Εφαρμογή Συνελικτικών Νευρωνικών Δικτύων σε περίπλοκα πρότυπα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σταματόπουλος Ιωάννης

Επιβλέπων: Χαράλαμπος Στρουθόπουλος, Καθηγητής

ΣΕΡΡΕΣ – Μάρτιος 2024

Πρόλογος

Ο σκοπός της διπλωματικής εργασίας είναι η διερεύνηση του βαθμού πρακτικής εφαρμογής συνελικτικών νευρωνικών δικτύων (ΣΝΔ) στο πλαίσιο της αναγνώρισης ελληνικών φαγητών. Μία πρόσθετη στόχευση της μελέτης αυτής είναι η σαφέστερη εννοιολογική επεξήγηση των διαστάσεων αλλά και των τρόπων λειτουργίας ενός συνελικτικού νευρωνικού δικτύου, η εξοικείωση με τα κύρια εργαλεία που συνδράμουν στην υλοποίηση ενός ΣΝΔ αλλά περαιτέρω οδηγίες για τον τρόπο της ορθής εγκατάστασης των εργαλείων αυτών για την εφαρμογή τους στην αναγνώριση εικόνων ελληνικών φαγητών.

Σχετικά με τη διάρθρωση της εργασίας, αρχικά αναπτύσσονται το θεωρητικό πλαίσιο με τις βασικές αρχές που διέπουν τα συνελικτικά νευρωνικά δίκτυα (ΣΝΔ). Στη συνέχεια, παρουσιάζονται οι τεχνικές που χρησιμοποιήθηκαν κατά τη φάση περάτωσης της μελέτης αυτής.

Τέλος παρουσιάζονται οι μέθοδοι που πραγματοποιήθηκαν στην παρούσα εργασία και ακολουθεί η σύγκριση των αποτελεσμάτων τους.

Περίληψη

Ένα συνελικτικό νευρωνικό δίκτυο (CNN) είναι ένας τύπος μοντέλου βαθιάς μάθησης που είναι ιδιαίτερα αποτελεσματικός για αναγνώριση εικόνων και οπτικών δεδομένων. Εμπνέεται από το ανθρώπινο οπτικό σύστημα, όπου οι οπτικές πληροφορίες επεξεργάζονται μέσω της διάταξης νευρώνων που είναι ευαίσθητοι σε συγκεκριμένα οπτικά ερεθίσματα.

Η κύρια ιδέα πίσω από τα CNN είναι να εφαρμοστούν φίλτρα (επίσης γνωστά ως συνελικτικοί πυρήνες) στις εικόνες της εισόδου του, τα οποία ουσιαστικά σαρώνουν την εικόνα και εξάγουν κατάλληλα χαρακτηριστικά. Αυτά τα χαρακτηριστικά μπορεί να είναι απλά, όπως άκρες και κλίσεις, ή πιο πολύπλοκα, όπως σχήματα και μοτίβα. Τα φίλτρα εξάγουν αυτά τα χαρακτηριστικά εκτελώντας μαθηματικές λειτουργίες όπως η συνέλιξη και η συσσώρευση.

Τα CNN συνήθως αποτελούνται από αρκετά επίπεδα. Το πρώτο επίπεδο είναι το επίπεδο εισόδου, που λαμβάνει τα αρχικά δεδομένα της εικόνας. Τα επόμενα επίπεδα αποτελούνται από συνελικτικά επίπεδα, τα οποία εφαρμόζουν φίλτρα στην είσοδο και εξάγουν διάφορα χαρακτηριστικά σε κάθε επίπεδο. Αυτά τα επίπεδα ακολουθούνται από επίπεδα σμίκρυνσης, τα οποία μειώνουν τις χωρικές διαστάσεις και καθιστούν την αναπαράσταση πιο διαχειρίσιμη. Τέλος, χρησιμοποιούνται πλήρως συνδεδεμένα επίπεδα για κατηγοριοποίηση ή προβλέψεις.

Ένα αξιοσημείωτο πλεονέκτημα των CNN είναι η ικανότητά τους να μαθαίνουν ιεραρχικές αναπαραστάσεις. Τα αρχικά επίπεδα εξάγουν χαμηλού επιπέδου χαρακτηριστικά, όπως ακμές και υφή , ενώ τα τελικά επίπεδα μαθαίνουν πιο αφηρημένα και πολύπλοκα χαρακτηριστικά, όπως σχήματα αντικειμένων. Αυτή η ιεραρχική μάθηση βοηθά τα CNN να εξάγουν αυτόματα και να "κατανοούν" τα κατάλληλα χαρακτηριστικά από τα δεδομένα εισόδου.

Τα CNN χρησιμοποιούνται ευρέως σε εργασίες υπολογιστικής όρασης, όπως η ταξινόμηση εικόνων, η ανίχνευση αντικειμένων και η τμηματοποίηση εικόνων. Έχουν επιτύχει εντυπωσιακή απόδοση σε διάφορες εφαρμογές, συμπεριλαμβανομένης της αναγνώρισης προσώπων, των αυτό-οδηγούμενων οχημάτων και της ανάλυσης ιατρικών εικόνων.

Συνολικά, τα CNN αποτελούν αποτελεσματικές τεχνικές βαθιάς μάθησης για εργασίες ανάλυσης εικόνων, με υψηλή αποδοτικότητα σε συγκριτικά με τις παραδοσιακές μεθόδους.

Λέξεις Κλειδιά: Συνελικτικά νευρωνικά δίκτυα, Μηχανική μάθηση, Αναγνώριση Προτύπων, ψηφιακή επεξεργασία εικόνας .

Summary

A convolutional neural network (CNN) is a type of deep learning model that is particularly effective for tasks involving images and visual data. It is inspired by the human visual system, where visual information is processed through the arrangement of neurons that are sensitive to specific visual stimuli.

The main idea behind CNNs is to apply filters (also known as convolutional kernels) to input images, which essentially scan the image for specific features. These features can be simple, like edges and gradients, or more complex, like shapes and patterns. The filters extract these features by performing mathematical operations such as convolution and pooling.

CNNs typically consist of several layers. The first layer is the input layer that receives the raw image data. Subsequent layers are composed of convolutional layers, which apply filters to the input and extract different features at each layer. These layers are followed by pooling layers, which reduce the spatial dimensions and make the representation more manageable. Finally, fully connected layers are used for classification or regression tasks.

One significant advantage of CNNs is their ability to learn hierarchical representations. Lower layers learn low-level features, such as edges and textures, while higher layers learn more abstract and complex features, such as object shapes. This hierarchical learning helps CNNs to automatically extract and understand relevant features from the input data.

CNNs are widely used in computer vision tasks, such as image classification, object detection, and image segmentation. They have achieved impressive performance in various applications, including facial recognition, self-driving cars, and medical image analysis.

Overall, CNNs are a powerful deep learning technique that revolutionized image analysis tasks, providing significant advancements in accuracy and efficiency compared to traditional methods.

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Στην σημερινή εποχή της πλήρους αυτοματοποίησης, αρκετές διεργασίες που χρειάζονταν ανθρώπινο παράγοντα, μειώνονται βαθμιαία, την περάτωση των οποίων διεκπεραιώνουν ρυθμισμένες από τον άνθρωπο μηχανές. Αυτές έχουν ανθρώπινες «ικανότητες» να αναγνωρίζουν ήχους, εικόνες κ.ά. Οι αλγόριθμοι (AI-Artificial Intelligence) μπορεί να αφορούν τη μαθησιακή διαδικασία, την αντίληψη γεγονότων, αναπαραστάσεων και εικόνων, την επίλυση προβλημάτων, την κατανόηση της γλώσσας και/ή τη λογική συλλογιστική (Tang et al, 2019).

Σε επιστημονικό επίπεδο τα παραπάνω εντάσσονται στο πλαίσιο της **Μηχανικής Μάθησης** (Machine Learning), χρονολογούμενη από το 1943, με το μαθηματικό μοντέλο του νευρικού δικτύου (Neural Network - NN), η πατρότητα του οποίου ανήκει στον McCulloch. Επιπλέον συνδέεται και με το ευρύτερο πεδίο της Μη Συμβολικής Τεχνητής Νοημοσύνης (Non-Symbolic Artificial Intelligence).

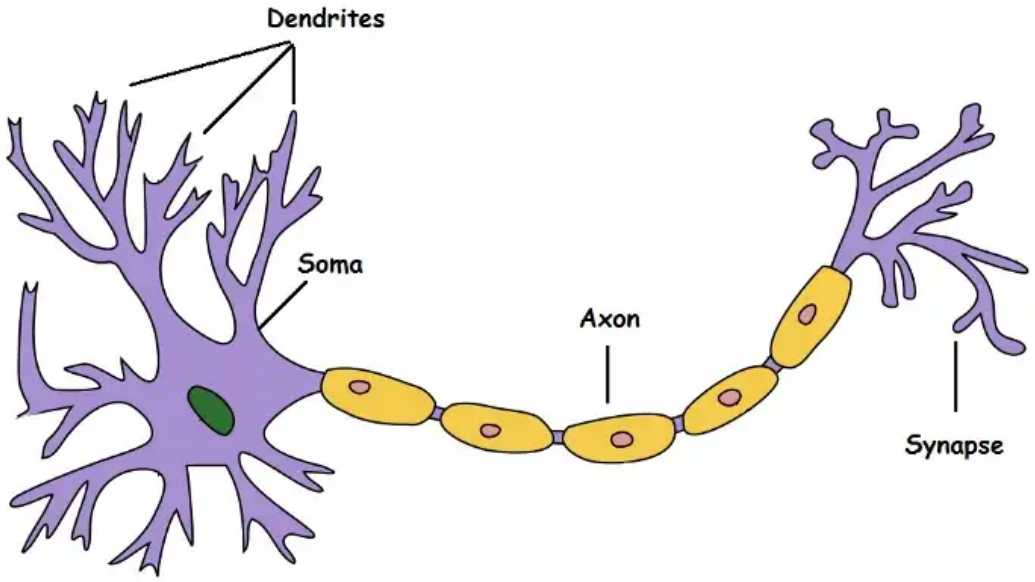
Η μηχανική μάθηση είναι διακριτός κλάδος της πληροφορικής και στοχεύει στη δημιουργία αλγορίθμων που μαθαίνουν, διαβάζουν και ερμηνεύουν το περιβάλλον τους μέσα από τα δεδομένα, χωρίς τη χρήση συμβολικών κανόνων. Πρόκειται ουσιαστικά για εκμάθηση μέσα από δεδομένα που επιτρέπουν τη λήψη αποφάσεων, την εκτέλεση προβλέψεων και τη μάθηση από προηγούμενους υπολογισμούς.

Τα μοντέλα μηχανικής μάθησης διακρίνονται στα ταξινόμησης, παλινδρόμησης και μάθησης. Τα πρώτα χρησιμοποιούνται για την επίλυση ταξινομήσεων ή προβλημάτων ταξινόμησης, τα δεύτερα για την εκτέλεση προβλέψεων και τα τρίτα χρησιμεύουν στην κατανόηση και επεξεργασία της φυσικής γλώσσας, την αναγνώριση ομιλίας (Speech Recognition), τη μηχανική όραση (Computer Vision), την αναγνώριση ηλεκτρονικών επιθέσεων στο διαδίκτυο (Cyberattack detection), την επεξεργασία φυσικής γλώσσας (Natural Language Processing), τις μηχανές αναζήτησης (Search Engines) κ.λπ. Η μηχανική μάθηση μπορεί να κατηγοριοποιηθεί σε εποπτευόμενη, μη εποπτευόμενη μάθηση και μάθηση ενίσχυσης (Reinforcement Learning - RL) (Nayak et al, 2021).

1.1 Νευρωνικά Δίκτυα

Ουσιαστικά πρόκειται για ένα επιστημονικό που προσπαθεί να μοντελοποιήσει με μαθηματικό τρόπο τη δομή του νευρικού συστήματος με στόχο τη λύση πολύπλοκων προβλημάτων. Κάθε νευρώνας στο συνελικτικό στάδιο συνδέεται μονάχα με νευρώνες του προηγούμενου επίπεδου. Στο πλαίσιο του συνελικτικού νευρικού δικτύου επιτυγχάνεται η εκμάθηση των ιεραρχιών χαρακτηριστικών με σημαντικό αριθμό δεδομένων και με χρήσεις όπως η κοινή χρήση παραμέτρων, η ισοδύναμη αναπαράσταση και αραιές αλληλεπιδράσεις. Το συνελικτικό νευρικό δίκτυο επιτυγχάνει εξαιρετική απόδοση στις εφαρμογές επεξεργασίας απεικόνισης (David et al, 2019).

Το νευρικό κύτταρο ή αλλιώς ο νευρώνας (Εικόνα 1) αποτελεί το κύριο δομικό μέρος του εγκέφαλου σε όλα τα έμβια οντά. Είναι ένα σχετικά ευμεγέθες κύτταρο του οποίου η ανατομία είναι: οι δενδρίτες, το σώμα, ο άξονας και οι συνάψεις που ενώνονται με αλλά νευρικά κύτταρα προκειμένου να δημιουργείται το νευρωνικό δίκτυο.



Εικόνα 1 Απλοποιημένο διάγραμμα ενός βιολογικού νευρώνα.

1.2 Ανατομία Νευρώνα

Δενδρίτες (Dendrites): Είναι υποδοχείς εισόδου του ηλεκτρικού σήματος που δέχεται το κύτταρο από τα αλλά κύτταρα.

Σώμα (Soma): Επεξεργάζονται την πληροφορία.

Άξονας(Axon): Είναι η έξοδος του ηλεκτρικού σήματος ως προς τα αλλά κύτταρα, αφού έχει δεχτεί την επεξεργασία από το σώμα. Μοιάζει με χορδή και το μήκος του κυμαίνεται από μερικά χιλιοστά έως ένα μετρό.

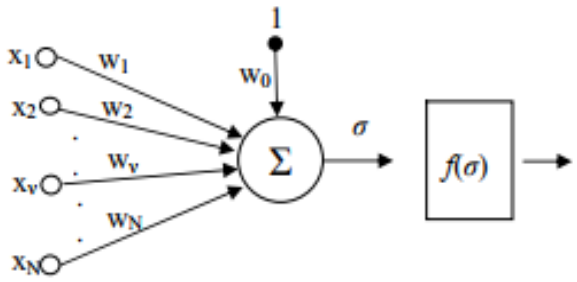
Συνάψεις(Synapse) : Είναι τα σημεία επαφής/σύνδεσης μεταξύ του άξονα ενός νευρώνα και των δενδριτων άλλων νευρώνων. Πρόκειται για κύστες με κύρια συστατικά της δομής τους τα ιόντα Νάτριου(Na^+) και Κάλιου(K^+). Παράμετροι που επηρεάζουν την διευκόλυνση μεταφοράς του ηλεκτρικού σήματος στην σύναψη είναι το πλάτος της σύναψης , το εύρος από τους δενδριτες και η πυκνότητα των νευροδιαβιβαστών (Διαμαντάρας, 2007).

Ο ανθρώπινος εγκέφαλος περιέχει σχεδόν 100 δισεκατομμύρια νευρώνες και 100 τρισεκατομμύρια συνάψεις. Η πληροφορία στον νευρώνα εισέρχεται με την μορφή ηλεκτρικού σήματος δια των δενδριτών. Στο σώμα γίνεται η σχετική επεξεργασία της πληροφορίας και κατόπιν μέσα από τον άξονα και μέσα από τις συνάψεις μεταβιβάζονται σε άλλους νευρώνες. Το μέρος του ηλεκτρικού σήματος που θα περάσει από τις συνάψεις στους δενδρίτες ονομάζεται συναπτικό βάρος. Έχουμε 2 κατηγορίες συνάψεων, τις **ενισχυτικές (excitatory)** και τις **ανασταλτικές (inhibitory).** Οι κατηγορίες αυτές διαχωρίζονται ανάλογα με το εάν το σήμα πυροδότη τον νευρώνα καθίσταται ικανό όχι να προκαλέσει παλμούς (Αργυράκης, 2001). Η λειτουργεία της σύναψης βασίζεται σε ουσίες που λέγονται νευροδιαβιβαστές (ντοπαμίνη, σεροτονίνη, νοραδρεναλίνη κ.α.)

1.3 Τεχνητά Νευρωνικά Δίκτυα

Μια ευρύτερη διάσταση της μηχανικής μάθησης είναι τα **Τεχνικά Νευρωνικά Δίκτυα,** τα οποία είναι μοντέλα εμπνευσμένα από το ανθρώπινο κεντρικό νευρικό σύστημα. Μια ομάδα αλγορίθμων νευρωνικών δικτύων που χρησιμοποιούνται κατά κόρον για την αναγνώριση εικόνων είναι τα λεγόμενα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks-CNN) όπου η πράξη της συνέλιξης είναι δυνατή σε δύο ή και περισσότερες διαστάσεις.

Ένα από τα πρώτα μοντέλα νευρωνικών δικτυών είναι και ο Perceptron (Αντίληπτρο), αποτελούμενο από έναν νευρώνα και δεδομένα εισόδου. Έχει τη δυνατότητα υποδοχής πολλών δεδομένων και παραγωγής μιας εξόδου. Το πολυεπίπεδο perceptron (MLP) είναι ένα μοντέλο που αποτελείται από διαδοχικά επίπεδα νευρώνων τεχνητού νευρικού δικτύου (Artificial Neural Network - Artificial Neural Network). Ενδείκνυται για τη χρήση στην εποπτευόμενη μάθηση, μη εποπτευόμενη μάθηση και την ενισχυτική μάθηση (Xiao et al, 2020).



Οπου **x** = [x_1,x_2,….,x_N] διανυσμα εισοδου

N=1,2,…..,n το πληθος

w = [w_1,w_2,….,w_N] τα συναπτικα βαροι

W_0=σταθ bias καθοριζει αν θα ενεργοποιηθει ο νευρωνας.

$$\sigma = \sum_{i=1}^{n} w_i^T x_i + w_0$$

Το αποτέλεσμα μετά την άθροιση λαμβάνει τις παρακάτω τιμές όπου επηρεάζεται από την σταθερά bias.

$$\begin{cases} \sigma > 0, & \alpha\nu \sum_{i=1}^{N} w_i^T x_i > w_0 \\ \sigma = 0, & \alpha\nu \sum_{i=1}^{N} w_i^T x_i = w_0 \\ \sigma < 0, & \alpha\nu \sum_{i=1}^{N} w_i^T x_i < w_0 \end{cases}$$

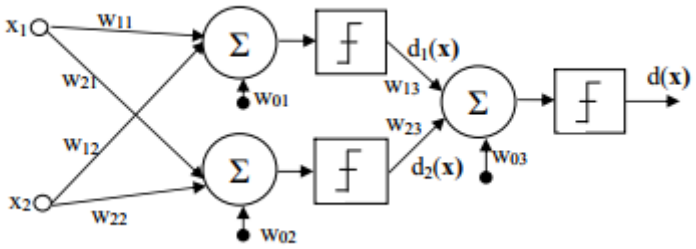
Η τιμή σ γίνεται όρισμα (είσοδος) μια συνάρτηση f(σ) που ονομάζεται συνάρτηση ενεργοποίησης (activation function). Υπάρχουν δυο παραλλαγές για την συνάρτηση ενεργοποίηση για το δίκτυο perceptron:

$$f(\sigma) = \begin{cases} 1, & \alpha\nu\ \sigma > 0 \\ 0, & \alpha\nu\ \sigma \leq 0 \end{cases}$$

ή

$$f(\sigma) = \begin{cases} 1, & \alpha\nu\ \sigma > 0 \\ -1, & \alpha\nu\ \sigma \leq 0 \end{cases}$$

Πρόκειται για μη-γραμμικές συναρτήσεις. Υπάρχουν κι άλλες συνάρτησεις ενεργοποίησης πού θα αναφερθούν σε επόμενα κεφάλαια.
Ο perceptron ήταν το πρώτο βήμα για την επίλυση γραμμικών προβλημάτων. Η ανάγκη επίλυσης σύνθετων προβλημάτων οδήγησε στην ιδέα των multi layer perceptron (MLP) με διακριτή είσοδο, κρυφά στρώματα και στρώμα εξόδου (Xiao et al, 2020).



1.4 Εκπαίδευση

Η εκπαίδευση των τεχνητών νευρωνικών δικτύων αφορά τη διαδικασία τροποποίησης των τιμών των βαρών. Από τη λήψη της πληροφορίας μεταβαίνουμε στην αξιοποίηση αυτής και την εξαγωγή συμπερασμάτων. Στα τεχνητά νευρωνικά δίκτυα χρησιμοποιούνται συναρτήσεις κόστους και οι επαναληπτικές διαδικασίες αποσκοπούν στην ελαχιστοποίηση του σφάλματος της προηγούμενης εξόδου από την επιθυμητη (Haykin, 1999).

