει έναν χάρτη χαρακτηριστικών με πολλά κανάλια, όπου κάθε κανάλι αντιστοιχεί σε διαφορετική κατηγορία ή κλάση.
  3. Σε προβλήματα παλινδρόμησης το επίπεδο εξόδου είναι συχνά ένας μεμονωμένος νευρώνας, που παράγει μια συνεχή τιμή.
  4. Για ανίχνευση αντικειμένων το επίπεδο εξόδου περιέχει προσαρμοσμένα επίπεδα νευρώνων. Καθένα από τα επίπεδα αυτά προβλέπει διαφορετικές πτυχές των ανιχνευόμενων αντικειμένων, όπως συντεταγμένες πλαισίου οριοθέτησης, πιθανότητες κλάσεων, μάσκες αντικειμένων.

Η αρχιτεκτονική ενός CNN αναπαρίσταται στο **Σχήμα 2.3.1.α**.



Σχήμα 2.3.1.α. Συνελικτικό Νευρωνικό Δίκτυο(CNN)

2.3.2. Μαθηματική αποτύπωση

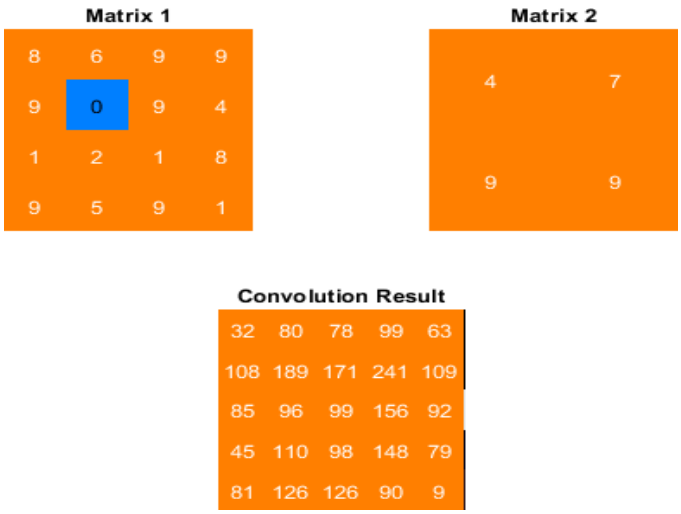
Η μαθηματική αποτύπωση της λειτουργίας των CNN δικτύων ορίζεται από διάφορους μαθηματικούς τύπους. Ένα τυπικό CNN δίκτυο χρησιμοποιεί τις παρακάτω εξισώσεις:

1. Συνέλιξη - εφαρμόζεται σε εικόνα ή σε χάρτες χαρακτηριστικών εισόδου χρησιμοποιώντας ένα σύνολο φίλτρων/πυρήνες και μπορεί να αναπαρασταθεί μαθηματικά ως:

$$C(i,j) = (IK)(x,y) = \sum_{\iota=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(x-i, y-j)K(i,j)$$

Όπου C(i,j) είναι ο χάρτης των χαρακτηριστικών εξόδου στη θέση (i,j). Το K(i,j) είναι το αντίστοιχο στοιχείο του φίλτρου στη θέση (i,j), I(i-x, j-y) είναι τα δεδομένα εισόδου του χάρτη χαρακτηριστικών στη θέση (i-x, j-y).

Στο Σχήμα 2.3.2.α απεικονίζεται το αποτέλεσμα της συνέλιξης δυο δισδιάστατων πινάκων με τυχαίες τιμές.



Σχήμα 2.3.2.α: Συνέλιξη δυο δισδιάστατων πινάκων.

2. Αφού πραγματοποιηθεί η συνέλιξη εφαρμόζεται σε κάθε στοιχείο μια συνάρτηση ενεργοποίησης όπως η ReLU, sigmoid, tanh για να εισαχθεί η μη γραμμικότητα στο δίκτυο.
3. Οι λειτουργίες συγκέντρωσης Max Pooling διατηρούν τα πιο σημαντικά χαρακτηριστικά της εισόδου μειώνοντας της διαστάσεις εισόδου. Η Average Pooling βοηθάει στον εντοπισμό της παρουσίας ακραίων τιμών.

Οι παρακάτω εξισώσεις υπολογίζουν τις διαστάσεις n x m του πίνακα εξόδου

$$n = \lfloor \frac{N - P_n}{S} \rfloor + 1$$

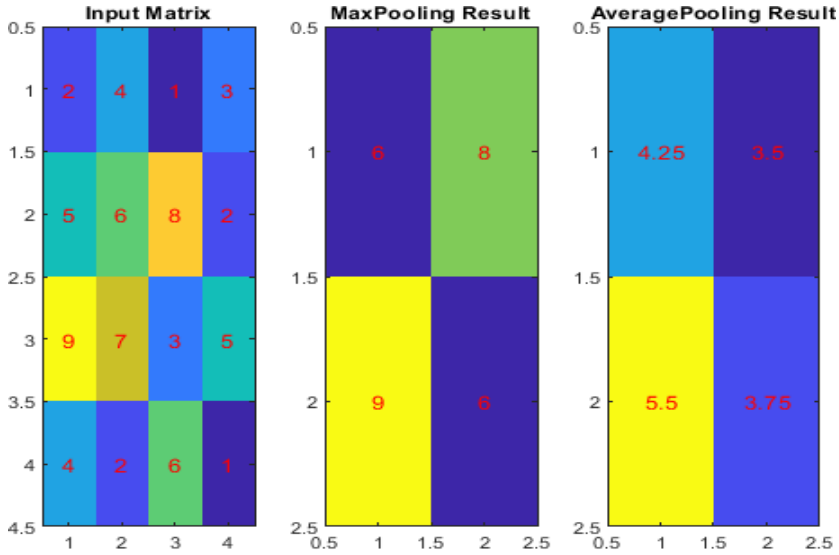
όπου n είναι η μειωμένη διάσταση του πίνακα εξόδου, N είναι η διάσταση του πίνακα εισόδου κατά το μήκος της διάστασης N, P_n είναι το μέγεθος του παραθύρου συγκέντρωσης κατά μήκος της διάτασης n και το S (stride) αντιπροσωπεύει το μέγεθος του βήματος μετακίνησης του παραθύρου συγκέντρωσης .

$$m = \lfloor \frac{M - P_m}{S} \rfloor + 1$$

Και εδώ ισχύουν τα παραπάνω αλλά για τη μειωμένη διάσταση m του πίνακα εξόδου με βάση τη διάσταση M του πίνακα εισόδου και του μέγεθος m του παραθύρου συγκέντρωσης (P_m) που κανει ολίσθηση με βήμα S πάνω στα δεδομένα της μήτρας εισόδου I_{NxM}.

Στη συνέχεια με βάση την περιοχή συγκέντρωσης P_{nxm} που ολισθαίνει με βήμα S από τα στοιχεία της μήτρας εισόδου I_{NxM} εξάγεται η μεγίστη τιμή ή υπολογίζεται ο μέσος όρος των τιμών αυτών και τοποθετούνται στο πίνακα εξόδου με βήμα 1.

Στο παρακάτω **Σχήμα 2.3.2.β.** απεικονίζεται η υλοποίηση Max Pooling και Average Pooling με το Matlab για δισδιάστατο πίνακα εισόδου I_{4x4} και περιοχή συγκέντρωσης P_{2x2} και βήμα (stride) S=2.



Σχήμα 2.3.2.β: Λειτουργίες Συγκεντρώσεις Max Pooling/Average Pooling.

4. Στα πλήρως συνδεδεμένα στρώματα χρησιμοποιείται ένα μοτίβο πυκνής σύνδεσης, όπου κάθε νευρώνας σε ένα στρώμα συνδέεται με κάθε νευρώνα στο επόμενο στρώμα. Μπορούμε να ορίσουμε ένα απλό πλήρως συνδεδεμένο στρώμα χωρίς πρόσθετες πολυπλοκότητες ή περιορισμούς (κοινόχρηστα βάρη, περιορισμούς στις τιμές των βαρών) με τον εξής τρόπο:

   Έστω ότι το τρέχον στρώμα αποτελείται από N νευρώνες και το προηγούμενο στρώμα αποτελείται από M νευρώνες, όπου κάθε νευρώνας στο τρέχον στρώμα έχει ένα βάρος (weight), που σχετίζεται με κάθε νευρώνα στο προηγούμενο στρώμα καθώς και μια νευρωνική πόλωση (bias), τότε για ένα πλήρως συνδεδεμένο στρώμα με n^2 νευρώνες (M= n^2 και N= n^2) ο αριθμός των παραμέτρων είναι $n^4+ n^2$ διότι ο αριθμός των βαρών σε αυτό το στρώμα ισούται με τον αριθμό των συνδέσεων και των δυο στρωμάτων, που είναι $MN = (n^2)^2$ και ο αριθμός των όρων μεροληψίας (biases) ισούται με τον αριθμό των νευρώνων, δηλαδή με n^2.

   Επομένως για κάθε νευρώνα στο τρέχον στρώμα θα χρειαστούν M πολλαπλασιασμοί και M-1 προσθέσεις και αφού υπάρχουν N νευρώνες τότε με απλή αντικατάσταση οι συνολικές πράξεις ισούνται με $n^2(2n^2 - 1)$.

2.3.3. Αλγόριθμος εκμάθησης των Νευρωνικών Συνελικτικών Δικτύων

Ο αλγόριθμος εκμάθησης αποτελείται από τα ακόλουθα βήματα:

1. Τυχαία αρχικοποίηση των βαρών και των προκαταλήψεων του δικτύου. Τα βάρη προσαρμόζονται σταδιακά κατά τη διάρκεια της εκπαίδευσης του δικτύου.

2. Forward Pass (Εμπρόσθια Διάδοση)
   Τα δεδομένα εισόδου διαβιβάζονται προς τα μπροστά μέσω μιας σειράς επιπέδων για τη δημιουργία προβλέψεων εφαρμόζοντας λειτουργίες συνέλιξης, ενεργοποίησης και ομαδοποίησης αν χρειαστεί.
3. Υπολογισμός Απώλειας
   Σε αυτό το βήμα συγκρίνονται οι προβλέψεις του δικτύου με τις πραγματικές τιμές για να υπολογιστεί το σφάλμα του δικτύου.
4. Backprobagation (Οπίσθια Διάδοση)
   Το σφάλμα μεταφέρεται προς τα πίσω μέσω των επιπέδων του δικτύου και για κάθε επίπεδο υπολογίζεται η απώλεια σε σχέση με το σταθμισμένο άθροισμα των εισροών του επιπέδου.
5. Ενημέρωση βαρών
   Τα βάρη προσαρμόζονται χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης (Stochastic Gradient Descent) για να ελαχιστοποιηθεί το σφάλμα δικτύου.
6. Πολλαπλή επανάληψη
   Γίνεται μια επανάληψη για πολλές εποχές της παραπάνω διαδικασίας στα δεδομένα εκπαίδευσης, προσαρμόζοντας τα βάρη έως ότου η απώλεια συγκλίνει σε ένα ικανοποιητικό επίπεδο.

2.3.4. Τεχνικές που χρησιμοποιούνται στα Νευρωνικά Δίκτυα Συνέλιξης

Συνέλιξη (Convolution)

Κατά την τεχνική της συνέλιξης εφαρμόζονται κάποια φίλτρα (πυρήνες) στα δεδομένα εισόδου, επιτρέποντας στο δίκτυο να μάθει ιεραρχικά χαρακτηριστικά από αυτά.

Συγκέντρωση (Pooling)

Στη συγκέντρωση μειώνονται οι διαστάσεις των χαρτών των χαρακτηριστικών της εισόδου, μειώνοντας την υπολογιστική πολυπλοκότητα και αυξάνοντας το οπτικό πεδίο του δικτύου.

Τεχνικές ενεργοποίησης (Activation Functions)

Οι μη γραμμικές συναρτήσεις όπως η ReLU εισάγουν τη μη γραμμικότητα στο δίκτυο επιτρέποντας του να μοντελοποιήσει περίπλοκες σχέσεις. Οι συναρτήσεις SoftMax παράγουν στην έξοδο μια πιθανότητα για κάθε μια από τις εικόνες, χρησιμοποιώντας την απώλεια Softmax (Softmax Loss), γνωστή και ως Categorical Cross Entropy, εντάσσοντας κάθε ένα από τα χαρακτηριστικά στην κλάση που τους αναλογεί. Η απώλεια Softmax μετρά τη διαφορά μεταξύ της προβλεπόμενης κατανομής πιθανότητας και της αληθινής κατανομής.

Εγκατάλειψη (Dropout)

Το Dropout θέτει τυχαία ένα κλάσμα των ενεργοποιήσεων των μονάδων εισόδου στο μηδέν σε κάθε ενημέρωση κατά τη διάρκεια του χρόνου εκπαίδευσης, βοηθώντας το δίκτυο να αποφύγει την υπερπροσαρμογή, καθιστώντας το πιο εύρωστο και λιγότερο εξαρτώμενο από συγκεκριμένους νευρώνες. Κατά τη διάρκεια της εκπαίδευσης η εγκατάλειψη απενεργοποιεί τους νευρώνες με μια συγκεκριμένη πιθανότητα P. Κατά την εξαγωγή συμπερασμάτων τα αποτελέσματα κλιμακώνονται με το ποσοστό εγκατάλειψης 1- P, που διατηρεί αναμενόμενη τιμή των εκροών.

Αύξηση δεδομένων (Data Augmentation)

Αυτή η τεχνική περιλαμβάνει την εφαρμογή τυχαίων μετασχηματισμών (περιστροφές, ανατροπές, τυχαία κλιμάκωση) στα δεδομένα εκπαίδευσης για να αυξηθεί η ποικιλομορφία τους και να εισάγουν μεταβλητότητα και ευρωστία σε διάφορες κλίμακες αντικειμένων στις εικόνες.

Κανονικοποίηση δεδομένων εισόδου(Input Data Normalization)

Τα συνελικτικά νευρωνικά δίκτυα (CNN) εκτελούν κανονικοποίηση των δεδομένων εισόδου για να διασφαλίσουν ότι τα δεδομένα που τροφοδοτούνται στο δίκτυο είναι σε συνεπή κλίμακα, κάτι που βοηθά στη βελτίωση της διαδικασίας εκπαίδευσής και της απόδοσης του μοντέλου. Κάποιες από τις πιο γνωστές τεχνικές κανονικοποίησης δεδομένων εισόδου είναι:

- **Μέση Αφαίρεση (Mean Subtraction)** - υπολογίζεται η μέση τιμή pixel σε ολόκληρο το σύνολο δεδομένων και στη συνέχεια αφαιρείται από την αρχική τιμή Pixel κάθε δεδομένου εισόδου. Η μαθηματική διατύπωση δίνεται από τον ακόλουθο τύπο:

$$p' = p - \bar{x}$$

Όπου:
p'- είναι η κανονικοποιημένη τιμή pixel μετά από την αφαίρεση της μέσης τιμής $\bar{x}$.
P - είναι η αρχική τιμή pixel κάθε δεδομένου εισόδου.
$\bar{x}$ – η μέση τιμή όλων των δεδομένων εισόδου.

Κανονικοποίηση Z-score (Z-score Normalization) - υπολογίζεται ο μέσος όρος και η τυπική απόκλιση όλων των δεδομένων εισόδου και στη συνέχεια σε κάθε δεδομένο εισόδου αφαιρείται η μέση τιμή και διαιρείται με την τυπική απόκλιση του συνόλου δεδομένων εισόδου. Ο μαθηματικός τύπος είναι ο ακόλουθος:

$$\mathrm{p}' = \frac{\mathrm{p} - \bar{\mathrm{x}}}{\sigma}$$

Όπου:
p'- είναι η κανονικοποιημένη τιμή pixel.
p - είναι η αρχική τιμή pixel κάθε δεδομένου εισόδου.
$\bar{\mathrm{x}}$- η μέση τιμή όλων των δεδομένων εισόδου
σ - η τυπική απόκλιση του συνόλου των δεδομένων εισόδου.

- **Κανονικοποίηση Ελάχιστο-Μέγιστο (Min-Max Normalization)** - προσδιορίζονται η ελάχιστη και η μεγίστη τιμή σε όλα τα δεδομένα εισόδου και στη συνέχεια από κάθε δεδομένο εισόδου αφαιρείται η ελάχιστη τιμή και διαιρείται με την τιμή που προκύπτει από την αφαίρεση της ελαχίστης τιμής από τη μεγίστη. Δίνεται από τον τύπο:

$$\mathrm{p}' = \frac{\mathrm{p} - \min}{\max - \min}$$

Όπου:

p'- είναι η κανονικοποιημένη τιμή pixel συνηθώς στο εύρος των τιμών [0,1] η [-1,1]
p - είναι η αρχική τιμή pixel κάθε δεδομένου εισόδου.
min - η ελάχιστη τιμή όλων των δεδομένων εισόδου
max - η μέγιστη τιμή όλων των δεδομένων εισόδου

- **Κανονικοποίηση παρτίδας (Batch Normalization)** - Η ομαλοποίηση του δείγματος εκπαίδευσης κανονικοποιεί τις ενεργοποιήσεις κάθε επιπέδου ώστε να έχουν μηδενικό μέσο όρο και διακύμανση μονάδας, γεγονός που μπορεί να επιταχύνει την εκπαίδευση και να βελτιώσει τη γενίκευση του δικτύου. Η κανονικοποίηση παρτίδας εφαρμόζεται σε μικρά σύνολα (mini-batches), προσθέτει δυο παραμέτρους με δυνατότητα εκμάθησης ανά χαρακτηριστικό, κλίμακα και μετατόπιση, επιτρέποντας στο μοντέλο να αναιρέσει την κανονικοποίηση εάν είναι απαραίτητο. Δίνεται από την ακόλουθη μαθηματική πράξη:

$$p' = \frac{p - \bar{x}_{batch}}{\sigma_{batch}}$$

Όπου:
p'- είναι η κανονικοποιημένη τιμή pixel κάθε δεδομένο της παρτίδας
p - είναι η αρχική τιμή pixel κάθε δεδομένου της παρτίδας.
$\bar{x}$ – η μέση τιμή όλων των δεδομένων παρτίδας
σ_{batch}– η τυπική απόκλιση παρτίδας.

Αποσύνθεση/Τακτοποίηση βάρους (Regularization)

Σημαντική τεχνική για την εκπαίδευση των συνελικτικών Νευρονικών Δικτύων (CNN), κατά την οποία τα βάρη προσαρμόζονται για να μειωθεί το σφάλμα του μοντέλου για την αποφυγή της υπερπροσαρμογής (overfitting) και για τη διασφάλιση της γενίκευσης του μοντέλου σε νέα δεδομένα.

Οι πιο γνωστές μέθοδοι αποσύνθεσης/τακτοποίησης είναι οι L1 και L2.

Η τακτοποίηση L1, γνωστή και ως παλινδρόμηση Lasso, προσθέτει μια ποινή στη συνάρτηση απώλειας (loss function) ανάλογη με τις απόλυτες τιμές των βαρών του μοντέλου. Αυτό οδηγεί στο να μηδενιστούν τα λιγότερο σημαντικά βάρη των χαρακτηριστικών, δημιουργώντας μοντέλα που είναι πιο εύκολο να ερμηνευτούν λόγω της αραιότητας των βαρών, μειώνοντας την υπερπροσαρμογή (overfitting), την πολυπλοκότητα του μοντέλου και προσδιορίζοντας τα πιο σημαντικά χαρακτηριστικά. Δίνεται από τον παρακάτω μαθηματικό τύπο:

$$Loss_{new} = Loss_{old} + \lambda \sum_{i=1}^{n} |w_i|$$

Όπου:

$Loss_{new}$ είναι η καινούργια συνάρτηση κόστους/απώλειας μετά από την εφαρμογή της αποσύνθεσης.

$Loss_{old}$ κρατάει την τιμή της παλιάς συνάρτησης κόστους.

Ο όρος λ αναφέρεται στη παράμετρο τακτοποίησης.

$\sum_{i=1}^{n} |w_i|$ είναι το άθροισμα των απόλυτων τιμών των βαρών του μοντέλου.

Η αποσύνθεση/τακτοποίηση L2, που επίσης τη συναντάμε και ως παλινδρόμηση κορυφογραμμής, προσθέτει μια ποινή στη συνάρτηση απώλειας ανάλογα με το άθροισμα των τετράγωνων των βαρών της συνάρτησης κόστους/απώλειας. Η αποσύνθεση L2 ενθαρρύνει το δίκτυο να διατηρεί τα βάρη σε μικρές τιμές χωρίς να μηδενίσει κανένα βάρος, γεγονός που μπορεί να οδηγήσει σε απλουστέρα μοντέλα που γενικεύουν καλυτέρα. Η μαθηματική διατύπωση της L2 είναι η ακόλουθη:

$$Loss_{new} = Loss_{old} + \lambda \sum_{i=1}^{n} |w_i^2|$$

Όπου:

$Loss_{new}$ είναι η καινούργια συνάρτηση κόστους/απώλειας μετά την εφαρμογή της αποσύνθεσης.

$Loss_{old}$ κρατάει την τιμή της παλιάς συνάρτησης κόστους.

Ο όρος λ αναφέρεται στη παράμετρο τακτοποίησης.

$\sum_{i=1}^{n} |w_i^2|$ είναι το άθροισμα των τετράγωνων των βαρών

Προγραμματισμός ποσοστού μάθησης (Learning rate Scheduling)

Αφορά την προσαρμογή του ρυθμού μάθησης του μοντέλου μετά από έναν ορισμένο αριθμό εποχών και βοηθάει στη σταθεροποίηση της εκπαίδευσης και στη βελτίωση της σύγκλησης.

2.3.5. Περιορισμοί των δικτυών CNNs

Τα συνελικτικά νευρωνικά δίκτυα αποτελούν ισχυρά εργαλεία για ανάλυση και επεξεργασία δεδομένων, ωστόσο η εκπαίδευση και η λειτουργία των CNN δικτύων με πάρα πολλές στρώσεις απαιτούν υψηλή μνήμη, εξειδικευμένο υλικό και σημαντικούς υπολογιστικούς πόρους.

Τα CNNs είναι επιρρεπή στην υπερπροσαρμογή, ειδικά όταν το σύνολο των δεδομένων εκπαίδευσης είναι περιορισμένο. Σε αυτή την περίπτωση το μοντέλο έχει καλή απόδοση στο σετ των δεδομένων εκπαίδευσης και αποτυγχάνει σε αόρατα δεδομένα.

Τα συνελικτικά δίκτυα είναι γνωστά και ως μοντέλα «μαύρου κουτιού», πράγμα που σημαίνει ότι είναι δύσκολο να ερμηνευτούν κάποιες προβλέψεις που πραγματοποιούν. Οι αρχιτεκτονικές που έχουν σχεδιαστεί για οπτικά δεδομένα απαιτούν είσοδο σταθερού μεγέθους, που σημαίνει ότι οι εικόνες πρέπει να αλλάζουν μέγεθος ή να περικόπτονται, ώστε να ταιριάζουν στο επίπεδο εισόδου, κάτι που μπορεί να οδηγήσει σε απώλεια πληροφοριών ή στην παραμόρφωση και ενδέχεται να μη γενικεύουν καλά σε άλλους τύπους δεδομένων. Είναι εγγενώς δίκτυα προώθησης που σημαίνει ότι δεν διαθέτουν σαφείς μηχανισμούς για το χειρισμό χρονικών η διαδοχικών δεδομένων.

2.4. Ανατροφοδοτούμενα δίκτυα (Recurrent Neural Networks - RNN)

Ένα ανατροφοδοτούμενο δίκτυο είναι ένας τύπος νευρωνικού δικτύου κατάλληλος για την επεξεργασία ακολουθιών δεδομένων όπως για παράδειγμα οι χρονοσειρές, που ποικίλουν ανάλογα με το χρόνο, τα δεδομένα κειμένου που χρησιμοποιούνται ευρέως σε εργασίες όπως η αυτόματη μετάφραση, ή στην ανάλυση συναισθημάτων και στη δημιουργία κειμένου, τα δεδομένα ήχου για την αναγνώριση ομιλίας, τα μουσικά δεδομένα για τη δημιουργία νέας μουσικής ή για την ανάλυση υπαρχουσών συνθέσεων, οι αλληλουχίες DNA για εργασίες όπως ή πρόβλεψη γονίδιων ή η εύρεση μοτίβων και η ταξινόμηση γενετικών δεδομένων, οι ακολουθίες που περιλαμβάνουν κινήσεις όπως η ερμηνεία της νοηματικής γλώσσας ή η καταγραφή κινήσεων, οι ακολουθίες εικόνων ή τα δεδομένα LiDAR για εργασίες όπως είναι η ανίχνευση αντικειμένων, η παρακολούθηση λωρίδας και η λήψη αποφάσεων.

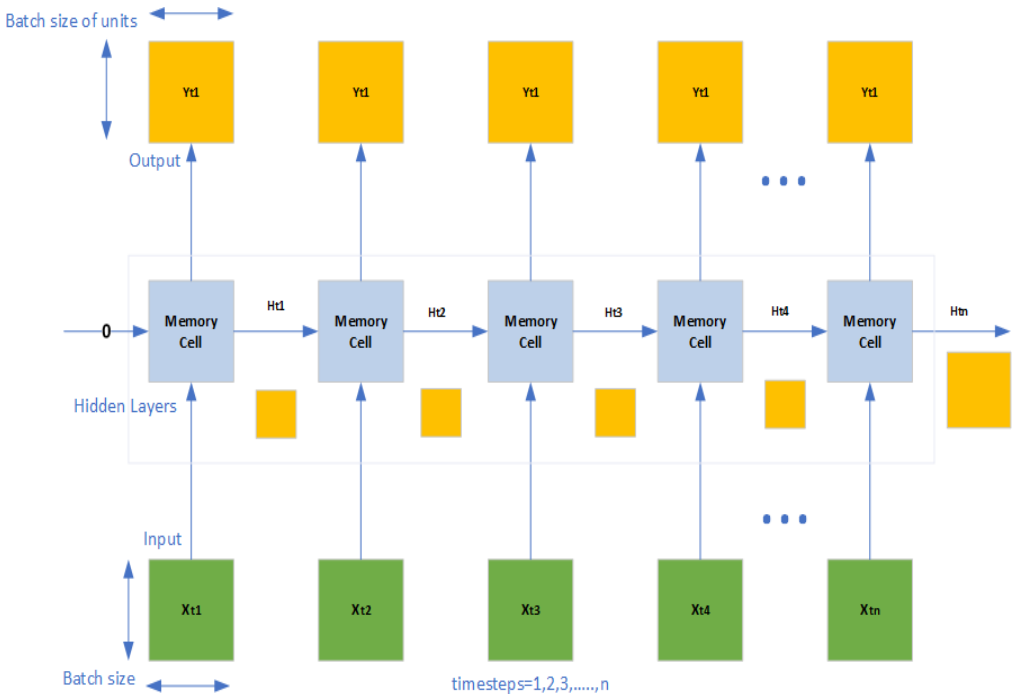
2.4.1. Απλά επαναλαμβανόμενα δίκτυα (Simple Recurrent Networks)

Τα απλά RNN δίκτυα έχουν συνδέσεις που σχηματίζουν μια δομή που μοιάζει με αλυσίδα, οπού η έξοδος ενός κελιού μνήμης μεταφέρεται στην είσοδο του επομένου κελιού και κάθε κελί δέχεται σαν είσοδο την τρέχουσα είσοδο καθώς και την κρυφή κατάσταση του προηγούμενου χρονικού βήματος. Αυτή η «μνήμη» αποτελείται από επαναλαμβανόμενες μονάδες, που ευθύνονται για τη διατήρηση πληροφοριών από προηγούμενα χρονικά βήματα και την ενημέρωσή τους. Υπάρχουν διάφοροί τύποι επαναλαμβανόμενων μονάδων, όπου η καθεμιά από αυτές έχει το δικό της τρόπο ενημέρωσης και μετάδοσης πληροφοριών στο χρόνο.

2.4.2. Βασική Αρχιτεκτονική των RNN

Η αρχιτεκτονική ενός απλού ανατροφοδοτούμενου δικτύου (Simple RNN) αποτελείται από:

1. Επίπεδο εισόδου που δέχεται την ακολουθία εισόδου (το batch size και τον αριθμό των χρονικών βημάτων timesteps).
2. Επαναλαμβανόμενες μονάδες. Αυτές οι μονάδες διατηρούν μια κρυφή κατάσταση που ενημερώνεται σε κάθε χρονικό βήμα. Η κρυφή κατάσταση βοηθά το δίκτυο να θυμάται προηγούμενες πληροφορίες.
3. Επίπεδο εξόδου. Παράγει την έξοδο του μοντέλου με βάση τις επεξεργασμένες πληροφορίες.



Σχήμα 2.4.2.α: Αρχιτεκτονική RNN

2.4.3. Μαθηματική αναπαράσταση ενός απλού RNN

Για ένα χρονικό βήμα t ο υπολογισμός σε ένα απλό RNN μπορεί να αναπαρασταθεί ως εξής:

$$h(t) = f(\sum_{i=1}^{n}(x_t \quad w_{it}) + b_{it} + \sum_{i=1}^{n}(x_t \quad w_{ih(t-1)}) + b_{ih(t-1)})$$

$$y(t) = g(\sum_{j=1}^{n}(h(t) \ W_j) + b_j)$$

Όπου $h(t)$ είναι το κρυφό στρώμα στο χρονικό βήμα t και περιέχει το σταθμισμένο άθροισμα της τρέχουσας κατάστασης $\sum^{n}{}_{i=1}\, x_t \quad w_{it}$ πολωμένο από τους σταθερούς όρους b_{it} (biases), όπου στη συνέχεια προστίθεται το σταθμισμένο άθροισμα της προηγουμένης κατάστασης $\sum_{i=1}^{n}(x_t\, w_{ih(t-1)})$ που είναι πολωμένη από τους σταθερούς ορούς $b_{ih(t-1)}$(biases). Τέλος η μαθηματική παράσταση που προκύπτει από τις δύο αυτές αθροίσεις ενεργοποιείται από τη συνάρτηση ενεργοποίησης f.

Η έξοδος y στο χρονικό βήμα t ενεργοποιείται από τη συνάρτηση g, η οποία δέχεται στην είσοδο της το σταθμισμένο άθροισμα μεταξύ του κρυφού στρώματος και των βαρών του επιπέδου εξόδου W_j, πολωμένο από τις νευρωνικές πολώσεις b_j.

2.4.4. Συναρτήσεις ενεργοποίησης των απλών RNNs

Οι συναρτήσεις ενεργοποίησης στα απλά ανατροφοδοτούμενα δίκτυα εφαρμόζονται τόσο στην έξοδο των κρυφών στρωμάτων όσο και στο επίπεδο εξόδου όπου παράγεται το προβλεπόμενο αποτέλεσμα. Οι πιο κοινές λειτουργίες ενεργοποίησης που χρησιμοποιούνται σε RNN είναι:

- Η σιγμοειδής συνάρτηση (Sigmoid function)

- Η υπερβολική συνάρτηση εφαπτομένης (Tanh function)
- Η συνάρτηση ράμπας (ReLU function)

2.4.5. Αλγόριθμος εκπαίδευσης απλών RNN

Η εκπαίδευση ενός απλού ανατροφοδοτούμενου δικτύου είναι μια επέκταση του αλγορίθμου Backpropagation, δηλαδή περιλαμβάνει μια διαδικασία οπισθοδρόμησης στο χρόνο (Backpropagation Through Time) και ακολουθεί τα παρακάτω βήματα:

1. Στο πρώτο βήμα γίνεται η τυχαία αρχικοποίηση των βαρών και των προκαταλήψεων.
2. Για κάθε χρονική στιγμή t το δίκτυο υπολογίζει την τρέχουσα κατάσταση h(t) χρησιμοποιώντας και τα δεδομένα της χρονικής στιγμής t-1 δηλαδή της προηγουμένης κατάστασης.
3. Στο τρίτο βήμα υπολογίζεται η απώλεια του δικτύου κατά το χρονικό βήμα t.
4. Σε αυτό το βήμα εφαρμόζεται η διαδικασία της πίσω διάδοσης στο χρόνο που ξεκινάει από τη χρονική στιγμή t και πηγαίνει προς τα πίσω, υπολογίζοντας τη διαβάθμιση της απώλειας σε σχέση με τις παραμέτρους κάθε χρονικού βήματος, χρησιμοποιώντας τον κανόνα της αλυσίδας.
5. Με τη βοήθεια ενός αλγορίθμου βελτιστοποίησης (π.χ. Στοχαστική Κλίση Κατάβασης) ενημερώνονται οι παράμετροι του δικτύου χρησιμοποιώντας τις υπολογίσιμες κλήσεις του δικτύου.
6. Τα βήματα 2-5 επαναλαμβάνονται για ένα σταθερό αριθμό χρονικών βημάτων t ή μέχρι να ικανοποιηθεί ένα κριτήριο σύγκλισης (π.χ. η απώλεια).
7. Το δίκτυο αξιολογείται ως προς την απόδοση με ένα σύνολο δεδομένων επικύρωσης για να παρακολουθείται η πρόοδος του δικτύου και να αποτραπεί η υπερπροσαρμογή.
8. Η προπόνηση του μοντέλου RNN τερματίζει με βάση προκαθορισμένα κριτήρια διακοπής, όπως η συμπλήρωση του μέγιστου αριθμού των χρονικών βημάτων t που έπρεπε να κάνει το δίκτυο ή στην περίπτωση που οι μετρήσεις απόδοσης του δικτύου σταματάνε να μειώνονται.

2.4.6. Περιορισμοί των απλών RNNs

Τα απλά RNNs με πολλά επίπεδα κατά τη διάρκεια τις εκπαίδευσης μπορούν να παρουσιάσουν τα ακόλουθα προβλήματα:

1. Το πρόβλημα της εξαφάνισης της κλίσης (Vanishing Gradient)
   Το πρόβλημα τουVanishing Gradient μπορεί να εμφανιστεί κατά τη διάρκεια της εκπαίδευσης και συγκεκριμένα κατά τη διάρκεια της αντίστροφης διάδοσης στο χρόνο (Backpropagation Through Time), οπού τα βάρη ενημερώνονται προς την κατεύθυνση της αρνητικής κλίσης της συνάρτησης απώλειας. Αν τα βάρη είναι μικρότερα από τη μονάδα και η λειτουργία ενημέρωσης της κρυφής κατάστασής επαναλαμβάνεται για πολλά χρονικά βήματα οι διαβαθμίσεις συρρικνώνονται εκθετικά οδηγώντας σε διαβαθμίσεις που εξαφανίζονται.
2. Το πρόβλημα της εκρηκτικής κλίσης (Exploding Gradient).
   Στο συγκεκριμένο πρόβλημα αντί οι διαβαθμίσεις να γίνονται πολύ μικρές γίνονται πολύ μεγάλες κατά τη διάρκεια της οπισθοδρόμησης στο χρόνο. Αυτό μπορεί να οδηγήσει σε αριθμητική αστάθεια του δικτύου (να παρουσιάζει τιμές

NaN) ή να ωθήσει τις προβλέψεις του μοντέλου σε ακραίες τιμές (extreme values).

Στη συνέχεια θα αναλύσουμε τους δυο πιο γνωστούς τύπους επαναλαμβανομένων νευρωνικών δικτυών που είναι:

- Δίκτυα μακράς βραχυπρόθεσμης μνήμης (Long-Short Term Memory)
- Τα Φραγμένα Ανατροφοδοτούμενα Δίκτυα (Gated Recurrent Units)

2.5. Δίκτυα μακράς βραχυπρόθεσμης μνήμης (Long-Short Term Memory - LSTM)

Τα μακράς βραχυπρόθεσμης μνήμης δίκτυα είναι ένας τύπος αρχιτεκτονικής επαναλαμβανομένου νευρωνικού δικτύου. Διαθέτουν εξειδικευμένους μηχανισμούς για τον καλύτερο χειρισμό μακροπροθέσμων εξαρτήσεων.

2.5.1. Δομή των δικτύων βραχυπρόθεσμης μνήμης

Η αρχιτεκτονική των δικτύων μακράς βραχυπρόθεσμης μνήμης αποτελείται από κελιά μνήμης που αλληλοεπιδρούν αναμεταξύ τους με περίπλοκο τρόπο με σκοπό να επιτύχουν τη διατήρηση και την ανάκτηση πληροφοριών σε μεγάλες ακολουθίες (long data sequences). Κάθε κελί μνήμης του δικτύου διατηρεί της πληροφορίες αυθαίρετων χρονικών διαστημάτων και είναι εξοπλισμένο με μια εσωτερική κατάσταση που λειτουργεί ως μνήμη (υποστηρίζει λειτουργίες εγγραφής, διαβάσματος, ενημέρωσης και αποθήκευσης δεδομένων) και με έναν αριθμό από αναλογικής φύσης πύλες. Αυτές οι πύλες υλοποιούνται με πολλαπλασιασμό στοιχείων, με συναρτήσεις σιγμοειδούς ή υπερβολικής εφαπτομένης και ρυθμίζουν τη ροή των δεδομένων έξω και μέσα από τα κύτταρα μνήμης.

Μια μεμονωμένη μονάδα μακράς βραχυπρόθεσμης μνήμης αποτελείται από μια κατάσταση κυψέλης και από τρεις πύλες. Παρακάτω ακολουθεί η λειτουργία της:

1. Αρχικοποίηση
   Στην αρχή της επεξεργασίας μιας ακολουθίας η κατάσταση κυψέλης αρχικοποιείται με ένα διάνυσμα μηδενικών ($\mathbf{c}_0$) και συμβολίζει μια κενή μνήμη.
2. Ροή των πληροφοριών
   Η ροή των πληροφοριών από μέσα προς τα έξω γίνεται με τη βοήθεια των πυλών και η κατάσταση κυψέλης στο χρόνο t($\mathbf{c}_0$(t)) επηρεάζεται από τα ακόλουθα :
   - Από την προηγούμενη κατάσταση της κυψέλης ($\mathbf{c}_0$(t-1)), η οποία μεταφέρει πληροφορίες από προηγούμενα χρονικά βήματα και ενημερώνεται σε κάθε χρονικό βήμα με βάση τις λειτουργίες που αφορούν τις πύλες λήθης (forget gate) και εισόδου (input gate).
   - Από την πύλη λήθης (forget gate), που καθορίζει ποιες πληροφορίες από την προηγουμένη κατάσταση της κυψέλης πρέπει να απορριφθούν, να ξεχαστούν ή να διατηρηθούν.
   - Από την πύλη εισόδου που καθορίζει ποιες νέες πληροφορίες πρέπει να αποθηκευτούν στην κατάσταση κυψέλης και την υποψηφία μνήμη που κρατάει τις νέες πληροφορίες, που θα μπορούσαν να προστεθούν στην κατάσταση κυψέλης.

3. Ενημέρωση της κατάστασης κυψέλης.

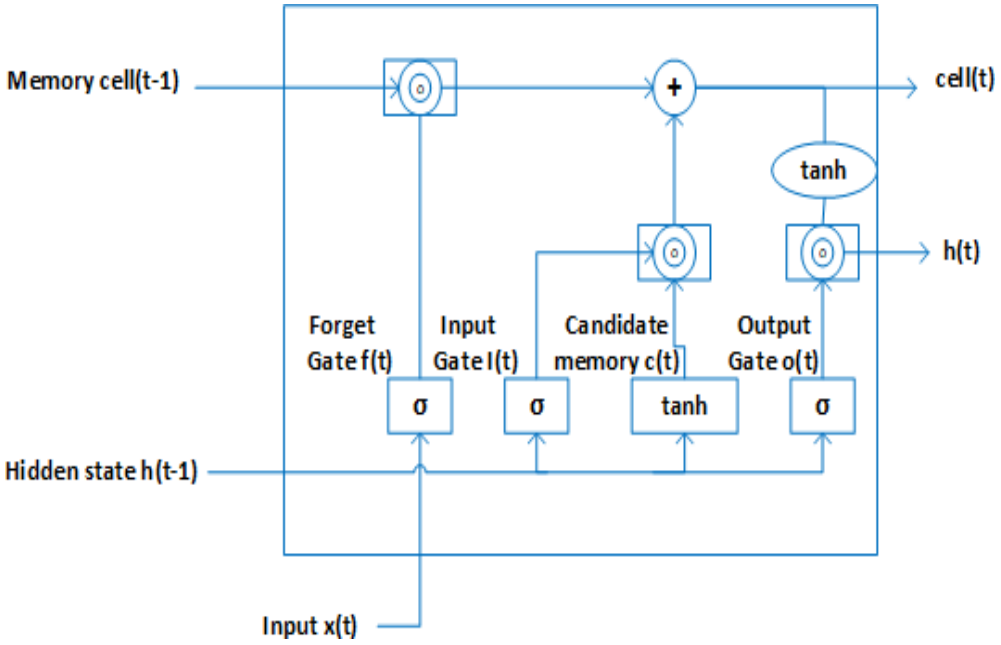
   Η κατάσταση κυψέλης στην τρέχουσα χρονική στιγμή t($\mathbf{c}_0$(t)) ενημερώνεται χρησιμοποιώντας την πύλη λήθης, η οποία αποφασίζει ποιες πληροφορίες θα ξεχάσει και ποιες θα διατηρήσει και την πύλη εισόδου που αποφασίζει ποιες νέες πληροφορίες να προσθέσει από την υποψήφια μνήμη χρησιμοποιώντας τη συνάρτηση ενεργοποίησης Tahn., συνδυάζοντας την παλιά μνήμη με τις νέες πληροφορίες.
4. Επόμενη φάση
   Η πύλη εξόδου είναι αυτή που θα καθορίσει ποια μέρη της κατάστασης κυψέλης θα διαβαστούν και θα περάσουν στην επόμενη κρυφή κατάσταση.
5. Τελική κρυφή κατάσταση.
   Η τελική κρυφή κατάσταση υπολογίζεται εφαρμόζοντας την πύλη εξόδου πάνω από την κατάσταση κυψέλης στη χρονική στιγμή t. Αυτή η κρυφή κατάσταση μπορεί να χρησιμοποιηθεί για προβλέψεις ή μπορεί να μεταφερθεί στο επόμενο χρονικό βήμα.

Οι πύλες μιας μεμονωμένης μονάδας LSTM είναι τρεις στον αριθμό και είναι οι ακόλουθες:

Πύλη εισόδου (Input Gate) - Λόγω της σιγμοειδούς συνάρτησης ενεργοποίησης οι τιμές της πύλης εισόδου βρίσκονται στο εύρος των (0 ,1) και βάσει αυτών των τιμών η πύλη εισόδου καθορίζει πόση από την πληροφορία που υπάρχει στην υποψήφια μνήμη θα προστεθεί στην κατάσταση κυψέλης.

Πύλη λήθης (Forget Gate) - Λαμβάνει μια τιμή στο εύρος των τιμών (0,1) για κάθε πληροφορία που βρίσκεται στη μνήμη και αν η τιμή είναι 0 τότε την «ξεχνάει εντελώς» ενώ αν η τιμή ισούται με 1 τότε διατηρεί την πληροφορία. Ο υπολογισμός πίσω από τη forget gate πραγματοποιείται χρησιμοποιώντας τη σιγμοειδή συνάρτηση ενεργοποίησης.

