ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ &
ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΩΝ ΚΑΙ ΑΝΑΛΟΓΙΣΤΙΚΩΝ
ΜΑΘΗΜΑΤΙΚΩΝ

Νευρωνικά Δίκτυα σε Μεγάλα Δεδομένα

Μεταπτυχιακή Διατριβή

Επιμέλεια: Μαυρόγιαννης Ιωάννης

Επιβλέπων καθηγητής: Ζήμερας Στέλιος

Σάμος, Φεβρουάριος 2022

Τριμελής Επιτροπή

Ζήμερας Στέλιος, Αναπληρωτής Καθηγητής

Καραγρηγορίου Αλέξανδρος, Καθηγητής

Ξανθόπουλος Στέλιος, Αναπληρωτής Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κύριο Ζήμερα Στέλιο, αναπληρωτή καθηγητή του τμήματος Στατιστικής και Αναλογιστικών – Χρηματοοικονομικών Μαθηματικών (ΣΑΧΜ), για την στήριξη και τις συζητήσεις του κατά τη διάρκεια της εκπόνησης της μεταπτυχιακής διατριβής.

Επίσης, θα ήθελα να ευχαριστήσω τον κύριο Καραγρηγορίου Αλέξανδρο, καθηγητή του τμήματος ΣΑΧΜ, για την αδιάκοπη πίστη του στις ικανότητές μου.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου και όλους όσους στάθηκαν δίπλα μου.

Πίνακας Περιεχομένων

Περίληψη

Τα μεγάλα δεδομένα (Big Data) αποτελούν μία σύγχρονη μορφή πλούτου. Η δημιουργία τους οφείλεται σε πηγές οι οποίες έχουν εν γένη μεγαλύτερη πολυπλοκότητα από αυτή των παραδοσιακών δεδομένων. Έχουν τη δυνατότητα να προωθήσουν την επιστήμη και τις στρατηγικές των επιχειρήσεων, με εφαρμογές από πρόβλεψη ταινιών στο Netflix και την προβολή εξατομικευμένων διαφημίσεων στο YouTube, μέχρι την έγκαιρη διάγνωση καρκινωμάτων και την κατασκευή ψηφιακών διδύμων. Ο συνδετικός κρίκος αυτών των υπηρεσιών με τα μεγάλα δεδομένα, είναι τα νευρωνικά δίκτυα, τα οποία δίχως να είναι ρητά προγραμματισμένα εξάγουν αυτόματα κανόνες με τους οποίους κάνουν προβλέψεις, με βάση τα μοτίβα που υπάρχουν στα δεδομένα. Σε όσα ακολουθούν, παρουσιάζουμε την σύνδεση των νευρωνικών δικτύων και των μεγάλων δεδομένων. Χωρίζουμε το πρώτο κεφάλαιο σε τρεις ενότητες, οι οποίες αποτελούν το θεωρητικό υπόβαθρο όσων θα αναφέρουμε αργότερα. Οι τρεις ενότητες περιλαμβάνουν πρώτα τα μεγάλα δεδομένα με τα χαρακτηριστικά και τις εφαρμογές τους, μετά την μηχανική μάθηση με τις κατηγορίες και την εκπαίδευση των αλγορίθμων της, και τελικά τα νευρωνικά δίκτυα, όπου παρουσιάζεται η βιολογική τους έμπνευση και αναπαράσταση, καθώς και οι έννοιες της συνάρτησης ενεργοποίησης και της συνάρτησης απώλειας. Στο δεύτερο κεφάλαιο, εμβαθύνουμε στις δομές των νευρωνικών δικτύων όπως αυτά των πολυστρωματικών αντιλήπτρων (MLP), τα συνελικτικά (CNN) και τα επαναλαμβανόμενα (RNN), ενώ επιπλέον περιγράφεται η διαδικασία της μάθησης σε αυτά. Έπειτα, στο τρίτο κεφάλαιο παρουσιάζουμε εφαρμογές των νευρωνικών δικτύων μέσω της χρήσης του λογισμικού της R και του πακέτου keras, όπου γίνεται αντιληπτή η πρακτική διαδικασία της εκπαίδευσης των νευρωνικών δικτύων, καθώς και οι δυσκολίες που αυτή περιέχει. Συνεπώς, στο τέταρτο κεφάλαιο είναι πλέον εφικτό να μιλήσουμε για την σύνδεση των μεγάλων δεδομένων με τα νευρωνικά δίκτυα, όπου αναφέρουμε κάποιες σύγχρονες εφαρμογές τους και έπειτα κάποιες δυσκολίες που δημιουργούνται κατά την επαφή τους. Τελικά, αναφέρουμε τα συμπεράσματα και τις μελλοντικές κατευθύνσεις που θεωρούμε ότι πρέπει να ακολουθήσουν οι ερευνητές που ψάχνουν καινοτόμες προσεγγίσεις στα μεγάλα δεδομένα με την χρήση νευρωνικών δικτύων.

Λέξεις – Κλειδιά: Μεγάλα Δεδομένα, Νευρωνικά Δίκτυα, Χαρακτηριστικά Μεγάλων Δεδομένων, Μηχανική Μάθηση, MLP, DNN, CNN, RNN.

Abstract

Big data make up the modern world's wealth. The sources from which they are generated have greater variability than that of classical data. They are able to propel forward science and the strategies of businesses, with applications ranging from the Netflix movie recommendation system and the YouTube personalized ad recommendation system, to the early diagnosis of cancer and the making of digital twins. What connects these services and big data, are the neural networks, that are able to mine rules and make predictions from patterns in the data automatically, without being explicitly programmed. In the following thesis, we will be presenting the connection of neural networks with big data. We are splitting the first chapter in three sections, which cover the theoretical background for what will be covered afterwards. These sections first include big data with its characteristics and applications, then machine learning alongside the categories, and the meaning of, "learning" in its algorithms, and lastly the neural networks, with a representation of their biological inspiration and also some common terms like the activation function and loss function. In the second chapter, we deepen into the neural network architectures of multilinear perceptrons (MLP), convolutional neural networks (CNN) and recurrent neural networks (RNN), while also detailing the learning process for each of them. Furthermore, in the third chapter we showcase some applications of neural networks through the use of the R programming language and the keras package, while the practical process of training neural networks, as well as the problems that arise, become transparent. Afterwards, in the fourth chapter, we are able to talk about the connection between neural networks and big data, where we mention some of the modern applications and the difficulties that happen at their conjunction. Finally, we give the concluding remarks and the future directions that we consider worthy for researchers that are pursuing a novel approach into big data, through neural networks.

Key – Words: Big Data, Neural Networks, Big Data Characteristics, Machine Learning, MLP, DNN, CNN, RNN.

Κεφάλαιο 1.
Θεωρητικό Πλαίσιο

Ενότητα 1
Μεγάλα Δεδομένα

1.1.1 Εισαγωγή

Τα μεγάλα δεδομένα (Big Data) εάν εκμεταλλευτούν, αποτελούν μία σύγχρονη μορφή πλούτου. Η δημιουργία τους οφείλεται σε πηγές οι οποίες έχουν εν γένη μεγαλύτερη πολυπλοκότητα από αυτή των παραδοσιακών δεδομένων, καθώς δημιουργούνται από τις δραστηριότητες του ανθρώπου στο διαδίκτυο και την αλληλεπίδρασή του με εφαρμογές. Αυτό έχει οδηγήσει στην έννοια του *Διαδικτύου των Πραγμάτων (Internet of Things)*, η οποία περιλαμβάνει συσκευές που είναι εξοπλισμένες με ανιχνευτές, οι οποίοι επιτρέπουν την σύνδεση στο διαδίκτυο και την ανταλλαγή πληροφοριών με βάση την χρήση τους. Επιπλέον, οι παραδοσιακές μέθοδοι αποθήκευσης σε σχεσιακές βάσεις δεδομένων και αξιοποίησης των δεδομένων με τις κλασικές μεθόδους ανάλυσης, δεν μπορούν να εφαρμοστούν για την άντληση πληροφορίας, λόγο του τεράστιου κόστους και των ιδιαίτερων χαρακτηριστικών των μεγάλων δεδομένων. Τα μεγάλα δεδομένα καθίστανται, συνεπώς, *μη – αξιοποιήσιμα* χωρίς την σωστή επεξεργασία και διαχείριση.
Όμως, εάν η διαχείριση γίνει με τις κατάλληλες μεθόδους, τα μεγάλα δεδομένα προσφέρουν αποτελέσματα τα οποία δεν μπορούν να συναγωνιστούν τα παραδοσιακά δεδομένα. Οι εφαρμογές των μεγάλων δεδομένων συνεισφέρουν στην πρόοδο της επιστήμης και των μεγάλων πολυεθνικών εταιριών, καθώς οι περισσότερες πλέον στρέφονται στην αξιοποίηση των παραγόμενων δεδομένων για την βελτίωση της εμπειρίας του καταναλωτή και την εύρεση της βέλτιστης στρατηγικής πωλήσεων. Σύμφωνα με την Oracle (2021) οι τομείς περιλαμβάνουν επίσης τα ακόλουθα:

- **Ανάπτυξη προϊόντων**
  Σε εταιρίες οι οποίες επιθυμούν να προτείνουν υπηρεσίες ή προϊόντα τους με αυτόματο τρόπο, ανάλογα την προηγούμενη συμπεριφορά του χρήστη, μπορούν να αξιοποιήσουν τα μεγάλα δεδομένα σε συνδυασμό με αλγόριθμους τεχνητής νοημοσύνης για να προβλέψουν την ζήτηση ατόμων με παρόμοιο ιστορικό. Για παράδειγμα το Netflix, το YouTube χρησιμοποιούν τέτοιες διαδικασίες για την πρόταση βίντεο σε χρήστες, ενώ εταιρίες όπως η P&G χρησιμοποιούν αναλύσεις σε μεγάλα δεδομένα για να ξεκινήσουν νέες καμπάνιες προϊόντων.
- **Πρόβλεψη αναγκών συντήρησης**
  Μέσω των μεγάλων δεδομένων, είναι εφικτό να εξορύξουμε γνώση για τους παράγοντες οι οποίοι οδηγούν τελικά σε βλάβες ενός προϊόντος. Αυτοί οι παράγοντες περιλαμβάνουν για παράδειγμα το έτος κατασκευής, τον κατασκευαστή, το μοντέλο, τα υλικά που χρησιμοποιήθηκαν. Ως αποτέλεσμα, είναι δυνατή η πρόβλεψη μίας βλάβης προτού συμβεί.
- **Βελτίωση εμπειρίας χρηστών**
  Τα μεγάλα δεδομένα μπορούν να αξιοποιηθούν και για την βελτίωση εμπειρίας χρηστών, όπως για παράδειγμα στις εφαρμογές των μέσων κοινωνικής δικτύωσης, όπου πληροφορίες από εκατομμύρια χρήστες μπορούν να αξιοποιηθούν ώστε να προτείνονται εξατομικευμένες εμπειρίες στην πλατφόρμα, συγκρίνοντας την συμπεριφορά του χρήστη με άλλων χρηστών που πραγματοποίησαν παρόμοιες ενέργειες.
- **Αναγνώριση απόπειρας απάτης / μηνύματος spam**
  Με παρόμοιο τρόπο, μέσω των μεγάλων δεδομένων είναι εφικτό να αναγνωριστούν απόπειρες απάτης και μηνύματα spam και να κατανεμηθούν καταλλήλως σε φακέλους, προειδοποιώντας τον χρήστη. Αυτό για παράδειγμα συμβαίνει σε υπηρεσίες e – mail όπως το Gmail και το Outlook.

Ενώ, σύμφωνα με την IBM (2020), τομείς εφαρμογών των μεγάλων δεδομένων συμπεριλαμβάνουν, αλλά δεν περιορίζονται, στους ακόλουθους:

- **Σύστημα Υγείας**
  Από το 2020 ο Covid – 19 έκανε παγκοσμίως γνωστή την ανάγκη για την γρήγορη και αξιόπιστη συλλογή και ανάλυση των δεδομένων. Μαζί με αυτήν όμως, οι φαρμακευτικές και ιατρικές εταιρίες και οργανισμοί έρχονται αντιμέτωποι με δεδομένα που συνεχώς παράγονται, τα οποία έρχονται σε διαφορετικές δομές, με απρόσμενο όγκο και ταχύτητα.
- **Οικονομικές υπηρεσίες**
  Τα μεγάλα δεδομένα είναι το επόμενο βήμα για τις εταιρείες, ενώ υπάρχουν περισσότερες ευκαιρίες για τις εταιρίες που θα καταφέρουν πρώτες να υλοποιήσουν μεθόδους τεχνητής νοημοσύνης και μηχανικής μάθησης για την αξιοποίηση των μεγάλων δεδομένων, σε σύγκριση με τις υπόλοιπες. Συγκεκριμένα, προβλέπεται πως η αξιοποίηση των μεγάλων δεδομένων θα μειώσει τα κόστη και θα συνεισφέρει σε υλοποίηση των αναγκών πελατών και προσωπικού.
- **Τηλεπικοινωνίες**
  Η διαχείριση των δεδομένων στις τηλεπικοινωνίες πλέον δεν αφορά μόνο παραδοσιακά δεδομένα όπως τα δεδομένα λογαριασμών, μάρκετινγκ και δικτύου, αλλά και δεδομένα από το ίντερνετ των πραγμάτων από όπου οι συνδεδεμένες συσκευές παρέχουν συνεχή δεδομένα από την αλληλεπίδραση του χρήστη, μέχρι και οι εικόνες και τα βίντεο που παρακολουθεί.

Εφόσον αναφέραμε κάποια πεδία στα οποία συναντάμε τα μεγάλα δεδομένα, θα αναφέρουμε τα χαρακτηριστικά που ξεχωρίζουν τα «μεγάλα δεδομένα» από τα παραδοσιακά δεδομένα.

1.1.2 Χαρακτηριστικά Μεγάλων Δεδομένων

Σχήμα 1. Τα «5V» των μεγάλων δεδομένων. *Αξιοποιήθηκε η ιστοσελίδα canva.com*

Τα μεγάλα δεδομένα ξεχώρισαν από τα «παραδοσιακά» δεδομένα, μέσα από τα ακόλουθα 3 χαρακτηριστικά τους, τα οποία αποκαλούνται ως τα «3 V» (Taleb et al. 2018):

- **Όγκος (Volume)**
  Ο όγκος αναφέρεται στο μέγεθος των μεγάλων δεδομένων (Vijendra, 2016), το οποίο έχει αυξηθεί πολύ τα τελευταία χρόνια, ξεπερνώντας την δυνατότητα αποθήκευσης που προσφέρουν οι τεχνολογίες αποθήκευσης σε σκληρούς δίσκους. Σύμφωνα με την έρευνα του See (2021) στην

ιστοσελίδα statista.com, η εκτίμηση των δεδομένων που δημιουργήθηκαν, αντιγράφηκαν και μεταφέρθηκαν ξεπερνάει τα 64.2 zettabyte (*1 zettabyte ισούται με 1 τρισεκατομμύριο gigabyte*), ενώ αναμένεται να ξεπεράσει τα 180 zettabyte έως το 2025. Επιπλέον, αναφέρεται πως η δυνατότητα αποθήκευσης το 2020 ήταν 6.7 zettabyte, η οποία προβλέπεται πως θα παρουσιάσει αύξηση κατά 19.2% κατά την περίοδο 2020 – 2025. Αυτό είναι που έχει οδηγήσει τα «μεγάλα δεδομένα» στην ονομασία τους. Για παράδειγμα, σύμφωνα με την έρευνα του Ceci (2021) στην ιστοσελίδα statista.com, τον Φεβρουάριο του 2020 στην πλατφόρμα βίντεο του YouTube οι χρήστες ανέβαζαν 500 ώρες βίντεο ανά λεπτό, ενώ το 27% των χρηστών δήλωσε πως παρακολούθησε πάνω από 10 ώρες βίντεο κάθε εβδομάδα.

- **Ταχύτητα (Velocity)**
  Η ταχύτητα είναι ο ρυθμός με τον οποίο παράγονται τα δεδομένα, αλλά μπορεί να συμπεριλαμβάνει και την ταχύτητα αξιοποίησης που έχουν τα δεδομένα. Έχει δύο συντελεστές: την ταχύτητα παραγωγής και την ταχύτητα αξιοποίησης (Geczy, 2014). Επιπλέον, η παραγωγή των δεδομένων μπορεί να γίνεται (α) σε πραγματικό χρόνο, (β) σχεδόν πραγματικό χρόνο, σε (γ) τεμάχια δεδομένων που φτάνουν την ταυτόχρονα (batch time) και σε (δ) ροή (steam).
  Η διαδικασία των χρονικών τεμαχίων ήταν από τις πρώτες που καθιερώθηκαν και, μέσω χρήσης του λογισμικού Hadoop και του Map-Reduce μοντέλου, γινόταν καταγραφή των δεδομένων στον σκληρό δίσκο. Το αποτέλεσμα της επεξεργασίας γινόταν διαθέσιμο στο τέλος της επεξεργασίας.
  Τα δεδομένα που παράγονται σε ροή, γίνονται σταδιακά διαθέσιμα. Το λογισμικό Spark προτιμάται έναντι του Hadoop (Shahrivari, 2014).
  Σχεδόν πραγματικό χρόνο χρειάζονται τα δεδομένα των διαφημίσεων στις ιστοσελίδες, ενώ δεδομένα με χρονικά τεμάχια αφορούν δεδομένα που μαζεύονται ανά τακτά χρονικά διαστήματα όπως το λιανεμπόριο, η ιατροδικαστική και η βιοπληροφορική. Δεδομένα ροής παράγονται από την τοποθεσία ενός ανθρώπου, από καιρικά φαινόμενα, από την ροή του Instagram, του Facebook, του Twitter και άλλων.
- **Ποικιλομορφία (Variety)**
  Η ποικιλομορφία των μεγάλων δεδομένων, αναφέρεται στο είδος της δομής των δεδομένων. Το είδος της δομής περιλαμβάνει δεδομένα τα οποία είναι δομημένα, μη δομημένα και ήμι-δομημένα. Τα δεδομένα τα οποία είναι οργανωμένα και τα οποία επιπλέον είναι εύκολα προσβάσιμα και διαθέσιμα από έναν υπολογιστή, χαρακτηρίζονται ως δομημένα. Συνήθως αποθηκεύονται σε σχεσιακές βάσεις δεδομένων ή υπολογιστικά φύλλα (όπως αυτά του Excel), δηλαδή είναι περιορισμένα σε συγκεκριμένα πρότυπα, με τιμές σε συγκεκριμένα πεδία. Τα δεδομένα τα οποία δεν είναι οργανωμένα, απαιτούν δηλαδή επιπλέον ενέργειες για την επεξεργασία τους και επιπρόσθετα δεν είναι διαθέσιμα με ευκολία, αποκαλούνται μη δομημένα, Δηλαδή, δεν είναι προκαθορισμένος ο τρόπος με τον οποίο αποθηκεύονται. Παραδείγματα αποτελούν οι εικόνες, τα βίντεο, οι ιστοσελίδες, τα δεδομένα από αισθητήρες, τα αρχεία PDF και οι παρουσιάσεις PowerPoint. Τα ήμι δομημένα έχουν στοιχεία από τα δομημένα και τα μη δομημένα δεδομένα. Δηλαδή, μπορεί να έχουν μία προκαθορισμένη δομή η οποία όμως δεν είναι περιορισμένη όπως στα δομημένα, δηλαδή δεν αρκεί ώστε να βγάλουμε ένα σημασιολογικό νόημα, και απαιτεί περαιτέρω επεξεργασία (Yang, 2019).

Με την πάροδο των χρόνων, έχουν υπάρξει περισσότερα «V» τα οποία συμπεριλαμβάνονται στα χαρακτηριστικά των μεγάλων δεδομένων (Taleb et al. 2018). Συγκεκριμένα, για τα «7 V» σύμφωνα με τα άρθρα (Vijendra, 2016), (Uddin et al. 2014) παραθέτουμε τα ακόλουθα:

- **Ακρίβεια (Veracity)**
  Η ακρίβεια ήταν το πρώτο χαρακτηριστικό που ξεχώρισε ως αυτό που χρειάστηκε να συμπεριληφθεί ώστε να πάμε από τα «3 V» στα «4 V» (Schroeck et al. 2012). Στα κλασικά δεδομένα, για την διεξαγωγή ελέγχων αξιοπιστίας συνήθως χρησιμοποιούνται τεχνικές κανονικοποίησης και ελέγχου για διπλότυπα δεδομένα (Uddin et al. 2014). Τα μεγάλα δεδομένα έχουν εκατομμύρια ή δισεκατομμύρια τιμές, συχνά συμπεριλαμβάνουν και δεδομένα όπως πολλές

λανθασμένες ακραίες τιμές και δεν είναι αξιόπιστα για την αξιοποίησή τους χωρίς την σωστή προεργασία.

- **Αξία (Value)**
  Τα μεγάλα δεδομένα έχουν αξία, εάν είναι σε θέση να απαντήσουν ένα επιχειρηματικό ερώτημα. Δηλαδή, αφορά την τελική (πρακτική) ικανότητα των δεδομένων να απαντήσουν στο επιχειρηματικό ερώτημα, συνυπολογίζοντας το κόστος για την ενσωμάτωση, την διαχείριση και την εξόρυξη της γνώσης, με το κέρδος που θα αποφέρει η νέα γνώση (Uddin et al. 2014). Απαιτεί την συνεργασία της επιχείρησης με τον επιστήμονα των δεδομένων καθώς και όλες τις πτυχές και τις συνδέσεις γύρω από το επιχειρηματικό ερώτημα (Vijendra, 2016).

Εδώ, στο άρθρο (Uddin et al. 2014) οι ερευνητές ξεχωρίζει την αξιοπιστία και την αστάθεια ως τα υπόλοιπα 2 χαρακτηριστικά για τα «7 V», ενώ ο ερευνητής στο (Vijendra, 2016) ξεχωρίζει την οπτικοποίηση και την μεταβλητότητα.

- **Αξιοπιστία (Validity)**
  Η αξιοπιστία των μεγάλων δεδομένων αφορά «την ορθότητα και την ακρίβεια των δεδομένων ως προς την προοριζόμενη χρήση τους» (Uddin et al. 2014). Εξαρτάται, δηλαδή, από την προοριζόμενη χρήση, με αποτέλεσμα να είναι αξιόπιστο για κάποια χρήση αλλά πιθανώς να μην είναι αξιόπιστο για μία άλλη. Για παράδειγμα, τα δεδομένα που θα συλλεχθούν για την αξιοπιστία του εμβολίου έναντι της νόσου COVID – 19 από αναρτήσεις χρηστών στο Facebook, δεν αποτελούν αξιόπιστα δεδομένα. Αλλά αυτά θα ήταν αξιόπιστα δεδομένα εάν θέλαμε να κάνουμε μία ανάλυση συναισθήματος (sentiment analysis) για τους χρήστες που αναρτούν στο Facebook σχετικά με τα εμβόλια έναντι της νόσου COVID – 19.
- **Αστάθεια (Volatility)**
  Η αστάθεια των δεδομένων υπεισέρχεται όταν τα δεδομένα ενδέχεται να αλλάξουν μελλοντικά, ή ακόμη και να καταστραφούν. Συγκεκριμένα, σύμφωνα με τους Uddin et al., η αστάθεια αυξάνεται μαζί με τον όγκο, την ποικιλομορφία και την ταχύτητα των δεδομένων (Uddin et al. 2014). Δεδομένα με αστάθεια μπορεί να είναι για παράδειγμα οι συναλλαγές 10 ετών ενός πελάτη, καθώς αυξάνεται το κόστος αποθήκευσης, ασφάλειας και ανάκτησης, το οποίο οδηγεί στην ανάγκη αντιμετώπισης ή καταστροφής τους μετά από ένα συγκεκριμένο χρονικό περιθώριο.
- **Μεταβλητότητα (Variability)**
  Η μεταβλητότητα ως χαρακτηριστικό των μεγάλων δεδομένων αναφέρεται στην απροσδόκητη αλλαγή που μπορεί να υποστούν τα χαρακτηριστικά των μεγάλων δεδομένων, όπως για παράδειγμα η δομή ή η ποιότητα. Αυτό συνεπάγεται την ανάγκη για την ευελιξία των επιχειρήσεων σε πιθανές αλλαγές στην επεξεργασία και την αξιοποίηση των μεγάλων δεδομένων λόγω της μεταβλητότητας (Vijendra, 2016).
- **Οπτικοποίηση (Visualization)**
  Η οπτικοποίηση των δεδομένων αφορά την ικανότητα της αφηρημένης απεικόνισης δεδομένων για την απόφαση ορθών τεχνικών αποθήκευσης, εξαγωγής και μετάδοσης. Αυτά τα δεδομένα μπορεί να προέρχονται από ετερογενείς πηγές και τύπους δεδομένων (Vijendra, 2016).

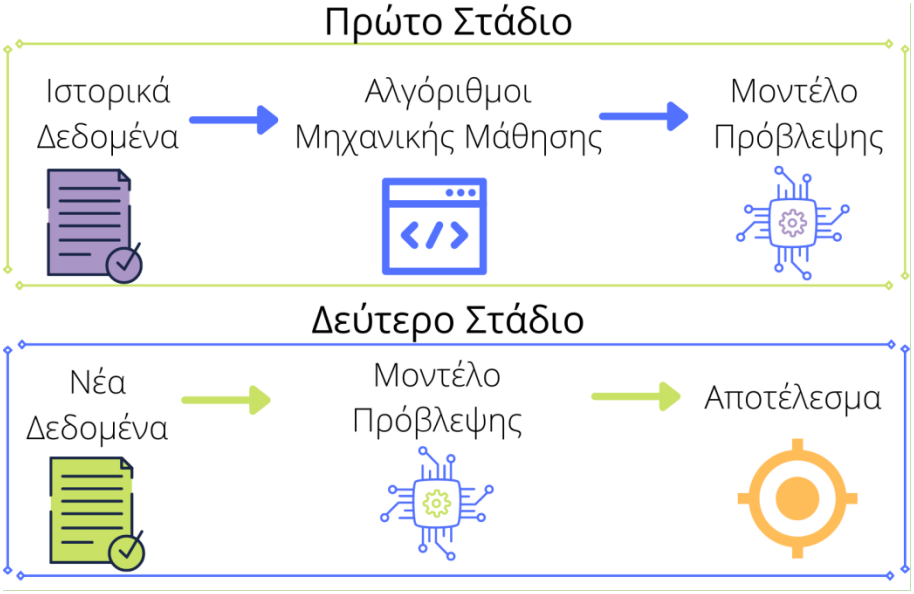
Ενότητα 2
Μηχανική Μάθηση

1.2.1 Εισαγωγή

Ο στόχος της μηχανικής μάθησης είναι η εύρεση χρήσιμων αναπαραστάσεων των δεδομένων και η αξιοποίηση αυτών των αναπαραστάσεων για την εξαγωγή συμπερασμάτων σε μελλοντικά δεδομένα, με παρόμοια χαρακτηριστικά (Najafabadi et al. 2015). Χρησιμοποιεί στοιχεία από τομείς της στατιστικής, της γραμμικής άλγεβρας, της τεχνητής νοημοσύνης και της επιστήμης των υπολογιστών. Η διαφορά, με τους πρώιμους αλγόριθμους της τεχνητής νοημοσύνης, είναι πως οι αλγόριθμοι της μηχανικής μάθησης δεν είναι ρητά προγραμματισμένοι από τον ερευνητή (Samuel, 1959). Αυτό απεικονίζεται στο Σχήμα 2.

Σχήμα 2. Περιγραφή λειτουργίας παραδοσιακών αλγόριθμων (επάνω) έναντι των αλγόριθμων της μηχανικής μάθησης (κάτω). Οι παραδοσιακοί αλγόριθμοι της τεχνητής νοημοσύνης είναι ρητά προγραμματισμένοι. Χρησιμοποιούν *κανόνες* όπως το εάν και το ίσως (if – else) για να βγάλουν συμπεράσματα. Οι αλγόριθμοι της μηχανικής μάθησης χρησιμοποιούν πολλά «λυμένα παραδείγματα», ώστε να «μάθουν» τους υποκείμενους κανόνες και σχέσεις που προϋπάρχουν στα δεδομένα, δίχως να τους έχουν προμηθευτεί (Muller & Guido, 2016). *Για την παραγωγή του σχήματος αξιοποιήθηκε η ιστοσελίδα canva.com.*

Αυτό, έχει ως αποτέλεσμα να μπορούν να αξιοποιηθούν σε προβλήματα για τα οποία *δεν υπάρχει γνωστός τρόπος προσέγγισης της πραγματικής λύσης τους*. Για παράδειγμα, δεν γνωρίζουμε πώς μπορούμε να προγραμματίσουμε ρητά προβλήματα αναγνώρισης ανθρώπινης ομιλίας (Geron, 2019). Συνολικά, η διαδικασία που εφαρμόζεται κατά την διεξαγωγή ενός αλγόριθμου της μηχανικής μάθησης συνοψίζεται στο *Σχήμα 3*.



Σχήμα 3. Στάδια διεξαγωγής αλγορίθμων μηχανικής μάθησης. Στο πρώτο στάδιο συλλέγονται τα ιστορικά δεδομένα, τα οποία αξιοποιούνται στον αλγόριθμο της μηχανικής μάθησης, ο οποίος στο πέρας της μάθησης, μας επιστρέφει ένα μοντέλο πρόβλεψης. Έπειτα, στο δεύτερο στάδιο μπορούμε να πραγματοποιήσουμε προβλέψεις χρησιμοποιώντας το μοντέλο πρόβλεψης, απλά με την χρήση δεδομένων τα οποία δεν έχει γνωρίσει ποτέ ο αλγόριθμος. *Για την παραγωγή του σχήματος αξιοποιήθηκε η ιστοσελίδα canva.com.*

Ο Tom Mitchell (1997), ανέφερε πως ένα πρόγραμμα υπολογιστή πραγματοποιεί μηχανική μάθηση εάν «μαθαίνει με βάση την εμπειρία του *E*, σχετικά με μία διεργασία Δ και έναν μετρητή απόδοσης *M*, εάν η απόδοσή του στην διεργασία *Δ*, με βάση την μέτρηση απόδοσης *M*, αυξάνει μαζί με την εμπειρία του *E*». Σε όσα ακολουθούν, αναφέρουμε κάποιες περιπτώσεις των διεργασιών, των μετρητών απόδοσης και της εμπειρίας που μπορεί να έχει ένας αλγόριθμος μηχανικής μάθησης.