1.5 Μέθοδοι Εκπαίδευσης

Η εκπαίδευση ενός νευρωνικου δικτύου, δηλαδή οι αλλαγές πάνω στα βάρη πραγματοποιούνται με τρεις μεθόδους:
Α) Τη μάθηση με επίβλεψη (supervised learning) όπου οι είσοδοι γίνονται δεκτοί από το δίκτυο και οι έξοδοι συγκρίνονται με τα επιθυμητά αποτελέσματα (δλδ. μια σωστή έξοδος του δικτύου για κάθε είσοδο). Τα βάρη στην αρχή λαμβάνουν τυχαίες τιμές και κατά την διάρκεια της εκπαίδευσης το δίκτυο διορθώνει τις τιμές αυτές ανάλογα με το σφάλμα που παίρνουμε. Εδώ ρυθμίζονται τα βάρη w και οι πολώσεις b, προκειμένου οι έξοδοι του δικτύου να αναδεικνύουν επιθυμητά αποτελέσματα.

Β) Τη μάθηση χωρίς επίβλεψη (unsupervised learning) με τις τιμές των βαρών w και των πολώσεων b να τροποποιούνται σε σχέση με τις εισόδους. Εδώ οι αλγόριθμοι εκτελούν τη διαδικασία κατηγοριοποίησης από ένα πρότυπο εισόδου σε ένα αριθμό ομάδων. Σε γενικές γραμμές δεν χρησιμοποιείται και τόσο συχνά λόγω της πολυπλοκότητας του και της μη εύκολης κατανόησής τους (όπως λέμε στην διάλεκτό του είναι BLACK BOX). Ίσως ο πιο γνωστός τέτοιάς αλγόριθμος είναι ο K-means.

Γ) Η ενισχυτική μάθηση (reinforcement learning) είναι μια μέθοδος όπου δίδεται ένας μόνο βαθμός (score) ως μέτρο εκτίμησης της απόδοσης του δικτύου για μια ακολουθία εισόδων. Πρόκειται για τη λιγότερο πρακτική μέθοδο, κατάλληλη όμως για τον έλεγχο εφαρμογών συστημάτων (Haykin, 1999).

Ανάλογα με το πρόβλημα και τα δεδομένα που έχουμε χρησιμοποιούμε την αντίστοιχη μέθοδο. Όταν σταματάει η διαδικασία εκπαίδευσης, τότε πλέον δεν αλλάζουν τα βάρη διότι το σφάλμα είναι μηδέν (ιδεατό) ή τείνει στο μηδέν.

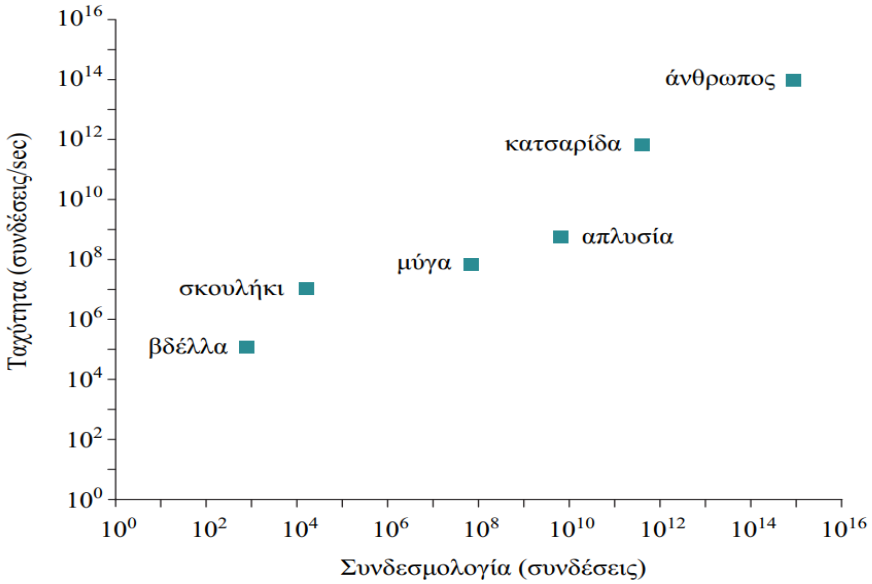
Η εκπαίδευση εξαρτάται από 4 παραμέτρους

1) Το Dataset
2) Το πλήθος των νευρώνων
3) Τη συνάρτηση ενεργοποίησης
4) Τη μέθοδο εκπαίδευσης

1.6 Σύγκριση του Βιολογικού με τον Τεχνητό Νευρώνα

Ο τεχνητός νευρώνας δέχεται ένα σήμα εισόδου (στο βιολογικό νευρώνα το κάνει ο δενδρίτης), το επεξεργάζεται η CPU- GPU (το αντίστοιχο σώμα), μεταφέρεται την έξοδο μέσω του δικτύου στους άλλους κόμβους (ο άξονας σε σύναψη με τον δενδρίτη άλλου νευρώνα). Ο ανθρώπινος εγκέφαλος διαχειρίζεται τεράστιο αριθμό νευρώνων και περίπλοκων συνάψεων ενώ ταχύτητα διάδοσης του σήματος στους τεχνητούς νευρώνες είναι πολύ μεγαλύτερη από ότι στους βιολογικούς. Η εκπαίδευση στον ανθρώπινο εγκέφαλο περατώνεται πολύ γρήγορα και σε συνεχή χρόνο ενώ στους υπολογιστές είναι χρονοβόρος διαδικασία και τελείται σε διακριτό χρόνο. Ο ανθρώπινος εγκέφαλος μπορεί να αντιλαμβάνεται, να μαθαίνει και να κατανοεί ταχύτατα, με τη μάθηση στα ΤΝΔ να χρειάζεται πολύ χρόνο. Ο ανθρώπινος εγκέφαλος τελεί τόσο σύγχρονη όσο και ασύγχρονη ενημέρωση των μονάδων του (σε συνεχή χρόνο), ενώ τα ΤΝΔ τελούν μόνο σύγχρονη ενημέρωση (σε διακριτό χρόνο) (Αργυράκης, 2001).

Σχηματικά, μια σύγκριση των δύο μορφών σε σχέση με έμβιους οργανισμούς καταγράφεται στο παρακάτω διάγραμμα με τα τεχνητά νευρωνικά δίκτυα μα έχουν ξεπεράσει την πολυπλοκότητα ενός σκουληκιού και πλησιάζουν να φτάσουν της μύγας.



Εικόνα 2.Διάγραμμα της ταχύτητας(πόσο γρήγορα μπορούν να γίνει οι υπολογισμοί) ως προς τον συνολικό αριθμό συνάψεων σε διάφορα έμβια οντά.

1.7 Σύνοψη Λειτουργίας Νευρώνα

Η λειτουργία του εκάστοτε νευρώνα διαλαμβάνει δύο εναλλακτικές καταστάσεις, μια ενεργό και μια μη ενεργό κατάσταση (Panksepp, 2004). Σημειώνεται ότι κάθε άφιξη οιουδήποτε σήματος σε ένα νευρώνα, σε οποιαδήποτε στιγμή, αθροίζεται. Το άθροισμα αυτό των σημάτων μπορεί να προσεγγίζει ή να ξεπερνάει μια καθορισμένη τιμή με τον νευρώνα να ενεργοποιείται. Όταν το άθροισμα είναι μικρότερο της ορισμένης τιμής τότε ο νευρώνας είναι ανενεργός. ΄Οταν ο νευρώνας είναι ενεργός αποστέλλει ένα παλμό, επανέρχεται στην αρχική του κατάσταση και επαενεργοποιείται (Αργυράκης, 2001).

Σχηματικά, ένα στρώμα εισόδου αποτελείται από μια ομάδα νευρώνων που δέχονται το σήμα εισόδου, χωρίς να το επεξεργάζονται αλλά απλώς το διαβιβάζουν. Στη συνέχεια, υπάρχουν εσωτερικά στρώματα με έναν αριθμό νευρώνων, οι οποίοι λαμβάνουν το σήμα από το πρώτο στρώμα, το επεξεργάζονται και το προωθούν προς την έξοδο. Τέλος, υπάρχει ένα στρώμα εξόδου με έναν αριθμό νευρώνων που λαμβάνουν σήμα από τα εσωτερικά στρώματα, χωρίς να το επεξεργάζονται, αλλά απλώς το δίνουν ως έξοδο του δικτύου. Συνήθως δεν υπάρχει ένας γενικός κανόνας για τον αριθμό των εσωτερικών στρωμάτων και των νευρώνων σε κάθε στρώμα (εισόδου, εξόδου, εσωτερικού). Αυτό εξαρτάται από το συγκεκριμένο πρόβλημα που επιλύεται. Οι νευρώνες των διαφορετικών στρωμάτων συνδέονται μεταξύ τους αλλά δεν υπάρχει ένας γενικός κανόνας για τον τρόπο που είναι συνδεδεμένοι μεταξύ τους. Κατά τη διαδικασία εκπαίδευσης, το σήμα φτάνει σε κάθε νευρώνα της εισαγωγικής στρώσης. Κάθε σύναψη πολλαπλασιάζει το σήμα εισόδου με το αντίστοιχο βάρος. Τα γινόμενα συγκεντρώνονται σε κάθε νευρώνα και υπολογίζεται ο συνολικός αθροιστής S. Αντί να συγκριθεί το άθροισμα με ένα κατώφλι (θ), χρησιμοποιείται μια συνάρτηση ενεργοποίησης (συνάρτηση αναπαράστασης) που υπολογίζει την τιμή της εξόδου. Αυτή η συνάρτηση είναι χαρακτηριστική του δικτύου και χρησιμοποιείται για να υπολογίσει ποια θα είναι η τιμή της εξόδου. Συνήθως, αυτή η τιμή συγκρίνεται με την αναμενόμενη τιμή για να υπολογιστεί το σφάλμα, το οποίο χρησιμοποιείται για να αποφασιστεί αν θα αλλάξουν τα βάρη του δικτύου.

Η διαδικασία αυτή αποτελεί ένα πέρασμα, είσοδος-έξοδος-είσοδος, και συνοψίζοντας γίνονται τα εξής βήματα:

- Παίρνουμε ένα πρότυπο από τα πολλά. Το εισάγουμε στο επίπεδο εισόδου.
- Υπολογίζουμε την έξοδο.

- Το προωθούμε με τον ίδιο τρόπο σε ολα τα επίπεδα μέχρι το τελικό επίπεδο εξόδου.
- Υπολογίζουμε το σφάλμα.
- Αλλάζουμε τα βάρη, επιστρέφοντας από την έξοδο προς την είσοδο, ένα-ένα, και επίπεδο-προς-επίπεδο.
- Προχωρούμε στο επόμενο πρότυπο.

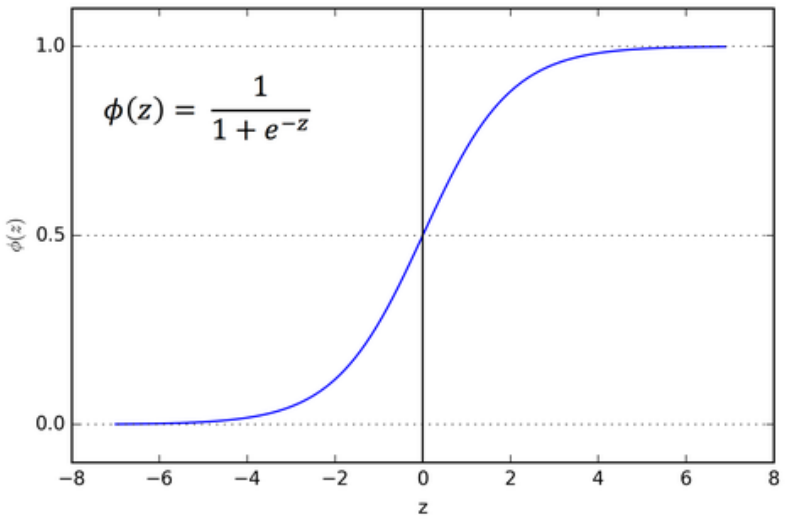
Μετά το τέλος μιας εποχής επαναλαμβάνουμε την διαδικασία για πολλές εποχές όσες χρειάζεται, έως ότου το σφάλμα να είναι αρκετά μικρό (Αργυράκης, 2001).

1.8 Συναρτήσεις ενεργοποίησης (Activation Functions)

Η συνάρτηση ενεργοποίησης, η οποία απαντάται στη βιβλιογραφία και ως συνάρτηση μεταφοράς νευρώνων, είναι μια μη γραμμική συνάρτηση που εφαρμόζεται σε έναν νευρώνα ή σε ένα στρώμα νευρώνων ενός τεχνητού νευρωνικού δικτύου. Ο κύριος στόχος κάθε νευρωνικού δικτύου είναι να μετασχηματίσει τα μη γραμμικά διαχωρίσιμα δεδομένα εισόδου σε πιο γραμμικώς διαχωρίσιμα αφηρημένα χαρακτηριστικά, χρησιμοποιώντας την ιεραρχία των στρωμάτων. Αυτή η διαδικασία επιτυγχάνεται μέσω συνδυασμών γραμμικών και μη γραμμικών συναρτήσεων στα στρώματα του δικτύου. Οι συναρτήσεις ενεργοποίησης αποτελούν ένα κρίσιμο στοιχείο των νευρωνικών δικτύων και επιτρέπουν την εισαγωγή μη γραμμικότητας στο εκάστοτε νευρωνικό μοντέλο. Αυτές οι συναρτήσεις χρησιμοποιούν ως είσοδο τη συνολική επιρροή ενός νευρώνα και εφαρμόζουν μια μη γραμμική μετασχηματιστική λειτουργία προκειμένου να συστήσουν την έξοδο του νευρώνα. Οι συναρτήσεις ενεργοποίησης πρέπει να έχουν ορισμένες ιδιότητες. Πρώτον, δεν πρέπει να αυξάνουν σημαντικά την πολυπλοκότητα του μοντέλου και δεύτερον, πρέπει να διατηρούν τη βαθμιδωτή ροή (gradient flow) κατά την εκπαίδευση του μοντέλου, διασφαλίζοντας ομαλή διάδοση των κλίσεων στα διάφορα στρώματα. Υπάρχουν πολλά είδη συναρτήσεων ενεργοποίησης. Η επιλογή της συνάρτησης ενεργοποίησης εξαρτάται από τον τύπο του προβλήματος που επιλύεται και τη φύση των δεδομένων. Οι κυριότερες συναρτήσεις ενεργοποίησης περιλαμβάνουν την σιγμοειδή συνάρτηση, τη συνάρτηση ReLU και τη συνάρτηση tanh (Hagan, Demuth and Beale, 1996).

1.9 Σιγμοειδής (Signmoid) Συνάρτηση

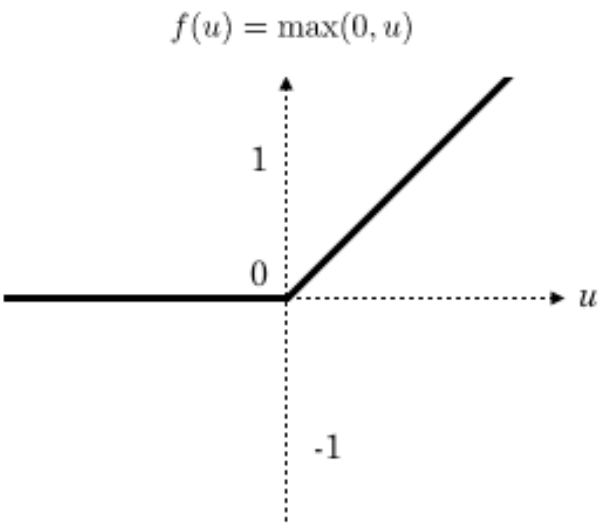
Ως σιγμοειδής (Sigmoid) αναφέρεται και αναγνωρίζεται η συνάρτηση που είναι κυρτή, μη γραμμική (non linear) συνάρτηση και παράλληλα λαμβάνει έναν αριθμό και τον μετασχηματίζει σε τιμή μεταξύ 0 και 1. Χρησιμοποιείται κυρίως σε προβλήματα δυαδικής ταξινόμησης και ως συνάρτηση ενεργοποίησης στο τελευταίο στρώμα εξόδου για πιθανότητες (Hagan, Demuth and Beale, 1996).



Εικόνα 3. Διάγραμμα σιγμοειδής συνάρτησης

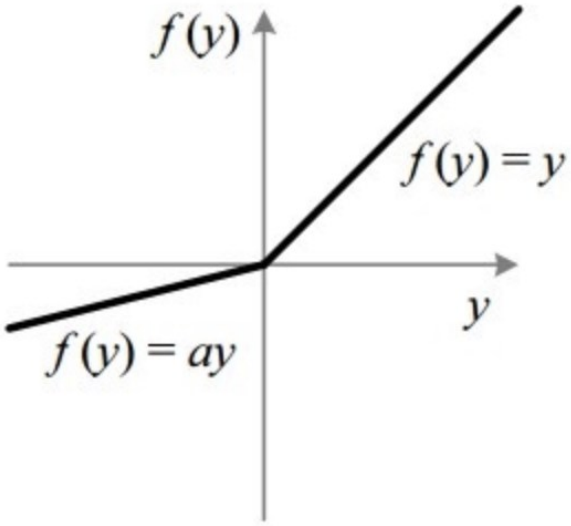
1.10 Rectified Linear Unit (ReLU)

Μια από τις πιο δημοφιλείς συναρτήσεις ενεργοποίησης που έχει γνωρίσει μεγάλη ανάπτυξη τα τελευταία χρόνια είναι η Rectified Linear Unit (ReLU). Η συνάρτηση ReLU επιστρέφει την είσοδό της στην περίπτωση που είναι θετική, διαφορετικά επιστρέφει την ένδειξη μηδέν και είναι γρήγορη στον υπολογισμό της. Ένα μειονέκτημα της ReLU είναι ότι κατά τη διάρκεια της εκπαίδευσης, τα βάρη του νευρώνα ενημερώνονται έτσι ώστε να μπορεί να μην ενεργοποιηθεί ποτέ, δηλαδή να μην εκπέμπει σήμα εξόδου (Hagan, Demuth and Beale, 1996).



Εικόνα 4. Διάγραμμα Relu συνάρτησης

Προς αντιμετώπιση του προαναφερθέντος προβλήματος υπάρχει η Leaky ReLU, όπου αντί να μηδενίζεται έχει μια σύγκλιση προς το μηδέν.

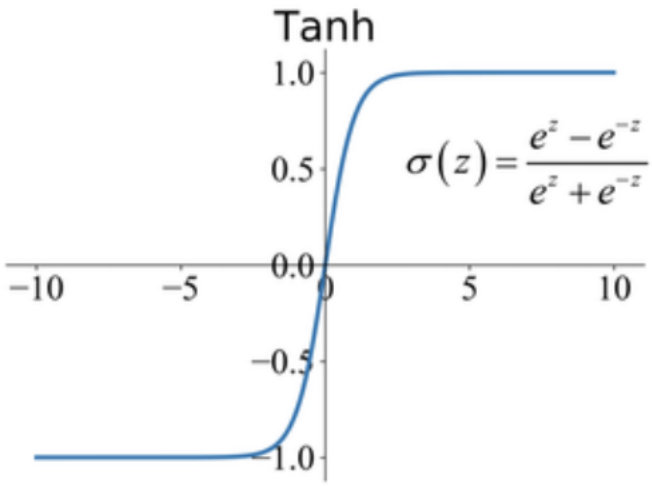


Εικόνα 5. Διάγραμμα Leaky Relu συνάρτησης

Οπού α η κλίση για αρνητικές τιμές και η σταθερά α ορίζεται σαν υπερπαράμετρος του νευρωνικού δικτύου.

1.11 Υπερβολική Εφαπτομένη(Tanh)

Ως υπερβολική εφαπτόμενη συνάρτηση είναι εκείνη η οποία δέχεται έναν αριθμό και τον μετασχηματίζει μεταξύ του -1 και 1. Αυτό γίνεται με παρόμοιο τρόπο με τη σιγμοειδή συνάρτηση, αλλά με κέντρο το μηδέν. Χρησιμοποιείται συχνά σε μεσαία στρώματα νευρωνικών δικτύων (Hagan, Demuth and Beale, 1996).



Εικόνα 6. Διάγραμμα Υπερβολική Εφαπτομένη συνάρτησης

1.12 Συναρτήσεις Απώλειας/Κόστους

Η απώλεια στη Μηχανική Μάθηση αναφέρεται στη διαφορά μεταξύ της προβλεπόμενης τιμής και της πραγματικής τιμής. Για την ποσοτικοποίηση αυτής της απώλειας κατά την εκπαίδευση, χρησιμοποιείται μια συνάρτηση γνωστή ως Συνάρτηση Απώλειας. Αυτές οι συναρτήσεις χρησιμοποιούνται σε προβλήματα υπο-επίβλεψης μάθησης, όπως η παλινδρόμιση (regression) και η λογιστική παλινδρόμιση (logistic regression). Οι όροι "συνάρτηση κόστους" και "συνάρτηση απώλειας" είναι αντίστοιχοι και χρησιμοποιούνται συχνά. Η συνάρτηση απώλειας αναφέρεται στο σφάλμα για ένα μόνο παράδειγμα εκπαίδευσης, ενώ η συνάρτηση κόστους αναφέρεται στον μέσο όρο των συναρτήσεων απώλειας σε ολόκληρο το σύνολο των δεδομένων εκπαίδευσης (Hagan, Demuth and Beale, 1996).