Πύλη εξόδου (Output Gate) - Οι τιμές που λαμβάνει κυμαίνονται στο ίδιο εύρος τιμών (0,1), καθώς χρησιμοποιεί τη σιγμοειδή συνάρτηση καθορίζοντας εάν η κατάσταση κυψέλης πρέπει να επηρεάσει την έξοδο στο τρέχον χρονικό βήμα. Στο παρακάτω σχήμα (Σχήμα 2.5.1.α) διακρίνεται η αρχιτεκτονική ενός νευρωνικού δικτύου LSTM.



Σχήμα 2.5.1.α: Αρχιτεκτονική LSTM.

2.5.2. Αλγόριθμος εκπαίδευσης

Τα δίκτυα LSTM κατά το εμπρόσθιο πέρασμα (Forward Pass) επεξεργάζονται την ακολουθία εισόδου υπολογίζοντας τις προβλέψεις και τη συνάρτηση απώλειας, η οποία μετρά την απόκλιση μεταξύ των προβλεπόμενων τιμών και των αναμενομένων τιμών. Κατά την οπίσθια διάδοση οι διαβαθμίσεις υπολογίζονται χρησιμοποιώντας τον κανόνα της αλυσίδας (Chain Rule), που σημαίνει ότι το σφάλμα δικτύου σε κάποιο δεδομένο χρονικό βήμα επηρεάζεται όχι μόνο από το τρέχον βήμα αλλά και από μελλοντικά χρονικά βήματα. Οι διαβαθμίσεις χρησιμοποιούνται για την ενημέρωση των βαρών του δικτύου και η προσαρμογή του κάθε βάρους με βάση τις υπολογιστικές κλίσεις καθορίζεται από τον αλγόριθμο βελτιστοποίησης (π.χ. SGD). Η συγκεκριμένη διαδικασία επαναλαμβάνεται για πολλές εποχές, επιτρέποντας στο δίκτυο να κάνει καλύτερες προβλέψεις.

Τα δίκτυα μακράς βραχυπρόθεσμης μνήμης αντί να επεξεργάζονται ένα στοιχείο κάθε φορά έχουν τη δυνατότητα να επεξεργαστούν μικρά σύνολα δεδομένων (Mini-Batches) χρησιμοποιώντας τον αλγόριθμο βελτιστοποίησης Mini-Batch Gradient Descendent για πιο αποτελεσματικό υπολογισμό των κλίσεων του δικτύου και ταχύτερη σύγκλιση. Επίσης αν είναι απαραίτητο εφαρμόζονται τεχνικές κανονικοποίησης για την αποφυγή της υπερβολικής προσαρμογής του δικτύου, η οποία συμβαίνει όταν το μοντέλο μαθαίνει πολύ καλά τα δεδομένα εκπαίδευσης και αποτυγχάνει να γενικεύσει σε νέα δεδομένα.

Τα βήματα που πραγματοποίει ο αλγόριθμος είναι:

1. Αρχικοποίηση παραμέτρων.
   Σε αυτό το βήμα αρχικοποιούνται τα βάρη τόσο της εισόδου όσο και της κρυφής κατάστασης καθώς και οι σταθεροί όροι (biases) για τις πύλες εισόδου, λήθης και εξόδου.
2. Αρχικοποίηση της κατάστασης κυψέλης και της κρυφής κατάστασης.

Η κρυφή κατάσταση και η κατάσταση μνήμης (κυψέλης) αρχικοποιούνται με διανύσματα μηδενικών.

3. Επανάληψη μέσω χρόνου.
   Για κάθε χρονικό βήμα t που πραγματοποιεί ο αλγόριθμος εισάγοντας τα στοιχεία της ακολουθίας γίνονται τα παρακάτω:
   - Η ενεργοποίησης της πύλης εισόδου
   - Ενεργοποιείται η πύλη λήθης και η υποψήφια μνήμη
   - Ενημερώνεται η κατάσταση της κυψέλης
   - Ενεργοποιείται η πύλη εξόδου
   - Υπολογίζεται η νέα κρυφή κατάσταση

4. Εάν το LSTM πραγματοποιεί μια συγκεκριμένη εργασία, π.χ. πρόβλεψη ακολουθίας, τα αποτελέσματα περνάνε στην τελική κρυφή κατάσταση και από εκεί στην έξοδο.
5. Backpropagation Trough Time (Οπισθοδιάδοση στο χρόνο)
   Υπολογίζεται η απώλεια του μοντέλου μεταξύ της προβλεπόμενης εξόδου και του πραγματικού στόχου. Η απώλεια χρησιμοποιείται για τον υπολογισμό των διαβαθμίσεων.
6. Ενημέρωση των βαρών και των προκαταλήψεων του δικτύου.
   Ενημερώνονται τα βάρη και οι σταθεροί όροι του δικτύου με βάση τις υπολογισμένες διαβαθμίσεις.
7. Επανάληψη.
   Τα βήματα 3 έως 6 επαναλαμβάνονται για έναν καθορισμένο αριθμό επαναλήψεων ή μέχρι να ικανοποιηθεί ένα κριτήριο διακοπής.
8. Mini-Batch εκπαίδευση και κανονικοποίηση (προαιρετικό).
   Η εκπαίδευση μπορεί να εφαρμοστεί πάνω σε μικρά σύνολα δεδομένων της ακολουθίας για πιο αποτελεσματικό υπολογισμό των κλίσεων, καθώς γίνεται η εφαρμογή τεχνικών κανονικοποίησης για την αποφυγή της υπερβολικής προσαρμογής (overfitting).
9. Επικύρωση και δοκιμή
   Το μοντέλο αξιολογείται ως προς την απόδοση του, χρησιμοποιώντας ένα σετ δεδομένων επικύρωσης για να διασφαλιστεί ότι γενικεύει καλά σε άγνωστη ακολουθία.

Οι συναρτήσεις ενεργοποίησης, ο αλγόριθμος βελτιστοποίησης και οι υπερπαράμετροι, που συνήθως επιλέγονται, διαφέρουν από ένα μοντέλο σε ένα άλλο ανάλογα με την εργασία που καλείται να εκτελέσει και το σύνολο των δεδομένων.

2.5.3. Μαθηματική αναπαράσταση του LSTM

Οι μαθηματικές εξισώσεις που διέπουν τους υπολογισμούς σε ένα κελί μνήμης είναι οι εξής:

- Η εξίσωση της πύλης εισόδου(Input gate)

$$i(t) = \sigma(W_i[h(t-1), X(t)] + b_i)$$

  Όπου $i(t)$ η πύλη εισόδου, που ενεργοποιείται από τη συνάρτηση σιγμοειδούς σ, αφού έχει προηγηθεί το σταθμισμένο άθροισμα των διανυσμάτων W_i, το οποίο περιλαμβάνει την τρέχουσα κατάσταση X(t), την προηγούμενη

κατάσταση h(t-1), δηλαδή την ακολουθία εισόδου στο χρονικό βήμα t, ενώ στο τέλος προστίθεται και το διάνυσμα με τους σταθερούς όρους b_i.

- Η εξίσωση της πύλης λήθης (Forget gate)

$$f(t) = \sigma(W_f[h(t-1), X(t)] + b_f)$$

Όπου $f(t)$ η πύλη λήθης που ενεργοποιείται από τη συνάρτηση σιγμοειδούς σ, αφού έχει προηγηθεί το σταθμισμένο άθροισμα των διανυσμάτων W_f, το οποίο περιλαμβάνει την τρέχουσα κατάσταση X(t), την προηγούμενη κατάσταση h(t-1), δηλαδή την ακολουθία εισόδου στο χρονικό βήμα t, ενώ στο τέλος προστίθεται και το διάνυσμα με τους σταθερούς όρους b_f.

- Η εξίσωση της υποψήφιας μνήμης

$$c(t) = tahn(W_c[h(t-1), X(t)] + b_c)$$

Η κατάσταση της υποψήφιας μνήμης $c(t)$ ενεργοποιείται από τη συνάρτηση Than, αφού έχει προηγηθεί το σταθμισμένο άθροισμα των βαρών της υποψήφιας μνήμης W_c με τα διανύσματα εισόδου της προηγουμένης κατάστασης h(t-1) και της τρέχουσας X(t), ενώ στο τέλος προστίθεται το διάνυσμα που περιλαμβάνει τους σταθερούς όρους b_c.

- Η ενημέρωση της κατάστασης κυψέλης υπολογίζεται με την παρακάτω εξίσωση, χρησιμοποιώντας το αποτέλεσμα της εξίσωσης της πύλης λήθης $f(t)$ και το αποτέλεσμα της υποψήφιας μνήμης c(t), λαμβάνοντας υπόψη τόσο την προηγούμενη κατάσταση της κυψέλης cell(t-1) όσο και την είσοδο του χρονικού βήματος t, i(t).

$$cell(t) = f(t)cell(t-1) + i(t)c(t)$$

- Η εξίσωση της κρυφής κατάστασης

$$h(t) = o(t)\ tahn(c(t))$$

h(t) αντιπροσωπεύει την κρυφή κατάσταση στο χρονικό βήμα t, o(t) το αποτέλεσμα της πύλης εξόδου, c(t) είναι η υποψήφια μνήμη στο χρονικό βήμα t που ενεργοποιείται από τη συνάρτηση Tahn.

- Ο υπολογισμός της πύλης εξόδου δίνεται από την εξίσωση:

$$o(t) = \sigma(W_o[h(t-1), X(t)] + b_o)$$

όπου o(t) είναι η πύλη εξόδου, W_o είναι το διάνυσμα των βαρών της πύλης εξόδου, h(t-1) είναι το διάνυσμα προηγουμένης κατάστασης, X(t) αντιπροσωπεύει το διάνυσμα της τρέχουσας κατάστασης, b_o το διάνυσμα που περιλαμβάνει τους σταθερούς όρους (biases) της πύλης εξόδου και σ είναι η σιγμοειδούς συνάρτηση ενεργοποίησης.

2.5.4. Μειονεκτήματα των LSTM

Τα δίκτυα μακράς βραχυπρόθεσμης μνήμης είναι ισχυρά μοντέλα και χρησιμοποιούνται ευρέως για διαδοχική επεξεργασία δεδομένων. Ωστόσο έχουν ορισμένα μειονεκτήματα.

Η υπολογιστική πολυπλοκότητα λόγω των πολλών παραμέτρων σε πολύ μεγάλες ακολουθίες δημιουργεί δυσκολίες κατά την εκπαίδευση και ένα αργό μοντέλο. Τα δίκτυα αυτά έχουν πολύπλοκή δομή και είναι δύσκολο να κατανοηθεί η εσωτερική λειτουργία τους. Λόγω της διαδοχικής φύσης τους παρουσιάζουν περιορισμένη παραλληλοποίηση των υπολογισμών στα χρονικά βήματα, που μπορεί να οδηγήσει σε αργούς ρυθμούς εκπαίδευσης. Η χρήση τους περιορίζεται σε συσκευές με ελάχιστη μνήμη, είναι ευαίσθητα στις αρχικές τιμές των βαρών και των προκαταλήψεων. Η κακή αρχικοποίηση μπορεί να οδηγήσει σε αργή σύγκληση και σε μη βέλτιστες λύσεις. Επίσης τα μοντέλα LSTM παρόλο που σχεδιάστηκαν να αντιμετωπίσουν τις μακροπρόθεσμες εξαρτήσεις εξακολουθούν να έχουν αυτό το πρόβλημα σε εξαιρετικά μεγάλες ακολουθίες.

2.6. Περιφραγμένη Επαναλαμβανόμενη Μονάδα (Gated Recurrent Unit)

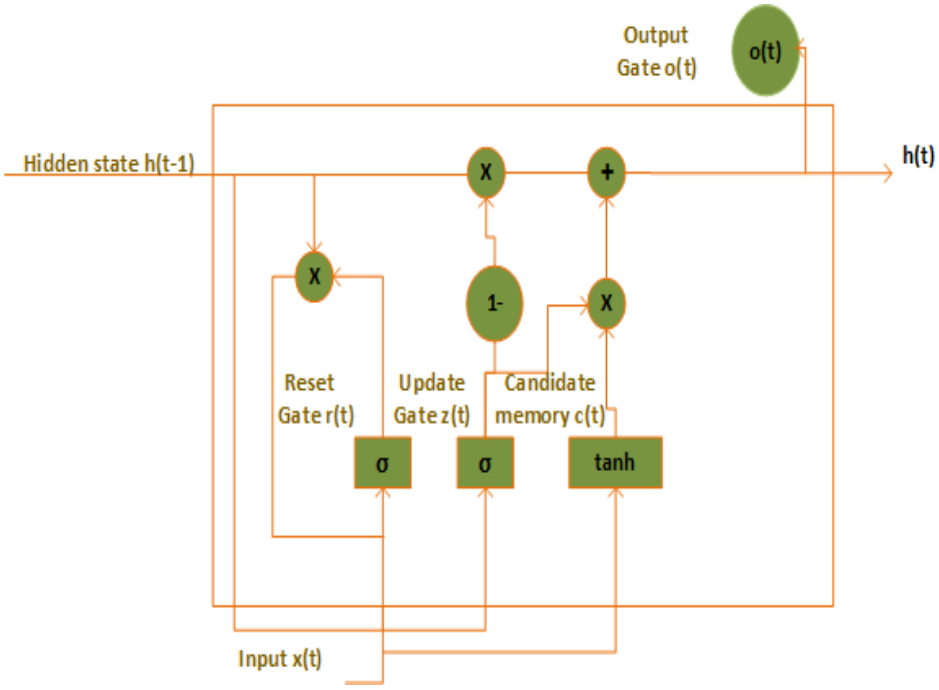
Η περιφραγμένη επαναλαμβανομένη μονάδα είναι μια παραλλαγή του επαναλαμβανομένου νευρωνικού δικτύου (RNN), που έχει αναπτυχθεί με σκοπό την αντιμετώπιση των περιορισμών του απλού RNN. Χρησιμοποιεί τις πύλες ενημέρωσης και επαναφοράς, επιτρέποντας στο δίκτυο να έχει δυνατότητες βραχυπρόθεσμης και μακροπρόθεσμης μνήμης καθώς και τη δυνατότητα να καταγράφει πιο αποτελεσματικά τις εξαρτήσεις μεγάλης εμβέλειας σε διαδοχικά δεδομένα. Διαθέτει ένα μηχανισμό πύλης που ελέγχει τη ροή των δεδομένων στο δίκτυο και τις κυψέλες μνήμης που διατηρούν μία κρυφή κατάσταση, η οποία ενημερώνεται σε κάθε χρονικό βήμα με βάση την είσοδο και την προηγουμένη κατάσταση.

2.6.1. Αρχιτεκτονική μιας Περιφραγμένης Επαναλαμβανόμενης Μονάδας (GRU)

Η δομή μιας περιφραγμένης επαναλαμβανόμενης μονάδας διαθέτει κελιά μνήμης που ενσωματώνουν ένα μηχανισμό πύλης. Ένα μεμονωμένο κελί αποτελείται από τα ακόλουθα στοιχεία:

1. Μηχανισμός Πύλης
   Η Περιφραγμένη Επαναλαμβανόμενη Μονάδα διαθέτει μηχανισμό πύλης που ελέγχει τη ροή των πληροφοριών μέσω του δικτύου. Αυτός ο μηχανισμός περιλαμβάνει μια πύλη ενημέρωσης και μια πύλη επαναφοράς.
   Η πύλη ενημέρωσης αποφασίζει πόσες από τις πληροφορίες της προηγουμένης κατάστασης πρέπει να ενημερωθούν και να διατηρηθούν κατά το τρέχον χρονικό βήμα και η πύλη επαναφοράς καθορίζει ποιες από τις προηγούμενες πληροφορίες πρέπει να ξεχαστούν, να απορριφθούν ή να επαναφέρονται.
2. Κύτταρα Μνήμης, τα οποία υπολογίζουν το τρέχον περιεχόμενο μνήμης συνδυάζοντας την προηγούμενη κρυφή κατάσταση με τη νέα υποψήφια κρυφή κατάσταση, η οποία στην πραγματικότητα είναι μια νέα έκδοση της τρέχουσας εισόδου.

   Ένα μοντέλο GRU αποτελείται από πολλά κελιά μνήμης, το καθένα από τα οποία επεξεργάζεται ένα χρονικό βήμα της ακολουθίας εισόδου, επιτρέποντας στο μοντέλο να ενημερώνει και να επαναφέρει επιλεκτικά την κρυφή του κατάσταση και να καταγράφει μακροπρόθεσμες εξαρτήσεις σε διαδοχικά δεδομένα. Στο παρακάτω σχήμα (Σχήμα 2.6.1.α.) διακρίνεται η αρχιτεκτονική ενός νευρωνικού δικτύου GRU.



Σχήμα 2.6.1.α: Αρχιτεκτονική GRU.

2.6.2. Μαθηματικό υπόβαθρο

Ένα κελί μιας GRU πραγματοποιεί τις ακόλουθες μαθηματικές πράξεις:

1. Η πύλη ενημέρωσης δέχεται στην είσοδο την προηγουμένη κρυφή κατάσταση και την τρέχουσα είσοδο και επιστρέφει στην έξοδό της μια τιμή στο εύρος των τιμών 0 και 1, που αντιπροσωπεύει την αναλογία πληροφοριών προς ενημέρωση ή προς διατήρηση για το τρέχον βήμα και υπολογίζεται από την ακόλουθη εξίσωση:

$$z(t) = \sigma(W_z[h(t-1), X(t)])$$

Όπου z(t) είναι η έξοδος της πύλης ενημέρωσης, το W_z είναι ο πίνακας των βαρών της πύλης ενημέρωσης, h(t-1) είναι η προηγουμένη κατάσταση του κελιού μνήμης και X(t) η τρέχουσα είσοδος.

2. Η πύλη επαναφοράς επιστρέφει μια τιμή μεταξύ 0 και 1, που αντιπροσωπεύει το ποσοστό των πληροφοριών, που είτε πρέπει να γίνει η επαναφορά τους είτε να «ξεχαστούν» από την προηγουμένη κρυφή κατάσταση. Υπολογίζεται κάνοντας επεξεργασία της προηγουμένης κατάστασης και της τρέχουσας εισόδου και δίνεται από την ακόλουθη μαθηματική σχέση:

$$r(t) = \sigma(W_r[h(t-1), X(t)])$$

Όπου r(t) είναι η έξοδος της πύλης επαναφοράς, το W_r είναι το διάνυσμα των βαρών της πύλης επαναφοράς, h(t-1) είναι η προηγουμένη κατάσταση του κελιού μνήμης και X(t) η τρέχουσα είσοδος.

3. Το τρέχον περιεχόμενο μνήμης παράγει μια υποψήφια κρυφή κατάσταση h(t). Υπολογίζεται μεταξύ του διανύσματος της προηγουμένης κατάστασης h_{t-1}, της τιμής εξόδου της πύλης επαναφοράς r(t) και του διανύσματος της τρέχουσας εισόδου X() και ενεργοποιείται από τη συνάρτηση Tanh.

$$h(t) = \tanh(W[r(t) \odot h(t-1), X(t)])$$

όπου ⊙ δηλώνει πολλαπλασιασμό των αντίστοιχων στοιχείων σε δυο πίνακες ή διανύσματα.

4. Η τρέχουσα κρυφή κατάσταση h`(t) υπολογίζεται χρησιμοποιώντας την προηγουμένη κατάσταση h(t-1), την έξοδο της πύλης ενημέρωσης z(t) και την υποψήφια κρυφή κατάσταση h(t).

$$h`(t) = (1 - z(t)) \odot h(t-1) + z(t) \odot h(t)$$

όπου ⊙ δηλώνει πολλαπλασιασμό των αντίστοιχων στοιχείων σε δύο πίνακες η διανύσματα.

2.6.3. Αλγόριθμος εκπαίδευσης

Ο παρακάτω αλγόριθμος αποτελεί ένα γενικό πλαίσιο για την εκπαίδευση των GRU και των υλοποιήσεων των μοντέλων αυτών και μάλιστα διαφοροποιείται ανάλογα με το πλαίσιο βαθιάς μάθησης που χρησιμοποιείται (Matlab, TensorFlow, PyTorch). Μπορεί να εφαρμοστεί σε παρτίδες ή σε πολλαπλές ακολουθίες, οι οποίες γίνονται αντικείμενο επεξεργασίας ταυτόχρονα για βελτίωση της αποκλειστικότητας της εκπαίδευσης. Εφαρμόζονται τεχνικές για την αποφυγή της υπερβολικής προσαρμογής (dropout, regularization) και μέθοδοι (learning rate schedules), που προσαρμόζουν το ρυθμό εκμάθησης κατά τη διάρκεια της εκπαίδευσης. Η εκπαίδευση της επαναλαμβανόμενης μονάδας GRU γίνεται χρησιμοποιώντας τον αλγόριθμο οπισθοδρόμησης μέσω του χρόνου (backpropagation). Ακολουθεί η ανάλυση των βημάτων που πραγματοποιεί ο αλγόριθμος εκπαίδευσης GRU:

1. **Αρχικοποίηση**
   Αρχικοποιούνται όλα τα βάρη και οι σταθεροί όροι του μοντέλου, επιλέγεται ένας αλγόριθμος βελτιστοποίησης και ορίζονται υπερπαράμετροι.
2. **Εμπρόσθιο βήμα (Forward Pass)**
   Για κάθε χρονικό βήμα t που θα πραγματοποιήσει ο αλγόριθμος υπολογίζεται η πύλη ενημέρωσης, η πύλη επαναφοράς, η υποψήφια κρυφή κατάσταση και ενημερώνεται η τρέχουσα κρυφή κατάσταση.
3. **Υπολογισμός απώλειας**
   Υπολογίζεται η απώλεια μεταξύ των προβλεπόμενων αποτελεσμάτων της ακολουθίας και των πραγματικών στόχων χρησιμοποιώντας την κατάλληλη συνάρτηση απώλειας (π.χ. μέσο τετραγωνικό σφάλμα).
4. **Οπίσθια διάδοση (Backpropagation Through Time-BPTT)**
   Σε αυτό το βήμα υπολογίζονται οι διαβαθμίσεις της απώλειας του μοντέλου, εφαρμόζεται η τεχνική αποκοπής κλίσης (gradient clipping), η οποία χρησιμοποιείται για την πρόληψη του προβλήματος των λεγόμενων «εκρηκτικών κλίσεων» (gradients exploding) και ενημερώνονται οι παράμετροι του μοντέλου χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης και τις υπολογισμένες κλίσεις.
5. **Επανάληψη βημάτων**
   Τα βήματα 2-4 επαναλαμβάνονται για έναν καθορισμένο αριθμό εποχών ή μέχρι τη σύγκλιση.
6. **Αξιολόγηση μοντέλου**

Το μοντέλο αξιολογείται σε ένα σύνολο δεδομένων επικύρωσης και δοκιμής καθώς παρακολουθείται η απόδοση του.

2.6.4. Περιορισμοί των GRU

Τα δίκτυα GRU έχουν σχεδιαστεί για να καταγράφουν εξαρτήσεις μεγάλης εμβέλειας σε διαδοχικά δεδομένα και για να αντιμετωπίσουν αποτελεσματικά ορισμένα από τα προβλήματα που σχετίζονται με τα παραδοσιακά RNN. Παρόλα αυτά παρουσιάζουν κάποιους περιορισμούς όπως δυσκολία στην παράλληλη επεξεργασία των δεδομένων, η οποία μπορεί να οδηγήσει σε πιο αργούς χρόνους εκπαίδευσης. Η απόδοση των δικτύων αυτών μπορεί να εξαρτηθεί από την επιλογή των υπερπαραμέτρων και η εύρεση των σωστών υπερπαραμέτρων μπορεί να απαιτεί πειραματισμό.

ΚΕΦΑΛΑΙΟ 3

3. ΥΛΟΠΟΙΗΣΗ ΜΟΝΤΕΛΩΝ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ

Στα πλαίσια της υλοποίησης των μοντέλων Βαθιάς Μάθησης θα χρησιμοποιήσουμε τις δυνατότητες του Matlab, που μέσω του περιβάλλοντός του και της βιβλιοθήκης της Βαθιάς Μάθησης παρέχει ένα ολοκληρωμένο σύνολο εργαλείων για το σχεδιασμό και την υλοποίηση βαθιών νευρωνικών δικτύων. Η εργαλειοθήκη Deep Learning του Matlab επιτρέπει τη δημιουργία, την εκπαίδευση και την ανάπτυξη βαθιών νευρωνικών δικτύων, υποστηρίζοντας διάφορους τύπους αρχιτεκτονικών νευρωνικών δικτύων, συμπεριλαμβανομένων των συνελικτικών (CNN) και επαναλαμβανόμενων (RNN) δικτύων. Η σχεδίαση των νευρωνικών δικτύων μπορεί να γίνει χρησιμοποιώντας μια γραφική διεπαφή χρήστη (Neural Network Designer), ή μέσω προγραμματισμού, χρησιμοποιώντας κώδικα Matlab. Αυτό περιλαμβάνει τον καθορισμό επιπέδων, των καθορισμό τεχνικών συνδέσεων που θα χρησιμοποιηθούν, καθώς και τη διαμόρφωση παραμέτρων δικτύου. Επίσης το Matlab υποστηρίζει την εκμάθηση μεταφοράς (transfer learning), επιτρέποντας να χρησιμοποιούμε προ εκπαιδευμένα μοντέλα και να τα προσαρμόζουμε ανάλογα με τις ανάγκες της καινούργιας εργασίας.

Τα συγκεκριμένα μοντέλα θα εφαρμόζονται στην πρόβλεψη και στην ταξινόμηση χρονοσειρών με τη βοήθεια των αναδρομικών και συνεκτικών τεχνιτών νευρωνικών δικτύων. Για την εκπαίδευση καθώς και για την επικύρωση των μοντέλων της βαθιάς μάθησης έχουν επιλεχθεί δυο σύνολα δεδομένων (dataset), το ένα από το TensorFlow, που περιλαμβάνει λουλούδια, ενώ το δεύτερο σύνολο δεδομένων (dataset) από την ιστοσελίδα της kaggle και συγκεκριμένα επιλέχθηκαν τρία σύνολα διαφορετικών κρυπτονομισμάτων.

3.1. Ταξινόμηση εικόνων με συνελικτικό νευρωνικό δίκτυο (Image Classification with CNN)

Η ταξινόμηση εικόνων είναι μια εργασία της βαθιάς μάθησης και έχει ένα ευρύ φάσμα εφαρμογών, όπως στην αναγνώριση προσώπου, στην ανίχνευση αντικειμένων, στα αυτόνομα οχήματα, στην ιατρική απεικόνιση, στην κατηγοριοποίηση εικόνων εισόδου με μια ή περισσότερες προκαθορισμένες κλάσεις ή ετικέτες.

Τα CNN είναι ένας τύπος αρχιτεκτονικής βαθιάς νευρωνικού δικτύου που έχουν σχεδιαστεί ειδικά για την επεξεργασία δεδομένων που μοιάζουν με πλέγμα, όπως είναι για παράδειγμα οι εικόνες. Αποτελούνται από στρώματα όπως:

- Το στρώμα εισόδου (Input Layer), που δέχεται τις μη επεξεργασμένες εικόνες, με κάθε νευρώνα εισόδου να αντιστοιχεί σε ένα pixel.
- Τα συνελικτικά στρώματα (Convolutional Layers), τα οποία είναι υπεύθυνα για την εκμάθηση χωρικών ιεραρχιών των χαρακτηριστικών και εφαρμόζουν φίλτρα στην εικόνα εισόδου για να ανιχνεύσουν μοτίβα όπως άκρες, υφές ή πιο συνθέτες δομές και περιλαμβάνουν τις συνελικτικές λειτουργίες, όπως την ολίσθηση του φίλτρου πάνω από την εικόνα εισόδου και τον υπολογισμό του γινομένου των κουκκίδων σε κάθε θέση.

- Τα επίπεδα συγκέντρωσης (Pooling Layers), που μειώνουν τις διαστάσεις των χαρτών των χαρακτηριστικών, συμβάλλοντας στη μείωση του υπολογιστικού φόρτου και του κινδύνου υπερπροσαρμογής του μοντέλου.
- Τα στρώματα κανονικοποίησης (Normalization Layers) βοηθούν στην σταθεροποίηση και την επιτάχυνση της προπονητικής διαδικασίας, μειώνοντας την μετατόπιση της εσωτερικής μεταβλητής (reducing internal covariate shift).
- Τα στρώματα εγκατάλειψης (Dropout Layers) ρίχνουν τυχαία ένα κλάσμα των νευρώνων κατά τη διάρκεια της προπόνησης. Αυτή η τεχνική βοηθά στην αποφυγή της υπερβολικής προσαρμογής, προωθώντας την εκμάθηση πιο σύνθετων χαρακτηριστικών.
- Το επίπεδο στρώμα (Flattening Layer), στο οποίο οι διαστάσεις των χαρτών των χαρακτηριστικών μετατρέπονται σε ένα μονοδιάστατο διάνυσμα και τα δεδομένα τροφοδοτούνται στα πλήρως συνδεδεμένα στρώματα.
- Τα στρώματα παράληψης (Skip Connection Layers) επιτρέπουν τις διαβαθμίσεις των μερικών παραγώγων απευθείας σε πολλαπλά επίπεδα. Αυτό βοηθά στην αντιμετώπιση του προβλήματος της κλίσης που εξαφανίζεται και διευκολύνει την εκπαίδευση των βαθιών δικτύων.
- Τα πλήρως συνδεδεμένα στρώματα (Fully Connected Layers) συνδέουν κάθε νευρώνα ενός στρώματος με κάθε νευρώνα του επόμενου στρώματος. Βρίσκονται συνηθώς στο τέλος της αρχιτεκτονικής και ευθύνονται για την πρόβλεψη του μοντέλου με βάση τα μαθημένα χαρακτηριστικά.
- Το στρώμα εξόδου (Output Layer) παράγει τις τελικές προβλέψεις του δικτύου. Ο αριθμός των νευρώνων σε αυτό το στρώμα αντιστοιχεί στον αριθμό των κλάσεων στην εργασία ταξινόμησης.

Αυτά τα στρώματα συνεργάζονται με ιεραρχικό τρόπο, με κάθε στρώμα να μαθαίνει διαφορετικά επίπεδα αφαίρεσης από τα δεδομένα εισόδου. Ο συνδυασμός συνελικτικών ομαδοποιημένων και πλήρως συνδεδεμένων επίπεδων επιτρέπει στα CNN να μαθαίνουν αυτόματα και να εξάγουν σημαντικές λειτουργίες για εργασίες ταξινόμησης εικόνων.

Για να κατασκευαστεί ένα συνελικτικό νευρωνικό δίκτυο για ταξινόμηση εικόνων θα πρέπει πρώτα να εξετάζουμε το σύνολο των δεδομένων μας διότι με βάση τα χαρακτηριστικά που έχουν τα δεδομένα μας θα μπορέσουμε να σχεδιάσουμε τη βασική αρχιτεκτονική του συνελικτικού νευρωνικού δικτύου.

3.1.1. Το Σύνολο των Δεδομένων (The Data Set)

Το σύνολο των δεδομένων αποτελείται από 3670 εικόνες, οι οποίες περιλαμβάνουν λουλούδια με διαφορετική ανάλυση megapixel με κάθε εικόνα να κυμαίνεται σε ένα εύρος από 0.025, 0.04 έως 0.1 μοιρασμένες σε πέντε κλάσεις ως εξής:

- Η πρώτη κλάση αποτελείται από 633 εικόνες οι οποίες περιλαμβάνουν μαργαρίτες (daisy).
- Η δεύτερη κλάση περιλαμβάνει 898 εικόνες με πικραλίδες (dandelion).
- Η τρίτη κλάση περιέχει 641 εικόνες με τριαντάφυλλα (roses).
- Η τέταρτη κλάση εμπεριέχει 699 εικόνες με ηλίανθους (sunflowers).
- Και η πέμπτη κλάση περιλαμβάνει 799 εικόνες με τουλίπες (tulips).

Πίνακας 3.1.1.α: Παρουσίαση του συνόλου δεδομένων.

	Label	Count
1	daisy	633
2	dandelion	898
3	roses	641
4	sunflowers	699
5	tulips	799

3.1.2. Διαχωρισμός του Συνόλου Δεδομένων (Data Set Split)

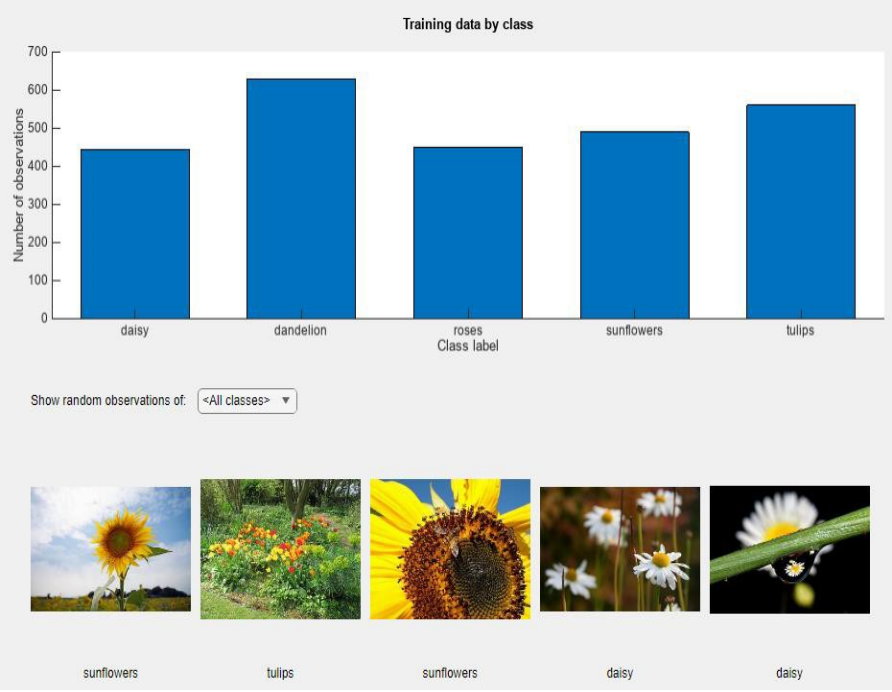
Εφόσον γνωρίζουμε το σύνολο των δεδομένων μας μπορούμε να ξεκινήσουμε να χτίσουμε την αρχιτεκτονική μας και σαν αρχικό βήμα καλούμαστε να προεπεξεργαστούμε τις εικόνες του συνόλου δεδομένων, να αλλάξουμε το μέγεθος των εικόνων, αν απαιτείται, χωρίς να οδηγούμαστε σε απώλεια πληροφοριών. Στη συνέχεια θα διαχωρίσουμε το σύνολο μας σε τρία υποσύνολα: σύνολο εκπαίδευσης (training data set), σύνολο επικύρωσης (validation data set) και σύνολο δοκιμής (test data set). Με τη χρήση κατάλληλων συναρτήσεων του λογισμικού μοιράσαμε τυχαία τις εικόνες ως εξής:

-το 80% των εικόνων του συνόλου μας καταχωρήθηκαν στο υποσύνολο εκπαίδευσης.

-το 10% των εικόνων στο υποσύνολο επικύρωσης.

-το 10% των εικόνων στο υποσύνολο δοκιμής.

Στις ακόλουθες εικόνες και στους παρακάτω πίνακες παρουσιάζονται τα τρία υποσύνολά μας όπως αυτά έχουν διαμορφωθεί.



Εικόνα 3.1.2.α: Υποσύνολο εκπαίδευσης της CNN αρχιτεκτονικής.

Πίνακας 3.1.2.α: Τυχαία διαμόρφωση του υποσυνόλου εκπαίδευσης .

TrainingImages_Labels = 5×2 table

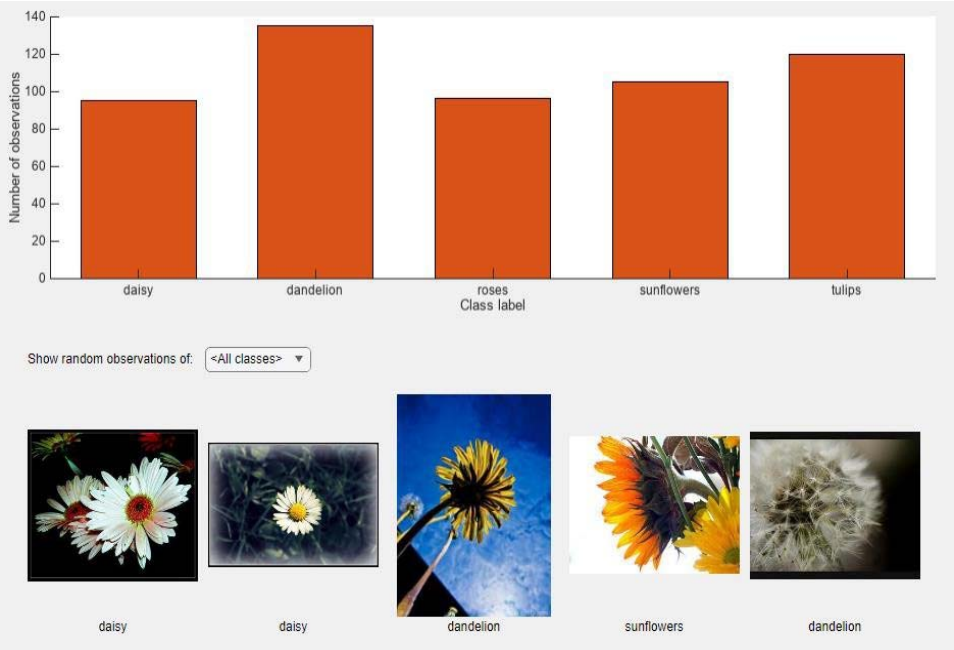
	Label	Count
1	daisy	506
2	dandelion	718
3	roses	513
4	sunflowers	559
5	tulips	639

Εικόνα 3.1.2.β: Υποσύνολο επικύρωσης του CNN μοντέλου.

Πίνακας 3.1.2.β: Τυχαία διαμόρφωση του υποσυνόλου επικύρωσης.

ValidationImages_Labels = 5×2 table

	Label	Count
1	daisy	64
2	dandelion	90
3	roses	64
4	sunflowers	70
5	tulips	80



Εικόνα 3.1.2.γ: Το υποσύνολο δοκιμής της CNN αρχιτεκτονικής.

Πίνακας 3.1.2.γ: Τυχαία διαμόρφωση του υποσυνόλου δοκιμής.

TestImages_Labels = 5×2 table

	Label	Count
1	daisy	63
2	dandelion	90
3	roses	64
4	sunflowers	70
5	tulips	80

3.1.3. Σχεδίαση της CNN αρχιτεκτονικής και επιλογή των ρυθμίσεων εκπαίδευσης (The CNN Architecture and the training options)

Αφού χωρίσαμε το σύνολο των δεδομένων σε δεδομένα εκπαίδευσης, δεδομένα επικύρωσης και δεδομένα δοκιμής μπορούμε να ξεκινήσουμε τη σχεδίαση του συνελικτικού νευρωνικού δικτύου και να προσθέσουμε τα επίπεδα του μοντέλου. Θα σχεδιάσουμε μια αρχιτεκτονική που θα αποτελείται από τα ακόλουθα επίπεδα (layers):

1. Επίπεδο εισόδου (Input Layer) - αναμένει εικόνες RGB 256 x 256 pixel με κανονικοποίηση z-score (z-score Normalization.).
2. Συνελικτικά επίπεδα (2-D Convolutional Layers) - σε κάθε συνελικτικό επίπεδο της CNN αρχιτεκτονικής εφαρμόζεται διαφορετικός αριθμός συνελικτικών φίλτρων και ίδια μεγέθη παραθύρου στις εικόνες εισόδου.
3. Επίπεδα κανονικοποίησης παρτίδας (Batch Normalization Layers) - βοηθά στη σταθεροποίηση και την επιτάχυνση της εκπαίδευσης.
4. ReLU συναρτήσεις - οι συναρτήσεις ενεργοποίησης εισάγουν τη μη γραμμικότητα στο δίκτυο, εφαρμόζοντας μια συνάρτηση στην έξοδο του προηγούμενου επίπεδου.
5. Επίπεδα Συγκέντρωσης (2-D Max Pooling Layers) - εκτελούν μέγιστη συγκέντρωση με μέγεθος παραθύρου 2x2 και διασκελισμό 2.
6. Flatten Layer - μετατρέπει την έξοδο του προηγούμενου στρώματος σε ένα μονοδιάστατο διάνυσμα, το οποίο απαιτείται ως είσοδος για τα πλήρως συνδεδεμένα στρώματα.
7. Πλήρως συνδεδεμένα επίπεδα (Fully Connected Layers) - τα πρώτα δυο πλήρως συνδεδεμένα επίπεδα αποτελούνται συγκεκριμένα από 1024, 512 νευρώνες και το τελευταίο πλήρως συνδεδεμένο στρώμα από μόνο 5 νευρώνες όσος και ο αριθμός των κλάσεων προς ταξινόμηση.
8. Επίπεδο Μέσης Συγκέντρωσης - εκτελεί μέση συγκέντρωση με μέγεθος παραθύρου 2x2 και διασκελισμό 2.
9. Επίπεδα εγκατάλειψης (Dropout Layer) - μηδενίζουν τυχαία ένα κλάσμα των μονάδων εισόδου κατά τη διάρκεια εκπαίδευσης, το ποσοστό εγκατάλειψης ορίστηκε στο 20%.
10. Συνάρτηση Softmax - μετατρέπει τις ακατέργαστες τιμές εξόδου του πλήρους συνδεδεμένου επιπέδου σε πιθανότητες.
11. Επίπεδο εξόδου (Classification Output Layer) - παράγει στην έξοδο μια πιθανότητα για κάθε μια από τις εικόνες χρησιμοποιώντας την απώλεια Softmax (Softmax Loss), γνωστή και ως Categorical Cross Entropy, εντάσσοντας κάθε μια από τις εικόνες στην κλάση που της αναλογεί.