- **Διεργασία (Δ)**
  Κλασικές διεργασίες για έναν αλγόριθμο μηχανικής μάθησης αποτελούν η κατηγοριοποίηση και η παλινδρόμηση (Goodfellow, 2016). Συνήθως αφορούν στοιχεία $x \in \mathbb{R}^n$, όπου $x = (x_1, x_2, \dots, x_n)$ και x_i είναι ένα χαρακτηριστικό που επιθυμούμε να μελετήσουμε.
  Η διεργασία της *κατηγοριοποίησης*, αφορά την μάθηση μίας συνάρτησης $f\colon \mathbb{R}^n \to \{1,2,\dots,k\}$, η οποία απεικονίζει τα $x \in \mathbb{R}^n$ σε έναν φυσικό αριθμό $y = f(x) \in \{1,2,\dots,k\}$, ο οποίος αντιστοιχεί σε μία από τις k υπό μελέτη κατηγορίες των χαρακτηριστικών που μας ενδιαφέρουν. Η διεργασία της *παλινδρόμησης* αντιστοιχεί στην μάθηση μίας συνάρτησης $f\colon \mathbb{R}^n \to \mathbb{R}$, δηλαδή σε κάθε $x \in \mathbb{R}^n$ αντιστοιχούμε έναν πραγματικό αριθμό $y = f(x)$. (Zhang, 2020).
  Άλλες διεργασίες αποτελούν, σύμφωνα με τον Goodfellow (2016),
  i. η *μηχανική μετάφραση*, όπου δεδομένα αλληλουχίας μετατρέπονται σε διαφορετικά δεδομένα αλληλουχίας, για παράδειγμα μία πρόταση στα γαλλικά μπορεί να μετατραπεί σε μία πρόταση στα ελληνικά,
  ii. η *μεταγραφή* όπου γίνεται αντιστοίχιση μη–δομημένων δεδομένων (πχ. ήχος) σε δεδομένα διακριτής μορφής κειμένου (πχ. περιγραφή με λέξεις),
  iii. η *ανίχνευση ανωμαλιών* (πχ. δόλια χρήση πιστωτικής κάρτας),
  iv. η *σύνθεση και δειγματοληψία,* κατά την οποία δημιουργούνται δεδομένα για αξιοποίηση στην μάθηση, και
  v. η *απομάκρυνση του θορύβου,* όπου ο αλγόριθμος μαθαίνεται μία συνάρτηση $f\colon \tilde{x} \mapsto x$, όπου $\tilde{x}, x \in \mathbb{R}^n$, δηλαδή αντιστοιχεί τις τιμές με «θόρυβο», οι οποίες απέχουν από την

πραγματική συνάρτηση κατανομής p, με τιμές που είναι πιο «κοντά» στην πραγματική συνάρτηση κατανομής τους.

- **Μετρητής Απόδοσης (M)**

Σύμφωνα με τους Zaki & Meira (2016), κάποιοι μετρητές απόδοσης για την ποιότητα του αλγόριθμου της μηχανικής μάθησης, όσο αφορά την διεργασία της *κατηγοριοποίησης,* αποτελούν

i. Ο *ρυθμός επιτυχιών* ή *ακρίβεια* (accuracy), η οποία αναφέρεται στο ποσοστό των προβλέψεων που αντιστοίχισε σωστά το μοντέλο πρόβλεψης του αλγόριθμου, αποτελεί δηλαδή μία *πιθανότητα*. Είναι εφικτό να «εκδηλώσουμε προτίμηση» για μία κλάση περισσότερο από μία άλλη, μέσω της χρήσης συντελεστών με βάρη για τον υπολογισμό της ακρίβειας. Για παράδειγμα, η ακρίβεια (*accuracy* ή *precision)* μπορεί να υπολογιστεί ως ο λόγος των επιτυχιών n_{ii} που πρόβλεψε ορθά ο αλγόριθμος να ανήκουν στην κλάση c_i, διά του συνόλου των τιμών που πρόβλεψε ο αλγόριθμος να ανήκουν στην κλάση c_i. Δηλαδή,

$$\text{Ακρίβεια ως προς μία κλάση = accuracy = precision} = \frac{n_{ii}}{m_i}$$

ii. Όσο αφορά την ακρίβεια, μπορούμε να υπολογίσουμε μία αντίστοιχη ποσότητα που αποκαλείται *κάλυψη* ή *ανάκληση* (Coverage/Recall), η οποία αφορά τον λόγο των επιτυχιών n_{ii} που πρόβλεψε σωστά ο αλγόριθμος να ανήκουν στην κλάση c_i, με τον συνολικό αριθμό των δεδομένων που γνωρίζουμε εξ' αρχής ότι ανήκουν στην κλάση c_i, τον οποίο συμβολίζουμε με n_i. Δηλαδή,

$$\text{Κάλυψη ως προς μία κλάση = coverage = recall} = \frac{n_{ii}}{n_i}$$

iii. Ένας ακόμη μετρητής απόδοσης είναι ο *ρυθμός σφάλματος* (error rate) που ορίζεται ως *1 – Ακρίβεια.*

iv. Τελικά, μέσω της κάλυψης και της ακρίβειας ορίζεται η F – μετρική (F – measure), η οποία έχει τον τύπο

$$F_i = \frac{2}{\frac{1}{prec_i} + \frac{1}{recall_i}} = \frac{2n_{ii}}{n_i + m_i},$$

Όπου με $prec_i$ και $recall_i$ συμβολίσαμε την ακρίβεια και την αντίστοιχη ανάκληση για τα $i-$ οστά δεδομένα που μας δίνονται. Ο σκοπός της F – μετρικής είναι η εύρεση μίας μετρικής που εξισορροπεί τις τιμές της ακρίβειας και της ανάκλησης για μία συγκεκριμένη κλάση των δεδομένων. Εάν προσθέσουμε όλες τις F_i που αντιστοιχούν σε κάθε μία κλάση c_i, και ο μέσος όρος τους είναι 1, τότε θα έχουμε τον *ιδανικό αλγόριθμο κατηγοριοποίησης*.

Όσο αφορά τις διεργασίες της *παλινδρόμησης*, οι Zaki & Meira (2020) αναφέρουν το R^2 στατιστικό, με τύπο

$$R^2 = \frac{RSS}{TSS},$$

όπου RSS είναι το άθροισμα των τετραγώνων της παλινδρόμησης (regression sum of squares) και το TSS είναι το ολικό άθροισμα των τετραγώνων (total sum of squares) οι οποίες αφορούν την απόκλιση των προβλεπόμενων τιμών από τις πραγματικές.

Επιπλέον, οι μετρητές απόδοσης για την διεργασία της *συσταδοποίησης* ποικίλουν αλλά οι Zaki & Meira (2020) ξεχωρίζουν τις ακόλουθες κατηγορίες τους:

i. *Εξωτερικοί.* Χρησιμοποιούν μετρήσεις οι οποίες δεν προκύπτουν από κάποιο συνδυασμό τιμών των δεδομένων, όπως για παράδειγμα η επισήμανση.

ii. *Εσωτερικοί.* Αξιοποιούν μετρήσεις οι οποίες προκύπτουν από κάποιο συνδυασμό των δεδομένων, όπως για παράδειγμα η απόσταση των συστάδων μεταξύ τους.

iii. *Σχετικοί.* Συγκρίνει διαφορετικές μετρήσεις με αλλαγή των παραμέτρων τους.

Επιπλέον, σύμφωνα με τον Goodfellow (2016), για την αξιολόγηση αλγόριθμων που υπολογίζουν πυκνότητες πιθανότητας, όπως για παράδειγμα η κατανομή των δεδομένων που δίνονται στον ερευνητή, μπορεί να αξιοποιηθεί η *μέση λογαριθμο – πιθανότητα* (log – probability).

- **Εμπειρία (Ε)**
  Σύμφωνα με τον Goodfellow (2016), η εμπειρία ενός αλγόριθμου της μηχανικής μάθησης είναι το σύνολο από δεδομένα το οποίο θα προμηθευτεί ο αλγόριθμος για να δημιουργήσει το μοντέλο πρόβλεψής του.

Για παράδειγμα, θα μπορούσαμε να θεωρήσουμε μία διεργασία (Δ) κατηγοριοποίησης, την αναγνώριση αντικειμένων σε εικόνες. Τότε, ο μετρητής απόδοσης (Μ) μπορεί να είναι η ακρίβεια, δηλαδή το πλήθος των σωστών αναγνωρίσεων διά το πλήθος των δεδομένων, ενώ η εμπειρία (Ε) θα μπορούσε να είναι ένα σύνολο δεδομένων με εικόνες που έχουν εμπρόσθιες όψεις από κατοικίδια, όπως γάτες και σκύλους.
Ενώ, μία διεργασία (Δ) παλινδρόμησης μπορεί να εφαρμοστεί στην τιμή πώλησης ενός σπιτιού, με βάση παράγοντες όπως τα τετραγωνικά μέτρα που καλύπτει το δάπεδο, η ημερομηνία κατασκευής του, το πλήθος των ορόφων, η απόσταση από το κέντρο της πόλης, η χρονική απόσταση της τελευταίας ανακαίνισης. Για μετρητή απόδοσης (Μ) θα μπορούσαμε να αξιοποιήσουμε το μέσο τετραγωνικό σφάλμα, ενώ για εμπειρία (Ε) θα μπορούσαμε να χρησιμοποιήσουμε ένα σύνολο δεδομένων με τιμές για σπίτια που περιλαμβάνουν τους παράγοντες που αναφέρθηκαν. Η εμπειρία (Ε) μπορεί να διαχωριστεί σε κατηγορίες ανάλογα την ανθρώπινη επίδραση στα δεδομένα, αναλόγως εάν έχουμε επισημασμένα δεδομένα ή μη – επισημασμένα δεδομένα, όπως θα δούμε στην επόμενη ενότητα.

1.2.2 Κατηγορίες Μηχανικής Μάθησης

Συνολικά, στην βιβλιογραφία προτείνονται αρκετά χαρακτηριστικά για την κατηγοριοποίηση των αλγόριθμων της μηχανικής μάθησης. Σύμφωνα με τον Geron, (2019), οι αλγόριθμοι της μηχανικής μάθησης είναι εύκολο να κατηγοριοποιηθούν με βάση τα παρακάτω γενικά χαρακτηριστικά:

- Πρώτον, εάν
  i. υπάρχει ανθρώπινη επίβλεψη στις ετικέτες των δεδομένων εκπαίδευσης (μάθηση με επίβλεψη),
  ii. ή εάν δεν υπάρχει, (μάθηση δίχως επίβλεψη).
  iii. ή εάν υπάρχει ενισχυτική μάθηση.
- Δεύτερον, εάν η μάθηση είναι:
  i. ταυτόχρονη (στιγμιαία), όπως η μάθηση σε σύνδεση (online learning), ή
  ii. σταδιακή, όπου αναφέρεται η μάθηση με τεμάχια (batch learning).
- Τρίτον, εάν για την διαδικασία της μάθησης
  i. συγκρίνονται τα γνωστά σημεία με τα καινούργια, όπως στην σημειακή μάθηση (instance – based learning),
  ii. είτε εάν χρησιμοποιούνται τα μοτίβα από τα γνωστά σημεία την δημιουργία ενός μοντέλου πρόβλεψης, με βάση το οποίο θα πραγματοποιηθεί η πρόβλεψη, την μάθηση μοντέλου (model – based learning).

Όταν έχουμε ανθρώπινη επίβλεψη στις ετικέτες των δεδομένων, αυτό πρακτικά σημαίνει ότι δημιουργείται ένα νέο χαρακτηριστικό στα δεδομένα από τον άνθρωπο (κατά προτίμηση έναν ειδικό ή επιστάτη), το οποίο δεν προϋπήρχε. Αυτό, σε μία διεργασία (που θα το αξιοποιήσει), αποκαλείται ως ο «στόχος» (target) τον οποίο προσπαθεί να προβλέψει ο αλγόριθμος της μηχανικής μάθησης. Για παράδειγμα, στην κατηγοριοποίηση εικόνας, επισημασμένες εικόνες είναι αυτές όπου ένας άνθρωπος έχει προσθέσει σε κάθε εικόνα τι κατοικίδια περιέχει. Οι *ετικέτες* των δεδομένων αντιστοιχούν σε μία ευρύτερη *κλάση* δεδομένων.

Στα προβλήματα κατηγοριοποίησης, η ετικέτα y_i αφορά την κλάση του αντικειμένου x_i, ενώ στα προβλήματα παλινδρόμησης αυτή αφορά την τιμή του αντικειμένου x_i. Αναφέρεται επίσης πως είναι δύσκολο να βρεθούν επισημασμένα δεδομένα, σε αντίθεση με τα μη – επισημασμένα δεδομένα (Zhang, 2020). Για παράδειγμα, δύο φωτογραφίες όπου έχουμε σκύλους μπορούμε να τις συμπεριλάβουμε στην κλάση «φωτογραφία με σκύλο», η οποία περιέχει όλες τις θεωρητικά δυνατές φωτογραφίες με σκύλους. Ιδανικά, ο αλγόριθμος θα χρησιμοποιήσει την πληροφορία πως μία εικόνα έχει ετικέτα με σκύλους, ώστε να βρει με μεγάλη βεβαιότητα πως μία άλλη εικόνα περιέχει και αυτή έναν σκύλο.
Όσο αφορά την επισήμανση των δεδομένων έχουμε δύο κατηγορίες, σύμφωνα με τον Zhang (2020), την *Ευκλείδεια Μάθηση Δεδομένων* και την *Μη – Ευκλείδεια Μάθηση Δεδομένων.* Η Μη – Ευκλείδεια μάθηση αφορά την μηχανική μάθηση με γραφήματα (graph machine learning), όπου ο αλγόριθμος μαθαίνει από την δομή ενός γραφήματος. Η Ευκλείδεια μάθηση περιλαμβάνει την Μάθηση υπό Επίβλεψη, την Μάθηση δίχως Επίβλεψη, την Μάθηση με ήμι – Επίβλεψη και την Ενισχυτική Μάθηση.

Ευκλείδεια μάθηση

- **Μάθηση υπό Επίβλεψη (Supervised Learning)**
  Ο σκοπός της μάθησης υπό επίβλεψη είναι η μάθηση μίας συνάρτησης $f: I \to O$, όπου $I = (X, Y) = (\boldsymbol{x}_i, \boldsymbol{y}_i)$, όπου $\boldsymbol{x}_i \in \mathbb{R}^n$ και $\boldsymbol{y}_i \in \mathbb{R}^k$ και $1 \leq i \leq m$, είναι τα ζευγάρια διανυσμάτων των δεδομένων εισόδου και των αντιστοίχων δεδομένων εξόδου. Δηλαδή, η μάθηση γίνεται μέσω *δειγμάτων από ζευγάρια δεδομένων εισόδου – εξόδου* (σύνολο I). Αναφέρεται πως είναι *μία προσέγγιση που βασίζεται στα έργα* (task – driven approach) (Sarker, 2021). Συνεπώς, έχουμε μάθηση υπό επίβλεψη όταν ο ερευνητής διαθέτει, *επιπλέον*, δεδομένα y_i επισημασμένα από ανθρώπους (κατά προτίμηση ειδικούς), με αποτέλεσμα να βοηθήσουν τον αλγόριθμο της μηχανικής μάθησης να γνωρίζει πότε κάνει λάθος πρόβλεψη και πότε όχι, το οποίο μέσω του μετρητή απόδοσης (M) οδηγεί στην «μηχανική μάθηση» του αλγόριθμου. (Zhang, 2020).
  Σύμφωνα με τον Geron, (2019) ορισμένοι αλγόριθμοι της μηχανικής μάθησης που μπορούν να αξιοποιήσουν τα επισημασμένα δεδομένα, αποτελούν:
  - Για την *κατηγοριοποίηση*, τα τυχαία δάση (Random Forests) και τα δέντρα αποφάσεων (Decision Trees), οι μηχανές διανυσμάτων υποστήριξης (Support Vector Machines), η λογιστική παλινδρόμηση.
  - Για την *παλινδρόμηση,* η γραμμική παλινδρόμηση.
  - Για την *συσταδοποίηση*, η μέθοδος k – κοντινότερων γειτόνων (k-Nearest Neighbors) (προβλήματα συσταδοποίησης) (Geron, 2019).

- **Μάθηση δίχως Επίβλεψη (Unsupervised Learning)**
  Συμβαίνει όταν ο ερευνητής δεν έχει επισημασμένα δεδομένα, δηλαδή οι αλγόριθμοι πρέπει να ξεχωρίσουν από μόνοι τους εάν και ποιες σχέσεις υπάρχουν στα δεδομένα. Συνεπώς, αποτελεί *μία προσέγγιση που βασίζεται στα δεδομένα* (data – driven approach) (Sarker, 2021). Συνολικά, δουλεύουν με ένα σύνολο δεδομένων $X = x_i$, όπου $x_i \in \mathbb{R}^n$ (Zhang, 2020).
  Σύμφωνα με τον Geron, (2019) παραδείγματα αλγορίθμων που μπορούν να αξιοποιήσουν μη – επισημασμένα δεδομένα, αποτελούν:
  - Για την *συσταδοποίηση*, η μέθοδος k – κοντινότερων γειτόνων (k-Nearest Neighbors) και η ιεραρχική ανάλυση συστάδων (Hierarchical Cluster Analysis).
  - Για την *ανίχνευση ανωμαλιών,* η μέθοδος των μηχανών διανυσμάτων υποστήριξης μίας κλάσης (One – class SVM) και το δάσος απομόνωσης (Isolation Forest).
  - Για την *οπτικοποίηση και μείωση διάστασης*, η ανάλυση κυρίων συνιστωσών (Principal Component Analysis), ενώ συμπεριλαμβάνεται και η παραγοντοποίηση ιδιαζουσών τιμών (Singular Value Decomposition) .
  - Για την *σχεσιακή μάθηση κανόνων,* οι αλγόριθμοι "*Apriori*" και "*Eclat*" (Geron, 2019).
  - Άλλες κοινές διεργασίες συμπεριλαμβάνουν τον *υπολογισμό πυκνοτήτων* και την *μάθηση χαρακτηριστικών* (Sarker, 2021).

- **Μάθηση με Ήμι-επίβλεψη (Semi-Supervised Learning)**
  Θεωρείται ως ένα υβρίδιο της μάθησης με επίβλεψη και χωρίς επίβλεψη. Ο στόχος της είναι η παροχή ενός μοντέλου καλύτερο από το αντίστοιχο που χρησιμοποιεί μόνο τα μη – επισημασμένα δεδομένα ή μόνο τα επισημασμένα δεδομένα που παρέχονται (Sarker, 2021). Σε αυτή την περίπτωση, τα δεδομένα διαχωρίζονται σε υποσύνολα, ανάλογα εάν είναι επισημασμένα (*labeled*) ή εάν δεν είναι (*unlabeled*). Συνολικά, έχουμε το σύνολο δεδομένων:
  $D_{unlabeled} \cup D_{labeled} = X_{unlabeled} \cup (X, Y)_{labeled}$
  $= (x_1, x_2, \dots, x_u) \cup ((x_{u+1}, y_1), (x_{u+2}, y_2), \dots, (x_{u+l}, y_l)),$
  όπου $x_i \in D_{unlabeled} \subseteq \mathbb{R}^n, i = 1,2, \dots, u$ και $x_j \in D_{labeled} \subseteq \mathbb{R}^n, y_i \in \mathbb{R}^k, j = 1,2, \dots, l.$
  Επιπλέον, συνηθίζεται $l \ll u$. Δηλαδή, τα δεδομένα με επισήμανση (πλήθος l) θα είναι κατά πολύ λιγότερα από αυτά δίχως επισήμανση (πλήθος u). Αυτό συμβαίνει διότι η επισήμανση έχει μεγάλο κόστος, το οποίο είναι σαφώς παρόν και στα μεγάλα δεδομένα. Επιπλέον, ανάλογα την μορφή που γίνεται η επισήμανση των δεδομένων, έχουμε ακόμη τις ακόλουθες περιπτώσεις (Zhang, 2020) :
  - Αυτό–εκπαίδευση (Self – Training)
    Χρησιμοποιούνται οι προβλέψεις από τον αλγόριθμο για την δημιουργία επισημασμένων δεδομένων εκπαίδευσης.
  - Συνεκπαίδευση (Co – Training)
    Αφορά δεδομένα πολλαπλής προβολής (Multi – View), ενώ χρησιμοποιούνται οι προβλέψεις από τον αλγόριθμο για την δημιουργία επισημασμένων δεδομένων εκπαίδευσης.
  - Ενεργή Μάθηση (Active Learning)
    Ο αλγόριθμος επιλέγει δεδομένα, τα οποία θα ζητήσει να επισημανθούν από έναν ειδικό ή δάσκαλο.

  Οι αλγόριθμοι της μηχανικής μάθησης που μπορούν να αξιοποιήσουν δεδομένα με ήμι – επίβλεψη, σύμφωνα με τον Geron (2019), συνήθως έχουν μέρη που αντιμετωπίζουν δεδομένα με επισήμανση και μέρη που αντιμετωπίζουν δεδομένα δίχως επισήμανση. Αναφέρει το παράδειγμα των Νευρωνικών Δικτύων Βαθιάς Πεποίθησης (Deep Belief Network), στο οποίο χρησιμοποιούνται αλγόριθμοι δίχως επίβλεψη (οι οποίοι αποκαλούνται *Περιορισμένες Μηχανές Boltzmann (restricted Boltzmann machines, RBM)*) ενώ τελικά γίνεται βελτιστοποίηση μέσω αλγόριθμων με επισήμανση. Εφαρμογές της ήμι-επίβλεψης υπάρχουν στην μηχανική μετάφραση, στην ανίχνευση απάτης, στην επισήμανση δεδομένων και στην κατηγοριοποίηση κειμένου (Sarker, 2021).
- **Ενισχυτική Μάθηση (Reinforcement Learning)**
  Είναι ικανή στην αντιμετώπιση των δεδομένων με χαρακτηριστικά μεγάλων διαστάσεων. Ο σκοπός της ενισχυτικής μάθησης είναι να δημιουργήσει μία αλληλεπίδραση μεταξύ των δεδομένων και της διαδικασίας της μάθησης του αλγόριθμου, συγκεκριμένα ένα δυναμικό περιβάλλον μάθησης (Zhang, 2020), δηλαδή έχουμε *μία προσέγγιση με βάση το περιβάλλον.* (Sarker, 2021). Σύμφωνα με τον Geron, 2019, το μοντέλο πρόβλεψης ονομάζεται *πράκτορας* (agent) και μπορεί να επιλέξει δεδομένα από το σύνολο δεδομένων και να πραγματοποιήσει πράξεις. Θα λάβει *επιβραβεύσεις* (rewards) εάν πραγματοποιήσει «θετικές» πράξεις, ενώ *ποινές* (penalties) εάν πραγματοποιήσει «αρνητικές» πράξεις. Ο πράκτορας μαθαίνει την βέλτιστη στρατηγική όπου λαμβάνει περισσότερες επιβραβεύσεις απ' ότι ποινές, η οποία καλείται *πολιτική* (policy), από την οποία θα πραγματοποιήσει και τις μελλοντικές του επιλογές για ένα νέο σύνολο δεδομένων.

Οι ανωτέρω κατηγορίες μάθησης είναι ευρέως διαδεδομένες στις εφαρμογές της μηχανικής μάθησης. Όμως, τα χαρακτηριστικά των μεγάλων δεδομένων όπως (i) ο όγκος (ii) η ταχύτητα (iii) η ποικιλομορφία (iv) η ακρίβεια και η (iv) αξία, βάζουν εμπόδια στην αξιοποίηση των ανωτέρω τεχνικών (Qiu et al. 2016). Συνεχίζοντας, θα αναφέρουμε κάποιες κατηγορίες της μηχανικής μάθησης οι οποίες καταφέρνουν να αντιμετωπίσουν κάποια από τα χαρακτηριστικά (i) έως (v), σύμφωνα με τους Qiu et al (2016).

- **Ενεργή Μάθηση (Active Learning)**
  Αφορά την μάθηση όπου ο αλγόριθμος έχει «ενεργό» ρόλο στην επιλογή των δεδομένων στα οποία θα πραγματοποιηθεί η μάθηση (Zhang, 2020). Συγκεκριμένα, μπορεί να επιλέξει δεδομένα για τα οποία θα ζητήσει την επισήμανση από έναν ειδικό ή δάσκαλο, με σκοπό να βελτιώσει την ακρίβεια του αλγόριθμου μειώνοντας το κόστος της επισήμανσης δεδομένων (Qiu et al. 2016).
- **Μάθηση Αναπαραστάσεων (Representation Learning)**
  Έχει ως στόχο την μάθηση μίας μεγάλης αναπαράστασης των δεδομένων, στην οποία εμπεριέχεται η πληροφορία για πολλά πιθανά καινούργια δεδομένα, με αποτέλεσμα βελτίωση της αποτελεσματικότητας. Προτιμάται για δεδομένα τα οποία έχουν μεγάλη διάσταση, ενώ αναφέρονται ικανοποιητικά αποτελέσματα σε διεργασίες μείωσης διάστασης. Βρίσκει εφαρμογές σε αναγνώριση ομιλίας, επεξεργασία φυσικού λόγου, έξυπνα συστήματα αυτοκινήτων (Qiu et al. 2016).
- **Βαθιά Μάθηση (Deep Learning)**
  Έχει ως στόχο την μάθηση σύνθετων ιεραρχικών αναπαραστάσεων στα δεδομένα, σε αντίθεση με τους συνηθισμένους αλγόριθμους της μηχανικής μάθησης που αποκαλούνται «ρηχοί» (Zaki & Meira, 2020). Περισσότερα θα αναφερθούν στο σχετικό κεφάλαιο.
- **Κατανεμημένη και Παράλληλη Μάθηση (Distributed and Parallel Learning)**
  Ο όγκος των μεγάλων δεδομένων αποτρέπει τους αλγόριθμους της μηχανικής μάθησης να αξιοποιήσουν την πληροφορία στα μεγάλα δεδομένα με μεγάλη ταχύτητα. Αυτό οδήγησε στην *κατανεμημένη* μάθηση, όπου αξιοποιούνται πολλαπλοί σταθμοί (workstations) για την αύξηση της ταχύτητας των διεργασιών. Επιπλέον, η *παράλληλη* μάθηση αξιοποιεί πολυπύρηνους επεξεργαστές και εφαρμογές «σύννεφου» (cloud) για να πετύχει παρόμοια αποτελέσματα (Qiu et al. 2016).
- **Μάθηση Μεταβίβασης (Transfer Learning)**
  Οι κλασικοί αλγόριθμοι της μηχανικής μάθησης προϋποθέτουν πως τα δεδομένα παράγονται ομοιόμορφα και ανεξάρτητα από μία κοινή κατανομή. Στις περιπτώσεις που αυτό δεν είναι εφικτό, η μάθηση μεταβίβασης μπορεί να εφαρμοστεί. Αξιοποιείται σε περιπτώσεις όπου περισσότερα δεδομένα είναι δύσκολο να βρεθούν, είτε γιατί είναι σπάνια, ακριβά στην συλλογή και επισήμανση ή μη – διαθέσιμα. Βρίσκει εφαρμογή στα μεγάλα δεδομένα καθώς μπορεί κάποια σύνολα να σχετίζονται, αλλά να μην είναι ακριβώς τα ίδια. Αυτό έχει ως αποτέλεσμα εφαρμογές σε τομείς των μεγάλων δεδομένων όπως η κατηγοριοποίηση συναισθήματος από κείμενο, η κατηγοριοποίηση εικόνας, η κατηγοριοποίηση ανθρώπινης δραστηριότητας και η κατηγοριοποίηση κειμένου σε πολλές γλώσσες. Εφαρμόζεται κατά κύριο λόγο σε περιπτώσεις όπου δύο υπό-τομείς ανήκουν σε έναν κοινό τομέα (Weiss et al. 2016).
- **Μάθηση με βάση Πυρήνες (Kernel – based Learning)**
  Οι εφαρμογές με πυρήνες είναι αρκετά συχνές στην βιβλιογραφία. Χρησιμοποιούν το «κόλπο πυρήνα» (kernel trick) για την αναπαράσταση των δεδομένων στον χώρο των χαρακτηριστικών, ο οποίος μπορεί να φτάσει και άπειρη διάσταση. Αποτελούν μία μη – γραμμική μέθοδο εξόρυξης γνώσης από τα δεδομένα αλλά έχουν μεγάλο υπολογιστικό κόστος. (Qiu et al. 2016) Σύμφωνα με τους Ghorbani et al (2021), οι αλγόριθμοι των νευρωνικών δικτύων φαίνεται να έχουν συνολικά καλύτερη συμπεριφορά από τις μεθόδους με πυρήνα (kernel methods).

Είναι σημαντική η διαπίστωση πως τα νευρωνικά δίκτυα μπορούν να συνδυαστούν με όλες τις παραπάνω κατηγορίες μάθησης, με εφαρμογές σε πολλούς τομείς στην βιβλιογραφία.

1.2.3 Εκπαίδευση Αλγόριθμων Μηχανικής Μάθησης

Εκπαίδευση

Στην μηχανική μάθηση, για την διεξαγωγή της εκπαίδευσης των αλγορίθμων, αρχικά χωρίζονται τα δεδομένα που δίνονται στον ερευνητή σε *δεδομένα εκπαίδευσης* και *δεδομένα ελέγχου* (training & test data). Ο σκοπός είναι η χρήση των δεδομένων εκπαίδευσης για την «μάθηση» του μοντέλου πρόβλεψης από τον

αλγόριθμο, και την χρήση των δεδομένων ελέγχου για την αξιολόγηση του μοντέλου πρόβλεψης σε δεδομένα που δεν χρησιμοποιήθηκαν στην μάθηση (Geron, 2019). Για παράδειγμα, ο δάσκαλος (συνάρτηση ελέγχου) εξετάζει τον μαθητή (μοντέλο πρόβλεψης) σε ασκήσεις παρόμοιες με αυτές που έχει κάνει (δεδομένα ελέγχου), αλλά όχι ίδιες με αυτές που έχει κάνει (δεδομένα εκπαίδευσης). Οι αλγόριθμοι της μηχανικής μάθησης, δηλαδή, *προϋποθέτουν πως τα δεδομένα εκπαίδευσης και τα δεδομένα ελέγχου θα προέρχονται από μία κοινή κατανομή*, δηλαδή αυτά θα είναι ανεξάρτητα και ομοιόμορφα κατανεμημένα (Goodfellow, 2016).

