ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΥΛΟΠΟΙΗΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΣΕ ΧΑΜΗΛΟΤΕΡΑ ΕΠΙΠΕΔΑ ΑΦΑΙΡΕΣΗΣ

ΦΙΛΙΠΠΟΣ ΠΑΠΠΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΑΝΤΩΝΙΟΣ ΔΑΔΑΛΙΑΡΗΣ

Λαμία, 2022-2023

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΥΛΟΠΟΙΗΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΣΕ ΧΑΜΗΛΟΤΕΡΑ ΕΠΙΠΕΔΑ ΑΦΑΙΡΕΣΗΣ

ΦΙΛΙΠΠΟΣ ΠΑΠΠΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΑΝΤΩΝΙΟΣ ΔΑΔΑΛΙΑΡΗΣ

Λαμία, 2022-2023

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

IMPLEMENTATION OF NEURAL NETWORKS IN LOWER LEVELS OF ABSTRACTION

FILIPPOS PAPPAS

FINAL THESIS

ADVISOR

ANTONIOS DADALIARIS

Lamia, 2022-2023

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 3/1/2023

Ο Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η υλοποίηση νευρωνικών δικτύων αποτελεί καίριο παράγοντα στην Πληροφορική καθώς ο κλάδος της Τεχνητής Νοημοσύνης αναπτύσσεται ραγδαία. Η προγραμματιστική υλοποίηση αυτών αποτελείται σε μεγάλο βαθμό από γλώσσες προγραμματισμού υψηλού επιπέδου. Το μοντέλο που παρουσιάζεται, αποτελεί την Βελτιστοποίηση που μπορούν να επιτύχουν τα νευρωνικά δίκτυα όταν υλοποιούνται σε γλώσσες προγραμματισμού χαμηλού επιπέδου (C++ στην δική μας περίπτωση). Η σύγκριση των Νευρωνικών Δικτύων γίνεται με την γλώσσα προγραμματισμού Python. Η υλοποίηση είναι αρκετά όμοια, με την διαφορά πως για την C++ απαιτήθηκε η δημιουργία μίας μαθηματικής βιβλιοθήκης στην οποία και πραγματοποιήθηκαν βελτιστοποιήσεις στοχευμένα για την χρήση νευρωνικών δικτύων. Σκοπός αυτής της εργασίας είναι να δημιουργηθεί ένα πιο κατανοητό και (προγραμματιστικά) αποσαφηνισμένο μοντέλο έτσι ώστε να υπάρχει η δυνατότητα για περεταίρω βελτιστοποίηση στην μοντελοποίηση Νευρωνικών Δικτύων χαμηλού επιπέδου υψηλής απόδοσης.

ABSTRACT

The implementation of neural networks is a key factor in computer science as the field of artificial intelligence is developing rapidly. The programming implementation of these, consists largely of high-level programming languages. The presented Model consists of the optimization that the neural networks can achieve, when implemented in low level programming languages (C++ in our case). The comparative programming language for this model is Python. The implementation is similar with respect to architecture apart from the fact that a Mathematical Library needed to be developed for C++ in which specific optimizations took place for implementing artificial neural networks. The purpose of this paper is to create a more understandable and (programmatically) clarified model so that there is a possibility for further optimization in the modelling of low-level high-performance Neural Networks.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Το Πρόβλημα

1.1 Το Πρόβλημα της αναγνώρισης χειρόγραφων ψηφίων

1.1.α Ορισμός

Η αναγνώριση χειρόγραφών ψηφίων αποτελεί ένα πολύ γνωστό πρόβλημα στην Επιστήμη των υπολογιστών και έχει μελετηθεί εδώ και αρκετές δεκαετίες. Ο στόχος αυτού του προβλήματος είναι η δημιουργία ενός συστήματος που να μπορεί να αναγνωρίζει με ακρίβεια χειρόγραφα ψηφία, όπως αυτά που βρίσκονται σε μία επιταγή ή μια ταχυδρομική διεύθυνση. Αυτή η Διαδικασία είναι σημαντική για διάφορες εφαρμογές, συμπεριλαμβανομένης της επεξεργασίας εγγράφων, του ταχυδρομικού αυτοματισμού, ακόμη και της ρομποτικής.

Η Αναγνώριση Χειρόγραφων Ψηφίων είναι ένα δύσκολο πρόβλημα για διάφορους λόγους. Η ανθρώπινη γραφή είναι μεταβαλλόμενη σε μεγάλο βαθμό, καθώς κάθε άτομο έχει μοναδικό τρόπο γραφής για το ίδιο ψηφίο. Αυτό καθιστά δύσκολο για ένα μηχάνημα να αναγνωρίσει το ψηφίο με υψηλό βαθμό ακρίβειας.

Τα χειρόγραφα ψηφία μπορούν να επηρεαστούν από διάφορους παράγοντες όπως ο θόρυβος (χαμηλή ανάλυση, κακό μολύβι ή στυλό κ.α), οι μουτζούρες και οι διακυμάνσεις στην ένταση του μελανιού, κάνοντας το έργο της αναγνώρισης ακόμη πιο δύσκολο

Τα χειρόγραφα ψηφία μπορούν επίσης να γραφτούν σε διαφορετικούς προσανατολισμούς, μεγέθη και κλίμακες, γεγονός που επηρεάζει την ακρίβεια της αναγνώρισης.

1.1.β Επίλυση του Προβλήματος

Η επίτευξη αξιόπιστης αναγνώρισης και κατάταξης των ψηφίων στις αντίστοιχες κλάσεις αποτελεί κλασσικό πρόβλημα βαθιάς μάθησης (Deep Learning).

Η βαθιά μάθηση περιλαμβάνει υπολογιστικά μοντέλα που αποτελούνται από πολλαπλά επίπεδα επεξεργασίας τα οποία «μαθαίνουν» αναπαραστάσεις δεδομένων. Η χρήση της βαθιάς μάθησης έχει συμβάλει δραστικά στη βελτίωση διαδικασιών που δυσκολεύουν την υπολογιστική μηχανή. Τέτοιες διαδικασίες είναι η αναγνώριση ομιλίας, η ανακάλυψη φαρμάκων, η ανίχνευση αντικειμένων και στην περίπτωση που εξετάζουμε, η οπτική αναγνώριση ψηφίων. Η βαθιά μάθηση εφευρίσκει ένα σύνθετο μοντέλο χρησιμοποιώντας ένα μεγάλο σύνολο δεδομένων στο οποίο εφαρμόζει τον αλγόριθμό **Σταδιακής Καθίζησης** (Gradient Descent, Κεφάλαιο 2.1.Γ) προκειμένου να βρει τον βαθμό τροποποίησης των εσωτερικών παραμέτρων που χρειάζεται ανάλογα με την περίπτωση. Οι εσωτερικές παράμετροι χρησιμοποιούνται για τον υπολογισμό της εισόδου ενός επιπέδου (Input Layer) βάσει της αναπαράστασης της εξόδου (Output Layer).

Για την επίτευξη αυτών των αποτελεσμάτων, ιδιαίτερα στον τομέα της οπτικής αναγνώρισης, χρησιμοποιούνται τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network) ή τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks).

1.1.γ Προτεινόμενη Λύση

Χάριν στο μικρό αριθμό κλάσεων στις οποίες γίνεται ο διαμοιρασμός των αριθμών (δέκα), αξίζει να χρησιμοποιηθούν τα Τεχνητά Νευρωνικά Δίκτυα (ANN's) για τους εξής λόγους:

- Τα Τεχνητά Νευρωνικά Δίκτυα μπορούν να πετύχουν πάνω από Ενενήντα τοις εκατό (> 90%) ποσοστό επιτυχίας πρόβλεψης όσον αφορά τις χειρόγραφες εικόνες ψηφίων της βάσης δεδομένων του MNIST. Περεταίρω ανάλυση στο Κεφάλαιο 2 «Θεωρητικό Υπόβαθρο»
- Τα Συνελικτικά Νευρωνικά Δίκτυα παρουσιάζουν μεγάλο βαθμό πολυπλοκότητας όσον αφορά την υλοποίηση τους, το οποίο θα καθυστερούσε πάρα πολύ την διάρκεια που θα τρέχει ο αλγόριθμος. Περεταίρω ανάλυση στο Κεφάλαιο 2 «Θεωρητικό Υπόβαθρο»

Καθώς τα Τεχνητά Νευρωνικά Δίκτυα μπορούν με την απλότητάς να πετύχουν σχεδόν ισάξια αποτελέσματα με τα Συνελικτικά Νευρωνικά Δίκτυα στην Αναγνώριση Χειρόγραφων Ψηφίων και ο σκοπός αυτής της έρευνας είναι να συγκριθούν οι δύο γλώσσες Προγραμματισμού, αποφασίστηκε να χρησιμοποιηθούν τα Τεχνητά Νευρωνικά Δίκτυα.

ΚΕΦΑΛΑΙΟ 2 Θεωρητικό Υπόβαθρο

2.1 Τεχνητή Νοημοσύνη

2.1.α Τι είναι η Τεχνητή Νοημοσύνη – Ορολογία

Η Τεχνητή Νοημοσύνη (Artificial Intelligence – AI) είναι ένας ευρύς όρος που χρησιμοποιείται για πολλές διαφορετικές τεχνολογίες. Ο John McCarthy έχει προσφέρει τον ακόλουθο ορισμό:

«Είναι η επιστήμη και η μηχανική της κατασκευής έξυπνων μηχανών, ιδιαίτερα ευφυών προγραμμάτων υπολογιστών. Σχετίζεται με το παρόμοιο καθήκον της χρήσης υπολογιστών για την κατανόηση της ανθρώπινης νοημοσύνης, αλλά η τεχνητή νοημοσύνη δεν χρειάζεται να περιοριστεί σε μεθόδους που είναι βιολογικά παρατηρήσιμες».

Ωστόσο, δεκαετίες πριν από τη διαμόρφωση αυτού του ορισμού, η Τεχνητή Νοημοσύνη προέκυψε ως ανακάλυψη από το θεμελιώδες έργο του Άλαν Τούρινγκ, «Υπολογιστικές Μηχανές και Νοημοσύνη», το οποίο δημοσιεύτηκε το 1950. Σε αυτό το έγγραφο, ο Turing, που αναφέρεται συχνά και ως «Πατέρας της Επιστήμης των Υπολογιστών» θέτει την ακόλουθη ερώτηση:

«Μπορούν οι Μηχανές να σκεφτούν;»

Με αφορμή αυτή την ερώτηση, προσφέρει ένα τεστ, γνωστό πλέον ως «Turing Test», όπου ένας ανθρώπινος ανακριτής θα προσπαθούσε θα προσπαθούσε να διακρίνει μεταξύ μιας απάντησης σε υπολογιστή και ανθρώπινου κειμένου. Αν και αυτό το τεστ έχει υποβληθεί σε μεγάλο έλεγχο μετά από τη δημοσίευσή του, παραμένει ένα σημαντικό μέρος της ιστορίας της τεχνητής νοημοσύνης καθώς και μία συνεχής ιδέα στη φιλοσοφία, καθώς χρησιμοποιεί ιδέες γύρω από τη γλωσσολογία.

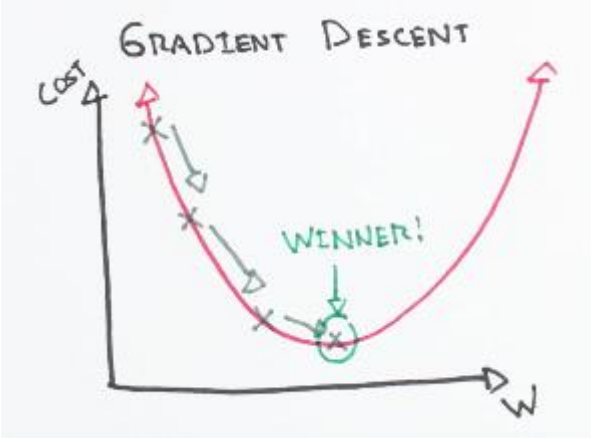
2.1.β Μηχανική Μάθηση – Βαθιά Μάθηση

Η Μηχανική Μάθηση (Machine Learning) και η Βαθιά Μάθηση (Deep Learning) είναι υποκατηγορίες της Τεχνητής Νοημοσύνης και στην πραγματικότητα η Βαθιά Μάθηση είναι υποκατηγορία της Μηχανικής Μάθησης.

Η Βαθιά Μάθηση στην πραγματικότητα αποτελείται από Νευρωνικά Δίκτυα. Ο όρος «Βαθιά» στο «Βαθιά Μάθηση» αναφέρεται σε ένα Νευρωνικό Δίκτυο που αποτελείται από τρία ή περισσότερα Επίπεδα (τα οποία περιλαμβάνουν την είσοδο και την έξοδο) μπορεί να θεωρηθεί Αλγόριθμος Βαθιάς Μάθησης.

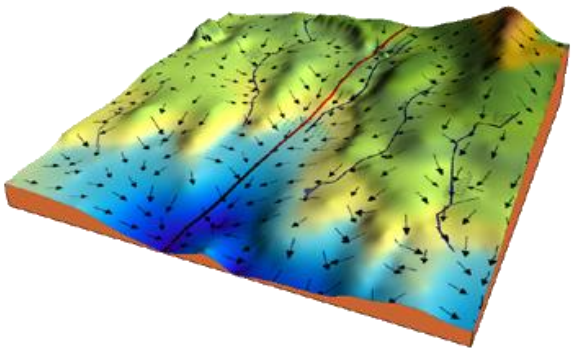
2.1.γ Σταδιακή Καθίζηση

Η Σταδιακή Καθίζηση (Gradient Descent) είναι ένας αλγόριθμος βελτιστοποίησης ο οποίος χρησιμοποιείται για την ελαχιστοποίησης κάποιας συνάρτησης κόστους αξιοποιώντας μία επαναληπτική κίνηση προς την κατεύθυνση της πιο απότομης κατάβασης χρησιμοποιώντας την αρνητική κλίση. Στην ουσία, ο σκοπός του αλγόριθμου είναι να βρει ένα τοπικό ελάχιστο (Local Minimum). Ο αλγόριθμος αυτός δεν βρίσκει απαραίτητα την τέλεια λύση καθώς το Τοπικό Ελάχιστο δεν τυχαίνει πάντα να είναι και το ολικό (Παράδειγμα στην Εικόνα δύο (2) φαίνεται πως ο Αλγόριθμός έχει πάνω από ένα τοπικό ελάχιστο το οποίο θα μπορούσε να εγκλωβίσει τον Αλγόριθμο ώστε να μην δώσει την βέλτιστη λύση). Στην περίπτωση της Μηχανικής Μάθησης, χρησιμοποιείται ο αλγόριθμος της Σταδιακής Καθίζησης για να ενημερωθούν οι παράμετροι του που χρησιμοποιούνται στο μοντέλο. Οι παράμετροι αναφέρονται σε συντελεστές στην Γραμμική Παλινδρόμηση και στα Νευρωνικά Δίκτυα.



Εικόνα (1) – ML Glossary Documentation

Για παράδειγμα, μπορούμε να υποθέσουμε ένα τρισδιάστατο Γράφημα (Εικόνα 2) στο οποίο καλούμαστε να βελτιστοποιήσουμε την θέση μας ως προς το ύψος ξεκινώντας από το πιο ψηλό σημείο (Πάνω Δεξιά Γωνία). Τα βέλη αναπαριστούν το σημείο στο οποίο η Κατεύθυνση είναι πιο απότομη (Αρνητική Κλίση) από οποιοδήποτε σημείο, δηλαδή το μονοπάτι που (πιθανώς) θα ελαχιστοποιήσει την συνάρτηση κόστους το συντομότερο.