3.1.4. Επιλογές εκπαίδευσης

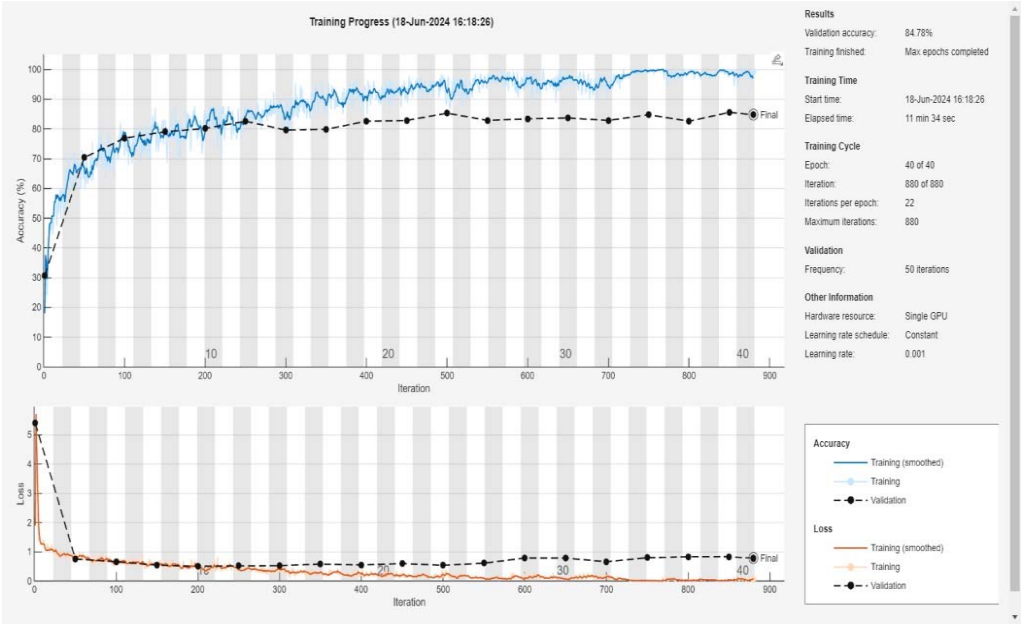
Για την εκπαίδευση του συνελικτικού νευρωνικού δικτύου χρησιμοποιήσαμε τον αλγόριθμο βελτιστοποίησης Adam με τον ρυθμό αποσύνθεσης για τον κινητό μέσο όρο των κλίσεων (Gradient Decay Factor), τον ρυθμό αποσύνθεσης για τον κινητό μέσο όρο των τετραγωνικών κλίσεων (Squared Gradient Dekay Factor), καθώς και την τιμή της σταθεράς e, ορισμένες στις προκαθορισμένες τιμές (default values) της συνάρτησης trainingOptions() της εργαλειοθήκης βαθιάς εκμάθησης (Deep Learning Toolbox) του λογισμικού Matlab. Η αρχική τιμή εκπαίδευσης εκτελείται για 20 εποχές και μετά μειώνεται και για άλλες 20 εποχές, το συνελικτικό δίκτυο εκπαιδεύεται με τη μισή τιμή

της αρχικής τιμής. Η επικύρωση των δεδομένων πραγματοποιείται κάθε 50 επαναλήψεις, χρησιμοποιώντας το σύνολο επικύρωσης με την πρόωρη διακοπή και τα σημεία ελέγχου να έχουν τεθεί στις προκαθορισμένες τιμές της συνάρτησης trainingOptions(). Τα δεδομένα εκπαίδευσης ανακατεύονται κάθε εποχή, η διαχείριση της τακτοποίησης γίνεται με τον όρο τακτοποίησης L2, ο οποίος ονομάζεται και παλινδρόμηση κορυφογραμμής, ορισμένο και το στοιχείο αυτό στην default τιμή, το περιβάλλον εκτέλεσης της εκπαίδευσης είναι ρυθμισμένο στην προκαθορισμένη τιμή της συνάρτησης και επιτρέπει στο λογισμικό να αποφασίσει εάν θα χρησιμοποιήσει την CPU ή την GPU για εκπαίδευση, ανάλογα με τη διαθεσιμότητα.

3.1.5. Αποτελέσματα εκπαίδευσης του Συνελικτικού Δικτιού

Μετά από το σχεδιασμό του συνελικτικού μοντέλου και τη ρύθμιση των επίλογων εκπαίδευσης σε περιβάλλον του λογισμικού Matlab βάλαμε το μοντέλο μας να εκπαιδευτεί για 40 εποχές με αλλαγή της αρχικής τιμής εκπαίδευσης στις 20 εποχές χρησιμοποιώντας τα τρία σύνολα δεδομένων, σύνολο δεδομένων εκπαίδευσης, σύνολο δεδομένων επικύρωσης , σύνολο δεδομένων δοκιμής όπως αυτά έχουν περιγραφεί στις παραπάνω υποενότητες. Το αποτέλεσμα της εκπαίδευσης έδειξε ότι το CNN μαθαίνει τις λεπτομέρειες και το θόρυβο στα δεδομένα εκπαίδευσης σε τέτοιο βαθμό που έχει καλή απόδοση στα δεδομένα εκπαίδευσης σε σύγκριση με τα δεδομένα επικύρωσης και δοκιμής, κάτι που μας οδηγεί στο συμπέρασμα ότι το μοντέλο μας υπερπροσαρμόζεται. Η υπερπροσαρμογή ανιχνεύτηκε με τη βοήθεια των μετρήσεων αξιολόγησης, των αποτελεσμάτων της ταξινόμησης και της γραφικής παράστασης.

Ακολουθούν η γραφική παράσταση της προόδου του μοντέλου κατά την εκπαίδευση (Σχήμα 3.1.5.α), τα αποτελέσματα της ταξινόμησης καθώς και οι μετρήσεις αξιολόγησης του μοντέλου.



Σχήμα 3.1.5.α: Υπερπροσαρμογή του μοντέλου στις 40 εποχές.

Στη συνέχεια θα αναλυθούν οι μήτρες σύγχυσης, που περιέχουν τις πέντε κλάσεις λουλουδιών (μαργαρίτα/daisy, πικραλίδα/dandelion, τριαντάφυλλα/roses,

ηλίανθους/sunflowers και τουλίπες/tulips), θα περιγράφει η ταξινόμηση που πραγματοποίησε το μοντέλο και θα αντληθούν οι μετρήσεις ακρίβειας ανά κλάση.

Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου εκπαίδευσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 506
  - Εσφαλμένη ταξινόμηση: 0
  - Ακρίβεια: 100,0%
  - Ποσοστό σφάλματος: -

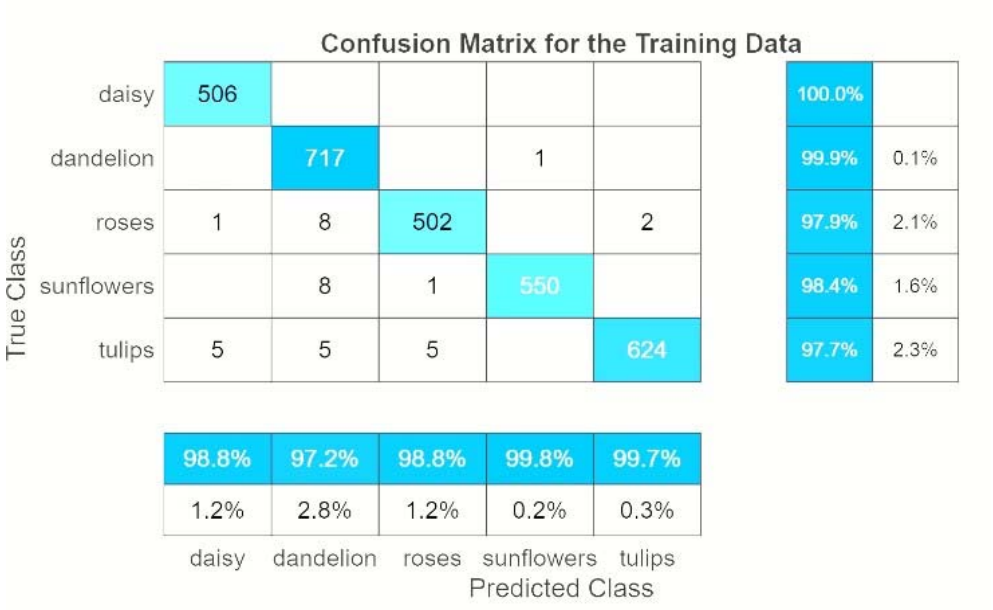
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 717
  - Εσφαλμένη ταξινόμηση: 1(1 ως ηλίανθους,)
  - Ακρίβεια: 99,9%
  - Ποσοστό σφάλματος: 0,1%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 502
  - Εσφαλμένη ταξινόμηση: 11(2 ως μαργαρίτα, 8 ως πικραλίδα, 2 ως τουλίπες)
  - Ακρίβεια: 97.9%
  - Ποσοστό σφάλματος: 2 ,1%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 550
  - Εσφαλμένη ταξινόμηση: 9 (8 ως πικραλίδα, 1 ως τριαντάφυλλα)
  - Ακρίβεια: 98,4%
  - Ποσοστό σφάλματος: 0 ,6%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 624
  - Εσφαλμένη ταξινόμηση: 15 (5 ως μαργαρίτα, 5 ως τριαντάφυλλα, 5 ως ηλίανθους)
  - Ακρίβεια: 97,7%
  - Ποσοστό σφάλματος: 2,3%

Πίνακας 3.1.5.α: Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 53
  - Εσφαλμένη ταξινόμηση: 11 (9 ως πικραλίδα, 2 ως τριαντάφυλλα)
  - Ακρίβεια: 82,8%
  - Ποσοστό σφάλματος: 17,2%

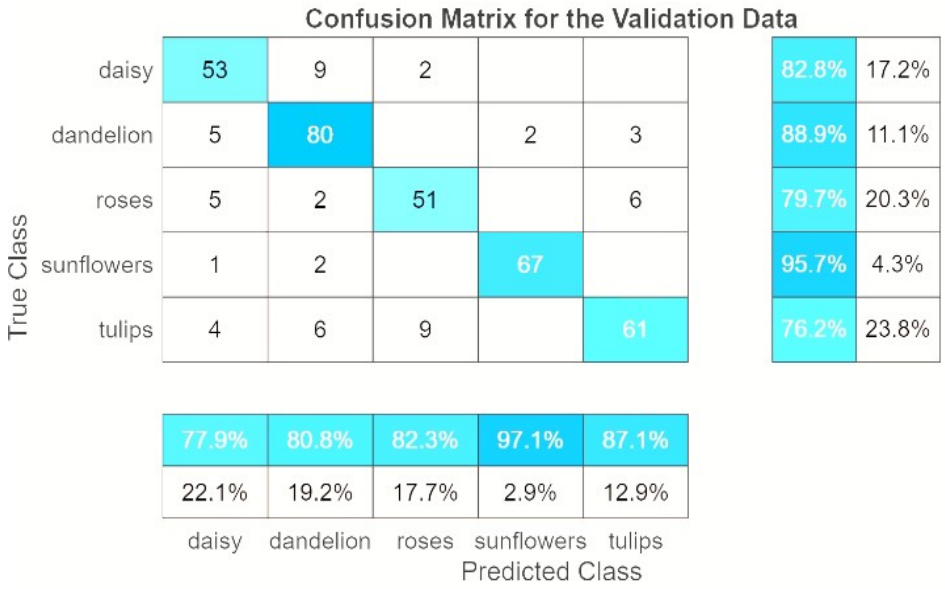
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 80
  - Εσφαλμένη ταξινόμηση: 10 (5 ως μαργαρίτα, 2 ως ηλίανθους, 3 ως τουλίπες)
  - Ακρίβεια: 88,9%
  - Ποσοστό σφάλματος: 11,1%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 51
  - Εσφαλμένη ταξινόμηση: 13 (5 ως μαργαρίτα, 2 ως πικραλίδα, 6 ως τουλίπες)
  - Ακρίβεια: 71,9%
  - Ποσοστό σφάλματος: 28,1%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 67
  - Εσφαλμένη ταξινόμηση: 3 (1 ως μαργαρίτα, 2 ως πικραλίδα)
  - Ακρίβεια: 95,7%
  - Ποσοστό σφάλματος: 4,3%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 61
  - Εσφαλμένη ταξινόμηση: 19 (4 ως μαργαρίτα, 6 ως πικραλίδα, 9 ως τριαντάφυλλα)
  - Ακρίβεια: 76,2%
  - Ποσοστό σφάλματος: 23,8%

Πίνακας 3.1.5.β: Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)

Στα δεδομένα επικύρωσης το συνελικτικό δίκτυο ταξινόμησε τα δεδομένα του συνόλου επικύρωσης με τον παρακάτω τρόπο:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 57
  - Εσφαλμένη ταξινόμηση: 17 (5 ως πικραλίδα, 5 ως τριαντάφυλλα, 7 ως τουλίπες)
  - Ακρίβεια: 77,0%
  - Ποσοστό σφάλματος: 23,0%

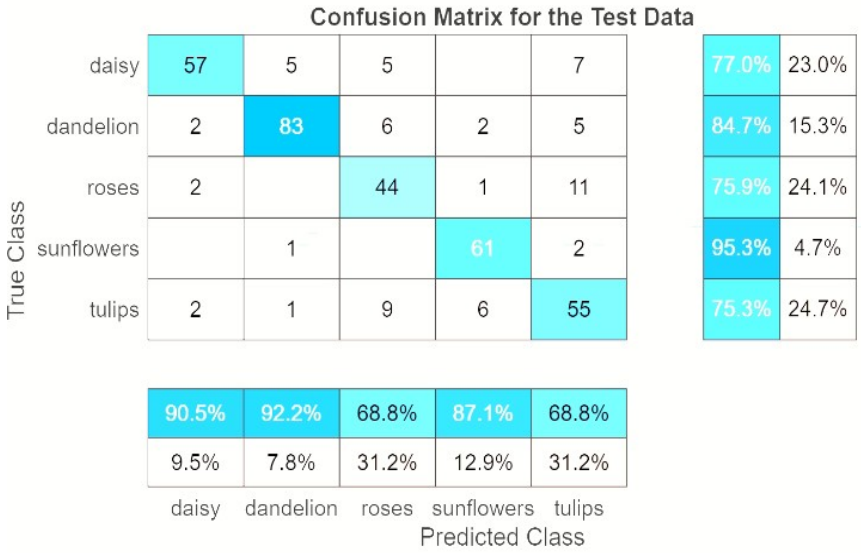
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 83
  - Εσφαλμένη ταξινόμηση: 15 (2 ως μαργαρίτα, 6 ως τριαντάφυλλα, 2 ως ηλίανθους, 5 ως τουλίπες)
  - Ακρίβεια: 84,7%
  - Ποσοστό σφάλματος: 15,3%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 51
  - Εσφαλμένη ταξινόμηση: 13 (2 ως μαργαρίτα, 1 ως ηλίανθους, 11 ως τουλίπες)
  - Ακρίβεια: 75,9%
  - Ποσοστό σφάλματος: 24,1%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 61
  - Εσφαλμένη ταξινόμηση: 3 (1 ως πικραλίδα, 1 ως τουλίπες)
  - Ακρίβεια: 95,3%
  - Ποσοστό σφάλματος: 4,7%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 55
  - Εσφαλμένη ταξινόμηση: 18(2 ως μαργαρίτα, 1 ως πικραλίδα, 9 ως τριαντάφυλλα, 6 ως ηλίανθους)
  - Ακρίβεια: 75,3%
  - Ποσοστό σφάλματος: 24,7%

Πίνακας 3.1.5.γ: Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)



Πίνακας 3.1.5.δ: Μετρήσεις απόδοσης του μοντέλου CNN στις 40 εποχές.

Metrics of CNN on 40 epoch
Training accuracy: 99.22%
Training loss: 0.06%
Validation accuracy: 84.78%
Validation loss: 0.78%

Test set accuracy: 81.74%

Από τα αποτελέσματα των πινάκων σύγχυσης συμπεραίνουμε ότι, όπως αναφέραμε και πιο πάνω, το μοντέλο αποδίδει εξαιρετικά στα δεδομένα εκπαίδευσης, με σχεδόν τέλεια ακρίβεια (99.22%), ενώ στο σύνολο δεδομένων επικύρωσης το μοντέλο εμφανίζει μεγάλο βαθμό εσφαλμένης ταξινόμησης με την απώλεια επικύρωσης να φτάνει στο 0,78% σε σχέση με την απώλεια εκπαίδευσής (0.06%), υποδηλώνοντας ότι το μοντέλο υπερπροσαρμόζεται στα δεδομένα του συνόλου επικύρωσης.

3.2. Υπερπροσαρμογή (Overfitting)

Η υπερπροσαρμογή συνήθως συμβαίνει υπό τις ακόλουθες συνθήκες:

1. Πολυπλοκότητα μοντέλου – όταν το μοντέλο έχει πάρα πολλές παραμέτρους σε σχέση με τον όγκο των δεδομένων εκπαίδευσης και έχει την ικανότητα να χωρέσει τον θόρυβο στα δεδομένα.
2. Ανεπαρκή δεδομένα εκπαίδευσης - όταν τα δεδομένα του συνόλου εκπαίδευσης δεν επαρκούν, το μοντέλο δεν μαθαίνει τα γενικά μοτίβα και είναι πιθανόν να μαθαίνει το θόρυβο.
3. Εκπαίδευση για πάρα πολλές εποχές - όταν το μοντέλο εκπαιδεύεται για πάρα πολλές εποχές τότε η υπερβολική προσαρμογή προκαλείται διότι το μοντέλο αρχίζει να απομνημονεύει τα δεδομένα του συνόλου εκπαίδευσης αντί να μαθαίνει να γενικεύει.
4. Θορυβώδη δεδομένα - αν το σύνολο των δεδομένων εκπαίδευσης περιέχουν πολύ θόρυβο και ακραίες τιμές, ένα σύνθετο μοντέλο μπορεί να μάθει αυτές τις ανωμαλίες αντί να τις αγνοήσει, οδηγώντας σε κακή γενίκευση.

Η μείωση της υπερπροσαρμογής γίνεται με τη χρήση στρατηγικών όπως:

1. Κανονικοποίηση (Regularization) – οι τεχνικές κανονικοποίησης προσθέτουν μια ποινή στη συνάρτηση κόστους/απώλειας για μεγάλους συντελεστές, αποθαρρύνοντας το μοντέλο να γίνει περίπλοκο.
2. Απλοποίηση του μοντέλου (Simplifying the Model) - η μείωση των αριθμού των παραμέτρων, η χρήση απλουστέρων μοντέλων η τεχνικών κλαδέματος σε δένδρα αποφάσεων μπορεί να βοηθήσει στην αποφυγή της υπερβολικής προσαρμογής του μοντέλου.
3. Επαύξηση δεδομένων (Data Augmentation) - η αύξηση του μεγέθους του συνόλου δεδομένων εκπαίδευσης με την προσθήκη τροποποιημένων εκδόσεων του ίδιου συνόλου δεδομένων μπορεί να βοηθήσει στη βελτίωση της γενίκευσης του μοντέλου.
4. Πρόωρη διακοπή (Early Stopping) - η διακοπή της διαδικασίας εκπαίδευσης μόλις αρχίζει να υποβαθμίζεται η απόδοση του μοντέλου στο σύνολο δεδομένων επικύρωσης εμποδίζει το μοντέλο να μάθει το θόρυβο στα δεδομένα εκπαίδευσης.
5. Διασταυρούμενη επικύρωση (Cross-Validation) - η χρήση διασταυρούμενης επικύρωσης k-fold διασφαλίζει ότι η απόδοση του μοντέλου αξιολογείται σε διαφορετικά υποσύνολα δεδομένων, παρέχοντάς μια καλύτερη εκτιμήση της ικανότητας του να γενικεύει.

6. Μέθοδοι συνόλου (Ensemble Methods) - Τεχνικές όπως το bagging και η boosting συνδυάζουν πολλά μοντέλα για να μειώσουν την υπερπροσαρμογή μειώνοντας το μέσο όρο του θορύβου.

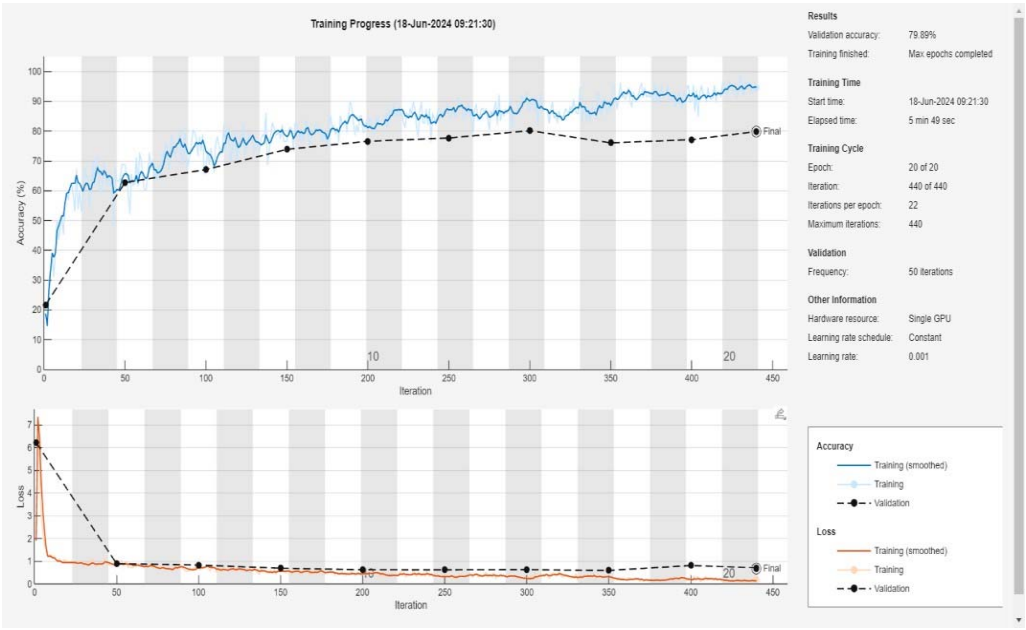
3.3. Εφαρμογή Στρατηγικών για την μείωση της υπερπροσαρμογής

Ένα από τα πρώτα βήματα που ακολουθήσαμε καθώς ήδη εφαρμόσαμε τεχνικές κανονικοποίησης (Regularization) ήταν να κοιτάξουμε αν η αρχιτεκτονική του μοντέλου CNN που υλοποιήσαμε είναι περίπλοκη και αν χρειάζεται να το απλοποιήσουμε. Εξετάζοντας από την αρχή το μοντέλο μας παρατηρήσαμε ότι η συγκεκριμένη αρχιτεκτονική περιλαμβάνει τυπικά στοιχεία ενός CNN και έχει σχεδιαστεί σχετικά με απλό τρόπο χωρίς προηγμένες τεχνικές όπως υπολειπόμενες συνδέσεις (residual connections), μονάδες έναρξης (inception modules) ή μηχανισμούς προσοχής (attention mechanism) που θα την καθιστούσαν περίπλοκη.

3.3.1. Μείωση του κύκλου εκπαίδευσης

Μείωση των εποχών από 40 στις 20.Το επόμενο μας βήμα ήταν να μειώσουμε τις εποχές στις 20 και να εκπαιδεύουμε εκ νέου το συνελικτικό μοντέλο μας. Από τους πίνακες σύγχυσης και από τους υπολογισμούς των μετρήσεων απόδοσης παρατηρήσαμε ότι το μοντέλο παρουσιάζει εσφαλμένη ταξινόμηση μεταξύ των τουλιπών και των τριαντάφυλλων καθώς και ότι χρειάζονται περαιτέρω βήματα για τη μείωση της υπερπροσαρμογής. Στο παρακάτω Σχήμα **3.3.1.α** φαίνεται ότι κατά την εκπαίδευση το μοντέλο υπερπροσαρμόζεται.

Η επισκόπηση της ταξινόμησης του μοντέλου καθώς και της απόδοσής του δίνονται παρακάτω από τα αποτελέσματα της ταξινόμησης του μοντέλου, τα αποτελέσματα των μετρήσεων απόδοσης καθώς και από την πρόοδο προπόνησης.



Σχήμα 3.3.1.α: Υπερπροσαρμογή του CNN στις 20 εποχές.

Η επισκόπηση της ταξινόμησης του μοντέλου έχει διαμορφωθεί ως εξής:

Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου εκπαίδευσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 500
  - Εσφαλμένη ταξινόμηση: 6 (2 ως πικραλίδα, 3 ως τριαντάφυλλα, 1 ως τουλίπες)
  - Ακρίβεια: 98,8%
  - Ποσοστό σφάλματος: 1,2%

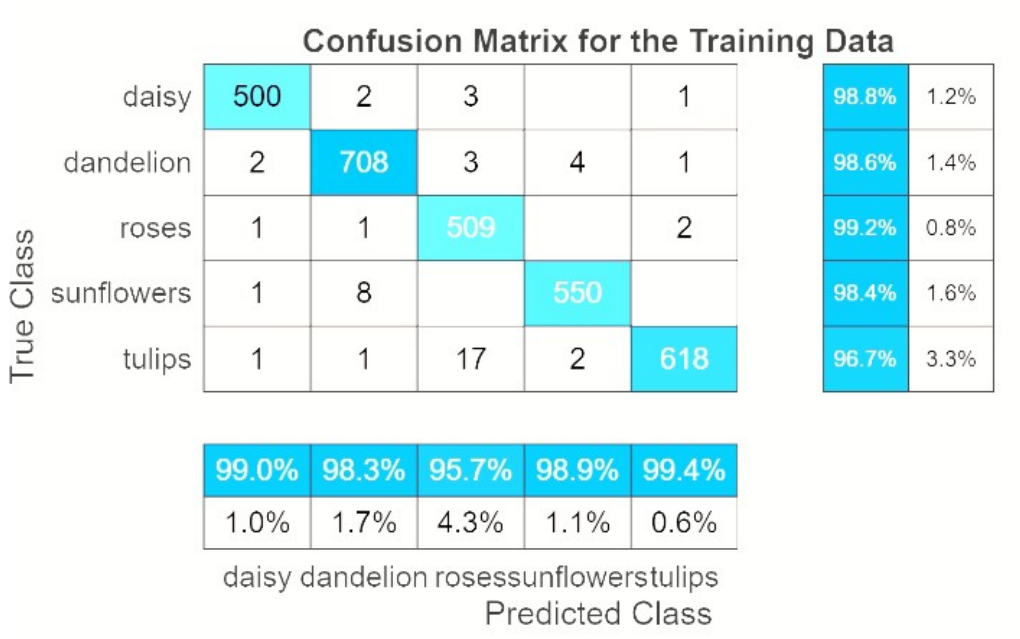
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 708
  - Εσφαλμένη ταξινόμηση: 9 (2 ως μαργαρίτα, 3 ως τριαντάφυλλα, 4 ως ηλίανθους, 1 ως τουλίπες)
  - Ακρίβεια: 98,6%
  - Ποσοστό σφάλματος: 1,4%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 509
  - Εσφαλμένη ταξινόμηση: 4 (1 ως μαργαρίτα, 1 ως πικραλίδα, 2 ως τουλίπες)
  - Ακρίβεια: 99.2%
  - Ποσοστό σφάλματος: 0,8%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 550
  - Εσφαλμένη ταξινόμηση: 9 (1 ως μαργαρίτα, 8 ως πικραλίδας)
  - Ακρίβεια: 98,4%
  - Ποσοστό σφάλματος:1 ,6%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 618
  - Εσφαλμένη ταξινόμηση: 21(1 ως μαργαρίτα, 1 ως πικραλίδα, 17 ως τριαντάφυλλα, 2 ως ηλίανθους)
  - Ακρίβεια: 96,7%
  - Ποσοστό σφάλματος: 3,3%

Πίνακας 3.3.1.α: Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 49
  - Εσφαλμένη ταξινόμηση: 15 (5 ως πικραλίδα, 5 ως τριαντάφυλλα, 1 ως ηλίανθους, 4 ως τουλίπες)
  - Ακρίβεια: 76,6%
  - Ποσοστό σφάλματος: 23,4%

- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 78
  - Εσφαλμένη ταξινόμηση: 12 (3 ως μαργαρίτα, 5 ως τριαντάφυλλα, 2 ως ηλίανθους, 2 ως τουλίπες)
  - Ακρίβεια: 86,7%
  - Ποσοστό σφάλματος: 13,3%

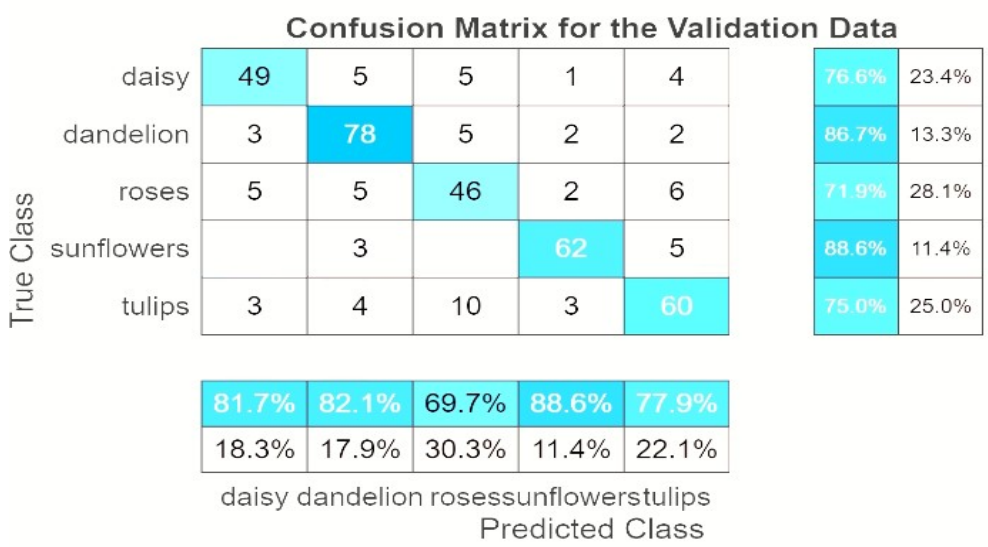
- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 46
  - Εσφαλμένη ταξινόμηση: 18 (5 ως μαργαρίτα, 5 ως πικραλίδα, 2 ως ηλίανθους , 6 ως τουλίπες)
  - Ακρίβεια: 71,9%
  - Ποσοστό σφάλματος: 28,1%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 62
  - Εσφαλμένη ταξινόμηση: 8 (3 ως πικραλίδα, 5 ως τουλίπες)
  - Ακρίβεια: 86,6%

- Ποσοστό σφάλματος: 11,4%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 60
  - Εσφαλμένη ταξινόμηση: 20 (3 ως μαργαρίτα, 4 ως πικραλίδα, 10 ως τριαντάφυλλα, 3 ως ηλίανθους)
  - Ακρίβεια: 75,0%
  - Ποσοστό σφάλματος: 25,0%

Πίνακας 3.3.1.β: Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 52
  - Εσφαλμένη ταξινόμηση:
  - 9 (2 ως πικραλίδα, 3 ως τριαντάφυλλα, 4 ως τουλίπες)
  - Ακρίβεια: 85,2%
  - Ποσοστό σφάλματος: 14,8%

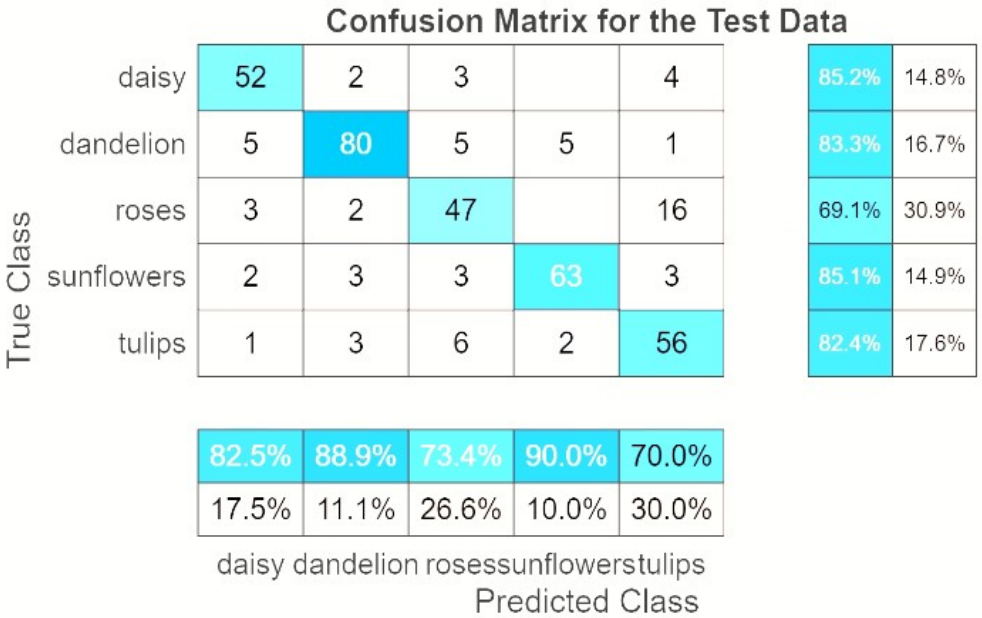
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 80
  - Εσφαλμένη ταξινόμηση: 16 (5 ως μαργαρίτα, 5 ως τριαντάφυλλα, 5 ως ηλίανθους, 1 ως τουλίπες)
  - Ακρίβεια: 83,3%
  - Ποσοστό σφάλματος: 16,7%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 47
  - Εσφαλμένη ταξινόμηση: 21 (3 ως μαργαρίτα, 2 ως πικραλίδα, 16 ως τουλίπες)
  - Ακρίβεια: 69,1%
  - Ποσοστό σφάλματος: 30 ,9%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 63
  - Εσφαλμένη ταξινόμηση: 11 (2 ως μαργαρίτα, 3 ως πικραλίδα, 3 ως τριαντάφυλλα, 3 ως τουλίπες)
  - Ακρίβεια: 85,1%
  - Ποσοστό σφάλματος: 14 ,9%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 56
  - Εσφαλμένη ταξινόμηση: 12 (1 ως μαργαρίτα, 3 ως πικραλίδα, 6 ως τριαντάφυλλα, 2 ως ηλίανθους)
  - Ακρίβεια: 82,4%
  - Ποσοστό σφάλματος: 17,6%

Πίνακας 3.3.1.γ: Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)



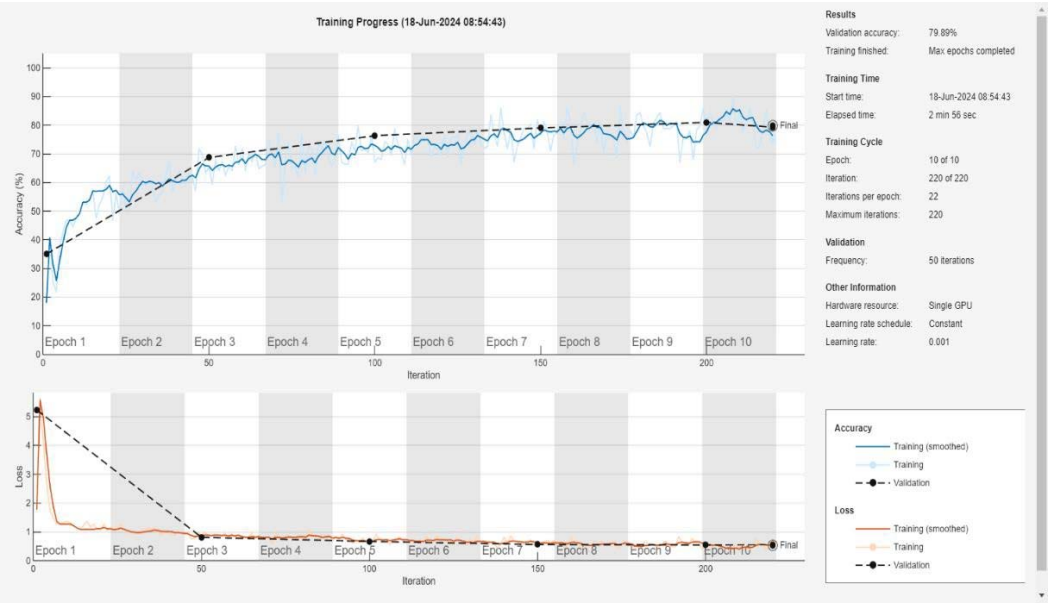
Πίνακας 3.3.1.δ: Μετρήσεις απόδοσης του μοντέλου CNN στις 20 εποχές.

Metrics of CNN on 20 epoch
Training accuracy: 94.53%
Training loss: 0.21%
Validation accuracy: 79.89%
Validation loss: 0.71%
Test set accuracy: 81.20%

Η ακρίβεια εκπαίδευσης (94.53%) είναι σημαντικά υψηλότερη από την ακρίβεια επικύρωσης (79.89%). Αυτό δείχνει ότι το μοντέλο αποδίδει πολύ καλά στα δεδομένα εκπαίδευσης αλλά όχι τόσο καλά στα δεδομένα επικύρωσης. Το μεγάλο χάσμα μεταξύ της εκπαίδευσης και της ακρίβειας επικύρωσης υποδηλώνει ότι το μοντέλο υπερπροσαρμόζεται στα δεδομένα εκπαίδευσης. Έχει μάθει πολύ καλά τα δεδομένα εκπαίδευσης, πιθανώς απομνημονεύοντάς τα, αλλά δεν γενικεύει τόσο αποτελεσματικά σε αόρατα δεδομένα.

Μείωση των εποχών από 20 εποχές στις 10.

Συνεχίσαμε μειώνοντας τις εποχές στις 10 για να αντιμετωπίσουμε την υπερπροσαρμογή του CNN μας χωρίς να κάνουμε καμία αλλαγή στην δομή του δικτύου. Παρατηρήσαμε ότι προκύπτει μια σχετική ταλάντωση μεταξύ της καμπύλης επικύρωσης και της καμπύλης εκπαίδευσης, με τελικές τιμές ακρίβειας εκπαίδευσης 75,0 και επικύρωσης 79,89 και τελικές τιμές απώλειας επικύρωσης 0.55 και εκπαίδευσης 0.62. Ακολουθούν η γραφική παράσταση της προόδου εκπαίδευσής, τα αποτελέσματα της ταξινόμησης του μοντέλου για κάθε σύνολο δεδομένων και τα αποτελέσματα της απόδοσης του στις 10 εποχές.



Σχήμα 3.3.1.β: Υποπροσαρμογή /Ανεπαρκής εκπαίδευση του μοντέλου CNN στις δέκα εποχές.

Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου εκπαίδευσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 439
  - Εσφαλμένη ταξινόμηση: 67 (34 ως πικραλίδα, 20 ως τριαντάφυλλα, 2 ως ηλίανθοι, 11 ως τουλίπες)
  - Ακρίβεια: 86,8%
  - Ποσοστό σφάλματος: 13,2%

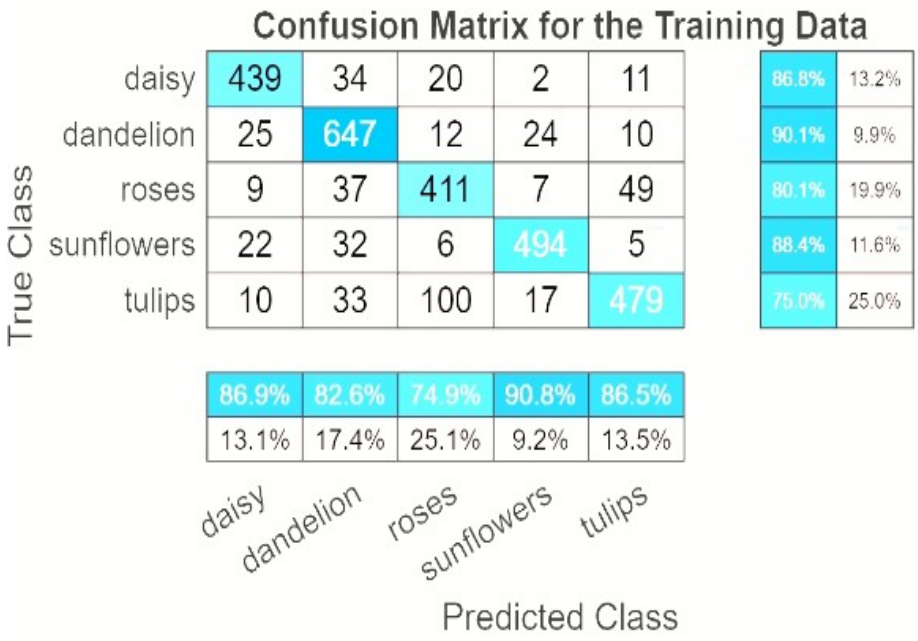
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 647
  - Εσφαλμένη ταξινόμηση: 71 (25 ως μαργαρίτα, 12 ως τριαντάφυλλα, 24 ως ηλίανθοι, 10 ως τουλίπες)
  - Ακρίβεια: 90,1%
  - Ποσοστό σφάλματος: 9,9%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 411
  - Εσφαλμένη ταξινόμηση: 102 (9 ως μαργαρίτα, 37 ως πικραλίδα, 7 ως ηλίανθοι, 49 ως τουλίπες)
  - Ακρίβεια: 80,1%
  - Ποσοστό σφάλματος: 19 ,9%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 494
  - Εσφαλμένη ταξινόμηση: 66 (22 ως μαργαρίτα, 32 ως πικραλίδα, 6 ως τριαντάφυλλα, 5 ως τουλίπες)
  - Ακρίβεια: 88,4%
  - Ποσοστό σφάλματος: 11,6%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 479
  - Εσφαλμένη ταξινόμηση: 160 (10 ως μαργαρίτα, 33 ως πικραλίδα, 100 ως τριαντάφυλλα, 17 ως ηλίανθοι)
  - Ακρίβεια: 75,0%
  - Ποσοστό σφάλματος: 25,0%

Πίνακας 3.3.1.ε: Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο:50
  - Εσφαλμένη ταξινόμηση: 14 (10 ως πικραλίδες, 4 ως τριαντάφυλλα)
  - Ακρίβεια: 78,1%
  - Ποσοστό σφάλματος: 21,9%

- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 79
  - Εσφαλμένη ταξινόμηση: 11(3 ως μαργαρίτα, 1 ως τριαντάφυλλα, 4 ως ηλίανθοι, 3 ως τουλίπες)
  - Ακρίβεια: 87,8%
  - Ποσοστό σφάλματος: 12.2%

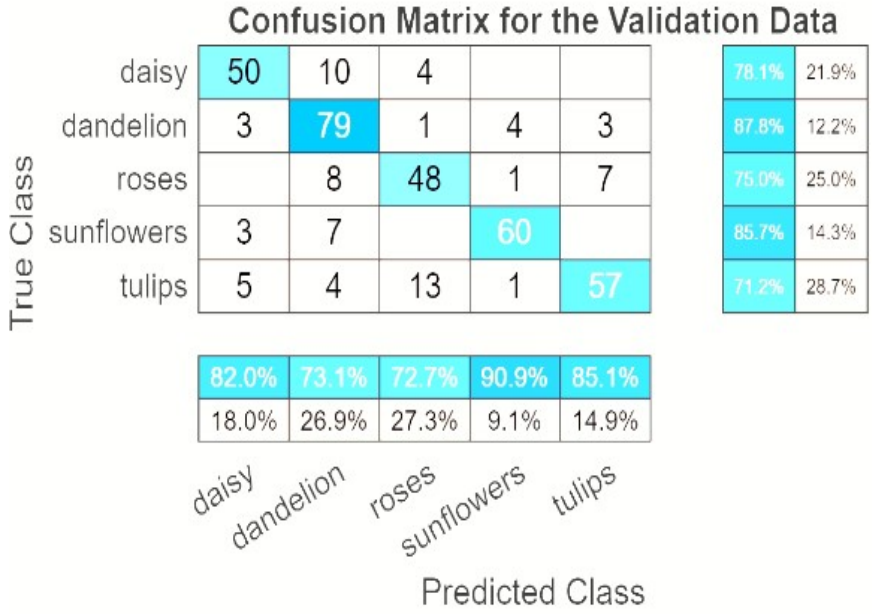
- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 48
  - Εσφαλμένη ταξινόμηση: 16 (8 ως πικραλίδες, 1 ως ηλίανθοι, 7 ως τουλίπες)
  - Ακρίβεια: 75,0%
  - Ποσοστό σφάλματος: 25,0%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 60
  - Εσφαλμένη ταξινόμηση: 10 (3 ως μαργαρίτα, 7 ως πικραλίδες)
  - Ακρίβεια: 85,7%

- Ποσοστό σφάλματος: 14,3%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 57
  - Εσφαλμένη ταξινόμηση: 23 (5 ως μαργαρίτα, 4 ως πικραλίδες, 13 ως τριαντάφυλλα, 1 ως ηλίανθοι)
  - Ακρίβεια: 71,3%
  - Ποσοστό σφάλματος: 28,7%

Πίνακας 3.3.1.στ: Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix).