Για την διαδικασία της εκπαίδευσης του αλγόριθμου, θα χρειαστούμε (Chollet & Allaire, 2018):

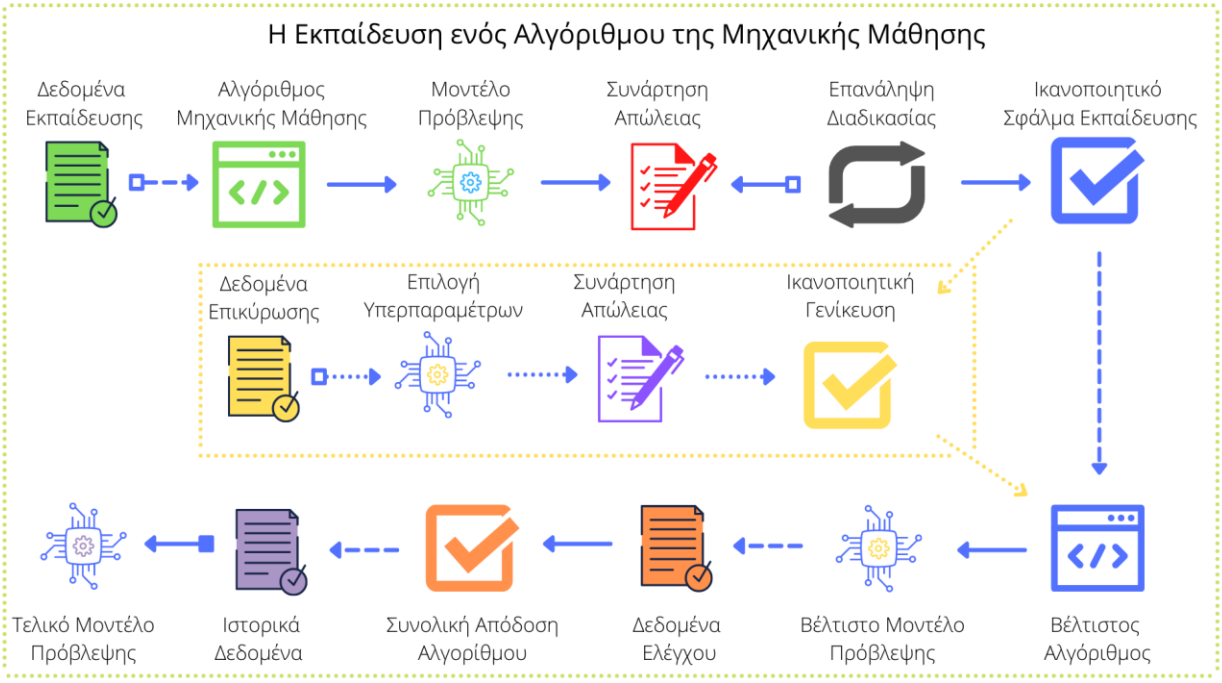
- **Μία *συνάρτηση απώλειας* (loss function)** η οποία θα μας επιστρέψει το *σφάλμα της εκπαίδευσης* (training error) που αποτελεί έναν τρόπο για την αξιολόγηση του μοντέλου πρόβλεψης του αλγόριθμου της μηχανικής μάθησης. Μεγάλη τιμή στο σφάλμα εκπαίδευσης υπονοεί ότι οι τιμές που προβλέπει ο αλγόριθμος έχουν μεγάλη απόκλιση από τις πραγματικές. Σκοπός είναι το σφάλμα της εκπαίδευσης να γίνει μικρό, δηλαδή οι τιμές πρόβλεψης να είναι «κοντά» με τις πραγματικές, δηλαδή «παρόμοιες».
- **Μία *συνάρτηση βελτιστοποίησης* (optimizer function)** με την οποία, ανάλογα την τιμή της συνάρτησης απώλειας, θα ανανεωθούν οι παράμετροι του μοντέλου πρόβλεψης, με σκοπό να μειωθεί το σφάλμα της εκπαίδευσης. Έπειτα θα επαναληφθεί η διαδικασία μέχρι να επιτευχθεί ένα σφάλμα εκπαίδευσης αρκετά «μικρό», καθώς εάν είναι σχετικά «μεγάλο», αυτό θα οδηγήσει σε προβλήματα *υποπροσαρμογής* (underfitting).
- **Μία *μετρική* (metric)** (είχαμε αναφερθεί σε αυτήν ως *μετρητή απόδοσης* (M) στην ενότητα 1.2.1 η οποία αξιολογεί τον αλγόριθμο στα δεδομένα. Αποτελεί μία ποσότητα που συνήθως θέλουμε να *μεγιστοποιήσουμε*, ενώ αυτό συμβαίνει ταυτόχρονα κατά την διάρκεια της ελαχιστοποίησης της συνάρτησης απώλειας. Το *σφάλμα εκπαίδευσης* αφορά την απόδοση του αλγόριθμου στα δεδομένα εκπαίδευσης, ενώ το *σφάλμα γενίκευσης* (generalization error) αναφέρεται στην απόδοση του αλγόριθμου στα δεδομένα ελέγχου, δηλαδή δεδομένα που δεν έχει συναντήσει κατά την εκπαίδευση. Γενικά, θέλουμε μικρό σφάλμα γενίκευσης και μικρό σφάλμα εκπαίδευσης (ή ταυτόχρονα μεγάλες τιμές στην μετρική των δεδομένων εκπαίδευσης και των δεδομένων ελέγχου) αλλά υπερτερεί να έχουμε μικρό σφάλμα γενίκευσης. Συνολικά, εάν έχουμε *μικρό* σφάλμα εκπαίδευσης και *μεγάλο* σφάλμα γενίκευσης, τότε ο αλγόριθμος είναι καλός στην αξιολόγηση του συνόλου των δεδομένων, αλλά μόνο σε αυτό το σύνολο των δεδομένων και όχι σε ένα καινούργιο, όπως στο σύνολο ελέγχου. Τότε λέμε πως έχουμε *υπερπροσαρμογή* του αλγόριθμου στα δεδομένα εκπαίδευσης.

Η μάθηση των παραμέτρων του μοντέλου πρόβλεψης αντιστοιχεί στην μάθηση μίας συνάρτησης που ορίζει το μοντέλο πρόβλεψης. Είναι αξιοσημείωτο πως εάν τα δεδομένα εκπαίδευσης περιλάμβαναν όλα τα πιθανά δεδομένα, τότε ο αλγόριθμος θα μπορούσε να μάθει το «πραγματικό» μοντέλο πρόβλεψης των δεδομένων, δίχως να κάνει ποτέ λάθος (Goodfellow, 2016). Όμως ο σκοπός της μηχανικής μάθησης δεν είναι η εύρεση της «πραγματικής συνάρτησης» αλλά μίας συνάρτησης η οποία είναι «αρκετά κοντά» στην πραγματική (Geron, 2018).

Σύμφωνα με τον Goodfellow (2016), η *χωρητικότητα* (capacity) ενός μοντέλου μηχανικής μάθησης, αφορά την ικανότητά του να προσεγγίζει πολλές συναρτήσεις μέσω της μάθησης. Μικρή χωρητικότητα συνεπάγεται πως το μοντέλο δεν θα είναι σε θέση να μάθει πολλές συναρτήσεις, με λογικό αποτέλεσμα να μην καταφέρει να βρει μία συνάρτηση «κοντά» στην πραγματική, το οποίο οδηγεί σε υποπροσαρμογή. Μεγάλη χωρητικότητα ενδέχεται να προκαλέσει υπερπροσαρμογή, καθώς η συνάρτηση μπορεί να μάθει μη χρήσιμες πληροφορίες, όπως τον θόρυβο που μπορεί να έχουν τα δεδομένα και την προκατάληψη των δεδομένων.

Μία μέθοδος για τον έλεγχο της χωρητικότητας του μοντέλου, είναι η επιλογή του *Χώρου των Υποθέσεων* (Hypothesis Space). Αυτός περιλαμβάνει τις πιθανές συναρτήσεις που μπορεί να μάθει ο αλγόριθμος της μηχανικής μάθησης (Goodfellow, 2016). Είναι λογικό πως εάν θεωρήσουμε έναν μικρό χώρο των

υποθέσεων, θα έχουμε μικρή χωρητικότητα, ενώ με έναν πολύ μεγάλο χώρο των υποθέσεων θα έχουμε μεγάλη χωρητικότητα.



Σχήμα 4. Παρουσιάζεται η εκπαίδευση ενός αλγόριθμου της μηχανικής μάθησης. Το πρώτο βήμα της εκπαίδευσης είναι πάνω αριστερά, στα *Δεδομένα Εκπαίδευσης,* ενώ το τελευταίο βήμα της εκπαίδευσης είναι κάτω αριστερά, στο *Τελικό Μοντέλο Πρόβλεψης,* ακολουθώντας τα βέλη. Συμπεριλαμβάνεται η περίπτωση όπου επιθυμούμε την χρήση δεδομένων επικύρωσης. Ως *Ιστορικά Δεδομένα,* αναφέρουμε ολόκληρο το σύνολο δεδομένων, δηλαδή αυτό που περιέχει τα *Δεδομένα Εκπαίδευσης,* τα *Δεδομένα Επικύρωσης* και τα *Δεδομένα Ελέγχου.* Περισσότερα για την διαδικασία της επιλογής των υπερπαραμέτρων αναφέρονται παρακάτω στην ενότητα. *Για την παραγωγή του σχήματος αξιοποιήθηκε η ιστοσελίδα canva.com.*

Επικύρωση και Επιλογή Μοντέλου

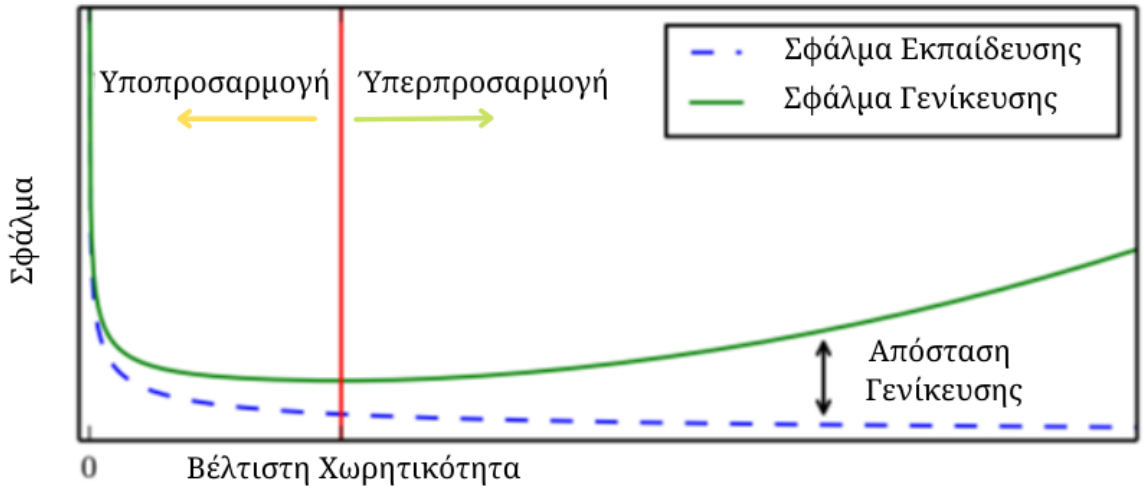
Κάθε φορά που ανανεώνουμε την επιλογή μας για τις υπερπαραμέτρους του μοντέλου (πχ. βαθμός πολυωνύμου, αριθμός στοιβάδων σε νευρωνικό δίκτυο, πλήθος συστάδων) της μηχανικής μάθησης, με σκοπό την ελαχιστοποίηση του σφάλματος γενίκευσης, χωρίς να το θέλουμε, *εισάγουμε στον αλγόριθμο προκατάληψη* (bias) *ως προς τα δεδομένα ελέγχου*, και ας μην έχει έρθει ποτέ άμεσα σε επαφή μαζί τους (Chollet & Allaire, 2018). Δηλαδή, μπορεί με τις επανειλημμένες προσπάθειές μας να προσαρμόσουμε το μοντέλο στα δεδομένα ελέγχου, δίχως καλά αποτελέσματα σε άλλα δεδομένα. Για να το αποφύγουμε αυτό, χρησιμοποιούμε ακόμα ένα σύνολο δεδομένων για την εκπαίδευση: τα *δεδομένα επικύρωσης* (Goodfellow, 2016). Χωρίζουμε το αρχικό σύνολο δεδομένων, σε τρία: (α) Τα *δεδομένα εκπαίδευσης*, (β) τα *δεδομένα επικύρωσης* και (γ) τα *δεδομένα ελέγχου*. Για παράδειγμα, από το 100% των δεδομένων που δίνονται στον ερευνητή, αυτός μπορεί να χρησιμοποιήσει το 70% για δεδομένα εκπαίδευσης, το 20% για δεδομένα επικύρωσης και το άλλο 10% για δεδομένα ελέγχου (Muller & Guido, 2016).

Ο σκοπός είναι να χρησιμοποιήσουμε τα δεδομένα εκπαίδευσης για την δημιουργία ενός μοντέλου πρόβλεψης, και έπειτα να χρησιμοποιήσουμε τα δεδομένα επικύρωσης για την αξιολόγηση αυτού του μοντέλου και την επιλογή των παραμέτρων που επιστρέφουν την μεγαλύτερη τιμή στην μετρική. Τελικά, αφού πραγματοποιηθεί αυτός ο κύκλος αρκετές φορές, θα ξεκινήσει να μεγαλώνει το σφάλμα των

δεδομένων επικύρωσης, συνεπώς ο αλγόριθμος θα έχει ξεκινήσει την υπερπροσαρμογή. Εάν σταματήσουμε την διαδικασία, και θεωρήσουμε τις υπερπαραμέτρους και τις παραμέτρους που έμαθε ο αλγόριθμος στην μέγιστη τιμή της μετρικής (ή στην ελάχιστη τιμή σφάλματος) των δεδομένων επικύρωσης, θα έχουμε πραγματοποιήσει την διαδικασία του *πρώιμου τερματισμού* (early stopping). (Chollet & Allaire, 2018).

Η προτεινόμενη μεθοδολογία σύμφωνα με τους Muller & Guido (2016) είναι να χρησιμοποιηθεί το σύνολο επικύρωσης για την επιλογή των υπερπαραμέτρων που οδηγούν στον καλύτερο μετρητή απόδοσης και τότε να γίνει *μία* φορά η αξιολόγηση του μοντέλου με το σύνολο ελέγχου, το οποίο θα αποτελεί και την τελική μέτρηση ικανότητας του μοντέλου. Έπειτα, χρησιμοποιείται το 100% των δεδομένων για την εύρεση της τελικής παραμέτρου του μοντέλου πρόβλεψης από τον αλγόριθμο της μηχανικής μάθησης, ο οποίος θα έχει την μεγαλύτερη ικανότητα να γενικεύσει τα αποτελέσματά του σε άγνωστα δεδομένα.

Εάν έχουμε μικρά σύνολα δεδομένων, συνιστάται η μέθοδος της *k - διασταυρωμένης επικύρωσης* (k – fold cross – validation) κατά την οποία θεωρούμε k επιλογές για τα δεδομένα επικύρωσης και ως σφάλμα ελέγχου θεωρούμε τον μέσο όρο από τα k σφάλματα ελέγχου που υπολογίζουμε για την κάθε επιλογή από τα δεδομένα επικύρωσης μέσα στο σύνολο των δεδομένων (Goodfellow, 2016).



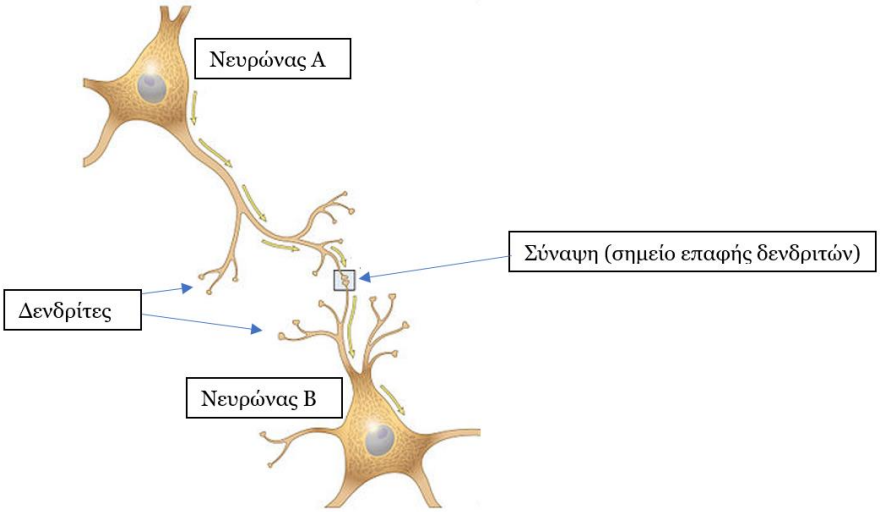
Σχήμα 5. Απεικόνιση καμπυλών από το σφάλμα εκπαίδευσης και το σφάλμα γενίκευσης (ή το σφάλμα από την επικύρωση). Στην κόκκινη γραμμή έχουμε το ελάχιστο σφάλμα γενίκευσης, οπότε αυτό θα ήταν το σημείο που μπορούσαμε να πραγματοποιήσουμε την μέθοδο του πρώιμου σταματήματος και να επιλέξουμε το καλύτερο μοντέλο για τα συγκεκριμένα δεδομένα εκπαίδευσης και επικύρωσης. *Σχήμα μεταφρασμένο από Goodfellow, 2016.*

Ενότητα 3.
Θεμελιώδεις Έννοιες Τεχνητών Νευρωνικών Δικτύων

1.3.1 Βιολογική Έμπνευση και Αναπαράσταση

Τα τεχνητά νευρωνικά δίκτυα, ή απλά *νευρωνικά δίκτυα* (neural networks, NN), αποτελούν μία υποκατηγορία των αλγόριθμων της μηχανικής μάθησης. Θεωρείται πως έχουν εμπνευστεί από την λειτουργία των νευρώνων του εγκεφάλου καθώς και την διαδικασία που αυτοί χρησιμοποιούν για την «μάθηση» του ανθρώπου, αν και υπάρχουν ερευνητές οι οποίοι διαφωνούν με αυτό καθώς δεν είναι πλήρης η αντιστοίχιση μεταξύ του εγκεφάλου και των νευρωνικών δικτύων (Geron, 2019), ενώ αναφέρεται πως η μάθηση δίχως επίβλεψη μέσω των νευρωνικών δικτύων είναι παρόμοια με κανόνες που έχουν ανακαλυφθεί

στο νευρικό σύστημα του ανθρώπου (Wallisch et al. 2014). Επιπλέον, πρόσφατες έρευνες συνδέουν περαιτέρω τα νευρωνικά δίκτυα με την λειτουργία του εγκεφάλου (Barrett et al. 2019), προτείνοντας νέους τρόπους κατανόησης των διαδικασιών της όρασης, της μνήμης και του ελέγχου κίνησης (Macpherson et al. 2021).

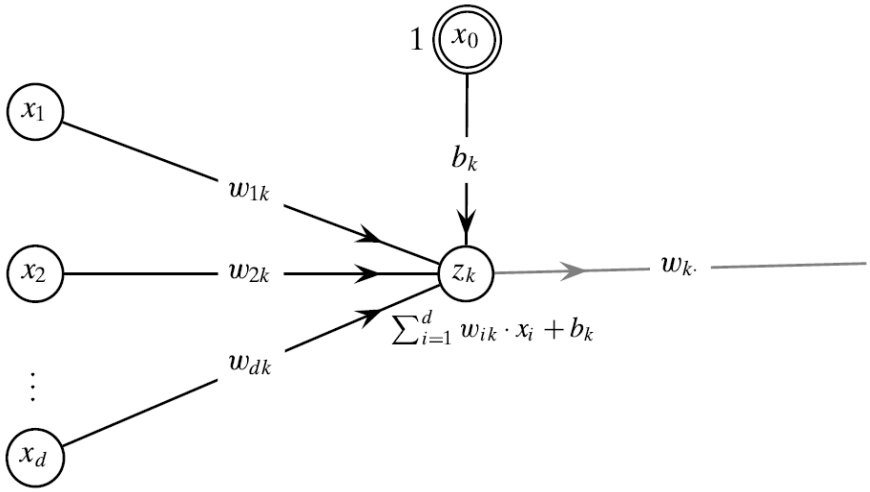


Σχήμα 6. Απεικονίζεται η σύνδεση των συνάψεων μεταξύ δύο βιολογικών νευρωνικών κυττάρων. Κάθε νευρικό κύτταρο (ή νευρώνας) αποτελείται από τους δενδρίτες, το σώμα του κυττάρου, και τον άξονα που οδηγεί στις συνάψεις. Ο ιονισμός των δενδριτών (στον Νευρώνα Α) προκαλεί μία ηλεκτρική αντίδραση που διέρχεται από τον άξονα και περνάει μέσω του ιονισμού των συνάψεων στο επόμενο νευρικό κύτταρο (Νευρώνας Β) μέσω των δενδριτών του. Έπειτα, το ηλεκτρικό σήμα μπορεί να συνεχίσει να μεταδίδεται μέσω του άξονα του Νευρώνα Β στους υπόλοιπους νευρώνες.

Βιολογικά, κάθε νευρικό κύτταρο συνδέεται με κάποιο άλλο μέσω των συνάψεων, από τις οποίες διέρχεται ηλεκτρικό ρεύμα μέσω του άξονα, όταν επέλθει ιονισμός στους δενδρίτες του νευρωνικού κυττάρου. Σύμφωνα με τον Hebb, (2005), όταν ο ηλεκτρικός παλμός διέρχεται επαναλαμβανόμενα από μία συγκεκριμένη σύναψη ενός νευρώνα Α, ενεργοποιώντας έναν νευρώνα Β, τότε η σύναψη *δυναμώνει,* από την άποψη ότι ο νευρώνας Α θα γίνει καλύτερος στην ενεργοποίηση του νευρώνα Β (αλλά όχι αναγκαστικά πως ο νευρώνας Β θα γίνει καλύτερος στην ενεργοποίηση του Α). Αντίστοιχα, εάν *δεν* διέρχεται ρεύμα από κάποια σύναψη μεταξύ των δενδριτών δύο νευρικών κυττάρων, τότε στο πέρασμα του χρόνου η σύναψη θα ιονίζεται δυσκολότερα, συνεπώς ο νευρώνας Α δεν θα συσχετιστεί με την ενεργοποίηση του νευρώνα Β. Αυτό χαρακτηρίζει την σύναψη, και ονομάζεται δυνατότητα δράσης (*action potential)* της σύναψης. Η «μάθηση» στους νευρώνες, επέρχεται κατά την διαδικασία της προσαρμογής των «δυνάμεων» των συνάψεων στο πέρασμα ή όχι των ηλεκτρικών παλμών, με μεγάλη δυνατότητα δράσης να αναφέρεται ως διεγερτική, ενώ μικρή δυνατότητα δράσης ως ανασταλτική (Zaki & Meira, 2020).

Στα *Τεχνητά Νευρωνικά Δίκτυα*, ως νευρικά κύτταρα θεωρούμε τους *κόμβους* ή *νευρώνες* v_i (nodes ή neurons). Ως τις συνάψεις μεταξύ των δενδριτών δύο νευρωνικών κυττάρων, θεωρούμε τα (κατευθυνόμενα) *τόξα* (v_i, v_j) (directed edges), και ως την δυνατότητα δράσης της συγκεκριμένης σύναψης θεωρούμε τα *βάρη* w_{ij} (weights). Τα παραπάνω ορίζουν πλήρως το *σταθμισμένο κατευθυνόμενο γράφημα* (weighted directed graph) του νευρωνικού δικτύου, το οποίο συμβολίζουμε με $G = (V, E)$, όπου κάθε κόμβος $vi \in V$ και κάθε τόξο $(v_i, v_j) \in E$. Τα βάρη δεν αποτελούν μέρος του G, αλλά λαμβάνουν τιμές στους πραγματικούς αριθμούς, δηλαδή $w_{ij} \in \mathbb{R}$, για κάθε $i \neq j$ (Zaki & Meira, 2020). Τα βάρη δηλώνουν την «συνεισφορά» που έχει ένας νευρώνας για τον επόμενο. Συνηθίζεται η συσχέτιση ενός κόμβου με την τιμή του. Οπότε, ο κόμβος v_i που έχει τιμή x_i θα αναφέρεται απλά ως «ο x_i κόμβος». Ως

x_0 συμβολίζουμε την τιμή του *κόμβου πόλωσης* (bias neuron). Το βάρος του τόξου που συνδέει τον κόμβο πόλωσης x_0 με τον κρυφό κόμβο z_k, αποκαλείται *κατώφλι* (bias term) (Zaki & Meira, 2020).
Τα βάρη που συνδέονται με έναν συγκεκριμένο κόμβο, συνιστούν την πληροφορία που εισέρχεται σε αυτόν, η οποία ονομάζεται *καθαρή εισροή* (net input) και συμβολίζουμε ως net_k. Εφόσον η καθαρή εισροή ξεπεράσει το *κατώφλι* b_o (bias), *ο κόμβος θα ενεργοποιηθεί*, παράγοντας την πληροφορία εξόδου z_k, η οποία θα περάσει στον επόμενο κόμβο μέσω του βάρους w_k (Zaki & Meira, 2020), όπως φαίνεται στο Σχήμα 7 που ακολουθεί, όπου σχολιάζονται περαιτέρω οι λεπτομέρειες του σχήματος.



Σχήμα 7. Απεικόνιση της εισαγωγής γραμμικής «πληροφορίας» στον κόμβο z_k. Αυτός συνδέεται με τους κόμβους εισόδου $x_1, x_2, \ldots, x_d$, μέσω των αντίστοιχων βαρών τους, $w_{1k}, w_{2k}, \ldots, w_{dk}$. Επιπλέον, με x_0 στο πάνω μέρος του γραφήματος απεικονίζεται ο κόμβος πόλωσης x_0, με αντίστοιχο *βάρος πόλωσης* b_k. Η «συνολική πληροφορία» που θα δεχτεί ο κόμβος z_k, αποκαλείται *καθαρή εισροή* και ισούται με $net_k = \sum_{i=1}^{d} w_{ik} x_i + b_k$, αφού συνηθίζεται $x_0 = 1$, ενώ η πληροφορία που θα εξάγει ο κόμβος, θα ισούται με z_k. Το πόσο αυτή θα είναι χρήσιμη στον επόμενο νευρώνα, εξαρτάται από το βάρος w_k, μεταξύ του νευρώνα z_k με τον επόμενο. Μεγάλο βάρος συνεπάγεται μεγάλη αξία (δηλαδή ο νευρώνας z_k συνεισφέρει στην πυροδότηση του επόμενου νευρώνα) ενώ μικρό βάρος συνεπάγεται μικρή αξία (δηλαδή πως ο νευρώνας *A δεν* συνεισφέρει στην πυροδότηση του επόμενου νευρώνα *B*). *Σχήμα από Zaki & Meira, 2020, σελ. 368.*

Εφόσον τα νευρωνικά δίκτυα αποτελούν μία κατηγορία αλγορίθμων της μηχανικής μάθησης, είναι αναμενόμενο πως και αυτά «μαθαίνουν» από τα δεδομένα που δίνονται στον χειριστή, τα οποία θα χωριστούν σε δεδομένα εκπαίδευσης, επικύρωσης και ελέγχου. Συνεπώς, ανάλογα την ύπαρξη δεδομένων με επισήμανση και το πώς αυτά είναι κατανεμημένα στο σύνολο των δεδομένων, μπορεί να έχουμε μάθηση υπό επίβλεψη, δίχως επίβλεψη, ήμι επίβλεψη αλλά ακόμα και ενισχυτική μάθηση, ενεργή μάθηση, μάθηση αναπαραστάσεων, βαθιά μάθηση, κατανεμημένη ή παράλληλη μάθηση, μάθηση με μεταβίβαση ή μάθηση με πυρήνες. Επιπλέον, μετά την μάθηση μίας συνάρτησης f για την επιθυμητή διεργασία, όπως η παλινδρόμηση, η κατηγοριοποίηση ή η συσταδοποίηση, θα χρειαστούν ενέργειες επικύρωσης ώστε να γνωρίζουμε πόσο καλά γενικεύονται τα αποτελέσματα του μοντέλου μας, και πιθανώς να χρειαστεί να αλλάξουμε κάποιες από τις υπερπαραμέτρους του νευρωνικού δικτύου που μελετάμε. Δηλαδή, θα εφαρμόσουμε όσα έχουμε αναφέρει σχετικά με τους αλγόριθμους της μηχανικής μάθησης στην προηγούμενη ενότητα.
Προτού επεκταθούμε στις διάφορες αρχιτεκτονικές που μπορεί να έχουν τα νευρωνικά δίκτυα, θα αναφερθούμε ακόμη σε μία αναγκαία, για την εκπαίδευση των νευρωνικών δικτύων, έννοια. Αυτή είναι η *συνάρτηση ενεργοποίησης*. Έπειτα θα επεκταθούμε και θα δώσουμε παραδείγματα για τις *συναρτήσεις απώλειας*, και τους *αλγόριθμος βελτιστοποίησης* που απαιτούνται για την ολοκλήρωση της εκπαίδευσης των νευρωνικών δικτύων, όπου ο σκοπός είναι η *ελαχιστοποίηση της συνάρτησης απώλειας*, μέσω των

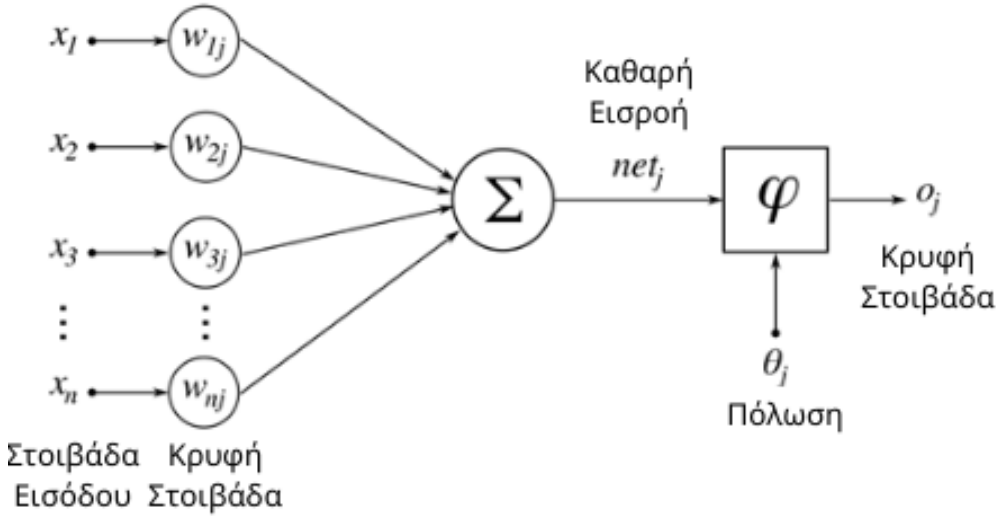
αλγόριθμων βελτιστοποίησης. Στην τελευταία ενότητα, θα αναφέρουμε κάποιες συναρτήσεις κόστους που απαιτούνται για τις ενέργειες της επικύρωσης των αποτελεσμάτων του νευρωνικού δικτύου.