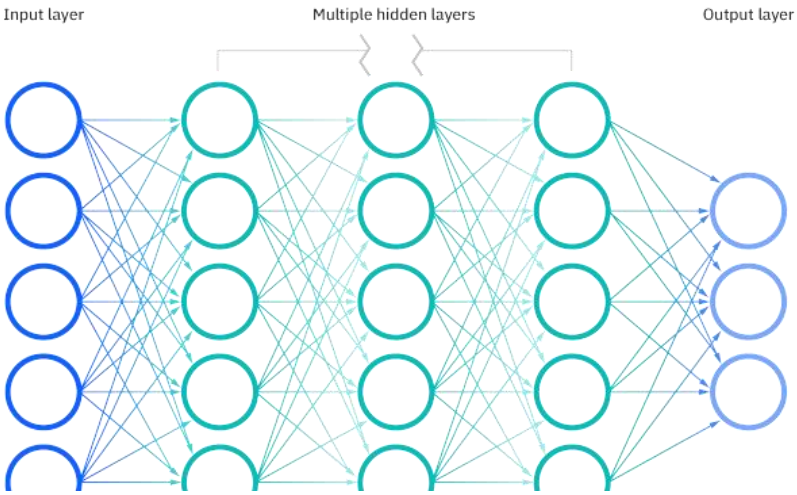


Εικόνα (2) – ML Glossary Documentation

Για να μπορέσουμε να φτάσουμε στο τοπικό ελάχιστο θα πρέπει να κάνουμε κάποια «βήματα» προς την κατεύθυνση που μας δείχνει η κλίση. Το μέγεθος αυτών των βημάτων ονομάζεται Ρυθμός Εκμάθησης (Learning Rate). Με υψηλό ποσοστό εκμάθησης μπορούμε να καλύψουμε περισσότερο έδαφος με κάθε βήμα, αλλά κινδυνεύουμε να προσπεράσουμε το Τοπικό Ελάχιστο καθώς η κλίση του λόφου αλλάζει συνεχώς. Μπορούμε να κινηθούμε με απόλυτη σιγουριά προς το ανάλογο Τοπικό Ελάχιστο χρησιμοποιώντας ένα πολύ μικρό ποσοστό εκμάθησης, αλλά ακόμα και σε αυτή την περίπτωση μπορεί να καθυστερήσουμε πάρα πολύ να φτάσουμε στο Τοπικό Ελάχιστο καθώς τα βήματα που κάνουμε είναι πολύ μικρά. Μία συνήθης τακτική είναι να χρησιμοποιούμε μεταβαλλόμενο ρυθμό εκμάθησης ή πειραματικά να βρούμε έναν ο οποίος κάνει τόσο μικρά βήματα ώστε να είναι παραγωγικός, αλλά ταυτόχρονα τόσο μεγάλος ώστε να μην χρονοτριβεί πολύ στην εύρεση του Τοπικού Ελάχιστου. Όσον αφορά τον Μεταβαλλόμενο Ρυθμό Εκμάθησης, τα βήματα ξεκινάνε μεγάλα και με την πάροδο του χρόνου μικραίνουν καθώς πλησιάζουμε κάποιο Ελάχιστο.

2.1.δ Νευρωνικά Δίκτυα

Τεχνητά Νευρωνικά Δίκτυα



Εικόνα (3) – IBM

Τα Τεχνητά Νευρωνικά Δίκτυα αποτελούν ένα υποσύνολο της Μηχανικής Μάθησης (Machine Learning) και βρίσκονται στο επίκεντρο των αλγορίθμων Βαθιάς Μάθησης (Deep Learning). Το όνομά τους είναι εμπνευσμένο από τον ανθρώπινο εγκέφαλο, από τον οποίο μιμούνται τον τρόπο που οι βιολογικοί νευρώνες δίνουν σήμα ο ένας στον άλλο.

Τα Τεχνητά Νευρωνικά Δίκτυα αποτελούνται από στρώματα (Layers) κόμβων τα οποία με τη σειρά τους απαρτίζονται από το Στρώμα Εισόδου (Input Layer), ένα ή και περισσότερα κρυφά στρώματα (Hidden Layers) και ένα Στρώμα Εξόδου (Output Layer). Κάθε Κόμβος, ή τεχνητός Νευρώνας, συνδέεται με έναν άλλο κόμβο του επόμενου Στρώματος. Η σύνδεση αυτή αποτελείται από ένα σχετικό βάρος και ένα Κατώφλι (Threshold). Εάν η έξοδος οποιουδήποτε μεμονωμένου κόμβου είναι πάνω από την καθορισμένη τιμή του κατωφλίου, ο κόμβος αυτός ενεργοποιείται, στέλνοντας δεδομένα στο επόμενο επίπεδο του δικτύου. Αυτό το βήμα έχεις και μία ακόμη υλοποίηση στην οποία δεν είναι δυαδικός ο ρόλος ενός Νευρώνα, αλλά ποσοστιαίος, δηλαδή αντί να υπάρχει ένα κατώφλι, κάθε κόμβος περνάει τα δεδομένα του στο επόμενο στρώμα, αλλά η επίδραση των δεδομένων είναι ανάλογη του βάρους του κόμβου. (Η συγκεκριμένη υλοποίηση επιλέχθηκε και στην δική μας περίπτωση).

Τα Τεχνητά Νευρωνικά δίκτυα βασίζονται σε δεδομένα εκπαίδευσης για να εμβαθύνουν και να βελτιώσουν την ακρίβειά τους με την πάροδο του χρόνου. Ωστόσο, από τη στιγμή που αυτοί οι αλγόριθμοι εκμάθησης βελτιστοποιηθούν ως προς την ακρίβεια, τα Τεχνητά Νευρωνικά Δίκτυα αποτελούν ισχυρά εργαλεία στον κλάδο της Επιστήμης των Υπολογιστών καθώς και στην Τεχνητή Νοημοσύνη, επιτρέποντάς μας να ταξινομούμε και να ομαδοποιούμε δεδομένα με υψηλή ταχύτητα. Οι εργασίες που αφορούν την αναγνώριση ομιλίας ή εικόνων μπορεί να διαρκέσουν λεπτά αντί για αρκετές ώρες, ή και μέρες που πιθανώς θα διαρκούσαν με την χειροκίνητη αναγνώριση από ειδικούς. Ένα από τα ευρέως διαδεδομένα Νευρωνικά Δίκτυα είναι ο αλγόριθμος αναζήτησης της Google.

Υλοποίηση Τεχνητών Νευρωνικών Δικτύων

Στρώμα Εισόδου (Input Layer)
Το Στρώμα Εισόδου αποτελεί την είσοδο του Νευρωνικού Δικτύου και συνήθως αποτελείται από έναν πίνακα (Vector ή και Array) μίας Διάστασης.

Κρυφό Στρώμα (Hidden Layer)
Ένα Νευρωνικό Δίκτυο μπορεί να αποτελείται από ένα ή πολλά κρυφά στρώματα. Τα κρυφά στρώματα αποτελούνται από έναν πίνακα (Vector ή και Array) μίας Διάστασης.

Στρώμα Εξόδου (Output Layer)
Το στρώμα εξόδου αποτελεί την έξοδο του Νευρωνικού δικτύου και συνήθως αποτελείται από έναν πίνακα (Vector ή και Array) μίας Διάστασης.

Πίνακας Βαρών (Weight Matrix)
Ο πίνακας Βαρών αποτελεί την διασύνδεση μεταξύ **δύο** Στρωμάτων στα Νευρωνικά Δίκτυα. Απαρτίζεται από έναν Πίνακα (Matrix) δύο διαστάσεων ο οποίος αναπαριστά το ευαισθησία ενεργοποίησης ενός Νευρώνα Εξόδου. (Θα δοθεί παράδειγμα παρακάτω, στην διαδικασία Εμπροσθοτροφοδότησης.)

Συνάρτηση Ενεργοποίησης (Activation Function)
Η Συνάρτηση Ενεργοποίησης αποτελεί μία διαδικασία η οποία δέχεται ως είσοδο μία Παράμετρο και εφαρμόζει μία **Μη** Γραμμική Συνάρτηση για να διαμορφώσει την Έξοδο. Η μη Γραμμικότητα είναι αυτή που διαχωρίζει ένα Τεχνητό Νευρωνικό Δίκτυο από μία πολύπλοκη γραμμική δομή και είναι αυτή που ξεχωρίζει κάθε κρυφό στρώμα. Σε περίπτωση που χρησιμοποιούσαμε μία γραμμική συνάρτηση Ενεργοποίησης το αποτέλεσμα όλων των κρυφών στρωμάτων θα μπορούσε να παρουσιαστεί και ως η ένωση τους καθώς το άθροισμα τους θα αποτελούσε άλλη μία γραμμική συνάρτηση.

Πόλωση (Bias)
Η Πόλωση αποτελεί μία Σταθερά η οποία συνεισφέρει στην σωστή ενεργοποίηση των Νευρώνων. Ο τρόπος που επιτυγχάνεται αυτή η συνεισφορά είναι με την πρόσθεση αυτής της Σταθεράς στο Γινόμενο που υπολογίζει την τιμή του κάθε Νευρώνα. (Περισσότερες πληροφορίες για τις μαθηματικές πράξεις που συμπεριλαμβάνουν την Πόλωση, παρακάτω στην διαδικασία Εμπροσθοτροφοδότησης και στο Κεφάλαιο 3.)

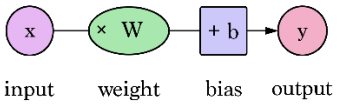
Διαδικασία Εμπροσθοτροφοδότησης (Forward Propagation)
Η διαδικασία Εμπροσθοτροφοδήτησης (Forward Propagation) αποτελεί την μία από τις δύο Πιο σημαντικές διαδικασίες ενός Νευρωνικού Δικτύου και απαρτίζεται από την μεταφορά της πληροφορίας από το ένα Στρώμα (Layer), στο επόμενο.

Τα Τεχνητά Νευρωνικά Δίκτυα τα Στρώματα (Εισόδου, Κρυφά, Εξόδου), διαθέτουν διασυνδέσεις μεταξύ τους, στις οποίες αναθέτονται βάρη. Τα βάρη αναπαριστούν την τάση που έχει ένας Νευρώνας να ενεργοποιήσει έναν άλλο.

Η διαδικασία της Εμπροσθοτροφοδότησης ξεκινάει από το Στρώμα Εισόδου (Input Layer), όπου τα δεδομένα τροφοδοτούνται στο δίκτυο. Έπειτα η είσοδος περνά από κάθε διαδοχικό κρυφό στρώμα του δικτύου, τα οποία την επεξεργάζονται και την διαμορφώνουν στους ανάλογους νευρώνες του επόμενου στρώματος.

Η διαδικασία μπορεί να αναλυθεί στα επόμενα βήματα:

1. Η είσοδος πολλαπλασιάζεται με τα βάρη που αποτελούν την σύνδεση κάθε νευρώνα μεταξύ των στρωμάτων εισόδου και πρώτου κρυφού επιπέδου χρησιμοποιώντας την ιδιότητα του εσωτερικού γινομένου (Dot Product).
2. Το αποτέλεσμα προστίθεται η πόλωση και έπειτα περνά μέσα από μία συνάρτηση ενεργοποίησης (Activation Function) η οποία εισάγει μη γραμμικότητα και διαμορφώνει την έξοδο.
3. Το αποτέλεσμα της εξόδου για κάθε νευρώνα αναπαριστά τη μορφή του πρώτου κρυφού στρώματος το οποίο χρησιμοποιείται σαν είσοδος στο επόμενο κρυφό στρώμα (αν υπάρχει) και η διαδικασία επαναλαμβάνεται μέχρι να παραχθεί το στρώμα εξόδου.



Εικόνα 5 (DeepAi.org) – Διαδικασία Εμπροσθοτροφοδότησης για δύο Στρώματα

Διαδικασία Οπισθοδιάδοσης (Back Propagation)
Ο αλγόριθμος της Οπισθοδιάδοσης, είναι ο Αλγόριθμος που χρησιμοποιείται για την εκπαίδευση των Νευρωνικών Δικτύων. Ανήκει στην κατηγορία της Εποπτευόμενης Μάθησης και χρησιμοποιείται για να ελαχιστοποιήσει το σφάλμα, όσον αφορά την προβλεπόμενη απάντηση σε σχέση με την επιθυμητή. Ο αλγόριθμος λειτουργεί, διαδίδοντας το σφάλμα από το Στρώμα Εξόδου στα προηγούμενα Στρώματα. Η διάδοση αυτή, επιτυγχάνεται χρησιμοποιώντας τον κανόνα της Αλυσίδας από τον Απειροστικό Λογισμό (Calculus) για να ενημερώσει τα Βάρη σε κάθε Στρώμα. Αυτή η διαδικασία ξεκινάει στο Στρώμα εξόδου, όπου το σφάλμα υπολογίζεται ως η διαφορά ανάμεσα στο επιλεγόμενο (από το Νευρωνικό Δίκτυο) αποτέλεσμα και το επιθυμητό. Έπειτα το σφάλμα αυτό διαδίδεται προς τα πίσω ενημερώνοντας τις τιμές που είχαν τα βάρη προηγουμένως. Η διαδικασία της Οπισθοδιάδοσης επαναλαμβάνεται για μεγάλο όγκο δεδομένων για να εκπαιδεύσει το Νευρωνικό Δίκτυο στις διάφορες εκδοχές των δεδομένων που μπορούν να εμφανιστούν. Για παράδειγμα στην οπτική αναγνώριση ψηφίων, ο αριθμός (5) γράφεται διαφορετικά από κάθε άνθρωπο το οποίο δημιουργεί την ανάγκη για το δίκτυο να πρέπει μπορεί να κρίνει διάφορες αναπαραστάσεις του αριθμού. Ο Αλγόριθμος Οπισθοδιάδοσης εκπαιδεύει το Νευρωνικό ως προς κάθε ψηφίο ξεχωριστά, το οποίο ωθεί το δίκτυο να μάθει πολύ συγκεκριμένα δεδομένα. Για την επίτευξη της βελτιστοποίησης του μοντέλου για μεγάλο όγκο δεδομένων,

χρησιμοποιείται μία Παράμετρος που ονομάζεται «Ρυθμός Μάθησης» (Learning Rate). Αυτή η σταθερά ονομάζεται και Υπέρ-παράμετρος. Η Υπέρ-παράμετρος στα Νευρωνικά Δίκτυα αποτελεί μία μεταβλητή η οποία δέχεται τιμές από τον ίδιο τον Προγραμματιστή. Ο Ρυθμός Μάθησης αποτελεί την ευαισθησία του Νευρωνικού Δικτύου ως προς την αλλαγή των Βαρών και της Πόλωσης μετά από κάθε διαδοχική εκτέλεση του Αλγόριθμου της Οπισθοδιάδοσης. Μία καλή πρακτική, είναι να δίνεται χαμηλή τιμή σε αυτή την παράμετρο καθώς ο σκοπός δεν είναι να εκπαιδευτεί το δίκτυο σε ένα μόνο παράδειγμα, αλλά σε μία πληθώρα δεδομένων, για την επίτευξη της οποίας αξιοποιείται ο Αλγόριθμος της Σταδιακής Καθίζησης (Κεφάλαιο 2.1.γ).

MSE (Mean Squared Error)

Η συνάρτηση Mean Squared Error αποτελεί μία πολύ συχνή συνάρτηση εύρεσης σφάλματος (Cost Function) λόγω της απλότητας χρήσης της.

Ο τύπος της συνάρτησης είναι ο

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2$$

Όπου A = Πραγματικό Αποτέλεσμα (Actual Output) και P = Αποτέλεσμα Πρόβλεψης (Predicted Output)

Εφόσον δεν απαιτείται να βρεθεί το συνολικό σφάλμα, αλλά το σφάλμα για κάθε ξεχωριστό κελί του πίνακα πρόβλεψης σε σχέση με τον πραγματικό πίνακα αποτελεσμάτων, η περιγραφή του κόστους για κάθε ζευγάρι μπορεί να εκφραστεί ως

$$MSE_{pair} = (A - B)^2$$

Όπου A = Πραγματικό Αποτέλεσμα (Actual Output) και P = Αποτέλεσμα Πρόβλεψης (Predicted Output)

Ο λόγος που θα προτιμάται αυτή η συνάρτηση κόστους είναι γιατί η παράγωγος της αποτελεί έναν πολύ εύκολο ως προς το υλικό του υπολογιστή υπολογισμό λόγω της εύκολης εξαγωγής της παραγώγου αυτής της συνάρτησης

Ο τύπος της παραγώγου της συνάρτησης MSE είναι για ένα ζεύγος αποτελεσμάτων είναι

$$MSE'_{pair} = 2(A - B)$$

Για απλοποίηση μπορεί να διαιρεθεί με μία σταθερά καθώς δεν θα αλλοιώσει ιδιαίτερα τα αποτελέσματα. Η βελτιστοποίηση αυτή μπορεί να απλοποιήσει την εξαγωγή του σφάλματος για κάθε ψηφίο που θα εκπαιδεύεται το Νευρωνικό Δίκτυο θα χρησιμοποιείται η παρακάτω πράξη

$$Delta_{output} = Actual\ Array - Predicted\ Array$$

Συνελικτικά Νευρωνικά Δίκτυα

Τα Συνελικτικά Νευρωνικά Δίκτυα (CNN's) είναι μία κατηγορία Νευρωνικών δικτύων που έχει γίνει κυρίαρχη σε θέματα που αφορούν την όραση του υπολογιστή. Προσελκύει το ενδιαφέρον σε διάφορους τομείς, συμπεριλαμβανομένης της ακτινολογίας και της αναγνώρισης εικόνων. Τα CNN έχουν σχεδιαστεί για να μαθαίνουν αυτόματα και προσαρμοστικά ιεραρχίες χαρακτηριστικών (Feature Extraction), χρησιμοποιώντας πολλαπλά δομικά στοιχεία, όπως Στρώματα συνέλιξης (Convolutional Layers), Στρώματα Συγκέντρωσης (Pooling Layers) και πλήρως συνδεδεμένα επίπεδα (Fully Connected Layers).