Η ανάλυση της μήτρας σύγχυσης του συνόλου εκπαίδευσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 46
  - Εσφαλμένη ταξινόμηση: 7 (2 ως πικραλίδα, 5 ως τουλίπες)
  - Ακρίβεια: 86,8%
  - Ποσοστό σφάλματος: 13,2%

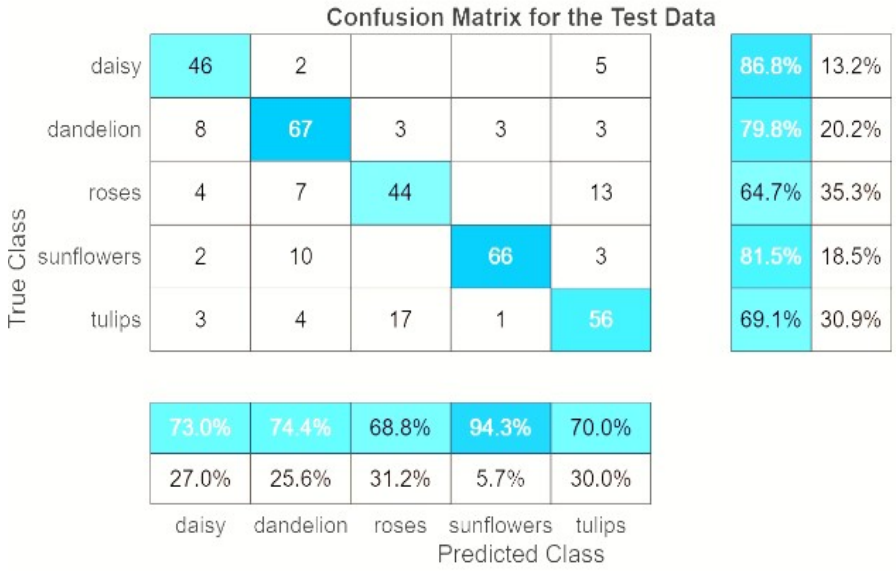
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 67
  - Εσφαλμένη ταξινόμηση: 17 (8 ως μαργαρίτα, 3 ως τριαντάφυλλα, 3 ως ηλίανθοι, 3 ως τουλίπες)
  - Ακρίβεια: 79,8%
  - Ποσοστό σφάλματος: 20.2%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 44
  - Εσφαλμένη ταξινόμηση: 24 (4 ως μαργαρίτα, 7 ως πικραλίδα, 13 ως τουλίπες)
  - Ακρίβεια: 64,7%
  - Ποσοστό σφάλματος: 35 ,3%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 66
  - Εσφαλμένη ταξινόμηση: 15 (2 ως μαργαρίτα, 10 ως πικραλίδα, 3 ως τουλίπες)
  - Ακρίβεια: 81,5%
  - Ποσοστό σφάλματος: 18 ,5%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 56
  - Εσφαλμένη ταξινόμηση: 25 (3 ως μαργαρίτα, 4 ως πικραλίδα, 17 ως τριαντάφυλλα, 1 ως ηλίανθοι)
  - Ακρίβεια: 69,1%
  - Ποσοστό σφάλματος: 30 ,9%

Πίνακας 3.3.1.ζ: Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)



Πίνακας 3.3.1.η: Μετρήσεις απόδοσης του μοντέλου στις δέκα εποχές.

Metrics of CNN on 10 epoch
Training accuracy: 75.00%
Training loss: 0.62%
Validation accuracy: 79.89%
Validation loss: 0.55%
Test set accuracy: 78.20%

Η ακρίβεια εκπαίδευσης είναι 75,00% και η ακρίβεια επικύρωσης είναι 79,89%. Αυτό υποδηλώνει ότι το μοντέλο αποδίδει ελαφρώς καλύτερα στο σετ επικύρωσης από ότι στο σετ εκπαίδευσης.

Απώλεια εκπαίδευσης (0,62) έναντι Απώλειας επικύρωσης (0,55): - Η απώλεια επικύρωσης είναι μικρότερη από την απώλεια εκπαίδευσης. Αυτό είναι επίσης ασυνήθιστο και μπορεί να υποδηλώνει: - Υποπροσαρμογή: Το μοντέλο μπορεί να υποχωρεί, να μη μαθαίνει τα δεδομένα εκπαίδευσης όσο καλά θα μπορούσε. Αυτό μπορεί να οφείλεται σε πολύ μικρή χωρητικότητα μοντέλου, υπερβολική τακτοποίηση ή ανεπαρκή εκπαίδευση. Η ακρίβεια δοκιμής (78.20%) πλησιάζει την ακρίβεια επικύρωσης (79,89%), υποδηλώνοντας ότι το μοντέλο γενικεύει καλά σε νέα αόρατα δεδομένα.

3.3.2. Επαύξηση Δεδομένων (Data Augmentation)

Για την αντιμετώπιση της υπερπροσαρμογής και προκειμένου να βελτιώσουμε την ευρωστία και την ικανότητα γενίκευσης του μοντέλου εφαρμόσαμε την επαύξηση δεδομένων (Data Augmentation), μια τεχνική επέκτασης του μεγέθους ενός συνόλου δεδομένων, η οποία δημιουργεί τροποποιημένες εκδόσεις εικόνων.

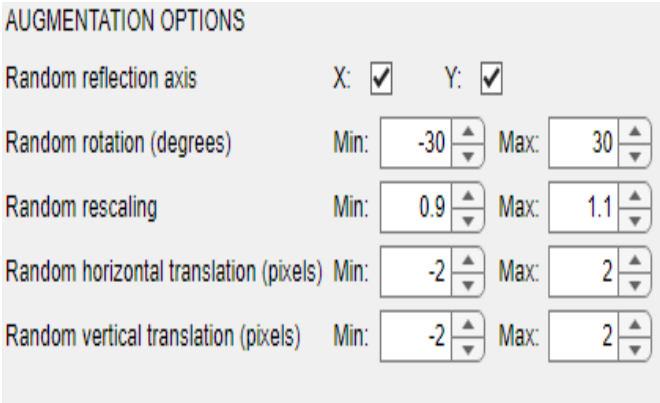
Με την εφαρμογή της παραπάνω μεθόδου μειώσαμε τα πλήρως συνδεδεμένα επίπεδα, αφήνοντας μόνο δυο αντί για τρία στρώματα, το πρώτο στρώμα να περιλαμβάνει 16 νευρώνες και το δεύτερο στρώμα 5 νευρώνες. Αφαιρέσαμε το ένα από τα δυο επίπεδα εγκατάλειψης, ορίζοντας νέο ποσοστό εγκατάλειψης ίσο με 0.15. Εκτός από τις παραπάνω αλλαγές αυξήσαμε την τιμή της κανονικοποίησης L2 στο 0.01, τις εποχές εκπαίδευσης του μοντέλου μας στις 80, ορίσαμε την τιμή του ποσοστού πτώσης του συντελεστή εκμάθησης, που θα εφαρμοστεί από τη $76^{η}$ εποχή, στο 0.1.

Δημιουργήσαμε έναν επαυξητή δεδομένων (Data augmenter) όπως φαίνεται στην Εικόνα 3.1.7.2.α. Παρακάτω ακολουθεί μια ανάλυση με το τί κάνει κάθε μέρος αυτού του επαυξητή (augmenter):

1. Rand X Reflection (Τυχαία Οριζόντια αντανάκλαση)
   Όταν οριστεί σε αληθής (true) επιτρέπει την τυχαία οριζόντια αντανάκλαση κατά το μήκος του άξονα X από αριστερά προς τα δεξιά.
2. Rand X Translation (Μετατόπιση κατά τον άξονα X)
   Οι εικόνες μετατοπίζονται οριζόντια κατά το μήκος του άξονα X κατά μια τυχαία ποσότητα εντός της περιοχής [-2 2] pixel.
3. Rand Y Translation (Τυχαία Μετατόπιση κατά τον άξονα Y)
   Οι εικόνες μετατοπίζονται κατακόρυφα κατά το μήκος του άξονα Y κατά μια τυχαία ποσότητα εντός της περιοχής [-2 2] pixel.
4. Rand Scale (Τυχαία Κλιμάκωση)

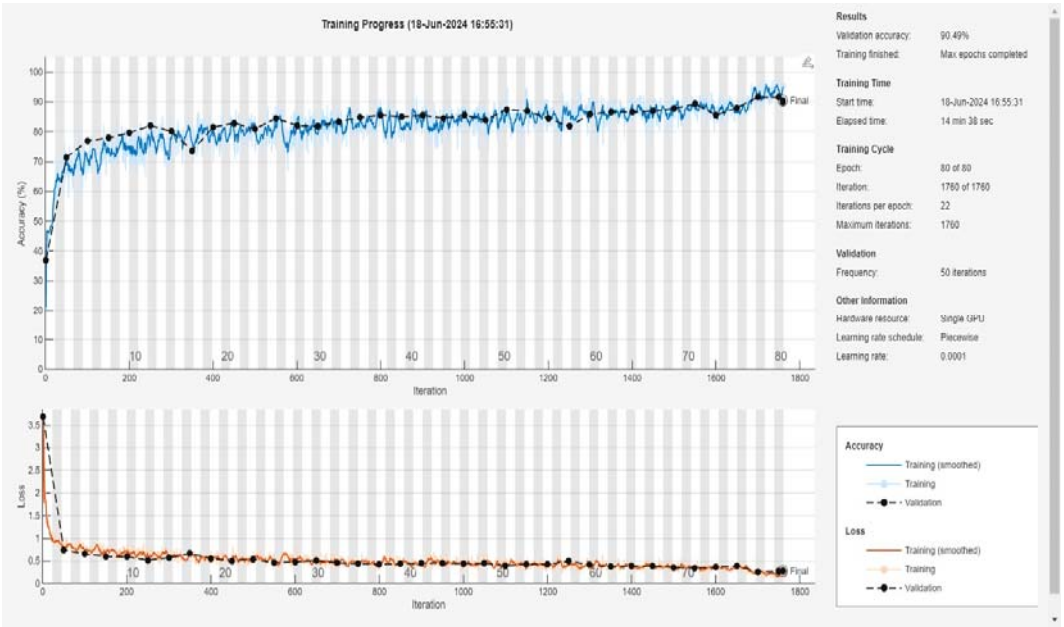
Επιτρέπει την κλιμάκωση των εικόνων με έναν τυχαίο παράγοντα εντός του εύρους [0.9 1.1].

5. Rand Rotation (Τυχαία Περιστροφή)
   Οι εικόνες περιστρέφονται τυχαία κατά το μήκος του άξονα X και Y εντός του εύρους [-30 30] μοιρών.



Εικόνα 3.3.2.α: Επαυξητής Δεδομένων (Data Augmenter)

Παρακάτω ακολουθούν: η γραφική παράσταση της προόδου εκπαίδευσής μετά από την εφαρμογή της τεχνικής της επαύξηση δεδομένων, τα αποτελέσματα της ταξινόμησης του μοντέλου για κάθε σύνολο δεδομένων και τα αποτελέσματα της απόδοσης του δικτύου.



Σχήμα 3.3.2.α: Αποτέλεσμα εκπαίδευσης μετά από την εφαρμογή της Επαύξησης Δεδομένων (Data Augmentation).

Η επισκόπηση της απόδοσης του μοντέλου μετά από την εφαρμογή της Επαύξησης Δεδομένων (Data Augmentation) έχει διαμορφωθεί ως εξής:

Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου εκπαίδευσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 491
  - Εσφαλμένη ταξινόμηση: 15 (6 ως τριαντάφυλλα, 9 ως τουλίπες)
  - Ακρίβεια: 97,0%
  - Ποσοστό σφάλματος: 3.0%

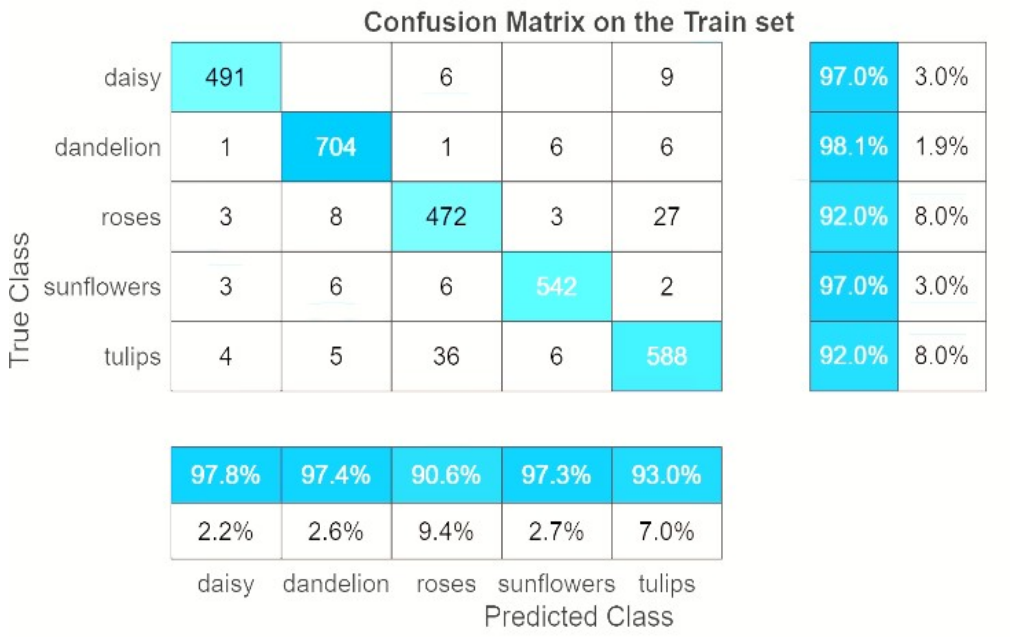
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 704
  - Εσφαλμένη ταξινόμηση: 14 (1 ως μαργαρίτα, 1 ως τριαντάφυλλα, 6 ως ηλίανθοι, 6 ως τουλίπες)
  - Ακρίβεια: 98,1%
  - Ποσοστό σφάλματος: 1,9%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 472
  - Εσφαλμένη ταξινόμηση: 41(3 ως μαργαρίτα, 8 ως πικραλίδα, 3 ως ηλίανθοι, 27 ως τουλίπες)
  - Ακρίβεια: 92,0%
  - Ποσοστό σφάλματος: 8,0%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 542
  - Εσφαλμένη ταξινόμηση: 17 (3 ως μαργαρίτα, 6 ως πικραλίδα, 6 ως τριαντάφυλλα, 2 ως τουλίπες)
  - Ακρίβεια: 97,0%
  - Ποσοστό σφάλματος: 3 ,0%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 588
  - Εσφαλμένη ταξινόμηση: 51 (4 ως μαργαρίτα, 5 ως πικραλίδα, 36 ως τριαντάφυλλα, 6 ως ηλίανθοι)
  - Ακρίβεια: 92,0%
  - Ποσοστό σφάλματος: 8 ,0%

Πίνακας 3.3.2.α: Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 59
  - Εσφαλμένη ταξινόμηση: 5 (3 ως πικραλίδες, 1 ως τριαντάφυλλα, 1 ως τουλίπες)
  - Ακρίβεια: 92,2%
  - Ποσοστό σφάλματος: 7,8%

- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 85
  - Εσφαλμένη ταξινόμηση: 5 (1 ως τριαντάφυλλα, 1 ως ηλίανθοι, 3 ως τουλίπες)
  - Ακρίβεια: 94,4%
  - Ποσοστό σφάλματος: 5,6%

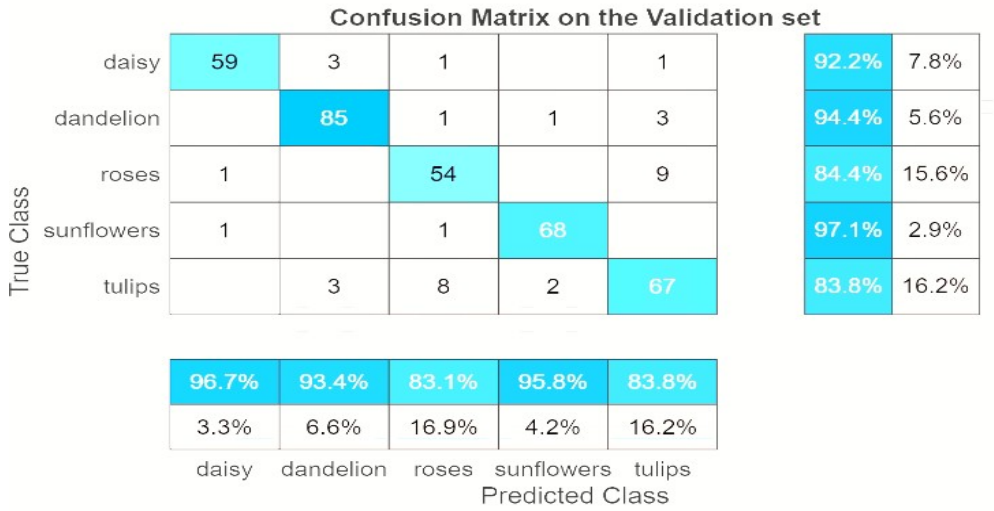
- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 54
  - Εσφαλμένη ταξινόμηση: 10 (1 ως μαργαρίτα, 9 ως τουλίπες)
  - Ακρίβεια: 84,4%
  - Ποσοστό σφάλματος: 15,6%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 68
  - Εσφαλμένη ταξινόμηση: 2 (1 ως μαργαρίτα, 1 ως πικραλίδα, 1 ως τριαντάφυλλα)
  - Ακρίβεια: 97,1%

- Ποσοστό σφάλματος: 2 ,9%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 67
  - Εσφαλμένη ταξινόμηση: 13 (3 ως πικραλίδες, 8 ως τριαντάφυλλα, 2 ως ηλίανθοι)
  - Ακρίβεια: 83,8%
  - Ποσοστό σφάλματος: 16,2%

Πίνακας 3.3.2.β: Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 56
  - Εσφαλμένη ταξινόμηση: 7 (5 ως πικραλίδες, 1 ως τριαντάφυλλα, 1 ως ηλίανθοι)
  - Ακρίβεια: 88,9%
  - Ποσοστό σφάλματος: 11,1%

- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 86
  - Εσφαλμένη ταξινόμηση: 4 (2 ως μαργαρίτα, 1 ως τριαντάφυλλα, 1 ως ηλίανθοι)
  - Ακρίβεια: 95,6%
  - Ποσοστό σφάλματος: 4,4%

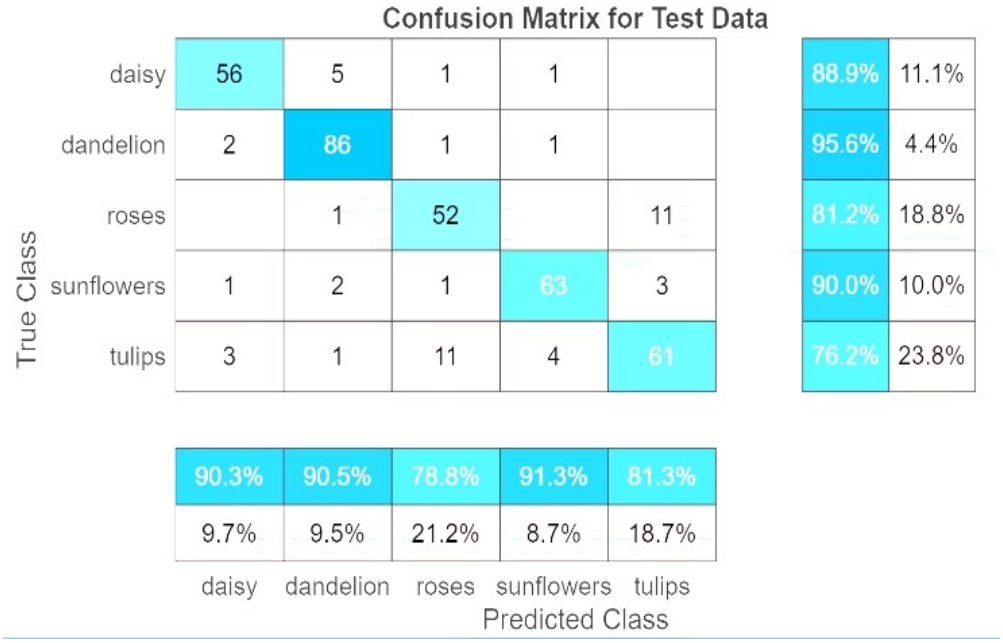
- Roses (Τριαντάφυλλα)

- Σωστά ταξινομημένο: 52
- Εσφαλμένη ταξινόμηση: 12 (1 ως πικραλίδα, 11 ως τουλίπες)
- Ακρίβεια: 81,2%
- Ποσοστό σφάλματος: 18 ,8%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 63
  - Εσφαλμένη ταξινόμηση: 7 (1 ως μαργαρίτα, 2 ως πικραλίδα, 1 ως τριαντάφυλλα, 3 ως τουλίπες)
  - Ακρίβεια: 90,0%
  - Ποσοστό σφάλματος: 10 ,0%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 63
  - Εσφαλμένη ταξινόμηση: 19 (3 ως μαργαρίτα, 1 ως πικραλίδα, 11 ως τριαντάφυλλα, 4 ως ηλίανθοι)
  - Ακρίβεια: 76,2%
  - Ποσοστό σφάλματος: 23 ,8%

Πίνακας 3.3.2.γ: Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)



Πίνακας 3.3.2.δ: Μετρήσεις απόδοσης του CNN μετά από την εφαρμογή της Επαύξησης Δεδομένων .

Metrics of CNN with Data Augmentation
Training accuracy: 91.41%
Training loss: 0.22%

Validation accuracy: 89.40%
Validation loss: 0.28%
Test set accuracy: 88.01%

Μετά από την εφαρμογή της μεθόδου της επαύξησης δεδομένων στα τρία υποσύνολα δεδομένων, εκπαίδευσης, επικύρωσης και δοκιμής του μοντέλου τα αποτελέσματα της εκπαίδευσης ήταν αρκετά θετικά. Φτάσαμε σε ποσοστό ακρίβειας 89.40% για το σύνολο δεδομένων επικύρωσης, 91.41% για το σύνολο εκπαίδευσης και 88.1% για το σύνολο δοκιμής.

3.3.3. Υπερδειγματοληψία (Oversampling)

Στην προσπάθεια μας να κάνουμε το μοντέλο μας πιο αποτελεσματικό ως προς την εργασία της ταξινόμησης εικόνων του συνόλου δεδομένων που περιέχει πέντε κλάσεις λουλουδιών εφαρμόσαμε την υπερδειγματοληψία, η οποία είναι μια τεχνική που χρησιμοποιείται στην ανάλυση δεδομένων και στη μηχανική μάθηση για την αντιμετώπιση του ζητήματος των μη ισορροπημένων συνόλων δεδομένων. Στα προβλήματα που αφορούν την ταξινόμηση είναι σύνηθες να συναντάμε σύνολα δεδομένων, όπου ο αριθμός των χαρακτηριστικών μιας κλάσης υπερτερεί σημαντικά ως προς τον αριθμό των χαρακτηριστικών μιας άλλης κλάσης, όπως και στην περίπτωση μας, όπου οι εικόνες της κλάσης dandelion/πικραλίδα υπερτερούν σε σχέση με τις εικόνες που βρίσκονται στις άλλες κλάσεις, όπως φαίνεται στον παρακάτω πίνακα.

Πίνακας 3.3.3.α: Ανισορροπία κλάσεων.

labelCount = 5×2 table

	Label	Count
1	daisy	633
2	dandelion	898
3	roses	641
4	sunflowers	699
5	tulips	799

Αυτό που κάναμε ήταν να επεξεργαστούμε το κάθε σύνολο δεδομένων (εκπαίδευσης, επικύρωσης, δοκιμής) και την κάθε κλάση ξεχωριστά, δηλαδή σε κάθε κλάση που είχαμε λιγότερες εικόνες από 898 (αριθμός εικόνων που αντιστοιχεί στις εικόνες τις κλάσης dandelion/πικραλίδα) προσθέσαμε μέσω αντιγραφής τυχαίες εικόνες έτσι ώστε να συμπληρωθεί ο αριθμός των 898 εικόνων.

Διαμορφώσαμε τα υποσύνολα εκπαίδευσης, επικύρωσης και δοκιμής ως εξής:

- 80% των εικόνων του συνόλου δεδομένων δόθηκε στο υποσύνολο εκπαίδευσης

Πίνακας 3.3.3.β: Υποσύνολο εκπαίδευσης.

Label	Count
daisy	718
dandelion	718
roses	718
sunflowers	718
tulips	718

- 10% των εικόνων του συνόλου δεδομένων δόθηκε στο υποσύνολο επικύρωσης

Πίνακας 3.3.3.γ: Υποσύνολο επικύρωσης

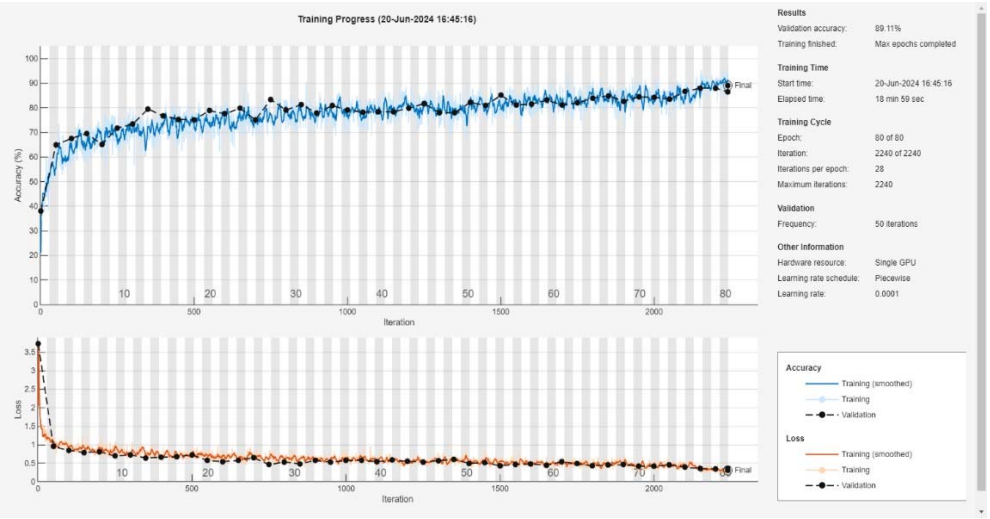
Label	Count
daisy	90
dandelion	90
roses	90
sunflowers	90
tulips	90

- 10% των εικόνων του συνόλου δεδομένων δόθηκε στο υποσύνολο δοκιμής

Πίνακας 3.3.3.δ: Υποσύνολο δοκιμής.

Label	Count
daisy	90
dandelion	90
roses	90
sunflowers	90
tulips	90

Εκπαιδεύσαμε το συνελικτικό μας δίκτυο με τις ίδιες επιλογές εκπαίδευσης αλλάζοντας μόνο την τιμή πιθανότητας στο dropoutLayer στο 0.45 για να αποφύγουμε την υπερπροσαρμογή. Παρακάτω ακολουθούν τα αποτελέσματα απόδοσης του μοντέλου μετά την εφαρμογή της τεχνικής της υπερδειγματοληψίας καθώς και η ανάλυση της ταξινόμησης που πραγματοποίησε το μοντέλο .



Σχήμα. 3.3.3.α: Πρόοδος εκπαίδευσης μοντέλου CNN μετά από την εφαρμογή της υπερδειγματοληψίας.

Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου εκπαίδευσης έδειξε ότι το μοντέλο μετά την εφαρμογή της υπερδειγματοληψίας ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 699
  - Εσφαλμένη ταξινόμηση: 19 (5ως πικραλίδα, 3 ως τριαντάφυλλα, 2 ως ηλίανθοι, 9 ως τουλίπες)

- Ακρίβεια: 97,4%
- Ποσοστό σφάλματος: 3.5%

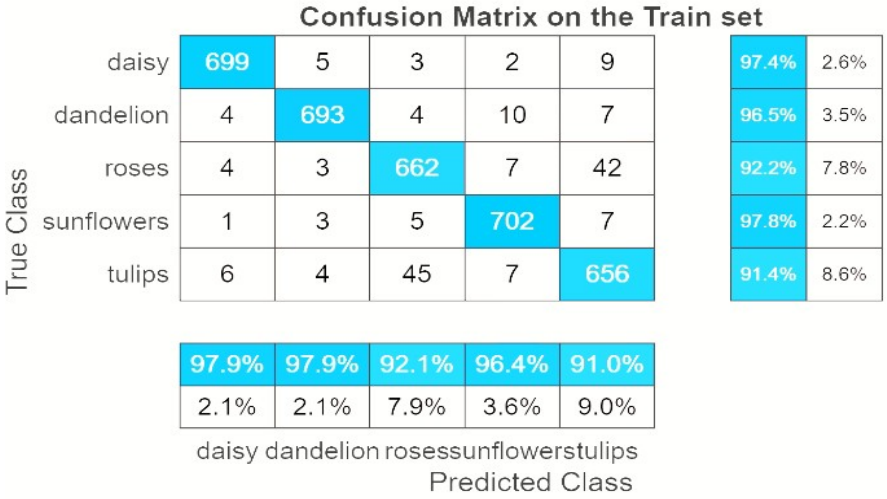
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 693
  - Εσφαλμένη ταξινόμηση: 25 (4 ως μαργαρίτα, 4 ως τριαντάφυλλα,10 ως ηλίανθοι, 7 ως τουλίπες)
  - Ακρίβεια: 96,5%
  - Ποσοστό σφάλματος: 3,5%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 662
  - Εσφαλμένη ταξινόμηση: 56 (4 ως μαργαρίτα, 3 ως πικραλίδα, 7 ως ηλίανθοι, 42 ως τουλίπες)
  - Ακρίβεια: 92,2%
  - Ποσοστό σφάλματος:7,8%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 702
  - Εσφαλμένη ταξινόμηση: 16 (1 ως μαργαρίτα, 3 ως πικραλίδα, 5 ως τριαντάφυλλα, 7 ως τουλίπες)
  - Ακρίβεια: 97,8%
  - Ποσοστό σφάλματος: 2,2%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 656
  - Εσφαλμένη ταξινόμηση: 62 (6 ως μαργαρίτα, 4 ως πικραλίδα, 45 ως τριαντάφυλλα, 7 ως ηλίανθοι)
  - Ακρίβεια: 91,4%
  - Ποσοστό σφάλματος: 8 ,6%

Πίνακας 3.3.3.ε: Μήτρα σύγχυσης του συνόλου εκπαίδευσης (Training Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

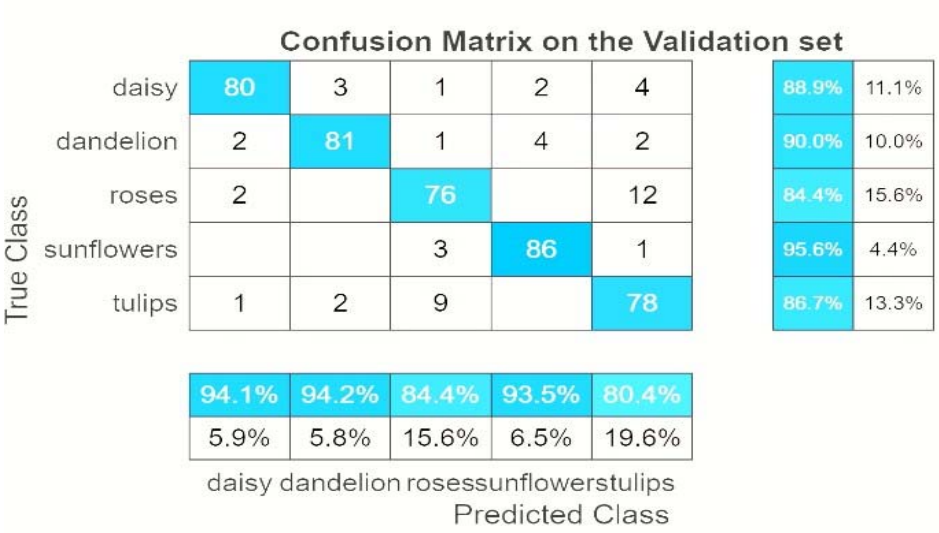
- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 80
  - Εσφαλμένη ταξινόμηση: 10 (3 ως πικραλίδα, 1 ως τριαντάφυλλα, 2 ως ηλίανθοι, 4 ως τουλίπες)
  - Ακρίβεια: 88,9%
  - Ποσοστό σφάλματος: 11,1%
- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 81
  - Εσφαλμένη ταξινόμηση: 9 (2 ως μαργαρίτα, 1 ως τριαντάφυλλα, 4 ως ηλίανθοι, 2 ως τουλίπες)
  - Ακρίβεια: 90,0%
  - Ποσοστό σφάλματος: 10,0%

- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 76
  - Εσφαλμένη ταξινόμηση: 14 (2 ως μαργαρίτα, 12 ως τουλίπες)
  - Ακρίβεια: 84,4%
  - Ποσοστό σφάλματος: 15,6%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 86
  - Εσφαλμένη ταξινόμηση: 4 (3 ως τριαντάφυλλα, 1 ως τουλίπες)
  - Ακρίβεια: 95,6%
  - Ποσοστό σφάλματος: 4,4%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 78
  - Εσφαλμένη ταξινόμηση: 12 (1 ως μαργαρίτα, 2 ως πικραλίδα, 9 ως τριαντάφυλλα)
  - Ακρίβεια: 86,7%
  - Ποσοστό σφάλματος:13,3%

Πίνακας 3.3.3.στ: Μήτρα σύγχυσης του συνόλου επικύρωσης (Validation Data Confusion Matrix)



Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)

Η ανάλυση της μήτρας σύγχυσης του συνόλου επικύρωσης έδειξε ότι το μοντέλο ταξινόμησε τα λουλούδια στις κλάσεις ως εξής:

- Daisy (Μαργαρίτα)
  - Σωστά ταξινομημένο: 79
  - Εσφαλμένη ταξινόμηση: 11 (3 ως πικραλίδα, 4 ως τριαντάφυλλα, 1 ως ηλίανθοι, 3 ως τουλίπες)
  - Ακρίβεια: 87,8%
  - Ποσοστό σφάλματος: 12,2%

- Dandelion (Πικραλίδα)
  - Σωστά ταξινομημένο: 79
  - Εσφαλμένη ταξινόμηση: 11 (2 ως μαργαρίτα, 1 ως τριαντάφυλλα, 4 ως ηλίανθοι, 4 ως τουλίπες))
  - Ακρίβεια: 87,8%
  - Ποσοστό σφάλματος: 12,2%

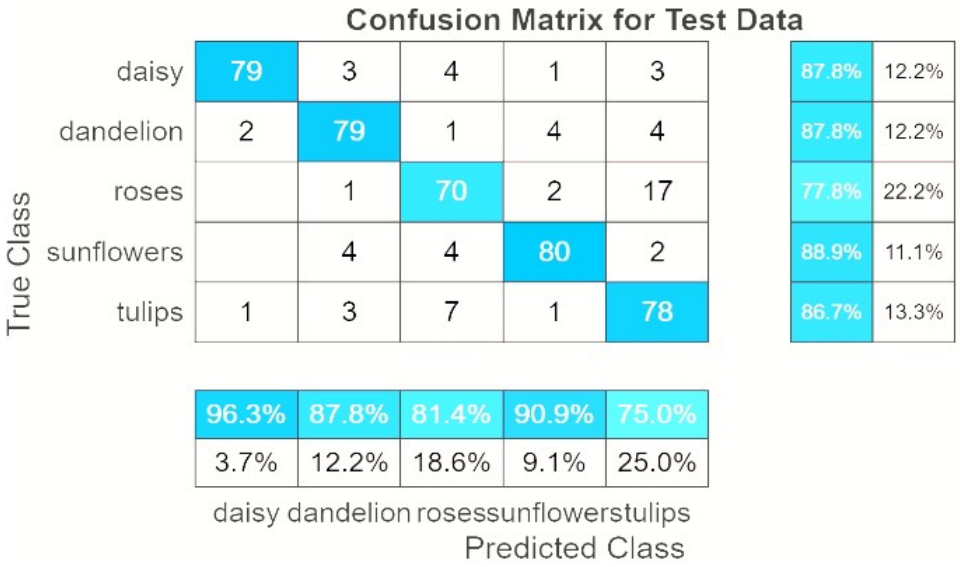
- Roses (Τριαντάφυλλα)
  - Σωστά ταξινομημένο: 70

- Εσφαλμένη ταξινόμηση: 20 (1 ως πικραλίδα, 2 ως ηλίανθοι, 17 ως τουλίπες)
- Ακρίβεια: 77,8%
- Ποσοστό σφάλματος: 22 ,2%

- Sunflowers (Ηλίανθοι)
  - Σωστά ταξινομημένο: 80
  - Εσφαλμένη ταξινόμηση: 10 (4 ως πικραλίδα, 4 ως τριαντάφυλλα, 2 ως τουλίπες)
  - Ακρίβεια: 88,9%
  - Ποσοστό σφάλματος: 11,1%

- Tulips (Τουλίπες)
  - Σωστά ταξινομημένο: 78
  - Εσφαλμένη ταξινόμηση: 12 (1 ως μαργαρίτα, 3 ως πικραλίδα, 7 ως τριαντάφυλλα, 1ως ηλίανθοι)
  - Ακρίβεια: 66,7%
  - Ποσοστό σφάλματος: 13,3%

Πίνακας 3.3.3.ζ: Μήτρα σύγχυσης του συνόλου δοκιμής (Test Data Confusion Matrix)



Πίνακας 3.3.3.η: Μετρήσεις απόδοσης του μοντέλου μετά από την εφαρμογή της υπερδειγματοληψίας.

Oversampling
Training accuracy: 90.62%
Training loss: 0.29%
Validation accuracy: 86.44%
Validation loss: 0.36%
Test set accuracy: 85.78%

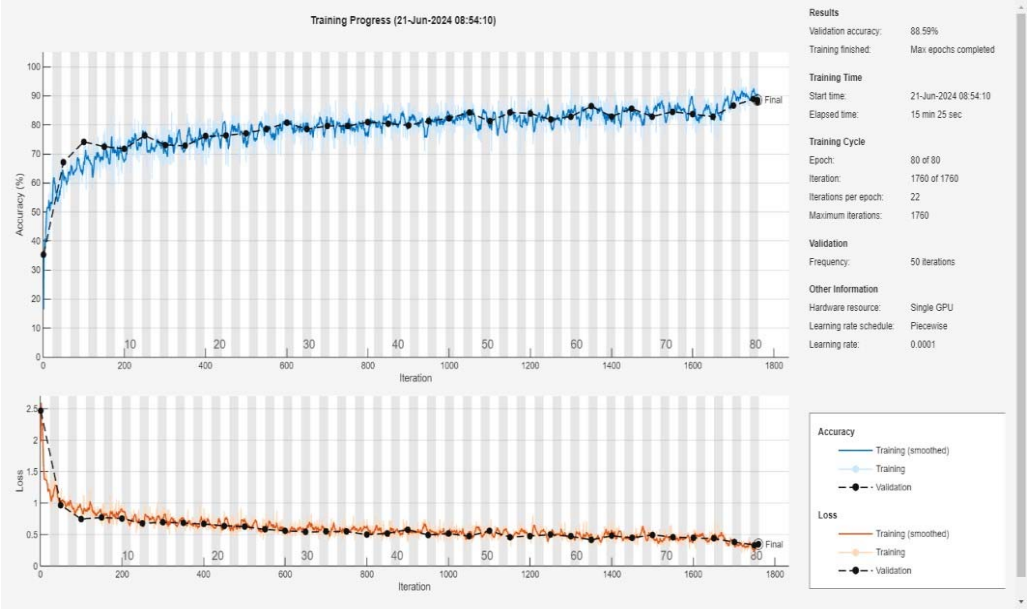
Η τιμές των μετρήσεων απόδοσης υποδηλώνουν ότι: το μοντέλο έχει μάθει αποτελεσματικά από το σύνολο δεδομένων εκπαίδευσης, επιδεικνύοντας καλή εκμάθηση με ελάχιστο σφάλμα 0.29 στο σύνολο δεδομένων εκπαίδευσης και αποδίδει καλά σε αόρατα δεδομένα με το σφάλμα επικύρωσης να είναι μεγαλύτερο 0.36. Ένα μεγαλύτερο χάσμα μεταξύ απώλειας εκπαίδευσης και επικύρωσης θα δήλωνε υπερβολική υποπροσαρμογή.

3.3.4. Διασταυρούμενη Επικύρωση K-fold (K-fold Cross-Validation)

Στο σύνολο των δεδομένων μας εφαρμόσαμε την τεχνική διασταυρούμενης επικύρωσης K-fold. Η τεχνική αυτή είναι μια τεχνική επαναδειγματοληψίας, η οποία χρησιμοποιείται για την αξιολόγηση της απόδοσης των μοντέλων βαθιάς μάθησης σε ένα σύνολο δεδομένων σχετικά περιορισμένο ως προς τα χαρακτηριστικά του, όπως είναι και το σύνολο των δεδομένων μας. Για την εφαρμογή του K-fold Cross-Validation επιλέξαμε το κ=10, δηλαδή το μοντέλο μας θα εκπαιδευθεί δέκα φορές χρησιμοποιώντας κάθε φορά ένα τυχαίο υποσύνολο δεδομένων εκπαίδευσης, επικύρωσης και δοκιμής. Ακολουθούν τα αναλυτικά αποτελέσματα του μοντέλου μετά από την εφαρμογή των K-fold Cross Validation.

Fold 1.

Ακολουθούν τα αποτελέσματα του μοντέλου μετά από την 1η επαναδειγματοληψία:



Σχήμα. 3.3.4.α: Πρόοδος εκπαίδευσης Fold 1

Πίνακας 3.3.4.α: Μετρήσεις απόδοσης Fold 1

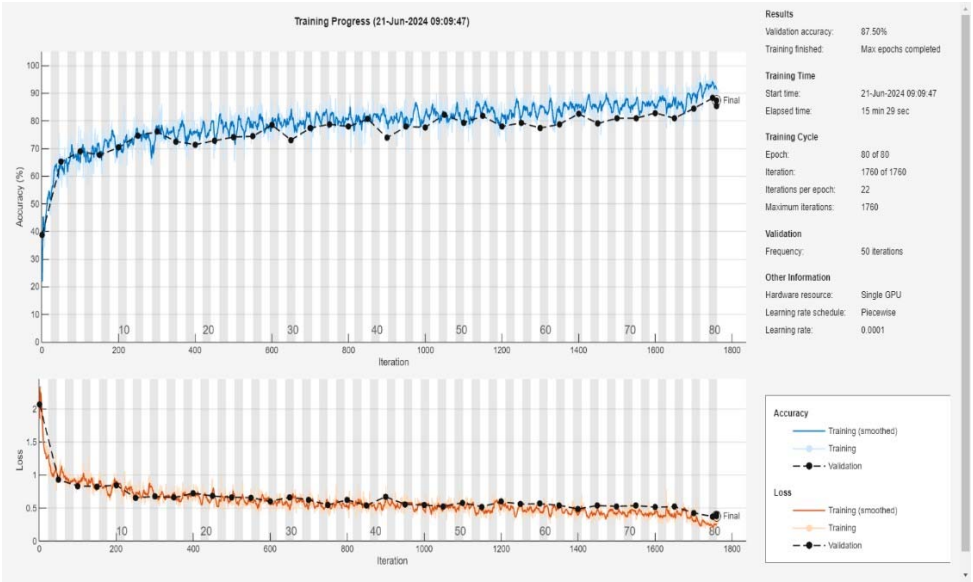
Fold 1
Training accuracy: 89.84%
Validation accuracy: 88.04%
Training loss: 0.36%
Validation loss: 0.35%

Test set accuracy: 86.10%

Από την πρόοδο εκπαίδευσης καθώς και από τις μετρήσεις απόδοσης το μοντέλο δείχνει μια σχετική καλή ισορροπία μεταξύ της ακρίβειας εκπαίδευσης (89,84) και της ακρίβειας επικύρωσης (88,04), που σημαίνει ότι ταιριάζει καλά τα δεδομένα εκπαίδευσης και επικύρωσης χωρίς υπερπροσαρμογή. Το ποσοστό ακρίβειας στο σύνολο δοκιμής 86,10, που είναι ελαφρώς χαμηλότερο, δείχνει μειωμένη γενίκευση σε αόρατα δεδομένα. Τόσο η απώλεια εκπαίδευσης (0,36%) όσο και η απώλεια επικύρωσης (0,35%) είναι χαμηλές, υποδηλώνοντας ότι το μοντέλο ταιριάζει καλά στα δεδομένα χωρίς σημαντική υπερπροσαρμογή. Αυτή η χαμηλή απώλεια υποδηλώνει ότι οι προβλέψεις του μοντέλου είναι κοντά στις πραγματικές τιμές τόσο κατά τη διάρκεια της εκπαίδευσης όσο και κατά τη διάρκεια της επικύρωσης.