1.3.2 Συνάρτηση Ενεργοποίησης

Η συνάρτηση ενεργοποίησης *f* (activation function), στην απλή της περίπτωση, αναπαριστά εάν διέρχεται ή όχι ηλεκτρικό ρεύμα από τη σύναψη μεταξύ των νευρώνων, και αποκαλείται *δυαδική συνάρτηση ενεργοποίησης,* ή *συνάρτηση Heaviside,* καθώς η παράγωγός της δίνει την διάσημη συνάρτηση του Dirac. Αναλυτικά, μία *συνάρτηση ενεργοποίησης f*, θα δράσει στην πληροφορία εισόδου, net_k, και θα δώσει την τελική τιμή z_k του συγκεκριμένου κόμβου, η οποία θα ισούται με:

$$z_k = f(net_k) = f\left(b_k + \sum_{i=1}^{d} w_{ik} \cdot x_i\right) = f\left(b_k + \boldsymbol{w_k^T x}\right),$$

όπου $\boldsymbol{w}_k = (w_{1k}, w_{2k}, \dots, w_{dk})^T \in \mathbb{R}^{\boldsymbol{d}}$ τα βάρη των τόξων και $\boldsymbol{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^{\boldsymbol{d}}$ οι τιμές εισόδου, των δεδομένων εισόδου (Zaki & Meira, 2020). Συγκεκριμένα, η καθαρή εισροή (ή πληροφορία εισόδου) net_k χωρίζεται σε δύο μέρη: το γραμμικό συνδυασμό των τιμών εισόδου επί τα βάρη τους, $\sum_{i=1}^{d} w_{ik} \cdot x_i$ συν την τιμή του κόμβου πόλωσης, b_k. Αυτό έχει ως αποτέλεσμα η καθαρή εισροή να δρα ως ένας *συγγενικός μετασχηματισμός* (affine transformation). Επιπρόσθετα, επειδή αποτελεί άθροισμα της γραμμικής συνάρτησης $\sum(\boldsymbol{w_k^T}, \boldsymbol{x}) = \sum_{i=1}^{d} w_{ik} \cdot x_i$ με το διάνυσμα $\boldsymbol{b} = (0, \dots, 0, b_k, 0, \dots, 0)$, αυτή θα είναι ένας *γραμμικός μετασχηματισμός* (linear transformation) (Polanco, 2018). Ο μετασχηματισμός του γραμμικού μετασχηματισμού net_k (όταν το $b_k = 0$) μέσω της συνάρτησης ενεργοποίησης, απεικονίζεται στο ακόλουθο σχήμα.

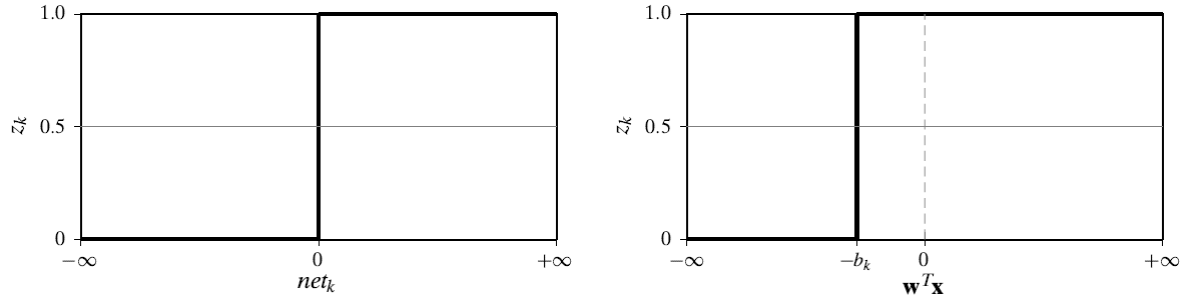


Σχήμα 8. Ανάλυση της πληροφορίας που δέχεται ένας νευρώνας. Έχουμε συμβολίσει με $x_1, x_2, \dots, x_n$ τις τιμές των αντίστοιχων κόμβων εισόδου $v_1, v_2, \dots, v_n$ ενώ με $w_{1j}, w_{2j}, \dots, w_{nj}$ αναπαριστούμε το βάρος του τόξου από τον κόμβο v_i προς τον κόμβο o_j. Συγκεκριμένα, η τιμή o_j του αντίστοιχου νευρώνα, αποτελεί έναν μετασχηματισμό των δεδομένων εισόδου $x_1, x_2, \dots, x_n$, σε δύο μέρη: το πρώτο είναι γραμμικό λόγω του net_j, και το δεύτερο μη – γραμμικό, λόγω της συναρτήσεως ενεργοποίησης που θα δράσει στο net_j, δηλαδή $f(net_j)$ (στο σχήμα η συνάρτηση ενεργοποίησης συμβολίζεται με φ). Επιπλέον, είναι προφανές πως εάν η συνάρτηση ενεργοποίησης είναι γραμμική, τότε συνολικά ο μετασχηματισμός θα είναι ένας γραμμικός μετασχηματισμός των δεδομένων εισόδου και των αντίστοιχων βαρών τους. *Από τον χρήστη Christlb, commons.wikimedia.org*

Δυαδική συνάρτηση ενεργοποίησης,
Εφόσον αυτή αναπαριστά εάν διέρχεται ή όχι ηλεκτρικό ρεύμα από τον νευρώνα, έχει τον απλό τύπο:

$$f(net_k) = \begin{cases} 0, & \text{εάν } net_k \leq 0 \\ 1, & \text{εάν } net_k > 0 \end{cases} = \begin{cases} 0, & \text{εάν } \boldsymbol{w^T x} \leq -b_k \\ 1, & \text{εάν } \boldsymbol{w^T x} > -b_k \end{cases}$$

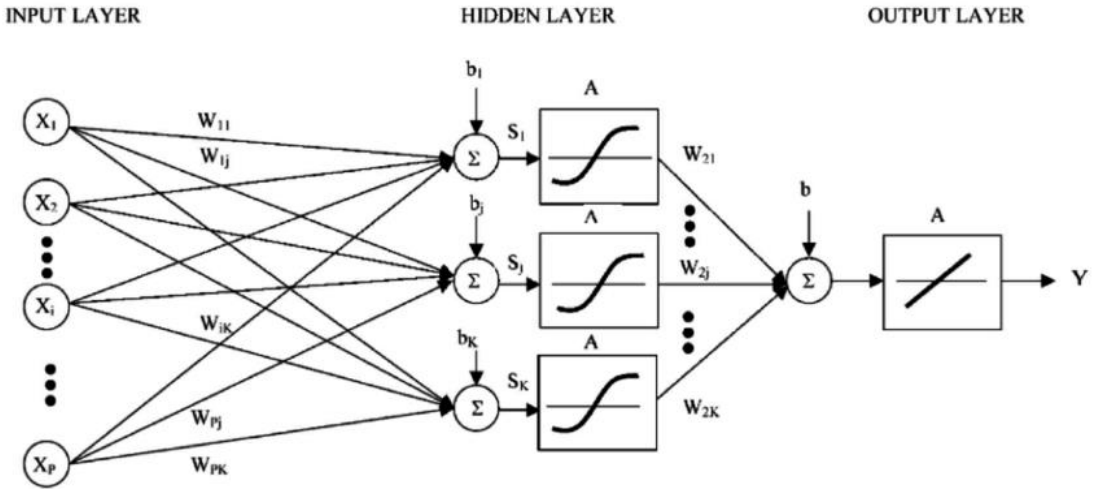
Όπου στην δεύτερη ισότητα χρησιμοποιήσαμε την σχέση $net_k = \sum_{i=1}^{d} w_{ik} x_i + b_k$. Οπότε, ο νευρώνας θα ενεργοποιείται όταν το εσωτερικό γινόμενο $\boldsymbol{w^T x}$ λαμβάνει τιμή μεγαλύτερη του $-b_k$, δηλαδή η ενεργοποίηση του νευρώνα σχετίζεται άμεσα με το βάρος του κόμβου πόλωσης. *Κάθε* συνάρτηση ενεργοποίησης χρησιμοποιεί με κάποιον τρόπο την τιμή της πόλωσης. Το γράφημα της δυαδικής συνάρτησης ενεργοποίησης απεικονίζεται στο ακόλουθο Σχήμα 9.



(c) Step (z_k versus net_k) (d) Step (z_k versus $\mathbf{w}^T\mathbf{x}$)

Σχήμα 9. Αριστερά, είναι το γράφημα της δυαδικής συνάρτησης ενεργοποίησης όταν θεωρούμε πως λαμβάνει τιμές στο net_k, ενώ δεξιά όταν αναφερόμαστε ως προς το εσωτερικό γινόμενο $\boldsymbol{w^T x}$. (Zaki & Meira, 2020)

Συνολικά, οι συναρτήσεις ενεργοποίησης αξιοποιούνται σε όλους τους κόμβους ενός νευρωνικού δικτύου, όμως αυτό εξαρτάται από την αρχιτεκτονική του νευρωνικού δικτύου, δηλαδή με ποιόν τρόπο αλλάζει το *σταθμισμένο κατευθυνόμενο γράφημα* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ του νευρωνικού δικτύου. Προς το παρόν, αναφέρουμε πως ένα νευρωνικό δίκτυο στην απλή του μορφή θα έχει μία *στοιβάδα εισόδου*, όπου θα εισάγονται οι τιμές των δεδομένων, τα οποία δεδομένα θα αλλάζουν ανάλογα το την εργασία του νευρωνικού δικτύου, είτε αυτή είναι η μάθηση, η επικύρωση, είτε η πρόβλεψη σε ένα καινούργιο σύνολο δεδομένων. Στο τέλος θα έχει την *στοιβάδα εξόδου*, η οποία θα δίνει το τελικό αποτέλεσμα που ζητάμε, για παράδειγμα εάν επιθυμούμε μία διεργασία απλής παλινδρόμησης, η στοιβάδα εξόδου θα έχει έναν νευρώνα o_j, ο οποίος θέλουμε να λαμβάνει τιμές στους πραγματικούς αριθμούς, δηλαδή $o_j \in \mathbb{R}$. Ενδιάμεσα στις στοιβάδες εισόδου και εξόδου, βρίσκονται οι λεγόμενες *κρυφές στοιβάδες*. Έως τώρα, ο νευρώνας που αναφέραμε ως z_k βρισκόταν στην πρώτη κρυφή στοιβάδα. Περισσότερα αναφέρονται στο Σχήμα 10 που ακολουθεί.



Σχήμα 10. Αριστερά, βλέπουμε την στοιβάδα εισόδου (Input Layer) η οποία έχει τιμές $x_1, x_2, \dots, x_P$, με τα αντίστοιχα βάρη $w_{1k}, w_{2k}, \dots, w_{PK}$ που τους συνδέουν με τον $k-$οστό κόμβο της κρυφής στοιβάδα (Hidden Layer) στα δεξιά της στοιβάδας εισόδου, όπου φαίνεται ξεκάθαρα η ανάλυση της πληροφορίας που δέχεται ο κάθε $k-$οστός κρυφός κόμβος z_k, σε δύο μέρη: πρώτα το άθροισμα των τιμών $\sum x_i w_{ik} + b_k$, και έπειτα η εισαγωγή αυτής της τιμής στην συνάρτηση ενεργοποίησης, που στο σχήμα συμβολίζεται με S_k, και ισούται με την *Σιγμοειδή συνάρτηση*, που αναφέρουμε παρακάτω. Έπειτα, στον ένα κόμβο στην στοιβάδα εξόδου (δεξιά της κρυφής στοιβάδας), οι τιμές των κρυφών κόμβων $z_k = S_k(net_k)$ εισάγονται πάλι σε δύο μέρη: πρώτα, στο γραμμικό άθροισμα $\sum z_k w_{2k} + b$ και έπειτα στην γραμμική συνάρτηση ενεργοποίησης, η οποία στο σχήμα είναι η ταυτοτική. Αυτή θα μπορούσε να αποτελεί μία διεργασία απλής παλινδρόμησης. *Σχήμα από Vuckovic, et al. 2002.*

Οι συναρτήσεις ενεργοποίησης χρησιμοποιούνται στους κόμβους της κρυφής στοιβάδας (κρυφούς κόμβους) και στους κόμβους της στοιβάδας εξόδου (κόμβους εξόδου), ενώ δεν συνηθίζεται να χρησιμοποιούνται στους κόμβους της στοιβάδας εισόδου (κόμβος εισόδου). Συνηθίζεται μία συνάρτηση ενεργοποίησης να εφαρμόζεται στους κόμβους από μία ολόκληρη στοιβάδα. Είναι εφικτό και χρήσιμο να είναι διαφορετικές οι συναρτήσεις ενεργοποίησης που χρησιμοποιούμε στις κρυφές στοιβάδες με αυτές που χρησιμοποιούμε στην στοιβάδα εξόδου. Για παράδειγμα, σε ένα πρόβλημα δυαδικής κατηγοριοποίησης θέλουμε ο κόμβος εξόδου να παίρνει τιμές $o_1 \in [0,1]$, ώστε να αναπαριστά μία πιθανότητα., ενώ σε ένα πρόβλημα απλής παλινδρόμησης θέλουμε ο κόμβος εξόδου να παίρνει τιμές $o_1 \in \mathbb{R}$. Για αυτό πρέπει να χρησιμοποιήσουμε μία αντίστοιχη συνάρτηση ενεργοποίησης, η οποία θα έχει ως σύνολο τιμών το $[0,1]$ ή το $\mathbb{R}$, αντίστοιχα, η οποία όμως ίσως θέλουμε να είναι μία διαφορετική στις κρυφές στοιβάδες.

Αναφέρουμε πως είναι επιθυμητή η *μη – γραμμικότητα* και η *συνέχεια της παραγώγου* μίας συναρτήσεως ενεργοποίησης. Η μη – γραμμικότητα βοηθάει στον περιορισμό της χωρητικότητας του μοντέλου (δηλαδή την ικανότητά του να προσεγγίζει τις επιθυμητές συναρτήσεις μάθησης), καθώς δίχως συναρτήσεις ενεργοποίησης το νευρωνικό δίκτυο θα μπορούσε να εξάγει οποιαδήποτε τιμή στο $[-\infty, \infty]$ (Feng & Lu, 2019). Επιπλέον, η συνέχεια της παραγώγου συνεισφέρει στην ποιότητα της μάθησης (Nwankpa et al. 2018), αντίστοιχα. Επιπλέον, αξίζει να αναφερθεί πως οι συναρτήσεις ενεργοποίησης επιτρέπουν στα νευρωνικά δίκτυα την προσέγγιση οποιασδήποτε συνεχούς συνάρτησης, ακόμα και με την χρήση μόνο μίας κρυφής στοιβάδας (Funahashi, 1989) ενώ πρόσφατα αποδείχτηκε πως, όταν υπάρχει μία κρυφή στοιβάδα, υπάρχει δυαδική συνάρτηση η οποία δεν μπορεί να προσεγγιστεί (Inoue et al. 2020). Σε όσα ακολουθούν, θα αναφέρουμε επιπλέον κάποιες συναρτήσεις ενεργοποίησης που βρίσκουν εφαρμογές στις στοιβάδες των νευρωνικών δικτύων. Αυτές αναφέρονται συνοπτικά μαζί με τον τύπο, το σύνολο τιμών

τους, τα αρνητικά και τα θετικά τους, καθώς και το είδος διεργασιών που συνηθίζεται να αξιοποιηθούν, στον Πίνακα 1 στο τέλος της ενότητας, μαζί με ένα σχήμα από τις γραφικές τους παραστάσεις.

<u>Γραμμικές συναρτήσεις ενεργοποίησης</u>

Η γραμμική συνάρτηση ενεργοποίησης δίνεται στον ακόλουθο τύπο, όπου έχουμε πολλαπλασιάσει την εισροή με μία σταθερά $k \in \mathbb{N}$.

$$f(x) = kx, x \in \mathbb{R}, \qquad k \in \mathbb{N}$$

όπου ως x_i αναφέρουμε την τιμή που λαμβάνει ο $i-$οστός κόμβος, είτε κρυφός είτε εξόδου. Εάν επιθυμούμε η καθαρή εισροή να μην μεταβληθεί καθόλου από την συνάρτηση ενεργοποίησης, μπορούμε να χρησιμοποιήσουμε την ταυτοτική συνάρτηση ως συνάρτηση ενεργοποίησης, δηλαδή θα θέσουμε $k = 1$. Η παράγωγος είναι

$$f'(x) = k, \forall x \in \mathbb{R}, \qquad k \in \mathbb{N}$$

Δηλαδή ίση με την σταθερά k. Από αυτό, καταλαβαίνουμε πως δεν εξαρτάται από τα δεδομένα εισόδου x, δηλαδή ούτε από το net_k. Ως αποτέλεσμα, όπως θα δούμε στην ενότητα της εκπαίδευσης των νευρωνικών δικτύων, *δεν είναι εφικτό να εκπαιδεύσουμε το νευρωνικό δίκτυο* μέσω της ελαχιστοποίησης της συνάρτησης απώλειας. Επιπλέον, δεν βοηθάει στον περιορισμό των τιμών που μπορεί να λάβει το νευρωνικό δίκτυο όταν αξιοποιηθεί, καθώς έχει σύνολο τιμών το $(-\infty, \infty)$ (Feng & Lu, 2019). Συνεπώς, είναι προφανής ο λόγος που προτιμάται τουλάχιστον ένας συνδυασμός μη – γραμμικών συναρτήσεων ενεργοποίησης, καθώς αυτές θα μπορέσουν να περιορίσουν το δυνατό σύνολο τιμών που μπορεί να λάβει ένα νευρωνικό δίκτυο, με αποτέλεσμα να αυξήσουν την μάθηση της επιθυμητής συνάρτησης.

<u>Μη γραμμικές Συναρτήσεις Ενεργοποίησης</u>

Διορθωμένη Γραμμική Μονάδα (Rectified Linear Unit (ReLU))

Θα αναφερόμαστε σε αυτήν στην πορεία ως ReLU. Προτάθηκε πρώτα από τους Nair & Hinton το 2010 όπου προτάθηκε πως βελτιώνουν την διαδικασία της μάθησης. Αργότερα, βρέθηκε πως δρα αποτελεσματικά σε διεργασίες κατηγοριοποίησης εικόνας και στην εξόρυξη κειμένου (Feng & Lu, 2019). Ο τύπος και αντίστοιχα η παράγωγος της ReLU δίνονται στην ακόλουθη εξίσωση

$$f(x) = \begin{cases} 0, & \text{εάν } x < 0 \\ x, & \text{εάν } x > 0 \end{cases}, \qquad f'(x) = \begin{cases} 0, & \text{εάν } x < 0 \\ 1, & \text{εάν } x > 0 \end{cases}$$

Χωρίς να ορίζεται στο 0, καθώς δεν ορίζεται εκεί η παράγωγος. Επειδή έχει τιμή παραγώγου 0 για τις αρνητικές τιμές, όταν έχουμε αρνητικές τιμές σε έναν νευρώνα αυτές δεν θα ανανεωθούν, το οποίο αποκαλείται ως το πρόβλημα «Dying ReLU». Επιπλέον, δεν έχει άνω φράγμα συνεπώς μπορεί κατά την μάθηση ένας νευρώνας να συνεισφέρει πολύ στην μάθηση, το οποίο δεν είναι θεμιτό στα νευρωνικά δίκτυα (Feng & Lu, 2019). Επιπλέον, αναφέρεται πως δεν συνεισφέρει και τόσο πολύ στα προβλήματα των εξαφανιζόμενων κλίσεων, ακριβώς επειδή δεν υπολογίζει τις τιμές κάτω από το 0 (Nair & Hinton, 2010). Βέβαια, αργότερα προτάθηκαν τροποποιήσεις της ReLU, με βελτιωμένες επιδώσεις σε διάφορους τομείς (Feng & Lu, 2019), όπως αυτές που αναφέρουμε παρακάτω.

Διορθωμένη Γραμμική μονάδα με διαρροή (Leaky ReLU)

Θα αναφερόμαστε σε αυτήν ως LReLU. Έχει πολλές εφαρμογές σε ακουστικά μοντέλα (Feng & Lu, 2019). Ο τύπος και η παράγωγος δίνονται από :

$$f(x_{ij}) = \begin{cases} a_i x_{ij}, & \text{εάν } x < 0 \\ x_{ij}, & \text{εάν } x > 0 \end{cases}, \qquad f'(x_{ij}) = \begin{cases} a_i, & \text{εάν } x < 0 \\ 1, & \text{εάν } x > 0 \end{cases}$$

Το a_i είναι μία σταθερά, η οποία αλλάζει ανάλογα την στοιβάδα του νευρώνα και συνήθως λαμβάνει πολύ μικρές τιμές, κοντά στο 0.01. Επειδή αυτή λαμβάνεται υπόψη για τις αρνητικές τιμές, καταφέρνει να αντιμετωπίσει το πρόβλημα που αναφέραμε, το «Dying ReLU» διατηρώντας παρόμοιες ιδιότητες (Feng & Lu, 2019).

Παραμετρική ReLU (PReLU)

Μία περαιτέρω βελτίωση της ReLU όσο αφορά τις αρνητικές τιμές, περιλαμβάνει την σταθερά α που είδαμε στην LReLU. Θα αναφερόμαστε σε αυτήν ως PReLU. Εδώ, θεωρείται πως η σταθερά a_i πλέον *είναι μεταβλητή,* συγκεκριμένα *μπορεί να την μάθει το νευρωνικό δίκτυο*. Αποτελεί δηλαδή μία ακόμη παράμετρο που θα μάθει το νευρωνικό δίκτυο, η οποία αλλάζει από στοιβάδα σε στοιβάδα (Feng & Lu, 2019). Ο τύπος και η παράγωγος δίνονται από:

$$f(x_{ij}) = \begin{cases} a_i x_{ij}, & \text{εάν } x < 0 \\ x_{ij}, & \text{εάν } x > 0 \end{cases}, \qquad f'(x_{ij}) = \begin{cases} a_i, & \text{εάν } x < 0 \\ 1, & \text{εάν } x > 0 \end{cases}$$

Τυχαιοποιημένη ReLU με διαρροή (Randomized Leaky Rectified Linear Unit, RReLU)

Θα αναφερόμαστε σε αυτήν ως RReLU. Μπορούμε επιπλέον να κάνουμε την τιμή α να αλλάζει όχι μόνο ανά στοιβάδα, αλλά και ανά κόμβο, με τυχαίο τρόπο, συνεπώς αυτή καθίσταται πλέον μία τυχαία μεταβλητή. Θεωρούμε επιπλέον πως η τυχαία μεταβλητή α_{ij}, ως προς την στοιβάδα i και τον κόμβο j, ακολουθεί την ομοιόμορφη κατανομή:

$$\alpha_{ij} \sim U(l, u), \;\; [l, u] \subseteq [0,1)$$

Συνεπώς, θα έχει μέση τιμή $E(a_{ij}) = \frac{l+u}{2}$. Αναφέρεται επίσης πως βρέθηκαν καλύτερα αποτελέσματα σε σύγκριση με τις ReLU, LReLU και PReLU. Η βελτίωση ως προς τις υπόλοιπες αφορά την μείωση της υπερπροσαρμογής των δεδομένων εκπαίδευσης. (Feng & Lu, 2019). Ο τύπος και η παράγωγος δίνονται από:

$$f(x_{ij}) = \begin{cases} a_{ij} x_{ij}, & \text{εάν } x \leq 0 \\ x_{ij}, & \text{εάν } x > 0 \end{cases}, \qquad f'(x_{ij}) = \begin{cases} a_{ij}, & \text{εάν } x \leq 0 \\ 1, & \text{εάν } x > 0 \end{cases}.$$

όπου ως x_{ij} αναφέρουμε την τιμή στον κόμβο j της στοιβάδας i.

Εκθετική Γραμμική Μονάδα (Exponential Linear Unit)

Θα αναφερόμαστε σε αυτήν ως ELU. Αναφέρεται πως επιταχύνει την μάθηση και αποφεύγει το πρόβλημα των εξαφανιζόμενων κλίσεων (Feng & Lu, 2019). Ο τύπος και η παράγωγος δίνονται από:

$$f(x) = \begin{cases} a_i(e^x - 1), & \text{εάν } x \leq 0 \\ x, & \text{εάν } x > 0 \end{cases}, \qquad f'(x_{ij}) = \begin{cases} a_i, & \text{εάν } x \leq 0 \\ 1, & \text{εάν } x > 0 \end{cases}.$$

Οι Xu et al. (2015) αναφέρουν επίσης ότι έχει βρεθεί πως έχει σημαντική βελτίωση της επίδοσης σε σχέση με την ReLU και άλλων βελτιωμένων συναρτήσεων ενεργοποίησης, σε προβλήματα όπως το ImageNet, CIFAR-100, CIFAR-10..

Σιγμοειδής (Sigmoid)

Ή σιγμοειδής συνάρτηση πήρε το όνομά της από το γράφημά της, το οποίο μοιάζει με «S». Λαμβάνει τιμές στο $[0,1]$. Έχει τύπο και παράγωγο τις ακόλουθες συναρτήσεις:

$$sigmoid(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}, \qquad (sigmoid)'(x) = \frac{e^{-x}}{(1 + e^x)^2}, x \in \mathbb{R}.$$

Σύμφωνα με τους Feng & Lu, (2019), επειδή έχει μεγάλη αύξηση η κλίση της συνάρτησης, *μικρές* αλλαγές στην καθαρή εισροή που είναι κοντά στο 0, μπορεί να αντιστοιχίσουν σε *μεγάλες* αλλαγές στην συμπεριφορά του νευρώνα, ενώ αντίστοιχα τιμές οι οποίες είναι μακριά από το 0 τείνουν να μην αλλάζουν καθόλου, καθώς εκεί ο ρυθμός μεταβολής της συνάρτησης δεν είναι μεγάλος. Αυτό συνεισφέρει στο πρόβλημα των «εξαφανιζόμενων κλίσεων» (vanishing gradients) το οποίο μπορεί να συμβεί κατά την μάθηση, και έχει ως αποτέλεσμα τον τερματισμό της μάθησης στον συγκεκριμένο νευρώνα (θα έχει βάρος ίσο με 0), κάτι το οποίο δεν είναι επιθυμητό, καθώς μπορεί να είχε σημαντική συνεισφορά (Feng & Lu, 2019). Συνήθως για τιμή μεγαλύτερη του 0.5 αντιστοιχεί σε μία κατηγορία ενώ για τιμή μικρότερη από 0.5 δεν αντιστοιχεί (Zaki & Meira, 2020).

Υπερβολική Εφαπτομένη (tanh)

Είναι ένας συνδυασμός της σιγμοειδούς συνάρτησης, όπως φαίνεται από τον τύπο και την παράγωγο:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} = \frac{2}{1 + e^{-2x}} - 1 = 2sigmoid(2x) - 1$$

$$\tanh'(x) = \frac{4e^{-2x}}{(1 + e^{-2x})^2}.$$

Λαμβάνει τιμές στο $[-1,1]$. Εφόσον αποτελεί έναν γραμμικό συνδυασμό της $sigmoid(2x)$, είναι προφανές ότι θα έχει παρόμοια χαρακτηριστικά με την σιγμοειδή, μόνο που θα είναι πλέον πιο ακραία και εμφανή, δηλαδή λόγω της μεγάλης κλίσης ενώ το πρόβλημα των εξαφανιζόμενων κλίσεων θα είναι ακόμη πιο εμφανές. Όμως, καθώς λαμβάνει τιμές στο $[-1,1]$, προκύπτει πως για τιμές του x κοντά στο 0, *δεν* θα έχουμε απρόσμενες ενεργοποιήσεις των νευρώνων. Επιπλέον, οι περισσότερες τιμές της θα είναι κοντά στο 0, με αποτέλεσμα την διευκόλυνση της μάθησης (Feng & Lu, 2019).

Συνάρτηση Softmax

Θα αναφερόμαστε σε αυτήν ως Softmax. Αποτελεί μία γενίκευση της σιγμοειδούς συνάρτησης. Χρησιμοποιείται συνήθως στην τελευταία στοιβάδα του νευρωνικού δικτύου για διεργασίες πολλών κλάσεων, ενώ εξαρτάται από την καθαρή εισροή, όχι μόνο του κόμβου που ανήκει, net_k αλλά και ολόκληρης της στοιβάδας εξόδου, $\boldsymbol{net} = (net_1, net_2, \dots, net_p)$, των κόμβων εξόδου $o_1, o_2, \dots, o_p$. Έχει τύπο:

$$f(x_k|\mathbf{x}) = \frac{\exp\{x_k\}}{\sum_i^p \exp\{x_i\}}.$$

Επειδή έχει δύο μεταβλητές, η παράγωγός της γίνεται μερική ως προς x_k (όπου λαμβάνει τιμές net_k) και μερική ως προς x_j (η οποία λαμβάνει τιμές net_j). Συνολικά, προκύπτει:

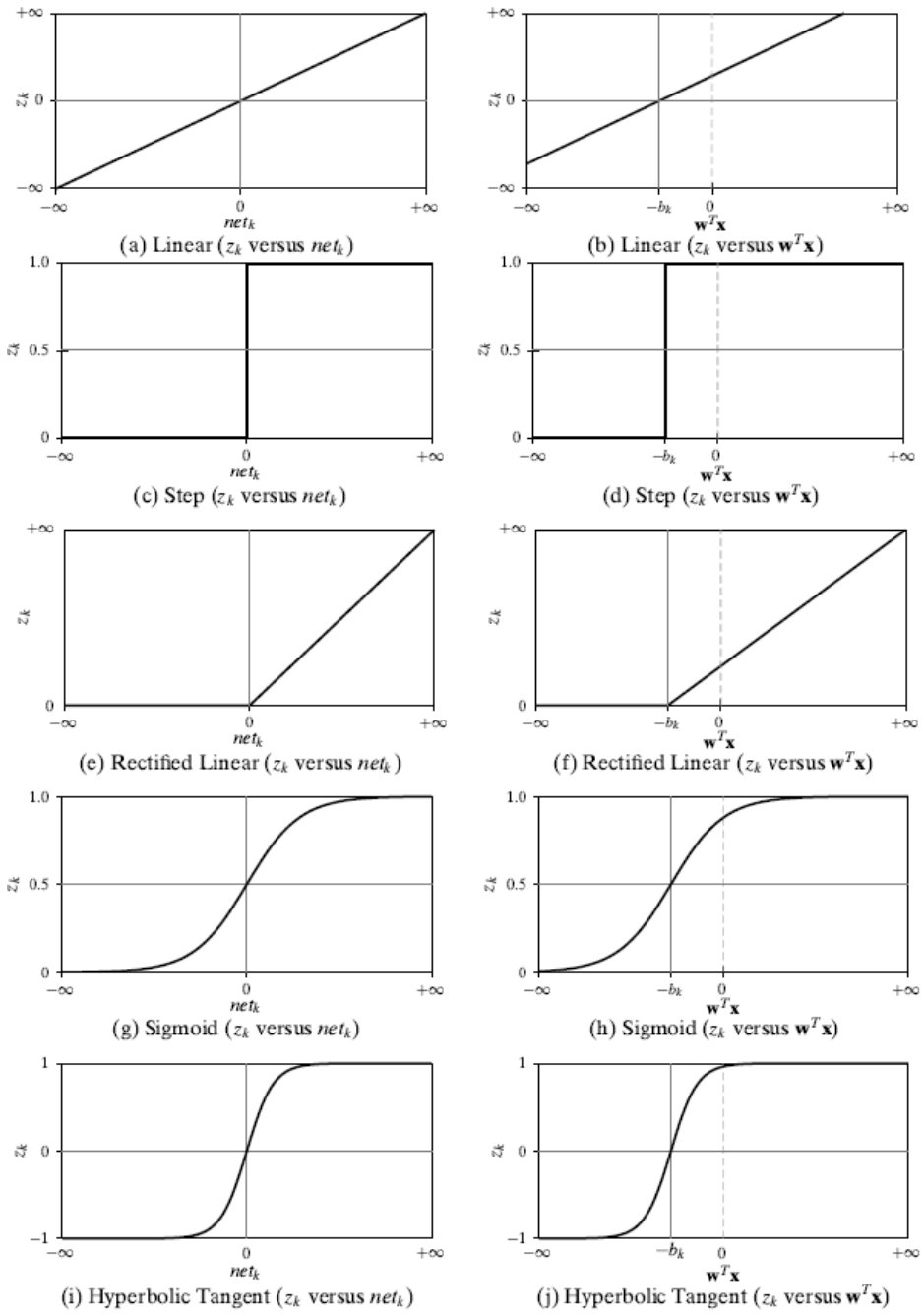
$$\frac{\partial f(x_j|\mathbf{x})}{\partial x_k} = \begin{cases} o_j \cdot (1 - o_j), & \text{εάν } k = j \\ -o_k \cdot o_k, & \text{εάν } k \neq j \end{cases}$$

όπου ως o_i αναφέρουμε την τιμή του i – οστού κόμβου εξόδου. Επιπλέον, αναφέρεται πως έχει συμπεριφορά παρόμοια με την σιγμοειδή συνάρτηση. (Zaki & Meira, 2020).