Οι συναρτήσεις απώλειας στοχεύουν να εντοπίσουν τις λανθασμένες προβλέψεις του Νευρωνικό Δίκτυο κατά τη διάρκεια της εκπαίδευσής του. Ο σκοπός είναι να αποτρέψουν το μοντέλο από την παραγωγή ανεπιθύμητων αποτελεσμάτων

$$\boldsymbol{Error = f(target - output)}$$

Η πιο συχνά χρησιμοποιούμενη συνάρτηση ταξινόμησης σε προβλήματα αναγνώρισης πολλών κλάσεων είναι η συνάρτηση Softmax:

$$\rho(y = y_i|x_i) = \frac{e^{W_c^T \cdot x_i}}{\sum_m e^{W_m^T \cdot x_i}}$$

Η συνάρτηση Softmax είναι μια συνάρτηση που συνδέεται συχνά με τα τελευταία στρώματα ενός νευρωνικού δικτύου, που είναι συνήθως ένα στρώμα εξόδου. Η Softmax μετατρέπει μια αριθμητική είσοδο σε ένα διάνυσμα πιθανοτήτων, όπου κάθε στοιχείο του διανύσματος αναπαριστά την πιθανότητα εμφάνισης μιας συγκεκριμένης κατηγορίας. Η Softmax συνάρτηση υπολογίζει την έξοδο ως εξής: παίρνει τον εκθετικό αριθμό της εισόδου και τους κανονικοποιεί ως πιθανότητες, διαιρώντας τον καθένα αριθμό με το άθροισμα όλων των εκθετικών αριθμών. Αυτό δημιουργεί ένα πιθανοτικό διάνυσμα, το οποίο έχει την ιδιότητα ότι οι τιμές του αθροίζονται σε 1. Η Softmax εξασφαλίζει ότι οι πιθανότητες που παράγονται είναι μη αρνητικές και αθροίζονται σε 1, καθιστώντας την, κατάλληλη για αυτόν τον σκοπό (Hagan, Demuth and Beale, 1996).

Η συνάρτηση κόστους που συνδυάζεται με την Softmax είναι η συνάρτηση *cross-entropy L(**W**)*:

$$L(\boldsymbol{W}) = -\sum_i \log\big(\rho(y = y_i|x_i)\big)$$

Όπου ρ είναι η δεσμευμένη πιθανότητα ένα δείγμα να ανήκει στην κλάση y_i δεδομένου του διανύσματος x_i.

1.13 Adam Optimizer

Ο αλγόριθμος βελτιστοποίησης Adam (Adaptive Moment Estimation) είναι ένας δημοφιλής και αποδοτικός αλγόριθμος βελτιστοποίησης που σχεδιάστηκε για την

εκπαίδευση νευρωνικών δικτύων. Αναπτύχθηκε από τους D. P. Kingma και J. Ba και περιγράφεται στη δημοσίευσή τους "Adam: A Method for Stochastic Optimization" που δημοσιεύτηκε το 2014.

Ένα από τα κύρια πλεονεκτήματα του βελτιστοποιητή Adam είναι η προσαρμοστική ρυθμιζόμενη τιμή μάθησης. Σε αντίθεση με τους παραδοσιακούς αλγορίθμους που διατηρούν μια μόνιμη, σταθερή τιμή μάθησης για όλες τις ενημερώσεις βάρους, ο βελτιστοποιητής Adam διατηρεί μια ξεχωριστή τιμή μάθησης για κάθε βάρος και ενημερώνει αυτές τις τιμές βάσει του πόσο γρήγορα αλλάζουν οι κλίσεις. Αυτή η δυνατότητα επιτρέπει στον αλγόριθμο να εκτελείται πιο γρήγορα και να επιτύχει καλύτερη απόδοση σε μια ευρεία γκάμα εργασιών μηχανικής μάθησης.

ης. Ας εξετάσουμε τα βασικά χαρακτηριστικά του Adam. Οι παράμετροι m_t και υ_t είναι ή μέση τιμή και η διακύμανση της κλίσης μάθησης. Όταν τα m_t και υ_t αρχικοποιούνται σαν μηδενικά διανύσματα τα β_1 και β_2 τείνουν στη μονάδα.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$\upsilon_t = \beta_2 \upsilon_{t-1} + (1-\beta_2) g_t^2$$

Με τον υπολογισμό των εκτιμήσεων:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{\upsilon}_t = \frac{\upsilon_t}{1-\beta_2^t}$$

υ πολογίζεται και ο κανόνας ενημέρωσης Adam ο οποίος προσαρμόζει το ρυθμό μάθησης η σε κάθε βήμα t για κάθε παράμετρο θ_t ως εξής:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{\upsilon}_t} + \epsilon} \cdot \hat{m}_t$$

όπου η είναι ο ρυθμός μάθησης και ϵ είναι ένας μικρός θετικός όρος που προστίθεται στον παρονομαστή για να αποφευχθεί η διαίρεση με μηδέν.

1.14 Διόρθωση βαρών – Back error propagation

Ο αλγόριθμος back error propagation (αντίστροφης διάδοσης) εμφανίστηκε για πρώτη φορά κατά την δεκαετία του 1960 αλλά γνώρισε τεράστια αποδοχή όταν το 1989 οι David Rumelhart, Geoffrey Hinton και Ronald Williams κατάφεραν να

εισάγουν με επιτυχία δικό τους μοντέλο στον τομέα της εκπαίδευσης των νευρωνικών δικτύων. Σήμερα ο αλγόριθμος αυτός χρησιμοποιείται κατά κόρον για την εκπαίδευση μεγάλων νευρωνικών δικτύων.

Ο σκοπός του back error propagation είναι να ρυθμίσει τα βάρη του νευρωνικού δικτύου, έτσι ώστε να μειώσει το σφάλμα μεταξύ των προβλέψεων του δικτύου και των πραγματικών τιμών. Η κεντρική ιδέα είναι αρκετά απλή: το δίκτυο ξεκινά την διαδικασία μάθησης από τυχαίες τιμές των βαρών του. Εάν δώσει λάθος απάντηση (που είναι και το πιο πιθανό) τότε τα βάρη διορθώνονται έτσι ώστε το λάθος να γίνει μικρότερο. Η ίδια διαδικασία επαναλαμβάνεται πολλές φορές έτσι ώστε σταδιακά το λάθος ελαττώνεται, μέχρις ότου γίνει πολύ μικρό και ανεκτό. Στο σημείο αυτό δεχόμαστε ότι το δίκτυο έχει μάθει τα παραδείγματα που του διδάξαμε με την ακρίβεια που θέλαμε να μάθει. Ο αλγόριθμος λειτουργεί με την εφαρμογή δύο κύριων βημάτων: τον αλγορίθμο διάδοσης (forward propagation) και το backward propagation. Σχηματικά, στο τεχνητό νευρωνικό δίκτυο χρησιμοποιούμε δύο διανύσματα. Το πρώτο διάνυσμα είναι εκείνο των δεδομένων που θα κινητοποιήσει το δίκτυο και το δεύτερο διάνυσμα εκείνο με τις επιθυμητές τιμές που αναμένουμε από το δίκτυο να δείξει στις εξόδους (Βλαχάβας και συν., 2006).

Η διαδικασία της εκπαίδευσης τελείται σε επαναλήψεις μέχρι οι τιμές των βαρών του δικτύου να αποφέρουν συγκεκριμένες τιμές και το σφάλμα να είναι το ελάχιστο δυνατό βάσει του δυνάμενου υπολογισμού. Η διαδικασία της εκπαίδευσης γίνεται με δύο δυνατούς τρόπους. Ο πρώτος ονομάζεται εκπαίδευση ανά ομάδα προτύπων (batch training) και ο δεύτερος εκπαίδευση ανά πρότυπο (stochastic/on-line training). Στην πρώτη τα βάρη αλλάζουν αφού κινηθούν όλα τα πρότυπα στο δίκτυο. Με τον τρόπο αυτό εκτιμάται το ολικό μέσο τετραγωνικό σφάλμα για όλα τα πρότυπα (Βλαχάβας και συν., 2006).

Ο αλγορίθμος forward propagation στο δίκτυο υπολογίζει τις εξόδους των νευρώνων σε κάθε στρώμα του δικτύου, ξεκινώντας από την είσοδο και προχωρώντας προς την έξοδο. Κατά τη διάρκεια αυτής της διαδικασίας, το δίκτυο αποθηκεύει τις ενδιάμεσες τιμές και στη συνέχεια, κατά την εφαρμογή του αλγορίθμου backward propagation ωθείται στον υπολογισμό των απαραίτητων ανανεώσεων των βαρών. Κατά τη διάρκεια του αλγορίθμου backward propagation, το σφάλμα υπολογίζεται με τη χρήση μιας συνάρτησης κόστους και διαδίδεται προς τα πίσω από το τελευταίο στρώμα προς τα προηγούμενα στρώματα του δικτύου. Αυτό γίνεται με τη χρήση του κανόνα της αλυσιδωτής παραγώγισης (gradient chain rule) για τον υπολογισμό των

μερικών παραγώγων. Οι παράγωγοι που υπολογίζονται μέσω του backward propagation χρησιμοποιούνται για την ενημέρωση των βαρών του δικτύου. (Βλαχάβας και συν., 2006).

Οι παράγωγοι που υπολογίζονται μέσω του Backpropagation χρησιμοποιούνται στη συνέχεια για την ενημέρωση των βαρών του δικτύου, με τη χρήση μιας μεθόδου βελτιστοποίησης, όπως η (gradient descent). Ο στόχος του αλγορίθμου Gradient Descent είναι να ελαχιστοποιήσει μια συνάρτηση κόστους (ή να μεγιστοποιήσει μια συνάρτηση κέρδους), προσαρμόζοντας τις παραμέτρους του μοντέλου. Ο αλγόριθμος Gradient Descent βασίζεται στην κλίση (gradient). Η κλίση είναι το διάνυσμα παραγώγων μιας συνάρτησης προς τις διάφορες κατευθύνσεις, στο χώρο των παραμέτρων. Ο αλγόριθμος εκμεταλλεύεται την πληροφορία που παρέχει ο κλίσης για να εντοπίσει την κατεύθυνση με τη μεγαλύτερη αύξηση ή μείωση της συνάρτησης κόστους και τείνει να ακολουθήσει αυτήν την κατεύθυνση για να προσαρμόσει τις παραμέτρους. Ο αλγόριθμος ξεκινάει από μια αρχική τιμή για τις παραμέτρους και επαναλαμβάνει τα εξής βήματα μέχρι να συγκλίνει σε μια ελάχιστη τιμή κόστους ή μια επιθυμητή ακρίβεια:

Αρχικά γίνεται ο υπολογισμός του κλίσης (gradient) της συνάρτησης κόστους ως προς τις παραμέτρους του μοντέλου. Ακολουθεί η ενημέρωση των παραμέτρων με βάση τον κλίση και έναν παράγοντα εκμάθησης (learning rate) που καθορίζει το μέγεθος του βήματος προς την κατεύθυνση της απόκλισης. Έπεται η επανάληψη των βημάτων 1 και 2 μέχρι να επιτευχθεί η σύγκλιση.

Ο προαναφερθείς αλγόριθμος Gradient Descent μπορεί να έχει διάφορες εκδοχές, όπως ο Gradient Descent με σταθερό learning rate, ο Stochastic Gradient Descent (SGD) που χρησιμοποιεί τυχαία δείγματα για τον υπολογισμό του κλίση και οι παραλλαγές του όπως οι Mini-Batch Gradient Descent και ο Adam. Κάθε εκδοχή έχει τις δικές της παραμέτρους και ρυθμίσεις που επηρεάζουν τη σύγκλιση και την ταχύτητα του αλγορίθμου. Ο αλγόριθμος backpropagation επαναλαμβάνεται για πολλά παραδείγματα εκπαίδευσης, με σκοπό τη σύγκλιση των βαρών προς μια βέλτιστη λύση και την εκπαίδευση του δικτύου για την επίτευξη ακριβών προβλέψεων ή αποφάσεων. (Βλαχάβας και συν., 2006).

Εφαρμογή της διόρθωσης των βαρών με τη μέθοδο οπισθοδιάδοσης του σφάλματος (back error propagation) σε MLP με συνάρτηση ενεργοποίησης τη σιγμοειδή και συνάρτηση απώλειας το μέσο τετραγωνικό σφάλμα.

Έστω I το πλήθος ζευγών από πίνακες εισόδου και των αντίστοιχων επιθυμητών πινάκων εξόδου με γνωστές τιμές, το σύνολο εκπαίδευσης S ορίζεται ως:

S={(x_i,y_i)/(x_i,y_i) ζεύγος με x_i πίνακα στήλης εισόδου και y_i τον αντίστοιχο επιθυμητό πίνακα στήλης εξόδου, i=1,…,I }.

Αν ο δείκτης ν αριθμεί τους νευρώνες του επιπέδου εξόδου Z, ν=1,…,N_Z και ο πίνακας εξόδου του ΝΔ είναι:

$$\mathbf{y}^z = [y_1^z, \ldots . y_v^z, \ldots \ldots y_{N_z}^z]^T$$

για συγκεκριμένο ζεύγος (x_i,y_i) ορίζουμε μία συνάρτηση Δ(i) με χρήση της Ευκλείδειας απόστασης σύμφωνα με την σχέση:

$$\Delta(i) = \frac{1}{2} D_E\left(\mathbf{y}^Z, \mathbf{y}_i\right)^2 = \frac{1}{2}\sum_{v=1}^{N_z}(y_v^Z - y_{vi})^2$$

Η σχέση Δ(i) είναι ένα άθροισμα τετραγωνικών σφαλμάτων μεταξύ της παραγόμενης εξόδου $\mathbf{y}^Z$ του ΝΔ όταν η είσοδος είναι $\mathbf{x}_i$ και της επιθυμητής εξόδου yi. Μπορούμε να ορίσουμε τώρα την συνάρτηση κόστους K(.) που θα έχει ανεξάρτητες μεταβλητές όλα τα $\mathbf{w}_v^\zeta$ για το συγκεκριμένο σύνολο εκπαίδευσης S σύμφωνα με την σχέση:

$$\mathrm{K}(w_v^\zeta, S) = \sum_{i=1}^{I}\Delta(\mathrm{i})$$

Αντιμετωπίζοντας την εύρεση ελάχιστης τιμής της K ως πρόβλημα βελτιστοποίησης (optimization) οι τιμές των wνζ μπορούν να εκτιμηθούν επαναληπτικά σύμφωνα με την σχέση:

$$\mathbf{w}_v^\zeta(t+1) = \mathbf{w}_v^\zeta(t) - \rho\frac{\partial \mathrm{K}}{\mathbf{w}_v^\zeta}\Big|_{\mathbf{w}_v^\zeta(t)} = \mathbf{w}_v^\zeta(t) - \rho\sum_{i=1}^{I}\frac{\partial\Delta(\mathrm{i})}{\mathbf{w}_v^\zeta}\Big|_{\mathbf{w}_v^\zeta(t)} \qquad (1)$$

Σύμφωνα με τον κανόνα παραγώγισης της αλυσίδας

$$\frac{\partial\Delta(i)}{\partial\mathbf{w}_v^\zeta} = \frac{\partial\Delta(i)}{\partial\sigma_v^\zeta}\cdot\frac{\partial\sigma_v^\zeta}{\partial\mathbf{w}_v^\zeta}$$

Οπου ο δεύτερος παράγοντα του γινομένου:

$$\frac{\partial \sigma_v^\zeta}{\partial \mathbf{w}_v^\zeta} = \frac{\partial}{\partial \mathbf{w}_v^\zeta}\left(\left(\mathbf{y}^{\zeta-1}\right)^{\mathrm{T}} \cdot \mathbf{w}_v^\zeta\right) = \frac{\partial}{\mathbf{w}_v^\zeta}\left(y_1^{\zeta-1} \cdot w_{1v}^\zeta + \quad \cdots \quad y_\mu^{\zeta-1} \cdot w_{\mu v}^\zeta + \quad \cdots \quad y_{\mathrm{N}_{\zeta-1}}^{\zeta-1} \cdot w_{\mathrm{N}_{\zeta-1}v}^\zeta + w_{0v}^\zeta\right) = \left(\mathbf{y}^{\zeta-1}\right)^{\mathrm{T}}$$

Για τον υπολογισμό του πρώτου παράγοντα θέτουμε:

$$\delta_v^\zeta = \frac{\partial \Delta(i)}{\partial \sigma_v^\zeta} \text{ και } \boldsymbol{\delta}^\zeta = \left[\delta_1^\zeta, \cdots, \delta_v^\zeta, \cdots, \delta_{\mathrm{N}_\zeta}^\zeta\right]^{\mathrm{T}}$$

Θα υπολογίσουμε πρώτα το δ_v^ζ για έναν νευρώνα ν του επιπέδου εξόδου (ζ=Z, $\mathrm{N}_\zeta=\mathrm{N}_Z$) και δείκτη αρίθμησης των νευρώνων μ=1,…,N_Z

$$\delta_v^Z = \frac{\partial \Delta(i)}{\partial \sigma_v^Z} = \frac{\partial}{\partial \sigma_v^Z}\left(\frac{1}{2}\sum_{\mu=1}^{\mathrm{N}_Z}(y_\mu^Z - y_{\mu i})^2\right) \text{με } y_\mu^Z = f(\sigma_\mu^Z) \Rightarrow$$

$$\delta_v^Z = \frac{1}{2}\sum_{\mu=1}^{\mathrm{N}_Z}\frac{\partial}{\partial \sigma_v^Z}(f(\sigma_\mu^Z) - y_{\mu i})^2 = \frac{2}{2}(f(\sigma_v^Z) - y_{vi}) f'(\sigma_v^Z) \Rightarrow$$

$$\delta_v^Z = (y_v^Z - y_{vi}) f'(\sigma_v^Z) \qquad (2)$$

Για τα κρυφά επίπεδα (0<ζ<Z) ο υπολογισμός του δ_v^ζ είναι περιπλοκότερος, επειδή δεν είναι δεδομένη η τιμή της εξόδου y_{vi}, κάθε νευρώνα. Ο υπολογισμός σε κάθε κρυφό επίπεδο θα βασισθεί στις τιμές διόρθωσης του επομένου του, αρχίζοντας από το επίπεδο που προηγείται του επιπέδου εξόδου και οδεύοντας διαδοχικά προς τα πίσω (*back-error propagation*). Συγκεκριμένα αν *ζ* ένα κρυφό επίπεδο, το επόμενό του θα είναι το *ζ+1*. Έστω ακόμη *μ* ένας μετρητής αρίθμησης των νευρώνων του *ζ* επιπέδου και *κ* ένας μετρητής αρίθμησης των νευρώνων του *ζ+1* επιπέδου. Η συνάρτηση *Δ*(*i*) εξαρτάται από τα $\boldsymbol{\sigma}^{\zeta+1}$ και κάθε $\boldsymbol{\sigma}^{\zeta+1}$ εξαρτάται από το $\boldsymbol{\sigma}_v^\zeta$ του ν-οστού νευρώνα του ζ επιπέδου. Σύμφωνα με τον κανόνα της αλυσιδωτής παραγώγισης:

$$\delta_{\nu}^{\zeta} = \frac{\partial \Delta_i}{\partial \sigma_{\nu}^{\zeta}} = \frac{\partial \Delta_i}{\partial \boldsymbol{\sigma}^{\zeta+1}} \frac{\partial \boldsymbol{\sigma}^{\zeta+1}}{\partial \sigma_{\nu}^{\zeta}}$$

$$\frac{\partial \Delta_i}{\partial \boldsymbol{\sigma}^{\zeta+1}} = \left[\frac{\partial \Delta_i}{\partial \sigma_1^{\zeta+1}}, \cdots \frac{\partial \Delta_i}{\partial \sigma_{\mu}^{\zeta+1}}, \cdots, \frac{\partial \Delta_i}{\partial \sigma_{\mathrm{N}_{\zeta}+1}^{\zeta+1}} \right] = \left[\delta_1^{\zeta+1}, \ldots, \delta_{\mu}^{\zeta+1}, \ldots, \delta_{\mathrm{N}_{\zeta+1}}^{\zeta+1} \right]$$

$$\boldsymbol{\sigma}^{\zeta+1} = \left(\mathbf{W}^{\zeta+1}\right)^{\mathrm{T}} \cdot f(\boldsymbol{\sigma}^{\zeta}) = \begin{bmatrix} \left(\mathbf{w}_1^{\zeta+1}\right)^{\mathrm{T}} \cdot f\left(\boldsymbol{\sigma}^{\zeta}\right) \\ \vdots \\ \left(\mathbf{w}_{\mu}^{\zeta+1}\right)^{\mathrm{T}} \cdot f\left(\boldsymbol{\sigma}^{\zeta}\right) \\ \vdots \\ \left(\mathbf{w}_{\mathrm{N}_{\zeta+1}}^{\zeta+1}\right)^{\mathrm{T}} \cdot f\left(\boldsymbol{\sigma}^{\zeta}\right) \end{bmatrix} = \begin{bmatrix} w_{11}^{\zeta+1} \cdot f\left(\sigma_1^{\zeta}\right) + \cdots + w_{\nu 1}^{\zeta+1} \cdot f\left(\sigma_{\nu}^{\zeta}\right) + \cdots + w_{\mathrm{N}_{\zeta}\mu}^{\zeta+1} \cdot f\left(\sigma_{\mathrm{N}_{\zeta}}^{\zeta}\right) \\ \vdots \\ w_{1\mu}^{\zeta+1} \cdot f\left(\sigma_1^{\zeta}\right) + \cdots + w_{\nu\mu}^{\zeta+1} \cdot f\left(\sigma_{\nu}^{\zeta}\right) + \cdots + w_{\mathrm{N}_{\zeta}\mu}^{\zeta+1} \cdot f\left(\sigma_{\mathrm{N}_{\zeta}}^{\zeta}\right) \\ \vdots \\ w_{1,\mathrm{N}_{\zeta+1}}^{\zeta+1} \cdot f\left(\sigma_1^{\zeta}\right) + \cdots + w_{\nu\mathrm{N}_{\zeta+1}}^{\zeta+1} \cdot f\left(\sigma_{\nu}^{\zeta}\right) + \cdots + w_{\mathrm{N}_{\zeta}\mathrm{N}_{\zeta}+1}^{\zeta+1} \cdot f\left(\sigma_{\mathrm{N}_{\zeta}}^{\zeta}\right) \end{bmatrix}$$

$$\frac{\partial \boldsymbol{\sigma}^{\zeta+1}}{\partial \sigma_{\nu}^{\zeta}} = \begin{bmatrix} w_{\nu 1}^{\zeta+1} \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \\ \vdots \\ w_{\nu\mu}^{\zeta+1} \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \\ \vdots \\ w_{\nu\mathrm{N}_{\zeta+1}}^{\zeta+1} \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \end{bmatrix}$$

$$\delta_{\nu}^{\zeta} = \frac{\partial \Delta_i}{\partial \sigma_{\nu}^{\zeta}} = \frac{\partial \Delta_i}{\partial \boldsymbol{\sigma}^{\zeta+1}} \frac{\partial \boldsymbol{\sigma}^{\zeta+1}}{\partial \sigma_{\nu}^{\zeta}} = \left[\delta_1^{\zeta+1}, \ldots, \delta_{\mu}^{\zeta+1}, \ldots, \delta_{\mathrm{N}_{\zeta+1}}^{\zeta+1} \right] \cdot \begin{bmatrix} w_{\nu 1}^{\zeta+1} \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \\ \vdots \\ w_{\nu\mu}^{\zeta+1} \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \\ \vdots \\ w_{\nu\mathrm{N}_{\zeta+1}}^{\zeta+1} \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \end{bmatrix} \Rightarrow$$

$$\delta_{\nu}^{\zeta} = \left[\sum_{\mu=1}^{\mathrm{N}_{\zeta+1}} \delta_{\mu}^{\zeta+1} \cdot w_{\nu\mu}^{\zeta+1} \right] \cdot f'\left(\sigma_{\nu}^{\zeta}\right) = \left(\mathbf{W}_{row_{\nu}}^{\zeta+1} \cdot \boldsymbol{\delta}^{\zeta+1} \right) \cdot f'\left(\sigma_{\nu}^{\zeta}\right) \qquad (3)$$