Στρώμα Συνέλιξης (Convolutional Layer):
Το Συνελικτικό Στρώμα είναι το βασικό δομικό στοιχείο ενός CNN. Εκεί Λαμβάνει χώρα η πλειοψηφία των υπολογισμών. Απαιτεί μερικά στοιχεία, τα οποία είναι τα δεδομένα εισόδου, ένα φίλτρο και ένας χάρτης χαρακτηριστικών. Με την υπόθεση πως η είσοδος είναι μία έγχρωμη εικόνα, η οποία αναπαρίσταται από έναν Τρισδιάστατο Πίνακα (3D Matrix, Μπορεί να αναφερθεί και ως μήτρα) ο οποίος αποτελείται από τα χρώματα της εικόνας (RGB Matrix). Στην πραγματικότητα είναι τρεις δισδιάστατοι πίνακες, όπου ο ένας αναπαριστά την ένταση του κόκκινου στοιχείου της εικόνας, ο δεύτερος το πράσινο στοιχείο της εικόνας και ο τρίτος το μπλε. Επίσης περιέχει έναν ανιχνευτή χαρακτηριστικών, γνωστό και ως πυρήνα ή φίλτρο (Kernel ή Filter), ο οποίος θα κινείται στα πεδία λήψης της εικόνας, ελέγχοντας αν υπάρχει το ανάλογο χαρακτηριστικό. Αυτή η διαδικασία είναι γνωστή ως συνέλιξη.

Ο Ανιχνευτής Χαρακτηριστικών (Filter ή Kernel) αποτελεί μία διάταξη βαρών, η οποία αντιπροσωπεύει μέρος της εικόνας. Ενώ μπορεί να διαφέρουν σε μέγεθος, το μέγεθος του φίλτρου είναι συνήθως ένας πίνακας 3x3. Αυτό καθορίζει επίσης το μέγεθος του Δεικτικού Πεδίου (Το πεδίο στο οποίο ο ανιχνευτής συγκρίνεται με τον ανάλογο πίνακα RGB). Στη συνέχεια, το φίλτρο εφαρμόζεται σε μία περιοχή της εικόνας και υπολογίζεται έναν γινόμενο μεταξύ αυτών των στοιχείων το οποίο στο τέλος τροφοδοτείται σε έναν πίνακα εξόδου. Στη συνέχεια, το φίλτρο μετατοπίζεται κατά ένα βήμα, επαναλαμβάνοντας την διαδικασία μέχρις ότου το φίλτρο να σαρώσει ολόκληρη την εικόνα. Η τελική έξοδος αποτελείται από έναν πίνακα που παράχθηκε από τη σειρά των γινομένων μεταξύ του φίλτρου και της εισόδου και είναι γνωστή ως χάρτης χαρακτηριστικό.

Μετά από κάθε λειτουργία συνέλιξης, ένα CNN εφαρμόζει μία μη γραμμική συνάρτηση ενεργοποίησης (Non-Linear Activation Function) στον χάρτη χαρακτηριστικών εισάγοντας μη γραμμικότητα στο μοντέλο, όπου συνήθως στα Συνελικτικά Νευρωνικά Δίκτυα εφαρμόζεται η ReLu (Rectified Linear Unit)

Ένα άλλο επίπεδο συνέλιξης μπορεί να ακολουθήσει το αρχικό επίπεδο συνέλιξης. Όταν συμβεί αυτό, η δομή του CNN μπορεί να γίνει ιεραρχική καθώς τα μεταγενέστερα στρώματα μπορούν να δουν τα εικονοστοιχεία εντός των δεκτικών πεδίων των προηγούμενων επιπέδων. Για παράδειγμα, αν υποθέσουμε πως προσπαθούμε να προσδιορίσουμε εάν μία εικόνα περιέχει ένα ποδήλατο, μπορούμε να σκεφτούμε το ποδήλατο ως άθροισμα εξαρτημάτων, το οποίο αποτελείται από σώμα, τιμόνι, τροχούς, πετάλια κλπ. Κάθε μεμονωμένο μέρους του ποδηλάτου δημιουργεί ένα μοτίβο χαμηλότερου επιπέδου στο Νευρωνικό Δίκτυο και ο συνδυασμός των μερών του αντιπροσωπεύει ένα μοτίβο υψηλότερου επιπέδου, δημιουργώντας μία ιεραρχία χαρακτηριστικών εντός του CNN.

Πολλαπλασιασμός Πινάκων Γραμμικής Άλγεβρας

Η γραμμική άλγεβρα είναι ένας σημαντικός κλάδος των μαθηματικών που ασχολείται με τη μελέτη γραμμικών μετασχηματισμών και διανυσμάτων. Μία από τις θεμελιώδεις έννοιες στη γραμμική άλγεβρα είναι ο πολλαπλασιασμός πινάκων. Ο πολλαπλασιασμός πίνακα είναι μία δυαδική πράξη που παίρνει δύο πίνακες και ως αποτέλεσμα παράγει έναν άλλο πίνακα.

Το γινόμενο δύο πινάκων ορίζεται μόνο εάν ο αριθμός των στηλών στον πρώτο πίνακα είναι ίσος με τον αριθμό των γραμμών στον δεύτερο πίνακα. Εάν ο πίνακας Α έχει διαστάσεις M x N και ο Πίνακας Β έχει διαστάσεις N x P, τότε ο πίνακας γινομένου C έχει διαστάσεις M x P. Τα στοιχεία του πίνακα γινομένων C λαμβάνονται αξιοποιώντας το εσωτερικό γινόμενο κάθε γραμμής του πίνακα Α με κάθε στήλη του πίνακα Β.

Ο τύπος για τον υπολογισμό του στοιχείου C_{ij} στον Πίνακα γινομένων C δίνεται από την παρακάτω παράσταση:

$$C_{ij} = sum_k(a_{ik} * B_{kj}),$$

Όπου A_{ik} και B_{kj} είναι στοιχεία των πινάκων Α και Β, αντίστοιχα

Ο Γραμμικός Πολλαπλασιασμός Πινάκων έχει πολυάριθμες εφαρμογές σε διάφορους τομείς, συμπεριλαμβανομένων το γραφικών στους υπολογιστές, της μηχανικής μάθησης και των επιστημονικών υπολογιστών.

Στην Μηχανική Μάθηση ο γραμμικός πολλαπλασιασμός πινάκων χρησιμοποιείται ιδιαίτερα στα Νευρωνικά Δίκτυα καθώς τον χρησιμοποιούμε για την εκτέλεση υπολογισμών σε εφαρμογές όπως η Εμπροσθοδιάδοση και η Οπισθοδιάδοση για την μετάδοση των αποτελεσμάτων μεταξύ των Νευρώνων.

Παράγωγοι Συναρτήσεων

Οι Παράγωγοι είναι μία θεμελιώδεις έννοια στον Απειροστικό Λογισμό και διαδραματίζουν κρίσιμο ρόλο στα μαθηματικά, τη φυσική, τη μηχανική, τα οικονομικά και πολλούς άλλους τομείς. Μία παράγωγος είναι ένα μέτρο αλλαγής μίας συνάρτησης ανάλογα με την είσοδο της. Μας δείχνει τον ρυθμο μεταβολής μίας συνάρτησης σε ένα συγκεικριμένο σημείο. Η παράγωγος μίας συνάρτησης f σε ένα σημείο x παριστάνεται με $f'(x)$ ή απλά $\mathrm{d}f/dx$. Ο επίσημος ορισμός της παραγώγου δίνεται από το παρακάτω όριο:

$$f'(x) = \lim_{\mathrm{n}\to 0} \frac{f(x+h) - f(x)}{h}$$

όπου h είναι ένας «μικρός» θετικός αριθμός που πλησιάζει το μηδέν.
Αυτό το όριο μας δίνει το ρυθμό μεταβολής της συνάρτησης f στο σημείο x.

Οι παράγωγοι έχουν πολυάριθμες εφαρμογές σε προβλήματα του πραγματικού κόσμου, όπως η βελτιστοποίηση, η κίνηση και τα οικονομικά και η Μηχανική Μάθηση.

Στην Μηχανική μάθηση χρησιμοποιούμε τις παραγώγους συγκεκριμένα για τον υπολογισμό της κλίσης της συνάρτησης κόστους στην οποία εφαρμόζεται η βελτιστοποίηση. Εφόσον όλο το Νευρωνικό Δίκτυο μπορεί να υλοποιηθεί ως μία συνάρτηση σύνθετη, μπορούμε να εκμεταλλευτούμε τον κανόνα της αλυσίδας που θα εξηγηθεί παρακάτω την μεταφορά της πληροφορίας που παρατηρούμε στην συνάρτηση κόστους σε προηγούμενα σημεία στο Νευρωνικό Δίκτυο.

Κανόνας της Αλυσίδας

Στον Απειροστικό Λογισμό, Ο κανόνας της αλυσίδας είναι ένας τύπος για τον υπολογισμό της παραγώγου της σύνθεσης δύο η περισσότερων συναρτήσεων (έστω $f(x)$ και $g(x)$). Δηλώνει πως αν η $f(x)$ και η $g(x)$ είναι διαφοροποιήσιμες συναρτήσεις, τότε η παράγωγος της σύνθετης συνάρτησης f(g(x)) δίνεται από το γινόμενο της παραγώγου της $f(x)$ αξιολογώντας σαν «x» την Συνάρτηση $g(x)$ με την παράγωγο της Συνάρτησης $g(x)$.

Ο Κανόνας της αλυσίδας μπορεί μαθηματικά να εκφραστεί ως:

$(d/dx)f(g(x)) = f'(g(x))(d/dx)g(x)$

Ο κανόνας της αλυσίδας μπορεί επίσης να εφαρμοστεί στη σύνθεση περισσότερων από δύο συναρτήσεων, εφαρμόζοντας τον κανόνα επανειλημμένα.

Ας υποθέσουμε πως έχουμε μία συνάρτηση $f(x) = x^3$ και μία άλλη συνάρτηση $g(x) = 2x + 1$. Θέλουμε να βρούμε την παράγωγο της σύνθετης συνάρτησης $f\big(g(x)\big) = (2x + 1)^3$. Χρησιμοποιώντας τον κανόνα της αλυσίδας έχουμε:

$$\frac{d}{dx} f\big(g(x)\big) = f'\big(g(x)\big) * \frac{d}{dx} g(x)$$

Η παράγωγος της $f(x) = x^3$ είναι $f'(x) = 3x^2$, και η παράγωγος της $g(x) = 2x + 1$ είναι $g'(x) = 2$. Άρα μπορούμε να τις αντικαταστήσουμε με τον κανόνα της αλυσίδας ως:

$\frac{d}{dx} f\big(g(x)\big) = f'\big(g(x)\big) \frac{d}{dx} g(x) \rightarrow 3(2x + 1)^2 2 = 6x^2 + 6x + 2$

2.2 Γλώσσες Προγραμματισμού

2.2.α Οι Συγκρίσιμες Γλώσσες Προγραμματισμού

Python

Η Python είναι μία ισχυρή, υψηλού επιπέδου γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως σε μία ποικιλία εφαρμογών, συμπεριλαμβανομένης της Ανάπτυξης Ιστοσελίδων (Web Development), της Προσομοίωσης Επιστημονικών Υπολογισμών (Scientific Computing Simulations), της Ανάλυσης Δεδομένων (Data Analysis), Τεχνητής Νοημοσύνης κ.α.

Η Python είναι μία γλώσσα που χρησιμοποιεί Διερμηνέα, (Περισσότερες πληροφορίες στο κεφάλαιο 2.2) που σημαίνει πως ο κώδικας της εκτελείται γραμμή-γραμμή σε αντίθεση με τις μεταγλωττιζόμενες γλώσσες που μεταφράζουν τον κώδικά τους σε κώδικα μηχανής πριν την εκτέλεση του. Η διερμηνεία του κώδικα σε πραγματικό χρόνο παρέχει πολλά πλεονεκτήματα, όπως ευκολία ανάπτυξης, ταχύτερη δοκιμή (Debugging) και μεγαλύτερη ευκολία στον εντοπισμό σφαλμάτων. Η δυναμικά τυποποιημένη φύση της Python απλοποιεί περαιτέρω τη διαδικασία κωδικοποίησης επιτρέποντας στους προγραμματιστές να επικεντρωθούν στην σύναξη και της δοκιμή κώδικα, χωρίς να χρειάζεται να ανησυχούν για τη δήλωση τύπων μεταβλητών εκ των προτέρων.

Ένας από τους λόγους που η Python είναι τόσο δημοφιλής, είναι η απλή και ευανάγνωστη σύνταξη, πράγμα το οποίο καθιστά εύκολο για προγραμματιστές όλων των επιπέδων και δεξιοτήτων να κατανοούν και να γράφουν κώδικα. Επίσης, η Python υποστηρίζει πολλαπλά προγραμματιστικά μοντέλα, όπως ο αντικειμενοστραφής προγραμματισμός (Object Oriented Programming) ή και πιο απλό, Συναρτησιακό προγραμματισμό (Functional Programming) που επιτρέπει στους προγραμματιστές να επιλέξουν την προσέγγιση που ταιριάζει καλύτερα στις ανάγκες τους.

Η εκτεταμένη υποστήριξη διαφόρων βιβλιοθηκών της Python, είναι ένα ακόμα σημαντικό Πλεονέκτημα. Η τυπική βιβλιοθήκη της Python (Python Standard Library) περιέχει μία μεγάλη ποικιλία λειτουργικών μονάδων (Modules) και πακέτων (Packages) που παρέχουν λειτουργικότητα για εργασίες όπως σύνδεση σε διακομιστές ιστού (Web Servers), ανάγνωση και εγγραφή αρχείων και εργασίας πάνω σε διάφορες δομές δεδομένων (Data Structures). Επιπλέον, υπάρχουν και πολλές βιβλιοθήκες τρίτων που είναι διαθέσιμες για διαδικασίες πως η οπτικοποίηση δεδομένων, μηχανική Μάθηση και επεξεργασία φυσικής γλώσσας.

Στον τομέα της ανάπτυξης ιστοσελίδων, η Python γίνεται όλο και πιο δημοφιλής τα τελευταία χρόνια. Στα πλαίσια Web Python όπως τα Django, Flash και Pyramid διευκολύνουν την δημιουργία και την συντήρηση εφαρμογών Ιστού. Αυτά τα πλαίσια παρέχουν ένα ευρύ φάσμα δυνατοτήτων, συμπεριλαμβανομένης της ενσωματωμένης υποστήριξης για έλεγχο ταυτότητας χρήστη, διαχείριση βάσης δεδομένων και πολλά άλλα.

Η Python χρησιμοποιείται επίσης ευρέως στον επιστημονικό λογισμό και την ανάλυση δεδομένων. Οι βιβλιοθήκες NumPy και SciPy παρέχουν ισχυρά εργαλεία για την εκτέλεση μαθηματικών πράξεων και την ανάλυση δεδομένων. Είναι επίσης μία δημοφιλής επιλογή για μηχανική μάθηση και Τεχνητή Νοημοσύνη. Οι βιβλιοθήκες της Python όπως το TesnorFlow, το PyTorch και το scikit-learn παρέχουν εύχρηστες διεπαφές για τη δημιουργία και την εκπαίδευση μοντέλων μηχανικής μάθησης.

C++

Η C++ είναι μία ισχυρή γλώσσα προγραμματισμού υψηλής απόδοσης που χρησιμοποιείται ευρέως σε μία ποικιλία εφαρμογών, συμπεριλαμβανομένου του Λογισμικού Συστήματος (System Software), της Ανάπτυξης παιχνιδιών και των επιστημονικών υπολογισμών.

Ένα από τα κύρια πλεονεκτήματα της C++, είναι η απόδοσης της. Η C++ είναι μία μεταγλωττισμένη γλώσσα, που σημαίνει πως ο κώδικας που γράφεται σε C++, μεταφράζεται σε κώδικα μηχανής πριν εκτελεστεί. Αυτό επιτρέπει ταχύτερους χρόνους εκτέλεσης σε σύγκριση με γλώσσες ερμηνείας, όπως η Python και η JavaScript. Επιπλέον, η C++ παρέχει λεπτομερή έλεγχο στους πόρους του συστήματος, ο οποίος μπορεί να είναι ιδιαίτερα χρήσιμος για εφαρμογές που απαιτούν υψηλό επίπεδο απόδοσης, όπως η ανάπτυξη παιχνιδιών και οι Επιστημονικές Εφαρμογές.