Περισσότερα για τις συναρτήσεις ενεργοποίησης μπορούν να βρεθούν στο άρθρο των Nwankpa et al. (2018) ενώ οι Panigrahi et al. (2019) αναφέρουν τις επιδράσεις των συναρτήσεων ενεργοποίησης σε νευρωνικά δίκτυα τα οποία έχουν πάρα πολλές παραμέτρους (overparametrized).

Συναρτήσεις Ενεργοποίησης				
Συνάρτηση Ενεργοποίησης	**Τύπος**	**Σύνολο Τιμών**	**Πλεονεκτήματα**	**Μειονεκτήματα**
Δυαδική/Βήματος	$\begin{cases} 0, & \text{εάν } x \leq 0 \\ 1, & \text{εάν } x > 0 \end{cases}$	$\{0,1\}$	Γρήγορη Σύγκλιση.	Μη παραγωγίσιμη, Μη συνεχής
Γραμμική	$\begin{cases} kx, & \text{εάν } x \leq 0 \\ kx, & \text{εάν } x > 0 \end{cases}$	$(-\infty, \infty)$	Μονότονη.	Άπειρο σύνολο τιμών, Σταθερή παράγωγος
ReLU	$\max(0, x)$	$[0, \infty)$	Μονότονη, Γρήγορη Σύγκλιση. Ανθεκτική στις εξαφανιζόμενες κλίσεις.	Μη συνεχής, Μη παραγωγίσιμη, Δίχως άνω φράγμα, Πρόβλημα σε $x < 0$.
LReLU	$\max(\alpha x, x)$, *όπου α «μικρό»*	$(\alpha, \infty]$	Μονότονη, Γρήγορη Σύγκλιση.	Παρόμοια με ReLU, Όμως είναι καλύτερη στις αρνητικές τιμές από την ReLU
PReLU	$\begin{cases} a_i x_{ij}, & \text{εάν } x < 0 \\ x_{ij}, & \text{εάν } x > 0 \end{cases}$	(a_i, ∞)	Μονότονη, Γρήγορη Σύγκλιση, Μάθηση a_i ανά στοιβάδα.	Παρόμοια με LReLU Παραπάνω παράμετροι προς μάθηση
RReLU	$\begin{cases} a_{ij} x_{ij}, & \text{εάν } x \leq 0 \\ x_{ij}, & \text{εάν } x > 0 \end{cases}$	(a_i, ∞)	Μονότονη, Γρήγορη Σύγκλιση, Μάθηση a_i ανά κόμβο, Μείωση Υπερπροσαρμογής στα δεδομένα εκπαίδευσης.	Παρόμοια με LReLU, Παραπάνω παράμετροι προς μάθηση
ELU	$\begin{cases} a_i(e^x - 1), & \text{εάν } x \leq 0 \\ x, & \text{εάν } x > 0 \end{cases}$	(a_i, ∞)	Μονότονη, Γρηγορότερη σύγκλιση από την RReLU, Αντιμετωπίζει τις εξαφανιζόμενες κλίσεις.	Παρόμοια με LReLU, Όμως είναι καλύτερη στις αρνητικές τιμές από την LReLU
Σιγμοειδής	$\frac{1}{1 + e^{-x}}$	$(0,1)$	Μονότονη, Παραγωγίσιμη.	Εξαφανιζόμενες κλίσεις, Ευαισθησία κοντά στο 0.
Υπερβολική εφαπτομένη	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$(-1,1)$	Μονότονη, Παραγωγίσιμη, Αντιμετωπίζει τις εξαφανιζόμενες κλίσεις καλύτερα από την Σιγμοειδή.	Εντονότερα φαινόμενα εξαφανιζόμενων κλίσεων από τη Σιγμοειδή.
Softmax	$\frac{\exp\{x_k\}}{\sum_i^p \exp\{x_i\}}$	$(0,1)$	Συμπεριφορά παρόμοια της Σιγμοειδούς.	Συμπεριφορά παρόμοια της Σιγμοειδούς.

Πίνακας 1. Συναρτήσεις Ενεργοποίησης



Σχήμα 11. Συναρτήσεις ενεργοποίησης. Στην αριστερή στήλη, απεικονίζονται οι τιμές $f(net_k)$, όπου όπως ξέρουμε $net_k = \sum_{i=1}^{d} w_{ik}x_i + b_k = \boldsymbol{w^t x} + b_k$, οι οποίες είναι ισοδύναμες με τις τιμές $f(x), x \in \mathbb{R}$. Στην δεξιά στήλη, είναι οι ίδιες συναρτήσεις αλλά μετατοπισμένες. Αυτό συμβαίνει μετά από την αντικατάσταση του net_k και την επίλυση της ανίσωσης $net_k \geq 0$. *Από Zaki & Meira, 2020.*

1.3.3. Συνάρτηση Απώλειας

Οι *συναρτήσεις απώλειας ή σφάλματος* (*loss functions* ή *error function*), είναι αυτές που επιτρέπουν την μάθηση του νευρωνικού δικτύου. Θεωρούμε το σύνολο των δεδομένων εκπαίδευσης $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \boldsymbol{D} \subset \mathbb{R}^{d\times p}$, αντιστοιχίζοντας τα n – το πλήθος δεδομένα εισόδου $\boldsymbol{x}_i \in \mathbb{R}^d$ με την αντίστοιχη πραγματοποίηση $\boldsymbol{y}_i \in \mathbb{R}^p$, της τιμής απόκρισης, με διάνυσμα $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \dots, y_{id})$. Συνεπώς, θεωρούμε προς το παρόν πως βρισκόμαστε σε μία διεργασία *επιβλεπόμενης μάθησης*. Συγκεκριμένα, η συνάρτηση ελέγχου θα αποτελέσει το μέτρο απόκλισης μεταξύ της τιμής απόκρισης που έχει παρατηρηθεί, $\boldsymbol{y}_i$, με την τιμή πρόβλεψης του νευρωνικού δικτύου, $\boldsymbol{o}_i$, με διάνυσμα $\boldsymbol{o}_i = (o_{i1}, o_{i2}, \dots, o_{id})$. Οπότε, εφόσον απαιτείται η *ελαχιστοποίηση* της συνάρτησης ελέγχου, θα χρειαστούμε *επιπλέον* την ικανότητα της παραγώγισης της συνάρτησης ελέγχου, ως προς τον κόμβο εξόδου o_i, και την ιδιότητα της *κυρτότητας*, ώστε αυτή να διαθέτει ολικό ελάχιστο. Επιπλέον, μία συνεχής παράγωγος θα βοηθήσει στην ομαλή λειτουργία της μάθησης.

<u>Παλινδρόμηση</u>

***Τετραγωνικό Άθροισμα Σφαλμάτων* (Squared Sum of Errors, SSE)**

Χρησιμοποιεί την L_2 νόρμα, η οποία ορίζεται ως $L_2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sum_{i=1}^{n}(x_i - y_i)^2$. Έχει τύπο και παράγωγο:

$$SSE_{\boldsymbol{x}_i} = \|\boldsymbol{y}_i - \boldsymbol{o}_i\|_2 = \sum_{j=1}^{p}(y_{ij} - o_{ij})^2, \qquad \frac{\partial}{\partial \boldsymbol{o}_i} SSE_{\boldsymbol{x}_i} = \boldsymbol{o}_i - \boldsymbol{y}_i$$

Θέλουμε όσο περισσότερους όρους προσθέτουμε στο μοντέλο μας, να πετυχαίνουμε μικρότερο σφάλμα. Όμως, το SSE όσο αυξάνουμε το πλήθος των παρατηρήσεων, θα συνεχίσει να αυξάνεται, το οποίο είναι μη – θεμιτό (Zaki & Meira, 2020).

Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error, MSE)

Το Μέσο Τετραγωνικό Σφάλμα (*Mean Square Error, MSE*), προσπαθεί να περιορίσει τις τιμές του SSE μέσω της διαίρεσής του με το πλήθος των όρων. Το MSE έχει τύπο και παράγωγο ίσα με:

$$MSE_{\boldsymbol{x}_i} = \frac{1}{p}\|\boldsymbol{y}_i - \boldsymbol{o}_i\|_2 = \frac{1}{p}\sum_{j=1}^{p}(y_{ij} - o_{ij})^2, \qquad \frac{\partial}{\partial \boldsymbol{o}_i}\,\varepsilon_{\boldsymbol{x}_i} = \frac{1}{p}(\boldsymbol{o}_i - \boldsymbol{y}_i)$$

Μέσο Απόλυτο Σφάλμα (Mean Absolute Error, MAE)

Χρησιμοποιεί την L_1 νόρμα αντί της L_2 του MSE. Αυτό, εάν και δημιουργεί ένα σημείο ασυνέχειας στο 0, κάνει την συνάρτηση απώλειας ανθεκτική σε μεγάλες ακραίες τιμές. Για την διαδικασία της παραγώγισης, αναγκαζόμαστε να καταφύγουμε σε έννοιες γενίκευσης όπως αυτές του *υπο – διαφορικού (subgradient)*. Η συνάρτηση Μέσου Απόλυτου Σφάλματος έχει τύπο

$$MAE_{\boldsymbol{x}_i} = \frac{1}{p}\|\boldsymbol{y}_i - \boldsymbol{o}_i\|_1 = \frac{1}{p}\sum_{j=1}^{p}|\,y_{ij} - o_{ij}|$$

Επιπλέον, να αναφέρουμε πως δεν συνηθίζεται η χρήση νορμών L_p, όπου για παράδειγμα $p \neq 0, 1$. Εάν θεωρήσουμε το $p = 3 + 2k$ έναν μονό αριθμό, θα έχουμε L_{3+2k}, για $k = 0,1,2, \dots, K$. Όμως τότε, αυτές δεν θα είναι πλέον κυρτές, με αποτέλεσμα να χάνουν την ιδιότητα του ολικού ελαχίστου. Επιπλέον, εάν θεωρήσουμε το $p = 4 + 2k$, οι χρήσεις των L_{4+2k}, για $k = 0,1,2, \dots, K$, είναι λίγες, καθώς η επίδραση των ακραίων τιμών θα είναι ακόμα μεγαλύτερη στις τιμές τους.

Ρίζα Μέσου Τετραγωνικού Σφάλματος (Root Mean Squared Error, RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_i - o_j)}$$

Τα αποτελέσματα που επιστρέφει ανήκουν στην ίδια κλίμακα όπως η πραγματική τιμή y_i και η προβλεπόμενη τιμή από τον νευρώνα εξόδου o_j, με αποτέλεσμα ευκολία στην κατανόηση. Επιπλέον, έχει συνεχή παράγωγο.

Κατηγοριοποίηση

Σταυρωτή Εντροπία (Cross Entropy)

Προέρχεται από μία εκτίμηση της μέγιστης εντροπίας. Μεταφράζεται ως η απόσταση των μεταξύ των εκτιμώμενων τιμών o_i και των πραγματικών τιμών y_i, στον χώρο των πιθανοτήτων, δηλαδή μία μεγαλύτερη τιμή (κοντά στο 1) συνεπάγεται πως είναι διαφορετικά τα o_i και y_i (Yang, 2019). Έχει τύπο

$$E_c = -\sum_{i}^{n} o_i \log(y_i) + (1 - o_i) \log(1 - y_i)$$

Συνάρτηση απώλειας των Kullback – Leibler (KL)

Η συνάρτηση απώλειας KL αποτελεί έναν συνδυασμό της εντροπίας και της σταυρωτής εντροπίας. Έχει τύπο:

$$E_{KL} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{o}_i \log(\boldsymbol{o}_i) - \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{o}_i \log(\boldsymbol{y_i})$$

Περισσότερες πληροφορίες σχετικά με την εξήγηση της προέλευσης και του τύπου της Kullback – Leibler, μπορεί να βρεθεί στο άρθρο του Shlens (2014).

Απώλεια Hinge (Hinge Loss)

Βρίσκει εφαρμογές στην κατηγοριοποίηση με SVM (Yang, 2019). Ορίζεται ως:

$$E_h = \sum_{i=1}^{N} \max\{0, s - \boldsymbol{o}_i \cdot \boldsymbol{y}_i\}, \text{ή } E_h = \sum_{i=1}^{N} \max\{0, s - \boldsymbol{o}_i \cdot \boldsymbol{y}_i\}$$

Ενώ έχει και μία μορφή ακόμα, την τετραγωνική Hinge συνάρτηση απώλειας,

$$E_h = \sum_{i=1}^{N} (\max\{0,1 - \boldsymbol{o}_i \cdot \boldsymbol{y}_i\})^2.$$

Δεν δίνει ποινή στα δείγματα τα οποία έχουν τιμή πάνω από μία συγκεκριμένη, $|f(x)| \geq s$, καθώς θεωρείται πως πλέον έχουν ολοκληρώσει την διαδικασία της μάθησης. Επιπλέον, απαιτεί το σφάλμα να είναι 0 όταν η κατηγοριοποίηση δεν θεωρείται πως ξεπερνά την συγκεκριμένη τιμή (Wang et al. 2020). Συνηθίζονται οι τιμές για το $s = 1$ ή $s = \frac{1}{2}$ (Yang, 2019). Περισσότερες πληροφορίες για τις συναρτήσεις απώλειας στην κλασική μηχανική μάθηση αλλά και στις διεργασίες της βαθιάς μάθησης, όπως η αναγνώριση αντικειμένων και η αναγνώριση προσώπου, μπορεί να βρεθεί στο άρθρο των Wang et al. (2020). Επιπρόσθετα, μία σύγκριση της σταυρωτής εντροπίας με το τετραγωνικό σφάλμα για τις διεργασίες κατηγοριοποίησης δίνεται στο άρθρο των Hui & Belkin (2020), ενώ οι Patel & Sastry (2021) αναφέρουν πως για νευρωνικά δίκτυα με πάρα πολλές παραμέτρους (overparameterization) βρέθηκαν αυξημένες επιδόσεις με συμμετρικές συναρτήσεις κόστους, έναντι της σταυρωτής εντροπίας και του τετραγωνικού σφάλματος.

Συναρτήσεις Απώλειας				
Συνάρτηση Απώλειας	**Τύπος**	**Διεργασίες**	**Πλεονεκτήματα**	**Μειονεκτήματα**
SSE	$\sum_{j=1}^{p} (y_{ij} - o_{ij})^2$	Παλινδρόμηση	Κυρτότητα Συνεχής Παράγωγος	Ακραίες τιμές

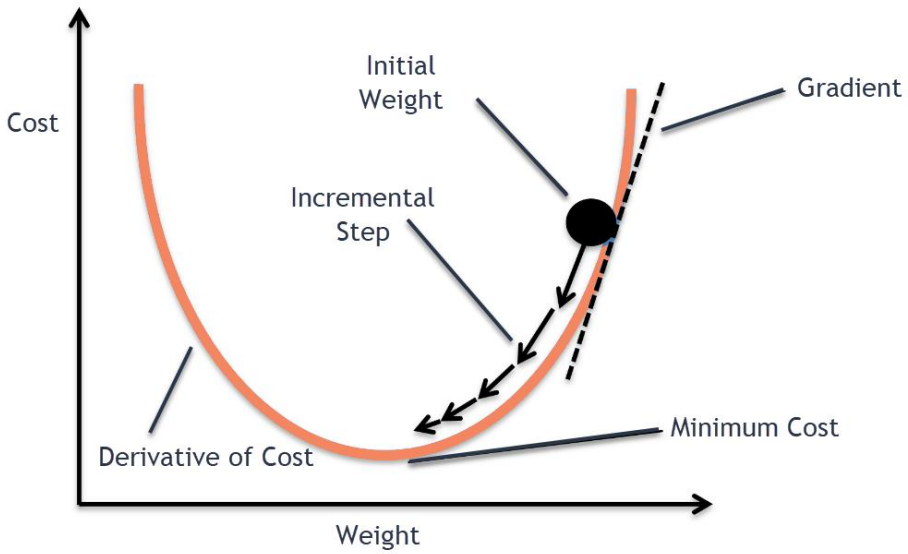
MSE	$\frac{1}{\rho}\sum_{j=1}^{p}(y_{ij}-o_{ij})^2$	Παλινδρόμηση	Κυρτότητα Συνεχής Παράγωγος	Ακραίες τιμές
MAE	$\frac{1}{\rho}\sum_{j=1}^{p}\lvert y_{ij}-o_{ij}\rvert$	Παλινδρόμηση	Ανθεκτική στις Ακραίες τιμές Κατανόηση	Συνέχεια Παραγώγου
RMSE	$\sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_i-o_j)}$	Παλινδρόμηση	Συνέχεια Παραγώγου Κατανόηση	Ακραίες Τιμές
Σταυρωτή Εντροπία	$-\sum_{i}^{n}(o_i\log(y_i)+(1-o_i)\log(1-y_i))$	Κατηγοριοποίηση	Συνέχεια παραγώγου Ερμηνεία	-
KL	$\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{o}_i\log(\boldsymbol{o}_i)-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{o}_i\log(y_i)$	Κατηγοριοποίηση	-	-
Hinge	$\sum_{i=1}^{N}\max\{0,1-\boldsymbol{o}_i\cdot\boldsymbol{y}_i\},$	Κατηγοριοποίηση	Διευκόλυνση στα SVM	Μη συνεχής παράγωγος
Τετραγωνική Hinge	$\sum_{i=1}^{N}(\max\{0,1-\boldsymbol{o}_i\cdot\boldsymbol{y}_i\})^2$	Κατηγοριοποίηση	Διευκόλυνση στα SVM Συνεχής παράγωγος	-

Πίνακας 2. Συναρτήσεις απώλειας

1.3.4. Αλγόριθμοι Βελτιστοποίησης

Οι αλγόριθμοι βελτιστοποίησης αξιοποιούνται στην μάθηση των νευρωνικών δικτύων, με σκοπό την ανανέωση των βαρών στις συνδέσεις μεταξύ των νευρώνων για την ελαχιστοποίηση της συνάρτησης απώλειας. Ιδανικά, θέλουμε μία *μεγάλη* τιμή στην συνάρτηση κόστους να αντιστοιχεί σε μία *μεγάλη* αλλαγή στα βάρη του νευρωνικού δικτύου, ενώ μία μικρή τιμή στην συνάρτηση κόστους να αντιστοιχεί σε μικρή αλλαγή στα βάρη του νευρωνικού δικτύου (Zaki & Meira, 2020). Με κάθε βήμα στον αναδρομικό τύπο των νέων βαρών, θέλουμε να προσεγγίζουμε την (ολική) ελάχιστη τιμή της συνάρτησης απώλειας (Haji & Abdulazeez, 2021). Σύμφωνα με την Yang, (2019), στους αλγόριθμους των νευρωνικών δικτύων χρησιμοποιούνται βελτιστοποιητές που χωρίζονται σε δύο κατηγορίες: αυτούς που αξιοποιούν την βελτιστοποίηση με κλίση, όπως η κλίση κατάβασης (gradient descent), οι οποίες μέσω της κλίσης βρίσκουν την «κατεύθυνση» κατά την οποία ελαχιστοποιείται η συνάρτηση. Μία ακόμη μέθοδος όμως, αναφέρεται ως αυτή «χωρίς κλίση» (gradient free), η οποία περιέχει για παράδειγμα εξελικτικούς αλγόριθμους βελτιστοποίησης.

Οι επιθυμητές ιδιότητες των αλγόριθμων βελτιστοποίησης που βασίζονται στην κλίση, σύμφωνα με τους Haji & Abdulazeez (2021), είναι (1) ο ρυθμός σύγκλισης, (2) η ταχύτητα που επιτυγχάνουν στην εκπαίδευση και (3) η γενικότερη απόδοση. Επίσης, παρατηρείται μία αύξηση στα νευρωνικά δίκτυα που πραγματοποιούν βελτιστοποίηση μέσω εξελικτικών αλγόριθμων, τα οποία βρίσκουν εφαρμογές στα μεγάλα δεδομένα (Cao et al. 2016), (Yoo & Chung, 2020), (Chen, 2017), (Essiet et al. 2019). Σε όσα ακολουθούν, θα αναφέρουμε σχετικά και κάποιους αλγόριθμους βελτιστοποίησης μέσω κλίσης σύμφωνα με τους Haji & Abdulazeez (2021).



Σχήμα 12. Διαδικασία βελτιστοποίησης μέσω κλίσης κατάβασης, πάνω στην κυρτή καμπύλη της συνάρτησης κόστους. Η σφαίρα που «κυλάει» στην «πλαγιά» προσομοιάζει τα βήματα που ακολουθεί ο αλγόριθμος της κλίσης κατάβασης μέχρι να «φτάσει» στην ελάχιστη τιμή της πλαγιάς, δηλαδή την ελάχιστη τιμή της κυρτής συνάρτησης κόστους. *Από Lanham (2018).*

Βελτιστοποίηση με κλίση (gradient)

Κλίση Κατάβασης (gradient descent)

Πραγματοποιεί ανανέωση των βαρών μετά από κάθε ένα δεδομένο εισόδου που εισάγεται στο νευρωνικό δίκτυο. Όταν τελειώσουν τα δεδομένα εκπαίδευσης, λέμε πως έχει περάσει μία *εποχή* (epoch). Ο αναδρομικός τύπος στο $t-$ οστό βήμα, ισούται με:

$$x^{t+1} = x^t - \frac{\eta}{m} \sum_{i=1}^{m} \nabla f_i\,(x),$$

όπου το $\eta \in (0,1]$ είναι το βήμα ή ο ρυθμός μάθησης και x η πληροφορία που εισέρχεται στην f_i. Η κλίση (ή ανάδελτα) της f_i, $\nabla f_i(x)$ αντιστοιχεί στην κατεύθυνση προς αυτήν που μεγαλώνει η ποσότητα $\nabla f_i(x)$. Συνεπώς, ο όρος $-\frac{\eta}{m} \sum_{i=1}^{m} \nabla f_i$, αντιστοιχεί σε μία κατεύθυνση *αντίθετη* προς αυτήν που μεγαλώνει η ποσότητα $\nabla f_i(x)$ (Zaki & Meira, 2020). Η ποιότητα της σύγκλισης είναι καλή, η ταχύτητα εκπαίδευσης είναι μέτρια για απλά μοντέλα αλλά μικρή για σύνθετα μοντέλα, ενώ έχει μεγάλο υπολογιστικό κόστος (Haji & Abdulazeez 2021).

Κλίση κατάβασης με Παρτίδα (Batch Gradient Descent, BGD)

Εάν επιθυμούμε να αυξήσουμε την ταχύτητα της σύγκλισης, το οποίο είναι εξαιρετικά σημαντικό όταν εργαζόμαστε με χιλιάδες δεδομένα και σκοπεύουμε να εκπαιδεύσουμε το νευρωνικό δίκτυο για πολλές εποχές, τότε μπορούμε να χρησιμοποιήσουμε την BGD. Αυτή, είναι μία παραλλαγή της κλίσης με κατάβαση. Υπολογίζει *ταυτόχρονα* το σφάλμα για μία παρτίδα δεδομένων, η οποία συνήθως έχει μέγεθος μία δύναμη του 2, ώστε να εκμεταλλευτούμε τους παράλληλους υπολογισμούς στους επεξεργαστές GPU (Vanhoucke et al. 2011). Τα βάρη ανανεώνονται μετά από κάθε παρτίδα, με βάση τον μέσο όρο του σφάλματος σε κάθε σημείο της παρτίδας. Δηλαδή, κατά τον υπολογισμό του σφάλματος του κάθε σημείου της παρτίδας τα βάρη είναι τα *ίδια*, το οποίο μπορεί να οδηγήσει σε ανεπιθύμητα αποτελέσματα. Για παράδειγμα η σταθερότητα των βαρών μπορεί να μην είναι η βέλτιστη. Επιπλέον, αυτή η ιδιαιτερότητα απαιτεί την χρήση προσωρινής μνήμης του υπολογιστή για την αποθήκευση της παρτίδας (Haji & Abdulazeez 2021). Έχει αναδρομικό τύπο:

$$x^{t+1} = x^t - \eta \sum_{i=1}^{m} \nabla f_i\,(x).$$

Στοχαστική Κλίση Κατάβασης (stochastic gradient descent, SGD)

Όταν βελτιστοποιούμε σύνθετες συναρτήσεις οι οποίες δεν είναι κυρτές, συχνό πρόβλημα είναι η προσέγγιση *τοπικών* ελαχίστων από τον αλγόριθμο, με αδυναμία εύρεσης του ολικού ελαχίστου. Η στοχαστική κλίση κατάβασης, την οποία θα αναφέρουμε ως SGD, σε κάποιες περιπτώσεις μπορεί να ξεπεράσει αυτό το πρόβλημα ενώ ταυτόχρονα μειώνει το υπολογιστικό κόστος. Συγκεκριμένα, οι Kleinberg et al. (2018) αναφέρουν πως η SGD είναι ικανή να αποφύγει τοπικά ελάχιστα με «μικρή» διάμετρο, χάρη στην στοχαστικότητα που εμπεριέχει.

Επεκτείνοντας στην διαδικασία, η SGD αντί για τον υπολογισμό όλων των τιμών στο άθροισμα $\sum_{i=1}^{m} \nabla f_i$ των κλίσεων, περιορίζεται στην τυχαία (εξού και «στοχαστική») επιλογή *μόνο μίας τιμής*, για παράδειγμα την ∇f_i, για ένα τυχαίο i. Δηλαδή, χρησιμοποιεί μόνο μία τυχαία κατεύθυνση στην οποία ελαχιστοποιείται η συνάρτηση και όχι όλες τους, το οποίο μπορεί να βοηθήσει τον αλγόριθμο να ξεπεράσει τοπικά ελάχιστα και να κατευθυνθεί προς το ολικό ελάχιστο. Ο αναδρομικός τύπος (Yang, 2019), ισούται με:

$$x^{t+1} = x^t - \eta_t \nabla f(x^t),$$

όπου ο ρυθμός αύξησης συνηθίζεται να ισούται με $\eta_t = \frac{1}{1+\beta t}, t \in \mathbb{N}$. Δηλαδή, πλέον το βήμα δεν είναι μία σταθερά αλλά μία μεταβλητή. Εφόσον συνηθίζεται $\beta > 0$, όπου β είναι μία υπερπαράμετρος, αυτή η μεταβλητή θα έχει φθίνουσες τιμές. Επιπλέον, έχει αποδειχθεί η βεβαιότητα της σύγκλισης εάν έχουμε $\sum_t \eta_t = \infty$ & $\sum_t \eta_t^2 < \infty$ (Yang, 2019). Η SGD έχει καλή ποιότητα σύγκλισης, με ταχύτητα παρόμοια με την GD. Επίσης, αναφέρεται πως είναι δύσκολη η επιλογή του κατάλληλου ρυθμού μάθησης καθώς αλλάζουν οι διαστάσεις (Haji & Abdulazeez 2021). Επιπλέον, έχει μεγάλη μεταβλητότητα, με αποτέλεσμα την δυσκολία της σύγκλισης (Yang, 2019).

Αναφέρεται πως και οι τρεις παραλλαγές της κλίσης κατάβασης (GD), η κλίση κατάβασης με παρτίδα (BGD) και η στοχαστική (SGD), αντιμετωπίζουν γενικά τις ίδιες δυσκολίες: (1) Την επιλογή του ρυθμού μάθησης, (2) την επιλογή μεταβλητότητας του ρυθμού μάθησης, (3) την μεταβλητότητα του ρυθμού μάθησης μπορεί να σχετίζεται με τις μεταβλητές του προβλήματος ενώ (4) σε μη – κυρτές συναρτήσεις απώλειας μπορεί να «κολλήσουν» μόνιμα σε ένα τοπικό ελάχιστο, θεωρώντας πως αυτό είναι το ολικό ελάχιστο (Haji & Abdulazeez 2021). Σε όσα ακολουθούν, αναφέρουμε κάποιες διαδικασίες που έχουν βελτιώσει την βελτιστοποίηση των τεχνικών με κλίση.

Κίνησης (Momentum)

Αποτελεί μία επέκταση της στοχαστικής κλίσης κατάβασης, SGD. Ο αναδρομικός τύπος χωρίζεται σε δύο μέρη, όπου το πρώτο χρησιμοποιείται για τον υπολογισμό του δεύτερου (Yang, 2019). Αξιοποιείται στην μάθηση των νευρωνικών δικτύων καθώς μπορεί να «παραβλέψει» τα τοπικά ελάχιστα που έχουν οι σύνθετες συναρτήσεις τους. Αναφέρεται πως έχει καλή ποιότητα σύγκλισης, γρήγορη για απλά μοντέλα και μέτρια για σύνθετα μοντέλα, γρηγορότερη από την GD. Είναι καλύτερο για μοντέλα με λιγότερες παραμέτρους (Haji & Abdulazeez 2021).