Από τις σχέσεις (1),(2) και (3) συνεπάγεται

$$\mathbf{w}_{\nu}^{\zeta}(t+1) = \mathbf{w}_{\nu}^{\zeta}(t) - \rho \sum_{\forall (x_i, y_i)} \delta_{\nu}^{\zeta} \cdot y^{\zeta-1}$$

(Στρουθόπουλος 2014).

ΚΕΦΑΛΑΙΟ 2

Βαθιά Μηχανική Μάθηση (Deep Learning)

2.1. Συνελικτικά Νευρωνικά Δίκτυα CNN

Το συνελικτικό νευρικό δίκτυο (Convolutional Neural Netwok - CNN) είναι μια ιδιαίτερα αρχιτεκτονική νευρικών δικτύων με εισδοχή ομαδοποιημένων στρωμάτων πριν από τη φάση της τροφοδοσίας εισόδου σε ένα ολικά συνδεδεμένο δίκτυο. Κάθε νευρώνας στο συνελικτικό επίπεδο συνδέεται με ορισμένους μόνο νευρώνες στο προηγούμενο παρακείμενο επίπεδο. Οι νευρώνες οργανώνονται και στοιβάζονται σε μορφή μήτρας με συγκεκριμένους χάρτες χαρακτηριστικών όπου σε κάθε χάρτη μοιράζονται τα ίδια βάρη. Σχετικά με το επίπεδο συγκέντρωσης, οι νευρώνες στους χάρτες χαρακτηριστικών οργανώνονται σε ομάδες προκειμένου να εκτιμήσουν τη μέση ή τη μέγιστη τιμή βάρους. Τα πρώτα CNNs παρουσιάστηκαν στα μέσα της δεκαετίας του '90 και χρησιμοποιήθηκαν για την αναγνώριση χειρόγραφων χαρακτήρων από τον Yann LeCun ο οποίος χρησιμοποίησε τη βάση δεδομένων MNIST, η οποία περιλαμβάνει χιλιάδες χειρόγραφα αριθμητικά ψηφία. Το 2012, τα CNNs γνώρισαν σημαντική ανάπτυξη μετά την εντυπωσιακή απόδοση του CNN που ονομάστηκε AlexNet εκπαιδευμένο στο σύνολο δεδομένων ImageNet.

Τα CNNs αντλούν έμπνευση από τη δομή και λειτουργία του οπτικού νευρικού συστήματος των θηλαστικών. Η δύναμή τους στην ανάλυση εικόνων προέρχεται από την ικανότητά τους να ανιχνεύουν και να εξάγουν χαρακτηριστικά από διάφορα επίπεδα της εικόνας. Η βασική ιδέα πίσω από τα CNNs είναι η εφαρμογή συνελικτικών φίλτρων στην είσοδο, με σκοπό την ανίχνευση διάφορων μοτίβων και χαρακτηριστικών σε διάφορες περιοχές της εικόνας.

Το προτέρημα του συνελικτικού νευρικού δικτύου είναι η εκμάθηση των χαρακτηριστικών από σημαντικό αριθμό δεδομένων χωρίς σήμανση. Αναβαθμίζει το συμβατικό πολυεπίπεδο perceptron μέσα από τρεις τεχνικές, όπως κοινή χρήση παραμέτρων, ισοδύναμη αναπαράσταση και αραιές αλληλεπιδράσεις. Με αντανάκλαση στις εφαρμογές επεξεργασίας απεικόνισης (David et al, 2019).

Πρόσθετο πλεονέκτημα είναι ο εύκολος βαθμός εκπαίδευσης και επίτευξη αναβαθμισμένης ικανότητας απεξάρτησης από τα ολικά συνδεδεμένα μεταξύ των γειτωνικών στρωμάτων δίκτυα, όπως και μπορεί να συγχωνεύσει πανομοιότυπα χαρακτηριστικά σε ένα όπως οι έξοδοι όμορων ομάδων νευρώνων με μια αντιπροσωπευτική τιμή, μολονότι, κατά κόρον, είναι αδύνατο με άλλο τρόπο η τεχνική τα γειτονικά παράθυρα ομαδοποιημένων νευρώνων να επικαλύπτονται (LeCun, et al., 2015).

2.2. Επίπεδα Συνέλιξης (Convolutional Layers)

Κατά τη διάρκεια της εκπαίδευσης, αυτά τα φίλτρα προσαρμόζονται για να αναγνωρίζουν συγκεκριμένα χαρακτηριστικά που σχετίζονται με την επιθυμητή κατηγορία ή αντικείμενο.

Τα CNNs αποτελούνται από διάφορα επίπεδα, όπως:

1. Επίπεδα Συνέλιξης (Convolutional Layers): Εφαρμόζουν συνελικτικά φίλτρα στην εικόνα για την εξαγωγή χαρακτηριστικών.
2. Επίπεδα Συγκέντρωσης (Pooling Layers): Μειώνουν τη διάσταση των χαρακτηριστικών και επιτρέπουν την ανεκτικότητα στις μετατοπίσεις των χαρακτηριστικών στην εικόνα.
3. Επίπεδα Πλήρως Συνδεδεμένων (Fully Connected Layers): Λειτουργούν ως ταξινομητές και αποφασίζουν την κατηγορία ή την ταξινόμηση της εικόνας.

Τα CNNs έχουν επιδείξει εντυπωσιακές επιδόσεις σε πολλές εικαστικές εργασίες, όπως η αναγνώριση αντικειμένων, η ταξινόμηση εικόνων, η ανίχνευση ανθρώπων και προσώπων, και η επεξεργασία εικόνων και βίντεο. Επιπλέον, τα CNNs έχουν τη δυνατότητα να εξάγουν χαρακτηριστικά από εικόνες, τα οποία μπορούν να χρησιμοποιηθούν σε άλλες εφαρμογές μηχανικής μάθησης (David et al, 2019).

Στο πλαίσιο ενός συνελικτικού νευρωνικού δικτύου, η συνέλιξη είναι μια μαθηματική πράξη που χρησιμοποιείται για την εξαγωγή χαρακτηριστικών από μια είσοδο. Αυτή η πράξη αποτελείται από τον πολλαπλασιασμό ενός συνόλου βαρών με την είσοδο του δικτύου(όπως ένα παραδοσιακό νευρωνικό δίκτυο). Τα βάρη αντιπροσωπεύουν την πληροφορία που το δίκτυο προσπαθεί να μάθει κατά τη διαδικασία εκπαίδευσης.

Η πράξη της συνέλιξης σε διακριτό χρόνο ορίζεται ως :

$$x(n) = \sum_{t=-\infty}^{t=+\infty} f(t)g(n-t)$$

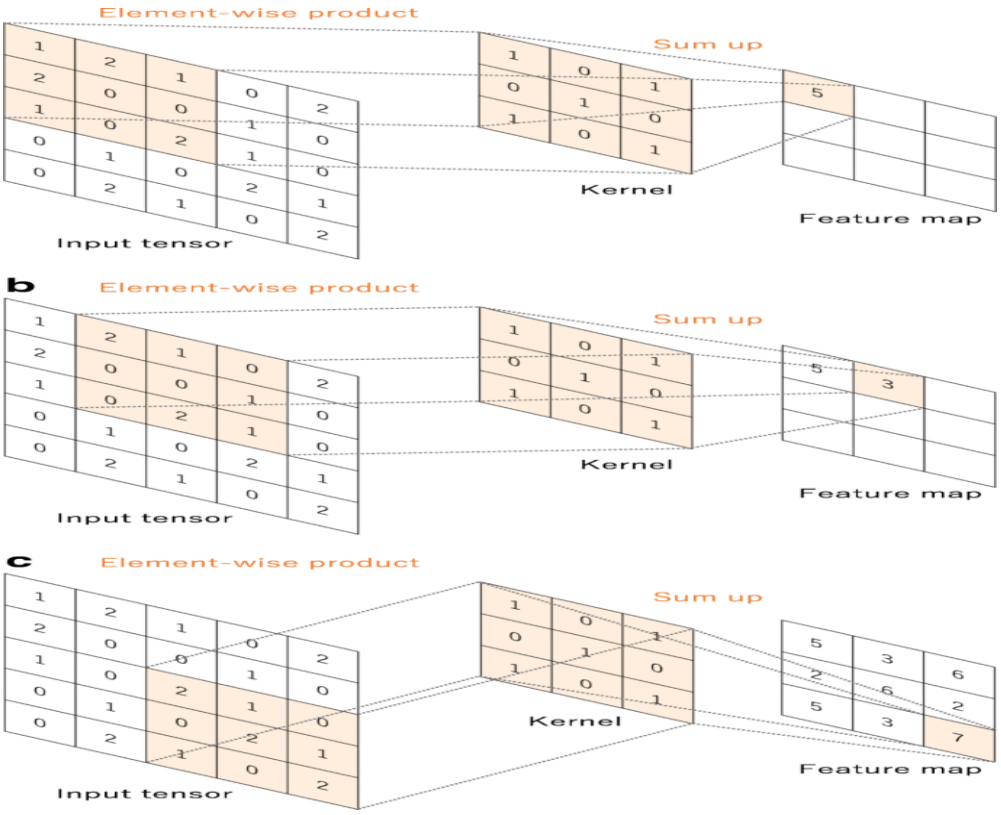
Στην πράξη, η συνέλιξη χρησιμοποιεί έναν μικρό πίνακα, γνωστό ως φίλτρο ή πυρήνας, και τον εφαρμόζει σε διάφορα σημεία της εισόδου. Κάθε φορά που το φίλτρο εφαρμόζεται σε ένα σημείο της εισόδου, γίνεται ο πολλαπλασιασμός των αντίστοιχων στοιχείων του φίλτρου με τα αντίστοιχα στοιχεία της εισόδου, και τα αποτελέσματα προστίθενται μαζί για να δοθεί η τελική τιμή της συνέλιξης σε αυτό το σημείο.

Με αυτόν τον τρόπο, η συνέλιξη επιτρέπει στο νευρωνικό δίκτυο να εντοπίζει και να ανιχνεύει διάφορα χαρακτηριστικά στην είσοδο, όπως γωνίες, ακμές, συνοχή του χρώματος κ.λπ. Αυτό επιτρέπει στο δίκτυο να αναπαράγει την ικανότητα ανίχνευσης χαρακτηριστικών που έχουν οι ανθρώπινοι ορατοί νευρώνες και να επιτύχει υψηλή απόδοση σε διάφορες εργασίες όπως η εικονική αναγνώριση, η αναγνώριση αντικειμένων και η επεξεργασία εικόνας γενικότερα.

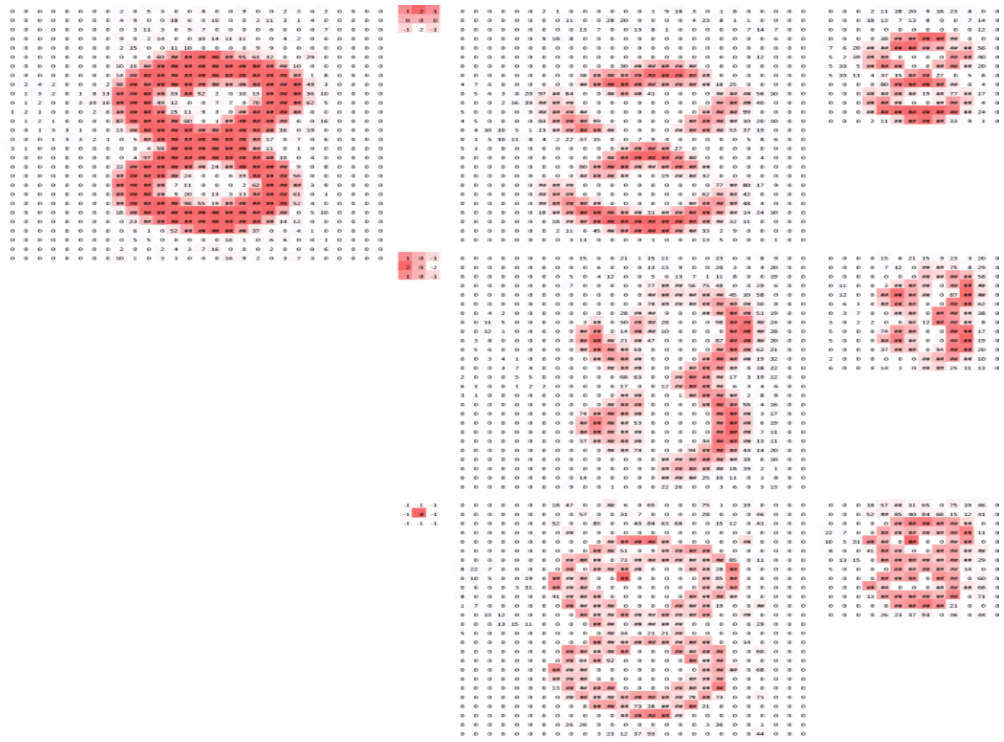
Η συνέλιξη εφαρμόζεται σε έναν όγκο εισόδου χρησιμοποιώντας ένα φίλτρο που μετακινείται πάνω από τον όγκο εισόδου με ένα βήμα που ονομάζεται "stride". Κάθε φορά που το φίλτρο εφαρμόζεται σε ένα σημείο του όγκου εισόδου, γίνεται ο πολλαπλασιασμός των αντίστοιχων στοιχείων του φίλτρου με τα αντίστοιχα στοιχεία του όγκου εισόδου, και τα αποτελέσματα προστίθενται μαζί για να δοθεί ο τελικός χάρτης χαρακτηριστικών. Οι διαστάσεις του χάρτη υπολογίζονται από τον τύπο (N - F) + 1.

Όταν ένα φίλτρο διαστάσεων 5×5×3 εφαρμόζεται σε έναν όγκο εισόδου διαστάσεων 32×32×3 τότε η συνέλιξη παράγει έναν χάρτη χαρακτηριστικών διαστάσεων 28×28×1. Οι μειώσεις στης διαστάσεις οφείλονται στον προαναφερθέντα τύπο.

Επιπλέον, ο αριθμός των φίλτρων που εφαρμόζονται στον όγκο εισόδου καθορίζει το βάθος του όγκου εξόδου. Για παράδειγμα, εάν εφαρμοστούν 10 φίλτρα συνέλιξης 5x5x3 σε έναν όγκο εισόδου διαστάσεων 32×32×3, ο όγκος εξόδου θα έχει διαστάσεις 28×28×10. Ο αριθμός των φίλτρων των επιπέδων συνέλιξης και το stride είναι θεωρούνται *υπερ-παράμετροι*. Παρακάτω φαίνεται η κίνηση ενός φίλτρου 3x3x1 με stride 1 σε ένα προτυπο 5x5x1. Παρατηρούμε πως αλλάζει αυτό την μέγεθος του τελικού χάρτη.



Εικόνα 7. Εικόνα εφαρμογή πράξης συνελίξεις



Εικόνα 8. Εικόνα εφαρμογή πράξης συνελίξεις

Σε μοντέλα με μεγάλο αριθμό επιπέδων, παρατηρείται η μείωση των διαστάσεων μήκους και πλάτους του χάρτη χαρακτηριστικών. Αυτή η μείωση συμβαίνει λόγω της διαδοχικής εφαρμογής πράξεων συνέλιξης στο μοντέλο όπως όταν εφαρμόζουμε πολλαπλά επίπεδα συνέλιξης σε ένα χάρτη χαρακτηριστικών. Κάθε επίπεδο συνέλιξης περιλαμβάνει την εφαρμογή φίλτρων στον χάρτη χαρακτηριστικών και τη μείωση των διαστάσεων μέσω της πράξης της συνέλιξης. Με κάθε εφαρμογή συνέλιξης, οι διαστάσεις του χάρτη μειώνονται, καθώς η πληροφορία συντομεύεται. Αυτή η διαδικασία επαναλαμβάνεται με τα διάφορα επίπεδα συνέλιξης, με αποτέλεσμα να παράγονται χάρτες χαρακτηριστικών με μειωμένες διαστάσεις μήκους και πλάτους.

Αυτή η συμπεριφορά που περιορίζει τις διαστάσεις των φίλτρων και μειώνει τις διαστάσεις των εξόδων των επιπέδων συνέλιξης είναι ανεπιθύμητη. Για να την αποφύγουμε, μπορούμε να χρησιμοποιήσουμε την τεχνική του zero-padding. Η διαδικασία αυτή επεκτείνει τις διαστάσεις μήκους και πλάτους του χάρτη εισόδου προσθέτοντας μηδενικά στα σύνορα του χάρτη. Με άλλα λόγια, προσθέτουμε μηδενικά πίξελ στις εξωτερικές περιοχές του χάρτη χαρακτηριστικών για να το επεκτείνουμε σε μεγαλύτερη διάσταση. Αυτό μας επιτρέπει να διατηρήσουμε τις αρχικές διαστάσεις κατά την εκτέλεση των πράξεων συνέλιξης και να αποφύγουμε τη μείωση των διαστάσεων στα επίπεδα. Η διαδικασία αυτή ονομάζεται zero-padding λόγω της προσθήκης μηδενικών τιμών. Με την εφαρμογή του zero – padding και του stirde οι διαστάσεις του χάρτη χαρακτηριστικών υπολογίζονται από την εξίσωση (N – F + 2P)/S + 1 όπου P η υπέρ-παράμετρος zero-padding και S η υπέρ παράμετρος stride.