Η C++ είναι επίσης μία ευέλικτη γλώσσα, με υποστήριξη τόσο για αντικειμενοστραφή, όσο και για διαδικαστικά παραδείγματα προγραμματισμού. Αυτό επιτρέπει στους προγραμματιστές να επιλέξουν την προσέγγιση που ταιριάζει καλύτερα στις ανάγκες τους. Η τυπική βιβλιοθήκη προτύπων (STL) της C++ παρέχει μία μεγάλη ποικιλία δομών δεδομένων και αλγορίθμων που μπορούν να χρησιμοποιηθούν για την εκτέλεση κοινών εργασιών, όπως η ταξινόμηση και η αναζήτηση, καθιστώντας την ένα ισχυρό εργαλείο για την επίλυση προβλημάτων. Στον τομέα της ανάπτυξης παιχνιδιών, η C++ είναι μία δημοφιλής επιλογή λόγω της απόδοσής της και της πρόσβασης χαμηλού επιπέδου στους πόρους του συστήματος. Πολλές δημοφιλείς μηχανές παιχνιδιών, όπως για παράδειγμα η Unreal Engine και η Unity, είναι γραμμένες σε C++ (Η μηχανή Unity έχει και κάποια Κομμάτια σε C#), το οποίο επιτρέπει στους προγραμματιστές να δημιουργούν παιχνίδια υψηλής απόδοσης με προηγμένα γραφικά και υλοποιήσεις προσομοιώσεων φυσικής.

Η C++ χρησιμοποιείται επίσης ευρέως στους Επιστημονικούς υπολογιστές, ιδιαίτερα για προσομοιώσεις υψηλής απόδοσης και ανάλυσης δεδομένων. Η βιβλιοθήκη C++ AMP (Accelerated Massive Parallelism) παρέχει μία εύχρηστη διεπαφή για την χρήση των δυνατοτήτων παράλληλου υπολογισμού των σύγχρονων καρτών γραφικών (GPU's), καθιστώντας την ένα ισχυρό εργαλείο για την επίλυση πολύπλοκων προβλημάτων.

Ωστόσο, η C++ μπορεί να είναι προκλητική για αρχάριους, καθώς έχει μία πιο απότομη καμπύλη Μάθησης σε σύγκριση με γλώσσες όπως η Python ή η JavaScript. Η γλώσσα έχει πολλά χαρακτηριστικά και πολυπλοκότητες που μπορεί να πάρει χρόνο για να κατακτηθούν. Έχει επίσης έλλειψη ενσωματωμένης διαχείρισης μνήμης, η οποία μπορεί να οδηγήσει σε σφάλματα εάν δεν αντιμετωπιστεί σωστά.

2.2 Διερμηνέας

Ο Διερμηνέας είναι ένα πρόγραμμα που εκτελεί εντολές γραμμένες σε μία γλώσσα προγραμματισμού, μεταφράζοντας τις σε κώδικα μηχανής που μπορεί να εκτελεστεί από έναν υπολογιστή. Οι Διερμηνείς αποτελούν σημαντικό μέρος της διαδικασίας ανάπτυξης λογισμικού, καθώς επιτρέπουν στους προγραμματιστές να γράφουν κώδικα σε μία γλώσσα προγραμματισμού υψηλού επιπέδου και στη συνέχεια να τον εκτελούν σε υπολογιστή χωρίς να απαιτείται ξεχωριστό βήμα μεταγλώττισης.

Ένα από τα κύρια πλεονεκτήματα της χρήσης διερμηνέα είναι πως επιτρέπει ταχύτερους χρόνους ανάπτυξης. Επειδή ένας διερμηνέας μπορεί να εκτελέσει κώδικα αμέσως μόλις γραφτεί, οι προγραμματιστές μπορούν να δοκιμάσουν και να διορθώσουν τον κώδικά τους πιο εύκολα και γρήγορα. Αυτό το καθιστά δημοφιλή επιλογή για γλώσσες δέσμης ενεργειών και για γρήγορα δημιουργία πρωτοτύπων, καθώς επιτρέπει στους προγραμματιστές να δοκιμάσουν γρήγορα και να επαναλάβουν τις ιδέες τους.

Οι Διερμηνείς έχουν επίσης το πλεονέκτημα ότι είναι ανεξάρτητοι από πλατφόρμα, πράγμα που σημαίνει πως ο ίδιος κώδικας μπορεί να εκτελεστεί σε διαφορετικούς τύπους υπολογιστών χωρίς τροποποίηση. Αυτό έρχεται σε αντίθεση με τις μεταγλωττισμένες γλώσσες, οι οποίες μεταφράζονται σε κώδικα μηχανής που είναι συγκεκριμένος για την πλατφόρμα-στόχο.

Οι Διερμηνείς μπορούν επίσης να παρέχουν πιο λεπτομερή μηνύματα σφάλματος, καθώς μπορούν να εντοπίσουν προβλήματα στον πηγαίο κώδικα καθώς τον εκτελούν. Αυτό επιτρέπει στους προγραμματιστές να εντοπίζουν και να διορθώνουν γρήγορα σφάλματα, σε αντίθεση με τις μεταγλωττισμένες γλώσσες, όπου τα σφάλματα δεν εντοπίζονται μέχρι την διαδικασία μεταγλώττισης.

Ωστόσο, υπάρχουν και ορισμένα μειονεκτήματα στη χρήση διερμηνέα. Ένα από τα κύρια μειονεκτήματα είναι πως ο Ερμηνευόμενος κώδικας, εκτελείται γενικά πιο αργά από τον μεταγλωττισμένο κώδικα, καθώς ο Διερμηνέας πρέπει να μεταφράσει τον κώδικα σε κώδικα μηχανής αμέσως. Αυτό μπορεί να καταστήσει τις γλώσσες που ερμηνεύονται λιγότερο κατάλληλες για εφαρμογές κρίσιμες για την απόδοση, όπως βιντεοπαιχνίδια ή διάφορες επιστημονικές προσομοιώσεις.

Ένα άλλο μειονέκτημα είναι πως οι Διερμηνείς μπορεί να είναι λιγότερο ασφαλείς, καθώς ο πηγαίος κώδικας είναι ορατός σε οποιονδήποτε έχει πρόσβαση στον Διερμηνέα. Αυτό μπορεί να διευκολύνει τους κακόβουλους φορείς να αναθεωρήσουν τον κώδικα και να εντοπίσουν ευάλωτα σημεία.

2.3 Μεταγλωττιστής

Ο μεταγλωττιστής είναι ένα πρόγραμμα λογισμικού που παίρνει τον πηγαίο κώδικα που έχει γράψει ένας προγραμματιστής και τον μετατρέπει σε κώδικα μηχανής, που είναι το σύνολο εντολών που μπορεί να εκτελεστεί από την κεντρική μονάδα επεξεργασίας (CPU) ενός υπολογιστή. Η μεταγλώττιση είναι ένα ουσιαστικό μέρος της ανάπτυξης λογισμικού, καθώς επιτρέπει στους προγραμματιστές να γράφουν κώδικα σε μία γλώσσα προγραμματισμού υψηλού επιπέδου και στη συνέχεια να τον μεταφράζουν σε κώδικα μηχανής.

Οι μεταγλωττιστές χρησιμοποιούνται σε ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένης της ανάπτυξης λειτουργικών συστημάτων, παιχνιδιών και εφαρμογών επιφάνειας εργασίας. Διαδραματίζουν κρίσιμο ρόλο στη διαδικασία ανάπτυξης λογισμικού, καθώς παρέχουν έναν τρόπο μετάφρασης κώδικα γραμμένου σε γλώσσες προγραμματισμού υψηλού επιπέδου σε κώδικα που μπορεί να εκτελεστεί από υπολογιστή. Αυτό επιτρέπει στους προγραμματιστές να γράφουν κώδικα με τρόπο που είναι εύκολο να διαβαστεί, να διατηρηθεί και να κατανοηθεί, ενώ παράλληλα μπορούν να επωφεληθούν από την ταχύτητα και την απόδοση του κώδικα μηχανής.

Υπάρχουν δύο κύριοι τύποι μεταγλωττιστών. Οι Διασταυρούμενοι Μεταγλωττιστές (Cross-Compilers) και οι Εγγενείς Μεταγλωττιστές (Native Compilers). Ένας Cross-Compiler χρησιμοποιείται για την μεταγλώττιση κώδικα σε διαφορετική πλατφόρμα ή αρχιτεκτονική από αυτή στην οποία εκτελείται ο Μεταγλωττιστής. Για παράδειγμα, ένας προγραμματιστής μπορεί να χρησιμοποιήσει έναν Cross-Compiler για να μεταγλωττίσει κώδικα για ένα σύστημα που βασίζεται σε ARM σε ένα σύστημα που βασίζεται σε αρχιτεκτονική Intel x86. Ένας εγγενής μεταγλωττιστής, από την άλλη πλευρά, χρησιμοποιείται για τη μεταγλώττιση κώδικα για την ίδια πλατφόρμα με αυτή στην οποία εκτελείται ο μεταγλωττιστής.

Η διαδικασία μεταγλώττισης κώδικα περιλαμβάνει διάφορα στάδια, συμπεριλαμβανομένης της λεξιλογικής ανάλυσης, της ανάλυσης κώδικα (Parsing), της σημασιολογικής ανάλυσης, της δημιουργίας κώδικα και της βελτιστοποίησης. Κατά τη λεξιλογική ανάλυση, ο Μεταγλωττιστής αναλύει τον πηγαίο κώδικα σε μεμονωμένα εμβλήματα (Tokens), όπως λέξεις-κλειδιά, μεταβλητές και τελεστές. Κατά την ανάλυση, ο Μεταγλωττιστής χρησιμοποιεί αυτά τα Tokens για να δημιουργήσει ένα αφηρημένο δέντρο σύνταξης, το οποίο αντιπροσωπεύει τη δομή του κώδικα. Στη συνέχεια, η σημασιολογική ανάλυση ελέγχει τον κώδικα για σημασιολογικά σφάλματα, όπως αναντιστοιχίες τύπου ή απροσδιόριστες μεταβλητές. Η δημιουργία κώδικα παίρνει το αφηρημένο δέντρο σύνταξης και δημιουργεί κώδικα μηχανής, ενώ η βελτιστοποίηση βελτιώνει την απόδοση του παραγόμενου κώδικα καθιστώντας τον ταχύτερο και πιο αποτελεσματικό.

ΚΕΦΑΛΑΙΟ 3 Προγραμματιστική Υλοποίηση

3.1 Python

Θεωρητικό Υπόβαθρο και Επεκτάσεις 3.1.α

Σε αυτό το κεφάλαιο, θα εμβαθύνουμε στην χρήση εργαλείων και επεκτάσεων που θα βοηθήσουν στην ανάπτυξη του Νευρωνικού Δικτύου στην γλώσσα Python. Για να μπορέσουμε να διασφαλίσουμε όμοια εκτέλεση με την συγκρίσιμη γλώσσα (C++), θα αναπτύξουμε το Νευρωνικό Δίκτυο από την αρχή.

NUMPY

Η Numpy είναι μία ισχυρή και ευρέως χρησιμοποιούμενη βιβλιοθήκη στη γλώσσα προγραμματισμού Python για αριθμητικούς υπολογισμούς. Παρέχει υποστήριξη για πίνακες (Matrixes), οι οποίοι είναι πολυδιάστατες δομές που χρησιμοποιούνται για μαθηματικές και στατιστικές πράξεις. Η βιβλιοθήκη Numpy χρησιμοποιείται ευρέως για επιστημονικές και μαθηματικές εφαρμογές, ειδικά σε τομείς όπως η Μηχανική Μάθηση (Machine Learning), η ανάλυση δεδομένων και η όραση του Υπολογιστή (Computer Vision).

Οι πίνακες Numpy είναι αποτελεσματικοί και γρήγοροι καθώς υλοποιούνται σε γλώσσες προγραμματισμού χαμηλού επιπέδου όπως η C ή η Fortran. Είναι επίσης αποδοτική στη μνήμη, επιτρέποντας στους χρήστες να αποθηκεύουν μεγάλες ποσότητες δεδομένων σε μία ενιαία δομή δεδομένων. Επιπλέον, οι πίνακες Numpy είναι επίσης ομοιογενείς, πράγμα που σημαίνει πως όλα τα στοιχεία σε έναν πίνακα πρέπει να είναι του ίδιου τύπου δεδομένων, επιτρέποντας αποτελεσματικούς υπολογισμούς.

Ένα από τα βασικά χαρακτηριστικά του της βιβλιοθήκης Numpy, είναι η υποστήριξή της για διανύσματα και λειτουργίες Πινάκων της Γραμμικής Άλγεβρας (Matrix Operations). Αυτό επιτρέπει τον γρήγορο και εύκολο υπολογισμό μαθηματικών πράξεων όπως το Εσωτερικό Γινόμενο, ο πολλαπλασιασμός πινάκων και ο υπολογισμός του ανάστροφου πίνακα. Η βιβλιοθήκη Numpy περιέχει ένα ευρύ φάσμα λειτουργιών για εργασία με πίνακες, όπως η μέση τιμή, τυπική απόκλιση και άθροισμα.

Μια άλλη σημαντική πτυχή της βιβλιοθήκης Numpy είναι η υποστήριξη για μετάδοση (Broadcasting). Η Μετάδοση είναι μία τεχνική που χρησιμοποιείται για την εκτέλεση λειτουργιών σε πίνακες που έχουν διαφορετικές διαστάσεις. Επιτρέπει στη βιβλιοθήκη να εκτελεί αυτές τις λειτουργίες σε πίνακες που έχουν διαφορετικές διαστάσεις χωρίς να χρειάζεται να κάνει την ευθυγράμμιση αυτών των πινάκων χειροκίνητα. Η διαδικασία αυτή είναι ιδιαίτερα χρήσιμη για εργασίες όπως η επεξεργασία εικόνας ή η ανάλυση δεδομένων όπου οι πίνακες έχουν συχνά διαφορετικές διαστάσεις.

Εκτός από τα βασικά Χαρακτηριστικά της, η Βιβλιοθήκη Numpy παρέχει επίσης υποστήριξη για εγγραφή και ανάγνωση Πινάκων από τον Δίσκο. Αυτό είναι πολύ χρήσιμο για την Διαχείριση Μεγάλων Συνόλων Δεδομένων (Big Data) τα οποία συνήθως δεν μπορούν να χωρέσουν στην Κύρια μνήμη του Υπολογιστή (Ram). Η βιβλιοθήκη παρέχει επίσης μία διεπαφή (Interface) για ενσωμάτωση με άλλες βιβλιοθήκες,, όπως η SciPy και η Matplotlib, οι οποίες χρησιμοποιούνται συνήθως για επιστημονικούς υπολογισμούς και οπτικοποίηση δεδομένων, αντίστοιχα.

PIL

Η Python Imaging Library (PIL) είναι μία ισχυρή βιβλιοθήκη στην Python για επεξεργασία και χειρισμό εικόνων. Αναπτύχθηκε και συντηρήθηκε από τον Fredrik Lundh και δεν αναπτύσσεται πλέον ενεργά από το 2011. Παρόλα αυτά, η βιβλιοθήκη εξακολουθεί να χρησιμοποιείται ευρέως στην Κοινότητα της Python και είναι συμβατή με πρόσφατες εκδόσεις της Python, καθιστώντας την μία δημοφιλή επιλογή για Διαδικασίες επεξεργασίας εικόνας.

Η βιβλιοθήκη PIL Παρέχει ένα ευρύ φάσμα δυνατοτήτων επεξεργασίας εικόνας, συμπεριλαμβανομένης της ανάγνωσης και εγγραφής αρχείων εικόνας σε διαφορετικές μορφές, όπως JPEG, PNG, BMP κ.α. Όσον αφορά τις δυνατότητες επεξεργασίας εικόνας, η PIL παρέχει δυνατότητες όπως η περικοπή, αλλαγή μεγέθους εικόνων, περιστροφή και ανατροπή εικόνων, μετατροπή εικόνων σε διαφορετικές λειτουργίες χρώματος και εφαρμογή φίλτρων εικόνας και εφέ. Η Βιβλιοθήκη παρέχει επίσης υποστήριξη για χειρισμό εικόνας, όπως διόρθωση χρώματος, κατώφλι (Thresholding) και μετατροπή εικόνων σε κλίμακα του γκρι.