Fold 2.

Ακολουθεί η απεικόνιση της προόδου εκπαίδευσής του CNN για τη δεύτερη επαναδειγματοληψία και ο πίνακας με τα αποτελέσματα της απόδοσης του μοντέλου στη συγκεκριμένη περίπτωση.



Σχήμα. 3.3.4.β: Πρόοδος εκπαίδευσης Fold 2

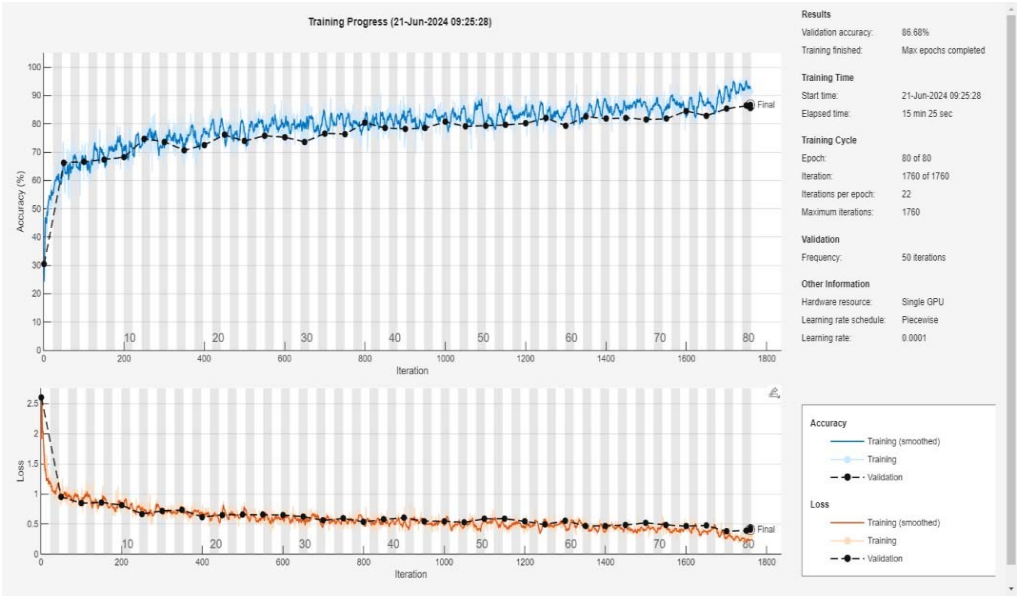
Πίνακας 3.3.4.β: Μετρήσεις απόδοσης Fold 2

Fold 2
Training accuracy: 90.62%
Validation accuracy: 85.33%
Training loss: 0.31%
Validation loss: 0.41%
Test set accuracy: 88.56%

Στο δεύτερο fold έχουμε υψηλότερη ακρίβεια εκπαίδευσης (90.62%) και μια σχετική μεγάλη πτώση στην ακρίβεια επικύρωσης (85.33%), γεγονός που υποδηλώνει ότι υπάρχει υπερπροσαρμογή του δικτύου στα δεδομένα επικύρωσης. Αυτό επιβεβαιώνεται και από τις τιμές της απώλειας επικύρωσης (0.41%) καθώς και της απώλειας εκπαίδευσης (0.31%). Ωστόσο το μοντέλο μας γενικεύει καλυτέρα σε αόρατα δεδομένα καθώς η ακρίβεια του συνόλου δοκιμής (88.56%) είναι μεγαλύτερη από την ακρίβεια επικύρωσης (85.33%).

Fold 3.

Ακολουθεί η γραφική παράσταση της προόδου εκπαίδευσης του μοντέλου CNN για την 3η επαναδειγματοληψία και ο πίνακας με τα αποτελέσματα της απόδοσης του μοντέλου στη συγκεκριμένη περίπτωση.



Σχήμα 3.3.4.γ: Πρόοδος εκπαίδευσης Fold 3

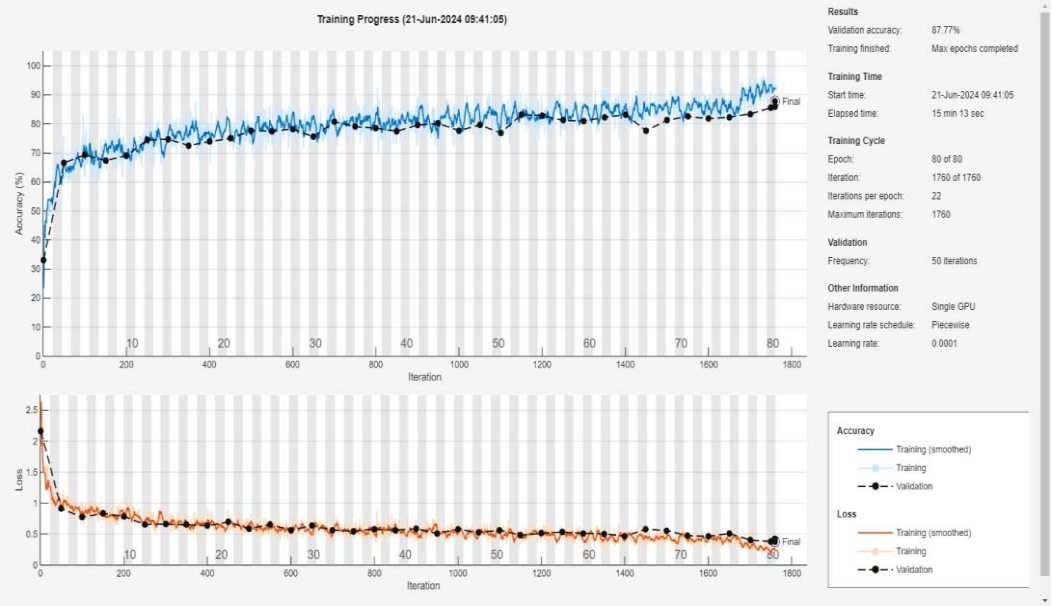
Πίνακας 3.3.4.γ: Μετρήσεις απόδοσης Fold 3

Fold 3
Training accuracy: 91.41%
Validation accuracy: 85.60%
Training loss: 0.30%
Validation loss: 0.43%
Test set accuracy: 86.38%

Στην 3η επαναδειγματοληψία παρατηρείται μια ελαφρά πτώση της ακρίβειας επικύρωσης (85.60%), με αυξημένη απώλεια επικύρωσης (0.43%) σε σύγκριση με την απώλεια εκπαίδευσης, που δείχνει μέτρια υπερπροσαρμογή (0.30%) του μοντέλου στα δεδομένα επικύρωσης. Παρατηρείται καλή γενίκευση του μοντέλου σε αόρατα δεδομένα με την ακρίβεια του συνόλου δοκιμής να φτάνει στο 86.38%.

Fold 4.

Ακολουθούν η γραφική παράσταση της προόδου εκπαίδευσής του δικτύου και ο πίνακας που περιλαμβάνει τις μετρήσεις απόδοσης του μοντέλου στην περίπτωση της 4ης επαναδειγματοληψίας.



Σχήμα 3.3.4.δ: Πρόοδος εκπαίδευσης Fold 4

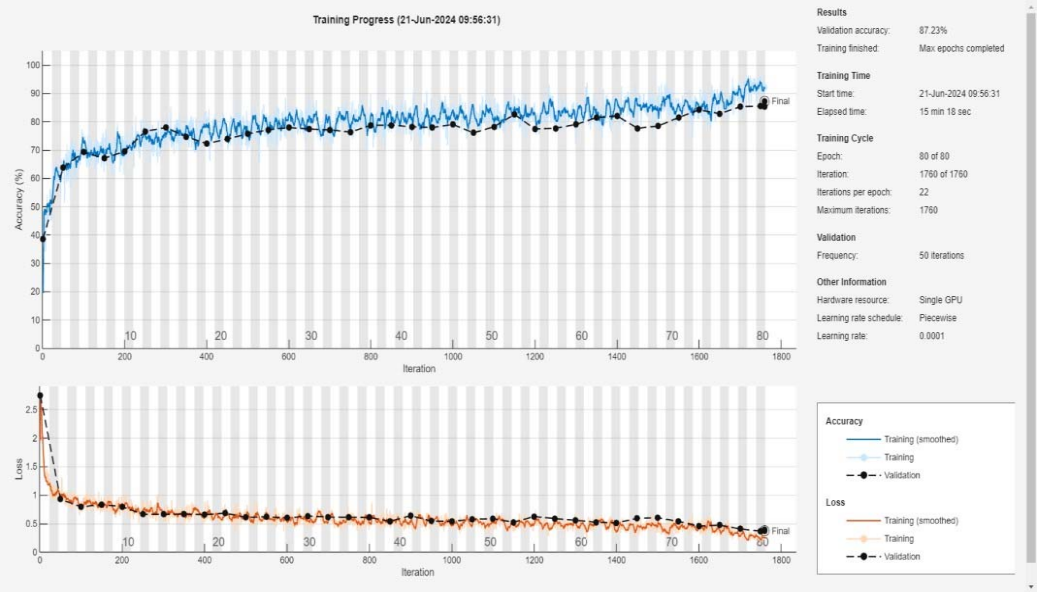
Πίνακας 3.3.4.δ: Μετρήσεις απόδοσης Fold 4

Fold 4
Training accuracy: 92.19%
Validation accuracy: 85.87%
Training loss: 0.36%
Validation loss: 0.43%
Test set accuracy: 86.65%

Εδώ παρατηρείται υψηλή ακρίβεια εκπαίδευσης (92.19%) με σταθερή ακρίβεια επικύρωσης (85.87%) και με την ακρίβεια του συνόλου δοκιμής (86,65%) να ευθυγραμμίζεται στενά με την ακρίβεια επικύρωσης, υποδηλώνοντας λογική γενίκευση σε νέα, αόρατα δεδομένα. Η απώλεια εκπαίδευσης (0,36%) είναι χαμηλή, υποδηλώνοντας καλή προσαρμογή στα δεδομένα εκπαίδευσης. Ωστόσο, η υψηλότερη απώλεια επικύρωσης (0,43%) υποδηλώνει ότι το μοντέλο μπορεί να μη γενικεύεται εξίσου καλά στα δεδομένα επικύρωσης, υποδεικνύοντας πιθανή υπερπροσαρμογή.

Fold 5.

Παρακάτω δίνεται η γραφική παράσταση της προόδου εκπαίδευσης του δικτύου και ο πίνακας με τα αποτελέσματα των μετρήσεων απόδοσής του στην περίπτωση της 5ης επαναδειγματοληψίας.



Σχήμα 3.3.4.ε: Πρόοδος εκπαίδευσης Fold 5

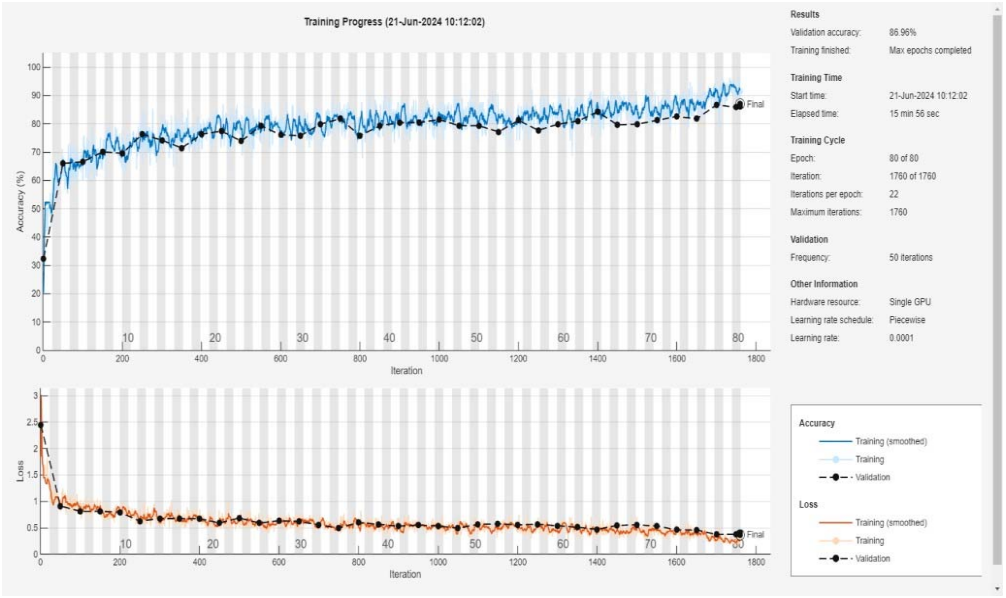
Πίνακας 3.3.4.ε: Μετρήσεις απόδοσης Fold 5

Fold 5
Training accuracy: 92.19%
Validation accuracy: 85.33%
Training loss: 0.34%
Validation loss: 0.39%
Test set accuracy: 87.47%

Εδώ το μοντέλο παρουσιάζει ελαφρώς χαμηλότερη ακρίβεια επικύρωσης (85.33%) από ότι στο προηγούμενο fold (fold 4) και υψηλότερη ακρίβεια εκπαίδευσής (92.19%), με την ακρίβεια στο σύνολο δοκιμής να είναι αρκετά καλή (87.47%), δείχνοντας ότι το μοντέλο στην περίπτωση αυτή υπερπροσαρμόζεται στα δεδομένα του συνόλου επικύρωσης και γενικεύει καλύτερα σε νέα δεδομένα από ότι στα δεδομένα επικύρωσης.

Fold 6.

Ακολουθεί η γραφική παράσταση της προόδου εκπαίδευσης καθώς και ο πίνακας που περιλαμβάνει τις μετρήσεις απόδοσης της 6^{ης} επαναδειγματοληψίας.



Σχήμα 3.3.4.στ: Πρόοδος εκπαίδευσης Fold 6

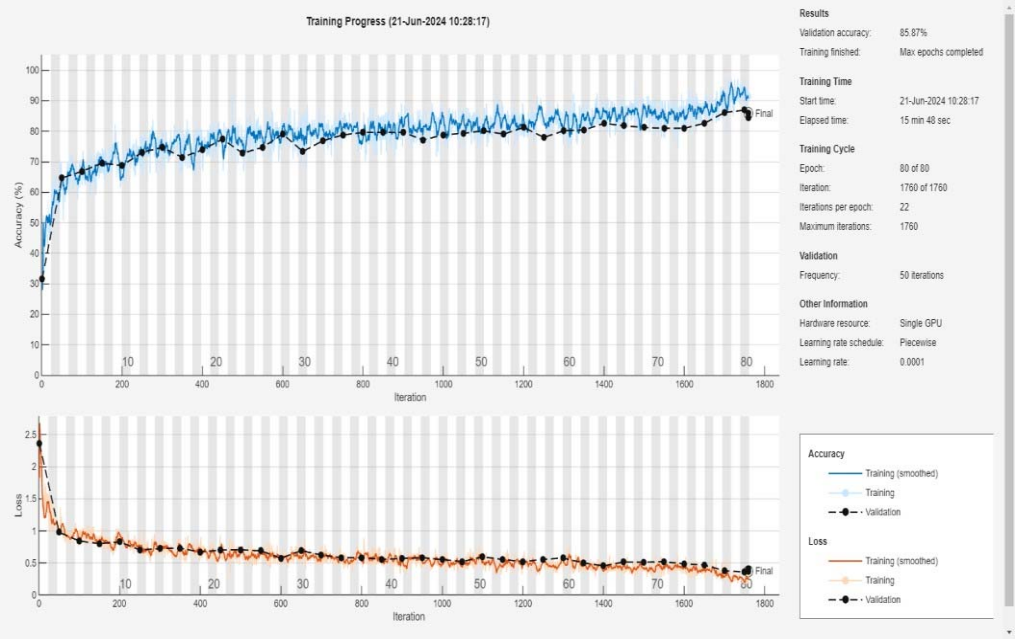
Πίνακας 3.3.4.στ: Μετρήσεις απόδοσης Fold 6

Fold 6
Training accuracy: 91.41%
Validation accuracy: 86.14%
Training loss: 0.33%
Validation loss: 0.41%
Test set accuracy: 88.83%

Στην 6η επαναδειγματοληψία το μοντέλο παρουσιάζει καλή ακρίβεια στο σύνολο εκπαίδευσης (91.41%) με ελαφρώς βελτιωμένη ακρίβεια στο σύνολο επικύρωσης (86.14%) και σταθερή γενίκευση σε αόρατα δεδομένα, με την ακρίβεια του συνόλου δοκιμής να φτάνει στο 88.83%. Η διαφορά στις τιμές των απωλειών των δυο συνόλων επικύρωσης και εκπαίδευσής δείχνουν την ύπαρξη υπερπροσαρμογής.

Fold 7.

Παρακάτω ακολουθεί η γραφική παράσταση της προόδου εκπαίδευσης και ο πίνακας με τις τιμές μετρήσεων της 7ης επαναδειγματοληψίας.



Σχήμα 3.3.4.ζ: Πρόοδος εκπαίδευσης Fold 7

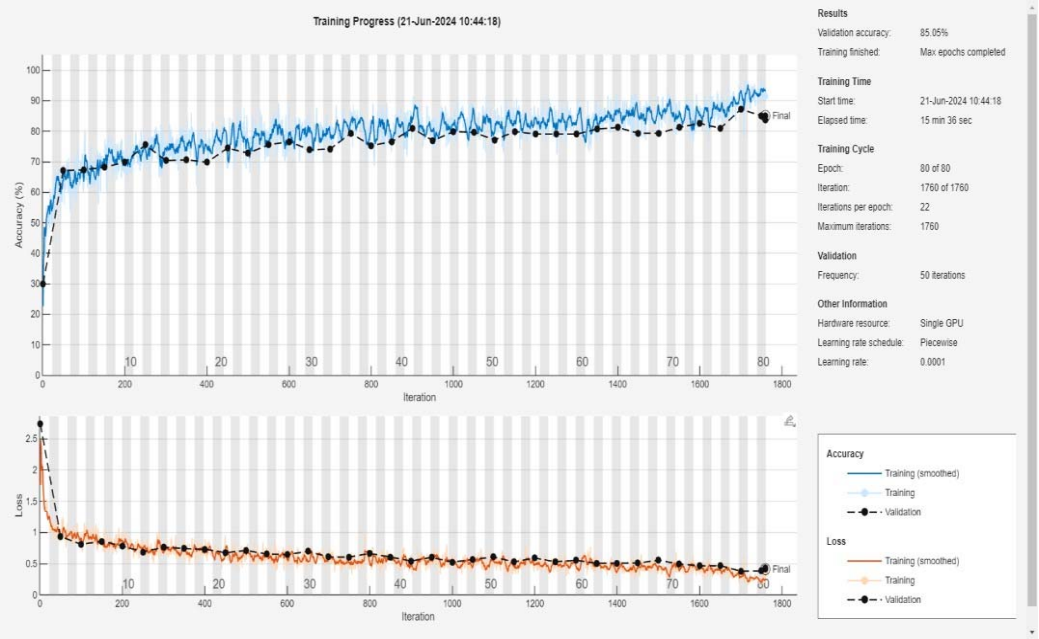
Πίνακας 3.3.4.ζ: Μετρήσεις απόδοσης Fold 7

Fold 7
Training accuracy: 91.41%
Validation accuracy: 84.51%
Training loss: 0.34%
Validation loss: 0.41%
Test set accuracy: 86.38%

Η υψηλή ακρίβεια εκπαίδευσης 91,41% δηλώνει αποτελεσματική εκμάθηση από τα δεδομένα εκπαίδευσης ενώ η ακρίβεια επικύρωσης (84.51%) σε σύγκριση με την ακρίβεια εκπαίδευσης υποδηλώνει την ύπαρξη υπερπροσαρμογής, η οποία επιβεβαιώνεται και από τις υψηλές απώλειες του συνόλου επικύρωσης (0.41%) και του συνόλου εκπαίδευσης (0.34%).

Fold 8.

Παρακάτω δίνεται η γραφική παράσταση της προόδου εκπαίδευσης και ο πίνακας με τις τιμές μετρήσεων της 8^{ης} επαναδειγματοληψίας.



Σχήμα 3.3.4.η: Πρόοδος εκπαίδευσης Fold 8

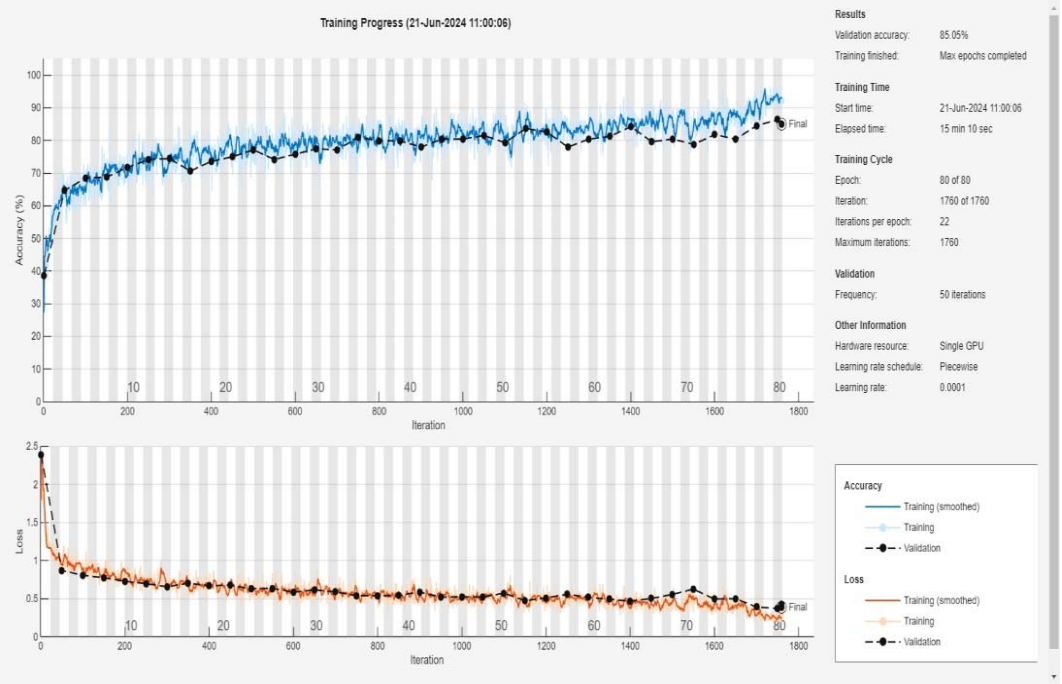
Πίνακας 3.3.4.η: Μετρήσεις απόδοσης Fold 8

Fold 8
Training accuracy: 91.41%
Validation accuracy: 83.70%
Training loss: 0.34%
Validation loss: 0.43%
Test set accuracy: 89.10%

Η ακρίβεια εκπαίδευσης (91.41%) είναι υψηλότερη από την ακρίβεια επικύρωσης (83.70%), πράγμα που δείχνει πιθανή υπερπροσαρμογή. Στην περίπτωση αυτή η ακρίβεια δοκιμής είναι υψηλότερη (89.10%) από εκείνη της επικύρωσης, υποδηλώνοντας πολύ καλή απόδοση σε αόρατα δεδομένα. Η διαφορά της απώλειας του συνόλου εκπαίδευσης (0.34%) και της απώλειας του συνόλου επικύρωσης δείχνει ότι το δίκτυο υπερπροσαρμόζεται στα δεδομένα επικύρωσης.

Fold 9.

Ακολουθεί η γραφική παράσταση της προόδου εκπαίδευσης και ο πίνακας με τις τιμές μετρήσεων της 9ης επαναδειγματοληψίας.



Σχήμα 3.3.4.θ: Πρόοδος εκπαίδευσης Fold 9

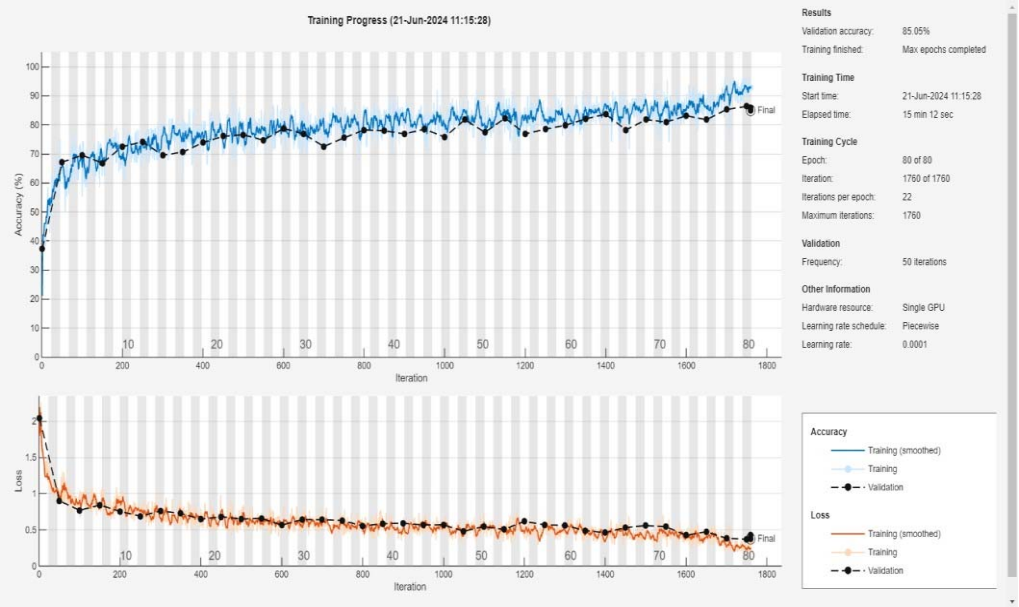
Πίνακας 3.3.4.θ: Μετρήσεις απόδοσης Fold 9

Fold 9
Training accuracy: 92.19%
Validation accuracy: 85.05%
Training loss: 0.32%
Validation loss: 0.43%
Test set accuracy: 86.65%

Το μοντέλο παρουσιάζει υψηλή ακρίβεια στο σύνολο εκπαίδευσης (92.19%) και σταθερή ακρίβεια στο σύνολο επικύρωσης (85.05%), υποδηλώνοντας υπερπροσαρμογή, ενώ η ακρίβεια δοκιμής (86.65%) είναι υψηλότερη από την ακρίβεια επικύρωσης, υποδηλώνοντας ελάχιστα καλύτερη γενίκευση σε νέα δεδομένα. Η απώλεια του συνόλου επικύρωσης (0.43%) είναι υψηλή σε σχέση με την απώλεια εκπαίδευσης (0.32%), που δείχνει ότι το μοντέλο δεν γενικεύει καλά τα δεδομένα επικύρωσης.

Fold 10.

Παρακάτω ακολουθούν η γραφική παράσταση της προόδου εκπαίδευσης και ο πίνακας με τις τιμές μετρήσεων του Fold 10.



Σχήμα 3.3.4.ι: Πρόοδος εκπαίδευσης Fold 10

Πίνακας 3.3.4.ι:. Μετρήσεις απόδοσης Fold 10

Fold 10
Training accuracy: 92.97%
Validation accuracy: 85.87%
Training loss: 0.34%
Validation loss: 0.43%
Test set accuracy: 87.19%

Η υψηλή ακρίβεια προπόνησης (92,97) δηλώνει αποτελεσματική εκμάθηση και αποτύπωση των δεδομένων εκπαίδευσης, ενώ η χαμηλότερη ακρίβεια επικύρωσης (85,87) καθώς και η τιμή απώλειας στα δεδομένα επικύρωσης (0.43%), που είναι υψηλότερη από την απώλεια εκπαίδευσης (0.34%), δείχνουν πιθανή υπερπροσαρμογή στο σύνολο επικύρωσης σε σύγκριση με τα δεδομένα εκπαίδευσης.

Τελικές τιμές απόδοσης του CNN

Ακολουθεί ο πίνακας που περιλαμβάνει τις τελικές τιμές απόδοσης, δηλαδή τον μέσο όρο των ποσοστών των μετρήσεων απόδοσης του μοντέλου μετά από την εφαρμογή της διασταυρούμενης επικύρωσης k-folds.

Πίνακας 3.3.4.κ: Τελικές τιμές απόδοσης των 10 - Folds

Final Values 10 Folds
Average Training set accuracy: 91.56%
Average Validation set accuracy: 85.54%
Average Training set loss: 0.33%
Average Validation set loss: 0.41%
Average Test set accuracy: 87.33%

Συμπεράσματα

- Το μοντέλο επιδεικνύει ισχυρή απόδοση με υψηλό ποσοστό εκπαίδευσης (91.56%) και καλό ποσοστό ακρίβειας επικύρωσης (85.54%).
- Η ελαφρά αύξηση στην απώλεια επικύρωσης (0.41%) σε σύγκριση με την απώλεια εκπαίδευσης (0.33%) υποδηλώνει μικρή υπερπροσαρμογή, η οποία είναι τυπική και διαχειρίσιμη.
- Η ακρίβεια του συνόλου δοκιμής (87.33%) ακολουθεί πιστά την ακρίβεια επικύρωσης (85.54%), υποδηλώνοντας ελαφρώς καλύτερη γενίκευση σε αόρατα δεδομένα.

3.4. Μεταφορά μάθησης προεκπαιδευμένου μοντέλου CNN (Pretrained CNN Transfer Learning)

Η μεταφορά γνώσεων είναι μια τεχνική βαθιάς μάθησης, όπου ένα προεκπαιδευμένο μοντέλο, που έχει αναπτυχθεί για μια εργασία, επαναχρησιμοποιείται ως το σημείο εκκίνησης για τη δημιουργία ενός νέου μοντέλου σε μια καινούργια εργασία. Αξιοποιεί τη γνώση που αποκτήθηκε κατά την επίλυση ενός προβλήματος και την εφαρμόζει σε ένα διαφορετικό αλλά σχετικό πρόβλημα. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη όταν η νέα εργασία έχει περιορισμένα δεδομένα, καθώς μπορεί να επωφεληθεί από τα ήδη μαθημένα μοτίβα και τα χαρακτηριστικά του προεκπαιδευμένου μοντέλου.

3.4.1. Βασικές έννοιες στη μεταφορά μάθησης

1. **Προεκπαιδευμένο μοντέλο**: είναι ένα μοντέλο που έχει προηγουμένως εκπαιδευτεί σε ένα μεγάλο σύνολο δεδομένων, συνήθως σε μια εργασία παρόμοια με αυτήν που διαθέτουμε.
2. **Fine-Tuning**: Προσαρμογή του προεκπαιδευμένου μοντέλου ώστε να ταιριάζει καλύτερα στη νέα εργασία. Αυτό μπορεί να περιλαμβάνει επανεκπαίδευση ορισμένων ή όλων των επιπέδων του μοντέλου στο νέο σύνολο δεδομένων.
3. **Εξαγωγή δυνατοτήτων**: Χρήση του προεκπαιδευμένου μοντέλου για εξαγωγή χαρακτηριστικών από το νέο σύνολο δεδομένων χωρίς περαιτέρω εκπαίδευση του μοντέλου. Τα εξαγόμενα χαρακτηριστικά χρησιμοποιούνται στη συνέχεια για την εκπαίδευση ενός νέου μοντέλου.

Παράδειγμα ροής εργασίας στο Transfer Learning:

1. **Επιλογή ενός προεκπαιδευμένου μοντέλου**: Επιλέγουμε ένα μοντέλο που έχει εκπαιδευτεί σε ένα μεγάλο και σχετικό σύνολο δεδομένων η ένα μοντέλο που έχει εκπαιδευτεί σε ένα παρόμοιο σύνολο δεδομένων.
2. **Τροποποίηση του μοντέλου**: Αντικαθιστούμε τα τελικά επίπεδα του μοντέλου ώστε να ταιριάζουν με τον αριθμό των κλάσεων ή τη συγκεκριμένη έξοδο, παγώνουμε στρώματα του δικτύου και προσαρμόζουμε το μοντέλο μας στις ανάγκες της νέας εργασίας.
3. **Βελτίωση του μοντέλου**: Το τροποποιημένο μοντέλο στο νέο σύνολο δεδομένων συχνά εκπαιδεύεται με χαμηλότερο ρυθμό εκμάθησης για να αποφύγουμε δραστικές αλλαγές στα προεκπαιδευμένα βάρη.
4. **Αξιολόγηση και προσαρμογή**: Αξιολόγηση του προεκπαιδευμένου μοντέλου ως προς την απόδοση του σε καινούργια δεδομένα.

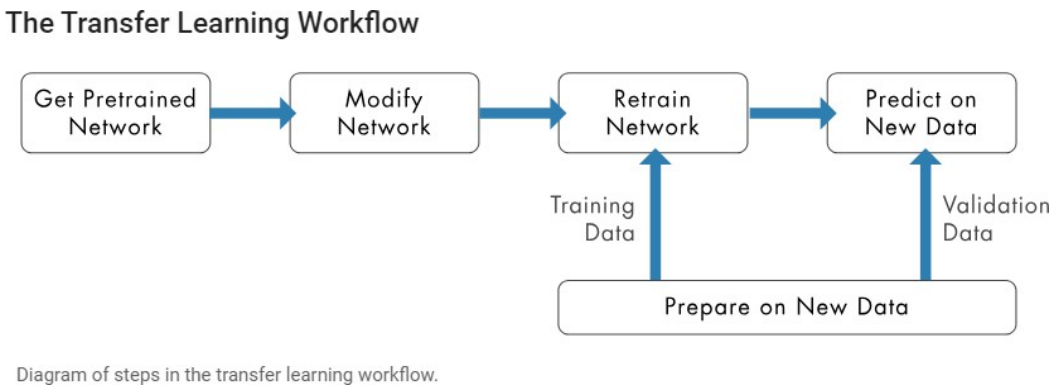


Diagram of steps in the transfer learning workflow.

Σχήμα. 3.4.1.α: Διάγραμμα ροής της διαδικασίας μεταφοράς μάθησης.

Εφαρμογές Transfer Learning:

1. **Ταξινόμηση εικόνων**: Μοντέλα όπως το VGG, το ResNet και το Inception που έχουν εκπαιδευτεί στο ImageNet χρησιμοποιούνται συχνά ως σημεία εκκίνησης για διάφορες εργασίες ταξινόμησης εικόνων.
2. **Επεξεργασία φυσικής γλώσσας**: Μοντέλα όπως το BERT, το GPT και το ELMo, προεκπαιδευμένα σε μεγάλα σώματα κειμένου, είναι βελτιωμένα για εργασίες όπως η ανάλυση συναισθήματος, η απάντηση ερωτήσεων και η ταξινόμηση κειμένου.
3. **Αναγνώριση ομιλίας**: Τα προεκπαιδευμένα μοντέλα σε μεγάλα σύνολα δεδομένων ομιλίας μπορούν να προσαρμοστούν σε συγκεκριμένες γλώσσες ή διαλέκτους.

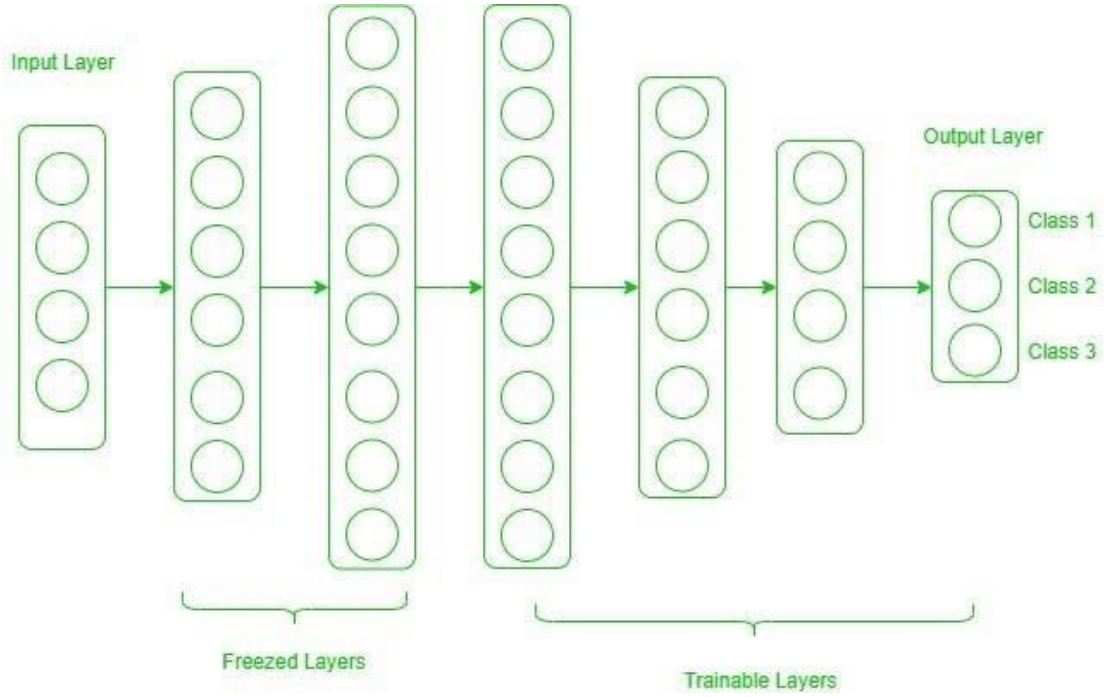
Οφέλη της Εκμάθησης Μεταβίβασης:

1. **Μειωμένος χρόνος εκπαίδευσης**: Εφόσον το μοντέλο έχει ήδη μάθει χρήσιμα χαρακτηριστικά, η εκπαίδευση στη νέα εργασία απαιτεί λιγότερο χρόνο και υπολογιστικούς πόρους.
2. **Βελτιωμένη απόδοση**: Ειδικά όταν η νέα εργασία έχει περιορισμένα δεδομένα, η εκμάθηση μεταφοράς μπορεί να οδηγήσει σε καλύτερη απόδοση αξιοποιώντας τις αναπαραστάσεις εμπλουτισμένων χαρακτηριστικών που αντλήθηκαν από το μεγαλύτερο σύνολο δεδομένων.
3. **Λιγότερες απαιτήσεις δεδομένων**: Η αποτελεσματική χρήση της εκμάθησης μεταφοράς μπορεί να μειώσει τον όγκο των δεδομένων με ετικέτα που απαιτούνται για τη νέα εργασία.

3.4.2. Μεταφορά μάθησης και το πάγωμα των στρωμάτων (Transfer Learning Freeze Layers)

Η μεταφορά γνώσεων προεκπαιδευμένου νευρωνικού δικτύου με χρήση παγωμένων στρωμάτων είναι μια δημοφιλής τεχνική στη βαθιά μάθηση για την αξιοποίηση υπαρχόντων, καλά εκπαιδευμένων μοντέλων. Μοντέλα όπως το Googlenet, squeezenet, ResNet50, inceptionv3, που έχουν προηγουμένως αποκτήσει γνώση πάνω σε μεγάλα και διαφορετικά σύνολα δεδομένων και έχουν ήδη μάθει να εντοπίζουν μια ποικιλία χαρακτηριστικών όπως άκρες υφές και μοτίβα τα οποία είναι χρήσιμα για πολλές διαφορετικές αναγνωρίσεις εικόνων.

Το "Freezing Layers" στο πλαίσιο της μεταφοράς γνώσης αναφέρεται στην πρακτική αποτροπής της ενημέρωσης ορισμένων στρωμάτων ενός προεκπαιδευμένου νευρωνικού δικτύου κατά τη διάρκεια της εκπαιδευτικής διαδικασίας σε μια νέα εργασία. Αυτή η τεχνική είναι χρήσιμη για τη διατήρηση των χαρακτηριστικών που είχαν μάθει προηγουμένως και για τη μείωση του υπολογιστικού κόστους. Στο παρακάτω σχήμα αναπαρίσταται ένα δίκτυο με παγωμένα τα αρχικά στρωματά, που χρησιμοποίει μόνο τα τελευταία επίπεδα για την εκπαίδευση, ταξινομώντας τα αποτελέσματα σε κλάσεις.



Σχήμα. 3.4.2.α: Αρχιτεκτονική ενός δικτύου με παγωμένα στρώματα.

Γιατί παγώνουν τα στρώματα;

1. Για να διατηρηθούν τα χαρακτηριστικά μάθησης (To Preserve Learned Features): Σε ένα προεκπαιδευμένο μοντέλο, τα αρχικά στρώματα καταγράφουν γενικά χαρακτηριστικά (π.χ. άκρες, υφές σε μοντέλα εικόνας), ενώ τα υπόλοιπα στρώματα καταγράφουν περισσότερες λειτουργίες, που αφορούν συγκεκριμένες εργασίες. Παγώνοντας τα αρχικά επίπεδα, διατηρούνται αυτές οι γενικές δυνατότητες που είναι συχνά χρήσιμες για πολλές και διαφορετικές εργασίες.

2. Για να μειωθεί η υπερπροσαρμογή (To reduce Overfitting): Όταν το σύνολο δεδομένων είναι περιορισμένο ως προς τα χαρακτηριστικά του για την εργασία που το μοντέλο καλείται να πραγματοποιήσει το συνελικτικό δίκτυο, το πάγωμα των στρωμάτων βοηθά στη μείωση του κινδύνου της υπερπροσαρμογής. Το μοντέλο δεν προσαρμόζεται πάρα πολύ στα περιορισμένα δεδομένα, διατηρώντας έτσι τη γενικότητα των μαθησιακών χαρακτηριστικών.

3. Ταχύτερους χρόνους εκπαίδευσης (Speed Up Training): Με τη μη ενημέρωση των βαρών των παγωμένων στρωμάτων το υπολογιστικό κόστος μειώνεται, οδηγώντας σε ταχύτερους χρόνους εκπαίδευσης.