Πρώτο βήμα: $u^{t+1} = \gamma u^t - \eta_t \nabla f(u^t), \quad 0 < \gamma < 1.$

Δεύτερο βήμα: $x^{t+1} = x^t - u^{t+1}.$

Επιταχυνόμενη κλίση του Nesterov (Nesterov accelerated gradient, NAG)

Αποτελεί μία βελτίωση της *Κίνησης*. Και αυτός ο αναδρομικός τύπος χωρίζει τους υπολογισμούς σε δύο μέρη, το οποίο άμεσα αυξάνει το κόστος των υπολογισμών. Στο πρώτο βήμα χρησιμοποιεί μία πρόβλεψη για την συνεισφορά της τυχαίας κατεύθυνσης μέσω της ποσότητας $\left(x^k - \gamma u^k\right)$. Αναφέρεται πως έχει καλή ποιότητα σύγκλισης, απαιτεί λιγότερη μνήμη από την SG, ενώ δεν είναι τόσος καλό για ένα σύνθετο μοντέλο. Παρουσιάζεται δυσκολία στην επιλογή ρυθμού μάθησης ενώ επιπρόσθετα «κολλάει» στα τοπικά ελάχιστα (Haji & Abdulazeez, 2021).

Πρώτο βήμα: $u^{t+1} = \gamma u^t - \eta_t \nabla f_i\left(x^k - \gamma u^k\right), \quad 0 < \gamma < 1.$

Δεύτερο βήμα: $x^{t+1} = x^t - u^{t+1}.$

Προσαρμοστική Εκτίμηση Κίνησης (Adaptive Moment Estimation, Adam)

Θα αναφερόμαστε σε αυτόν ως Adam. Αποτελεί μία βελτίωση των παραπάνω τεχνικών, ως μία μίξη των τεχνικών *RMSProp* και *Κίνησης*. Είναι εξαιρετικά συχνή επιλογή βελτιστοποίησης στην διαδικασία της *Οπισθοδρομικής Διάδοσης* (Backpropagation) των νευρωνικών δικτύων (Haji & Abdulazeez 2021), η οποία χρησιμοποιεί προσαρμοστική μάθηση (adaptive learning), δηλαδή δεν έχει υπερπαραμέτρους. Ο κύριος στόχος του αναδρομικού αλγόριθμου Adam, είναι στα βήματα να αξιοποιηθούν η μέση τιμή και η αποκεντροποιημένη διασπορά, οι οποίες υπολογίζονται από κάθε προηγούμενη επανάληψη (Yang, 2019). Τα θετικά χαρακτηριστικά περιλαμβάνουν μειωμένα κόστη, λιγότερη απαίτηση από μνήμη, γρήγορη σύγκλιση, ενώ κρίνεται χρήσιμος για χαρακτηριστικά τα οποία είναι αραιά, σε ένα σύνθετο μοντέλο. Τα αρνητικά του περιλαμβάνουν την μη – δυνατότητα σύγκλισης σε δεδομένα με πολλές παρατηρήσεις και υψηλή διάσταση (Haji & Abdulazeez 2021).

$$x^{k+1} = x^k - \frac{\eta\, m_1^k}{\sqrt{m_2^k + \varepsilon}}, \qquad m_1^k = \frac{n_1^k}{1-\alpha^k}, \quad m_2^k = \frac{n_2^k}{1-\beta^k},$$

όπου
$$\begin{cases} n_1^k = a n_1^{k-1} + (1-a)\nabla f(x^k) \\ n_2^k = a n_2^{k-1} + (1-a)\nabla f(x^k) \end{cases}$$

Οι Haji & Abdulazeez (2021) αναφέρουν πως στην συγκριτική τους έρευνα, ο καλύτερος αναδρομικός αλγόριθμος ήταν μία παραλλαγή του Adam, αποκαλούμενος NAdam, ο οποίος βρέθηκε πολύ χρήσιμος για δεδομένα με μεγάλο όγκο και διάσταση, με σταθερότητα στον ρυθμό σύγκλισης, ταχύτητα εκπαίδευσης και απόδοση καλύτερη από αυτή των υπόλοιπων αλγόριθμων, όπως η SGD, Momentum, NAG, Adam, BGD και άλλες.

Αλγόριθμοι Βελτιστοποίησης			
Αλγόριθμος Βελτιστοποίησης	**Αναδρομικός Τύπος**	**Ταχύτητα Εκπαίδευσης για Απλό / Σύνθετο Σύνολο**	**Ικανότητα Σύγκλισης καθώς $n \to \infty$**
GD	$x^{t+1} = x^t - \frac{\eta}{m}\sum_{i=1}^{m}\nabla f_i(x)$	Μέτρια / Αργή	Με σωστή επιλογή ρυθμού μάθησης έχει καλές επιδώσεις.
BGD	$x^{t+1} = x^t - \eta\sum_{i=1}^{m}\nabla f_i(x)$	Μέτρια / Αργή	Με σωστή επιλογή ρυθμού μάθησης έχει καλές επιδώσεις.
SGD	$x^{t+1} = x^t - \eta_t\nabla f(x^t), \qquad \eta_t = \frac{1}{1+\beta t}$	Μέτρια / Αργή	Με σωστή επιλογή ρυθμού μάθησης έχει καλές επιδώσεις. Αποφυγή «μικρών» τοπικών ελαχίστων.
Momentum	$\begin{cases} u^{t+1} = \gamma u^t - \eta_t\nabla f(u^t) \\ x^{t+1} = x^t - u^{t+1} \end{cases}$	Γρήγορη / Μέτρια	Προτιμάται για ένα απλό μοντέλο.
NAG	$\begin{cases} u^{t+1} = \gamma u^t - \eta_t\nabla f_i(x^k - \gamma u^k) \\ x^{t+1} = x^t - u^{t+1} \end{cases}$	Γρήγορη / Μέτρια	Σε δεδομένα με πολλά χαρακτηριστικά υπάρχει αργή και δύσκολη σύγκλιση.
Adam	$x^{k+1} = x^k - \frac{\eta\, m_1^k}{\sqrt{m_2^k+\varepsilon}}$, $m_1^k = \frac{n_1^k}{1-\alpha^k}$, $m_2^k = \frac{n_2^k}{1-\beta^k}$, $n_1^k = a n_1^{k-1} + (1-a)\nabla f(x^k)$, $n_2^k = a n_2^{k-1} + (1-a)\nabla f(x^k)$	Γρήγορη / Γρήγορη	Καλή σύγκλιση σε δεδομένα με μεγάλη διάσταση και πολλά χαρακτηριστικά.

Πίνακας 3. Αλγόριθμοι Βελτιστοποίησης

1.3.5 Συνάρτηση Κόστους

Η συνάρτηση κόστους αφορά την επικύρωση των αποτελεσμάτων από τις διάφορες διεργασίες που θα διατελέσει το νευρωνικό δίκτυο. Στα παρόντα, θα αναφέρουμε κάποιες μετρικές που αφορούν τις διεργασίες της παλινδρόμησης και της κατηγοριοποίησης.

Διεργασίες Παλινδρόμησης

Στην ενότητα 2.1 αναφέραμε το στατιστικό $R^2 = \frac{RSS}{TSS}$, ενώ στην ενότητα 3.3 των συναρτήσεων απώλειας, αναφέραμε τις ακόλουθες συναρτήσεις, οι οποίες μπορούν να θεωρηθούν και ως συναρτήσεις κόστους: (1) Μέσο Σφάλμα, (2) Μέσο Τετραγωνικό Σφάλμα, (3), Μέσο Απόλυτο Σφάλμα. (4) Ρίζα Μέσου Τετραγωνικού Σφάλματος

Περισσότερες μετρικές παλινδρόμησης συγκρίνονται στο άρθρο του Botchkarev (2019).

Διεργασίες Κατηγοριοποίησης

Υπενθυμίζουμε πως, στην ενότητα 2.1, αναφέραμε τις περιπτώσεις των μετρικών όπως η ακρίβεια (ως προς μία κλάση) η ακρίβεια ως προς όλες τις κλάσεις, καθώς και την κάλυψη ή ανάκληση, τον *ρυθμός σφάλματος* (error rate) που ορίζεται ως *1 – Ακρίβεια,* μέσω των οποίων ορίσαμε την F – μετρική (F – measure), η οποία είχε τον τύπο:

$$F_i = \frac{2}{\frac{1}{prec_i} + \frac{1}{recall_i}} = \frac{2n_{ii}}{n_i + m_i}$$

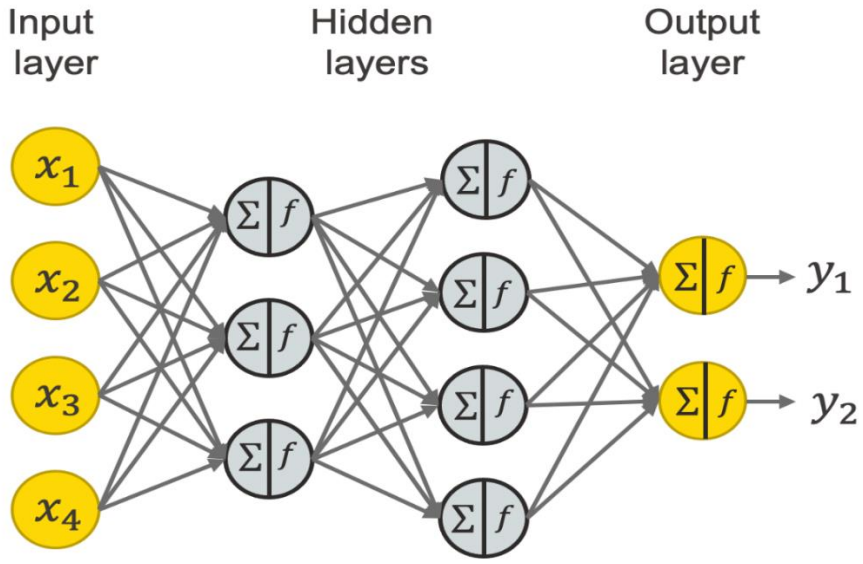
Όπου με $prec_i$ και $recall_i$ συμβολίσαμε την ακρίβεια και την αντίστοιχη ανάκληση για τα $i-$ οστά δεδομένα που μας δίνονται. Επιπλέον, αναφέραμε στην ενότητα 3.3 τα σφάλματα της Σταυρωτής Εντροπίας, KL, Hinge και τετραγωνικό Hinge. Περισσότερες μετρικές κατηγοριοποίησης συγκρίνονται στο άρθρο των Hossin & Sulaiman (2015).

Κεφάλαιο 2.
Δομές Νευρωνικών Δικτύων

Ενότητα 1.
Νευρωνικά Δίκτυα Πολυστρωματικών Αντιλήπτρων

2.1.1 Αρχιτεκτονική

Σε αυτή την ενότητα, θα χρησιμοποιήσουμε τους συμβολισμούς από το βιβλίο των Zaki & Meira (2020). Στα νευρωνικά δίκτυα *πολυστρωματικών αντιλήπτρων* (*Multi Layered Perceptrons, MLP*), θεωρούμε πλέον πως έχουμε την στοιβάδα εισόδου με n – το πλήθος κόμβους v_i, πως στην κρυφή στοιβάδα έχουμε τους m – το πλήθος κόμβους z_k, ενώ μπορεί να έχουμε πολλές κρυφές στοιβάδες με διαφορετικό πλήθος κόμβων. Τελικά, στην στοιβάδα εξόδου έχουμε p – κόμβους $\boldsymbol{o_j}$. Επιπλέον, όλοι οι κόμβοι v_i της στοιβάδας εισόδου συνδέονται με ένα κατευθυνόμενο τόξο (v_i, z_k) προς κάθε έναν από τους κόμβους w_j της κρυφής στοιβάδας, και όλοι οι κόμβοι της κρυφής στοιβάδας z_k συνδέονται με κάθε έναν από τους κόμβους εξόδου o_j. Επειδή οι κόμβοι της κάθε στοιβάδας συνδέονται με όλους τους κόμβους της επόμενης στοιβάδας, λέμε πως οι στοιβάδες είναι *πλήρως συνδεδεμένες* (fully connected). Επιπρόσθετα, να τονίσουμε ότι είναι αναγκαίο οι κόμβοι της κάθε στοιβάδας να μην συνδέονται μεταξύ τους, ενώ τα κατευθυνόμενα τόξα έχουν φορά από την κάθε στοιβάδα αποκλειστικά προς την επόμενη. Με αυτούς τους περιορισμούς, η πληροφορία *τροφοδοτείται προς τα μπροστά* (*feed – forward*), το οποίο έχει αποτέλεσμα την αντίστοιχη ονομασία της κατηγορίας όπου ανήκουν τα Πολυστρωματικά Αντίληπτρα, η οποία αποκαλείται *Νευρωνικά Δίκτυα Εμπρόσθιας Τροφοδότησης* (Feed – Forward Neural Networks). Ένα νευρωνικό δίκτυο πολυστρωματικών αντιλήπτρων απεικονίζεται στο *Σχήμα 13*.



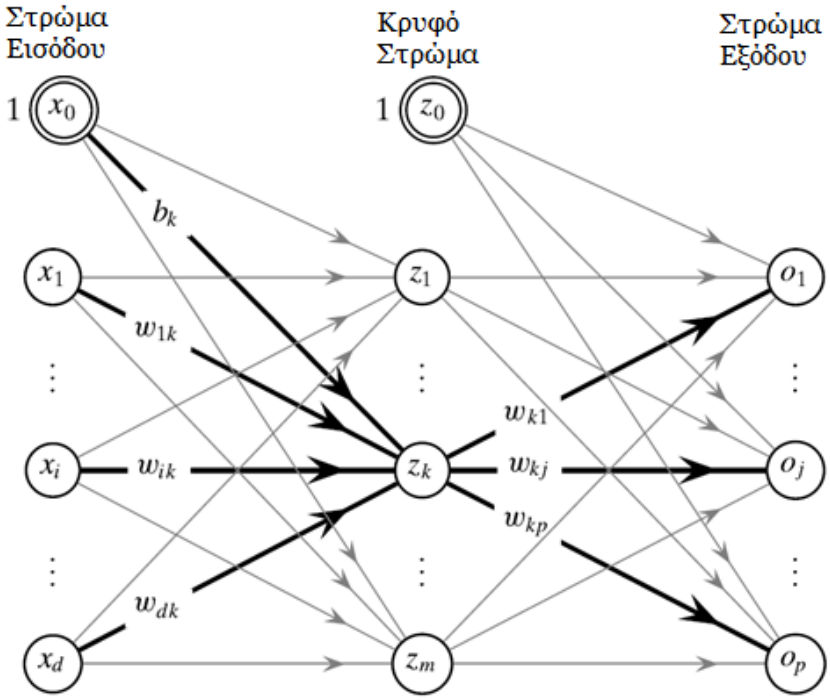
Σχήμα 13. Νευρωνικό δίκτυο πολυστρωματικών αντιλήπτρων με δύο κρυφές στοιβάδες, στις οποίες μπορούμε να δούμε αναλυτικά την διαδικασία που συμβαίνει στον κάθε κόμβο, όπου συμβαίνουν δύο μετασχηματισμοί, πρώτα ο γραμμικός μέσω της καθαρής εισροής, $\sum \boldsymbol{w_i x_i}$, και έπειτα ο (μη) γραμμικός μετασχηματισμός μέσω της συναρτήσεως ενεργοποίησης, f. Η στοιβάδα εισόδου που παρουσιάζεται εδώ,

έχει 4 κόμβους, x_1, x_2, x_3, x_4, ενώ η πρώτη κρυφή στοιβάδα έχει 3 κόμβους και η δεύτερη κρυφή στοιβάδα έχει 4 κρυφούς κόμβους. Η στοιβάδα εξόδου έχει δύο κόμβους που εξάγουν τις τιμές y_1, y_2.

Τονίζουμε πως όσα είχαμε αναφέρει στο *Κεφάλαιο 1, Ενότητα 3 Θεμελιώδεις Έννοιες Τεχνητών Νευρωνικών Δικτύων*, αρχικά αφορούσαν ένα νευρωνικό δίκτυο πολυστρωματικών αντιλήπτρων με μία στοιβάδα εισόδου με κόμβους x_i, μία κρυφή στοιβάδα με έναν κρυφό νευρώνα z_k και μία στοιβάδα εξόδου με έναν νευρώνα o_j, ενώ αργότερα στο *Σχήμα 10*, παρουσιάσαμε ένα νευρωνικό δίκτυο πολυεπίπεδων αντιλήπτρων με μία στοιβάδα εισόδου και $x_1, \dots, x_p$ νευρώνες εισόδου, μία κρυφή στοιβάδα με τρεις νευρώνες z_1, z_2, z_3 και μία στοιβάδα εξόδου με έναν νευρώνα o_1. Στην επόμενη ενότητα, θα αναφέρουμε την διαδικασία της εκπαίδευσης σε ένα πολυστρωματικό νευρωνικό δίκτυο με μία κρυφή στοιβάδα.

2.1.2 Εκπαίδευση με μία κρυφή στοιβάδα

Για την εκπαίδευση των νευρωνικών δικτύων, αρχικά θα χρειαστεί να επιλέξει ο ερευνητής τα δεδομένα εκπαίδευσης από το σύνολο των δεδομένων. Έπειτα, θα πρέπει να επιλέξει τις υπερπαραμέτρους του μοντέλου. Δηλαδή, να διαλέξει τον αριθμό των κρυφών στοιβάδων, το πλήθους των νευρώνων στην τελευταία στοιβάδα ανάλογα την διεργασία, τον αριθμό των κρυφών νευρώνων. Συγκεκριμένα, στην περίπτωση που θα αναφέρουμε, θα έχει $n-$ το πλήθος νευρώνες εισόδου $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \dots, x_{id})$, μία κρυφή στοιβάδα με $m-$ το πλήθος κρυφούς νευρώνες z_k, και στην στοιβάδα εξόδου θα έχουμε $p-$ το πλήθος νευρώνες εξόδου, όπως φαίνεται στο Σχήμα 14.



Σχήμα 14. Νευρωνικό δίκτυο πολυστρωματικών αντιλήπτρων (MLP) με μία κρυφή στοιβάδα. Ο κόμβος πόλωσης που θα δράσει στο κρυφό στρώμα είναι ο x_0, ενώ ο κόμβος πόλωσης που θα δράσει στο στρώμα εξόδου είναι ο z_0. *Από Zaki & Meira, 2020.*

Έπειτα θα πρέπει να επιλέξει τις συναρτήσεις ενεργοποίησης που θα χρησιμοποιήσει σε κάθε κρυφή στοιβάδα αλλά και στην στοιβάδα εξόδου. Επιπλέον θα πρέπει να επιλέξει την συνάρτηση απώλειας με έναν αντίστοιχο αλγόριθμο βελτιστοποίησης, με σκοπό την ελαχιστοποίηση του σφάλματος της μάθησης

που προκύπτει από την συνάρτηση απώλειας, ενώ τελικά απαιτείται και μία μετρική ώστε να μπορεί να παρακολουθήσει την επίδοση του νευρωνικού δικτύου στα δεδομένα εκπαίδευσης. Τότε, μπορεί να ξεκινήσει η εκπαίδευση και αυτή θα χωριστεί σε δύο βήματα:

- Πρώτα, εφαρμόζεται ο αλγόριθμος της *εμπρόσθιας διάδοσης* (forward propagation) όπου για μία επιλογή από τα βάρη μεταξύ των νευρώνων, η πληροφορία θα ξεκινήσει από τους νευρώνες εισόδου θα διαδοθεί μέχρι την στοιβάδα εξόδου, όπου θα λάβουμε τις τιμές των κόμβων εξόδου.
- Έπειτα, με βάση τις τιμές στους κόμβους εξόδου, θα ξεκινήσει η διαδικασία της *οπίσθιας διάδοσης* (backpropagation), όπου θα μετρηθεί το σφάλμα της εκπαίδευσης από την συνάρτηση απώλειας. Έπειτα, ο αλγόριθμος βελτιστοποίησης θα προτείνει νέες τιμές στα βάρη, ώστε να επιτευχθεί η ελαχιστοποίηση της συνάρτησης απώλειας. Ο σκοπός είναι να βρεθούν τα βάρη έτσι ώστε να ελαχιστοποιηθεί το σφάλμα της εκπαίδευσης.
- Όταν αυτό έχει ολοκληρωθεί για όλα τα δεδομένα εκπαίδευσης, θα έχει τελειώσει μία *εποχή* της εκπαίδευσης και έχει νόημα η αξιολόγηση του μοντέλου με βάση την μετρική.

Εμπρόσθια Διάδοση (forward propagation)

Αναφερόμαστε στο σύνολο των δεδομένων εκπαίδευσης, όπου $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \boldsymbol{D} \subseteq \mathbb{R}^{d\times p}$, με $i = 1,2,\dots,n$ το πλήθος στοιχεία. Δηλαδή, για $i = 1$, το πρώτο στοιχείο των δεδομένων εκπαίδευσης είναι το ζευγάρι $(\boldsymbol{x}_1, \boldsymbol{y}_1) \in \mathbb{R}^{d\times p}$. Αντιστοιχούμε κάθε μία από τις τιμές $x_{i1}, x_{i2}, \dots, x_{id}$, του $\boldsymbol{x}_i$, σε κάθε έναν από τους κόμβους εισόδου $v_1, \dots, v_d$. Θεωρούμε μία αρχική τιμή στα βάρη που συνδέουν τους κόμβους του νευρωνικού δικτύου, η οποία εξαρτάται από το αντικείμενο της έρευνας του ερευνητή. Συχνά, συνηθίζεται μία τυχαία επιλογή τιμών αλλά πρέπει να γίνει με προσοχή, ανάλογα τις συναρτήσεις ενεργοποίησης της κάθε στοιβάδας που θα επιλέξει να χρησιμοποιήσει ο κάθε χρήστης, καθώς έτσι μπορεί να προκύψουν προβλήματα όπως οι εξαφανιζόμενες κλίσεις στην ReLU (Feng & Lu, 2019). Μία άλλη μέθοδος αποτελεί η αρχικοποίηση μέσω «κανονικού ήχου» (*Gaussian Noise*) με μέση τιμή ίση με 0 και τυπική απόκλιση ίση με 0.01, και μέτρο πόλωσης ίσο με 1 για κάποια στρώματα. Περισσότερα αναφέρονται στο άρθρο των Narkhede et al (2022).

Υπολογισμός τιμών κρυφής στοιβάδας

Συμβολίζουμε τις τιμές που συνδέουν τον $i-$ οστό κόμβο εισόδου με τον $k-$ οστό κρυφό κόμβο, με w_{ik}, ενώ την πόλωση που θα δράσει στην κρυφή στοιβάδα θα την συμβολίσουμε με b_k και τον αντίστοιχο κόμβο πόλωσης (που συνηθίζεται να έχει τιμή 1) με $x_0 = 1$. Οπότε, με βάση όσα αναφέραμε στο προηγούμενο κεφάλαιο, η τιμή του $k-$ οστού κρυφού κόμβου θα είναι, για την συνάρτηση ενεργοποίησης f:

$$z_k = f(net_k) = f(b_k x_0 + w_{1k}x_1 + \cdots + w_{dk}x_d) = f\left(b_k + \sum_{i=1}^{d} w_{ik}\cdot x_i\right)$$

Συνολικά, καταφέραμε να υπολογίσουμε το διάνυσμα των τιμών της κρυφής στοιβάδας,

$$\boldsymbol{z} = (z_1, z_2, \dots, z_m)$$

Υπολογισμός τιμών στοιβάδας εξόδου

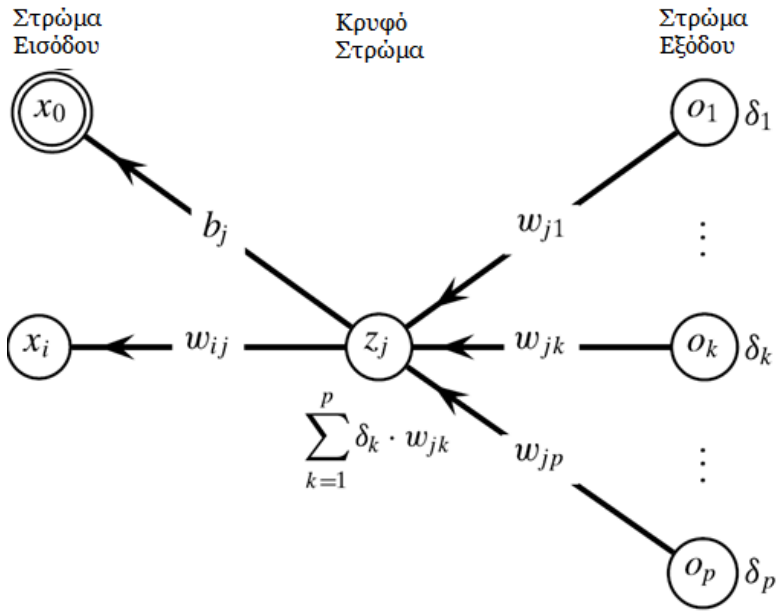
Έστω g η συνάρτηση ελέγχου της στοιβάδας εξόδου. Επιπλέον, έστω τα βάρη w_{kj} μεταξύ του $k-$ οστού κρυφού νευρώνα z_k με τον $j-$ οστό νευρώνα εξόδου. Θεωρούμε επίσης τον νευρώνα πόλωσης z_0, με πόλωση b_j ως προς τον $j-$ οστό νευρώνα εξόδου. Τότε, η τιμή στον $j-$ οστό νευρώνα εξόδου, θα είναι ίση με:

$$o_j = g\left(b_j + \sum_{k=1}^{m} w_{kj}\cdot z_k\right) = g\left(b_j + \sum_{k=1}^{m} w_{kj}\cdot\left[f\left(b_k + \sum_{i=1}^{d} w_{ik}\cdot x_i\right)\right]\right)$$

Συνολικά, καταφέραμε να υπολογίσουμε το διάνυσμα των τιμών της στοιβάδας εξόδου, $\boldsymbol{o} = (o_1, o_2, \dots, o_p)$.

Οπίσθια διάδοση (backpropagation)

Δοθέντος του $\boldsymbol{o}$, μπορεί να ξεκινήσει η διαδικασία της *οπίσθιας διάδοσης (back propagation)*, με τον οποίο θα «ανανεώσουμε» τις τιμές από τα βάρη μεταξύ των στοιβάδων. Ο υπολογισμός γίνεται αρχικά για τα βάρη που συνδέουν την στοιβάδα εξόδου με την κρυφή w_{jk}, συμπεριλαμβανομένου του βάρους του κρυφού κόμβου πόλωσης, την πόλωση b_k. Τότε, και μόνο τότε, έχει νόημα να ανανεώσουμε τα βάρη από το κρυφό στρώμα προς το στρώμα εισόδου (συμπεριλαμβανομένης και της πόλωσης b_j). Αυτό συμβαίνει διότι το «σφάλμα διαδίδεται προς τα πίσω» και απαιτείται να έχουμε τις ανανεωμένες τιμές για τα βάρη των κρυφών κόμβων προτού βρούμε τα βάρη για τους κόμβους εισόδου (Zaki & Meira, 2020).



Σχήμα 15. Παρουσιάζεται συνοπτικά η διαδικασία οπίσθιας διάδοσης σε έναν κρυφό κόμβο από το σύνολο των $m-$ κρυφών κόμβων. Με $\delta_j = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j}$ συμβολίζουμε την τιμή από τον κόμβο εξόδου o_j που θα θεωρήσουμε ως την νέα «τιμή» του κόμβου εξόδου, πηγαίνοντας προς τα πίσω. *Από Zaki & Meira (2020).*

Ανανέωση βαρών μεταξύ της στοιβάδας εξόδου και της κρυφής στοιβάδας

Σε αυτό το σημείο υποθέτουμε ακόμη πως (α) θα πραγματοποιήσουμε βελτιστοποίηση με αλγόριθμους κλίσεων και (β) πως η συνάρτηση ελέγχου γράφεται ως $E_{\boldsymbol{x}}$. Το (α) θα μπορούσε να αντιστοιχεί στην κλίση κατάβασης (GD) ενώ το (β) θα μπορούσε να είναι το Μέσο Τετραγωνικό Σφάλμα (MSE), τα οποία αναφέραμε στο Κεφάλαιο 1. Σε αυτό το σημείο, η διαφορά της μεθοδολογίας με την εμπρόσθια διάδοση, έγκειται στις τιμές της *καθαρής κλίσης* (net gradient) η οποία θα υπολογιστεί σε κάθε κόμβο εξόδου o_j:

$$\delta_j = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial f(net_j)} \cdot \frac{\partial f(net_j)}{\partial net_j},$$

από όπου βλέπουμε την ανάγκη που αναφέραμε προηγουμένως ως προς την παραγωγισιμότητα των συναρτήσεων απώλειας και ενεργοποίησης. Επιπλέον, θεωρούμε ως $\mathbf{W}_\mathbf{o}$ τον πίνακα από βάρη μεταξύ των κόμβων εξόδου και των κρυφών κόμβων και ως $\nabla_{\mathbf{W}_\mathbf{o}}$ τον αντίστοιχο πίνακας με τις αποκλίσεις των στοιχείων του $\mathbf{W}_\mathbf{o}$. Ορίζουμε ακόμα το διάνυσμα με τις πολώσεις της κρυφής στοιβάδας προς την στοιβάδα εξόδου, με $\boldsymbol{b}_o$ και με διάνυσμα κλίσεων το $\nabla_{\mathbf{b}_\mathbf{o}}$. Δηλαδή έχουμε τις σχέσεις:

$$\boldsymbol{W}_0 = \begin{pmatrix} w_{11} & \cdots & w_{1p} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mp} \end{pmatrix}, \qquad \nabla_{\mathbf{W}_\mathbf{o}} = \begin{pmatrix} \nabla_{w_{11}} & \cdots & \nabla_{w_{1p}} \\ \vdots & \ddots & \vdots \\ \nabla_{w_{m1}} & \cdots & \nabla_{w_{mp}} \end{pmatrix}$$

$$\boldsymbol{b}_o = (b_1, b_2, \dots, b_p), \qquad \nabla_{\mathbf{b_o}} = \left(\nabla_{b_1}, \nabla_{b_2}, \dots, \nabla_{b_p}\right).$$

Σε αυτές, έχουμε θεωρήσει ότι:

$$\nabla_{w_{ij}} = \frac{\partial E_{\boldsymbol{x}}}{\partial w_{ij}} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial}{\partial w_{ij}}\left\{b_j + \sum_{k=1}^{m} w_{kj} \cdot z_k\right\} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot z_i = \delta_j \cdot z_i$$

$$\nabla_{b_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial b_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial net_j}{\partial b_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial}{\partial b_j}\left\{b_j + \sum_{k=1}^{m} w_{kj} \cdot z_k\right\} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} = \delta_j$$

Δηλαδή, παρατηρούμε πως υπάρχει άμεση σχέση με τα διανύσματα κλίσεων και την τιμή της καθαρής κλίσης δ_j. Συνολικά, τα βάρη θα ανανεωθούν για ρυθμό μάθησης η, στην t επανάληψη μέσω του αναδρομικού τύπου:

$$\boldsymbol{W}_o^{t+1} = \boldsymbol{W}_o^t - \eta \cdot \nabla_{\boldsymbol{W}_o^t} = \boldsymbol{W}_o^t - \eta \cdot \delta_j \cdot z_i$$

$$\boldsymbol{b}_o^{t+1} = \boldsymbol{b}_o^t - \eta \cdot \nabla_{\boldsymbol{b}_o^t} = \boldsymbol{b}_o^t - \eta \cdot \delta_j$$

Συνεπώς, παρατηρούμε πως υπάρχει άμεση σχέση μεταξύ της ανανέωσης των βαρών μεταξύ της κρυφής στοιβάδας και της στοιβάδας εξόδου, με την παράγωγο των συναρτήσεων απώλειας και συναρτήσεων ενεργοποίησης.