Η βιβλιοθήκη PIL παρέχει επίσης ένα ολοκληρωμένο σύνολο λειτουργιών επεξεργασίας εικόνας, καθιστώντας δυνατή την εκτέλεση προηγμένων διαδικασιών χειρισμού εικόνας. Για παράδειγμα, η βιβλιοθήκη παρέχει λειτουργίες για την δημιουργία μικρογραφιών, τη δημιουργία ιστογραμμάτων και την εφαρμογή φίλτρων σε εικόνας. Παρέχει επίσης υποστήριξη για εργασία με πολλές εικόνας, καθιστώντας δυνατή την δημιουργία κινούμενων εικόνων, Μοντάζ και Κολάζ.

Συμπερασματικά, η Python Imaging Library (PIL) είναι μία ισχυρή και ευέλικτη βιβλιοθήκη για επεξεργασία εικόνας σε Python. Παρόλο που δεν αναπτύσσεται πλέον ενεργά, παραμένει μία δημοφιλής επιλογή για Λειτουργίες επεξεργασίας εικόνας λόγω της ευκολίας χρήσης και του ολοκληρωμένου συνόλου λειτουργιών επεξεργασίας εικόνας.

MATH

Η Βιβλιοθήκη Math είναι μία ισχυρή βιβλιοθήκη στην Python και αποτελεί έναν από τους λόγους που η δημοτικότητα της Python είναι υψηλή στον τομέα της ανάλυσης δεδομένων και της Μηχανικής Μάθησης. Αυτή η βιβλιοθήκη παρέχει πληθώρα μαθηματικών συναρτήσεων και εργαλείων που διευκολύνουν τους προγραμματιστές να εκτελούν σύνθετους μαθηματικούς υπολογισμούς, να οπτικοποιούν δεδομένα και να εκτελούν άλλες μαθηματικές διεργασίες.

Η βιβλιοθήκη Μαθηματικών στην Python Είναι γνωστή ως βιβλιοθήκη Μαθηματικών και η Εισαγωγή (Import) της στον Κώδικα είναι τόσο απλή όσο η εισαγωγή της δήλωσης «import math». Παρέχει ένα ευρύ φάσμα μαθηματικών συναρτήσεων και σταθερών, συμπεριλαμβανομένων τριγωνομετρικών συναρτήσεων, λογαριθμικών συναρτήσεων, τετραγωνικών ριζών και άλλων. Παρέχει επίσης Λειτουργίες για εργασία με μιγαδικούς αριθμούς, εκτέλεση πράξεων μεταξύ Πινάκων (Matrixes), και άλλες προηγμένες μαθηματικές διεργασίες.

Οι συναρτήσεις στη βιβλιοθήκη μαθηματικών έχουν σχεδιαστεί για να είναι εύχρηστες και κατανοητές και καλά τεκμηριωμένες με σαφείς και συνοπτικές επεξηγήσεις. Αυτό διευκολύνει τους προγραμματιστές να βρουν γρήγορα τις λειτουργίες που χρειάζεται και να αρχίσουν να τις χρησιμοποιούν στον κώδικά τους. Η βιβλιοθήκη παρέχει επίσης μία σειρά από λειτουργείες ευκολίας που διευκολύνουν την εκτέλεση κοινών μαθηματικών πράξεων, όπως στρογγυλοποίηση αριθμών, μετατροπή μοιρών σε ακτίνια και άλλα.

Ένα από τα πλεονεκτήματα της Χρήσης της Βιβλιοθήκης μαθηματικών στην Python είναι πως διευκολύνει την εκτέλεση πολύπλοκων μαθηματικών υπολογισμών ακόμα και αν ο χρήστης δεν είναι μαθηματικός. Με την διαισθητική σύνταξη και τις καλά σχεδιασμένες λειτουργίες της βιβλιοθήκης, γίνεται εύκολο να εκτελεστούν γρήγορα και εύκολα μαθηματικές πράξεις και βελτιώνεται η διαδικασία ανάλυσης δεδομένων χωρίς την ανάγκη γνώσης υλοποίησης αυτών των μαθηματικών πράξεων ή την ανάγκη χρήσης εξωτερικών εργαλείων καθώς είναι ήδη υλοποιημένες από την βιβλιοθήκη.

TIMEIT

Στον τομέα της ανάπτυξης λογισμικού, η μέτρηση της απόδοσης του κώδικα αποτελεί λειτουργία ζωτικής σημασία για τη διασφάλιση της αποδοτικής και αποτελεσματικής εκτέλεσης του. Η Python παρέχει ένα βολικό εργαλείο για το σκοπό αυτό – τη βιβλιοθήκη timeit. Η βιβλιοθήκη timeit είναι μία λειτουργική μονάδα (Module) που παρέχει έναν απλό τρόπο μέτρησης του χρόνου εκτέλεσης κομματιών κώδικα Python.

Η βιβλιοθήκη timeit μετρά τον χρόνο που απαιτείται για την εκτέλεση μίας μεμονωμένης δήλωσης κώδικα (statement) ή ενός μπλοκ κώδικα, το οποίο μπορεί να είναι χρήσιμο για τον προσδιορισμό της απόδοσης αλγορίθμων, διαδικασιών επεξεργασίας δεδομένων ή άλλων κρίσιμων για το χρόνο λειτουργιών. Έχει σχεδιαστεί για να είναι απλό στη χρήση και παρέχει δύο κύριες λειτουργίες, την «timeit.timeit» και την «timeit.repeat», για την μέτρηση του χρόνου εκτέλεσης του κώδικα.

Η συνάρτηση «timeit.timeit» εκτελεί μία μεμονωμένη πρόταση πολλές φορές και επιστρέφει το χρόνο που χρειάστηκε για την εκτέλεση της δήλωσης. Αυτή η συνάρτηση είναι χρήσιμη για τη μέτρηση της απόδοσης μεμονωμένων γραμμών κώδικα, συναρτήσεων ή ολόκληρων Λειτουργειών Κελύφους (Script). Η «timeit.repeat», από την άλλη πλευρά, εκτελεί πολλαπλές επαναλήψεις μίας δήλωσης και επιστρέφει μία λίστα με τους χρόνους εκτέλεσης, επιτρέποντας μία πιο ακριβή μέτρηση της απόδοσης λαμβάνοντας υπόψη τις διακυμάνσεις στην απόδοση από εκτέλεση σε εκτέλεση.

Η βιβλιοθήκη timeit είναι επίσης ευέλικτη ως προς το τι μπορεί να μετρήσει. Μπορεί να μετρήσει την απόδοση μεμονωμένων γραμμών κώδικα, ολόκληρων Λειτουργιών Κελύφους (Script), ή οτιδήποτε άλλο ενδιάμεσο. Είναι ένα ουσιαστικό εργαλείο για προγραμματιστές που ενδιαφέρονται για την βελτιστοποίηση της απόδοσης του κώδικά τους και την αποτελεσματικότερη εκτέλεση του, καθώς παρέχει πολύτιμες πληροφορίες που μπορούν να χρησιμοποιηθούν για τον εντοπισμό σημείων συμφόρησης και για να παρέχουν βελτιώσεις.

Υλοποίηση στην Γλώσσα Python 3.1.β

Ο κώδικας υλοποιεί ένα Τεχνητό Νευρωνικό Δίκτυο για αναγνώριση εικόνας. Το ένα κρυφό στρώμα, το στρώμα εισόδου και το στρώμα εξόδου. Το Νευρωνικό Δίκτυο εκπαιδεύεται σε ένα σύνολο δεδομένων Τεσσάρων Χιλιάδων εικόνων και έπειτα αξιολογείται σε Χίλιες εικόνες. Τα δεδομένα εισόδου είναι εικόνες χειρόγραφων ψηφίων και η και η έξοδος είναι το ψηφίο που αναγνωρίζεται από το Νευρωνικό δίκτυο.

Η διαδικασία διαχείρισης της εισόδου αποτελείται από το κομμάτι δημιουργίας εισόδου με τη συνάρτηση create_weights ή τη διαδικασία που φορτώνονται ήδη υπάρχοντα βάρη με τη συνάρτηση load_weights. Εφόσον το Νευρωνικό μας Δίκτυο αποτελείται από τρία στρώματα (Μαζί με το στρώμα εισόδου και εξόδου), θα χρειαστούν δύο πίνακες βαρών (Ένας πίνακας βαρών για το ζευγάρι Στρώμα Εισόδου – Κρυφό Στρώμα και ένας ακόμα πίνακας βαρών για το ζευγάρι Κρυφό Στρώμα – Στρώμα Εξόδου) Αντίστοιχα θα χρειαστούν και δύο πίνακες πολώσεων (Bias Matrix). Η αρχικοποίηση των τιμών για κάθε πίνακα βαρών υλοποιείται με την αξιοποίηση της συνάρτησης δημιουργίας τυχαίων τιμών αριθμών της Python np.random.rand της οποίας οι τυχαίοι αριθμοί αποτελούν τυπική κανονική κατανομή με μέσο όρο μηδέν και τυπική απόκλιση ένα. Μετά την ολοκλήρωση της εκπαίδευσης του Νευρωνικού Δικτύου, τα υπολογίσημα βάρη μπορούν να αποθηκευτούν με την χρήση της συνάρτησης save_weights η οποία αποθηκεύει τις τιμές των βαρών και των πολώσεων του Νευρωνικού Δικτύου σε αντίστοιχα αρχεία μορφής κειμένουν (.txt).

```python
def create_weights():
    """Completed"""

    input_hidden_weights = np.random.randn(16, 784)
    input_hidden_bias = np.random.randn(16, 1)
    hidden_output_weights = np.random.randn(10, 16)
    hidden_output_bias = np.random.randn(10, 1)

    return input_hidden_weights, input_hidden_bias, hidden_output_weights, hidden_output_bias


def load_weights():
    """Completed"""

    input_hidden_weights = np.loadtxt(input_hidden_w_filename, dtype=float)

    input_hidden_bias = np.loadtxt(input_hidden_b_filename, dtype=float)

    hidden_output_weights = np.loadtxt(hidden_output_w_filename, dtype=float)

    hidden_output_bias = np.loadtxt(hidden_output_b_filename, dtype=float)

    return np.asmatrix(input_hidden_weights), np.asmatrix(input_hidden_bias).T,\
        np.asmatrix(hidden_output_weights), np.asmatrix(hidden_output_bias).T


def save_weights(input_hidden_w, input_hidden_b, hidden_out_w, hidden_out_b):
    """Completed"""

    np.savetxt(input_hidden_w_filename, input_hidden_w, fmt='%f')

    np.savetxt(input_hidden_b_filename, input_hidden_b, fmt='%f')

    np.savetxt(hidden_output_w_filename, hidden_out_w, fmt='%f')

    np.savetxt(hidden_output_b_filename, hidden_out_b, fmt='%f')
```

Εικόνα (6) – Κώδικας Python – Αρχικοποίηση και Αποθήκευση Δεδομένων

Το Τεχνητό Νευρωνικό Δίκτυο αρχικά τροφοδοτείται με μία εικόνα εισόδου, η οποία αρχικά αναπαρίσταται ως ένας δυσδιάστατος πίνακας Numpy Επτακοσίων ογδόντα τεσσάρων (784) στοιχείων (Το μέγεθος της εικόνας είναι 28x28). Η εικόνα μετατρέπεται από την αρχική της αναπαράσταση (.png) σε Numpy Matrix χρησιμοποιώντας τη συνάρτηση image_to_numpy. Η συνάρτηση διαβάζει μία εικόνα χρησιμοποιώντας τη βιβλιοθήκη PIL (Αναφορά στην βιβλιοθήκη PIL παραπάνω) και στη συνέχεια, μετατρέπει την εικόνα σε έναν μονοδιάστατο πίνακα (1D Numpy Array) του οποίου τα κελιά αντιπροσωπεύουν τις τιμές των Pixels της εικόνας. Στη συνέχεια, ο πίνακας διαιρείται με το 255 για να συρρικνωθούν οι τιμές των εικονοστοιχείων μεταξύ 0 και 1, το οποίο είναι ένα κοινό βήμα προ-επεξεργασίας στις εργασίες αναγνώρισης εικόνας. Το τελικό αποτέλεσμα αυτής της συνάρτησης είναι μία αναπαράσταση Πίνακα στήλη της εικόνας.

```python
def image_to_numpy_flattened(filename):
    """Completed"""

    im = Image.open(filename)
    array = np.array(im.getdata()).flatten().astype(float)

    for index, value in enumerate(array):
        # shrink values between 0 and 1
        array[index] /= 255.0

    im.close()

    # We need a Column Matrix
    return np.asmatrix(array).T
```

Εικόνα (7) – Κώδικας Python – Διαχείριση Εισόδου (1)

Η διαδικασία Εμπροσθοτροφοδότησης ξεκινάει περνώντας τον πίνακα στήλη (είσοδο) μέσω του Νευρωνικού Δικτύου. Το πρώτο βήμα στη Εμπροσθοτροφοδότηση, είναι ο υπολογισμός της εξόδου του πρώτου κρυφού στρώματος, το οποίο γίνεται πολλαπλασιάζοντας τον πίνακα εισόδου με τον πίνακα βάρους που συνδέει το στρώμα εισόδου με το κρυφό στρώμα και προσθέτοντας το κρυφό στρώμα πόλωσης. Στη συνέχεια, το αποτέλεσμα περνά μέσα από τη συνάρτηση ενεργοποίησης. Η συνάρτηση ενεργοποίησης η οποία επιλέχθηκε για υλοποίηση είναι η Σιγμοειδής. Η Σιγμοειδής συνάρτηση ενεργοποίησης χαρτογραφεί την τιμή εισόδου σε ένα εύρος μεταξύ του μηδέν και του ένα και χρησιμοποιείται συνήθως σε όλων των ειδών τα νευρωνικά δίκτυα για την εισαγωγή μη γραμμικότητας.

Ένα βήμα βελτιστοποίησης για την Σιγμοειδή Συνάρτηση είναι η τάση που εμφανίζεται στην συνάρτηση να ωθεί αριθμούς που παρατηρούνται να έχουν αξία μεγαλύτερη του έξι (6) ή του μείον έξι (-6) πολύ κοντά στον αριθμό ένα (1) και μηδέν (0) αντίστοιχα. Μετά από πολλές δοκιμές παρατηρήθηκε πως δεν ελαχιστοποιείται η αποδοτικότητα του Νευρωνικού Δικτύου αν χρησιμοποιηθεί ως ελάχιστη τιμή το -4.6 το οποίο θα αντικατοπτρίζει το μηδέν και το 4.6 το οποίο θα αντικατοπτρίζει το ένα. Σε κάθε άλλη τιμή μεγαλύτερη ή μικρότερη ανατίθεται απευθείας ο αριθμός ένα ή μηδέν αντίστοιχα καθώς η Σιγμοειδής συνάρτηση αποτελεί μία πολύπλοκη μαθηματική διαδικασία όσον αφορά την επεξεργαστική ισχύ καθώς εμπεριέχει εκθετικούς υπολογισμούς.

```python
def sigmoid(value):
    if value < -4.6:
        return 0
    elif value > 4.6:
        return 1

    sig = 1 / (1 + math.exp(-value))
    return sig
```

Εικόνα (8) – Κώδικας Python – Βελτιστοποιημένη Σιγμοειδής Συνάρτηση

Μόλις ληφθεί η έξοδος του πρώτου κρυφού στρώματος, χρησιμοποιείται ως είσοδος για τον υπολογισμό της εξόδου του δεύτερου στρώματος, που είναι το αντίστοιχο επίπεδο εξόδου ως προς τη νέα είσοδο. Η διαδικασία που θα ακολουθηθεί είναι η αξιοποίηση της διαδικασίας εμπροσθοτροφοδότησης, με τη χρήση της εξόδου της πρώτης διαδικασίας εμπροσθοτροφοδότησης ως είσοδο και του δεύτερου κρυφού στρώματος, ως «κρυφό στρώμα». Τέλος θα χρησιμοποιηθεί η ίδια ακριβώς διαδικασία και για τον υπολογισμό της εξόδου με είσοδο στη διαδικασία την έξοδο της προηγούμενης που αφορούσε το δεύτερο κρυφό στρώμα.

```python
# Forward Propagation Input → Hidden Layer
hidden_layer = forward_propagation(neural_input, input_hidden_weights, input_hidden_bias)

# Forward Propagation Hidden → Output Layer
output_layer = forward_propagation(hidden_layer, hidden_output_weights, hidden_output_bias)
```