3.4.3 Freezy Layers χρησιμοποιώντας το ίδιο σύνολο δεδομένων

Τεχνικά είναι εφικτό να παγώνουμε τα στρωματά ενός προεκπαιδευμένου μοντέλου και να το επανεκπαιδεύουμε μόνο με το επίπεδο ταξινόμησης, χρησιμοποιώντας το ίδιο σύνολο δεδομένων. Μερικές φορές αυτή η τακτική είναι και χρήσιμη υπό συγκεκριμένες συνθήκες, όπως:

1. **Βελτιστοποίηση ενός πολύπλοκου μοντέλου με περιορισμένα δεδομένα.**
   Σε ένα πολύπλοκο μοντέλο και περιορισμένο όγκο δεδομένων η επανεκπαίδευση ολοκλήρου του μοντέλου θα μπορούσε να οδηγήσει σε υπερπροσαρμογή, το πάγωμα των προεκπαιδευμένων στρωμάτων συμβάλλει στο μετριασμό αυτού του κίνδυνου και στη διατήρηση των χαρακτηριστικών των δεδομένων.
2. **Σταθερότητα και ταχύτητα**
   Τα βάρη του προεκπαιδευμένου μοντέλου παραμένουν σταθερά, αυτό βοηθά στη διατήρηση της σταθερότητας των μαθησιακών χαρακτηριστικών και αποφεύγεται ο κίνδυνος καταστροφικής λήθης (όπου το μοντέλο ξεχνά ό,τι έχει μάθει).

Μετά την εκπαίδευση του CNN μοντέλου μας, το αποθηκεύσαμε για να το επαναχρησιμοποιήσουμε για αρχή στο ίδιο σύνολο δεδομένων και να παρατηρήσουμε τη συμπεριφορά του. Το φορτώσαμε και παγώσαμε όλα τα επίπεδα εκτός από τα πλήρως συνδεδεμένα επίπεδα και το στρώμα εξόδου. Επειδή θα χρησιμοποιήσουμε το ίδιο σύνολο δεδομένων που χρησιμοποιήθηκε και για να εκπαιδευτεί το μοντέλο, το στρώμα εξόδου δεν χρειάζεται να αντικατασταθεί. Μειώσαμε την τιμή εκμάθησης στο 0.0001 από 0.001 του προεκπαιδευμένου CNN και ο χρόνος επανεκπαίδευσης του μοντέλου μειώθηκε στα 3 λεπτά και 58 δευτερόλεπτα (15 εποχές) από τα 14 λεπτά και 49 δευτερόλεπτα (80 εποχές), αρχικός χρόνος που χρειάστηκε το μοντέλο να εκπαιδευτεί. Δεν χρησιμοποιήσαμε τη μέθοδο της επαύξησης δεδομένων σε κανένα από τα υποσύνολα των δεδομένων (εκπαίδευσής, επικύρωσης, δοκιμής).

Στο παρακάτω Σχήμα **3.4.3.α.** παρουσιάζεται η πρόοδος εκπαίδευσης του προεκπαιδευμένου μοντέλου στο ίδιο σύνολο δεδομένων μετά από την εφαρμογή της τεχνικής freezy layers.

Σχήμα 3.4.3.α: Πρόοδος επανεκπαίδευσης του μοντέλου

Στον παρακάτω πίνακα παρουσιάζονται οι τιμές των μετρήσεων απόδοσης και των δυο δικτύων.

Πίνακας 3.4.3.α: Τελικές τιμές απόδοσης επανεκπαίδευσης του μοντέλου

Pretrained Net	Freezy Layers Net with the same data set
Training accuracy: 88.28%	Training accuracy: 93.75%
Training loss: 0.26%	Training loss: 0.20%
Validation accuracy: 89.13%	Validation accuracy: 94.57%
Validation loss: 0.31%	Validation loss: 0.18%
Test set accuracy: 88.83%	Test set accuracy: 93.73%

Τα αποτελέσματα των μετρήσεων απόδοσης έδειξαν ότι:

Η επανεκπαίδευση του προεκπαιδευμένου στο ίδιο σύνολο δεδομένων μοντέλου ήταν εξαιρετικά αποτελεσματική. Το μοντέλο επιδεικνύει εξαιρετική απόδοση, με υψηλή ακρίβεια τόσο στο σύνολο εκπαίδευσης (93,75), όσο και στο σύνολο επικύρωσης (94.57%). Χαμηλές ήταν οι τιμές της απώλειας εκπαίδευσης (0.20%) και της απώλειας επικύρωσης, υποδηλώνοντας επιτυχή μάθηση και γενίκευση.

Το δίκτυο δείχνει καλύτερη γενίκευση, όπως φαίνεται από την υψηλότερη ακρίβεια επικύρωσης (94.57%) και δοκιμής (93.73%). Το μοντέλο αποδίδει σταθερά καλά όχι μόνο στα δεδομένα εκπαίδευσης αλλά και σε μη ορατά δεδομένα (δεδομένα επικύρωσης και δεδομένα δοκιμής).

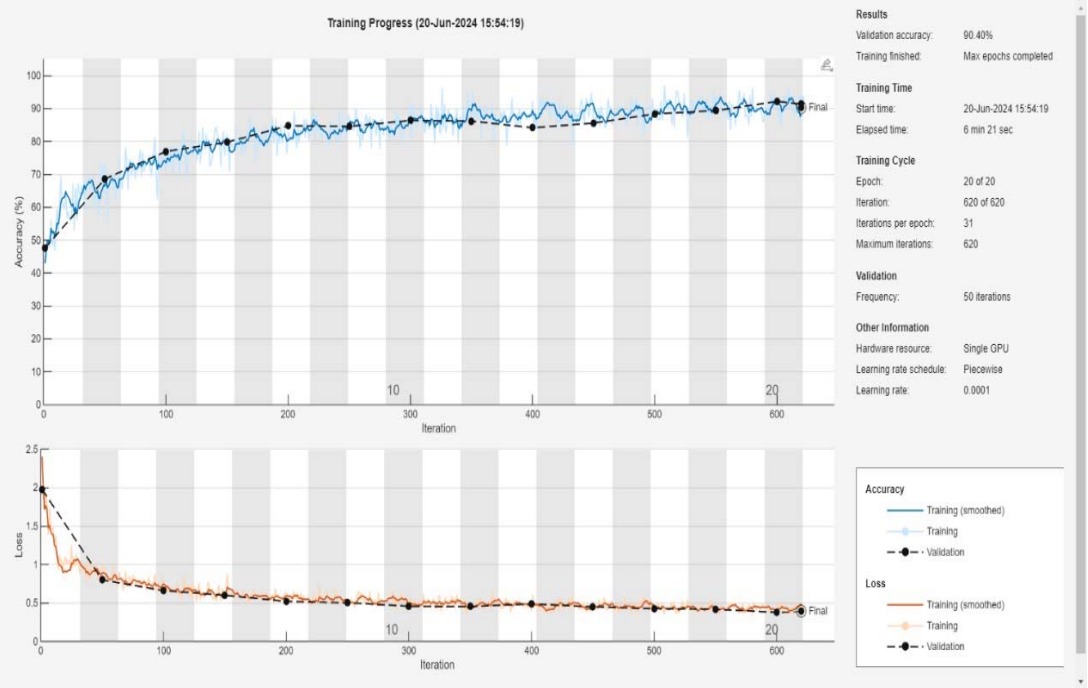
3.4.4. Freezy Layers διαφορετικού συνόλου δεδομένων

Για να εφαρμόσουμε εκ νέου την τεχνική του παγώματος των στρωμάτων επιλέξαμε ένα καινούργιο σύνολο δεδομένων παρόμοιο με το αρχικό σύνολο των δεδομένων που χρησιμοποιήθηκε κατά την εκπαίδευση του προεκπαιδευμένου μοντέλου μας. Το καινούργιο μας σύνολο δεδομένων περιλαμβάνει 5000 εικόνες διαφορετικού μεγέθους μοιρασμένες κατά 1000 σε πέντε διαφορετικές κλάσεις:

1. Lilly
2. Lotus
3. Sunflower
4. Orchid
5. Tulip

Εικόνα 3.4.4.α: Το καινούργιο σύνολο δεδομένων

Φορτώσαμε το προεκπαιδευμένου δίκτυο και το καινούργιο σύνολο δεδομένων, παγώσαμε όλα τα στρώματα εκτός από τα πλήρως συνδεδεμένα στρώματα και το στρώμα εξόδου. Αντικαταστήσαμε το επίπεδο εξόδου του προεκπαιδευμένου δικτύου με καινούργιο στρώμα εξόδου και το επίπεδο εγκατάλειψης του προεκπαιδευμένου δικτύου με καινούργιο, ορίζοντας την πιθανότητα εγκατάλειψης στο 0.45. Στη συνέχεια ξεπαγώσαμε τα βαθύτερα στρώματα του μοντέλου και ορίσαμε τα βάρη και τις προκαταλήψεις στην τιμή 1, την αρχική τιμή εκπαίδευσης στην τιμή 0.0001 και το μοντέλο να εκπαιδευτεί για 20 εποχές. Στο **Σχήμα 3.4.4.α**. απεικονίζεται η πρόοδος του δικτύου των παγωμένων στρωμάτων κατά την εκπαίδευση.



Σχήμα 3.4.4.α: Πρόοδος εκπαίδευσης δικτύου παγωμένων στρωμάτων.

Στον παρακάτω πίνακα καταγράφονται οι μετρήσεις απόδοσης και των δυο δικτύων, του προεκπαιδευμένου και του δικτύου των παγωμένων στρωμάτων.

Πίνακας 3.4.4.α: Τελικές τιμές απόδοσης του Freezy Layers Net

Pretrained Net	Freezy Layers Net with different data set
Training accuracy: 88.28%	Training accuracy: 92.97%
Training loss: 0.26%	Training loss: 0.43%
Validation accuracy: 89.13%	Validation accuracy: 91.40%
Validation loss: 0.31%	Validation loss: 0.39%
Test set accuracy: 88.83%	Test set accuracy: 89.00%

Οι μετρήσεις απόδοσης έδειξαν ότι το μοντέλο γενικεύει καλυτέρα σε νέα δεδομένα από ότι στα δεδομένα εκπαίδευσης του προεκπαιδευμένου μοντέλου λόγω της υψηλότερης ακρίβειας εκπαίδευσής (92.97%) και επικύρωσης (91.40%) του Freeze net. Ωστόσο οι μεγαλύτερες απώλειές του δικτύου με τα παγωμένα στρώματα σε σχέση με το προεκπαιδευμένο έδειξαν ότι το ποσοστό εσφαλμένης ταξινόμησης του δικτύου αυξήθηκε. Η μικρή βελτίωση της ακρίβειας του συνόλου δοκιμής (89.00%) του δικτύου των παγωμένων στρωμάτων σε σχέση με την ακρίβεια του συνόλου δοκιμής του προεκπαιδευμένου δικτύου (88.83%) υποδηλώνει ότι το δίκτυο με τα παγωμένα επίπεδα είναι καλύτερο στο χειρισμό αοράτων δεδομένων.

3.5 Πρόβλεψη κρυπτονομισμάτων με ανατροφοδοτούμενα δίκτυα (Rnn Cryptocurrency Prediction)

Η πρόβλεψη στη βαθιά μάθηση αναφέρεται στη διαδικασία χρήσης νευρωνικών δικτύων για την εξαγωγή συμπερασμάτων ή προβλέψεων σχετικά με άγνωστα δεδομένα, με βάση μοτίβα που αντλήθηκαν από ιστορικά δεδομένα. Τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) είναι ένας τύπος μοντέλου βαθιάς μάθησης, που έχει σχεδιαστεί για να χειρίζεται διαδοχικά δεδομένα όπως είναι οι χρονοσειρές, η φυσική γλώσσα και δεδομένα ήχου. Είναι ιδιαίτερα χρήσιμα για την πραγματοποίηση προβλέψεων, όπου το τρέχον βήμα εξαρτάται από τα προηγούμενα βήματα.

Περιγραφή των Συνόλων Δεδομένων (Data Sets description)

Τα δεδομένα μας αφορούν τρία διαφορετικά κρυπτονομίσματα Bitcoin, Binance Coin (BNB), Ethereum classic, το κάθε ένα από αυτά αποθηκευμένο σε αρχείο με κατάληξη csv. Εμπεριέχουν ιστορικά δεδομένα τιμών (Historical Price Data) για κάθε κρυπτονόμισμα όπως:

Ημερομηνία (Date) - η ημερομηνία του καταγεγραμμένου σημείου δεδομένων

Άνοιγμα (Open) - η τιμή στο άνοιγμα της αγοράς τη συγκεκριμένη ημερομηνία

Υψηλό (High) - η υψηλότερη τιμή του κρυπτονομίσματος κατά τη διάρκεια αυτής της ημέρας.

Χαμηλό (Low) - η χαμηλότερη τιμή του κρυπτονομίσματος στην αναφερομένη ημερομηνία.

Κλείσιμο (Close) - η τιμή του κρυπτονομίσματος στο κλείσιμο της αγοράς εκείνη την ημέρα.

Όγκος (Volume) - Ο συνολικός όγκος του κρυπτονομίσματος που έγινε αντικείμενο διαπραγμάτευσης κατά τη διάρκεια αυτής της ημέρας.

Νόμισμα (Currency) - το νόμισμα στο οποίο εκφράζονται οι τιμές σε αυτή την περίπτωση.

Στον παρακάτω πίνακα ακολουθούν οι πρώτες πέντε χρονοσειρές του αρχείου με κατάληξη .cvs, το οποίο περιέχει τα δεδομένα του κρυπτονομίσματος bitcoin.

Πίνακας 3.5.α: Δείγμα Δεδομένων (Data Set Sample)

Date	Open	High	Low	Close	Volume	Currency
2010-07-18	0.0	0.1	0.1	0.1	75	USD
2010-07-19	0.1	0.1	0.1	0.1	574	USD

…….	…….	…….	…….	…….	…….	……...
2022-08-21	21138.9	21692.4	21077.4	21517.2	177522	USD
2022-08-22	21516.8	21517.4	20912.1	21416.3	251833	USD
2022-08-23	21416.5	21517.4	21271.2	21309.0	251695	USD

Στην ανάλυση χρηματοοικονομικών χρονοσειρών είναι σύνηθες να χρησιμοποιούνται διάφορα χαρακτηριστικά μιας δεδομένης χρονικής περιόδου για να προβλέψουμε ένα άλλο χαρακτηριστικό ενδιαφέροντος. Ο στόχος μας είναι να προβλέψουμε την τιμή Κλείσιμο (Close) η οποία είναι μια βασική αξία για τους έμπορους και του αναλυτές και αντιστοιχεί στην τελική τιμή του κρυπτονομίσματος στο τέλος της χρονικής περιόδου. Συχνά θεωρείται κρίσιμος δείκτης του κλίματος της αγοράς.

Στην περίπτωση πρόβλεψής της τιμής Κλείσιμο (Close) επιλέγουμε εκείνα τα χαρακτηριστικά τα οποία παρέχουν ολοκληρωμένες πληροφορίες σχετικά με την αγορά των κρυπτονομισμάτων σε μια δεδομένη χρονική περίοδο:

- Άνοιγμα (Open): Η τιμή ανοίγματος στην αρχή της χρονικής περιόδου.
- Υψηλό (High): Η υψηλότερη τιμή κατά τη διάρκεια της χρονικής περιόδου.
- Χαμηλό (Low): Η χαμηλότερη τιμή κατά τη διάρκεια της χρονικής περιόδου.
- Κλείσιμο (Close): Η τελική τιμή του κρυπτονομίσματος στο τέλος της ημέρας.
- Όγκος (Volume): Το ποσό του κρυπτονομίσματος που έγινε αντικείμενο διαπραγμάτευσης κατά τη διάρκεια της χρονικής περιόδου.

Στη συνέχεια χωρίσαμε το κάθε σύνολο δεδομένων σε δυο υποσύνολα, υποσύνολο εκπαίδευσής και υποσύνολο επικύρωσης και εφαρμόζουμε στα δεδομένα τεχνικές όπως:

1. Ταξινόμηση των δεδομένων με βάση την ημερομηνία.
2. Αφαίρεση των γραμμών του πίνακα στις οποίες οι στήλες περιέχουν τιμή NAN.
3. Κατάργηση των γραμμών του πίνακα οπού τα χαρακτηριστικά παραμένουν σταθερά για όλα τα χρονικά βήματα.
4. Εφαρμογή της τεχνικής της ομαλοποίησης z-score.

Για να πέτυχουμε το στόχο μας, να μπορέσουμε να κάνουμε πρόβλεψη χρηματοοικονομικών χρονοσειρών, δημιουργήσαμε δυο μοντέλα RNN ένα LSTM και ένα GRU. Στη συνέχεια θα περιγράψουμε και τις δυο αυτές αρχιτεκτονικές.

3.6. Το LSTM μοντέλο

Τα δίκτυα Long Short -Term Memory (LSTM) είναι ένας τύπος επαναλαμβανόμενου νευρωνικού δικτύου (RNN), που έχει σχεδιαστεί για να ξεπερνά τους περιορισμούς των παραδοσιακών RNN, ιδιαίτερα την αδυναμία τους να μάθουν μακροπρόθεσμες εξαρτήσεις λόγω ζητημάτων όπως η εξαφάνιση και η έκρηξη κλίσεων. Τα LSTM που εισήχθησαν από τους Hochreiter και Schmidhuber το 1997, έχουν μια μοναδική αρχιτεκτονική, που τους επιτρέπει να συλλαμβάνουν και να θυμούνται αποτελεσματικά

πληροφορίες σε εκτεταμένες ακολουθίες δεδομένων. Αυτό επιτυγχάνεται μέσω μιας σειράς πυλών (πύλες εισόδου, λήθης και εξόδου), που ρυθμίζουν τη ροή πληροφοριών εντός του δικτύου, επιτρέποντάς του να διατηρεί και να ενημερώνει μια κατάσταση κυψέλης με την πάροδο του χρόνου. Ως αποτέλεσμα, τα LSTM χρησιμοποιούνται ευρέως σε εφαρμογές που περιλαμβάνουν διαδοχικά δεδομένα, όπως η πρόβλεψη χρονοσειρών, η επεξεργασία φυσικής γλώσσας (NLP) και η αναγνώριση ομιλίας.

Η αρχιτεκτονική του μοντέλου μας LSTM έχει σχεδιαστεί για μια εργασία πρόβλεψης χρονοσειρών και αποτελείται από τέσσερα επίπεδα:

1. Επίπεδο Εισαγωγής Ακολουθίας (Sequence Input Layer) - το συγκεκριμένο επίπεδο ορίζεται από το πεδίο inputSize που καθορίζει τον αριθμό των χαρακτηριστικών εισόδου σε κάθε χρονικό βήμα. Στην περίπτωση μας τα χαρακτηριστικά είναι πέντε. Αυτό το στρώμα δέχεται ακολουθίες διαφορετικού μήκους και τις τροφοδοτεί στο στρώμα LSTM.
2. Επίπεδο LSTM (LSTM Layer) - αποτελεί τον πυρήνα του μοντέλου και συμπεριέχει πενήντα κρυφές μονάδες, οι οποίες μπορούν να συλλάβουν μακροπρόθεσμες εξαρτήσεις.
3. Πλήρως συνδεδεμένο επίπεδο (Fully Connected Layer) - ένα πυκνό στρώμα που αντιστοιχίζει την έξοδο από το επίπεδο LSTM στο επιθυμητό μέγεθος εξόδου. Το μέγεθος εξόδου καθορίζει τον αριθμό τον νευρώνων, ο οποίος συνηθώς ταιριάζει με τον αριθμό των χαρακτηριστικών εξόδου που θέλουμε να προβλέψουμε. Στην περίπτωση μας το μέγεθος εξόδου είναι ένας νευρώνας που θα αποθηκεύσει την πρόβλεψη της τιμής Close.
4. Επίπεδο Εξόδου (Regression Layer) - χρησιμοποιείται για εργασίες παλινδρόμησης.

Επιλέξαμε το μοντέλο μας να εκπαιδευτεί ανάλογα με διαφορετικό αριθμό εποχών και ξεχωριστό ρυθμό εκμάθησης για κάθε κρυπτονόμισμα, χρησιμοποιώντας τον αλγόριθμο βελτιστοποίησης Adam (Adaptive Moment Estimation), ο οποίος υπολογίζει προσαρμοστικούς ρυθμούς για κάθε παράμετρο. Η τεχνική κανονικοποίησης L2, ο συντελεστής αποσύνθεσης τετραγωνικής κλήσης (Square gradient decay factor) έχουν οριστεί με διαφορετικές τιμές για κάθε κρυπτονόμισμα.

Παρακάτω ακολουθούν οι τιμές εκπαίδευσης για κάθε κρυπτονόμισμα:

Bitcoin

Ρυθμός εκμάθησης - 0.0001.

Εποχές - 20.

Κανονικοποίηση L2 - 0.

Συντελεστής αποσύνθεσης - 0.999.

Μέγεθος μίνι παρτίδας – 64.

Περίοδος πτώσης ρυθμού εκμάθησης – 10.

Ποσοστό πτώσης ρυθμού εκμάθησης - 0.5.

Μέγεθος ακολουθίας – 5.

Ethereum

Ρυθμός εκμάθησης - 0.005

Εποχές - 50.

Κανονικοποίηση L2 - 0.01.

Συντελεστής αποσύνθεσης - 0.9.

Μέγεθος μίνι παρτίδας - 64.

Μέγεθος ακολουθίας - 5.

Bnb

Ρυθμός εκμάθησης - 0.002

Εποχές - 50.

Κανονικοποίηση L2 - 0.01.

Συντελεστής αποσύνθεσης - 0.9.

Μέγεθος μίνι παρτίδας - 64.

Μέγεθος ακολουθίας - 5.

3.6.1. Εκπαίδευση και αξιολόγηση του LSTM μοντέλου

Τα αποτελέσματα της εκπαίδευσης για το κάθε ένα από τα κρυπτονομίσματα έδειξαν ότι:

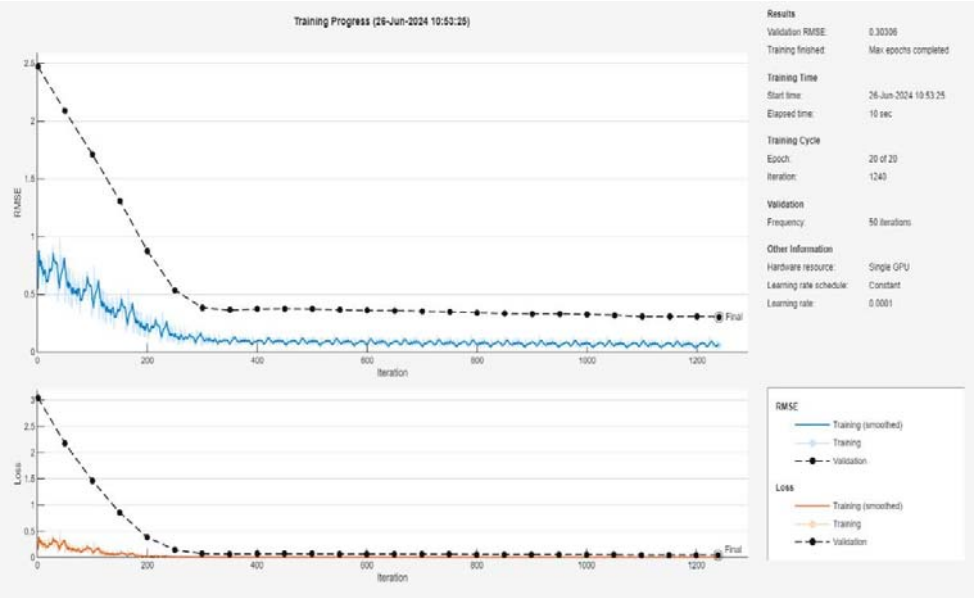
1. Για το bitcoin οι μετρήσεις απόδοσης έδειξαν ότι το μέσο τετραγωνικό σφάλμα /απώλεια (MSE) του συνόλου εκπαίδευσής (0.0016) είναι χαμηλότερο από την απώλεια (MSE) του συνόλου επικύρωσης (0.0459) καθώς και η τελική τιμή μέσου σφάλματος πρόβλεψης (RMSE) του συνόλου εκπαίδευσης είναι κατά πολύ χαμηλότερη (0,0561) από την τιμή μέσου σφάλματος πρόβλεψης του συνόλου επικύρωσης (0,3031), υποδηλώνοντας ότι το μοντέλο προσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης, με αποτέλεσμα κακή γενίκευση στο σύνολο επικύρωσης.

   Οι αναλυτικές τιμές της απόδοσης του μοντέλου όσον αφορά το Bitcoin είναι:

Πίνακας 3.6.1.α: Μετρήσεις αξιολόγησης LSTM - Bitcoin

Final Values Bitcoin
Final Training Loss: 0.0016
Final Training RMSE: 0.0561
Final Validation Loss: 0.0459
Final Validation RMSE: 0.3031

Στο παρακάτω σχήμα ακολουθεί η γραφική παράσταση της προόδου εκπαίδευσης του LSTM για το Bitcoin με τη μεγάλη διαφορά των τιμών MSE και των τιμών RMSE αναμεσά στα δυο σύνολα μέσω των καμπύλων εκπαίδευσης.



Σχήμα 3.6.1.α: Πρόοδος εκπαίδευσης LSTM μοντέλου για το Bitcoin.

Για την ανάλυση του συνόλου των δεδομένων που αφορούν το Bitcoin το μοντέλο Garch θα ήταν ιδανικό, διότι κάθε προσπάθεια μας να πέτυχουμε μια καλή επίδοση του LSTM απέτυχε, δείχνοντας ότι υπάρχει μεταβλητότητα και αστάθεια των τιμών του συνόλου. Το μοντέλο Garch παρέχει μια ακριβή εκτιμήση της αστάθειας των τιμών του συνόλου δεδομένων Bitcoin, καθώς το μοντέλο καταγράφει μη γραμμικές σχέσεις μεταξύ των προηγουμένων τιμών και της μεταβλητότητας, κάτι που μπορεί να είναι χρήσιμο στη μοντελοποίηση της πολύπλοκης δυναμικής των τιμών Bitcoin. Με την ακριβή εκτιμήση της αστάθειας που πραγματοποιεί το μοντέλο Garch αξιολογούνται καλυτέρα οι κίνδυνοι που σχετίζονται με τις επενδύσεις του κρυπτονομίσματος.

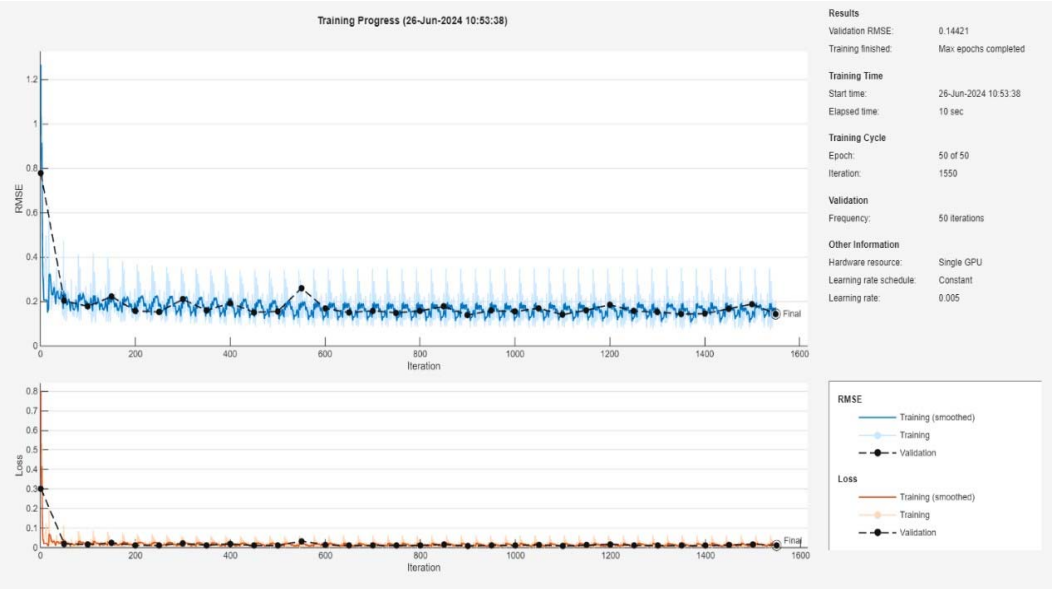
2. Ethereum :

Οι απώλειες εκπαίδευσης (0,0090) και επικύρωσης (0,0104) είναι σχετικά κοντινές, υποδηλώνοντας μια καλή ισορροπία μεταξύ της προσαρμογής των δεδομένων εκπαίδευσης και της γενίκευσης στο σύνολο επικύρωσης.
Οι τιμές RMSE είναι επίσης κοντινές, τόσο για το σύνολο εκπαίδευσης (0,1341) όσο και για το σύνολο επικύρωσης (0,1442), υποδηλώνοντας ότι τα σφάλματα πρόβλεψης του μοντέλου είναι συνεπή και στα δύο σύνολα. Ακολουθούν οι μετρήσεις απόδοσης του μοντέλου για το σύνολο των δεδομένων που αφορούν το Ethereum.

Πίνακας 3.6.1.β: Μετρήσεις αξιολόγησης LSTM - Ethereum

Final Values Ethereum
Final Training Loss: 0.0090
Final Training RMSE: 0.1341
Final Validation Loss: 0.0104
Final Validation RMSE: 0.1442

Συνολικά, το μοντέλο για το Ethereum φαίνεται να έχει καλή εφαρμογή χωρίς σημαντική υπερπροσαρμογή. Στο παρακάτω σχήμα απεικονίζεται η πρόοδος εκπαίδευσης του μοντέλου LSTM για το Ethereum.



Σχήμα 3.6.1.β: Πρόοδος κατά την εκπαίδευση του LSTM μοντέλου για το Ethereum

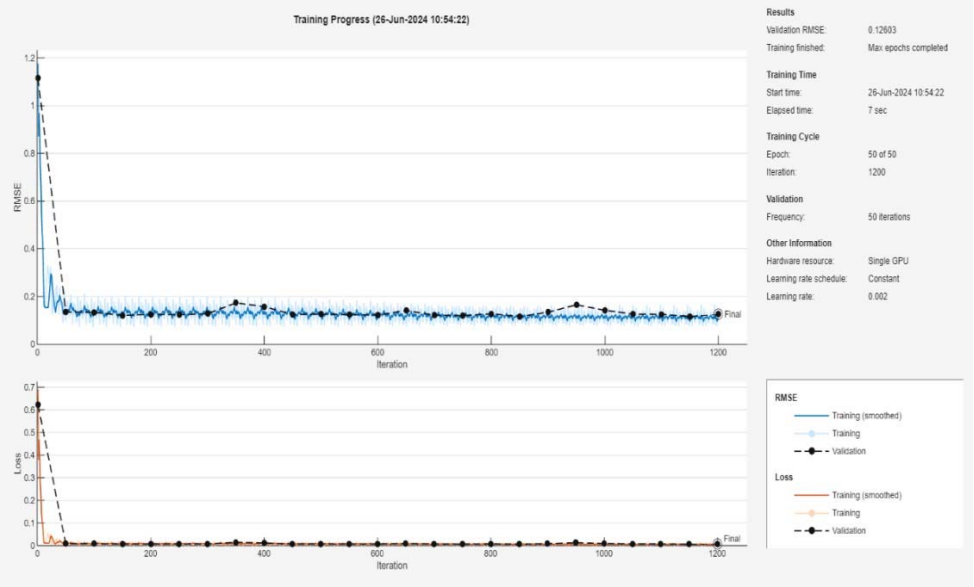
3. BNB :

Η απώλεια προπόνησης (0,0032) και η απώλεια επικύρωσης (0,0033) είναι σχεδόν ταυτόσημες. Αυτό υποδηλώνει εξαιρετική γενίκευση και στα δύο σύνολα. Το σφάλμα πρόβλεψης εκπαίδευσης RMSE (0,0803) και η επικύρωση RMSE (0,0815) είναι πολύ κοντά, υποδηλώνοντας ότι το μοντέλο έχει ισχυρή ικανότητα γενίκευσης. Ακολουθούν οι μετρήσεις απόδοσης του LSTM για το κρυπτονόμισμα BNB:

Πίνακας 3.6.1.γ: Μετρήσεις αξιολόγησης του LSTM - BNB

Final Values BNB
Final Training Loss: 0.0083
Final Training RMSE: 0.1288
Final Validation Loss: 0.0079
Final Validation RMSE: 0.1260

Παρακάτω ακολουθεί η απεικόνιση της γραφικής παράστασης της προόδου εκπαίδευσης του μοντέλου για το BNB δείχνοντας ελάχιστες ταλαντώσεις των καμπυλών.



Σχήμα 3.6.1.γ:. Πρόοδος κατά την εκπαίδευση του LSTM μοντέλου για το BNB

Συμπερασματικά, το μοντέλο BNB παρουσιάζει εξαιρετική απόδοση, το μοντέλο Ethereum αποδίδει επίσης καλά με μια καλή ισορροπία μεταξύ σφαλμάτων εκπαίδευσης και επικύρωσης. Το μοντέλο Bitcoin απαιτεί προσοχή για την αντιμετώπιση προβλημάτων υπερβολικής προσαρμογής για καλύτερη γενίκευση σε αόρατα δεδομένα.

3.7. Το μοντέλο GRU

Το GRU εισήχθη σε ένα έγγραφο του 2014 από τον Kyunghyun Cho και τους συνεργάτες του, ως εναλλακτική λύση στο δίκτυο Μακροπρόθεσμης Βραχυπρόθεσμης Μνήμης (LSTM). Κέρδισε γρήγορα δημοτικότητα λόγω της απλούστερης δομής του σε σύγκριση με τα LSTM, προσφέροντας ανταγωνιστική απόδοση, ενώ απαιτούσε λιγότερους υπολογιστικούς πόρους.

Αυτή η αρχιτεκτονική έχει σχεδιαστεί για εργασίες παλινδρόμησης που περιλαμβάνουν διαδοχικά δεδομένα. Το επίπεδο GRU καταγράφει αποτελεσματικά τις χρονικές εξαρτήσεις στα δεδομένα, ενώ το πλήρως συνδεδεμένο επίπεδο και το επίπεδο παλινδρόμησης συνεργάζονται για να παράγουν ακριβείς συνεχείς προβλέψεις.

Ολόκληρη η ρύθμιση είναι βελτιστοποιημένη για εκμάθηση από ακολουθίες και για προβλέψεις με βάση τα μοτίβα που βρίσκονται στα ιστορικά δεδομένα.

Η αρχιτεκτονική του GRU μοντέλου μας

Η αρχιτεκτονική του μοντέλου μας αποτελείται από:

Επίπεδο εισόδου ακολουθίας - δέχεται σε κάθε χρονικό βήμα 5 χαρακτηριστικά δεδομένων (Open, Low, High, Close,Volume).

Στρώμα GRU - επεξεργάζεται την ακολουθία εισόδου καταγράφοντας τις χρονικές εξαρτήσεις και τα μοτίβα των χαρακτηριστικών και αποτελείται από 50 κρυφές μονάδες. Είναι ρυθμισμένο να εξάγει μόνο την τελευταία κρυφή κατάσταση, αφού προηγήθηκε η επεξεργασία ολόκληρης της ακολουθίας εισόδου, κάτι πολύ συνηθισμένο στις εργασίες ακολουθίας προς ένα.

Πλήρως συνδεμένο Στρώμα - αποτελείται από έναν νευρώνα, ο οποίος παίρνει την τελική κρυφή κατάσταση που παράγει το στρώμα gru και την αντιστοιχίζει σε μια ενιαία συνεχή τιμή.

Το στρώμα παλινδρόμησης - συγκρίνει την προβλεπόμενη τιμή με την πραγματική τιμή στόχο, υπολογίζοντας την απώλεια. Η απώλεια χρησιμοποιείται για να ενημερωθούν οι παράμετροι του μοντέλου κατά την εκπαίδευση, ώστε το μοντέλο να πέτυχει την ελαχιστοποίηση του σφάλματος πρόβλεψης.

Για κάθε κρυπτονόμισμα επιλέξαμε διαφορετικές ρυθμίσεις εκπαίδευσης. Ανάλογα με τις ανάγκες κάθε κρυπτονομίσματος, εφαρμόσαμε διάφορες τιμές στις παραμέτρους εκπαίδευσης. Ο αλγόριθμος βελτιστοποίησης που επιλέχθηκε και για τα τρία σύνολα δεδομένων είναι ο Adam και το μέγεθος της ακολουθίας ισούται με πέντε. Ακολουθούν οι τιμές των ρυθμίσεων εκπαίδευσης για κάθε σύνολο δεδομένων, όπως αυτές έχουν διαμορφωθεί.

Bitcoin

Ρυθμός εκμάθησης - 0.0009.

Εποχές - 20.

Κανονικοποίηση L2 - 0.0001

Συντελεστής αποσύνθεσης - 0.1.

Μέγεθος μίνι παρτίδας - 64.

Περίοδος πτώσης ρυθμού εκμάθησης - 10.

Ποσοστό πτώσης ρυθμού εκμάθησης - 0.5.

Ethereum

Ρυθμός εκμάθησης - 0.003.

Εποχές - 50.

Κανονικοποίηση L2 - 0.001.

Συντελεστής αποσύνθεσης - 0.9.

Μέγεθος μίνι παρτίδας - 64.

Bnb

Ρυθμός εκμάθησης - 0.003.

Εποχές - 50.

Κανονικοποίηση L2 - 0.001.

Συντελεστής αποσύνθεσης - 0.9.

Μέγεθος μίνι παρτίδας - 64.

3.7.1 Εκπαίδευση και αξιολόγηση του μοντέλου.

Τα αποτελέσματα αξιολόγησης του μοντέλου διαμορφώθηκαν για κάθε κρυπτονόμισμα ως εξής:

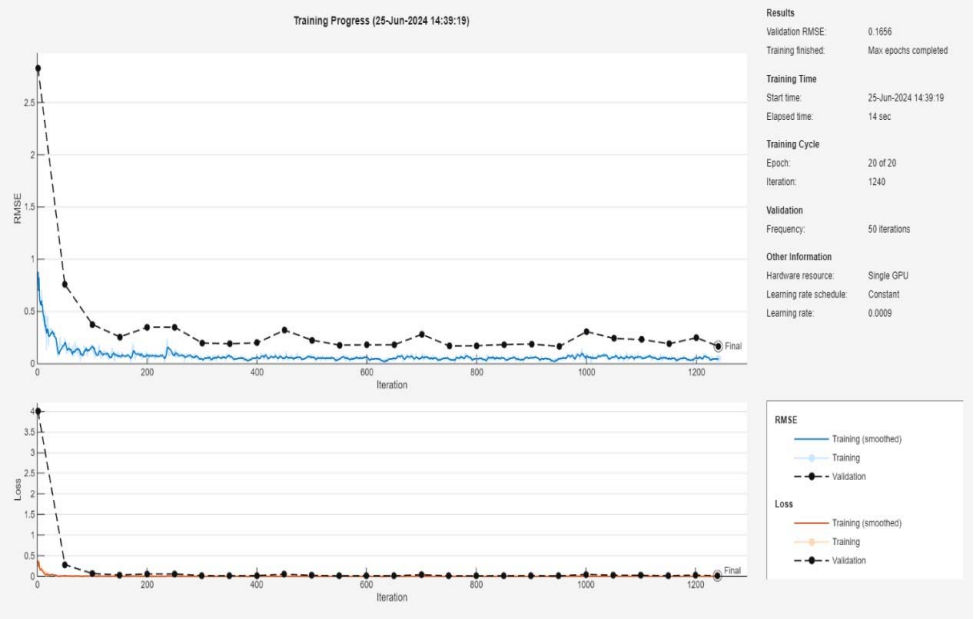
1. Bitcoin
   Ακολουθούν οι μετρήσεις απόδοσης του GRU για το σύνολο δεδομένων του Bitcoin

Πίνακας 3.7.1.α: Μετρήσεις αξιολόγησης GRU- Bitcoin

Final Values Bitcoin
Final Training Loss: 0.0011
Final Training RMSE: 0.0476
Final Validation Loss: 0.0137
Final Validation RMSE: 0.1656

Υπάρχει σημαντική διαφορά μεταξύ της απώλειας προπόνησης (0,0011) και της απώλειας επικύρωσης (0,0137). Η απώλεια επικύρωσης είναι μια τάξη μεγέθους μεγαλύτερη από την απώλεια εκπαίδευσης, η οποία μπορεί να υποδηλώνει ότι το μοντέλο ταιριάζει υπερβολικά στα δεδομένα εκπαίδευσης. Ομοίως, η τιμή RMSE (0,0476) στο σύνολο εκπαίδευσης είναι πολύ χαμηλότερη από την τιμή RMSE (0,1656) στο σύνολο επικύρωσης. Αυτή η απόκλιση υποστηρίζει περαιτέρω το ενδεχόμενο ζήτημα υπερπροσαρμογής.

Στο παρακάτω σχήμα καταγράφεται η πρόοδος εκπαίδευσης του μοντέλου για το Bitcoin.



Σχήμα 3.7.1.α: Πρόοδος κατά την εκπαίδευση του GRU μοντέλου για το Bitcoin

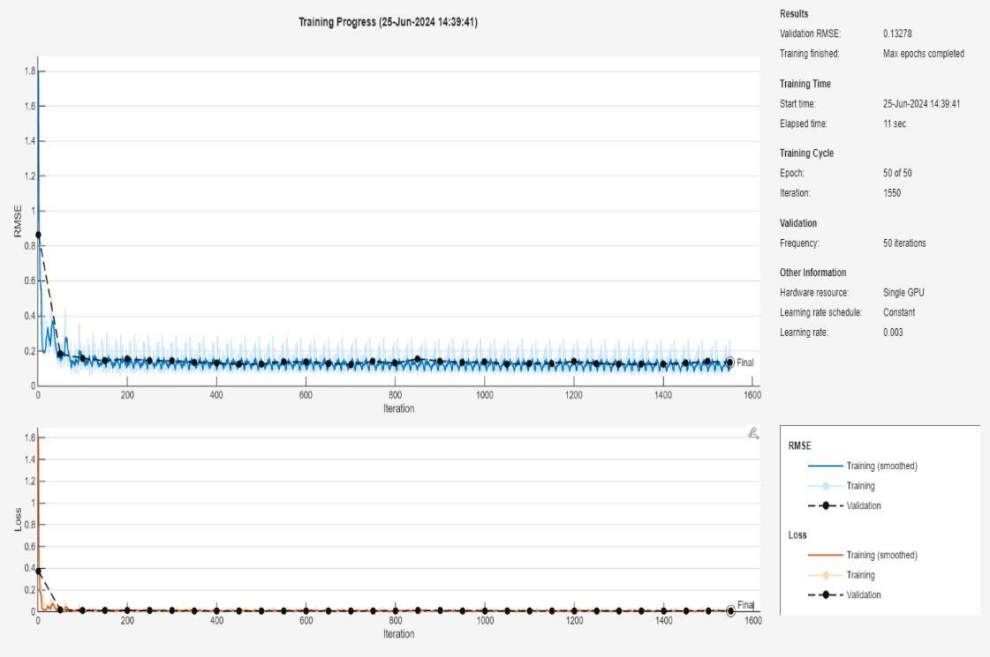
Στην περίπτωση του συγκεκριμένου κρυπτονομίσματος συνιστάται το μοντέλο Garch για να γίνει μια ακριβή εκτιμήση της αστάθειας των τιμών του συνόλου αυτού.