Ανανέωση βαρών μεταξύ της κρυφής στοιβάδας και της στοιβάδας εισόδου

Ακολουθώντας παρόμοια διαδικασία, θεωρούμε τους πίνακες $\mathbf{W}_h$ από τα βάρη μεταξύ των κρυφών κόμβων και των κόμβων εισόδου, με $\nabla_{\boldsymbol{W}_h}$ είναι ο αντίστοιχος πίνακας με τις κλίσεις ενώ με $\boldsymbol{b}_j$ συμβολίζουμε τα βάρη του κόμβου πόλωσης στο στρώμα εισόδου, και με $\nabla_{\boldsymbol{b}_j}$ τις αντίστοιχες κλίσεις. Υπολογίζουμε πάλι τις αντίστοιχες ποσότητες, οι οποίες όμως είναι όσο αφορά την στοιβάδα *εισόδου*:

$$\nabla_{w_{ij}} = \frac{\partial E_{\boldsymbol{x}}}{\partial w_{ij}} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial}{\partial w_{ij}}\left\{b_j + \sum_{k=1}^{m} w_{kj} \cdot x_k\right\} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot z_i = \delta_j \cdot x_i$$

$$\nabla_{b_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial b_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial net_j}{\partial b_j} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial}{\partial b_j}\left\{b_j + \sum_{k=1}^{m} w_{kj} \cdot x_k\right\} = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot 1 = \delta_j$$

Ενώ η καθαρή κλίση που προέκυψε στην κρυφή στοιβάδα, θα είναι ίση με:

$$\delta_j = \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} = \sum_{k=1}^{p} \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial net_k}{\partial z_j} \frac{\partial z_j}{\partial net_j} = \frac{\partial z_j}{\partial net_j} \sum_{k=1}^{p} \frac{\partial E_{\boldsymbol{x}}}{\partial net_j} \cdot \frac{\partial net_k}{\partial z_j}$$

Συνεπώς, $\delta_j = \frac{\partial f(net_j)}{\partial net_j} \sum_{k=1}^{p} \delta_k \cdot w_{jk}$.

Όπου με δ_k είναι η τιμή από τις *καθαρές κλίσεις* που υπολογίσαμε για την *στοιβάδα εξόδου*, και w_{jk} είναι τα βάρη που συνδέουν την στοιβάδα εξόδου με την κρυφή στοιβάδα. Συνεπώς, αυτή η εξίσωση είναι η αφορμή που η *οπίσθια διάδοση* έχει την αξία της: οι κλίσεις της συνάρτησης απώλειας στην στοιβάδα εξόδου, συνεισφέρουν στις καθαρές κλίσεις της κρυφής στοιβάδας, η οποία θα ανανεώσει τα βάρη μεταξύ της κρυφής στοιβάδας και της στοιβάδας εισόδου. Τελικά, τα βάρη θα ανανεωθούν μέσω του αναδρομικού τύπου:

$$\boldsymbol{W}_h^{t+1} = \boldsymbol{W}_h^t - \eta \cdot \nabla_{\boldsymbol{W}_h^t} = \boldsymbol{W}_h^t - \eta \cdot \delta_j \cdot x_i$$

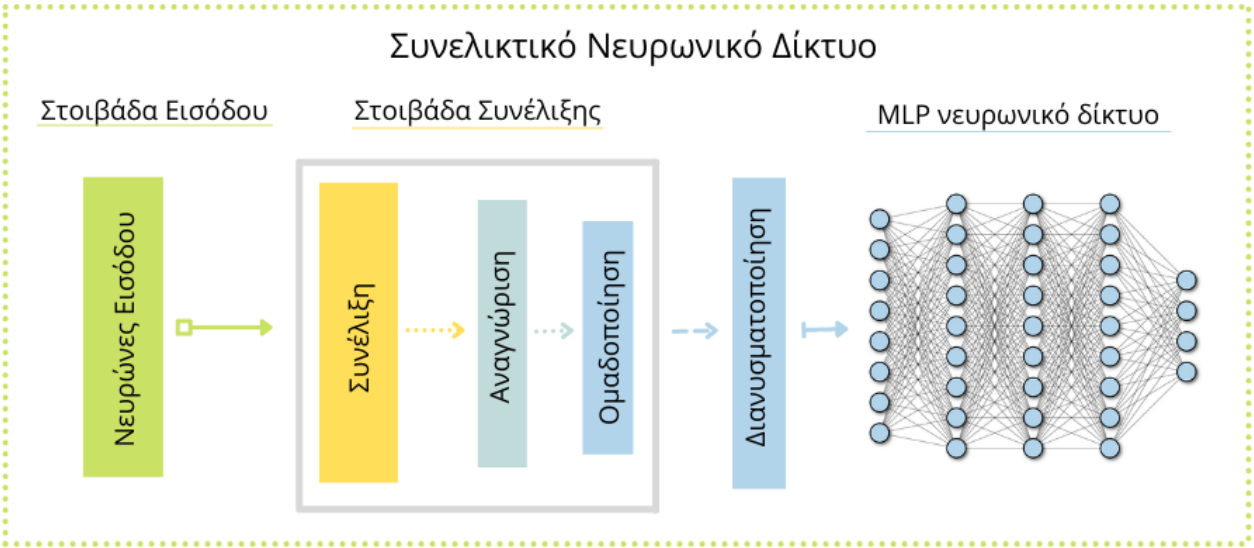
$$\boldsymbol{b}_h^{t+1} = \boldsymbol{b}_h^t - \eta \cdot \nabla_{\boldsymbol{b}_h^t} = \boldsymbol{b}_h^t - \eta \cdot \delta_j$$

Σε αυτό το σημείο, ολοκληρώσαμε μία επανάληψη της εκπαίδευσης του νευρωνικού δικτύου, δηλαδή καλύψαμε την μεθοδολογία για το $t = 1$. Κάθε επανάληψη αποκαλείται ως *εποχή* (epoch). Η επανάληψη αυτή αφορούσε **ένα** σημείο των δεδομένων εκπαίδευσης. Έπειτα, θα θεωρήσουμε ένα άλλο σημείο και θα επαναλάβουμε πάλι τα βήματα, μέχρι να έχουμε τα επιθυμητά αποτελέσματα.

Ενότητα 2.
Συνελικτικά Νευρωνικά Δίκτυα

2.2.1 Αρχιτεκτονική

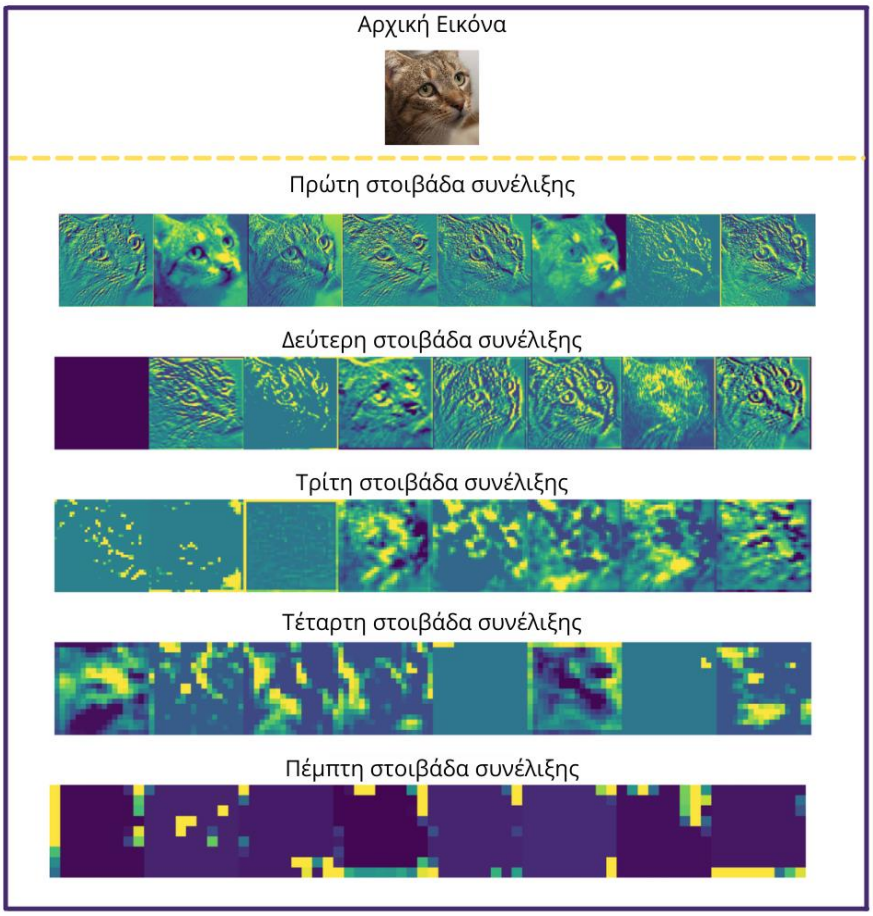
Τα *συνελικτικά νευρωνικά δίκτυα* (convolutional neural networks, CNN) είναι φτιαγμένα για να εκμεταλλεύονται τις χωρικές ή χρονικές δομές στα δεδομένα εισόδου (Zaki & Meira, 2020), δηλαδή αφορούν χωρικά ή χρονικά συσχετισμένα δεδομένα. Έχουν αρχιτεκτονική από βαθιά νευρωνικά δίκτυα πολυεπίπεδων αντιλήπτρων, με τους πρώτους που τα πρότειναν τους LeCun et al (1998) με εφαρμογή σε εικόνες από χειρόγραφους αριθμούς. Μετά από λίγα χρόνια, προτάθηκε μία βελτίωση των CNN από τους Le Cun et al (1990). Η διαφορά τους με τα MLP νευρωνικά δίκτυα είναι πως στις κρυφές στοιβάδες συμπεριλαμβάνουν στοιβάδες που εκτελούν διεργασίες μείωσης της διάστασης των δεδομένων και εξαγωγής χαρακτηριστικών. Αυτές οι στοιβάδες περιλαμβάνουν κυρίως τις στοιβάδες *συνέλιξης* (convolutional layers). Οι στοιβάδες συνέλιξης, περιέχουν αρκετές διεργασίες όπως το αρχικό στάδιο της συνέλιξης, στο στάδιο της *αναγνώρισης* (detector) όπου γίνεται η εφαρμογή μίας μη – γραμμικής συνάρτησης ενεργοποίησης, και το στάδιο της *ομαδοποίησης* (pooling) (Goodfellow, 2016). Μετά από αυτή τη στοιβάδα συνέλιξης, συνηθίζεται η σύνδεσή με ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο (Chiroma et al. 2019), δηλαδή συνδέονται με ένα MLP. Τα διαφορετικά στάδια *μίας στοιβάδας συνέλιξης*, δεν έχουν κόμβους μεταξύ τους. Υπάρχουν μόνο τα βάρη που συνδέουν τους νευρώνες μεταξύ της στοιβάδας εισόδου και της στοιβάδας που πραγματοποιείται η συνέλιξη, ενώ δεν υπάρχουν βάρη μεταξύ τα βάρη μεταξύ του σταδίου της συνέλιξης και της αναγνώρισης της ομαδοποίησης (Goodfellow, 2016). απεικονίζονται στο σχήμα 20. Επιπλέον, το στάδιο της μη – γραμμικής ενεργοποίησης αναφέρεται ως το στάδιο *αναγνώρισης* (detector) και το επόμενο στάδιο, ως το στάδιο της ομαδοποίησης. Μπορεί να αναφερθούμε στην διεργασία της ομαδοποίησης ως «στοιβάδα ομαδοποίησης» για διευκόλυνση, γνωρίζοντας όμως ότι αποτελεί μόνο μία σύμβαση. Στο στάδιο της αναγνώρισης, πρακτικά η μη – γραμμική συνάρτηση που αναφέρεται, θα είναι μία *συνάρτηση ενεργοποίησης*, η οποία δέχεται ως *καθαρή εισροή* ***net*** την συνέλιξη μαζί με την πόλωση της στοιβάδας (Zaki & Meira, 2020).



Σχήμα (20). Αναπαράσταση Συνελικτικού Νευρωνικού Δικτύου με μία στοιβάδα συνέλιξης. Το διαβάζουμε από αριστερά προς τα δεξιά. Αρχικά, τα δεδομένα διέρχονται από την στοιβάδα εισόδου (στο σχήμα, το παραλληλόγραμμο με το πράσινο χρώμα) στην στοιβάδα συνέλιξης (παραλληλόγραμμο με κίτρινο χρώμα) όπου εκτελείται η πράξη της συνέλιξης για τα γειτονικά δεδομένα. Έπειτα, τα δεδομένα διέρχονται από μία μη – γραμμική συνάρτηση ενεργοποίησης (στάδιο της αναγνώρισης, γκρι

παραλληλόγραμμο) και στην συνέχεια πραγματοποιείται μείωση διάστασης μέσω της ομαδοποίησης (μπλε παραλληλόγραμμο). Μπορούμε να έχουμε όσες στοιβάδες συνέλιξης θέλουμε (εδώ απεικονίζεται μία φορά), ενώ οι διαδικασίες τους αποτελούν μία υπερπαραμέτρο του συνελικτικού νευρωνικού συστήματος. Έπειτα, θεωρούμε το αποτέλεσμα από την στοιβάδα συνέλιξης ως ένα μονοδιάστατο διάνυσμα (Yang, 2019) (δεύτερο μπλε παραλληλόγραμμο, *διανυσματοποίηση* (vectorization) (Chollet & Allaire, 2018)), ώστε να μπορεί να αξιοποιηθεί ως δεδομένα εισόδου σε ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο πολυεπίπεδων αντιλήπτρων (MLP) (απεικονίζεται δεξιά στο σχήμα). Η μάθηση συμπεριλαμβάνει τους νευρώνες που υπάρχουν στη στοιβάδα συνέλιξης και της ομαδοποίησης, δηλαδή περιέχει τα βήματα της *εμπρόσθιας τροφοδότησης,* μέχρι να φτάσουμε στους νευρώνες εξόδου, και μετά της *οπίσθιας διάδοσης* όπου ανανεώνονται τα βάρη στις στοιβάδες που πραγματοποιείται η συνέλιξη και τα βάρη στο πλήρως συνδεδεμένο νευρωνικό δίκτυο (Goodfellow, 2016). *Αξιοποιήθηκε η ιστοσελίδα canva.com.*

Η *συνέλιξη* αποτελεί μία *γραμμική συνάρτηση,* η οποία δέχεται ως δεδομένα εισόδου μόνο μία δισδιάστατη, μικρή επιλογή από γειτονικά δεδομένα (Goodfellow, 2016), δηλαδή δεν δέχεται ως είσοδο την ποσότητα net_k όλων των δεδομένων εισόδου. Η «σάρωση» των γειτονικών δεδομένων γίνεται σε 2 διαστάσεις και αποκαλείται *κυλιόμενο παράθυρο* (sliding window), επειδή μετά θα «κυλίσει» σε ένα παρόμοιο σύνολο δεδομένων κοντά στο πρώτο που συλλέχθηκε (Zaki & Meira, 2020).
Συνηθίζεται ενδιάμεσα στις συνελικτικές στοιβάδες να τοποθετούνται στοιβάδες ομαδοποίησης, αλλά επειδή η συνέλιξη αποτελεί έναν γραμμικό μετασχηματισμό, συνηθίζεται ακόμη ένας μη – γραμμικός μετασχηματισμός μεταξύ της στοιβάδας συνέλιξης και της στοιβάδας ομαδοποίησης (Chiroma et al. 2019). Η ομαδοποίηση θα λάβει την πιο σημαντική πληροφορία από μία ομάδα νευρώνων που προέκυψαν από την συνέλιξη μετά την εφαρμογή μίας μη – γραμμικής συνάρτησης ενεργοποίησης (Goodfellow, 2016). Αυτό γίνεται για παράδειγμα επιλέγοντας την μέγιστη τιμή από δέκα νευρώνες στην στοιβάδα συνέλιξης. Αυτό έχει ως αποτέλεσμα την *ανθεκτικότητα* του μοντέλου σε μικρές αλλαγές στα δεδομένα εισόδου, δηλαδή αυτό χαρακτηρίζεται ως *αμετάβλητο* (invariant) (Goodfellow, 2016).
Το βασικό χαρακτηριστικό των συνελικτικών νευρωνικών δικτύων, είναι πως αρχικά η συνέλιξη εξάγει τα χαρακτηριστικά που «σαρώνει» στα δεδομένα, αυτά ενεργοποιούνται μέσω της μη – γραμμικής συνάρτησης και έπειτα η στοιβάδα ομαδοποίησης διαλέγει το πιο σημαντικό χαρακτηριστικό εξ αυτών (Zaki & Meira, 2020) και αυτό επαναλαμβάνεται για όσες στοιβάδες συνέλιξης έχουμε. Αυτός είναι ένας λόγος που τα συνελικτικά νευρωνικά δίκτυα έχουν καλή συμπεριφορά όταν δέχονται εικόνες ως δεδομένα εισόδου, επειδή μπορούν να δουλέψουν με δεδομένα τα οποία είναι συσχετισμένα με μία τοπολογία σε δύο διαστάσεις (Goodfellow, 2016). Επιπρόσθετα, λόγο μίας μικρής επιλογής των δεδομένων που απαιτεί η συνέλιξη, δεν συνδέονται όλοι οι νευρώνες της στοιβάδας εισόδου με όλους τους νευρώνες της στοιβάδας συνέλιξης. Επιπλέον, οι συναρτήσεις συνέλιξης συνηθίζεται να χρησιμοποιούνται πολλές φορές. Σε αυτό οφείλεται η ικανότητα των βαθιών συνελικτικών νευρωνικών δικτύων (CNN) να δημιουργούν μία *χωρική, αφηρημένη ιεραρχική αναπαράσταση* των δεδομένων (Chollet & Allaire, 2018). Για παράδειγμα, αν θεωρήσουμε ως δεδομένα την φωτογραφία μίας γάτας, η πρώτη στοιβάδα συνέλιξη (ως το τελικό αποτέλεσμα των διεργασιών της συνάρτησης ενεργοποίησης και της ομαδοποίησης) θα «αναπαριστά» τα σημαντικά χαρακτηριστικά της γάτας, ενώ όσο αυτή συνδέεται με περισσότερες στοιβάδες συνέλιξης, αυτές θα καταλήξουν να αναπαριστούν ολοένα και περισσότερα αφηρημένα χαρακτηριστικά, εξού και η *ιεράρχηση.* Αυτό απεικονίζει το Σχήμα 21 που ακολουθεί.

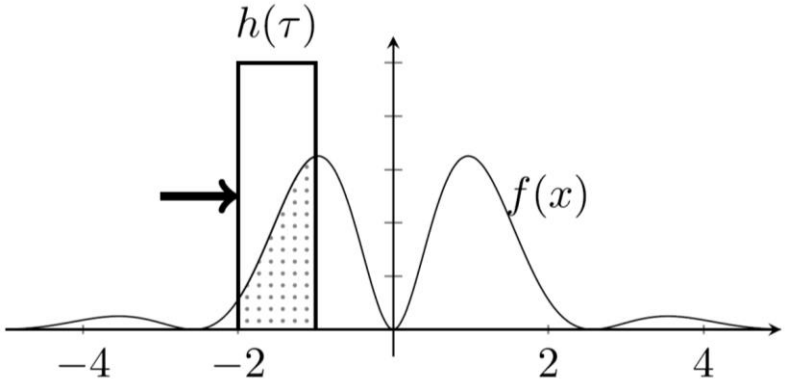


Σχήμα 21. Απεικόνιση της εικόνας μίας γάτας, καθώς αυτή διέρχεται από τις συνελικτικές στοιβάδες. Στην κορυφή του σχήματος είναι η αρχική εικόνα της γάτας, ενώ προς τα κάτω είναι διαδοχικά οι συνελικτικές στοιβάδες. Με έντονο χρώμα είναι τα χαρακτηριστικά των νευρώνων που *ενεργοποιήθηκαν*. Παρατηρούμε πως καθώς προχωράμε στις επόμενες στοιβάδες, οι αναπαραστάσεις γίνονται ολοένα και περισσότερο αφηρημένες, μέχρι που στην τέταρτη και πέμπτη στοιβάδα συνέλιξης πλέον δεν είναι αναγνωρίσιμο ότι απεικονίζεται. Αυτό συμβαίνει διότι αυτές οι στοιβάδες περιέχουν πληροφορίες οι οποίες θα είναι χρήσιμες στην *κατηγοριοποίηση* της εικόνας, όπως «αιχμές από μάτια γάτας» ή «μύτη γάτας». Οι μαύρες εικόνες της στοιβάδας 5, οφείλονται στο ότι το νευρωνικό δίκτυο του παραδείγματος έχει εκπαιδευτεί με διάφορες εικόνες, με αποτέλεσμα λχ. να μην μπορεί να βρεθεί στην φωτογραφία το αντίστοιχο χαρακτηριστικό «ουρά αλόγου» με αποτέλεσμα να περιέχει μαύρα κενά, αλλά οι κίτρινες ενότητες πιθανώς σχετίζονται με τα χαρακτηριστικά της κλάσης «γάτα». *Από Arden Dertat (2017) "Applied Deep Learning - Part 4: Convolutional Neural Networks" towardsdatascience.com*.

Στοιβάδα Συνέλιξης

Η συνέλιξη που αναφέρεται στον τομέα της επιστήμης των δεδομένων, συνήθως διαφέρει από την συνέλιξη των θεωρητικών μαθηματικών, αν και δεν διαφέρουν πάρα πολύ (Goodfellow, 2016). Μαθηματικά, η (μονοδιάστατη) συνέλιξη δύο συναρτήσεων f, h, συμβολίζεται με $f * h$ και έχει την ακόλουθη μορφή. Επίσης, στο επόμενο σχήμα απεικονίζεται η επίδραση της h στο εμβαδόν της f.

$$(f * h)(x) = \int_{-\infty}^{+\infty} f(x-\tau)h(\tau)d\tau = \int_{-\infty}^{+\infty} f(\tau)h(x-\tau)d\tau$$

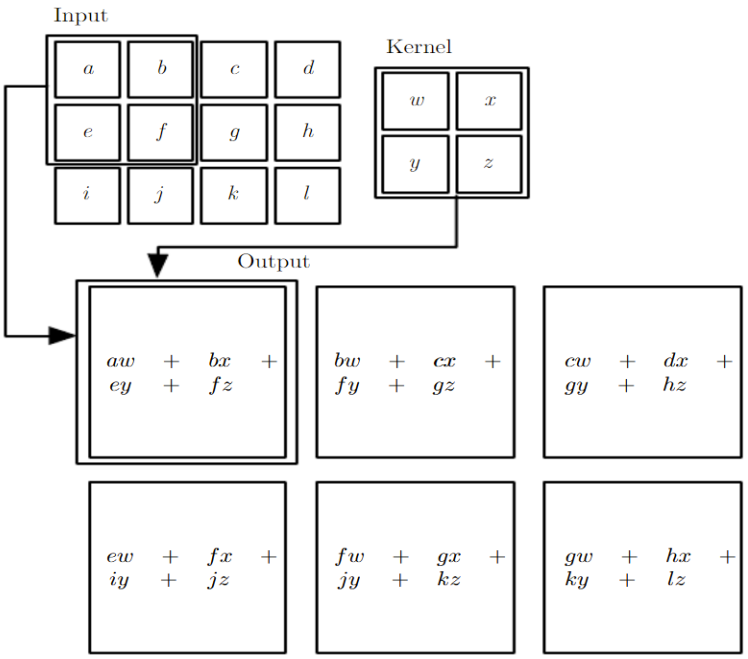


Στο Γράφημα 4, απεικονίζεται η συνέλιξη $f * h$, των συναρτήσεων f, h, στο πεδίο ορισμού $[-4,4]$ όπου $h = a\ (\sigma\tau\alpha\theta.)$ Σε όλο το $[-4,4]$, ενώ $\alpha > \max f(x), x \in [-4,4]$. Η γραμμοσκιασμένη περιοχή είναι η τιμή της συνέλιξης. Διαισθητικά, η συνέλιξη σαρώνει ένα μέρος από ολόκληρο το εμβαδό $\int_{-4}^{4} f(x)dx$ και εστιάζει σε αυτό που ορίζει η συνάρτηση βήματος $h(x)$. Δηλαδή, η h δρα ως ένα «παράθυρο» μέσα από το οποίο «βλέπουμε» μόνο ένα κομμάτι από το εμβαδό. Κατ' αναλογία στην εικόνα που εισάγεται σε ένα νευρωνικό δίκτυο, η συνέλιξη θα δράσει ως ένα «παράθυρο» ή ως ένα «φίλτρο», το οποίο θα σαρώσει την εικόνα, αντί να την δει ολόκληρη. *Από Yang (2019).*

Η συνέλιξη στην επιστήμη των δεδομένων επίσημα ονομάζεται *διασυσχέτιση* (cross – correlation) αλλά συνηθίζεται ο συμβιβασμός με το όνομα *συνέλιξη*. Στον τύπο που ακολουθεί, έχουμε θεωρήσει την δισδιάστατη (2D) συνέλιξη όπου ως f έχουμε θεωρήσει τον *πίνακα εισόδου* (input) I, και ως συνάρτηση h έχουμε θεωρήσει τον *πυρήνα* (kernel) ο οποίος περιλαμβάνει τα βάρη μεταξύ της στοιβάδας εισόδου και της στοιβάδας της διεργασίας της συνέλιξης. Επιπλέον, έχουμε θεωρήσει πως έχουμε ένα πλήθος από σημεία στην στοιβάδα εισόδου, τα οποία είναι διακριτά, συνεπώς ορίζουμε την *διακριτή 2D συνέλιξη* με τύπο (Goodfellow, 2016):

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)\, K(m,n)$$

Ο πυρήνας $K(m,n)$ στον μαθηματικό του ορισμό, σε αυτή την μορφή της εξίσωσης οφείλει να είναι ο ανάστροφος από τον κανονικό. Στην επιστήμη των δεδομένων όμως, τον θεωρούν κανονικά σαν να μην έχει αναστραφεί. Πρακτικά όμως δεν έχει διαφορά στην μάθηση του νευρωνικού δικτύου, ούτε μεταφέρεται η πληροφορία πως είναι ανάστροφος στις επόμενες διεργασίες ή στοιβάδες (Goodfellow, 2016). Συνολικά, έχουμε πως η συνέλιξη μεταξύ ενός πίνακα $X \in \mathbb{R}^{n \times n}$ και ενός πίνακα $W \in \mathbb{R}^{k \times k}$, θα έχει διάσταση $\dim(X * W) = (n-k+1) \times (n-k+1)$ (Zaki & Meira, 2020).
Δηλαδή, η συνέλιξη έχει ως αποτέλεσμα ένα νέο στοιχείο (εδώ είναι πίνακας) το οποίο θα έχει μικρότερη διάστασης από αυτό που ξεκινήσαμε. Για να γίνει κατανοητό αυτό, δίνουμε το ακόλουθο σχήμα της διαδικασίας της συνέλιξης μεταξύ των νευρώνων της στοιβάδας εισόδου και της πρώτης στοιβάδας συνέλιξης, στην εκτέλεση της συνέλιξης.