Εικόνα (9) – Κώδικας Python – Διαδικασία Εμπροσθοτροφοδότησης

Τα βάρη και οι πολώσεις στο νευρωνικό δίκτυο ενημερώνονται κατά τη διάρκεια του βήματος της οπισθοδιάδοσης. Η οπισθοδιάδοση χρησιμοποιείται για τον υπολογισμό της διαβάθμισης της συνάρτησης απώλειας σε σχέση με τα βάρη και τις πολώσεις, οι οποίες στη συνέχεια χρησιμοποιούνται για την ενημέρωση των βαρών και των πολώσεων χρησιμοποιώντας τον Αλγόριθμο της σταδιακής καθίζησης. Η συνάρτηση απώλειας που χρησιμοποιείται σε αυτήν την υλοποίηση είναι η συνάρτηση απώλειας MSE (Mean Squared Error), η οποία μετρά τη διαφορά μεταξύ της προβλεπόμενης κλάσης και της πραγματικής κλάσης της εικόνας εισόδου. Η κλίση της συνάρτησης απώλειας σε σχέση με τα βάρη και τις πολώσεις υπολογίζεται χρησιμοποιώντας τον κανόνα της αλυσίδας και τα βάρη και οι πολώσεις ενημερώνονται αφαιρώντας τη διαβάθμιση πολλαπλασιασμένη με το ρυθμό εκμάθησης από τα τρέχοντα βάρη και τις πολώσεις.

```python
# Backpropagation output → hidden
hidden_output_weights += delta_output @ hidden_layer.T * learning_rate_hyperparameter
hidden_output_bias += delta_output * learning_rate_hyperparameter

# Backpropagation hidden → input
hidden_sigmoid_derivative = sigmoid_derivative_matrix(hidden_layer)
delta_hidden_output_weights = hidden_output_weights.T @ delta_output
delta_hidden = scalar_matrix_multiplication(delta_hidden_output_weights, hidden_sigmoid_derivative)
input_hidden_weights += delta_hidden @ neural_input.T * learning_rate_hyperparameter
input_hidden_bias += delta_hidden * learning_rate_hyperparameter
```

Εικόνα (10) – Κώδικας Python – Διαδικασία Οπισθοδιάδοσης

Τα βήματα Εμπροσθοτροφοδότης και οπισθοδιάδοσης για όλο το σύνολο δεδομένων επαναλαμβάνονται πολλές φορές, όπου κάθε επανάληψη ονομάζεται εποχή. Ο αριθμός των εποχών ελέγχεται από την Υπερπαράμετρο εποχής, η οποία μπορεί να οριστεί σε κάθε διαφορετική εκτέλεση. Μετά από κάθε εποχή, η ακρίβεια του νευρωνικού δικτύου στο δοκιμαστικό σύνολο υπολογίζεται για την παρακολούθηση της προόδου της εκπαιδευτικής διαδικασίας. Η Διαδικασία εκπαίδευσης συνεχίζεται μέχρι να επιτευχθεί ο μέγιστος αριθμός εποχών.

3.2 C++

Θεωρητικό Υπόβαθρο και Επεκτάσεις 3.2.α

Σε αυτό το κεφάλαιο, θα εμβαθύνουμε στην χρήση εργαλείων και επεκτάσεων που θα βοηθήσουν στην ανάπτυξη του Νευρωνικού Δικτύου στην γλώσσα C++. Η C++ είναι μία γλώσσα χαμηλού επιπέδου, πράγμα που δίνει μεγαλύτερη ελευθερία στον προγραμματιστή όσον αφορά τους πόρους του υπολογιστή με κύριο κόστος την ευκολία συγγραφής κώδικα. Η C++ περιέχει διάφορες βιβλιοθήκες, αλλά καμία τους δεν έχει υποστήριξη για πράξεις μεταξύ πινάκων (Matrixes) όπως η Βιβλιοθήκη Numpy στην Python. Αντίστοιχα, η εντολές εισόδου εξόδου αποτελούν δυσκολότερο έργο καθώς δεν υπάρχουν έτοιμες συναρτήσεις που να μπορούν να διαβάσουν πίνακες άμεσα σε κάποια δομή πίνακα. Ο Δομές που θα χρησιμοποιήσουμε θα αναφερθούν παρακάτω.

Header Files

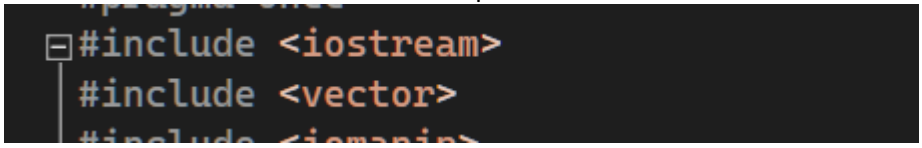
Τα αρχεία κεφαλίδας (Header Files) στη C++ αποτελούν βασικό εργαλείο της γλώσσας. Χρησιμοποιούνται για να παρέχουν τη δήλωση συναρτήσεων, μεταβλητών και άλλων δομών δεδομένων που χρησιμοποιούνται σε ένα πρόγραμμα. Τα αρχεία κεφαλίδας είναι μία συλλογή προ-γραμμένου κώδικα που χρησιμοποιείται για τον καθορισμό των συναρτήσεων και των κλάσεων που απαιτούνται για ένα συγκεκριμένο πρόγραμμα. Τα Header Files στη C++ χρησιμοποιούνται εκτενώς στην ανάπτυξη εφαρμογών λογισμικού. Ένα Header File μπορεί να συμπεριληφθεί σε οποιοδήποτε αρχείο προέλευσης C++ και παρέχει έναν τρόπο πρόσβασης στο αρχείο προέλευσης όσον αφορά τις λειτουργίες και της μεταβλητές που ορίζονται στο αρχείο αυτό.

Υπάρχουν δύο τύποι αρχείων κεφαλίδας στη C++. Τα Header Files του συστήματος και τα Header Files που ορίζονται από τον προγραμματιστή. Τα Header Files του συστήματος είναι αυτά που παρέχονται από τον μεταγλωττιστή της C++ και δεν είναι απαραίτητα ίδια για κάθε διαφορετικό μεταγλωττιστή. Υπάρχουν τυπικά Header Files που παρέχονται από τον μεταγλωττιστή όπως το «iostream», το «vector» και το «string». Τα Header Files που ορίζονται από τον χρήστη είναι αυτά που δημιουργούνται από τον προγραμματιστή για να περιέχουν δηλώσεις για συναρτήσεις και δομές δεδομένων που είναι συγκεκριμένες για ένα αντίστοιχα συγκεκριμένο πρόγραμμα.

Τα Header Files μπορούν με μεγάλη ευκολία να συμπεριληφθούν σε παραπάνω από ένα αρχείο καθώς αρκεί μόνο η δήλωση τους για να μπορεί ο προγραμματιστής να χρησιμοποιήσει τις δυνατότητες ενός Header File. Αυτό βοηθά στη βελτίωση της οργάνωσης του κώδικα και στη μείωση του διπλασιασμού του.

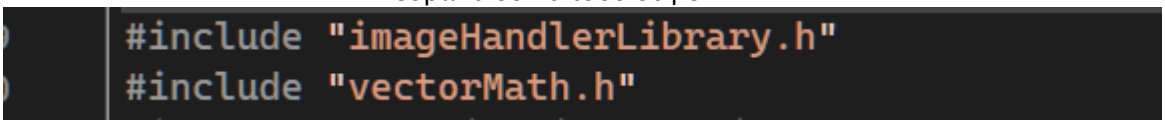
Για να περιληφθεί ένα Header File στον κώδικα ενός αρχείου προέλευσης C++, χρησιμοποιείται η οδηγία «#include». Όσον αφορά την δήλωση της βιβλιοθήκης υπάρχει μία σύμβαση για την διαφοροποίηση των βιβλιοθηκών της C με την C++. Σε αυτή τη σύμβαση το όνομα της βιβλιοθήκης της C περιλαμβάνει το όνομα της βιβλιοθήκης ακολουθούμενο από το επίθεμα «.h». Η σύμβαση για την χρήση βιβλιοθηκών στην C++ περιλαμβάνει το όνομα της βιβλιοθήκης χωρίς την χρήση κάποιου επιθέματος. Η C++ αποτελεί υπερκλάση της C με μικρές εξαιρέσεις με αποτέλεσμα να μπορεί να χρησιμοποιήσεις τις βιβλιοθήκες της C. Αντίστοιχα και η C Μπορεί να χρησιμοποιήσεις κάποιες βιβλιοθήκες της C++ με την προϋπόθεση ο κώδικας μπορεί να μεταγλωττιστεί. Η σύμβαση της ονομασίας των βιβλιοθηκών αποτελεί μόνο σύμβαση και όχι απαραίτητη προϋπόθεση.

Παράδειγμα κώδικα για την χρήση ενός header File συστήματος στον κώδικα του Νευρωνικού Δικτύου στη C++:



Εικόνα (11) – Κώδικας Νευρωνικού Δικτύου στη C++

Παράδειγμα κώδικα για την χρήση ενός header File προγραμματιστή στον κώδικα του Νευρωνικού Δικτύου στη C++:



Εικόνα (12) – Κώδικας Νευρωνικού Δικτύου στη C++

Οι γωνιακές αγκύλες (<>) χρησιμοποιούνται για να υποδείξουν πως το Header Files που περιλαμβάνεται είναι Header File συστήματος. Εάν περιλαμβάνεται ένα Header File που ορίζεται από τον χρήστη, το όνομα του αρχείου περικλείεται σε διπλά (Αγγλικά) εισαγωγικά (“”) αντί για αγκύλες. Σε περίπτωση που ο προγραμματιστής τοποθετήσει τα Header Files που δημιουργήσει στην ίδια τοποθεσία με τα Header Files συστήματος, τότε τα Header Files του προγραμματιστή πρέπει να συμπεριληφθούν από το πρόγραμμα ως Header Files συστήματος.

Όταν ένα Header File περιλαμβάνεται σε ένα αρχείο C++, τα περιεχόμενα του αρχείο αυτού αντιγράφονται στο αρχείο προέλευσης στο σημείο όπου χρησιμοποιείται η οδηγία #include. Αυτό σημαίνει πως ο προγραμματιστής πρέπει να προσέξει την διαχείριση των Header Files που δημιουργεί για περιπτώσεις στις οποίες εμφανίζονται πολλαπλές χρήσεις του ίδιου Header File. Ο τρόπος που διαχειρίζεται η C++ αυτό το πρόβλημα είναι με την χρήση της Λέξης-κλειδί «#pragma once». Η λέξη κλειδί «#pragma once» βοηθάει στην αποφυγή δημιουργίας διπλότυπων στον κώδικα αφαιρώντας τα διπλότυπα αυτά.

Namespaces

Οι χώροι ονομάτων (Namespaces) στη C++ είναι ένα από τα ισχυρά χαρακτηριστικά της γλώσσας C++ που επιτρέπει στους προγραμματιστές να αποφεύγουν τις συγκρούσεις ονομάτων και να οργανώνουν τον κώδικά τους σε λογικές ομάδες. Ένα Namespace, είναι ένας τρόπος ομαδοποίησης σχετικών μεταβλητών, συναρτήσεων και κλάσεων κάτω από ένα μόνο όνομα.

Στη C++ τα Namespaces δηλώνονται χρησιμοποιώντας τη λέξη κλειδί «namespace». Για παράδειγμα, ο ακόλουθος κώδικας δηλώνει ένα Namespace που ονομάζεται imgHl (Image Handler Library) ο οποίος διαχειρίζεται την είσοδο (Input) από το σύνολο των εικόνων εκπαίδευσης.

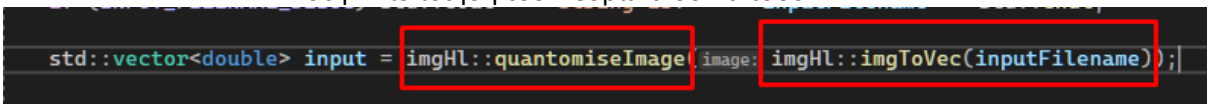
Παράδειγμα Κώδικα για την χρήση Namespace σε ένα από τα Header Files που απαιτήθηκαν στην Υλοποίηση του Νευρωνικού Δικτύου

```
4   #include <fstream>
5   #include <opencv2/core.hpp>
6   #include <opencv2/imgcodecs.hpp>
7   #include <opencv2/highgui.hpp>
8
9    #define IMAGE_FILE_ERROR_CHECK 0
10
11  namespace imgHl{
12
13      //IMAGE TO 1D VECTOR
14      std::vector<double> imgToVec(std::string filename){ ... }
40
41
42      //CONVERT IMAGE VALUES FROM RANGE 0-255 TO 0-1
43      std::vector<double> quantomiseImage(std::vector<double> image){ ... }
50
51
52      //2D VECTOR TO GREYSCALE IMAGE TO PRINT FUNCTION
53      void printImage(std::vector<std::vector<double>> image){ ... }
67
68
69      //2D VECTOR TO GREYSCALE IMAGE TO PRINT FUNCTION
70      //(it also converts all values to MAX value for debug purposes)
71      void debugImage(std::vector<std::vector<double>> image){ ... }
86
87
88      //2D VECTOR TO GREYSCALE IMAGE TO SAVE LOCATION
89      //TAKES A 2D VECTOR AND SAVES IT AS A .JPG IMAGE TO THE "filename" LOCATION
90      void saveImage(std::vector<std::vector<double>> image, std::string filename){ ... }
102
103 }
```

Εικόνα (13) – Κώδικας Νευρωνικού Δικτύου στην C++

Σε αυτό το παράδειγμα, το Namespace «imgHl» περιέχει πέντε συναρτήσεις διαχείρισης εικόνας με την βοήθεια στης βιβλιοθήκης OpenCV (Θα αναλυθεί παρακάτω). Αυτές οι συναρτήσεις είναι ορατές μόνο εντός του Namespace «imgHl». Για την πρόσβαση σε αυτά εκτός του Namespace, το όνομα του Namespace πρέπει να έχει το πρόθεμα της μεταβλητής ή του ονόματος της συνάρτησης. Για παράδειγμα αν θέλουμε να χρησιμοποιήσουμε την συνάρτηση «imgToVec», θα πρέπει να την καλέσουμε χρησιμοποιώντας τον παρακάτω κώδικα:

Παράδειγμα Κώδικα για την χρήση Namespace σε ένα από τα Header Files που απαιτήθηκαν στην Υλοποίηση του Νευρωνικού Δικτύου



Εικόνα (14) – Κώδικας Νευρωνικού Δικτύου στην C++

Ένα από τα κύρια πλεονεκτήματα των Namespaces είναι πως βοηθούν στην αποφυγή συγκρούσεων ονομάτων. Για παράδειγμα, αν θεωρήσουμε δύο βιβλιοθήκες όπου και οι δύο ορίζουν μία συνάρτηση που ονομάζεται «foo()». Εάν αυτές οι δύο βιβλιοθήκες χρησιμοποιούνται στο ίδιο πρόγραμμα, θα προκύψει σφάλμα λόγω σύγκρουσης ονομασίας. Ωστόσο, τοποθετώντας τις συναρτήσεις και τις μεταβλητές κάθε βιβλιοθήκης στο δικό της Namespace, επιτυγχάνεται η αποφυγή αυτού του σφάλματος σύγκρουσης ονομασίας.

Ένα άλλο πλεονέκτημα των Namespaces είναι πως μπορούν να βελτιώσουν την οργάνωση του κώδικα. Με την ομαδοποίηση σχετικών μεταβλητών, συναρτήσεων και κλάσεων μαζί, ο κώδικας μπορεί να γίνει πολύ πιο ευανάγνωστος και κατανοητός. Επιπλέον, τα Namespaces μπορούν να χρησιμοποιηθούν για να παρέχουν μία σαφή ένδειξη από που προήλθε μία συγκεκριμένη συνάρτηση η μεταβλητή, διευκολύνοντας τον εντοπισμό κώδικα και των εντοπισμό σφαλμάτων.

OpenCV

Το OpenCV είναι μία δημοφιλής βιβλιοθήκη ανοιχτού κώδικα που χρησιμοποιείται ευρέως στον τομέα της όρασης υπολογιστών (Computer Vision) και της επεξεργασίας εικόνας. Είναι γραμμένη σε C++ και έχει κώδικα για πολλές γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Python, Java και MATLAB. Το OpenCV παρέχει ένα ευρύ φάσμα λειτουργιών για επεξεργασία εικόνας και βίντεο, συμπεριλαμβανομένου του χειρισμού εικόνας, της ανίχνευσης χαρακτηριστικών, της ανίχνευσης και αναγνώρισης αντικειμένων και της Μηχανικής Μάθησης.