2. Ethereum
   Ακολουθούν οι μετρήσεις απόδοσης του GRU για το σύνολο δεδομένων του Ethereum

Πίνακας 3.7.1.β: Μετρήσεις αξιολόγησης GRU - Ethereum

Final Values Ethereum
Final Training Loss: 0.0070
Final Training RMSE: 0.1186
Final Validation Loss: 0.0088
Final Validation RMSE: 0.1328

Στο παρακάτω σχήμα απεικονίζεται η πρόοδος εκπαίδευσης του μοντέλου για το Ethereum.



Σχήμα 3.7.1.β: Πρόοδος κατά την εκπαίδευση του GRU μοντέλου για το Ethereum.

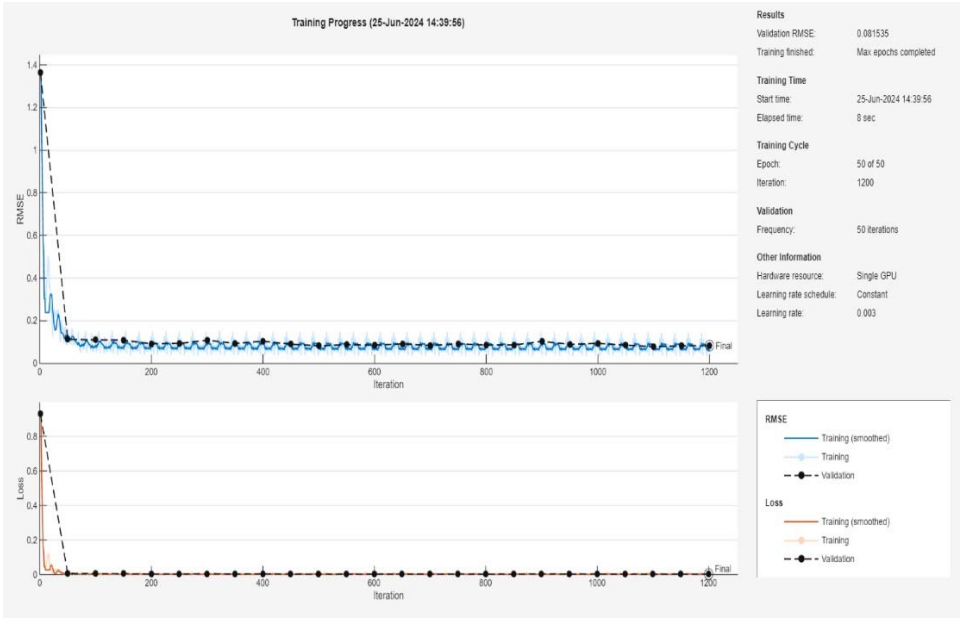
Η απώλεια εκπαίδευσης (0,0070) και η απώλεια επικύρωσης (0,0088) είναι σχετικά κοντά η μία στην άλλη. Αυτό υποδηλώνει ότι το μοντέλο γενικεύεται καλύτερα στο σύνολο επικύρωσης σε σύγκριση με το Bitcoin. Η εκπαίδευση RMSE (0,1186) και η επικύρωση RMSE (0,1328) είναι επίσης αρκετά κοντά, υποδεικνύοντας ότι το μοντέλο αποδίδει αρκετά καλά σε αόρατα δεδομένα.

3. BNB.
   Ακολουθούν οι μετρήσεις απόδοσης του GRU για το σύνολο δεδομένων του BNB

Πίνακας 3.7.1.γ: Μετρήσεις αξιολόγησης GRU - BNB

Final Values BNB
Final Training Loss: 0.0032
Final Training RMSE: 0.0803
Final Validation Loss: 0.0033
Final Validation RMSE: 0.0815

Στο παρακάτω σχήμα απεικονίζεται η πρόοδος εκπαίδευσης του μοντέλου για το BNB.



Σχήμα 3.7.1.γ: Πρόοδος κατά την εκπαίδευση του GRU μοντέλου για το BNB.

Η απώλεια προπόνησης (0,0032) και η απώλεια επικύρωσης (0,0033) είναι σχεδόν ταυτόσημες. Αυτό υποδηλώνει εξαιρετική γενίκευση από την εκπαίδευση στα δεδομένα επικύρωσης. Η εκπαίδευση RMSE (0,0803) και η επικύρωση RMSE (0,0815) είναι πολύ κοντά, υποδηλώνοντας ότι το μοντέλο έχει ισχυρή ικανότητα γενίκευσης.

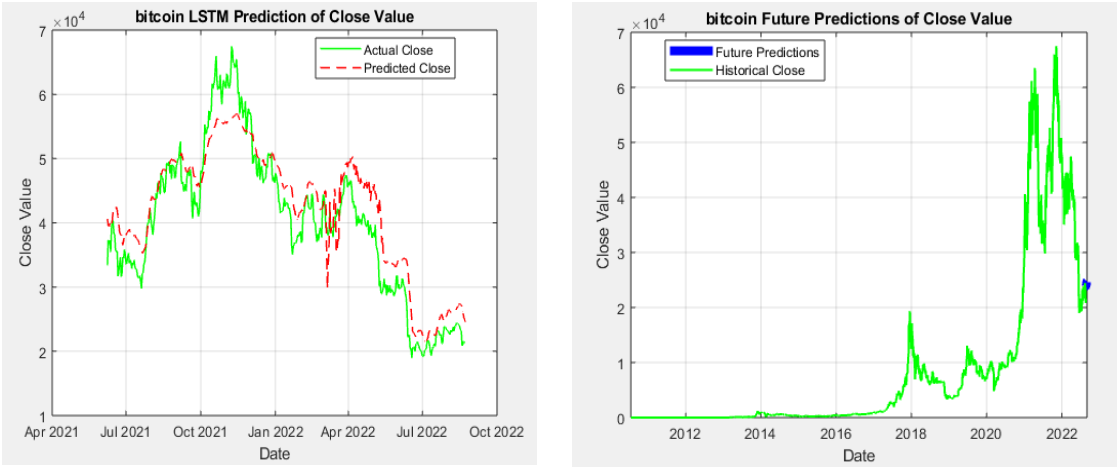
Συμπερασματικά, ενώ τα μοντέλα Ethereum και BNB παρουσιάζουν ικανοποιητική απόδοση, το μοντέλο Bitcoin μπορεί να χρειάζεται διαφορετική προσέγγιση για να βελτιώσει την προγνωστική του ακρίβεια σε αόρατα δεδομένα.

3.8. Μελλοντική πρόβλεψη των μοντέλων GRU - LSTM

Τα μοντέλα GRU - LSTM εκτός από την πρόβλεψη μιας χρονικής περιόδου χρησιμοποιούνται και στην πρόβλεψη μελλοντικών τιμών με βάση τιμές που έχουν παρατηρηθεί προηγούμενος. Η συγκεκριμένη διαδικασία μελλοντικής πρόβλεψης περιλαμβάνει την ανάλυση χρονικά ταξινομημένων σημείων δεδομένων για τον εντοπισμό προτύπων που μπορούν να χρησιμοποιηθούν για την πραγματοποίηση τεκμηριωμένων προβλέψεων.

Όπως προαναφερθήκαμε η μελλοντική πρόβλεψη προϋποθέτει την ανάλυση των τιμών μιας χρονικής περιόδου οι οποίες θα χρησιμοποιηθούν κατά την διαδικασία πρόβλεψης μιας μελλοντικής τιμής. Στην περίπτωση μας επιλέξαμε να προβλέψουμε την τιμή κλεισίματος(Close value) μιας χρονικής περιόδου και την επόμενη τιμή κλεισίματος μετά από 10 χρονικά βήματα και των τριών κρυπτονομισμάτων. Στα παρακάτω σχήματα ακολουθούν οι γραφικές παραστάσεις των προβλέψεων που πραγματοποίησαν τα δυο δίκτυα (LSTM, GRU) πάνω στα πραγματικά δεδομένα καθώς και την πρόβλεψη της τιμής Close μετά από 10 χρονικά βήματα.

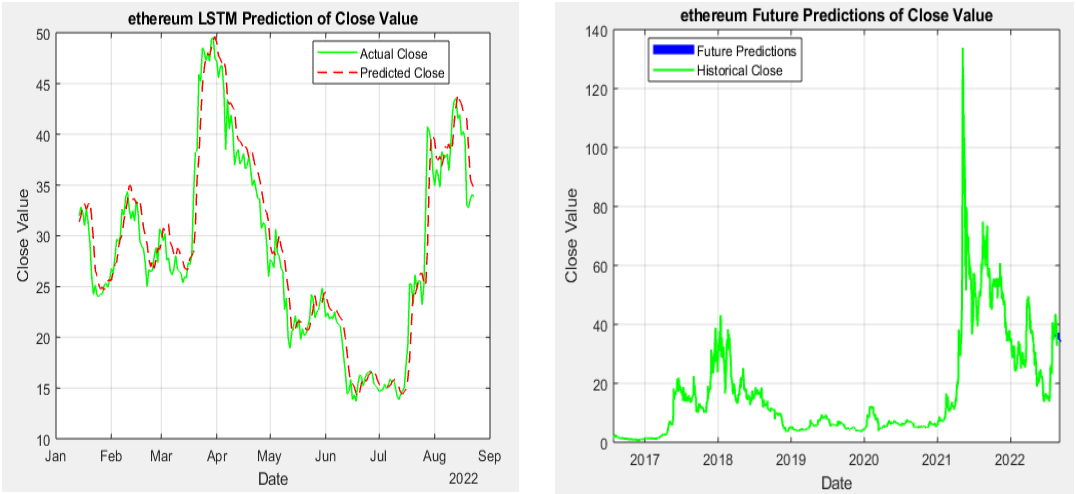
- **Προβλέψεις LSTM**



Σχήμα 3.8.α: Πρόβλεψη χρονικής περιόδου και επόμενης μελλοντικής τιμής κλεισίματος του LSTM - Bitcoin

Αριστερά με κόκκινη γραμμή φαίνονται οι προβλέψεις που πραγματοποίησε το μοντέλο κατά την περίοδο Απρίλιος 2021 - Οκτώβριος 2022 στα υπάρχοντα δεδομένα και δεξιά φαίνονται το ιστορικό της τιμής κλεισίματος του νομίσματος Bitcoin καθώς και η μελλοντική τιμή με μπλε γραμμή, που προέβλεψε το μοντέλο. Τα αποτελέσματα της μελλοντικής πρόβλεψης που πραγματοποίησε το μοντέλο είναι ικανοποιητικά καθώς το μοντέλο κατάφερε να προβλέψει την επόμενη τιμή Close, ενώ όσον αφορά τη συγκεκριμένη χρονική περίοδο υπάρχουν αρκετά σημεία όπου οι προβλέψεις του δικτύου δεν πλησιάζουν τις πραγματικές τιμές κλεισίματος του νομίσματος.

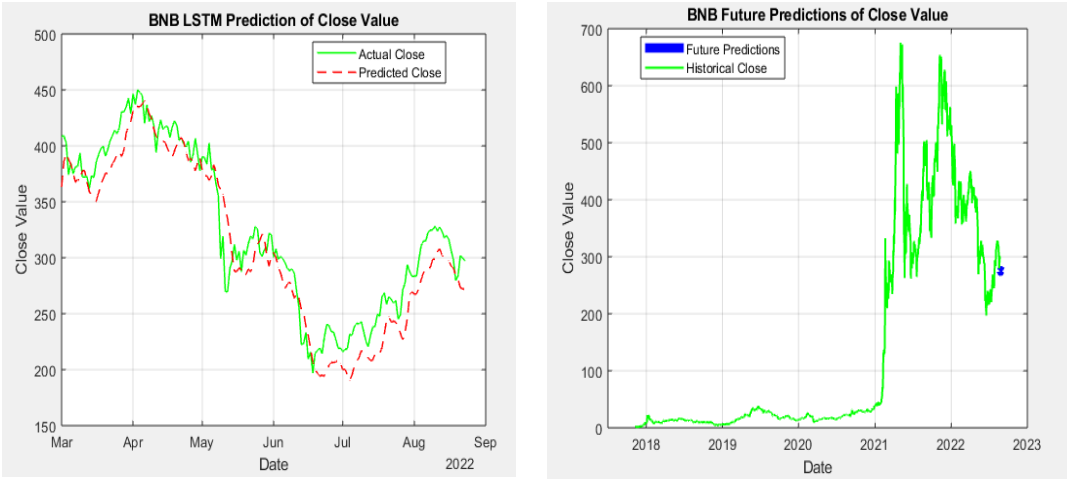
Στο παρακάτω σχήμα παρατηρείται ότι το μοντέλο κατάφερε να προβλέψει τη μελλοντική τιμή κλεισίματος του κρυπτονομίσματος Ethereum. Όσον αφορά τη χρονική περίοδο Ιανουάριος 2022 – Σεπτέμβριος 2022 η ικανότητα πρόβλεψης του μοντέλου είναι καλύτερη στο συγκεκριμένο κρυπτονόμισμα από ότι στο Bitcoin.



Σχήμα 3.8.β: Πρόβλεψη χρονικής περιόδου και επόμενης μελλοντικής τιμής κλεισίματος του LSTM - Ethereum

Στο παρακάτω σχήμα παρατηρείται ότι το μοντέλο κατάφερε να προβλέψει την μελλοντική τιμή, ενώ η ικανότητα πρόβλεψης όσον αφορά τη χρονική περίοδο Μάρτιος

2022 – Αύγουστος 2022 παρουσίασε μικρές διάφορες των προβλεπόμενων τιμών σε σχέση με τις πραγματικές τιμές.

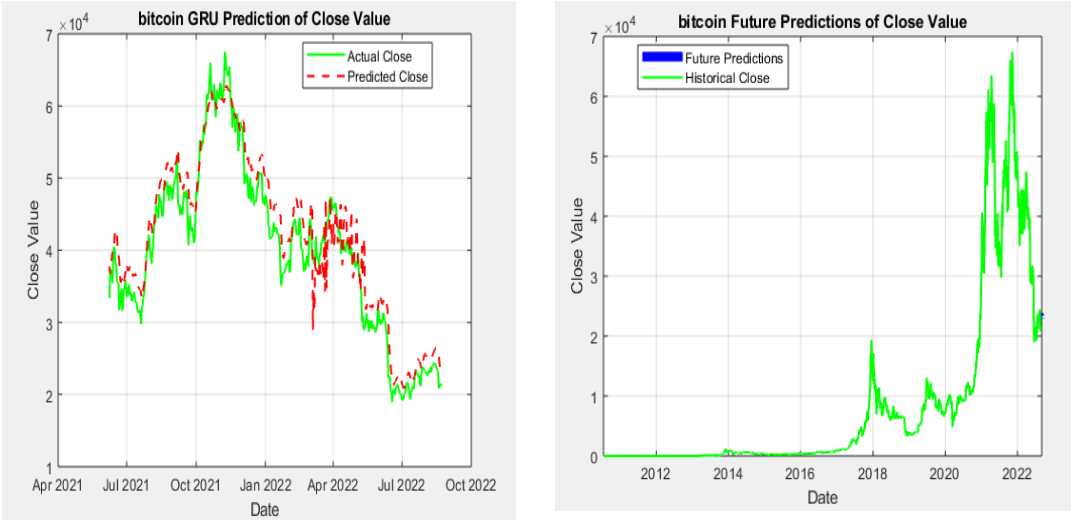


Σχήμα 3.8.γ: Πρόβλεψη χρονικής περιόδου και επόμενης μελλοντικής τιμής κλεισίματος του LSTM -BNB

- **Προβλέψεις GRU**

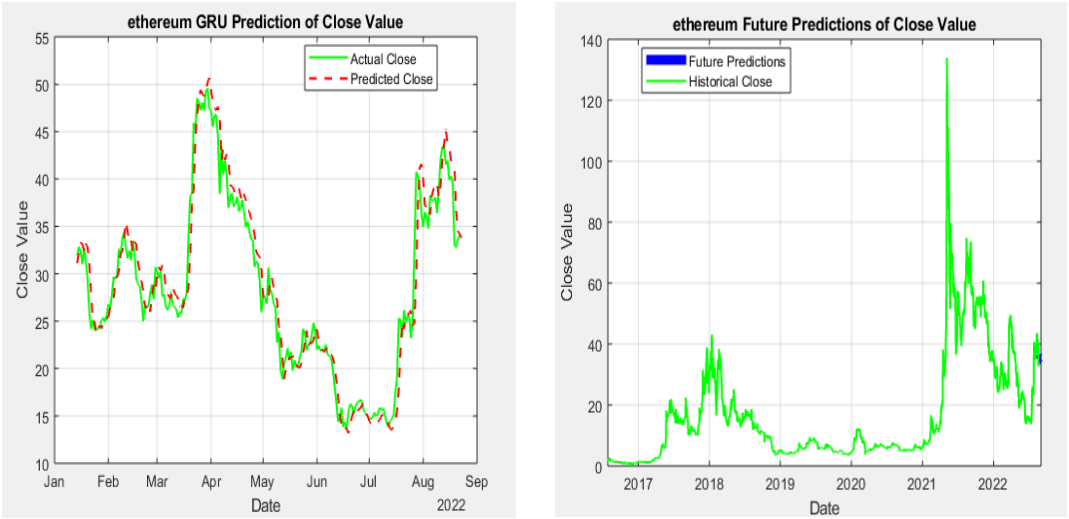
Στα παρακάτω σχήματα παρουσιάζονται οι γραφικές παραστάσεις των προβλέψεων του δικτύου GRU, τόσο σε μια χρονική περίοδο όσο και μελλοντικά, σε δέκα χρονικά βήματα.

Στο παρακάτω σχήμα παρατηρούμε ότι κατά τη χρονική περίοδο Απρίλιος-Οκτώβριος οι προβλέψεις του δικτύου είναι πολύ μέτριες και το μοντέλο δεν κατάφερε να προβλέψει τη μελλοντική τιμή κλεισίματος του κρυπτονομίσματος.



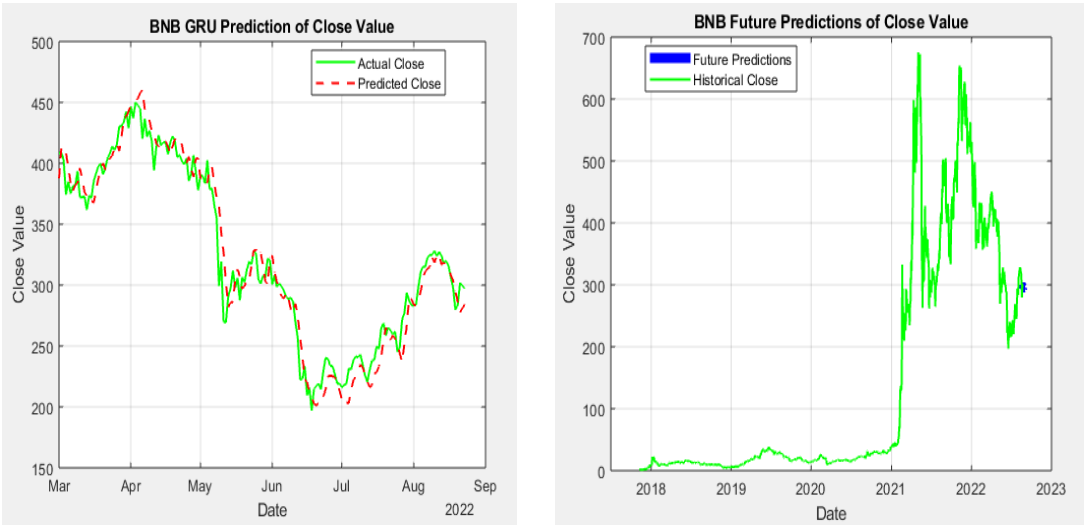
Σχήμα 3.8.δ: Πρόβλεψη χρονικής περιόδου και επόμενης μελλοντικής τιμής κλεισίματος του GRU - Bitcoin

Στο παρακάτω σχήμα παρατηρείται ότι το μοντέλο προβλέπει ικανοποιητικά κατά τη χρονική περίοδο και ταυτίζεται με την επόμενη μελλοντική τιμή κλεισίματος του κρυπτονομίσματος Ethereum.



Σχήμα 3.8.ε: Πρόβλεψη χρονικής περιόδου και μελλοντικής πρόβλεψης του GRU – Ethereum

Στην παρακάτω γραφική παράσταση φαίνεται ότι η ικανότητα πρόβλεψης του μοντέλου όσον αφορά το κρυπτονόμισμα BNB είναι ικανοποιητική τόσο στην πρόβλεψη τιμών της συγκεκριμένης χρονικής περιόδου, όσο και στην πρόβλεψης της επόμενης μελλοντικής τιμής κλεισίματος του κρυπτονομίσματος.



Σχήμα 3.8.στ: Πρόβλεψη χρονικής περιόδου και επόμενης μελλοντικής τιμής κλεισίματος του GRU – BNB

ΚΕΦΑΛΑΙΟ 4

1. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

Στο πρώτο τεχνικό μέρος της εργασίας ξεκινήσαμε από ένα απλό συνελικτικό δίκτυο για να κάνουμε ταξινόμηση εικόνων. Στη συνέχεια, αφού το εκπαιδεύσαμε, παρατηρήσαμε ότι το μοντέλο στις 40 εποχές υποπροσαρμοζόταν. Συνεπώς έπρεπε να εφαρμόσουμε μεθόδους ή τεχνικές για να βελτιώσουμε την απόδοσή του. Υπάρχουν πάρα πολλές τεχνικές για την αντιμετώπιση της υπερπροσαρμογής και την βελτίωση της απόδοσης του δικτύου. Επιλέχθηκαν οι παρακάτω μεθοδολογίες:

Ερευνήσαμε εκ νέου την αρχιτεκτονική του μοντέλου για να διαπιστώσουμε αν το μοντέλο είναι περίπλοκο ώστε στη συνέχεια, αν χρειαστεί, να το απλοποιήσουμε. Παρατηρήσαμε ότι η αρχιτεκτονική αυτού του μοντέλου περιλαμβάνει τυπικά στοιχεία ενός CNN και έχει σχεδιαστεί με σχετικά απλό τρόπο χωρίς προηγμένες τεχνικές, όπως υπολειπόμενες συνδέσεις, μονάδες έναρξης ή μηχανισμούς προσοχής, που θα την έκαναν περίπλοκη. Συνεπώς δεν χρειάστηκε να γίνει καμία αλλαγή και να απλοποιηθεί. Στη συνέχεια μειώσαμε τον κύκλο εκπαίδευσης αρχικά στις μισές εποχές (20 εποχές) παρατηρώντας ότι η υπερπροσαρμογή μειώθηκε αλλά παρέμεινε, οπότε μειώσαμε τις εποχές περαιτέρω στις 10. Στις 10 εποχές παρατηρήσαμε ότι το μοντέλο δεν υπερπροσαρμόζεται. Οι τιμές της απώλειας εκπαίδευσης και επικύρωσης έδειξαν ότι μπορεί να υπάρχει υποπροσαρμογή κάτι που δείχνει ότι το μοντέλο μπορεί να μην μαθαίνει τα δεδομένα εκπαίδευσης όσο καλά θα έπρεπε. Αυτό μπορεί να οφείλεται στην ανεπαρκή εκπαίδευση, η οποία δικαιολογεί και τη μέτρια απόδοση του μοντέλου σ' αυτήν την περίπτωση. Κατά συνέπεια χρειάστηκε να εφαρμόσουμε και άλλες τεχνικές βελτίωσης, όπως:

- **Εγκατάλειψη (Dropout).** Αυτή η μέθοδος τακτοποίησης αποκλείει τυχαία κάποιους νευρώνες μιας δεδομένης πιθανότητας, συμβάλλοντας στη μείωση της υπερπροσαρμογής του μοντέλου. Στην περίπτωση μας η συγκεκριμένη μέθοδος είχε καλά αποτελέσματα σε συνδυασμό με άλλες μεθόδους.
- **Τακτοποίηση L2 (L2 Regularization).** Η συγκεκριμένη μέθοδος βάζει μια ποινή στην συνάρτηση κόστους / απώλειας μειώνοντας τους συντελεστές του μοντέλου. Χρησιμοποιήθηκε ουσιαστικά για την κατάργηση χαρακτηριστικών υψηλής συσχέτισης, που οδηγούν στην υπερπροσαρμογή καθώς περιέχουν την ίδια πληροφορία και δεν προσθέτουν νέες πληροφορίες εκμάθησης κατά την εκπαίδευση. Όπως και η παραπάνω μέθοδος είχε καλά αποτελέσματα σε συνδυασμό με άλλες μεθόδους.
- **η επαύξηση δεδομένων (Data Augmentation).** Αυτή η μέθοδος επεκτείνει το μέγεθος ενός συνόλου δεδομένων δημιουργώντας τροποποιημένες εκδόσεις εικόνων, αυξάνοντας την ποικιλομορφία των δεδομένων χωρίς να συλλέγει νέα δεδομένα, βοηθώντας το μοντέλο να μάθει αμετάβλητα χαρακτηριστικά και μειώνοντας τον κίνδυνο απομνημόνευσης των δεδομένων εκπαίδευση. Τα αποτελέσματα ήταν πολύ θετικά, η υπερπροσαρμογή εξαλείφθηκε, οι τιμές απόδοσης σε όλα τα σύνολα δεδομένων επειδή βρίσκονται πολύ κοντά μεταξύ

τους, δείχνουν ότι το μοντέλο μαθαίνει ικανοποιητικά τα δεδομένα εκπαίδευσης και γενικεύει καλά τόσο στα δεδομένα επικύρωσης όσο και στα δεδομένα δοκιμής, δηλαδή σε αόρατα δεδομένα.

- **Η Υπερδειγματοληψία (oversampling).** Η εφαρμογή της μεθόδου αυτής δεν ήταν αρκετά αποτελεσματική ενώ έπρεπε να βελτιώσει την απόδοση του μοντέλου και να μειώσει την υπερπροσαρμογή στην πραγματικότητα το δίκτυο έδειχνε να υπερπροσαρμόζεται και ο λόγος είναι ότι αυτή η μέθοδος δεν προσθέτει νέες πληροφορίες στο σύνολο δεδομένων, καθώς οι νέες εικόνες δημιουργούνται με αντιγραφή των εικόνων που ήδη υπάρχουν, κάτι που μπορεί να αυξήσει το θόρυβο.
- **Η Διασταυρούμενη Επικύρωση K – fold (Crossvalidation K – fold).** Πρόκειται για μέθοδο τυχαίας επαναδειγματοληψίας των συνόλων δεδομένων. Επιλέξαμε το μοντέλο μας να εκτελεί επαναδειγματοληψία δέκα φορές (K=10). Κάθε επαναδειγματοληψία έδειξε ότι υπάρχει διαφορά στις μετρήσεις απόδοσης του μοντέλου εξαιτίας της υπερπροσαρμογής, με το μοντέλο να επιδεικνύει ισχυρή απόδοση στα δεδομένα εκπαίδευσης και μέτρια απόδοση στα δεδομένα επικύρωσης καθώς και καλύτερη γενίκευση σε αόρατα δεδομένα σε σχέση με τα δεδομένα επικύρωσης. Παρόλο που η Διασταυρούμενη Επικύρωση K – fold χρησιμοποιείται για την εκτίμηση της απόδοσης του μοντέλου και καταπολεμά την υπερπροσαρμογή δίνοντας τη δυνατότητα χρήσης περισσότερων συνδυασμών δεδομένων, στην περίπτωσή μας το μοντέλο υπερπροσαρμόζεται. Επίσης η συγκεκριμένη μέθοδος αυξάνει το χρόνο εκπαίδευσης και το υπολογιστικό κόστος.

Διαπιστώσαμε ότι από τις παραπάνω τεχνικές μόνο η εφαρμογή της Επαύξησης Δεδομένων (Data Augmentation) σε συνδυασμό με την Εγκατάλειψη, την τακτοποίηση L2 και την Z-score κανονικοποίηση αύξησε ικανοποιητικά την απόδοση του δικτύου και εξάλειψε την υπερπροσαρμογή. Αποφασίσαμε να αποθηκεύσουμε το μοντέλο στο οποίο εφαρμόσαμε τη συγκεκριμένη μέθοδο (Data Augmentation) σε συνδυασμό με τις παραπάνω τεχνικές και να το επαναχρησιμοποιήσουμε για μία εργασία ταξινόμησης εφαρμόζοντας τη μέθοδο Μεταφοράς Γνώσεων (Transfer Learning). Με αυτόν τον τρόπο αξιοποιήσαμε τη γνώση που απόκτησε το μοντέλο στην ταξινόμηση εικόνων μετά από την εκπαίδευσή του.

Αρχικά χρησιμοποιήσαμε το ίδιο σύνολο δεδομένων για να επανεκπαιδεύσουμε το μοντέλο και στη συνέχεια ένα διαφορετικό σύνολο δεδομένων για μια εργασία ταξινόμησης. Η εφαρμογή του Transfer Learning στο ίδιο σύνολο δεδομένων, δεν προσέφερε νέες πληροφορίες, δηλαδή το μοντέλο δεν είχε να μάθει κάτι καινούργιο όπως άκρες υφές μοτίβα. Η εκπαίδευση στην περίπτωση αυτή αποδείχθηκε περιττή, παρόλη την πολύ καλή απόδοση επικύρωσης του μοντέλου, στην πραγματικότητα δεν βελτίωσε τη γενίκευση πάνω στα δεδομένα εκπαίδευσης. Στη δεύτερη περίπτωση, όπου εφαρμόσαμε το Transfer Learning σε ένα διαφορετικό σύνολο δεδομένων, που ήταν και αυτό σχετικά μικρό, η μεθοδολογία που ακολουθήσαμε είχε μεγάλη επιτυχία καθώς το Transfer Learning βελτίωσε τη γενίκευση του προεκπαιδευμένου μοντέλου, χωρίς να παρουσιάσει υπερπροσαρμογή. Επίσης με τη χρήση του Transfer Learning μειώθηκε το απαιτούμενο υπολογιστικό κόστος για τη δημιουργία μοντέλων που θα

χρησιμοποιηθούν σε νέες εργασίες. Με την επαναχρησιμοποίηση του προεκπαιδευμένου μοντέλου μειώθηκε ο χρόνος της εκπαίδευσης δηλαδή σε αυτή περίπτωση μειώσαμε τις εποχές από 80 στις 20, όπως επίσης μειώθηκαν και οι αρχικοί υπολογιστικοί πόροι.

Το συγκεκριμένο CNN μοντέλο έχει δείξει πολύ καλή προσαρμογή στην εργασία της ταξινόμησης εικόνων και θα μπορούσε να αναπτυχθεί, να εξελιχθεί και να χρησιμοποιηθεί σε εφαρμογές πραγματικού κόσμου, αφού προηγηθεί η απαραίτητη μελέτη και προσαρμογή του δικτύου στις νέες εργασίες, όπως για παράδειγμα σε:

- Εφαρμογές οι οποίες βασίζονται στην αυτόνομη οδήγηση, όπου το μοντέλο καλείται να ανιχνεύσει αντικείμενα, να κατανοήσει σκηνές ζωτικής σημασίας, να επεξεργάζεται τα δεδομένα που δέχεται σε πραγματικό χρόνο με υψηλή ακρίβεια για τη διασφάλιση της ασφάλειας των πεζών καθώς και των οδηγών και επιβατών των αυτοκινήτων αυτόνομης οδήγησης.
- Εφαρμογές αναγνώρισης προσώπου σε συστήματά ασφάλειας, που απαιτούν από το μοντέλο να χειρίζεται διάφορες συνθήκες φωτισμού, εκφράσεις προσώπου και αποφράξεις.
- Σε εφαρμογές αγροτικής παρακολούθησής των φυτών, οι οποίες απαιτούν προσαρμοστικότητα σε διαφορετικούς τύπους καλλιεργειών και περιβαλλοντικές συνθήκες, όπου το μοντέλο θα πρέπει να προσδιορίζει τις ασθένειες των φυτών και την παρακολούθηση της υγείας των καλλιεργειών μέσω ανάλυσης εικόνας.

Για να ενταχθεί το μοντέλο μας στις παραπάνω εφαρμογές πραγματικού κόσμου οι προκλήσεις είναι μεγάλες. Θα πρέπει να γίνει διασφάλιση ότι τα δεδομένα προστατεύονται ειδικά για εφαρμογές όπως είναι η αναγνώριση προσώπων σε συστήματά ασφάλειας. Το μοντέλο θα πρέπει να ενημερωθεί μέσω των μηχανισμών ενημέρωσης ώστε να εντοπίζονται οι αλλαγές κατά τη διανομή των δεδομένων καθώς και να εφαρμοστεί η διαδικτυακή μάθηση και η επαυξητική εκπαίδευση, ώστε να υπάρχει περιοδική ενημέρωση του μοντέλου σε νέες παρτίδες δεδομένων. Επίσης χρειάζεται το μοντέλο να προσαρμοστεί σε διαφορετικές συνθήκες και σενάρια, να μπορεί να επεξεργάζεται με ακρίβεια τα δεδομένα σε πραγματικό χρόνο, καθώς και να σχεδιαστεί ώστε να λειτουργήσει αποτελεσματικά σε περιβάλλοντα είτε για διακομιστές Cloud είτε για συσκευές αιχμής.

Μέσω της εφαρμογής της μεταφοράς γνώσης το μοντέλο θα μπορούσε να χρησιμοποιηθεί π.χ. στη γενική ανίχνευση αντικειμένων για τον εντοπισμό συγκεκριμένων τύπων μηχανημάτων σε ένα βιομηχανικό περιβάλλον.

Στο δεύτερο τεχνικό μέρος της εργασίας αναπτύξαμε δύο αρχιτεκτονικές RNN, τις οποίες χρησιμοποιήσαμε στην πρόβλεψη κρυπτονομισμάτων. Ο πρώτος τύπος επαναλαμβανόμενης αρχιτεκτονικής που υλοποιήσαμε ήταν μία LSTM, που σχεδιάστηκε με απλό τρόπο. Χρειάστηκε να εφαρμόσουμε στα δεδομένα που χρησιμοποιήθηκαν τεχνικές όπως η ταξινόμησή τους με βάση την ημερομηνία, να αφαιρέσουμε από τους πίνακες τις γραμμές στις οποίες οι στήλες περιείχαν τιμή NaN, να κάνουμε κανονικοποίηση χρησιμοποιώντας τη μέθοδο z-score. Τα αποτελέσματα της πρόβλεψης που έκανε το μοντέλο ήταν σχετικά ικανοποιητικά όσον αφορά τα

κρυπτονομίσματα Ethereum και BNB, ενώ το μοντέλο έδειχνε μεγάλη υπερπροσαρμογή στα δεδομένα του Bitcoin, υποδηλώνοντας ότι υπάρχει αστάθεια τιμών στο συγκεκριμένο σύνολο. Για την ανάλυση του συνόλου δεδομένων Bitcoin επιβάλλεται μια διαφορετική προσέγγιση, όπως είναι το μοντέλο Garch, που προσφέρει μια ακριβή εκτιμήση της αστάθειας των δεδομένων Bitcoin.

Ο δεύτερος τύπος RNN που υλοποιήσαμε ήταν μία αρχιτεκτονική GRU. Εφαρμόσαμε τις ίδιες τεχνικές που προαναφέραμε πάνω στα δεδομένα μας με σκοπό την καλύτερη συσχέτιση των δεδομένων και το βέλτιστο αποτέλεσμα.

Τα αποτελέσματα της πρόβλεψης ήταν πολύ ικανοποιητικά στα δύο σύνολα κρυπτονομισμάτων, Ethereum και BNB, ενώ στο σύνολο Bitcoin το μοντέλο έδειξε υπερπροσαρμογή. Συνιστάται να χρησιμοποιηθεί ένα διαφορετικό μοντέλο που να κάνει μία ακριβή εκτίμηση της αστάθειας των δεδομένων Bitcoin.

Το GRU και στις τρεις περιπτώσεις Bitcoin, Ethereum, BNB, παρουσιάζει σημαντικά καλύτερη απόδοση από το LSTM, με χαμηλότερες απώλειες και RMSE, ξεπερνά το LSTM τόσο στις φάσεις εκπαίδευσης όσο και στην επικύρωση, υποδηλώνοντας καλύτερη ακρίβεια και γενίκευση και επιδεικνύει καλύτερες μετρήσεις απόδοσης από το LSTM. Αυτό δείχνει ότι τα μοντέλα GRU είναι πιο αποτελεσματικά και αξιόπιστα για την πρόβλεψη τιμών κρυπτονομισμάτων σε αυτή την ανάλυση.

Το μοντέλο GRU έδειξε ότι ήταν πιο αποτελεσματικό και πιο αξιόπιστο από ότι το LSTM και θα μπορούσε να εξελιχθεί και να τελειοποιηθεί σε ένα μοντέλο υβριδικό, ενσωματώνοντας το μοντέλο GARCH και αξιοποιώντας τις δυνατότητες και των δυο μοντέλων για καλύτερη πρόβλεψη χρονοσειρών. Οι ικανότητες ενός υβριδικού μοντέλου GRU – GARCH είναι οι ακόλουθες:

- Αποτύπωση χρονικών εξαρτήσεων.
  Το στοιχείο GRU του υβριδικού μοντέλου θα αποτυπώνει αποτελεσματικά τις διαδοχικές εξαρτίσεις στα δεδομένα, παρέχοντας πιο ακριβείς και ισχυρές προβλέψεις, που βασίζονται στις προηγούμενες παρατηρήσεις.
- Εκτίμηση της αστάθειας των δεδομένων.
  Το στοιχείο GARCH του υβριδικού μοντέλου θα καταγράφει την αστάθεια των δεδομένων, επιτρέποντας να ληφθούν υπόψη περίοδοι, όπου τα δεδομένα μεταβάλλονται.
- Βελτιωμένη ακρίβεια πρόβλεψης.
  Το καινούργιο μοντέλο παρέχει πιο ακριβείς και ισχυρές προβλέψεις, συνδυάζοντας τα δυνατά σημεία και των δυο μοντέλων, σε σύγκριση με την πρόβλεψη που θα πραγματοποιούσε το κάθε μοντέλο ξεχωριστά.
- Ενισχυμένη εκτίμηση κινδύνων.
  Το στοιχείο GARCH θα επιτρέπει την πρόβλεψη της διακύμανσης υπό ορούς και την καλύτερη αξιολόγηση και διαχείριση κινδύνου, η όποια είναι ιδιαίτερα πολύτιμη σε χρηματοοικονομικές εφαρμογές.

Συνολικά η μετατροπή του GRU μοντέλου σε ένα υβριδικό μοντέλο GRU – GARCH θα αποτελέσει ένα ισχυρό εργαλείο για την πρόβλεψη χρονοσειρών, ιδιαίτερα σε τομείς όπου η μεταβλητότητα και οι χρονικές εξαρτήσεις είναι εξέχουσες.

ΚΕΦΑΛΑΙΟ 5

2. ΠΑΡΑΡΤΗΜΑ Α΄

Στο συγκεκριμένο παράρτημα παρατίθεται ο κώδικας υλοποίησης του συνελικτικού νευρωνικού δικτύου και οι τροποποιήσεις που χρειάστηκε να πραγματοποιηθούν για την εφαρμογή των μεθόδων βελτίωσης του δικτύου ως προς την απόδοση του. Παρουσιάζουμε τον κώδικα του αρχικού μας μοντέλου και συνεχίζουμε με τις αλλαγές του κώδικα ως εξής:

- **Ο κώδικας του αρχικού συνελικτικού νευρωνικού δικτύου**

```
% URL and folder information
clc;
clearvars;
url = 'http://download.tensorflow.org/example_images/flower_photos.tgz';
downloadFolder = tempdir;
filename = fullfile(downloadFolder,'flower_dataset.tgz');
dataFolder = fullfile(downloadFolder,'flower_photos');
% Download the dataset if it doesn't exist
if ~exist(dataFolder,'dir')
   fprintf("Downloading Flowers data set (218 MB)... ")
   websave(filename,url);
   untar(filename,downloadFolder)
   fprintf("Done.\n")
end
% Load images from the ImageDatastore and resize them
images = imageDatastore(dataFolder, ...
   'IncludeSubfolders', true, ...
   'LabelSource', 'foldernames', ...
   'ReadFcn',@(x) imresize(imread(x), [256,256]));
% Display some of the images
numImages=numel(images.Labels);
figure;
perm = randperm(numImages, 9);
for i = 1:9
   subplot(3,3,i);
   imshow(readimage(images, perm(i)));
end
% Get the size of the sample image
sampleImage = readimage(images, 1);
input_size = size(sampleImage);
%split data
splitRatioTrain = 0.8;
splitRatioValidation = 0.1;
splitRatioTest = 0.1;
```

```
[trainingImages,validationImages,testImages]=splitEachLabel(images,
splitRatioTrain, splitRatioValidation,splitRatioTest,'randomized');
TestImages_Labels = countEachLabel(testImages);
display(TestImages_Labels);
ValidationImages_Labels = countEachLabel(validationImages);
display(ValidationImages_Labels);
TrainingImages_Labels = countEachLabel(trainingImages);
display(TrainingImages_Labels);
layers = [
   imageInputLayer([256 256 3],"Normalization","zscore")
   convolution2dLayer(3, 32, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 64, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2 )
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last1')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   averagePooling2dLayer(2, 'Stride', 2)
   flattenLayer('Name', 'flatten') % Flattening layer
   fullyConnectedLayer(1024)
   reluLayer
   dropoutLayer(0.2) % Increase dropout rate for better regularization
   fullyConnectedLayer(512)
   dropoutLayer(0.2)
   fullyConnectedLayer(5)
   softmaxLayer
   classificationLayer];
learnRate = 0.001;
options = trainingOptions('adam', ...
   'ValidationData',validationImages,...
   'MaxEpochs',40, ...
```

```
    'InitialLearnRate',learnRate,...
    'Shuffle','every-epoch',...
    'LearnRateDropPeriod',21,...
    'LearnRateDropFactor',0.1,...
    'OutputNetwork','last-iteration',...
    'ValidationFrequency',50, ...
    'Verbose',false,...
   'Plots','training-progress');
[net,info] = trainNetwork(trainingImages,layers,options);
% Evaluate the model on the training set
trainingLabels = trainingImages.Labels;
predictedLabels=classify(net, trainingImages);
% Evaluate the model on the test set
TPred = classify(net, testImages);
TLabels = testImages.Labels;
VPred=classify(net, validationImages);
VLabels=validationImages.Labels;
accuracy=mean(TPred==TLabels);
%model metrics
fprintf('Training accuracy: %.2f%%\n',info.TrainingAccuracy(end) );
fprintf('Training loss: %.2f%%\n', info.TrainingLoss(end));
fprintf('Validation accuracy: %.2f%%\n',info.ValidationAccuracy(end))
fprintf('Validation loss: %.2f%%\n',info.ValidationLoss(end))
fprintf('Test set accuracy: %.2f%%\n', accuracy*100);
% Display confusion chart for the Training Data
figure('Units', 'normalized', 'Position', [0.2 0.2 0.4 0.4]);
chart = confusionchart(trainingLabels, predictedLabels);
chart.Title = 'Confusion Matrix for the Training Data';
chart.ColumnSummary = 'column-normalized';
chart.RowSummary = 'row-normalized';
% Display confusion chart for the Validation Data
figure('Units', 'normalized', 'Position', [0.2 0.2 0.4 0.4]);
chart = confusionchart(VLabels, VPred);
chart.Title = 'Confusion Matrix for the Validation Data';
chart.ColumnSummary = 'column-normalized';
chart.RowSummary = 'row-normalized';
% Display confusion chart for the Test Data
figure('Units', 'normalized', 'Position', [0.2 0.2 0.4 0.4]);
chart = confusionchart(TPred, TLabels);
chart.Title = 'Confusion Matrix for the Test Data';
chart.ColumnSummary = 'column-normalized';
chart.RowSummary = 'row-normalized';
```

- **Μείωση εποχών από 40 στις 20**
  Η μοναδική τροποποίηση πραγματοποιήθηκε στις ρυθμίσεις εκπαίδευσης και είναι η ακόλουθη γραμμή.