Συνοψίζοντας, ένα επίπεδο συνέλιξης έχει τα εξής χαρακτηριστικά:

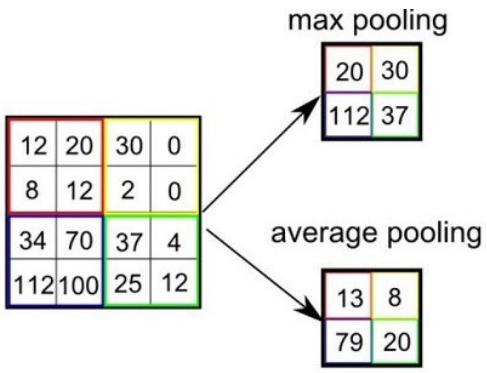
- Διαστάσεις όγκου εισόδου: W1 × H1 × D1
- Hyperparameters:
  - F: Μέγεθος του φίλτρου (F × F)
  - K: Αριθμός φίλτρων
  - S: Βήμα μετατόπισης
  - P: Ποσότητα zero-padding

• Διαστάσεις όγκου εξόδου: W2 × H2 × D2, D2 = K όπου:

- – W2 = (W1 − F)/S + 1K: Αριθμός φίλτρων
- – H2 = (H1 − F)/S + 1
- – D2 = D1

2.3. Επίπεδο Υπό-Δειγματοληψίας (Pooling Layer)

Τα επίπεδα υπο-δειγματοληψίας είναι μια τεχνική που συνήθως χρησιμοποιείται στα νευρωνικά δίκτυα μεταξύ, συνήθως, διαδοχικών επιπέδων συνέλιξης. Ο σκοπός της υποδειγματοληψίας είναι να μειώσει τις χωρικές διαστάσεις των αναπαραστάσεων, ενεργώντας σαν μία συνάρτηση υποδειγματοληψίας, ελαττώνοντας έτσι τον αριθμό των παραμέτρων και των υπολογισμών που απαιτούνται στο νευρωνικό δίκτυο. Οι πιο συνηθισμένες συναρτήσεις υποδειγματοληψίας είναι οι συναρτήσεις "max" και "average".



Εικόνα 9. Εικόνα εφαρμογή της υπο-δειγματοληψίας

Η υποδειγματοληψία με συνάρτηση "max" επιλέγει το μέγιστο στοιχείο από ένα παραθυράκι των δεδομένων, ενώ η υποδειγματοληψία με συνάρτηση "average" υπολογίζει τον μέσο όρο των τιμών στο παραθυράκι.

2.4. Πλήρως Συνδεδεμένο Επίπεδο

Στο πλαίσιο λειτουργίας των ΤΝΔ και σχετικά με τον τρόπο σύνδεσης των νευρώνων τους αναφέρονται και τα πλήρως συνδεδεμένα (Fully connected). Εδώ κάθε νευρώνας

συνδέεται με το σύνολο των νευρώνων του επόμενου επιπέδου όπως και στο MLP. Στα συνελικτικά νευρωνικά δίκτυα (CNN), το τελευταίο επίπεδο είναι συνήθως πλήρως συνδεδεμένο και ο αριθμός των νευρώνων σε αυτό το επίπεδο είναι ίσος με τον αριθμό των κλάσεων που προβλέπουμε, ενώ η αναγνώριση προτύπων τελείται με βάση τις πληροφορίες που έχουν εξαχθεί από τα προηγούμενα συνελικτικά επίπεδα (Haykin,1999).

Τα δίκτυα αυτά πραγματοποιούν μη γραμμικές απεικονίσεις από το χώρο εισόδων R d στο χώρο εξόδων R p και προσεγγίζουν οποιαδήποτε συνάρτηση, ανεξάρτητα από τον αριθμό των χρησιμοποιούμενων νευρώνων. Οι συναρτήσεις ταξινόμησης διαχωρίζουν το χώρο των προτύπων με ορισθείσες τομές των υπερεπιπέδων. Στα δίκτυα MLP οι είσοδοι στις συναρτήσεις ενεργοποίησης ισούται με το εσωτερικό γινόμενο του διανύσματος εισόδου με το διάνυσμα των βαρών του εκάστοτε νευρώνα (Kasabov,1996).

2.5. Υπερεκπαίδευση (Overfitting):

Η υπερεκπαίδευση είναι μια κατάσταση στη μηχανική μάθηση όταν ένα μοντέλο μαθαίνει τα δεδομένα εκπαίδευσης γίνεται τόσο ακριβές ώστε να προσαρμόζεται υπερβολικά σε αυτά και χάνει την ικανότητα γενίκευσης σε νέα, προς εκπαίδευση, δεδομένα. Αυτό συμβαίνει όταν το μοντέλο έχει πολλές μεταβλητές (βάρη) ή όταν υπάρχουν πολύ λίγα δεδομένα εκπαίδευσης. Ουσιαστικά τότε το μοντέλο απομνημονεύει τις εισόδους/εξόδους χωρίς να περιγράφει τα ενυπάρχοντα μοτίβα του συνόλου εκπαίδευσης.

2.6. Κανονικοποίηση (Regularization):

Η κανονικοποίηση είναι ένα σύνολο τεχνικών που στοχεύουν στην αντιμετώπιση του προβλήματος της υπερεκπαίδευσης. Μια κοινή μέθοδος είναι η προσθήκη ενός όρου κανονικοποίησης στη συνάρτηση κόστους, που επηρεάζει την ενημέρωση των βαρών του μοντέλου κατά την εκπαίδευση.

Για παράδειγμα, η L2 κανονικοποίηση προσθέτει έναν όρο στο κόστος που είναι ανάλογο του τετραγώνου της Ευκλείδειας απόστασης των βαρών. Αυτό αναγκάζει τα βάρη να παραμένουν μικρά, αποτρέποντας την υπερεκπαίδευση.

2.7. Βελτιστοποίηση με Batch (Batch Normalization):

Η Batch Normalization (BN) είναι μια τεχνική που χρησιμοποιείται για να βελτιστοποιήσει την εκπαίδευση των νευρωνικών δικτύων. Σκοπός της είναι να προσαρμόσει την είσοδο κάθε επιπέδου έτσι ώστε να έχει μέση τιμή μηδέν και τυπική απόκλιση μονάδας σύμφωνα με το τύπο που είναι γνωστός και ως z-score κανονικοποίηση των δεδομένων:

$x_i' = (x_i - \mu)/\sigma$

όπου μ η μέση τιμή και σ η τυπική απόκλιση. Αυτή η τεχνική μπορεί να βοηθήσει στην ταχύτερη σύγκλιση του μοντέλου.

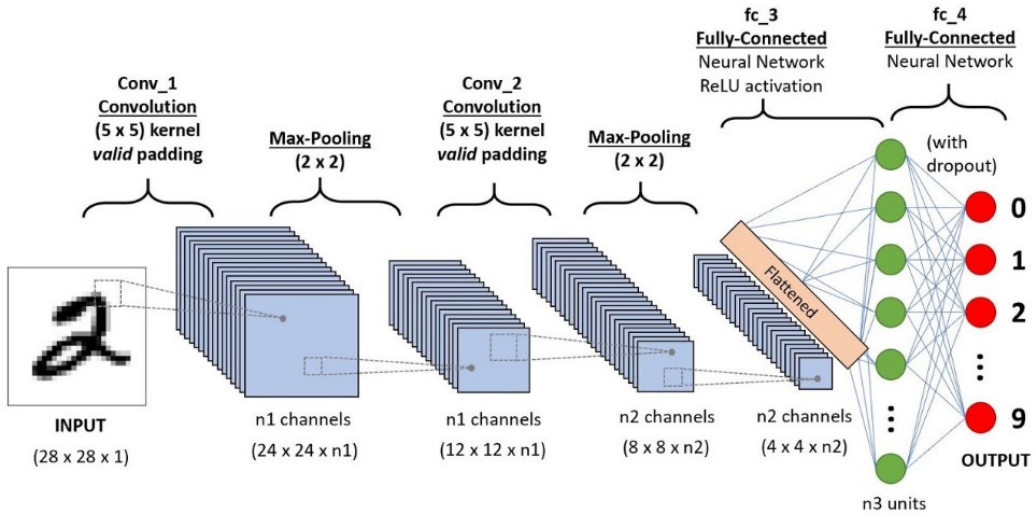
2.8. Απόρριψη (Dropout):

Η τεχνική απόρριψης είναι ένα είδος κανονικοποίησης που εφαρμόζεται κατά τη διάρκεια της εκπαίδευσης. Κατά την εκπαίδευση, τυχαία επιλεγμένοι νευρώνες "απορρίπτονται" (η έξοδος τους μηδενίζεται), με πιθανότητα p (π.χ., 0.5). Αυτό επιτρέπει στο μοντέλο να μην εξαρτάται υπερβολικά από συγκεκριμένες μονάδες, ενισχύει την γενίκευση και επιταχύνει τη διαδικασία εκπαίδευσης. Πιο αναλυτικά, αν είχαμε 120 νευρώνες με p=0,5 θα απέμεναν 60 τυχαίοι νευρώνες από αυτούς, δηλαδή θα μηδενίζονταν οι έξοδοί τους.

2.9. Μεροληψία (Bias):

Οι όροι μεροληψίας (bias terms) προστίθενται σε κάθε νευρώνα ενός νευρωνικού δικτύου. Οι όροι αυτοί επιτρέπουν την προσθήκη μιας μεταβλητής βάρους στην έξοδο του αθροιστή, ανεξάρτητα από τις εισόδους.

Στην ακόλουθη εικόνα παρουσιάζεται ένα παράδειγμα ενός CNN συνελικτικού δικτύου βαθιάς μάθησης με 6 επίπεδα βάθος όπου αρχικά εφαρμόζονται φίλτρα συνέλιξης 5x5 σε εικόνες μεγέθους 28x28x1 στην έξοδο τη συνέλιξης εφαρμόζεται Max Pooling με πίνακα 2x2 και η ίδια διαδικασία επαναλαμβάνεται άλλη μια φορά ώστε το προτελευταίο επίπεδο υπάρχει ένα πλήρες συνδεδεμένο νευρωνικό δίκτυο με ReLu συνάρτηση ενεργοποίησης. Τέλος γίνεται μια κανονικοποίηση με dropout και εφαρμογή της Softmax στην έξοδο για κατηγοριοποίηση 6 πιθανών κλάσεων.



Εικόνα 10. Εικόνα Παράδειγμα CNN με φίλτρα συνέλιξης, Max Pooling, πλήρες συνδεδεμένο νευρωνικό δίκτυο με ReLu συνάρτηση ενεργοποίησης, με dropout και Softmax έξοδο.

2.10.Transfer Learning

Η μάθηση με μεταφορά (Transfer Learning)είναι μια τεχνική της μηχανικής μάθησης με βάση ένα προεκπαιδευμένο μοντέλο που ήδη έχει εκπαιδευτεί σε ένα σύνολο δεδομένων με γενική γνώση για την αναγνώριση χαρακτηριστικών. Πρόκειται για παράμετρο της βαθιάς μάθησης με τα προεκπαιδευμένα μοντέλα για την επίλυση προβλημάτων, συναρτήσει και του απαιτούμενου όγκου υπολογισμών για την εκπαίδευσή τους. Κάθε μοντέλο εκπαιδεύεται σε ένα διαφορετικό σύνολο δεδομένων, συνήθως το ImageNet για τα συνελικτικά νευρωνικά δίκτυα. Κατά την μάθηση, το δίκτυο χρησιμοποιεί τα βάρη (weights) και τα χαρακτηριστικά (features) που έχουν εξαχθεί από το προηγούμενο μοντέλο. Αυτό μπορεί να γίνει περιορίζοντας την εκπαίδευση σε ορισμένα επίπεδα του μοντέλου ή προσθέτοντας νέα επίπεδα στο τέλος του μοντέλου για την εκμάθηση των νέων χαρακτηριστικών. Στη μεταφορά

μάθησης χρησιμοποιούνται οι έννοιες του τομέα (domain) και i (task) (Torrey and Shavlik, 2010).

Η περίπτωση της μη-επιβλεπόμενης προεκπαίδευσης (unsupervised pretraining) είναι μια συνήθης περίπτωση μεταφοράς μάθησης με διαθέσιμες ποσότητες μη επιβλεπόμενων δεδομένων εκπαίδευσης, αλλά πολύ λίγα δεδομένα εκπαίδευσης με ετικέτες.

Οι κύριοι λόγοι που το Transfer Learning είναι αποτελεσματικό είναι:

- Έχει γενική γνώση: Τα προεκπαιδευμένα μοντέλα έχουν εκπαιδευτεί σε μεγάλα και πολύπλοκα σύνολα δεδομένων, οπότε έχουν αποκτήσει γενική γνώση για την αναγνώριση μοτίβων. Αυτή η γνώση μπορεί να μεταφερθεί και να χρησιμοποιηθεί σε νέες εργασίες.
- Αποφεύγει την υπερεκπαίδευση: Η υπερεκπαίδευση είναι ένα πρόβλημα όταν ένα μοντέλο έχει εκπαιδευτεί με ένα μικρό σύνολο δεδομένων και δεν γενικεύει καλά σε νέα δεδομένα. Το Transfer Learning μπορεί να βοηθήσει να αποφευχθεί η υπερεκπαίδευση, καθώς εκμεταλλεύεται την προηγούμενη γνώση που έχει αποκτηθεί.
- Απαιτεί λιγότερα δεδομένα: Η εκπαίδευση ενός μοντέλου από την αρχή μπορεί να απαιτεί μεγάλα σύνολα δεδομένων. Το Transfer Learning μπορεί να χρησιμοποιήσει τα προηγούμενα δεδομένα και να απαιτήσει λιγότερα νέα δεδομένα για την εκπαίδευση του μοντέλου στη νέα εργασία (Torrey and Shavlik, 2010).

ΚΕΦΑΛΑΙΟ 3

Ανάλυση και σχεδίαση

3.1. Λογισμικό εφαρμογής

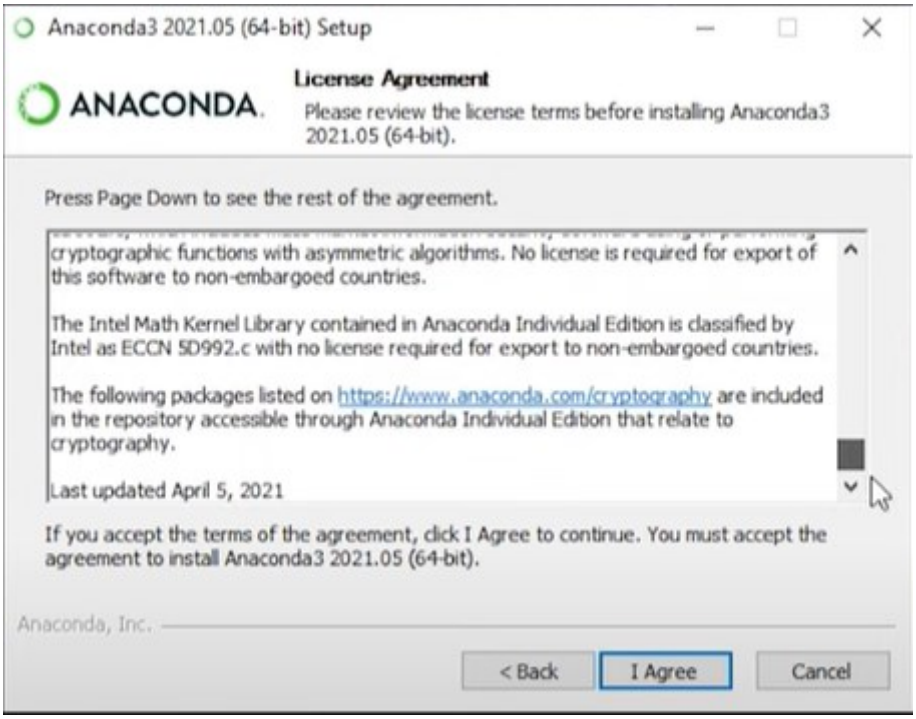
Στο παρόν κεφάλαιο παρουσιάζονται τα κύρια εργαλεία υλικού και λογισμικού που χρησιμοποιήθηκαν για την υλοποίηση του έργου. Το έργο αναπτύχθηκε στο διαχειριστικό περιβάλλον το anaconda, με γλώσσα προγραμματισμού python και το περιβάλλον αναπτύξεις το Jupyter Notebook. Επιπλέον, χρησιμοποιήθηκαν η βιβλιοθήκη Tensorflow καθώς και οι βιβλιοθήκες Pandas και Numpy. Για τη δημιουργία γραφικών παραστάσεων αξιοποιήθηκε η βιβλιοθήκη Matplotlib. Όσον αφορά το λειτουργικό σύστημα, επιλέχθηκαν τα Windows 11.

3.2. Anaconda

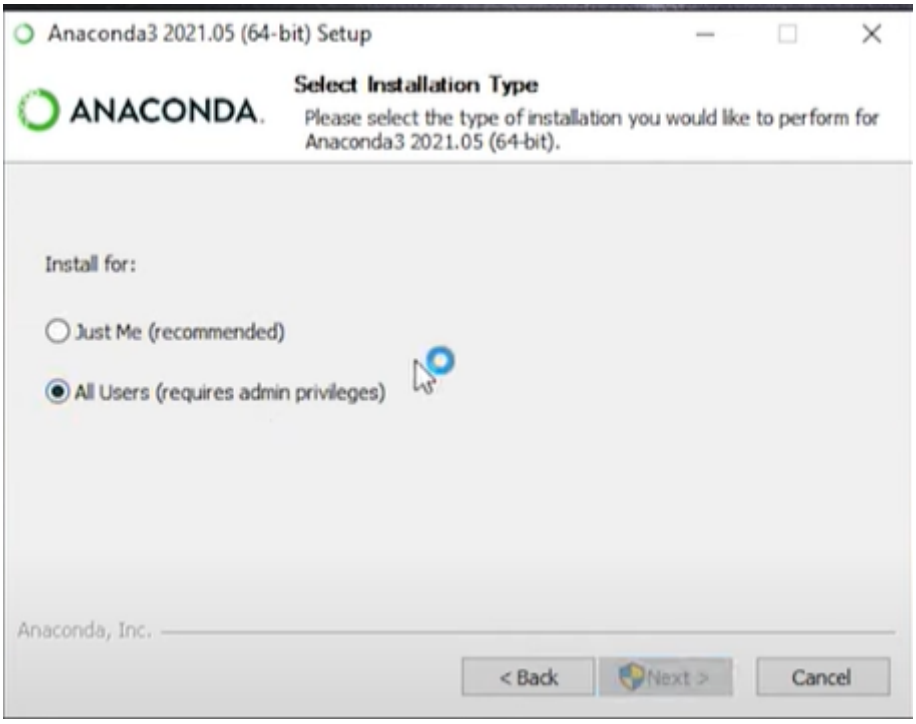
Το Anaconda είναι μια πλατφόρμα διαχείρισης περιβάλλοντος και διανομής πακέτων που χρησιμοποιείται για προγραμματισμό σε μορφή Python καθώς και σε άλλες γλώσσες προγραμματισμού. Το σημαντικό χαρακτηριστικό του είναι ότι περιλαμβάνει πολλά προεγκατεστημένα πακέτα και εργαλεία για επιστημονικό υπολογισμό όπως και για την ανάλυση δεδομένων όπως τα jyputer notebook, spyder, vs code κ.ά. Τα παραπάνω καθιστούν το πρόγραμμα αυτό, ως πλατφόρμα, ιδανικό για προγραμματιστές και επιστήμονες δεδομένων που ασχολούνται με διάφορες εφαρμογές, όπως η μηχανική μάθηση, η επεξεργασία εικόνων και πολλές άλλες. Το Anaconda είναι συμβατό με τα λειτουργικά συστήματα Windows, macOS και Linux (https://www.anaconda.com/download, 6-11-2023).