Το OpenCV είναι Cross-Compatible, που σημαίνει πως είναι συμβατό και μπορεί ο κώδικάς του να χρησιμοποιηθεί από Windows, Mac και Linux και υποστηρίζει τόσο επιτραπέζιες, όσο και φορητές Πλατφόρμες. Έχει άδεια χρήσης βάσει της άδειας BSD, η οποία επιτρέπει τη χρήση και διανομή δωρεάν και ανοιχτού κώδικα. Εφόσον γίνει η εγκατάσταση του της βιβλιοθήκης OpenCV η χρήση του είναι τόσο απλή όσο να τοποθετηθούν τα απαραίτητα Header Files στο πρόγραμμα C++ το οποίο τα απαιτεί. Η πρακτική που ακολουθήθηκε όσον αφορά την εγκατάσταση είναι η εισαγωγή των αρχείων της βιβλιοθήκης OpenCV στο εργαλείο Visual Studio, που ήταν ο κειμενογράφος και μεταγλωττιστής που χρησιμοποιήθηκε.

Το OpenCV παρέχει ένα ευρύ φάσμα λειτουργιών για επεξεργασία εικόνας και βίντεο, όπως φιλτράρισμα, ανίχνευση χαρακτηριστικών, ανίχνευση και αναγνώριση αντικειμένων και μηχανική μάθηση. Παρέχει επίσης υποστήριξη για ανάγνωση και εγγραφή αρχείων εικόνας και βίντεο σε διάφορες μορφές.

Ένα από τα δυνατά σημεία της βιβλιοθήκης OpenCV είναι η υποστήριξη του για εφαρμογές όρασης υπολογιστή σε πραγματικό χρόνο, όπως η παρακολούθηση αντικειμένων, η ανίχνευση προσώπου και η επαυξημένη πραγματικότητα. Το OpenCV παρέχει υποστήριξη για δημοφιλείς βιβλιοθήκες μηχανικής Μάθησης, όπως το TensorFlow και το Pytorch, οι οποίες μπορούν να χρησιμοποιηθούν για προηγμένες εργασίες όρασης υπολογιστή.

Το OpenCV χρησιμοποιείται ευρέως σε διάφορους τομείς, όπως η ρομποτική, τα αυτόνομα οχήματα, η ιατρική απεικόνιση και πολλά άλλα. Είναι μια ισχυρή βιβλιοθήκη που εξελίσσεται και βελτιώνεται συνεχώς, με νέες δυνατότητες και λειτουργικότητα που προστίθεται σε κάθε νέα κυκλοφορία νέας εκδοχής της βιβλιοθήκης.

Hash Tables

Οι πίνακες κατακερματισμού (Hash Tables), γνωστοί και ως χάρτες κατακερματισμού (Hash Maps), είναι μία δημοφιλής δομή δεδομένων στην επιστήμη των υπολογιστών που επιτρέπουν την αποτελεσματική αποθήκευση και ανάκτηση ζευγών κλειδιών-τιμών. Η βασική ιδέα πίσω από έναν πίνακα κατακερματισμού είναι να χρησιμοποιηθεί μία συνάρτηση κατακερματισμού για να αντιστοιχιστεί κάθε κλειδί σε έναν πίνακα ευρετήριο και να αποθηκευτεί τη σχετική τιμή σε αυτό το ευρετήριο. Αυτό επιτρέπει σταθερή χρονική πρόσβαση στην τιμή που σχετίζεται με ένα δεδομένο κλειδί.

Οι συναρτήσεις κατακερματισμού είναι ένα ουσιαστικό συστατικό των πινάκων κατακερματισμού, καθώς χρησιμοποιούνται για την αντιστοίχιση των κλειδιών σε δείκτες πίνακα. Στην ιδανική περίπτωση, μία καλή συνάρτηση κατακερματισμού θα κατανείμει ομοιόμορφα τα κλειδιά σε όλο τον πίνακα, ελαχιστοποιώντας την πιθανότητα συγκρούσεων (δύο κλειδιά αντιστοιχίζονται στο ίδιο κελί πίνακα) Η συνάρτηση κατακερματισμού θα πρέπει επίσης να είναι ντετερμινιστική, έτσι ώστε με το ίδιο κλειδί εισόδου, να επιστρέφει πάντα τον ίδιο δείκτη εξόδου.

Οι συγκρούσεις μπορούν να αντιμετωπιστούν με διάφορους τρόπους. Μία προσέγγιση είναι η χρήση ξεχωριστής αλυσίδας, όπου κάθε δείκτης κελιού πίνακα αποθηκεύεται σε μία συνδεδεμένη λίστα ζευγών κλειδιών-τιμών που έχουν κατακερματιστεί σε αυτόν τον δείκτη κελιού. Εάν ένα νέο κλειδί κατακερματίζεται σε έναν δείκτη κελιού που περιέχει ήδη ένα κλειδί, η τιμή του μπορεί να προστεθεί στη συνδεδεμένη λίστα σε αυτόν τον δείκτη κελιού. Μία άλλη προσέγγιση είναι η ανοιχτή διεθυνσιοδότηση, όπου εάν συμβεί μία σύγκρουση, ο αλγόριθμος προσπαθεί να βρει μία κενή υποδοχή στον πίνακα για να αποθηκεύσει το ζεύγος κλειδιού-τιμής.

Κατά την εφαρμογή ενός πίνακα κατακερματισμού στη C++, Πρέπει να ληφθούν αρκετές αποφάσεις σχεδιασμού. Μία βασική απόφαση είναι η επιλογή συνάρτησης κατακερματισμού. Υπάρχουν πολλές πιθανές συναρτήσεις κατακερματισμού, που κυμαίνονται από απλές αριθμητικές πράξεις όμως Modulo και πολλαπλασιασμός, έως πιο σύνθετους αλγόριθμους όπως κρυπτογραφικούς κατακερματισμούς. Η επιλογή της συνάρτησης κατακερματισμού θα εξαρτηθεί από τη συγκεκριμένη περίπτωση χρήσης και της απαιτήσεις διανομής και αποφυγής σύγκρουσης.

Μία άλλη απόφαση είναι το μέγεθος του πίνακα που χρησιμοποιείται για την αποθήκευση των ζευγών κλειδιού-τιμής. Αυτό θα εξαρτηθεί από τον αναμενόμενο αριθμό στοιχείων που θα αποθηκευτούν, καθώς και από τον επιθυμητό συντελεστή φορτίου (αναλογία των στοιχείων προς το μέγεθος του πίνακα). Ένας υψηλότερος συντελεστής φορτίου μπορεί να οδηγήσει σε περισσότερες συγκρούσεις και πιο αργούς χρόνους πρόσβασης, ενώ ένας χαμηλότερος συντελεστής φορτίου θα σπαταλήσει τη μνήμη.

Υλοποίηση στην Γλώσσα C++ 3.2.β

Ο κώδικας στη C++ υλοποιεί δύο Τεχνητά Νευρωνικά Δίκτυά για αναγνώριση εικόνας. Το κάθε ένα από αυτά περιέχει ένα κρυφό στρώμα, το στρώμα εισόδου και το στρώμα εξόδου. Το Νευρωνικό Δίκτυο εκπαιδεύεται σε ένα σύνολο δεδομένων Τεσσάρων χιλιάδων εικόνων και έπειτα αξιολογείται σε Χίλιες εικόνας. Τα δεδομένα εισόδου είναι εικόνες χειρόγραφων ψηφίων και η έξοδος του Τεχνητού Νευρωνικού Δικτύου είναι το ψηφίο το οποίο αναγνωρίζεται από το δίκτυο.

Η διαδικασία διαχείρισης της εισόδου αποτελείται από δύο συναρτήσεις. Την **importNumersFromFile** και την **importNumbersFromFile2**. Εφόσον το Νευρωνικό μας Δίκτυο αποτελείται από τρία στρώματα (Μαζί με το στρώμα εισόδου και εξόδου), θα χρειαστούν δύο πίνακες βαρών (Ένας πίνακας βαρών για το ζευγάρι Στρώμα Εισόδου – Κρυφό Στρώμα και ένας ακόμα πίνακας βαρών για το ζευγάρι Κρυφό Στρώμα – Στρώμα Εξόδου) Αντίστοιχα θα χρειαστούν και δύο πίνακες πολώσεων (Bias Matrix). Η αρχικοποίηση των τιμών για κάθε πίνακα βαρών υλοποιείται με την αξιοποίηση της συνάρτησης **createNeuralNetworkWeights** για του πίνακες δύο διαστάσεων και με την **numberToVector** για τους πίνακες μίας διάστασης που αναφέρονται στις πολώσεις. Η **createNeuralNetworkWeights** δημιουργεί έναν τυχαίο δυσδιάστατο πίνακα με τιμές ανάμεσα στο μείον μηδέν κόμμα πέντε (0.5) και το μηδέν κόμμα πέντε (0.5). Η συνάρτηση **numberToVector** φτιάχνει έναν πίνακα με διαστάσεις που ορίζει ο χρήστης ο οποίος αποτελείται από ένα νούμερο που επίσης ορίζει ο Χρήστης. Οι πίνακες πολώσεων αρχικοποιούνται με την τιμή μηδέν (0).

Σε περίπτωση που ο προγραμματιστής θέλει να συνεχίσει μία εκπαίδευση, υπάρχουν οι συναρτήσεις **importNumbersFromFile** και **importNumbersFromFile2** που φορτώνουν τις τιμές από τον δίσκο. Υπάρχει επιλογή για τον προγραμματιστή να αποθηκεύσει τα αποτελέσματά του μετά το πέρας κάθε εκπαίδευσης με την χρήση των συναρτήσεων **saveMatrix** και **saveVector**.

Παράδειγμα Κώδικα για την χρήση των συναρτήσεων που δημιουργούν τα βάρη για τις πολώσεις και των πινάκων βαρών που απαιτήθηκαν στην Υλοποίηση του Νευρωνικού Δικτύου

```
//Weights
if (startFromScratch) {
    i_h_w = createNeuralNetworkWeights(sizeI: 16, sizeJ: IMAGE_SIZE * IMAGE_SIZE);
    i_h_b = vecMath::numberToVector(num: 0, size: 16);
    h_o_w = createNeuralNetworkWeights(sizeI: 10, sizeJ: 16);
    h_o_b = vecMath::numberToVector(num: 0, size: 10);
```

Εικόνα (15) – Κώδικας Νευρωνικού Δικτύου στην C++

Παράδειγμα Κώδικα για την χρήση των συναρτήσεων που φορτώνουν τα βάρη για τις πολώσεις και των πινάκων βαρών που απαιτήθηκαν στην Υλοποίηση του Νευρωνικού Δικτύου

```
else {
    i_h_w = vecDebug::importNumbersFromFile(filename: FILTER_INPUT_HIDDEN_SAVE_PATH, SizeI: 16, SizeJ: IMAGE_SIZE * IMAGE_SIZE);
    h_o_w = vecDebug::importNumbersFromFile(filename: FILTER_HIDDEN_OUTPUT_SAVE_PATH, SizeI: 10, SizeJ: 16);
    i_h_b = vecDebug::importNumbersFromFile2(filename: BIAS_INPUT_HIDDEN_SAVE_PATH, size: 16);
    h_o_b = vecDebug::importNumbersFromFile2(filename: BIAS_HIDDEN_OUTPUT_SAVE_PATH, size: 10);
}
```

Εικόνα (16) – Κώδικας Νευρωνικού Δικτύου στην C++

Παράδειγμα Κώδικα για την χρήση των συναρτήσεων που φορτώνουν τα βάρη για τις πολώσεις και των πινάκων βαρών που απαιτήθηκαν στην Υλοποίηση του Νευρωνικού Δικτύου

```
if (SAVE_FILTERS) {
    vecDebug::saveMatrix(matrix: i_h_w, filename: FILTER_INPUT_HIDDEN_SAVE_PATH);
    vecDebug::saveMatrix(matrix: h_o_w, filename: FILTER_HIDDEN_OUTPUT_SAVE_PATH);

    vecDebug::saveVector(vector: i_h_b, filename: BIAS_INPUT_HIDDEN_SAVE_PATH);
    vecDebug::saveVector(vector: h_o_b, filename: BIAS_HIDDEN_OUTPUT_SAVE_PATH);
}
```

Εικόνα (17) – Κώδικας Νευρωνικού Δικτύου στην C++

Το Τεχνητό Νευρωνικό Δίκτυο αρχικά τροφοδοτείται με μία εικόνα εισόδου, η οποία αρχικά αναπαρίσταται ως ένας δυσδιάστατος πίνακας τύπου **std::vector<std::vector<int>>** Επτακοσίων ογδόντα τεσσάρων (784) στοιχείων (Το μέγεθος της εικόνας είναι 28x28). Η εικόνα μετατρέπεται από την αρχική της αναπαράσταση (.png) σε Vector δύο διαστάσεων με τη βοήθεια της συνάρτησης **imgToVec**. Για να την αξιοποίηση της συνάρτησης Sigmoid έπρεπε να συμπιεστούν τα στοιχεία της εικόνας σε τιμές μεταξύ του μηδέν και του ένα. Για την επίτευξη αυτού του σκοπού δημιουργήθηκε η συνάρτηση **quantomizeImage** η οποία κάνει αυτή τη δουλειά.

Παράδειγμα Κώδικα για την χρήση των συναρτήσεων που φορτώνουν τα βάρη για τις πολώσεις και των πινάκων βαρών που απαιτήθηκαν στην Υλοποίηση του Νευρωνικού Δικτύου

```
std::vector<double> input = imgHl::quantomiseImage(image: imgHl::imgToVec(inputFilename));
```

Εικόνα (18) – Κώδικας Νευρωνικού Δικτύου στην C++

Η διαδικασία Εμπροσθοτροφοδότησης ξεκινάει περνώντας τον πίνακα στήλη (είσοδο) μέσω του Νευρωνικού Δικτύου. Το πρώτο βήμα στη Εμπροσθοτροφοδότηση, είναι ο υπολογισμός της εξόδου του πρώτου κρυφού στρώματος, το οποίο γίνεται πολλαπλασιάζοντας τον πίνακα εισόδου με τον πίνακα βάρους που συνδέει το στρώμα εισόδου με το κρυφό στρώμα και προσθέτοντας το κρυφό στρώμα πόλωσης. Στη συνέχεια, το αποτέλεσμα περνά μέσα από τη συνάρτηση ενεργοποίησης. Η συνάρτηση ενεργοποίησης που επιλέχθηκε για το μοντέλο είναι η Σιγμοειδής. Η Σιγμοειδής συνάρτηση ενεργοποίησης χαρτογραφεί την τιμή εισόδου σε ένα εύρος μεταξύ του μηδέν και του ένα και χρησιμοποιείται συνήθως σε όλων των ειδών τα νευρωνικά δίκτυα για την εισαγωγή μη γραμμικότητας.

Ένα βήμα βελτιστοποίησης για την Σιγμοειδή Συνάρτηση είναι η τάση που εμφανίζεται στην συνάρτηση να ωθεί αριθμούς που παρατηρούνται να έχουν αξία μεγαλύτερη του έξι (6) ή του μείον έξι (-6) πολύ κοντά στον αριθμό ένα (1) και μηδέν (0) αντίστοιχα.

Παράδειγμα κώδικα για την Χρήση της Συνάρτησης Sigmoid στην C++

```
double sigmoid(double x) {
    return 1 / (1 + exp(_x: -x));
}
```

Εικόνα (19) – Κώδικας στο Header File Activation Functions

Μόλις ληφθεί η έξοδος του πρώτου κρυφού στρώματος, χρησιμοποιείται ως είσοδος για τον υπολογισμό της εξόδου του δεύτερου στρώματος, που είναι το αντίστοιχο επίπεδο εξόδου ως προς τη νέα είσοδο. Η διαδικασία που θα ακολουθηθεί είναι η αξιοποίηση της διαδικασίας εμπροσθοτροφοδότησης, με τη χρήση της εξόδου της πρώτης διαδικασίας εμπροσθοτροφοδότησης ως είσοδο και του δεύτερου κρυφού στρώματος, ως «κρυφό στρώμα». Τέλος θα χρησιμοποιήθηκε η ίδια ακριβώς διαδικασία και για τον υπολογισμό της εξόδου με είσοδο στη διαδικασία την έξοδο της προηγούμενης που αφορούσε το δεύτερο κρυφό στρώμα.