  'MaxEpochs',20, ...

- **Μείωση εποχών από 20 στις 10**

Παρομοίως και εδώ τροποποιήθηκε η παραπάνω γραμμή με την ακόλουθη γραμμή.

'MaxEpochs',10, ...

- **Επαύξηση Δεδομένων (Data Augmentation)**
  Για να εφαρμόσουμε τη μέθοδο της επαύξησης δεδομένων κάναμε αρκετές τροποποιήσεις και για αυτό τον λόγο παραθέτουμε όλον τον κώδικα.

```
% URL and folder information
clearvars;
url = 'http://download.tensorflow.org/example_images/flower_photos.tgz';
downloadFolder = tempdir;
filename = fullfile(downloadFolder,'flower_dataset.tgz');
dataFolder = fullfile(downloadFolder,'flower_photos');
% Download the dataset if it doesn't exist
if ~exist(dataFolder,'dir')
   fprintf("Downloading Flowers data set (218 MB)... ")
   websave(filename,url);
   untar(filename,downloadFolder)
   fprintf("Done.\n")
end
% Load images from the ImageDatastore and resize them
images = imageDatastore(dataFolder,...
   'IncludeSubfolders', true,...
   'LabelSource', 'foldernames');
%Display some of the images in the datastore.
figure;
perm = randperm(3670,9);
for i = 1:9
   subplot(3,3,i);
   imshow(images.Files{perm(i)});
end
% Create a path in the current working directory
checkpoint = fullfile(pwd, 'checkpoints1');
labelCount = countEachLabel(images)
%Split Data
splitRatioTrain = 0.8;
splitRatioValidation = 0.1;
splitRatioTest = 0.1;
[trainingImages,validationImages,testImages]=splitEachLabel(images,
splitRatioTrain, splitRatioValidation, splitRatioTest, 'randomized');
% Use imageDataAugmenter for data augmentation
augmenter = imageDataAugmenter(...
   'RandXReflection', true, ...
   'RandXTranslation', [-2 2], ...
   'RandYTranslation', [-2 2], ...
   'RandRotation', [-30 30],...
   'RandScale', [0.9 1.1]);
%Data Augumentation
```

```
augmentedTrainingImages=augmentedImageDatastore([256,256],trainingImages,
'DataAugmentation', augmenter);
augmentedValidationImages=augmentedImageDatastore([256,256],
validationImages, 'DataAugmentation', augmenter);
augmentedTestImages=augmentedImageDatastore([256,256],testImages,
'DataAugmentation', augmenter);
% Define the CNN architecture
layers = [
   imageInputLayer([256 256 3],"Normalization","zscore")
   convolution2dLayer(3, 32, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 64, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2 )
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last1')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   averagePooling2dLayer(2, 'Stride', 2)
   flattenLayer('Name', 'flatten')
   fullyConnectedLayer(16)
   dropoutLayer(0.15)
   fullyConnectedLayer(5)
   softmaxLayer
   classificationLayer];
lgraph = layerGraph(layers);
learnRate = 0.001;
% Define training options
options = trainingOptions('adam', ...
   'MaxEpochs',80, ...
   'InitialLearnRate', learnRate, ...
   'LearnRateDropPeriod',76,...
   'LearnRateDropFactor',0.1,...
```

```
    'L2Regularization',0.01,...
    'ValidationData', augmentedValidationImages, ...
    'ValidationFrequency', 50, ...
    'ValidationPatience',15,...
    'OutputNetwork','last-iteration',...
    'LearnRateSchedule', 'piecewise', ...
    'Shuffle', 'every-epoch', ...
    'Verbose', 0, ...
    'CheckpointPath', checkpoint, ...
    'Plots', 'training-progress');
% Train the CNN using the trainNetwork function with Datastore input
rng('default');
rng(123);
[net,info] = trainNetwork(augmentedTrainingImages,lgraph,options);
% Extract labels for the test,validation,training set
testLabels = testImages.Labels;
predictedLabels = classify(net, augmentedTestImages);
trainLabels = trainingImages.Labels;
trainPred = classify(net, augmentedTrainingImages);
valLabels = validationImages.Labels;
valPred = classify(net, augmentedValidationImages);
% element-wise comparison
fprintf('Training accuracy: %.2f%%\n',info.TrainingAccuracy(end) );
fprintf('Training loss: %.2f%%\n', info.TrainingLoss(end));
fprintf('Validation accuracy: %.2f%%\n',info.ValidationAccuracy(end))
fprintf('Validation loss: %.2f%%\n',info.ValidationLoss(end))
accuracy=mean(predictedLabels==testLabels);
fprintf('Test set accuracy: %.2f%%\n', accuracy*100);
analyzeNetwork(net);
% Confusion Matrix
figure(units="normalized",Position=[0.2 0.2 0.4 0.4]);
cm = confusionchart(testLabels, predictedLabels);
cm.Title = "Confusion Matrix for Test Data";
cm.ColumnSummary = "column-normalized";
cm.RowSummary = "row-normalized";
% Confusion Matrix
figure(units="normalized",Position=[0.2 0.2 0.4 0.4]);
cm = confusionchart(trainLabels,trainPred );
cm.Title = "Confusion Matrix on the Train set";
cm.ColumnSummary = "column-normalized";
cm.RowSummary = "row-normalized";
% Confusion Matrix
figure(units="normalized",Position=[0.2 0.2 0.4 0.4]);
cm = confusionchart(valLabels, valPred);
cm.Title = "Confusion Matrix on the Validation set";
cm.ColumnSummary = "column-normalized";
cm.RowSummary = "row-normalized";
```

- **Υπερδειγματοληψία (Oversampling)**
  Στον παραπάνω κώδικα υλοποιείται η μέθοδος της υπερδειγματοληψίας.

```
% URL and folder information
clearvars;
url = 'http://download.tensorflow.org/example_images/flower_photos.tgz';
downloadFolder = tempdir;
filename = fullfile(downloadFolder,'flower_dataset.tgz');
dataFolder = fullfile(downloadFolder,'flower_photos');
% Download the dataset if it doesn't exist
if ~exist(dataFolder,'dir')
   fprintf("Downloading Flowers data set (218 MB)... ")
   websave(filename,url);
   untar(filename,downloadFolder)
   fprintf("Done.\n")
end
% Load images from the ImageDatastore and resize them
images = imageDatastore(dataFolder, ...
   'IncludeSubfolders', true, ...
   'LabelSource', 'foldernames');
%Display some of the images in the datastore.
figure;
perm = randperm(3670,9);
for i = 1:9
   subplot(3,3,i);
   imshow(images.Files{perm(i)});
end
%Split Data
splitRatioTrain = 0.8;
splitRatioValidation = 0.1;
splitRatioTest = 0.1;
[trainingImages,validationImages,testImages]=splitEachLabel(images,
splitRatioTrain, splitRatioValidation, splitRatioTest, 'randomized');
%Count each labels
labelCountTr = countEachLabel(trainingImages);
labelCountTest=countEachLabel(testImages);
labelCountVal=countEachLabel(validationImages);
% Find the maximum number of images in any class
maxNumObservationsTr = max(labelCountTr.Count);
maxNumObservationsVal = max(labelCountVal.Count);
maxNumObservationsTest = max(labelCountTest.Count);
% Apply the function to balance the datastore
imagesBalancedTr=balanceImageDatastore(trainingImages,
maxNumObservationsTr);
imagesBalancedVal=balanceImageDatastore(validationImages,
maxNumObservationsVal);
imagesBalancedTest=balanceImageDatastore(testImages,
maxNumObservationsTest);
% Display the number of images per class in the balanced datastore
labelCountBalancedTr = countEachLabel(imagesBalancedTr);
disp(labelCountBalancedTr);
labelCountBalancedVal = countEachLabel(imagesBalancedVal);
disp(labelCountBalancedVal);
```

```
labelCountBalancedTest = countEachLabel(imagesBalancedTest);
disp(labelCountBalancedTest);
%checkpoint path
checkpoint = fullfile(pwd, 'checkpoints1');
% Use imageDataAugmenter for data augmentation
augmenter = imageDataAugmenter(...
   'RandXReflection', true, ...
   'RandXTranslation', [-2 2], ...
   'RandYTranslation', [-2 2], ...
   'RandScale', [0.9 1.1], ...
   'RandRotation', [-30 30]);
%Data Augumentation
augmentedTrainingImages=augmentedImageDatastore([256,256], imagesBalancedTr,
'DataAugmentation', augmenter);
augmentedValidationImages=augmentedImageDatastore([256,256],
imagesBalancedVal, 'DataAugmentation', augmenter);
augmentedTestImages = augmentedImageDatastore([256,256], imagesBalancedTest,
'DataAugmentation', augmenter);
%Define the CNN architecture
layers = [
   imageInputLayer([256 256 3],"Normalization","zscore")
   convolution2dLayer(3, 32, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 64, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2 )
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last1')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   averagePooling2dLayer(2, 'Stride', 2)
   flattenLayer('Name', 'flatten')
   fullyConnectedLayer(8)
```

```
    dropoutLayer(0.45)
    fullyConnectedLayer(5)
    softmaxLayer
    classificationLayer];
% Create a LayerGraph
lgraph = layerGraph(layers);
% Define training options
learnRate = 0.001;
% Define training options
 options = trainingOptions('adam', ...
    'MaxEpochs',80, ...
    'InitialLearnRate', learnRate, ...
    'LearnRateDropPeriod',76,...
    'LearnRateDropFactor',0.1,...
    'L2Regularization',0.01,...
    'ValidationData', augmentedValidationImages, ...
    'ValidationFrequency', 50, ...
    'ValidationPatience',15,...
    'OutputNetwork','last-iteration',...
    'LearnRateSchedule','piecewise', ...
    'Shuffle', 'every-epoch', ...
    'Verbose', 0, ...
    'CheckpointPath', checkpoint, ...
    'Plots', 'training-progress');
rng(123);
% train the network
[net,info] = trainNetwork(augmentedTrainingImages,lgraph,options);
% Extract labels for the model metrics
testLabels = imagesBalancedTest.Labels;
predictedLabels = classify(net, augmentedTestImages);
trainLabels = imagesBalancedTr.Labels;
trainPred = classify(net, augmentedTrainingImages);
valLabels = imagesBalancedVal.Labels
valPred = classify(net, augmentedValidationImages);
% model final metrics
fprintf('Training accuracy: %.2f%%\n',info.TrainingAccuracy(end) );
fprintf('Training loss: %.2f%%\n', info.TrainingLoss(end));
fprintf('Validation accuracy: %.2f%%\n',info.ValidationAccuracy(end))
fprintf('Validation loss: %.2f%%\n',info.ValidationLoss(end))
accuracy=mean(predictedLabels==testLabels);
fprintf('Test set accuracy: %.2f%%\n', accuracy*100);
analyzeNetwork(net);
%Confusion Matrix
figure(units="normalized",Position=[0.2 0.2 0.4 0.4]);
cm = confusionchart(testLabels, predictedLabels);
cm.Title = "Confusion Matrix for Test Data";
cm.ColumnSummary = "column-normalized";
cm.RowSummary = "row-normalized";
%Confusion Matrix
figure(units="normalized",Position=[0.2 0.2 0.4 0.4]);
cm = confusionchart(trainLabels,trainPred );
```

```
cm.Title = "Confusion Matrix on the Train set";
cm.ColumnSummary = "column-normalized";
cm.RowSummary = "row-normalized";
%Confusion Matrix
figure(units="normalized",Position=[0.2 0.2 0.4 0.4]);
cm = confusionchart(valLabels, valPred);
cm.Title = "Confusion Matrix on the Validation set";
cm.ColumnSummary = "column-normalized";
cm.RowSummary = "row-normalized";
```

- **Διασταυρούμενη Επικύρωση K-fold (K-fold Cross-Validation)**
  Ακολουθεί ο κώδικας, που υλοποιεί την μέθοδο K-fold Cross-Validation

```
% URL and folder information
clearvars;
url = 'http://download.tensorflow.org/example_images/flower_photos.tgz';
downloadFolder = tempdir;
filename = fullfile(downloadFolder,'flower_dataset.tgz');
dataFolder = fullfile(downloadFolder,'flower_photos');
accuraciesVal = zeros(1, 10);
accuraciesTrain=zeros(1, 10);
accuraciesTest=zeros(1, 10);
lossTrain=zeros(1, 10);
lossVal=zeros(1, 10);
% Download the dataset if it doesn't exist
if ~exist(dataFolder,'dir')
   fprintf("Downloading Flowers data set (218 MB)... ")
   websave(filename,url);
   untar(filename,downloadFolder)
   fprintf("Done.\n")
end
% Load images from the ImageDatastore and resize them
images = imageDatastore(dataFolder, ...
   'IncludeSubfolders', true, ...
   'LabelSource', 'foldernames');
%Display some of the images in the datastore.
figure;
perm = randperm(3670,9);
for i = 1:9
   subplot(3,3,i);
   imshow(images.Files{perm(i)});
end
labels=images.Labels;
% Create a path in the current working directory
checkpoint = fullfile(pwd, 'checkpoints1');
labelCount = countEachLabel(images)
% Split the dataset into k folds
k = 10;
indices = crossvalind('Kfold', labels, k);
% Initialize variables to hold accuracy for each fold
```

```
accuracies = zeros(k, 1);
layers = [
   imageInputLayer([256 256 3],"Normalization","zscore")
   convolution2dLayer(3, 32, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 64, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2 )
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 128, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1, 'Name', 'conv_last1')
   batchNormalizationLayer
   reluLayer
   maxPooling2dLayer(2, 'Stride', 2)
   convolution2dLayer(3, 256, 'Padding', 1)
   batchNormalizationLayer
   reluLayer
   averagePooling2dLayer(2, 'Stride', 2)
   flattenLayer('Name', 'flatten')
   fullyConnectedLayer(16)
   dropoutLayer(0.45)
   fullyConnectedLayer(5)
   softmaxLayer
   classificationLayer];
learnRate = 0.001;
%for k times do
for i = 1:k
%split data
splitRatioTrain = 0.8;
splitRatioValidation = 0.1;
splitRatioTest = 0.1;
[trainingImages,validationImages,testImages]=splitEachLabel(images,
splitRatioTrain, splitRatioValidation, splitRatioTest, 'randomized');
 % Use imageDataAugmenter for data augmentation
 augmenter = imageDataAugmenter(...
   'RandXReflection', true, ...
   'RandXTranslation', [-2 2], ...
   'RandYTranslation', [-2 2], ...
```

```
    'RandScale', [0.9 1.2], ...
    'RandRotation', [-30 30]);
%Data Augumentation
augmentedTrainingImages = augmentedImageDatastore([256,256], trainingImages,
'DataAugmentation', augmenter);
augmentedValidationImages=augmentedImageDatastore([256,256],
validationImages, 'DataAugmentation', augmenter);
augmentedTestImages=augmentedImageDatastore([256,256],testImages,
'DataAugmentation', augmenter);
% options = trained.options;
options = trainingOptions('adam', ...
    'MaxEpochs',80, ...
    'InitialLearnRate', learnRate, ...
    'LearnRateDropPeriod',76,...
    'LearnRateDropFactor',0.1,...
    'L2Regularization',0.01,...
    'ValidationData', augmentedValidationImages, ...
    'ValidationFrequency', 50, ...
    'ValidationPatience',15,...
    'OutputNetwork','last-iteration',...
    'LearnRateSchedule', 'piecewise', ...
    'Shuffle', 'every-epoch', ...
    'Verbose',0, ...
    'CheckpointPath', checkpoint, ...
    'Plots', 'training-progress');
lgraph = layerGraph(layers);
% Train the CNN
rng('default');
rng(123);
[net,info] = trainNetwork(augmentedTrainingImages,lgraph,options);
 % Evaluate on validation and training set
    accuraciesTrain(i) = info.TrainingAccuracy(end);
    fprintf('Fold %d Training accuracy: %.2f%%\n',i,info.TrainingAccuracy(end) );
    accuraciesVal(i)= info.ValidationAccuracy(end);
    fprintf('Fold %d Validation accuracy: %.2f%%\n',i,info.ValidationAccuracy(end))
    lossTrain(i) = info.TrainingLoss(end);
    fprintf('Fold %d Training loss: %.2f%%\n', i,info.TrainingLoss(end));
    lossVal(i)= info.ValidationLoss(end);
    fprintf('Fold %d Validation loss: %.2f%%\n',i,info.ValidationLoss(end))
    testLabels = testImages.Labels;
    predictions = classify(net, augmentedTestImages);
    accuracy = mean(predictions == testImages.Labels);
    fprintf('Fold %d Test set accuracy: %.2f%%\n',i, accuracy * 100);
    accuraciesTest(i)=accuracy;
end
%Calculate and display the average accuracy/loss across folds
avgAccuracyTrain = mean(accuraciesTrain);
fprintf('Average Training set accuracy: %.2f%%\n', avgAccuracyTrain);
avgAccuracyVal = mean(accuraciesVal);
fprintf('Average Validation set accuracy: %.2f%%\n', avgAccuracyVal);
avgLossTrain = mean(lossTrain);
```

```
fprintf('Average Training set loss: %.2f%%\n', avgLossTrain);
avgLossVal = mean(lossVal);
fprintf('Average Validation set loss: %.2f%%\n', avgLossVal)
avgAccuracyTest = mean(accuraciesTest);
fprintf('Average Test set accuracy: %.2f%%\n', avgAccuracyTest*100);
```

- **Freezy Layers χρησιμοποιώντας το ίδιο σύνολο δεδομένων**
  Με τον ακόλουθο κώδικα υλοποιείται η μέθοδος Freezy Layers στο ίδιο σύνολο δεδομένων.

```
clc;
clearvars;
% URL and folder information
url = 'http://download.tensorflow.org/example_images/flower_photos.tgz';
downloadFolder = tempdir;
filename = fullfile(downloadFolder,'flower_dataset.tgz');
dataFolder = fullfile(downloadFolder,'flower_photos');
% Download the dataset if it doesn't exist
if ~exist(dataFolder,'dir')
   fprintf("Downloading Flowers data set (218 MB)... ")
   websave(filename,url);
   untar(filename,downloadFolder)
   fprintf("Done.\n")
end
% Load images from the ImageDatastore and resize them
images = imageDatastore(dataFolder, ...
   'IncludeSubfolders', true, ...
   'LabelSource', 'foldernames', ...
   'ReadFcn',@(x) imresize(imread(x), [256,256]));
%Display some of the images in the datastore.
figure;
perm = randperm(3670,9);
for i = 1:9
   subplot(3,3,i);
   imshow(images.Files{perm(i)});
end
%Label count Data
labelCount = countEachLabel(images);
%Split Data
splitRatioTrain = 0.8;
splitRatioValidation = 0.1;
splitRatioTest = 0.1;
[trainingImages, validationImages, testImages] = splitEachLabel(images,
splitRatioTrain, splitRatioValidation, splitRatioTest, 'randomized');
% Load the pre-trained model
modelFileName = 'net.mat';
loadedModel = load(modelFileName ,'net','info','testAccuracy');
pretrainedNet = loadedModel.net;
layers = pretrainedNet.Layers;
%Freeze the layers
```

```
for i = 1:length(layers)
    if isa(layers(i), 'nnet.cnn.layer.FullyConnectedLayer') || ...
       isa(layers(i), 'nnet.cnn.layer.ClassificationLayer')
%Do not freeze the fully connected and classification layers
        continue;
    end
    if isprop(layers(i), 'WeightLearnRateFactor')
        layers(i).WeightLearnRateFactor = 0;
    end
    if isprop(layers(i), 'BiasLearnRateFactor')
        layers(i).BiasLearnRateFactor = 0;
    end
end
LearnRate=0.00001;
%Specify training options for transfer learning
transferLearningOptions = trainingOptions('adam', ...
    'MaxEpochs', 15, ...
    'InitialLearnRate', LearnRate, ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod',11,...
    'ValidationData', validationImages, ...
    'ValidationFrequency', 50, ...
    'LearnRateSchedule', 'piecewise', ...
    'Shuffle', 'every-epoch', ...
    'Plots', 'training-progress', ...
    'Verbose', 0, ...
    'ExecutionEnvironment', 'gpu');
rng('default');
rng(2);
% Train the model on the new dataset
[transferLearningNet,infoLearningNet] = trainNetwork(trainingImages, layers,
transferLearningOptions);
% Evaluate the trained model
fprintf('Training accuracy of the pretrained net:
%.2f%%\n',loadedModel.info.TrainingAccuracy(end) );
fprintf('Training loss of the pretrained net: %.2f%%\n',
loadedModel.info.TrainingLoss(end));
fprintf('Validation accuracy of the pretrained net:
%.2f%%\n',loadedModel.info.ValidationAccuracy(end))
fprintf('Validation loss of the pretrained net:
%.2f%%\n',loadedModel.info.ValidationLoss(end))
fprintf('Test set accuracy of the pretrained net: %.2f%%\n',
loadedModel.testAccuracy);
fprintf('Training accuracy of the freeze net:
%.2f%%\n',infoLearningNet.TrainingAccuracy(end) );
fprintf('Training loss of the freeze net: %.2f%%\n',
infoLearningNet.TrainingLoss(end));
fprintf('Validation accuracy of the freeze net:
%.2f%%\n',infoLearningNet.ValidationAccuracy(end))
fprintf('Validation loss of the freeze net:
%.2f%%\n',infoLearningNet.ValidationLoss(end))
```

```
testLabels = testImages.Labels;
predictions = classify(transferLearningNet, testImages);
accuracy = mean(predictions == testImages.Labels);
fprintf('Test set accuracy of the freeze net: %.2f%%\n', accuracy * 100);
%Analyze network
analyzeNetwork(transferLearningNet );
```

- **Freezy Layers χρησιμοποιώντας διαφορετικό σύνολο δεδομένων**
  Ο κώδικας που παρατίθεται στη συνέχεια υλοποιεί τη μέθοδο Freezy Layers σε διαφορετικό σύνολο δεδομένων.

```
clearvars;
% URL and folder information
dataFolder = './flower_images';
% Download the dataset if it doesn't exist
if ~exist(dataFolder,'dir')
   fprintf("Flowers data set (250 MB)... ")
end
% Load images from the ImageDatastore and resize them
images = imageDatastore(dataFolder, ...
   'IncludeSubfolders', true, ...
   'LabelSource', 'foldernames')
%Display some of the images in the datastore.
figure;
perm = randperm(3670,9);
for i = 1:9
   subplot(3,3,i);
   imshow(images.Files{perm(i)});
end
labelCount = countEachLabel(images);
%Split Data
splitRatioTrain = 0.8;
splitRatioValidation = 0.1;
splitRatioTest = 0.1;
[trainingImages, validationImages, testImages] = splitEachLabel(images,
splitRatioTrain, splitRatioValidation, splitRatioTest, 'randomized');
% Use imageDataAugmenter for data augmentation
augmenter = imageDataAugmenter(...
   'RandXReflection', true, ...
   'RandXTranslation', [-2 2], ...
   'RandYTranslation', [-2 2], ...
   'RandScale', [0.9 1.1], ...
   'RandRotation',[-30 30]);
% Data Augmentation
augmentedTrainingImages = augmentedImageDatastore([256,256], trainingImages,
'DataAugmentation', augmenter);
augmentedValidationImages = augmentedImageDatastore([256,256],
validationImages, 'DataAugmentation', augmenter);
augmentedTestImages = augmentedImageDatastore([256,256], testImages,
'DataAugmentation', augmenter);
```

```
% Load the pre-trained model
modelFileName = 'net.mat';
loadedModel = load(modelFileName ,'net','info','testAccuracy');
pretrainedNet = loadedModel.net;
% Freeze the layers except for the fully connected and classification layers
layers = pretrainedNet.Layers;
for i = 1:length(layers)
   if isa(layers(i), 'nnet.cnn.layer.FullyConnectedLayer') || ...
     isa(layers(i), 'nnet.cnn.layer.ClassificationLayer')
     continue;
   end
   if isprop(layers(i), 'WeightLearnRateFactor')
      layers(i).WeightLearnRateFactor = 0;
   end
   if isprop(layers(i), 'BiasLearnRateFactor')
      layers(i).BiasLearnRateFactor = 0;
   end
end
% Create a new dropoutLayer
newDropoutLayer=dropoutLayer(0.45,"Name",'DropTL');
% Create a new classification layer
newClassificationLayer = classificationLayer();
% Replace the last layer with the new classification layer
layers(end) = newClassificationLayer;
% Replace the dropoutLayer
layers(end-3)=newDropoutLayer;
% Unfreeze specific layers for fine-tuning
% Set to 1 to allow training of weights
% Set to 1 to allow training of biases
for i = 1:length(layers)
   if isprop(layers(i), 'WeightLearnRateFactor')
      layers(i).WeightLearnRateFactor=1;
   end
   if isprop(layers(i), 'BiasLearnRateFactor')
      layers(i).BiasLearnRateFactor=1;
   end
end
LearnRate= 0.0001;
% Specify training options for transfer learning
transferLearningOptions = trainingOptions('adam', ...
   'MaxEpochs', 20, ...
   'InitialLearnRate',LearnRate,...
   'L2Regularization',0.02,...
   'LearnRateDropFactor', 0.1,...
   'LearnRateDropPeriod',17,...
   'ValidationData', augmentedValidationImages, ...
   'ValidationFrequency', 50,...
   'LearnRateSchedule', 'piecewise',...
   'Shuffle', 'every-epoch', ...
   'Plots', 'training-progress', ...
   'Verbose', 0, ...
```

```
    'ExecutionEnvironment', 'gpu'); % Use 'cpu' if you don't have a GPU
rng('default');
rng(123);
% Train the model on the new dataset
[transferLearningNet,infoLearningNet] = trainNetwork(augmentedTrainingImages,
layers, transferLearningOptions);
% Evaluate the trained model on the test set
disp(transferLearningOptions);
% Evaluate the trained model on the test set
fprintf('Training accuracy of the pretrained net:
%.2f%%\n',loadedModel.info.TrainingAccuracy(end) );
fprintf('Training loss of the pretrained net: %.2f%%\n',
loadedModel.info.TrainingLoss(end));
fprintf('Validation accuracy of the pretrained net:
%.2f%%\n',loadedModel.info.ValidationAccuracy(end))
fprintf('Validation loss of the pretrained net:
%.2f%%\n',loadedModel.info.ValidationLoss(end))
fprintf('Test set accuracy of the pretrained net: %.2f%%\n',
loadedModel.testAccuracy);
fprintf('Training accuracy of the freeze net:
%.2f%%\n',infoLearningNet.TrainingAccuracy(end) );
fprintf('Training loss of the freeze net: %.2f%%\n',
infoLearningNet.TrainingLoss(end));
fprintf('Validation accuracy of the freeze net:
%.2f%%\n',infoLearningNet.ValidationAccuracy(end))
fprintf('Validation loss of the freeze net:
%.2f%%\n',infoLearningNet.ValidationLoss(end))
testLabels = testImages.Labels;
predictions = classify(transferLearningNet, augmentedTestImages);
accuracy = mean(predictions == testImages.Labels);
fprintf('Test set accuracy of the freeze net: %.2f%%\n', accuracy * 100);
% Analyze network
analyzeNetwork(transferLearningNet );
```

ΚΕΦΑΛΑΙΟ 6

3. ΠΑΡΑΡΤΗΜΑ Β΄

Στο παράρτημα Β΄ παρατίθεται ο κώδικας υλοποίησης των δυο τύπων επαναλαμβανομένων μοντέλων, LSTM και GRU.

- **Ακολουθεί ο κώδικας, που υλοποιεί το LSTM, μοντέλο πρόβλεψής κρυπτονομισμάτων.**

```
% Loading data from a table
coins = {'bitcoin', 'ethereum', 'BNB'};
train_files = {'bitcoin.csv', 'ethereum classic.csv', 'BNB.csv'};
sequenceLength = 5; % Fixed sequence length
for i = 1:length(coins)
    coin = coins{i};
    train_file = train_files{i};
    fprintf('Processing %s...\n', coin);
    data = readtable(train_file);
    % Select relevant columns
    data = data(:, {'Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Currency'});
    % Convert Date column to datetime format
    data.Date = datetime(data.Date);
    % Sort data by Date (if not already sorted)
    data = sortrows(data, 'Date');
    % Remove rows with missing values
    data = rmmissing(data);
    % Extract date-related features
    data.Year = year(data.Date);
    data.Month = month(data.Date);
    data.Day = day(data.Date);
    data.DayOfWeek = weekday(data.Date);
    data.DayOfYear = day(data.Date, 'dayofyear');
    % Normalize numerical columns (Open, High, Low, Close, Volume) within the table
using z-score normalization
    numCols = {'Open', 'High', 'Low', 'Close', 'Volume'};
    means = zeros(1, length(numCols));
    stds = zeros(1, length(numCols));
    for j = 1:length(numCols)
        colName = numCols{j};
        means(j) = mean(data.(colName));
        stds(j) = std(data.(colName));
        data.(colName) = (data.(colName) - means(j)) / stds(j);
    end
    % Optionally, convert Currency to categorical if needed
    data.Currency = categorical(data.Currency);
    % Features that remain constant for all time steps can negatively impact the training
    % Remove rows where features remain constant
```

```
    constantRows = false(height(data), 1);
    for j = 1:length(numCols)
        colName = numCols{j};
        constantRows = constantRows | (data.(colName) == min(data.(colName)) &
data.(colName) == max(data.(colName)));
    end
    data = data(~constantRows, :);
    % Define the LSTM network architecture
    numFeatures = length(numCols); % Number of input features
    numHiddenUnits = 50;
    layers = [
        sequenceInputLayer(numFeatures)
       lstmLayer(numHiddenUnits, 'OutputMode','last')
        fullyConnectedLayer(1)
        regressionLayer
    ];
    % Prepare sequences for LSTM input
    numObservations = height(data) - sequenceLength;
    X = cell(numObservations, 1);
    Y = zeros(numObservations, 1); % Change Y to numeric array
   % Close value is the target, assigned as a scalar
   % Adjust to match sequence length and feature dimensions
    for j = 1:numObservations
        X{j} = data{j:j+sequenceLength-1, numCols};
        Y(j) = data.Close(j+sequenceLength);
    end
    % Split data into training and validation sets
    trainSize = floor(0.9 * numObservations);
    XTrain = X(1:trainSize);
    YTrain = Y(1:trainSize);
    XVal = X(trainSize+1:end);
    YVal = Y(trainSize+1:end);
    % Set a different learning rate for bitcoin
    if strcmp(coin, 'bitcoin')
        initialLearnRate = 0.0001;
        maxEpochs=20;
        regularization=0;
        gradient=0.999;
        batchsize=64;
        dropPeriod=10;
        dropFactor=0.5;
     elseif strcmp(coin,'ethereum')
        initialLearnRate = 0.005;
        maxEpochs=50;
        regularization=0.01;
        gradient=0.9;
        batchsize=64;
        dropPeriod=51;
     else
         initialLearnRate = 0.002;
         maxEpochs=50;
```

```
        regularization=0.01;
        gradient=0.9;
        batchsize=64;
        dropPeriod=51;
    end
    % Train options
    options = trainingOptions('adam', ...
      'MaxEpochs', maxEpochs, ...
      'MiniBatchSize',batchsize, ...
      'L2Regularization',regularization,...
      'InitialLearnRate', initialLearnRate, ...
      'LearnRateDropPeriod',dropPeriod,...
      'LearnRateDropFactor',dropFactor,...
      'SquaredGradientDecayFactor',gradient,...
      'GradientThresholdMethod','global-l2norm',...
      'SequenceLength', sequenceLength, ...
      'ValidationData', {XVal, YVal}, ...
      'Plots', 'training-progress');
    rng('default');
    rng(123);
    % Train the LSTM network
    [net,inforatio] = trainNetwork(XTrain, YTrain, layers, options);
     % Display final training loss and RMSE
    fprintf('Final Training Loss: %.4f\n', inforatio.TrainingLoss(end));
    fprintf('Final Training RMSE: %.4f\n', inforatio.TrainingRMSE(end));
    fprintf('Final Validation Loss: %.4f\n',inforatio.ValidationLoss(end));
    fprintf('Final Validation RMSE: %.4f\n',inforatio.ValidationRMSE(end));
    % Predict on validation set
    YPred = predict(net, XVal, 'MiniBatchSize', 1);
    % Denormalize the predictions using the stored mean and standard deviation
    closeMean = means(strcmp(numCols, 'Close'));
    closeStd = stds(strcmp(numCols, 'Close'));
    YDenorm = YPred * closeStd + closeMean;
    % Denormalize the actual values
    YValDenorm = YVal * closeStd + closeMean;
    % Plot predictions vs actual values
    figure;
    plot(data.Date(trainSize+sequenceLength+1:end), YValDenorm, 'g', 'LineWidth',
1.5);
    hold on;
    plot(data.Date(trainSize+sequenceLength+1:end), YDenorm, 'r--', 'LineWidth', 1.5);
    grid on;
    legend('Actual Close', 'Predicted Close', 'Location', 'best');
    xlabel('Date');
    ylabel('Close Value');
    title(sprintf('%s LSTM Prediction of Close Value', coin));
    % Rolling Predictions for Future Dates
    numFutureSteps = 10; % Number of future steps to predict
    futurePreds = zeros(numFutureSteps, 1);
    % Get the last sequence from the training data
    lastSequence = data{end-sequenceLength+1:end,numCols};
```

```
    % Predict the next close value
    for k = 1:numFutureSteps
        futurePred = predict(net, {lastSequence}, 'MiniBatchSize', 1);
        futurePreds(k) = futurePred;
        % Update the sequence with the new prediction
        lastSequence = [lastSequence(2:end, :); [futurePred, lastSequence(end, 2:end)]];
    end
    % Denormalize the future predictions
    futurePredsDenorm = futurePreds * closeStd + closeMean;
    % Generate future dates for plotting
    futureDates = (data.Date(end) + caldays(1:numFutureSteps))';
    % Plot future predictions
    figure;
    plot(futureDates, futurePredsDenorm, 'b', 'LineWidth', 2.5);
    hold on;
    plot(data.Date, data.Close * closeStd + closeMean, 'g', 'LineWidth', 1.5);
    grid on;
    legend('Future Predictions', 'Historical Close', 'Location', 'best');
    xlabel('Date');
    ylabel('Close Value');
    title(sprintf('%s Future Predictions of Close Value', coin));
end
```

- **Στη συνέχεια παρατίθεται ο κώδικας που υλοποιεί το GRU, μοντέλο πρόβλεψης χρονοσειρών.**

```
% Loading data from table
coins = {'bitcoin', 'ethereum', 'BNB'};
train_files = {'bitcoin.csv', 'ethereum classic.csv', 'BNB.csv'};
% Fixed sequence length
sequenceLength = 5;
for i = 1:length(coins)
    coin = coins{i};
    train_file = train_files{i};
    fprintf('Processing %s...\n', coin);
    data = readtable(train_file);
    % Select relevant columns
    data = data(:, {'Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Currency'});
    % Convert Date column to datetime format
    data.Date = datetime(data.Date);
    % Sort data by Date (if not already sorted)
    data = sortrows(data, 'Date');
    % Remove rows with missing values
    data = rmmissing(data);
    % Extract date-related features
    data.Year = year(data.Date);
    data.Month = month(data.Date);
    data.Day = day(data.Date);
    data.DayOfWeek = weekday(data.Date);
    data.DayOfYear = day(data.Date, 'dayofyear');
```

```
    % Normalize numerical columns (Open, High, Low, Close, Volume) within the table
using z-score normalization
    numCols = {'Open', 'High', 'Low', 'Close', 'Volume'};
    means = zeros(1, length(numCols));
    stds = zeros(1, length(numCols));
    for j = 1:length(numCols)
        colName = numCols{j};
        means(j) = mean(data.(colName));
        stds(j) = std(data.(colName));
        data.(colName) = (data.(colName) - means(j)) / stds(j);
    end
    % Optionally, convert Currency to categorical if needed
    data.Currency = categorical(data.Currency);
    % Features that remain constant for all time steps can negatively impact the training
    % Remove rows where features remain constant
    constantRows = false(height(data), 1);
    for j = 1:length(numCols)
        colName = numCols{j};
        constantRows = constantRows | (data.(colName) == min(data.(colName)) &
data.(colName) == max(data.(colName)));
    end
    data = data(~constantRows, :);
    % Define the GRU network architecture
    numFeatures = length(numCols); % Number of input features
    numHiddenUnits = 50;
    layers = [
        sequenceInputLayer(numFeatures)
        gruLayer(numHiddenUnits, 'OutputMode','last')
        fullyConnectedLayer(1)
        regressionLayer
    ];
    % Prepare sequences for GRU input
    numObservations = height(data) - sequenceLength;
    X = cell(numObservations, 1);
    Y = zeros(numObservations, 1);
    % Close value is the target, assigned as a scalar
    % Adjust to match sequence length and feature dimensions
    for j = 1:numObservations
        X{j} = data{j:j+sequenceLength-1, numCols};
        Y(j) = data.Close(j+sequenceLength);
    end
    % Split data into training and validation sets
    trainSize = floor(0.9 * numObservations);
    XTrain = X(1:trainSize);
    YTrain = Y(1:trainSize);
    XVal = X(trainSize+1:end);
    YVal = Y(trainSize+1:end);
    % Set a different learning rate for bitcoin
    if strcmp(coin, 'bitcoin')
        initialLearnRate = 0.0009;
        maxEpochs=20;
```

```
    regularization=0.0001;
    gradient=0.1;
    batchsize=64;
    dropPeriod=10;
    dropFactor=0.5;
 elseif strcmp(coin, 'ethereum')
    initialLearnRate = 0.003;
    maxEpochs=50;
    regularization=0.001;
    gradient=0.9;
    batchsize=64;
    dropPeriod=51;
 else
     initialLearnRate = 0.003;
     maxEpochs=50;
     regularization=0.001;
     gradient=0.9;
     batchsize=64;
     dropPeriod=51;
 end
% Train options
options = trainingOptions('adam', ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize',batchsize,...
    'L2Regularization',regularization,...
    'InitialLearnRate', initialLearnRate, ...
    'LearnRateDropPeriod',dropPeriod,...
    'LearnRateDropFactor',dropFactor,...
    'SquaredGradientDecayFactor',gradient,...
    'GradientThresholdMethod','global-l2norm',...
    'SequenceLength', sequenceLength, ...
    'ValidationData', {XVal, YVal}, ...
    'Plots', 'training-progress');
rng('default');
rng(123);
% Train the LSTM network
[net,inforatio] = trainNetwork(XTrain, YTrain, layers, options);
 % Display final training loss and RMSE
fprintf('Final Training Loss: %.4f\n',inforatio.TrainingLoss(end));
fprintf('Final Training RMSE: %.4f\n',inforatio.TrainingRMSE(end));
fprintf('Final Validation Loss: %.4f\n',inforatio.ValidationLoss(end));
fprintf('Final Validation RMSE: %.4f\n',inforatio.ValidationRMSE(end));
% Predict on validation set
YPred = predict(net, XVal, 'MiniBatchSize', 1);
% Denormalize the predictions using the stored mean and standard deviation
closeMean = means(strcmp(numCols,'Close'));
closeStd = stds(strcmp(numCols,'Close'));
YDenorm = YPred * closeStd + closeMean;
% Denormalize the actual values
YValDenorm = YVal * closeStd + closeMean;
% Plot predictions vs actual values
```

```
    figure;
    plot(data.Date(trainSize+sequenceLength+1:end),YValDenorm,'g','LineWidth',
1.5);
    hold on;
    plot(data.Date(trainSize+sequenceLength+1:end),YDenorm,'r--','LineWidth', 1.5);
    grid on;
    legend('Actual Close','Predicted Close','Location','best');
    xlabel('Date');
    ylabel('Close Value');
    title(sprintf('%s GRU Prediction of Close Value',coin));
    % Rolling Predictions for Future Dates
    numFutureSteps = 10; % Number of future steps to predict
    futurePreds = zeros(numFutureSteps, 1);
    % Get the last sequence from the training data
    lastSequence = data{end-sequenceLength+1:end, numCols};
    % Predict the next close value
    for k = 1:numFutureSteps
        futurePred = predict(net, {lastSequence},'MiniBatchSize', 1);
        futurePreds(k) = futurePred;
        % Update the sequence with the new prediction
        lastSequence = [lastSequence(2:end, :); [futurePred,lastSequence(end, 2:end)]];
    end
    % Denormalize the future predictions
    futurePredsDenorm = futurePreds * closeStd + closeMean;
    % Generate future dates for plotting
    futureDates = (data.Date(end) + caldays(1:numFutureSteps))';
    % Plot future predictions
    figure;
    plot(futureDates, futurePredsDenorm,'b','LineWidth',3.5);
    hold on;
    plot(data.Date, data.Close * closeStd + closeMean,'g','LineWidth', 1.5);
    grid on;
    legend('Future Predictions', 'Historical Close','Location','best');
    xlabel('Date');
    ylabel('Close Value');
    title(sprintf('%s Future Predictions of Close Value',coin));
end
```

ΚΕΦΑΛΑΙΟ 7

4. ΒΙΒΛΙΟΓΡΑΦΙΑ.

[1] PALUSZEK .M, THOMAS S., MATLAB Machine Learning Recipes, A Problem-Solution Approach, Second Edition.

[2] PALUSZEK .M, THOMAS S., Practical MATLAB Deep Learning, a Project-Based Approach.

[3] https://www.tensorflow.org/datasets/catalog/tf_flowers

[4] https://www.kaggle.com/datasets/kaushiksuresh147/top-10-cryptocurrencies-historical-dataset

[5] https://www.kaggle.com/datasets/kausthubkannan/5-flower-types-classification-dataset?resource=download

[6] https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

[7] https://en.wikipedia.org/wiki/Perceptron

[8] https://en.wikipedia.org/wiki/Convolutional_neural_network

[9] https://en.wikipedia.org/wiki/Recurrent_neural_network

[10] https://en.wikipedia.org/wiki/Long_short-term_memory

[11] https://en.wikipedia.org/wiki/Gated_recurrent_unit

[12] https://en.wikipedia.org/wiki/Activation_function

[13] https://www.mathworks.com/help/deeplearning/ug/deep-learning-in-matlab.html

[14] https://www.mathworks.com/help/deeplearning/gs/create-simple-deep-learning-classification-network.html

[15] https://www.mathworks.com/help/deeplearning/ref/imagedataaugmenter.html

[16] https://www.mathworks.com/matlabcentral/fileexchange/78020-oversampling-for-_deep-learning-classification-example

[17] https://www.mathworks.com/discovery/cross-validation.html

[18] https://www.mathworks.com/discovery/transfer-learning.html

[19] Transfer Learning - MATLAB & Simulink (mathworks.com)

[20] https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html

[21] https://www.mathworks.com/help/deeplearning/ref/dlarray.gru.html

[22] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.grulayer.html

[23] https://www.mathworks.com/help/curvefit/exploring-and-customizing-plots.html