Η εγκατάσταση:

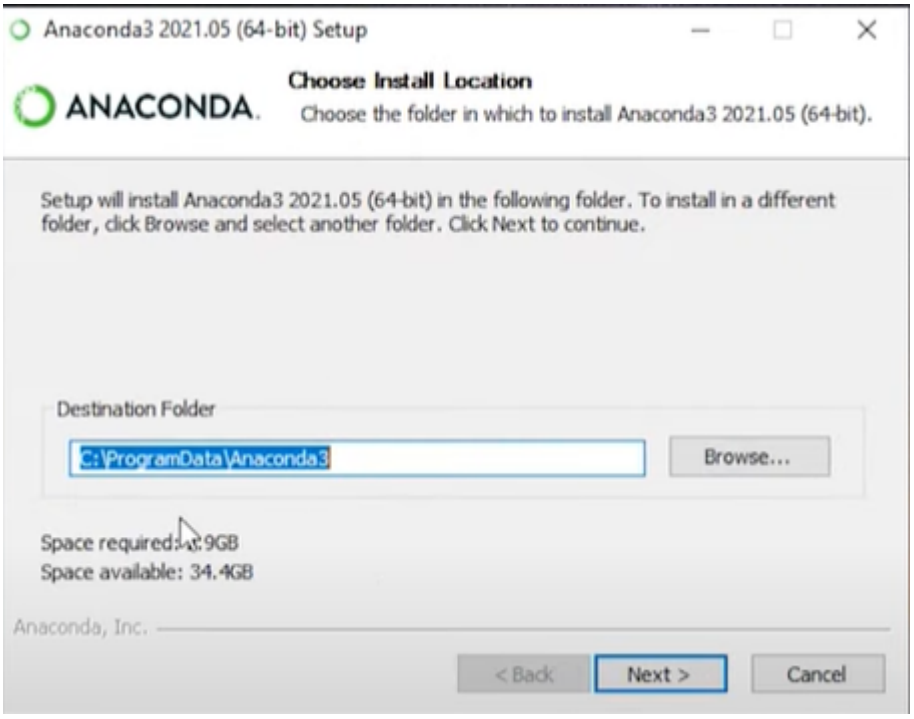
Εικόνα 11. Βήμα 1 εγκατάστασης anaconda



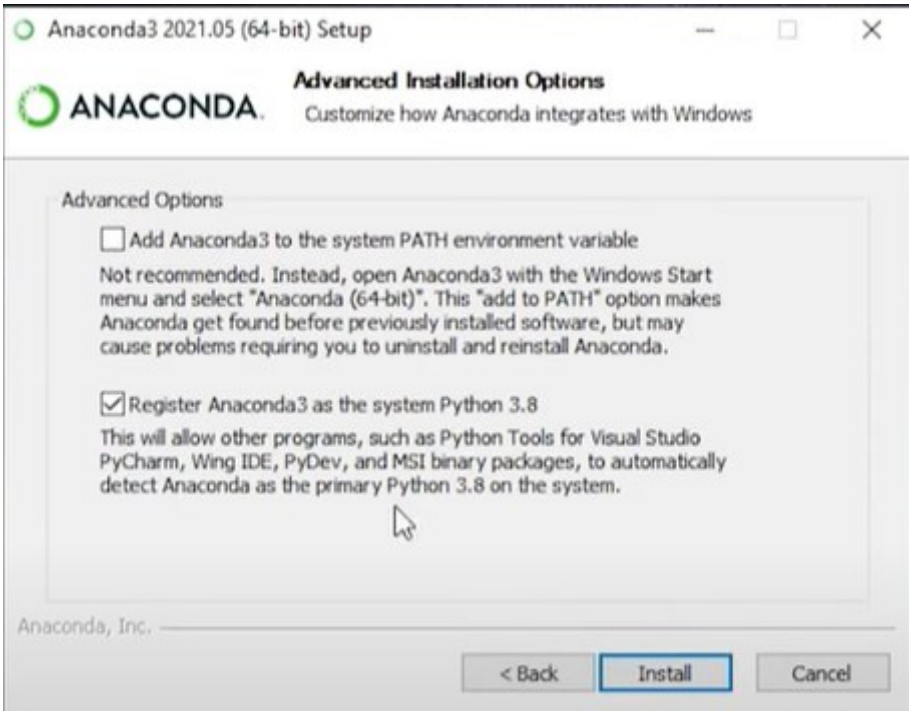
Εικόνα 12. Βήμα 2 εγκατάστασης anaconda



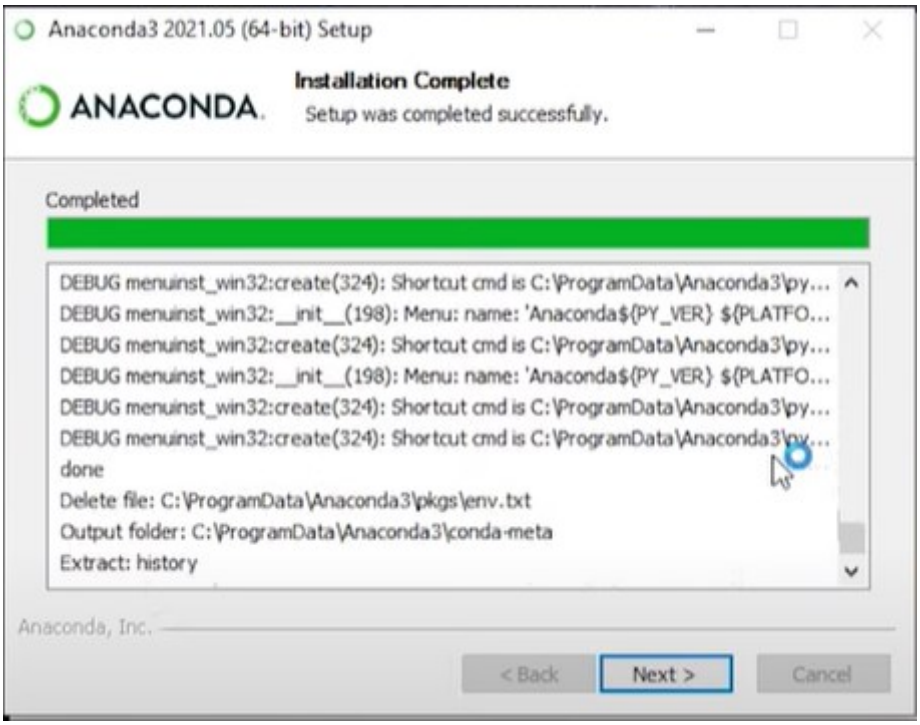
Εικόνα 13. Βήμα 3 εγκατάστασης anaconda



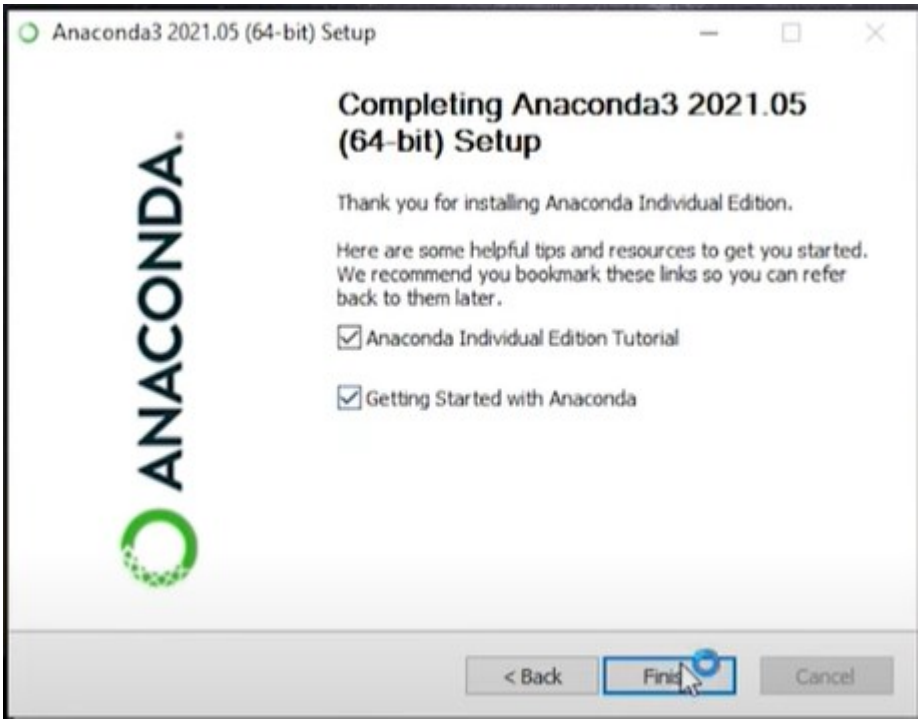
Εικόνα 14. Βήμα 4 εγκατάστασης anaconda



Εικόνα 15. Βήμα 5. εγκατάστασης anaconda



Εικόνα 16. Βήμα 6 εγκατάστασης anaconda



Εικόνα 17. Βήμα 7 εγκατάστασης anaconda

3.3. PYTHON

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου που δημιουργήθηκε από τον Guido van Rossum και πρωτοκυκλοφόρησε το 1991. Είναι γνωστή για την απλότητα και την ευανάγνωστη σύνταξή της, κάτι που την καθιστά ιδανική για αρχάριους προγραμματιστές. Ένα από τα χαρακτηριστικά της Python είναι η δυναμική της τυποποίηση, που σημαίνει ότι δεν χρειάζεται να δηλωθεί ο τύπος μιας μεταβλητής. Η Python διαθέτει επίσης μια ευρεία γκάμα βιβλιοθηκών και πακέτων που καλύπτουν διάφορους τομείς, όπως επιστημονικοί υπολογισμοί, μηχανική μάθηση, διαχείριση δεδομένων, δικτύωση και ιστοσελίδες. Μερικές από τις δημοφιλείς βιβλιοθήκες περιλαμβάνουν το NumPy για επεξεργασία πινάκων και το Pandas για διαχείριση δεδομένων.

Επιπλέον, η Python έχει κατακτήσει μεγάλη δημοτικότητα στον τομέα της μηχανικής μάθησης και της τεχνητής νοημοσύνης. Η βιβλιοθήκη TensorFlow, ανάμεσα σε άλλες, παρέχει ισχυρά εργαλεία για την ανάπτυξη μοντέλων μηχανικής μάθησης και νευρωνικών δικτύων. Γενικά, η Python θεωρείται μια ευέλικτη και ισχυρή γλώσσα προγραμματισμού που καλύπτει μια ευρεία γκάμα εφαρμογών και χρησιμοποιείται

ευρέως σε διάφορους τομείς της πληροφορικής (https://docs.python.org/3/tutorial/index.html, 6-11-2023).

3.4. JUPYTER NOTEBOOK

Το Jupyter Notebook είναι μια εφαρμογή ανοικτού κώδικα που χρησιμοποιείται για την ανάπτυξη και την κοινή χρήση διαδραστικών περιεχομένων σε μορφή notebook. Αυτό το περιβάλλον προγραμματισμού επιτρέπει την εκτέλεση κώδικα, την οπτικοποίηση δεδομένων και τη σύνταξη κειμένου σε ένα ενιαίο περιβάλλον. Οι σημαντικότερες χαρακτηριστικές του Jupyter Notebook περιλαμβάνουν:

- Διαδραστική Εκτέλεση: Πρόκειται για την εκτέλεση κώδικα σε κελιά, που μας επιτρέπει την εκτέλεση μόνο μιας μεμονωμένης ενότητας κώδικα κάθε φορά με τα αποτελέσματα ορατά και άμεσα. Αυτό καθιστά την ανάπτυξη και την εξερεύνηση του κώδικα πιο διαισθητική.
- Πολυμορφική Υποστήριξη Γλωσσών: Υποστηρίζει πολλές γλώσσες προγραμματισμού, όπως Python, R και άλλες.
- Οπτικοποίηση Δεδομένων: Δυνατότητα να δημιουργηθούν γραφήματα, πίνακες και άλλα είδη οπτικοποίησης δεδομένων απευθείας στο Notebook, προσδίδοντας ακόμη περισσότερη αναλυτικότητα στα αποτελέσματα του κώδικα σας.
- Κοινή Χρήση και Αναπαραγωγή: Δυνατότητα διαμοιρασμού των Jupyter Notebooks και δυνατότητα προσβασιμότητας στον προσωπικό κώδικα καθώς και επαναχρησιμοποίησης των αποτελεσμάτων, γεγονός που διευκολύνει τη συνεργασία και την ανταλλαγή γνώσης.

Είναι ένα ισχυρό εργαλείο για ανάπτυξη, ανάλυση δεδομένων και αναπαραγωγή αποτελεσμάτων. Είναι ευέλικτο, φιλικό προς τον χρήστη και ευρέως χρησιμοποιούμενο από επιστήμονες δεδομένων, ερευνητές και προγραμματιστές (https://jupyter.org/, 6-11-2023).

3.5. TENSORFLOW

Το Tensorflow είναι μια δωρεάν βιβλιοθήκη ανοιχτού κώδικα που αναπτύχθηκε από την ομάδα έρευνας και ανάπτυξης Google Brain της Google το 2015. Χρησιμοποιείται κυρίως για την υλοποίηση και εκπαίδευση νευρωνικών δικτύων

βάθους. Το Tensorflow έχει διανύσει μια μεγάλη πορεία ανάπτυξης, με τη μεγαλύτερη αναβάθμισή του να πραγματοποιείται το 2019 με την έκδοση 2.0.

Το Tensorflow είναι πολύ ευέλικτο και υποστηρίζει πολλές γλώσσες προγραμματισμού, όπως οι Python, C++ και Javascript. Αυτό του επιτρέπει να χρησιμοποιηθεί σε διάφορους τομείς και εφαρμογές. Επίσης, έχει αναπτύξει διάφορες επεκτάσεις για να καλύψει διάφορες απαιτήσεις.

Μια από τις επεκτάσεις είναι το Tensorflow.js, το οποίο επιτρέπει τη μετατροπή υπαρχόντων μοντέλων Tensorflow σε Tensorflow.js, επιτρέποντας έτσι τη χρήση τους σε διαδικτυακές εφαρμογές. Επίσης, υπάρχει και η πλήρης ανάπτυξη του Tensorflow.js για τη δημιουργία μοντέλων και εκτέλεσή τους απευθείας στον πελάτη, προσφέροντας έτσι μεγαλύτερη ευελιξία στις διαδικτυακές εφαρμογές.

Μια άλλη επέκταση είναι το TFLite, το οποίο επιτρέπει την ανάπτυξη μοντέλων για συσκευές με μικρή υπολογιστική ισχύ, όπως κινητές συσκευές. Αυτή η επέκταση επιτρέπει την εκτέλεση των μοντέλων Tensorflow σε συσκευές με περιορισμένους πόρους, επιτυγχάνοντας την αποτελεσματική εκτέλεσή τους σε αυτό το περιβάλλον.

Συνοψίζοντας, το Tensorflow είναι μια ισχυρή βιβλιοθήκη που υποστηρίζει την ανάπτυξη και εκπαίδευση νευρωνικών δικτύων βάθους. Με τις επεκτάσεις του, όπως το Tensorflow.js και το TFLite, μπορεί να χρησιμοποιηθεί σε διαφορετικές εφαρμογές και περιβάλλοντα, προσφέροντας ευελιξία και αποτελεσματική ανάπτυξη μοντέλων (https://developers.google.com/machine-learning/, 6-11-2023).

3.6. NUMPY

Η βιβλιοθήκη NumPy (Numerical Python) είναι μια βιβλιοθήκη για την επιστημονική και αριθμητική υπολογιστική. Παρέχει υψηλής απόδοσης πίνακες και πράξεις με πίνακες, καθώς και λειτουργίες για την εκτέλεση μαθηματικών, λογικών και στατιστικών εργασιών. Οι βασικές δομές δεδομένων στο NumPy είναι οι πολυδιάστατοι πίνακες (ndarrays). Οι πίνακες NumPy είναι πολύ πιο αποτελεσματικοί από τις κανονικές λίστες της Python, καθώς επιτρέπουν την εκτέλεση γρήγορων πράξεων με πίνακες, όπως αναγνώριση προτύπων, γραμμική άλγεβρα, αριθμητικές πράξεις και πολλά άλλα. Οι πίνακες NumPy μπορούν να έχουν διαστάσεις (n-dimensional) και να περιέχουν στοιχεία του ίδιου τύπου δεδομένων, πράγμα που τους επιτρέπει να είναι αποδοτικοί και απλοί στη χρήση όπως αριθμητικές πράξεις δηλαδή πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση

μεταξύ πινάκων, καθώς και πράξεις σύγκρισης και λογικών πράξεων. Επιπλέον, παρέχονται συναρτήσεις για την εκτέλεση μαθηματικών λειτουργιών, όπως τριγωνομετρικές, εκθετικές, λογαριθμικές, αριθμητικές συναρτήσεις και πολλές άλλες. Με τη βοήθεια της NumPy εκτελούνται γρήγορες και αποτελεσματικές λειτουργίες σε μεγάλα σύνολα δεδομένων, όπως φόρτωση, αποθήκευση, επεξεργασία και ανάλυση δεδομένων. Η βιβλιοθήκη NumPy είναι επίσης στενά συνδεδεμένη με άλλες βιβλιοθήκες επιστημονικού υπολογισμού στην Python, όπως η SciPy (Scientific Python) και η Matplotlib (για οπτικοποίηση δεδομένων).

Το Numpy αποτελεί βασικό κομμάτι του Tensorflow καθώς τα αντικείμενα ndarray μετατρέπονται αυτόματα σε tensors καθώς και το ανάποδο. Γενικά, η NumPy είναι μια απαραίτητη βιβλιοθήκη για επιστημονικούς υπολογισμούς και αριθμητικές εργασίες στην Python, προσφέροντας απόδοση, ευελιξία και ευκολία χρήσης (https://numpy.org/, 7-11-2023).

3.7. PANDAS

Η βιβλιοθήκη Pandas είναι μια ισχυρή βιβλιοθήκη ανάλυσης και επεξεργασίας δεδομένων στη γλώσσα προγραμματισμού Python. Προσφέρει υψηλής απόδοσης δομές δεδομένων και εργαλεία για την εύκολη και αποτελεσματική εργασία με δομές δεδομένων όπως πίνακες και χρονοσειρές. Ορισμένα από τα βασικά χαρακτηριστικά της βιβλιοθήκης Pandas περιλαμβάνουν:

- DataFrame: Ο βασικός τύπος δεδομένων στο Pandas είναι ο DataFrame, ο οποίος αντιπροσωπεύει έναν δισδιάστατο πίνακα με ετικέτες στις σειρές και στήλες του. Η ευελιξία επιτρέπει την αποθήκευση και επεξεργασία δεδομένων από πολλαπλές πηγές, όπως αρχεία CSV, SQL βάσεις δεδομένων και άλλες μορφές δεδομένων.
- Διαχείριση δεδομένων: Η Pandas παρέχει ευέλικτες λειτουργίες για τη διαχείριση δεδομένων, όπως φιλτράρισμα, ταξινόμηση, αναδιάταξη, συνένωση, ομαδοποίηση και περιλαμβανόμενα/αποκλειόμενα σετ δεδομένων. Αυτές οι λειτουργίες επιτρέπουν τον εύκολο μετασχηματισμό και ανάλυση δεδομένων, ακόμη και σε περιπλοκές δομές δεδομένων.
- Επεξεργασία χρονοσειρών: Η Pandas παρέχει προηγμένες λειτουργίες για την επεξεργασία και ανάλυση χρονοσειρών. Αυτό επιτρέπει τη δημιουργία, την

ευθυγράμμιση, τη δειγματοληψία, την ανάλυση και την οπτικοποίηση χρονοσειρών δεδομένων με ευκολία.

• Επιδόσεις και αποτελεσματικότητα: Η Pandas είναι σχεδιασμένη για να είναι γρήγορη και αποτελεσματική στην επεξεργασία μεγάλων συνόλων δεδομένων. Εκμεταλλεύεται τις δυνατότητες του NumPy για υψηλές αποδόσεις και χρησιμοποιεί αποκλειστικές δομές δεδομένων και αλγορίθμους για την επίτευξη αποτελεσματικής επεξεργασίας.

• Ευελιξία και επέκταση: Η Pandas είναι ευέλικτη και επεκτάσιμη, καθώς υποστηρίζει την προσθήκη προσαρμοσμένων συναρτήσεων και λειτουργιών επεξεργασίας δεδομένων. Επίσης, συνεργάζεται καλά με άλλες βιβλιοθήκες Python (https://pypi.org/project/pandas/, 7-11-2023).

3.8. MATPLOTLIB

Η βιβλιοθήκη Matplotlib είναι μια από τις πιο δημοφιλείς βιβλιοθήκες οπτικοποίησης δεδομένων στη γλώσσα προγραμματισμού Python. Χρησιμοποιείται ευρέως για τη δημιουργία γραφημάτων, διαγραμμάτων και απεικονίσεων που βοηθούν στην κατανόηση και απεικόνιση δεδομένων.

Ορισμένα από τα βασικά χαρακτηριστικά της βιβλιοθήκης Matplotlib περιλαμβάνουν:

• Ευελιξία στην οπτικοποίηση: Η Matplotlib παρέχει μια ευρεία γκάμα εργαλείων για τη δημιουργία διαφόρων ειδών γραφημάτων, όπως γραφήματα γραμμών, απεικονίσεις διασποράς, γραφήματα μπαρών, πίτας, προσανατολισμένα γραφήματα, ιστογράμματα και πολλά άλλα. Υπάρχει δυνατότητα προσαρμογής σχεδόν σε κάθε πτυχή των γραφημάτων, όπως τα χρώματα, τα σύμβολα, τις ετικέτες, τις αξίες άξονα, τον τίτλο και τα περιθώρια, για κάθε επιθυμητή οπτική αισθητική.

• Απλή σύνταξη: Η Matplotlib παρέχει μια απλή και ευανάγνωστη σύνταξη για την δημιουργία γραφημάτων. Δίνει τη δυνατότητα στο χρήστη μα συνθέσει γραφήματα με μερικές γραμμές κώδικα και να προσαρμόσει τα γραφήματα σταδιακά για τα επιθυμητά αποτελέσματα.

• Υποστήριξη για διάφορες πλατφόρμες: Η Matplotlib είναι μια πλατφόρμα ανεξάρτητη και μπορεί να χρησιμοποιηθεί σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των Windows, Linux και macOS αλλά και σε συνδυασμό με

γραφικά περιβάλλοντα όπως το Jupyter Notebook για διαδραστική απεικόνιση δεδομένων.

• Επαγγελματικής ποιότητας απεικόνιση: Η Matplotlib παράγει γραφήματα υψηλής ποιότητας με εξαιρετική ανάλυση και λεπτομέρεια δηλαδή γραφήματα σε διάφορες μορφές, όπως εικόνες PNG, JPEG, PDF, SVG και άλλες, για τη χρήση τους σε αναφορές, εκτυπώσεις ή δημοσιεύσεις.

Η Matplotlib είναι μια πολύ ισχυρή βιβλιοθήκη οπτικοποίησης δεδομένων και μπορεί να προσφέρει πολλές δυνατότητες για την απεικόνιση και ανάλυση των δεδομένων στη γλώσσα προγραμματισμού (https://matplotlib.org/, 7-11-2023).