Παράδειγμα κώδικα για την διαδικασία εμπροσθοτροφοδότησης

```
hidden = vecMath::addVectors(first: vecMath::matrixVectorMultiplication(matrix: i_h_w, vector: input), second: i_h_b);
for (int i = 0; i < hidden.size(); i++) {
    if (hidden[i] > 4.6) hidden[i] = 1;
    else if (hidden[i] < -4.6) hidden[i] = 0;
    else hidden[i] = acf::sigmoid(x: hidden[i]);
}

output = vecMath::addVectors(first: vecMath::matrixVectorMultiplication(matrix: h_o_w, vector: hidden), second: h_o_b);
for (int i = 0; i < output.size(); i++) {
    if (output[i] > 4.6) output[i] = 1;
    else if (output[i] < -4.6) output[i] = 0;
    else output[i] = acf::sigmoid(x: output[i]);
}
```

Εικόνα (20) – Κώδικας Νευρωνικού Δικτύου στη C++

Τα βάρη και οι πολώσεις στο νευρωνικό δίκτυο ενημερώνονται κατά τη διάρκεια του βήματος της οπισθοδιάδοσης. Η οπισθοδιάδοση χρησιμοποιείται για τον υπολογισμό της διαβάθμισης της συνάρτησης απώλειας σε σχέση με τα βάρη και τις πολώσεις, οι οποίες στη συνέχεια χρησιμοποιούνται για την ενημέρωση των βαρών και των πολώσεων χρησιμοποιώντας τον Αλγόριθμο της σταδιακής καθίζησης. Η συνάρτηση απώλειας που χρησιμοποιείται σε αυτήν την υλοποίηση είναι η συνάρτηση απώλειας MSE (Mean Squared Error), η οποία μετρά τη διαφορά μεταξύ της προβλεπόμενης κλάσης και της πραγματικής κλάσης της εικόνας εισόδου. Η κλίση της συνάρτησης απώλειας σε σχέση με τα βάρη και τις πολώσεις υπολογίζεται χρησιμοποιώντας τον κανόνα της αλυσίδας και τα βάρη και οι πολώσεις ενημερώνονται αφαιρώντας τη διαβάθμιση πολλαπλασιασμένη με το ρυθμό εκμάθησης από τα τρέχοντα βάρη και τις πολώσεις.

Παράδειγμα Κώδικα για την διαδικασία Οπισθοδιάδοσης

```
//Backpropagation Output -> Hidden Layer
//std::vector<double> errorVector = calculateCost(output, trueOutput);
std::vector<double> delta_o = vecMath::subtractVectors(first: output, second: trueOutput);
//calculate deltas for Output -> Hidden layer
std::vector<std::vector<double>> delta_w_o_h = vecMath::scalarMatrixMultiplication(-LEARNING_RATE, matrix: vecMath::columnRowMultiplication(rowMatrix: hidden, columnMatrix: delta_o));

//Update Hidden -> Output weights based on those
h_o_w = vecMath::addMatrix(first: delta_w_o_h, second: h_o_w);
//update the bias based on the output
h_o_b = vecMath::addVectors(first: vecMath::scalarVectorMultiplication(-LEARNING_RATE, vec: delta_o), second: h_o_b);

//Backpropagation Hidden -> Input Layer
std::vector<double> delta_h = vecMath::vectorMultiplication(vectorA: vecMath::matrixVectorMultiplication(vecMath::matrixTranspose(matrix: h_o_w), vector: delta_o), vectorB: sigmoidDerivative(hiddenOut: hidden));
//calculate deltas for Hidden -> Input layer
std::vector<std::vector<double>> delta_w_h_i = vecMath::scalarMatrixMultiplication(-LEARNING_RATE, matrix: vecMath::columnRowMultiplication(rowMatrix: input, columnMatrix: delta_h));
//Update Input -> Hidden weights based on those deltas
i_h_w = vecMath::addMatrix(first: delta_w_h_i, second: i_h_w);

//update the bias based on the output
i_h_b = vecMath::addVectors(first: vecMath::scalarVectorMultiplication(-LEARNING_RATE, vec: delta_h), second: i_h_b);
```

Εικόνα (21) – Κώδικας Νευρωνικού Δικτύου στη C++

Τα βήματα Εμπροσθοτροφοδότης και οπισθοδιάδοσης για όλο το σύνολο δεδομένων επαναλαμβάνονται πολλές φορές, όπου κάθε επανάληψη ονομάζεται εποχή. Ο αριθμός των εποχών ελέγχεται από την Υπερπαράμετρο εποχής, η οποία μπορεί να οριστεί σε κάθε διαφορετική εκτέλεση. Μετά από κάθε εποχή, η ακρίβεια του νευρωνικού δικτύου στο δοκιμαστικό σύνολο υπολογίζεται για την παρακολούθηση της προόδου της εκπαιδευτικής διαδικασίας. Η Διαδικασία εκπαίδευσης συνεχίζεται μέχρι να επιτευχθεί ο μέγιστος αριθμός εποχών.

Όσον αφορά την υλοποίηση έχουν χρησιμοποιηθεί διάφορες μικρές αλλαγές στον κώδικα όπως η υλοποίηση Hash Table για την συνάρτηση Sigmoid και για την παράγωγο της, καθώς είναι συναρτήσεις που κοστίζουν στην επίλυση τους. Εκτός αυτών έχει υλοποιηθεί έμμεσα διαδικασία για χρήση σταθερούς υποδιαστολής πράξεις. Αυτή η υλοποίηση θα αναφερθεί κυρίως στο κεφάλαιο με τα συμπεράσματα καθώς ένας από τους πιθανούς μελλοντικούς στόχους είναι να υλοποιηθεί αυτό το μοντέλο σε ακόμα πιο χαμηλό επίπεδό όπως ένα FPGA

Παράδειγμα κώδικα για την υλοποίηση με Πίνακα Κατακερματισμού

```
//Start is the beginning of the array and precision is 10 ^ Decimals so
//to get 2 Decimals of precision we need Decimals to be 10 ^ 2 = 100
//Example for number 4.5321, in order to get 2 decimals of precision we will
//do 4.5321 * 100 to get the precision.
int sigmoidHashLUT(double x, double negativeRange, int decimals) {
    return (int)((x + abs(_Xx: negativeRange)) * decimals);
}


std::vector<double> generateSigmoidLUT(double start , double end , double step) {
    std::vector<double> retval((end - start) / step);

    for (double i = start; i < (end - start) / step; i += step) {
        retval.push_back(_Val: sigmoid(x: i));
    }

    return retval;
}
```

Εικόνα (22) – Κώδικας στο αρχείο Activation Functions

Παράδειγμα χρήσης Πίνακα Κατακερματισμού στην διαδικασία Εμπροσθοτροφοδότησης

```
hidden = vecMath::addVectors(first: vecMath::matrixVectorMultiplication(matrix: i_h_w, vector: input), second: i_h_b);
for (int i = 0; i < hidden.size(); i++) {
    if (hidden[i] > 4.6) hidden[i] = 1;
    else if (hidden[i] < -4.6) hidden[i] = 0;
    else hidden[i] = sigLUT[(int)acf::sigmoidHashLUT(x: hidden[i], negativeRange: -4.6, decimals: 100)];
}
output = vecMath::addVectors(first: vecMath::matrixVectorMultiplication(matrix: h_o_w, vector: hidden), second: h_o_b);
for (int i = 0; i < output.size(); i++) {
    if (output[i] > 4.6) output[i] = 1;
    else if (output[i] < -4.6) output[i] = 0;
    else output[i] = sigLUT[(int)acf::sigmoidHashLUT(x: output[i], negativeRange: -4.6, decimals: 100)];
}
```

Εικόνα (22) – Κώδικας Νευρωνικού Δικτύου στη C++

ΚΕΦΑΛΑΙΟ 4 Αποτελέσματα

4.1 Υλικό που Χρησιμοποιήθηκε

Το υλικό που χρησιμοποιήθηκε για την μέτρηση της απόδοσης αποτελείται από τρεις ξεχωριστούς υπολογιστές. Ο σκοπός του Μοντέλου που υλοποιήθηκε ήταν να επιτευχθεί ποσοστό επιτυχίας ανώτερο του 90%. Οι προσπάθειες για να επιτευχθεί αυτό το ποσοστό αποτέλεσαν περίπου πέντε έως δέκα επαναλήψεις της διαδικασίας εκπαίδευσης (5-10 εποχές). Για να συγκριθούν τα αποτελέσματα κάθε υλοποίηση αξιοποίησε την εκπαίδευση για δέκα εποχές και μετρήθηκε ο χρόνος για αυτή τη διαδικασία.

Υλικό Υπολογιστών

Specs		
	CPU	Ram
High Performance PC	Intel Core I9 - 12900k 5.2 GHz	DDR5 6000 MT/s
Average Performance Pc	Intel Core I7 - 8700K 4.2 GHz	DDR4 3200 MT/s
Low Performance Laptop	Intel Core I5 - 7200U 3.1 GHz	DDR4 3200 MT/s

4.1 Μετρήσεις Απόδοσης

Η απόδοση για κάθε αλγόριθμο, για κάθε ξεχωριστό υλικό φαίνεται στον παρακάτω πίνακα

	High Performance PC			
	Lookup Table	No Lookup Table	Low Level Implemenation	No LookupTable Python
Time in Seconds	48.576	49.595	42.201	355.14
	48.152	48.609	42.261	357.554
	48.17	49.349	42.234	350.811
	48.207	48.593	42.265	349.322
	48.586	48.321	42.339	343.553

	Average Performance Pc			
	Lookup Table	No Lookup Table	Low Level Implemenation	No LookupTable Python
Time in Seconds	80.167	79.962	75.055	589.152
	80.761	80.194	75.332	581.776
	80.138	79.595	74.998	581.038
	80.329	79.545	75.015	585.523
	80.382	80.663	75.564	581.038

	Low Performance Laptop			
	Lookup Table	No Lookup Table	Low Level Implemenation	No LookupTable Python
Time in Seconds	92.457	93.082	78.167	676.922
	88.938	93.365	76.789	668.674
	92.388	91.803	77.553	706.223
	90.041	90.08	78.011	671.062
	91.085	90.134	76.883	712.24

ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα

5.1 Παρατηρήσεις Απόδοσης

Η χρήση της γλώσσας προγραμματισμού C++ αποδεικνύεται πως είναι μία γλώσσα που μπορεί να προσφέρει ταχύτητα στην εκπαίδευση των Τεχνητών Νευρωνικών Δικτύων. Η συνολική διαφορά απόδοσης σε σχέση με την γλώσσα Python είναι περίπου Επτακόσια Δέκα Οκτώ τοις εκατό (718%) βελτίωση όσον αφορά τον χρόνο εκπαίδευσης όσον αφορά τον Υπολογιστή υψηλής απόδοσης, όσον αφορά τον υπολογιστή μέτριας απόδοσης η βελτίωση ήταν περίπου επτακόσια είκοσι εννιά τοις εκατό (729%) και για το χαμηλής απόδοσης Λάπτοπ η βελτίωση ήταν περίπου επτακόσια πενήντα πέντε τοις εκατό (755%). Με το πέρας των αποτελεσμάτων μπορούμε να παρατηρήσουμε πως η μηχανική μάθηση είναι ένας κλάδος ο οποίος μπορεί να χρησιμοποιήσει γλώσσες προγραμματισμού χαμηλού επιπέδου για την χρήση νευρωνικών δικτύων και ιδιαίτερα για την εκπαίδευσή τους.

5.2 Μελλοντικές Προτάσεις

Το μοντέλο που παρουσιάστηκε μπορεί να εξελιχθεί με διάφορους τρόπους. Ένας από αυτούς είναι η περεταίρω εξέλιξη όσον αφορά τις υποστηριζόμενες δομές, όπως οι συναρτήσεις ενεργοποίησης. Οι συναρτήσεις ενεργοποίησης παίζουν καθοριστικό ρόλο και η Sigmoid που χρησιμοποιείται στο μοντέλο της C++ και της Python προσθέτει μεγάλο κόστος όσον αφορά τον χρόνο εκτέλεσης. Μία από τις εκδοχές είναι να προστεθεί η Συνάρτηση Softmax και η Relu. Αυτό επίσης απαιτεί και την εισαγωγή των παραγώγων αυτών των συναρτήσεων εφόσον η χρήση τους είναι απαραίτητη στο βήμα της οπισθοδιάδοσης.

Η εισαγωγή περισσότερων υπερπαραμέτρων μπορεί να αποτελέσει ένα βήμα που θα βοηθήσει την αυτοματοποίηση της εκπαίδευσης καθώς με την χρήση μεταβλητών που μετρούν την απόδοση κατά τη διάρκεια της εκπαίδευσης θα μπορούσαν να προσφέρουν την δυνατότητα να αφαιρεθεί η μεταβλητή των εποχών και να προστεθεί ένας βρόγχος (While Loop) όπου η εκτέλεση θα σταματούσε μετά το πέρας του επιθυμητού ποσοστού επιτυχίας.

Μία από τις πιο σημαντικές βελτιώσεις, θα ήταν η παραλληλοποίηση της εκπαίδευσης. Ένας ξεχωριστός και πολύ χρήσιμος τρόπος εκπαίδευσης είναι η χρήση πολυνηματικού προγραμματισμού. Είτε με την βοήθεια καρτών γραφικών ή ακόμα και με τα νήματα ενός κλασσικού επεξεργαστή πολλών πυρήνων. Η χρήση πολυνηματικού προγραμματισμού θα ανέλυε το πρόβλημα του μεγάλου όγκου δεδομένων και θα βοηθούσε στην ταυτόχρονη εκτέλεση εκπαίδευσης στα ψηφία που αποτελούν τα δεδομένα μάθησης.

Μία ακόμα βελτιστοποίηση θα μπορούσε να ήταν η υλοποίηση αυτού του μοντέλου σε ακόμα χαμηλότερο επίπεδο. Πιο συγκεκριμένα θα μπορούσε αυτό το μοντέλο να υλοποιηθεί για FPGA με στόχο την φτηνή και αποδοτική εκπαίδευση.

Τέλος μία ακόμα βελτίωση θα ήταν η υλοποίηση του μοντέλου με αντικειμενοστραφή τρόπο τοποθετώντας μόνο τις απαραίτητες λειτουργίες με στόχο την ευκολία στην χρήση από άλλους προγραμματιστές. Η χρήση στατικών

πινάκων θα βοηθούσε στην ταχύτητα εκτέλεσης όσον αφορά τις γλώσσες χαμηλού επιπέδου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

"Artificial Intelligence [AI] Market Growth, Trends: Forecast, 2029." *Artificial Intelligence [AI] Market Growth, Trends | Forecast, 2029*, https://www.fortunebusinessinsights.com/industry-reports/artificial-intelligence-market-100114.

"What Are Neural Networks?" *IBM*, https://www.ibm.com/topics/neural-networks.

"Gradient Descent." *Gradient Descent - ML Glossary Documentation*, https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html.

Ali, Amir. "Artificial Neural Network (ANN)." *Medium*, The Art of Data Scicne, 7 Dec. 2019, https://medium.com/machine-learning-researcher/artificial-neural-network-ann-4481fa33d85a.

Turing. "Importance of Neural Network Bias and How to Add It." *Importance of Neural Network Bias and How to Add It*, Turing Enterprises Inc, 29 Sept. 2022, https://www.turing.com/kb/necessity-of-bias-in-neural-networks.

DeepAI. "Weight (Artificial Neural Network)." *DeepAI*, DeepAI, 17 May 2019, https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network.

Calculus I - Chain Rule, https://tutorial.math.lamar.edu/classes/calci/chainrule.aspx.

Turin, Arseny. "Gradient Descent from Scratch." *Medium*, Towards Data Science, 4 Feb. 2020, https://towardsdatascience.com/gradient-descent-from-scratch-e8b75fa986cc.

"Writing Compilers and Interpreters." *Google Books*, Google, https://books.google.gr/books?hl=en&lr=&id=EUs94gRZHEUC&oi=fnd&pg=PT12&dq=compilers%2Bvs%2Binterpreters&ots=X9soBdKeRd&sig=h6tTulaubpdahcHXCkNP29kVLAE&redir_esc=y#v=onepage&q=compilers%20vs%20interpreters&f=false.

"Compiler vs Interpreter." *GeeksforGeeks*, GeeksforGeeks, 17 Jan. 2022, https://www.geeksforgeeks.org/compiler-vs-interpreter-2/.

Dan Margalit, Joseph Rabinoff. "Interactive Linear Algebra." *Matrix Multiplication*, https://textbooks.math.gatech.edu/ila/matrix-multiplication.html.

"What Is Artificial Intelligence (AI) ?" *IBM*, https://www.ibm.com/topics/artificial-intelligence.

"What Is Machine Learning?" *IBM*, https://www.ibm.com/topics/machine-learning.

Toomet, Ott. "Machine Learning in Python." *Chapter 3 Numpy and Pandas*, 22 Feb. 2023, http://faculty.washington.edu/otoomet/machinelearning-py/numpy-and-pandas.html.

"Image Module." *Pillow (PIL Fork)*, https://pillow.readthedocs.io/en/stable/reference/Image.html.

"Introduction." *OpenCV*, https://docs.opencv.org/4.x/d1/dfb/intro.html.

"C++ Program for Hashing with Chaining." *GeeksforGeeks*, GeeksforGeeks, 2 Jan. 2023, https://www.geeksforgeeks.org/c-program-hashing-chaining/.