3.9. HARDWARE

Η βιβλιοθήκη Tensorflow είναι σχεδιασμένη για να χρησιμοποιεί παραγωγικά την πλατφόρμα CUDA της NVIDIA. Το CUDA είναι μια πλατφόρμα παράλληλων υπολογιστών και ένα μοντέλο προγραμματισμού που αναπτύχθηκε από την NVIDIA για να επιτρέπει γενικούς υπολογισμούς σε μονάδες γραφικής επεξεργασίας (GPU) . Χρησιμοποιώντας το CUDA, οι προγραμματιστές μπορούν να επιταχύνουν τις υπολογιστικές εφαρμογές εκμεταλλευόμενοι τη δύναμη των GPU. Ωστόσο, αν δεν υπάρχει διαθέσιμη μια GPU της NVIDIA, το Tensorflow μπορεί να χρησιμοποιήσει μόνο τον επεξεργαστή του υπολογιστή (CPU), αλλά με μεγάλη αύξηση στον χρόνο εκτέλεσης των εφαρμογών αυτών.

ΚΕΦΑΛΑΙΟ 4
Υλοποίηση

4.1 ΥΛΟΠΟΙΗΣΕΙΣ

Η παρούσα διπλωματική ασχολείται με την ανάπτυξης ενός νευρωνικού δικτύου για την αναγνώριση ελληνικών φαγητών από τις ψηφιακές φωτογραφίες τους. Το σύνολο εκπαίδευσης αποτελείται από εικόνες 70 διαφορετικών φαγητών (πιάτων). Πραγματοποιήθηκε μελέτη και σύγκριση μεταξύ ενός ανεκπαίδευτου δικτύου όμοιο με VGG16 και ενός VGG16 δικτύου προεκπαιδευόμενου. Η σύγκριση θα βασιστεί στην ακρίβεια (accuracy) και την επικαιροποίηση του κάθε μοντέλου. Από τις μετρήσεις αυτές θα παραχθούν γραφικές παραστάσεις που θα απεικονίζουν τα αποτελέσματα.
Όπως αναφέρθηκε στην υποενότητα 3.1. το βασικό εργαλείο λογισμικού που χρησιμοποιήθηκε είναι η βιβλιοθήκη Tensorflow, η οποία βασίζεται στην βιβλιοθήκη Keras. Η Keras είναι αναπόσπαστο κομμάτι της Tensorflow 2 και περιέχει όλες τις υλοποιήσεις των μοντέλων και επιπέδων που θα χρειαστούμε για την εκπόνηση αυτής της διπλωματικής εργασίας. Οι βασικές δομές δεδομένων (Data Structure) της Keras είναι οι layers και οι models.

4.2. VGG16

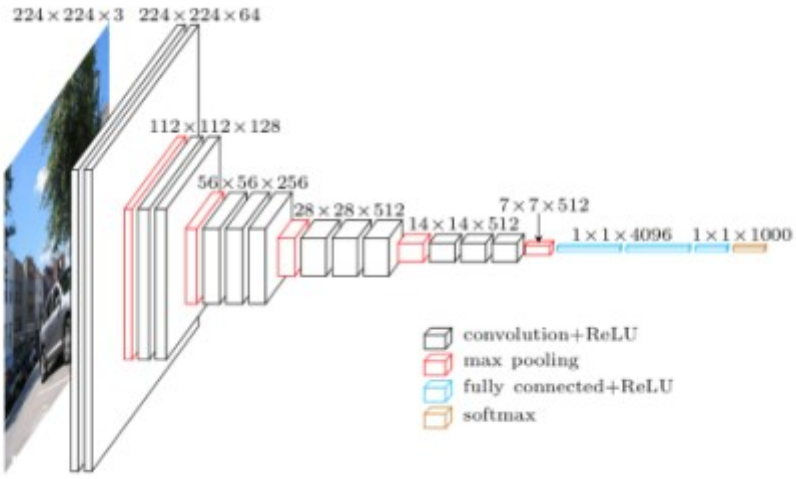
Το VGG16 είναι ένα αξιοσημείωτο μοντέλο συνελικτικού νευρωνικού δικτύου (CNN), που αναπτύχθηκε από τους Karen Simonyan και Andrew Zisserman από το Πανεπιστήμιο της Οξφόρδης και κατάφερε να επιτύχει πολύ υψηλή απόδοση στον διαγωνισμό ImageNet το 2014. Έχει καταφέρει να επιτύχει εξαιρετική απόδοση σε διάφορες τεχνικές αναγνώρισης και εργασίες ταξινόμησης εικόνων.

Το μοντέλο VGG16 αποτελείται από 16 συνελικτικά επίπεδα, που ακολουθούνται από επίπεδα συγκέντρωσης (pooling) και ένα πλήρως συνδεδεμένο επίπεδο (fully

connected layer) στο τέλος. Τα συνελικτικά φίλτρα 3x3 που εφαρμόζονται στην εικόνα είναι για την εξαγωγή χαρακτηριστικών, ενώ τα επίπεδα pooling χρησιμοποιούνται για τη μείωση της διάστασης των χαρακτηριστικών και την αύξηση της ανεκτικότητας στις μικρές μετατοπίσεις στην εικόνα. Το πλήρως συνδεδεμένο επίπεδο λειτουργεί ως ταξινομητής για τον προσδιορισμό των κατηγοριών των εικόνων.

Το VGG16 έχει εκπαιδευτεί σε μεγάλα σύνολα δεδομένων, όπως το ImageNet, που περιέχει εκατομμύρια εικόνες και χιλιάδες κατηγορίες. Αυτό το έχει καταστήσει ικανό να αναγνωρίζει και να ταξινομεί ευρεία γκάμα αντικειμένων στις εικόνες.

Ένα από τα κύρια πλεονεκτήματα του VGG16 είναι η απλότητα της αρχιτεκτονικής του, γεγονός που το καθιστά ευκολότερο στην κατανόηση και την επανεκπαίδευση για προσαρμογή σε διάφορα προβλήματα.



Εικόνα 18. Η δομή του μοντέλου VGG16

Επίπεδο	Τύπος	Αριθμός καναλιών	Διαστάσεις φίλτρων
1	Conv	64	3×3
2	Conv+Pool	64	3×3
3	Conv	128	3×3
4	Conv+Pool	128	3×3
5	Conv	256	3×3
6	Conv	256	3×3
7	Conv+Pool	256	3×3
8	Conv	512	3×3
9	Conv	512	3×3
10	Conv+Pool	512	3×3
11	Conv	512	3×3
12	Conv	512	3×3
13	Conv+Pool	512	3×3
14	FullyConnected	4096	N/A
15	FullyConnected	4096	N/A
16	FullyConnected	1000	N/A

Πίνακας 1. Η δομή και τα επίπεδα του VGG16

4.3. Προσαρμοσμένο μοντέλο (custom model)

Για τις ανάγκες της έρευνας αυτής αναπτύχθηκε ένα μοντέλο χρησιμοποιώντας την αρχιτεκτονική που προσφέρει το Tensorflow και ονομάζεται Διαδοχικό Μοντέλο(Sequential Model). Αυτό μας επιτρέπει βάλουμε τα στρώματα (Layers) του νευρωνικού δικτύου μας το ένα μετά το άλλο όπως φαίνεται και στο παρακάτω παράδειγμα:

Επίπεδο	Τύπος	Αριθμός Καναλιων	Παραμετροι	Συναρτήσεις
1	Conv+Input	64	3x3	
2	Batch			
3	Pool		2x2	
4	Drop		0,1	
5	Conv	128	3x3	relu
6	Conv	128	3x3	relu
7	Batch			
8	Pool		2x2	
9	Drop		0,1	
10	Conv	256	3x3	relu
11	Conv	256	3x3	relu
12	Conv	256	3x3	relu
13	Conv	64	3x3	relu
14	Batch			
15	Pool		2x2	
16	Drop		0,1	
17	Flat			
18	Dense	1024		relu
19	Dense	1024		
20	Dense	70		Softmax

Εικόνα 19: Η δομή και τα επίπεδα του custom model

Στην είσοδο του μοντέλου εισάγονται οι εικόνες διαστάσεων 224 x 224 με βάθος 3 χρωμάτων(RGB). Εισάγονται τα επίπεδα συνέλιξης με τα φίλτρα τους και χρησιμοποιώντας το στρώμα BatchNormalization όπου μειώνεται ο χρόνος εκπαίδευσης ενώ παράλληλα αντιμετωπίζει την τάση του μοντέλου για υπερπροσαρμογή (overfit), ένα κύριο πρόβλημα που καλούνται να αντιμετωπίσουν τα συνελικτικά νευρωνικά δίκτυα. Στην συνέχεια περνάει από ένα επίπεδο Υπό-Δειγματοληψίας και τέλος εφαρμόζεται ένα στρώμα Dropout. Το στρώμα αυτό εκτελεί μια μέθοδο τακτοποίησης(regularization method) η οποία κατά την διάρκεια της εκπαίδευσης παραβλέπει κάποιον αριθμό εξόδων(outputs) με τυχαία σειρά. Ο αριθμός των εξόδων που θα παραβληθεί ορίζεται επί του στρώματος αυτού. Με την διαδικασία επιτυγχάνεται ακόμα περισσότερο το φαινόμενα της υπερπροσαρμογή(overfit). Όλη η παραπάνω διαδικασία αποτελεί έναν κύκλο εκπαίδευσης. Έπειτα καλείται το Flatten στρώμα όπου πολλαπλασιάζει όλες τις διαστάσεις σε έναν ενιαίο τανιστή(tensor). Τέλος υπάρχουν 3 πλήρως συνδεδεμένα

στρώμα εκ των οποίων το τελευταίο έχει έξοδο από νευρώνες όσες είναι και κλάσεις μας.

4.4. DATASET

Το dataset που χρησιμοποιήθηκε περιλαμβάνει 68.372 εικόνες ελληνικών φαγητών, οι οποίες έχουν ταξινομηθεί σε 70 διαφορετικές κατηγορίες. Το dataset αυτό είναι χωρισμένο σε δύο σύνολα, το Trainset και το Testset. Το Trainset αποτελεί το 80% του συνόλου των εικόνων, δηλαδή περιλαμβάνει 54.602 εικόνες. Το Testset αποτελεί το υπόλοιπο 20% του συνόλου, περιλαμβάνοντας 13.770 εικόνες.

Κατά τη διάρκεια της εκπαίδευσης του μοντέλου, το Trainset χρησιμοποιείται για την εξαγωγή χαρακτηριστικών από κάθε κατηγορία και την εκπαίδευση του μοντέλου. Στη συνέχεια, το Testset χρησιμοποιείται για τον έλεγχο των προβλέψεων του μοντέλου.

Στον παρακάτω πίνακα παρουσιάζονται όλες οι κατηγορίες των εικόνων και ο αριθμός των εικόνων που ανήκουν σε κάθε μια από αυτές. Οι εικόνες θα προεπεξεργαστούν έτσι ώστε να έχουν ανάλυση 224x224 pixels και βάθος 3 χρωμάτων (RGB). Τα ονόματα των κλάσεων τροποιήθηκαν σε greeklish ώστε να μπορούν να τα αναγνωρίσουν τα μοντέλα που χρησιμοποιήθηκαν και οι εντολές της python.

Παρατηρούμε στον **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**. Φαίνεται ότι υπάρχει αρκετή απόκλιση στον αριθμό των εικόνων μεταξύ των διαφορετικών κατηγοριών, κάτι που μπορεί να επηρεάσει την ακρίβεια των προβλέψεων του μοντέλου. Ωστόσο, αργότερα θα διαπιστωθεί ότι αυτή η ανισορροπία επηρεάζει ελάχιστα τη συνολική απόδοση του μοντέλου χάρη στην ικανότητα των προεκπαιδευμένων δικτύων να αντιμετωπίζουν αυτές τις ανισορροπίες.

Πίνακας 2. Κατανομή των εικόνων φαγητών στα σύνολα δεδομένων εκπαίδευσης (Trainset) και ελέγχου (Testset)

Α/α	Κατηγορία	Train	Test	Σύνολο
1	Αστακομακαρονάδα	609	153	762
2	Μπακλαβάς	862	216	1078
3	Λαχανοσαλάτα	821	206	1027
4	Καλαμάρι Σχάρας	890	223	1113
5	Κασσεροκροκέτες	849	213	1062
6	Ντάκος	859	215	1074
7	Ντολαμάδες	920	229	1149
8	Εκμέκ	692	174	866
9	Φάβα	918	230	1148
10	Γαλακτομπούρεκο	594	239	833
11	Γαρίδες	908	227	1135
12	Γαριδομακαρονάδα	856	214	1070
13	Γαύρος	926	232	1158
14	Γιαουρτλού	491	123	614
15	Γιουβέτσι	526	132	658
16	Γύρος μερίδα	904	227	1131
17	Καλαμάρι τηγανιτό	942	236	1178
18	Κανταΐφι	850	213	1063
19	Καργίοκες	263	65	328
20	Καζάν ντιπί	992	248	1240
21	Κεμπάπ	1092	273	1365
22	Κιουνεφέ	417	105	522
23	Κοκκινιστό	808	203	1011
24	Κολοκυθοκεφτέδες	456	115	571
25	Κολοκυθάκια	856	214	1070
26	Κοτόπουλο φιλέτο	1023	256	1279
27	Κουραμπιέδες	1034	259	1293
28	Λαχματζούν	992	249	1241
29	Λουκάνικο	964	241	1205
30	Λουκουμάδες	921	231	1152
31	Μανιτάρια	535	134	669
32	Μελομακάρονα	950	239	1189
33	Μύδια αχνιστά	860	216	1076
34	Μουσακάς	964	242	1206

35	Μπακαλιάρος	850	213	1063
36	Μπιφτέκι	861	216	1077
37	Μπουγιουρντί	440	110	550
38	Μπριζόλα	1066	267	1333
39	Μπρόκολο	573	144	717
40	Μυδομακαρονάδα	352	89	441
41	Ντοματοκεφτέδες	176	44	220
42	Παϊδάκια	842	211	1053
43	Πανσέτα	244	62	306
44	Παστίτσιο	834	209	1043
45	Πατάτες τηγανίτες	976	244	1220
46	Πατατοσαλάτα	800	200	1000
47	Ρεβάνι	896	224	1120
48	Ρεβυδοκεφτέδες	168	43	211
49	Σαγανάκι θαλασσινών	776	195	971
50	Σαγανάκι τυρί	1100	275	1375
51	Σαραγλί	707	176	883
52	Σαρδέλα	941	236	1177
53	Σέκερ Παρέ	798	200	998
54	Σνίτσελ	1176	295	1471
55	Σουτζούκ Λουκούμ	253	64	317
56	Σουτζουκάκια	849	213	1062
57	Σουβλάκι	1260	315	1575
58	Ταμπούλε	852	213	1065
59	Ταραμοσαλάτα	841	210	1051
60	Τηγανιά	783	196	979
61	Τσιπούρα	819	205	1024
62	Τυλιχτά	866	217	1083
63	Τζατζίκι	808	203	1011
64	Τζιγεροσαρμάς	94	23	117
65	Βαρβάρα	556	139	695
66	Χαλβάς σιμιγδαλένιος	610	153	763
67	Χωριάτικη	1075	269	1344

68	Χόρτα	765	192	957
69	Χουνκιάρ μπεγιεντί	929	233	1162
70	Χταπόδι	1122	280	1402
	Σύνολο	54602	13770	68372

Εικόνα 20. Κιουνεφέ δείγμα από το dataset

Εικόνα 21. Χταπόδι δείγμα από το dataset

ΚΕΦΑΛΑΙΟ 5

Όπως αναφέραμε στο κεφάλαιο 4 η διπλωματική εργασίας ασχολείται με το πρόβλημα της ανάπτυξης ενός νευρωνικού δικτύου για την αναγνώριση Ελληνικών φαγητών. Η μελέτη αφορά 2 φάσης. Η πρώτη ενός προεκπαιδευμένου με τη μέθοδο της μεταφοράς μάθησης (Transfer Learning) και η δεύτερη ενός "custom" νευρωνικού δικτύο.

5.1. Transfer Learning VGG16

Εισαγωγή βιβλιοθηκών

```
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D, AveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
```

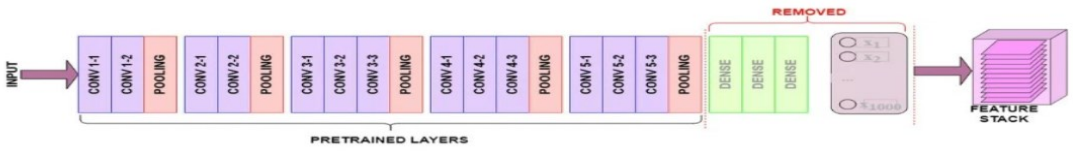
Εικόνα 22. Εισαγωγή βιβλιοθηκών

Εισαγωγή προ εκπαιδευμένου μοντέλου VGG16.

```
base_model = VGG16(weights='imagenet', include_top=False,input_shape=(224,224,3))
model_name = 'VGG16'
epoch_num = 25
```

Εικόνα 23. Εισαγωγή βιβλιοθηκών

- weights='imagenet': Αξιοποιούμαι τα βαρη(γνώση) στα συνελικτικά επιπεδα που προέρχονται από τα την εκπαίδευση στο Dataset imagenet.
- include_top=False : Εξαιρούνται τα πλήρως συνδεδεμένα επίπεδα του μόντελου.
- input_shape=(224,224,3): Το επίπεδο εισόδου



Εικόνα 24. include_top=False : Εξαιρούνται τα πλήρως συνδεδεμένα επίπεδα του μόντελου

.

Προ επεξεργασία δεδομένων εκπαίδευσης

```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
```

Εικόνα 25. Εντολές για επεξεργασία δεδομένων εκπαίδευσης

- rescale=1. / 255 : Κανονικοποίηση δεδομένων με εύρος τιμών απο 0 εως 1.
- shear_range= 0.2 : Αριστερόστροφη διάτμηση δεδομένων κατα 20%.
- zoom_range=0.2 : Μεγέθυνση/σμίκρυνση δεδομένων κατα 20%.
- horizontal_flip=True : Οριζόντια περιστροφή δεδομένων.

Προ επεξεργασία δεδομένων επικύρωσης.

```
test_datagen = ImageDataGenerator(rescale=1. / 255)
```

- rescale=1. / 255 : Κανονικοποίηση δεδομένων με εύρος τιμών απο 0 εως 1.

Ποιοτικές πληροφορίες

```
img_width, img_height = 224, 224
train_data_dir = "D:\\Johnny\\Data_Food\\train"
validation_data_dir = 'D:\\Johnny\\Data_Food\\test'
batch_size = 32
```

Εικόνα 26. πληροφορίες μοντέλου

- img_width, img_height = 224, 224 : Οι επιθυμήτες διαστάσεις εισαγωγής εικόνων.
- train_data_dir = "D:\Johnny\Data_Food\train" : Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα εκπαίδευσης.
- validation_data_dir = 'D:\Johnny\Data_Food\test' : Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα επικύρωσης.
- batch_size = 32 : Το μεγέθος των παρτιδών εισαγωγής δεδομένων.

Εισαγωγή δεδομένων εκπαίδευσης

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    color_mode='rgb',
    class_mode='categorical')
```

Εικόνα 27. Κώδικας για εισαγωγή δεδομένων εκπαίδευσης

- directory =train_data_dir: Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα τοπίκα στον υπολογιστή.
- target_size=(img_height, img_width): Οι διαστάσεις που θα γίνουν οι είκονες που βρέθηκαν στην διαδρομή.
- batch_size=batch_size : Το μεγέθος των παρτιδών εισαγωγής δεδομένων.
- color_mode='rgb': Έγχρωμες εικόνες
- class_mode='categorical' : Καθορίζει τον τύπο των ετικετών. categorical: 2-διαστάσεων πίνακας (εικόνα,κατηγορία).

Εισαγωγή δεδομένων επικύρωσης

```
validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    color_mode='rgb',
    class_mode='categorical')
```

Εικόνα 28. Εισαγωγή δεδομένων επικύρωσης

- directory = validation_data_dir: Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα τοπίκα στον υπολογιστή.
- target_size=(img_height, img_width): Οι διαστάσεις που θα γίνουν οι είκονες που βρέθηκαν στην διαδρομή.
- batch_size=batch_size : Το μεγέθος των παρτιδών εισαγωγής δεδομένων.
- color_mode='rgb': Έγχρωμες εικόνες
- class_mode='categorical' : Καθορίζει τον τύπο των ετικετών. categorical: 2-διαστάσεων πίνακας (εικόνα,κατηγορία)

Ποιοτικές πληροφορίες

```
nb_train_samples = train_generator.n
nb_validation_samples = validation_generator.n
n_classes = train_generator.num_classes
```

- nb_train_samples = train_generator.n : Το πλήθος των εικόνων απο τα δεδομένα εκπαίδευσης.
- nb_validation_samples = validation_generator.n : Το πλήθος των εικόνων απο τα δεδομένα επικύρωσης.
- n_classes = train_generator.num_classes : Το πλήθος των κλάσεων που υπολογίζεται από την δομή του συνόλου εκπαίδευσης από την βιβλιοθήκη.

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512,activation='relu')(x)
x = Dropout(0.2)(x)
```

Εικόνα 29. Σχεδιασμός της εξόδου του μοντέλου

- x = base_model.output : Το επίπεδο εισόδου και τα κρυφά επίπεδα από το VGG16.
- x = GlobalAveragePooling2D()(x) : Ενα επίπεδο Υπό-Δειγματοληψίας.
- x = Dense(512,activation='relu')(x): Επίπεδο με 512 νευρώνες πλήρως συνδεδεμένους με συνάρτηση ενεργοποίησης relu.
- x = Dropout(0.2)(x) : Επίπεδο όπου βάζει σε τυχαίες θέσεις το μηδέν(για αποφυγή υπερπροσαρμογής).

Επίπεδο εξόδου

```
predictions = Dense(n_classes,
                    kernel_regularizer=regularizers.l2(0.005),
                    activation='softmax')(x)
```

Εικόνα 30. Τελευταίο επίπεδο εξόδου

- predictions = Dense(n_classes,kernel_regularizer=regularizers.l2(0.005),activation='softmax')(x): Επιπεδό με αριθμό νευρώνων οσές είναι οι κλάσεις πυκνά συνδεδεμένο με συνάτρηση ενεργοποίησης softmax και ρυθμιστές που εφαρμόζουν ποινές στις παραμέτρους(για βελτιστοποίηση).

Το μοντέλο και αριθμός βαρών του κάθε επιπέδου

```
model = Model(inputs=base_model.input, outputs=predictions)
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (G lobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 512)	262656
dropout (Dropout)	(None, 512)	0

dense_1 (Dense) (None, 70) 35910

==========

Total params: 15,013,254

Trainable params: 15,013,254

Non-trainable params: 0

Οι παράμετροι εκπαίδευσεις:

```
model.compile(optimizer=SGD(learning_rate=0.0001, momentum=0.9),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

- optimizer=SGD(learning_rate=0.0001, momentum=0.9) : Ο βελτιστοποιητής Stochastic gradient descent.
- loss='categorical_crossentropy' : Συνάρτηση κόστους.
- metrics=['accuracy']: Μετρική για την αποτίμηση του μοντέλου.

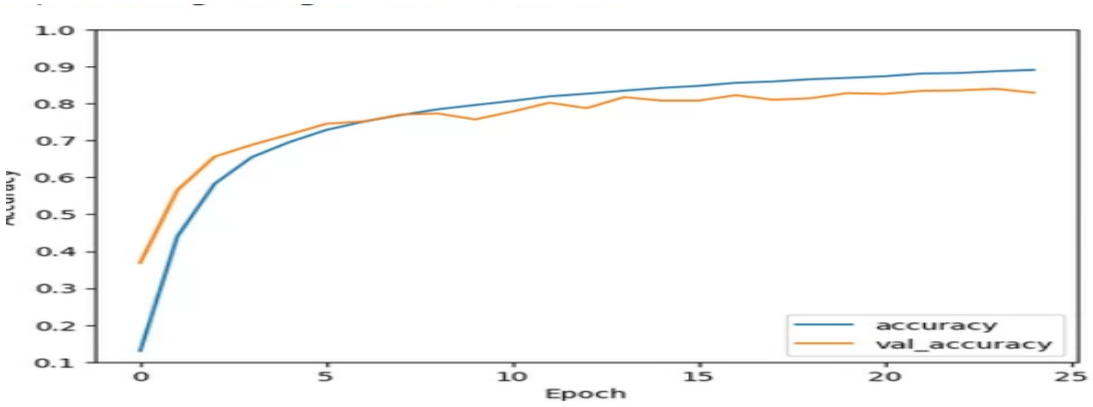
Εκπαίδευση

```
hist = model.fit(train_generator,
                         steps_per_epoch = nb_train_samples // batch_size,
                         validation_data = validation_generator,
                         validation_steps = nb_validation_samples // batch_size,
                         epochs = epoch_num,
                        )
```

```
ccuracy: 0.8142
Epoch 20/25
1717/1717 [==============================] - 1551s 903ms/step - loss: 0.7529 - accuracy: 0.8697 - val_loss: 0.9007 - val_a
ccuracy: 0.8283
Epoch 21/25
1717/1717 [==============================] - 1567s 913ms/step - loss: 0.7279 - accuracy: 0.8740 - val_loss: 0.9007 - val_a
ccuracy: 0.8262
Epoch 22/25
1717/1717 [==============================] - 1588s 925ms/step - loss: 0.6984 - accuracy: 0.8812 - val_loss: 0.8637 - val_a
ccuracy: 0.8344
Epoch 23/25
1717/1717 [==============================] - 1557s 907ms/step - loss: 0.6790 - accuracy: 0.8829 - val_loss: 0.8665 - val_a
ccuracy: 0.8357
Epoch 24/25
1717/1717 [==============================] - 1554s 905ms/step - loss: 0.6554 - accuracy: 0.8876 - val_loss: 0.8281 - val_a
ccuracy: 0.8400
Epoch 25/25
1717/1717 [==============================] - 1582s 921ms/step - loss: 0.6350 - accuracy: 0.8912 - val_loss: 0.8579 - val_a
ccuracy: 0.8294
```

- train_generator: Δεδομένα εκπαιδευσής
- steps_per_epoch = nb_train_samples // batch_size : Το μέγεθος των παρτίδων.
- validation_data = validation_generator : Δεδομένα επικύρωσης.
- epochs = epoch_num : Ποσες φόρες θα επαναλάβη την διαδικασία.

```
plt.plot(hist.history['accuracy'], label='accuracy')
plt.plot(hist.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.1, 1])
plt.legend(loc='lower right')
```



Εικόνα 31. Απόδοση μοντέλου VGG16

Το προεκπαιδευμένο μοντέλο VGG16παρουσιάζει ακρίβεια με ποσοστό περίπου 80% μετά από 24 epochs στο σύνολο δεδομένων ελέγχου και 87% στο σύνολο δεδομένων εκπαίδευσης. Το αποτέλεσμα αυτό δηλώνει επαρκή εκπαίδευση του μοντέλου VGG16 με αρχική χρήση των προεκπαιδευμένων βαρών του, καθώς και επαρκή απόδοση του σε άγνωστα δεδομένα.

5.2. CUSTOM MODEL

Εισαγωγή βιβλιοθηκών

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Rescaling
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.preprocessing.image import array_to_img
from tensorflow.keras import optimizers
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import array_to_img
import random
import os
import shutil
```

Εικόνα 32. Εισαγωγή βιβλιοθηκών CUSTOM MODEL

Ποιοτηκές Πληροφορίες

```
data_dir= "D:\\Johnny\\Data_Food"
train_data_dir_2 = "D:\\Johnny\\Data_Food\\train"
test_data_dir_2 = 'D:\\Johnny\\Data_Food\\test'
img_width, img_height = 224, 224
batch_size = 32
```

```
train_data_dir = train_data_dir_2
val_data_dir = test_data_dir_2
```

- img_width, img_height = 224, 224 : Οι επιθυμήτες διαστάσεις εισαγωγής εικόνων.
- train_data_dir = "D:\Johnny\Data_Food\train" : Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα εκπαίδευσης.
- validation_data_dir = 'D:\Johnny\Data_Food\test' : Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα επικύρωσης.
- batch_size = 32 : Το μεγέθος των παρτιδών εισαγωγής δεδομένων.

Προ επεξεργασία δεδομένων εκπαίδευσης /επικύρωσης

```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1. / 255)
```

Εικόνα 33.Προ επεξεργασία δεδομένων εκπαίδευσης /επικύρωσης

Για τα δεδομένα εκπαίδευσης

- rescale=1. / 255 : Κανονικοποίηση δεδομένων με εύρος τιμών απο 0 εως 1.
- shear_range= 0.2 : Αριστερόστροφη διάτμηση δεδομένων κατα 20%.
- zoom_range=0.2 : Μεγέθυνση/σμίκρυνση δεδομένων κατα 20%.
- horizontal_flip=True : Οριζόντια περιστροφή δεδομένων.

Για τα δεδομένα επικύρωσης

- rescale=1. / 255 : Κανονικοποίηση δεδομένων με εύρος τιμών απο 0 εως 1.

Εισαγωγή δεδομένα εκπαίδευσης

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    color_mode='rgb',
    class_mode='categorical')
```

Εικόνα 34. Εισαγωγή δεδομένα εκπαίδευσης

- directory =train_data_dir: Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα τοπίκα στον υπολογιστή.
- target_size=(img_height, img_width): Οι διαστάσεις που θα γίνουν οι είκονες που βρέθηκαν στην διαδρομή.
- batch_size=batch_size : Το μεγέθος των παρτιδών εισαγωγής δεδομένων.
- color_mode='rgb': Έγχρωμες εικόνες
- class_mode='categorical' : Καθορίζει τον τύπο των ετικετών. categorical: 2-διαστάσεων πίνακας (εικόνα,κατηγορία).

Εισαγώγη δεδομένων επικύρωσης

```
validation_generator = val_datagen.flow_from_directory(
    val_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    color_mode='rgb',
    class_mode='categorical')
```

- directory = val_data_dir: Η διαδρομή όπου ειναι αποθηκευμένα τα δεδομένα τοπίκα στον υπολογιστή.
- target_size=(img_height, img_width): Οι διαστάσεις που θα γίνουν οι είκονες που βρέθηκαν στην διαδρομή.
- batch_size=batch_size : Το μεγέθος των παρτιδών εισαγωγής δεδομένων.
- color_mode='rgb': Έγχρωμες εικόνες
- class_mode='categorical' : Καθορίζει τον τύπο των ετικετών. categorical: 2-διαστάσεων πίνακας (εικόνα,κατηγορία)

Το μοντέλο / παράμετροι εκπαίδευσης

```
model = Sequential()
model.add( Conv2D(64, (3, 3), activation='relu', input_shape=(img_height, img_width, 3) ) )
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(70, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Εικόνα 35. Μοντέλο custom/ παράμετροι εκπαίδευσης

- optimizer=Adam(learning_rate=0.0001) : Ο βελτιστοποιητής

- loss='categorical_crossentropy' : Συνάρτηση κόστους.
- metrics=['accuracy']: Μετρική για την αποτίμηση του μοντέλου.

Ο αριθμός βαρών του κάθε επιπέδου

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 64)	1792
batch_normalization (BatchN ormalization)	(None, 222, 222, 64)	256
max_pooling2d (MaxPooling2D)	(None, 111, 111, 64)	0
dropout (Dropout)	(None, 111, 111, 64)	0
conv2d_1 (Conv2D)	(None, 109, 109, 128)	73856
conv2d_2 (Conv2D)	(None, 107, 107, 128)	147584
batch_normalization_1 (Batc hNormalization)	(None, 107, 107, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 53, 53, 128)	0
dropout_1 (Dropout)	(None, 53, 53, 128)	0
conv2d_3 (Conv2D)	(None, 51, 51, 256)	295168
conv2d_4 (Conv2D)	(None, 49, 49, 256)	590080
conv2d_5 (Conv2D)	(None, 47, 47, 256)	590080
conv2d_6 (Conv2D)	(None, 45, 45, 16)	36880
batch_normalization_2 (BatchNormalization)	(None, 45, 45, 16)	64
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 16)	0
dropout_2 (Dropout)	(None, 22, 22, 16)	0
flatten (Flatten)	(None, 7744)	0
dense (Dense)	(None, 1024)	7930880
dense_1 (Dense)	(None, 1024)	1049600
dense_2 (Dense)	(None, 70)	71750

Total params: 10,788,502

Trainable params: 10,788,086

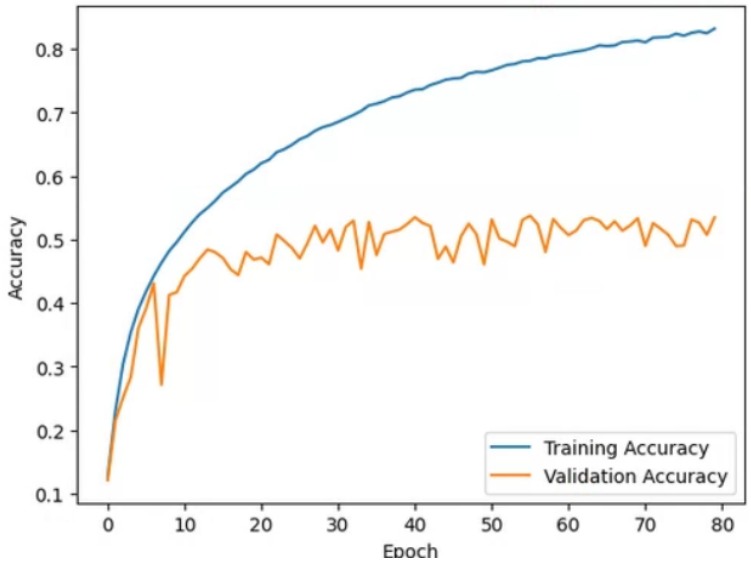
Non-trainable params: 416

Αριθμός εποχών/ εκπαίδευση

```
epochs = 80
```

```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples//batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples//batch_size
)
```

```
430/430 [==============================] - 325s 757ms/step - loss: 2.7493 - accuracy: 0.5346
Test loss: 2.7493, Test accuracy: 0.5346
```



Εικόνα 36. Απόδοση μοντέλου Custom

Το μοντέλο custom παρουσιάζει ακρίβεια με ποσοστό περίπου 55% μετά από 80 epochs στο σύνολο δεδομένων ελέγχου και 87% στο σύνολο δεδομένων εκπαίδευσης. Το αποτέλεσμα αυτό δηλώνει υπερεκπαίδευση του μοντέλου που αναπτύχθηκε και φτωχή απόδοση του μοντέλου σε άγνωστα δεδομένα.

5.3. Παράρτημα Α. ΒΟΗΘΗΤΙΚΟ SCRIPT

Η χρήση του παρακάτω script είναι σε ένα πλήρης dataset να γίνει διαχωρισμός του σε dataset εκπαίδευσης και σε dataset επικύρωσης σε ποσοστιαία αναλογία. Αυτό που χρειάζεται είναι να δημιουργηθούν δυο κενοί φάκελοι μέσα στο dataset ένας για τα δεδομένα εκπαίδευσης και ένας για τα δεδομένα επικύρωσης.

Εισαγωγή βιβλιοθηκών

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import array_to_img
import random
import os
import shutil
```

Πίνακας 3. Ο αλγόριθμος διαχωρισμού του dataset

```
main_data_folder = "D:\\Johnny\\Data_Food"
train_data_dir = "D:\\Johnny\\Data_Food\\train"
val_data_dir = "D:\\Johnny\\Data_Food\\test"

validation_ratio = 0.25
all_image_paths = []
for class_folder in os.listdir(main_data_folder):
    class_path = os.path.join(main_data_folder, class_folder)
    image_files = os.listdir(class_path)
    image_paths = [os.path.join(class_path, file) for file in image_files]
    all_image_paths.extend(image_paths)

random.shuffle(all_image_paths)


num_validation = int(validation_ratio * len(all_image_paths))

train_image_paths = all_image_paths[num_validation:]
val_image_paths = all_image_paths[:num_validation]

for image_path in train_image_paths:
    class_folder = os.path.basename(os.path.dirname(image_path))
    dest_folder = os.path.join(train_data_dir, class_folder)
    os.makedirs(dest_folder, exist_ok=True)
    shutil.copy(image_path, os.path.join(dest_folder, os.path.basename(image_path)))

for image_path in val_image_paths:
    class_folder = os.path.basename(os.path.dirname(image_path))
    dest_folder = os.path.join(val_data_dir, class_folder)
    os.makedirs(dest_folder, exist_ok=True)
    shutil.copy(image_path, os.path.join(dest_folder, os.path.basename(image_path)))
```

- Στην μεταβλητή validation_ratio ορίζεται το ποσοστό διαχωρισμού του dataset.

ΚΕΦΑΛΑΙΟ 6

Συμπεράσματα

Στην εργασία αυτή παρουσιάστηκαν και επεξηγήθηκαν οι βασικές αρχές της Μηχανικής Μάθησης και η ταύτιση τους με το βιολογικό νευρωνικό δίκτυο. Στη συνέχεια, αναλύθηκαν διεξοδικά τα συνελεκτικά νευρωνικά δίκτυα, τα οποία αποτελούν τμήμα της βαθιάς μηχανικής μάθησης και ακολουθήθηκε η παρουσίαση των εργαλείων και των βιβλιοθηκών που απαιτούνται για την υλοποίηση των αλγορίθμων. Αναπτύχθηκαν δυο μοντέλα βαθιάς μηχανικής μάθησης, το εκ των δυο βασίστηκε στην μέθοδο του Transfer Learning, ενώ το δεύτερο μοντέλο αναπτύχθηκε και μελετήθηκε η δυνατότητα να εκπαιδευτεί επαρκώς και να συγκριθεί η απόδοσή του με το προεκπαιδευμένο μοντέλο. Τα δύο αυτά μοντέλα εφαρμόσθηκαν σε ένα σύνολο δεδομένων με εικόνες από ελληνικά φαγητά. Υστέρα από την εφαρμογή των δυο αυτών αλγορίθμων σε αυτό το σύνολο δεδομένων, προέκυψε το συμπέρασμα ότι η το μοντέλο VGG16 αποδίδει με καλύτερη ακρίβεια σε σχέση με το μοντέλο custom με ποσοστό ακρίβειας 80% στο σύνολο δεδομένων ελέγχου και 86% στο σύνολο δεδομένων εκπαίδευσης. Ενώ το μοντέλο custom παρουσίασε ακρίβεια με ποσοστό 55% στο σύνολο δεδομένων ελέγχου και 87% στο σύνολο δεδομένων εκπαίδευσης που δηλώνει υπερεκπαίδευση του μοντέλου.

Βιβλιογραφία

Αργυράκης Π. (2001). *Νευρωνικά Δίκτυα και Εφαρμογές*, Πάτρα: Ελληνικό Ανοιχτό Πανεπιστήμιο (Ε.Α.Π).

Βλαχάβας Ι., Κεφάλας Π., Βασιλειάδης Ν., Κοκκορας Φ. & Σακελλαρίου Η., (2006). Τεχνητή Νοημοσύνη, Γ' Έκδοση, Θεσσαλονίκη:ΑΠΘ.

Χαράλαμπος Π. Στρουθόπουλος (2014). Σημειώσεις «Αναγνώριση Προτύπων – Νευρωνικά Δίκτυα», Τεχνολογικό Εκπαιδευτικό Ίδρυμα Σερρών, Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Μηχανικών Πληροφορικής.

Διαμαντάρας, Κ. (2007) *Τεχνητά Νευρωνικά Δίκτυα*, Αθήνα:Κλειδάριθμος.

David, K., Elmirghani, J., Haas, H. and You, X.H. (2019). Defining 6G: Challenges and opportunities [from the guest editors]. IEEE Vehicular Technology Magazine, 14(3), pp.14-16.

Hagan, M.T., Demuth, H.B. and Beale, M. (1996). Neural Network Design. PWS Publishing Co., Boston.

Haykin, S. (1999). Neural Networks a Comprehesive Foundation, Second Edition. Ontario, Canada:McMaster University.

M. Jarke, M.A. Jeusfeld, C. Quix, P. Vassiliadis. (1998). Architecture and quality in data warehouses. In Proc. 10th Conference on Advanced Information Systems Engineering (CAiSE '98), pp. 93-113, Pisa, Italy.

M. Jarke, Y. Vassiliou. (1997). Foundations of data warehouse quality – a review of the DWQ project. In Proc. 2nd Intl. Conference

Information Quality (IQ-97), pp. 299-313, Cambridge, Mass., USA.

Kasabov,N. (1996). Foundations of neural networks, fuzzy systems, and knowledge engineering, Marcel Alencar.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). "Deep learning", Nature, vol. 521, No. 7553, 436–444.

Nayak, S. and Patgiri, R., 2021. 6G communication technology: A vision on intelligent healthcare. In Health Informatics: A Computational Perspective in Healthcare (pp. 1-18). Springer, Singapore.

Openshaw S. and Openshaw C. (1997). Artificial Intelligence in Geography, England : John Wiley & Sons Ltd.

K. Orr. Data quality and systems theory. In Communications of the ACM, 41(2), pp. 54-57, Feb. 1998.

Panksepp, J. (2004). Affective neuroscience: The foundations of human and animal emotions, Oxford university press.

Tang, F., Kawamoto, Y., Kato, N. and Liu, J., 2019. Future intelligent and secure vehicular network toward 6G: Machine-learning approaches. Proceedings of the IEEE, 108(2), pp.292-307.

Torrey L. and Shavlik, J. (2010). "Transfer learning," in Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, pp. 242–264.

Xiao, Y., Shi, G., Li, Y., Saad, W. and Poor, H.V., 2020. Toward Self-Learning Edge Intelligence in 6G. IEEE Communications Magazine, 58(12), pp.34-40.

Ηλεκτρονικές πηγές

https://www.anaconda.com/download , ανακτήθηκε στις 6-11-2023.

https://developers.google.com/machine-learning/, ανακτήθηκε στις 6-11-2023.

https://jupyter.org/, ανακτήθηκε στις 6-11-2023.

https://matplotlib.org/, ανακτήθηκε στις 7-11-2023.

https://numpy.org/, ανακτήθηκε στις 7-11-2023.

https://pypi.org/project/pandas/, ανακτήθηκε στις 7-11-2023.

https://docs.python.org/3/tutorial/index.html, ανακτήθηκε στις 6-11-2023.