Σχήμα 22. Πίνακας όπου φαίνονται οι διεργασίες της συνέλιξης στον πίνακα εισόδου I με τον πυρήνα (kernel) K. Πάνω στον πίνακα εισόδου απεικονίζονται τα «κυλιόμενα παράθυρα» τα οποία αντιστοιχούν στην συνέλιξη με τον πυρήνα. Στα έξι κουτιά κάτω από την μέση της εικόνας, το πάνω αριστερά όπου συγκλίνουν τα δύο βέλη θα είναι το αποτέλεσμα του πολλαπλασιασμού μεταξύ των πινάκων $\begin{pmatrix} a & b \\ e & f \end{pmatrix}\begin{pmatrix} w & x \\ y & z \end{pmatrix} = aw + by + ex + fz$, το οποίο αποτελεί το πρώτο στοιχείο του πίνακα της συνέλιξης. Μετά, για την επιλογή του κυλιόμενου παραθύρου ίση με τον πίνακα $\begin{pmatrix} b & c \\ f & g \end{pmatrix}$ θα υπολογίσουμε πάλι το γινόμενο με τον πίνακα $\begin{pmatrix} w & x \\ y & z \end{pmatrix}$. Δηλαδή, τα έξι κουτιά αποτελούν τις έξι τιμές του πίνακα που θα δώσει ως αποτέλεσμα η συνέλιξη. Αυτός αναφέρεται ως ο *χάρτης των χαρακτηριστικών* (feature map). *Σχήμα από Goodfellow, (2016).*

Αναλυτικότερα, για την συνέλιξη σε 3 διαστάσεις σύμφωνα με τους Zaki & Meira (2020), θεωρούμε τον πίνακα εισόδου $\mathbf{I} \in \mathbb{R}^{n \times n \times m}$, όπου έχουμε n – το πλήθος γραμμές και στήλες, και m – το πλήθος *κανάλια* (channels). Τα κανάλια αφορούν, για παράδειγμα, 3 χρώματα που αντιστοιχούν σε ένα πίξελ από τα $n \times n$ μίας εικόνας. Επίσης θεωρούμε τον πυρήνα ή αλλιώς το *τρισδιάστατο φίλτρο* (3D filter) $\mathbf{K} \in \mathbb{R}^{k \times k \times m}$, όπου $k \leq n$. Το k ονομάζεται *μέγεθος του παραθύρου* (window size). Τελικά, θα ορίσουμε και το κυλιόμενο παράθυρο $\mathbf{I}_{\mathrm{k}}(i,\ j) \in \mathbb{R}^{k \times k \times m}$, το οποίο είναι ένας τανυστής (γενικευμένο διάνυσμα) που βρίσκεται στο (i, j) σημείο του $\mathbf{I}$, και εκτείνεται σε όλα του τα στοιχεία. Αυτός φαίνεται στο επόμενο σχήμα. Θεωρούμε τις τιμές των ανωτέρω:

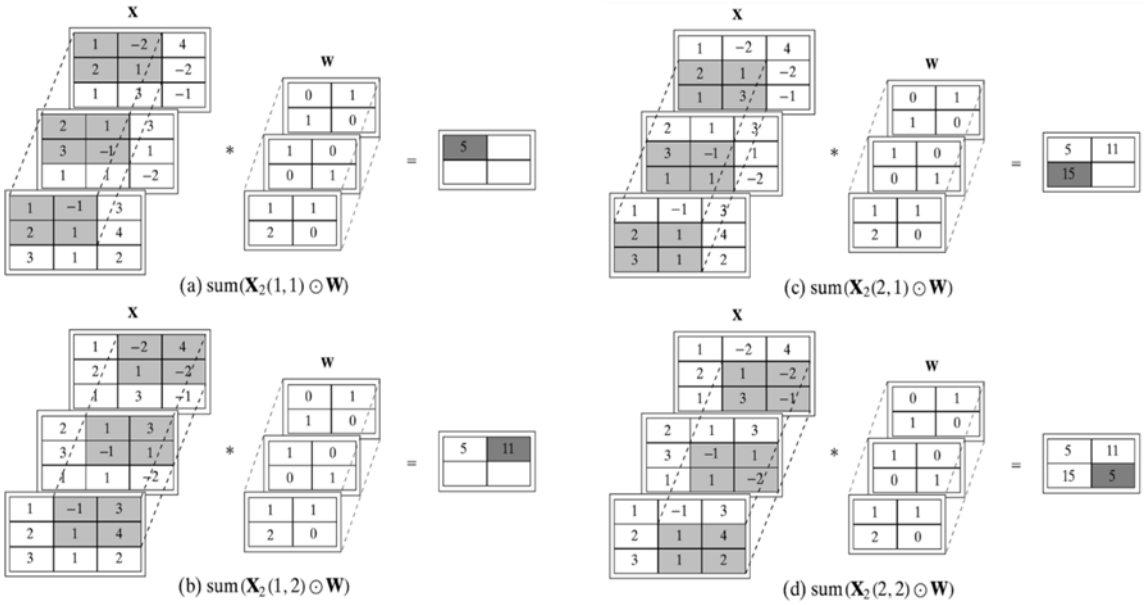
$$\mathbf{I} = \left(x_{ijq} : 1 \leq i, j \leq n \text{ και } 1 \leq q \leq m \right) \in \mathbb{R}^{n \times n \times m}$$
$$\mathbf{K} = \left(w_{abc} : 1 \leq \alpha, \beta \leq k, \text{και } 1 \leq c \leq m \right) \in \mathbb{R}^{k \times k \times m}$$
$$\mathbf{I}_{\mathrm{k}}(i,j) = \left(x_{ijq} : 1 \leq i, j \leq n - k + 1, 1 \leq q \leq m \right) \in \mathbb{R}^{k \times k \times m}$$

Συνολικά, θα έχουμε ότι η συνέλιξη μεταξύ του $\mathbf{X}$ και του $\mathbf{W}$ θα δίνεται από τον πίνακα με στοιχεία:

$$\mathbf{I} * \mathbf{W} = (\mathrm{sum}(\mathbf{I}_{\mathrm{k}}(i,j) \odot \mathbf{K}) \mid 1 \leq i,\ j \leq n - k + 1),$$
$$\mathrm{sum}(\mathbf{I}_{\mathrm{k}}(i,j,q) \odot \mathbf{K}) = \sum_{\alpha=1}^{k} \sum_{b=1}^{k} \sum_{c=1}^{m} x_{i+\alpha-1, j+b-1, q+c-1} w_{abc}$$

Είναι σημαντικό να αναφερθεί ότι, στην παρούσα περίπτωση, η συνάρτηση sum, ονομάζεται <u>*συνάρτηση συσσωμάτωσης*</u> *(aggregation function)* (Zaki & Meira, 2020). Συνεπώς, η συνέλιξη των $\mathbf{I} * \mathbf{K}$ είναι ένας

$(n-k+1)\times(n-k+1)$ – διάστατος πίνακας, μικρότερος στις δύο διαστάσεις από τον αντίστοιχο αρχικό πίνακα εισόδου **I**, ενώ δεν έχει τρίτη διάσταση, το οποίο διευκολύνει τις πράξεις. Αυτά απεικονίζονται στο επόμενο γράφημα.



Σχήμα 22. Διαδικασία της συνέλιξης μεταξύ του τρισδιάστατου πίνακα εισόδου $\mathbf{I}\in\mathbb{R}^{3\times3\times3}$, και του πυρήνα $\mathbf{K}=\mathbf{W}$. Επιπλέον, με ελαφρύ γκρι χρώμα και διακεκομμένες γραμμές πάνω στον **I**, εικονίζονται τα «κυλιόμενα παράθυρα» $\mathbf{I}_k(i,\ j)$ στις θέσεις $(i,\ j)$, διάστασης $2\times2\times3$. Με σκούρο χρώμα απεικονίζονται οι τιμές $(i,\ j)$ του πίνακα συνελίξεως που προκύπτει για την κάθε θέση των (i,j). *Σχήμα από Zaki & Meira (2020).*

Παραγέμισμα & Βηματισμός

Επειδή όπως είδαμε η συνέλιξη επιστρέφει έναν πίνακα μικρότερης διάστασης από τον πίνακα εισόδου, όταν βάλουμε πολλές στοιβάδες συνέλιξης μαζί θα έχουμε ακόμα εντονότερα αποτελέσματα μείωσης της διάστασης ενώ δεν είναι δύσκολο να καταλήξουμε ακόμα και με ένα στοιχείο σε μία στοιβάδα. Για αυτό για τα μοντέλα με περισσότερες στοιβάδες συνέλιξης και ομαδοποίησης, είναι αναγκαία η τεχνική του *παραγεμίσματος* (padding) (Goodfellow, 2016).

Η διαδεδομένη τεχνική είναι το *παραγέμισμα με μηδενικά* (zero – padding) κοντά στους «εξωτερικούς» νευρώνες της κάθε στοιβάδας (ή τον πρώτο και τον τελευταίο που θα μπουν στην στοιβάδα). Στην στοιβάδα, αυτοί οι κόμβοι έχουν τιμή 0 και απλά συνεισφέρουν στην διατήρηση της διάστασης, ενώ θα έχουν το αποτέλεσμα πως η τιμή της συνέλιξης θα «απλωθεί» όταν αναφέρεται σε αυτούς τους νευρώνες, καθώς έχουν τιμή 0. Όσο αφορά τους πίνακες της συνέλιξης, το ίδιο αποτέλεσμα έχουμε με το να βάλουμε γύρω από τον πίνακα μηδενικά, ώστε να αυξηθεί η διάστασή του όσο απαιτείται. Το πόσα μηδενικά χρειάζονται, εξαρτάται από το *δεκτικό πεδίο* (receptive field) των νευρώνων στις συνελικτικές στοιβάδες (Goodfellow, 2016).

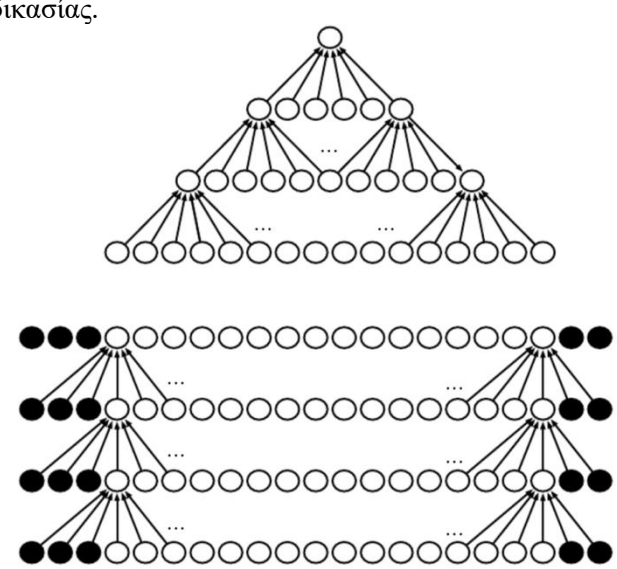
Επίσης, η διάσταση του πίνακα που προκύπτει μετά την συνέλιξη με παραγέμισμα, θα ισούται με $(n+2p)\times(n+2p)\times m$. Προκύπτει ότι, για να διατηρηθεί το μέγεθος του αρχικού πίνακα, οφείλουμε να έχουμε $p=\left\lceil\frac{k-1}{2}\right\rceil$, όπου $\lceil n+q\rceil=n+1$, όταν $n\in\mathbb{N}$ και $0<q<1$, η συνάρτηση *«ταβάνι» (ceiling).*

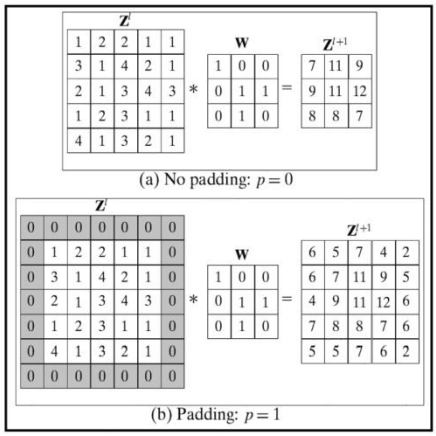
Μέσω της διαδικασίας του παραγεμίσματος, ξεπερνάμε τον περιορισμό της μείωσης της διάστασης και πλέον μπορούμε να έχουμε αυθαίρετο πλήθος στοιβάδων συνέλιξης (Zaki & Meira, 2020). Σύμφωνα με

τους Zaki & Meira, εάν θεωρήσουμε $s \geq 1$, τότε έχουμε, για έναν τρισδιάστατο πίνακα $\mathbf{Z}^l$ διάστασης $n_l \times n_l \times m_l$ και φίλτρου $\mathbf{W}$ διάστασης $k \times k \times m_l$,

$$\mathbf{Z}^l * \mathbf{W} = \left(\text{sum}(\boldsymbol{Z}_k^l(1+ts, 1+ts) \odot \mathbf{W}) : 1 \leq t \leq \left\lceil \frac{n_l - 1}{2} \right\rceil\right) \in \mathbb{R}^{(t+1)\times(t+1)}$$

Στο παρακάτω σχήμα απεικονίζονται αυτές οι τεχνικές και αναφέρονται περαιτέρω χαρακτηριστικά της διαδικασίας.





Σχήμα 23. Απεικονίζονται δύο σχήματα που αφορούν το παραγέμισμα.
Αριστερά, απεικονίζονται οι στοιβάδες της συνέλιξης σε δύο γραφήματα CNN. Στο ***πάνω γράφημα της αριστερής εικόνας*** είναι οι στοιβάδες της συνέλιξης η διαμόρφωσή τους *χωρίς παραγέμισμα*. Προχωράμε στις στοιβάδες προς τα πάνω (ίδια φορά με τα βέλη) ενώ τα βέλη που συνδέουν τους κόμβους της μίας στοιβάδας με τον ένα κόμβος της επόμενης, αναφέρονται στα βάρη των νευρώνων, δηλαδή στα στοιχεία του πυρήνα **K** που αναφέραμε προηγουμένως. Το «πλήθος από βέλη» που απαιτεί ένας νευρώνας για να κάνει την συνέλιξη, ονομάζεται *δεκτικό πεδίο* (receptive field) (Goodfellow, 2016). Παρατηρούμε πως καθώς προχωράμε στις επόμενες στοιβάδες (προς τα πάνω) το πλήθος των νευρώνων μειώνεται. Δηλαδή, εδώ φαίνεται η ιδιότητα της συνέλιξης, που αναφέραμε νωρίτερα, να μειώνει την διάσταση από την μία στοιβάδα στην επόμενη. Στο ***κάτω μέρος της αριστερής εικόνας***, είναι η ίδια δομή στοιβάδων, αλλά επιπλέον έχουμε προσθέσει 5 μηδενικούς κόμβους, 3 αριστερά της δομής και 2 δεξιά της δομής. Με αυτό τον τρόπο, παρατηρούμε πως δεν χάνονται πλέον κόμβοι στις επόμενες στοιβάδες, δηλαδή η διάσταση από τον πίνακα εισόδου **I** παραμένει σταθερή. Επιπλέον, βλέπουμε πως το δεκτικό πεδίο των κόμβων ορίζει πόσα μηδενικά θα βάλουμε. Επισημαίνεται πως είναι δύσκολο να βρεθεί ακριβώς πόσα μηδενικά είναι ιδανικά για την μάθηση (Goodfellow, 2016). *Σχήμα από Goodfellow, 2016.*
Δεξιά, απεικονίζεται η συνέλιξη με δύο πίνακες. Στο ***σχήμα (α) της δεξιάς εικόνας (πάνω)*** πραγματοποιείται η συνέλιξη χωρίς παραγέμισμα, και παρατηρούμε πως το αποτέλεσμα $\boldsymbol{Z}^{l+1}$ είναι διαστάσεων 3×3, ενώ ο αρχικός πίνακας $\boldsymbol{Z}^l$ είναι διαστάσεων 5×5, επειδή χρησιμοποιήθηκε πυρήνας με διάσταση 2×2. Στο ***σχήμα (β) της δεξιάς εικόνας (κάτω)*** είναι οι ίδιοι πίνακες, αλλά χρησιμοποιήθηκε παραγέμισμα. Παρατηρούμε πως στο κέντρου του αποτελέσματος, $\boldsymbol{Z}^{l+1}$, *οι τιμές παραμένουν οι ίδιες όπως χωρίς το παραγέμισμα*, δηλαδή αφορά τον υπό πίνακα:

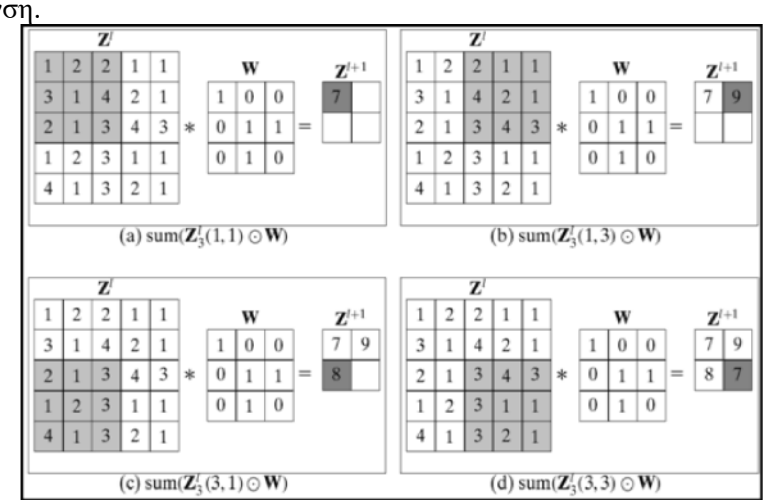
$$\begin{pmatrix} 7 & 11 & 9 \\ 9 & 11 & 12 \\ 8 & 8 & 7 \end{pmatrix}$$

Οι τιμές που έχουν αλλάξει είναι στο εξωτερικό του πίνακα. Εάν είχαν χρησιμοποιηθεί περισσότερα μηδενικά, θα παρατηρούσαμε πως οι τιμές στην άκρη του πίνακα θα ήταν πολύ κοντά στο 0 (Goodfellow, 2016). *Σχήμα από Zaki & Meira (2020).*

Επιπλέον, για μεγάλες εικόνες συνηθίζεται η τεχνική του *βηματισμού* (stride) κατά τον υπολογισμό στην στοιβάδα συνέλιξης, κατά τον οποίο μπορούμε να παραλείψουμε έναν μικρό αριθμό από λχ. πίξελ στην εικόνα, σε κάθε υπολογισμό της συνέλιξης, ώστε να επιταχύνουμε την ταχύτητα των πράξεων (Goodfellow, 2016). Με βάση τον Goodfellow (2016), εάν θέλουμε να υπολογίζουμε τον πυρήνα μόνο κάθε $s-$ το πλήθος πίξελ, τότε θα εφαρμόσουμε μία συνάρτηση c στον τύπο της συνέλιξης:

$$Z_{i,j,k} = c(K,V,s)_{i,j,k} = \sum_{l,m,n} \left[I_{l,(j-1)\times s+m,(k-1)\times s+n}\, K_{i,l,m,n} \right]$$

Ενώ αναφέρει πως υπάρχουν και τεχνικές ώστε ο βηματισμός να μπορεί να γίνεται με διαφορετικό μέγεθος προς κάθε κατεύθυνση.



Σχήμα 24. Απεικονίζεται η διαδικασία του *βηματισμού* (striding) για τον οποίο υπολογίζονται τα παράθυρα $\boldsymbol{Z}_3^l(i+ts, j+ts)$, όπου $s=2$, για τις τιμές του $0 \le t \le 1$, έναντι των συνηθισμένων $\boldsymbol{Z}_3^l(i,j)$. Αυτό έχει ως αποτέλεσμα να υπολογιστούν λιγότερα στοιχεία, με συνέπεια την περαιτέρω μείωση της διάστασης του πίνακα συνέλιξης. Ξεκινάμε από έναν πίνακα βαρών $\boldsymbol{Z}^l$ διάστασης 5×5 και μέσω της συνέλιξης με τον 3×3 πίνακα $\boldsymbol{W}$, για $s=2$, θα καταλήξουμε σε διάσταση 2×2. Τα παράθυρα $\boldsymbol{Z}_3^l(i+2, j+2)$, φαίνονται με απαλό γκρι χρώμα, ενώ οι αντίστοιχες τιμές της συνέλιξης είναι σε σκούρο χρώμα. *Σχήμα από Zaki & Meira (2020).*

Στοιβάδα ομαδοποίησης

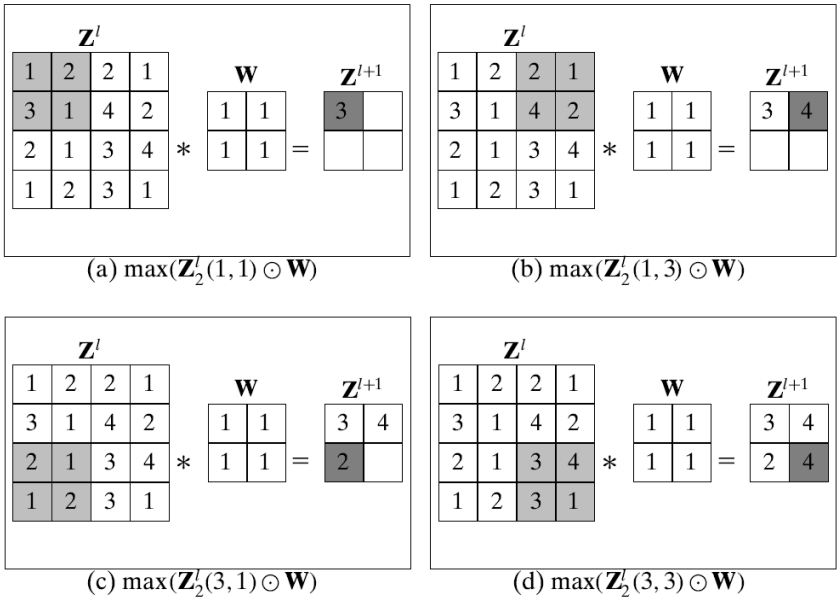
Αναφέραμε προηγουμένως πως στην ποσότητα που υπολόγιζε τα στοιχεία του πίνακα συνέλιξης,

$$sum(\mathbf{I}_k(i,j) \odot K)$$

η συνάρτηση που άθροιζε τα στοιχεία, sum, λέγεται *συνάρτηση συσσωμάτωσης*. Αντί αυτής, μπορούν να χρησιμοποιηθούν και οι συναρτήσεις της μέσης τιμής, avg, καθώς και του μεγίστου, max. Οι στοιβάδες που προκύπτουν μετά την εφαρμογή αυτών των συναρτήσεων συσσώρευσης (avg, max), ονομάζονται *στοιβάδες συγκέντρωσης* (pooling layers) (Zaki & Meira, 2020).

Συνηθίζεται, όταν χρησιμοποιείται η συνάρτηση της μέγιστης τιμής max, να συνδυάζεται με βηματισμό μεγέθους $s=k$, με αποτέλεσμα να προκύπτει το μέγιστο από ξεχωριστά, $k \times k$ παράθυρα για κάθε κανάλι του $\mathbf{Z}^l$. Επιπρόσθετα, χρησιμοποιείται φίλτρο (πυρήνας) με διαστάσεις $k \times k \times 1$, τέτοιο ώστε $\mathbf{K} = \mathbf{1}_{k\times k\times 1}$, δηλαδή με στοιχεία σταθερά και ίσα με το 1, τα οποία δεν ανανεώνονται στην διαδικασία της προς τα πίσω διάδοσης, ενώ επιπλέον έχει ως νευρώνα πόλωσης το 0. Όσο αφορά την συνάρτηση ενεργοποίησης, χρησιμοποιείται αποκλειστικά η ταυτοτική συνάρτηση. Τελικά, προκύπτει πως η συνέλιξη θα έχει ως αποτέλεσμα, για $d = \left\lfloor \frac{n_l}{s} \right\rfloor \times \left\lfloor \frac{n_l}{s} \right\rfloor \times m_l$, το εξής:

$$\mathbf{Z}^{\mathrm{l}} * \mathbf{W} = \mathbf{Z}^{\mathrm{l}+1} \in \mathbb{R}^d$$



Σχήμα 25. Απεικόνιση της διαδικασίας της ομαδοποίησης. Με την σκιασμένη περιοχή του $\boldsymbol{Z}_l \in \mathbb{R}^{5\times5}$ απεικονίζονται τα παράθυρα $\boldsymbol{Z}_2^l(i+ts, j+ts) \in \mathbb{R}^{2\times2}$, όπου εφαρμόζουμε βηματισμό $s=2$, για τις τιμές του $0 \le t \le 1$, ενώ με σκούρο χρώμα απεικονίζονται οι τιμές που προκύπτουν στον πίνακα της συνέλιξης $\boldsymbol{Z}^{l+1} \in \mathbb{R}^{2\times2}$, μέσω του πίνακα βαρών $\mathbf{W} \in \mathbb{R}^{2\times2}$. *Από Zaki & Meira, 2020.*

2.2.2 Εκπαίδευση

Προτού προχωρήσουμε στην εμπρόσθια και την οπίσθια διάδοση, θα αναφέρουμε πρώτα την τιμή των νευρώνων στις συνελικτικές στοιβάδες του νευρωνικού δικτύου.

Συνέλιξη με ένα 3D φίλτρο
Όταν χρησιμοποιήσουμε ένα φίλτρο σε μία κρυφή στοιβάδα συνέλιξης, τότε είναι σαν να θεωρούμε πως θέλουμε η στοιβάδα να σαρώσει μικρά κομμάτια μία μεγαλύτερης εικόνας και να εξάγει *ένα χαρακτηριστικό* από αυτήν, για παράδειγμα την εξαγωγή αιχμών (Goodfellow, 2016).
Σύμφωνα με τους Zaki & Meira (2020), όπως και στα MLP νευρωνικά δίκτυα, έτσι και στα CNN κάθε στοιβάδα έχει τους δικούς της νευρώνες πόλωσης και συναρτήσεις ενεργοποίησης. Θεωρούμε τον τανυστή νευρώνων της κρυφής στοιβάδας l, ο οποίος έχει n_l το πλήθος γραμμές και στήλες, και m_l το πλήθος κανάλια, με τον τύπο:

$$\boldsymbol{Z}^l = \left(z_{i,j,q}^l : 1 \le i,j \le n_l \; και \; 1 \le q \le m_l\right) \in \mathbb{R}^{n_l \times n_l \times m_l}$$

όπου $z_{i,j,q}^l$ αναπαριστά την τιμή του νευρώνα στην γραμμή i, στήλη j και κανάλι q, του τανυστή $\boldsymbol{Z}^l$. Θεωρούμε επιπλέον το τρισδιάστατο (3D) φίλτρο (ή πυρήνα) $\mathbf{W} \in \mathbb{R}^{k \times k \times m_l}$, με την αντίστοιχη πόλωση $b \in \mathbb{R}$. Υπενθυμίζουμε πως η συνέλιξη $\left(\mathbf{Z}^l * \mathbf{W}\right) \in \mathbb{R}^{(n_l-k+1)\times(n_l-k+1)}$, είναι δηλαδή στις δύο διαστάσεις, δεν περιέχει τρίτη διάσταση ή κανάλι. Επίσης, υπενθυμίζουμε πως το παράθυρο $\boldsymbol{Z}_k^l(i,j) \in \mathbb{R}^{k\times k\times m_l}$ είναι ένα υποσύνολο του τανυστής $\boldsymbol{Z}^l$ στην θέση (i,j), φάρδους k και «βάθους» m_l. Τότε, σε αντιστοιχία με τα MLP, έχουμε πως η πληροφορία που εισέρχεται στον νευρώνα της επόμενης κρυφής στοιβάδας, $z_{i,j}^{l+1}$, δηλαδή η *καθαρή εισροή*, θα είναι:

$$net_{i,j}^{l+1} = \text{sum}\left(\boldsymbol{Z}_{\text{k}}^l(i,j) \; \odot \; \mathbf{W}\right) + b \in \mathbb{R}$$

Παρατηρούμε πως η καθαρή εισροή $net_{i,j}^{l+1}$ θα έχει δείκτες μόνο στα i,j, λόγω του αποτελέσματος της συνέλιξης που είναι μόνο σε δύο διαστάσεις. Η πληροφορία που εξάγεται από τον νευρώνα, θα είναι:

$$z_{i,j}^{l+1} = f\left(net_{i,j}^{l+1}\right) = f\left(\text{sum}\left(\boldsymbol{Z}_{\text{k}}^l(i,j) \; \odot \; \mathbf{W}\right) + b\right)$$

όπου f είναι μία συνάρτηση ενεργοποίησης της επιλογής μας. Εάν θεωρήσουμε ως $\oplus$ την πρόσθεση ανά στοιχείο διανύσματος, $\boldsymbol{x} \oplus \mathbf{1} = (x_1 + 1, \dots, x_n + 1)$, τότε για τους νευρώνες ολόκληρης της στοιβάδας, θα έχουμε πως (Zaki & Meira, 2020):

$$\mathbf{Z}^{l+1} = f\left((\mathbf{Z}^l * \mathbf{W}) \oplus b\right)$$

Η επιλογή της συνάρτησης ενεργοποίησης ως μία μη – γραμμική, σηματοδοτεί το στάδιο της *αναγνώρισης* σε μία στοιβάδα συνέλιξης, σύμφωνα με το πρώτο σχήμα που αναφέραμε σε αυτή την ενότητα. Επιπλέον, με την επιλογή ενός κοινού φίλτρου, μειώνουμε σημαντικά το πλήθος από βάρη που πρέπει να μάθει το CNN σε αυτή τη στοιβάδα, καθώς τα βάρη θα είναι *κοινά* για όλους τους νευρώνες που «σαρώνει» η συνέλιξη. Αυτό ονομάζεται *μοίρασμα παραμέτρων* (parameter sharing). Αυτό προσδίδει στο CNN την ιδιότητα της ισοδιακύμανσης, κατά την οποία δεν έχει σημασία πότε θα εφαρμοστεί μία συνάρτηση *μετάφρασης* (translation) στα δεδομένα, πριν ή μετά την συνέλιξη: θα έχει το ίδιο αποτέλεσμα. Αλλά αυτό δεν ισχύει για όλους τους μετασχηματισμούς (Goodfellow, 2016).

Συνέλιξη με πολλά 3D φίλτρα

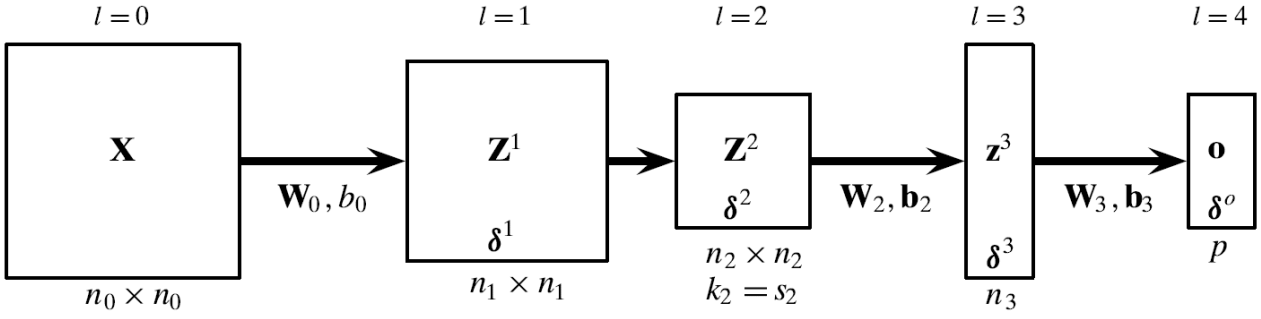
Εάν θέλουμε να σαρώσουμε μία εικόνα και να εξάγουμε αρκετές πληροφορίες, μπορούμε να χρησιμοποιήσουμε πολλά φίλτρα σε μία στοιβάδα συνέλιξης (Goodfellow, 2016). Σύμφωνα με τους Zaki & Meira (2020), η τιμή των κόμβων της επόμενης στοιβάδας θα είναι:

$$\boldsymbol{Z}^{l+1} = f\left((\boldsymbol{Z}^l * \boldsymbol{W}_1) \oplus b_1, (\boldsymbol{Z}^l * \boldsymbol{W}_2) \oplus b_2, \dots, \left(\boldsymbol{Z}^l * \boldsymbol{W}_{m_l+1}\right) \oplus b_{m_l+1}\right)$$

όπου με b_{m_l+1} συμβολίζουμε το βάρος που αντιστοιχεί στο φίλτρο $\boldsymbol{W}_{m_l+1}$ της στοιβάδας l και του καναλιού m_{l+1}. Χρησιμοποιούμε m_{l+1} διαφορετικά φίλτρα καθώς μόνο με αυτό τον τρόπο μπορούμε να έχουμε m_{l+1} διαφορετικά κανάλια στην στοιβάδα $l + 1$.

Συνολικά, στις στοιβάδες συνελίξεως χρησιμοποιούμε ως συνάρτηση συσσωμάτωσης το άθροισμα *sum*, και φίλτρα και όρους πόλωσης τα οποία θα μάθει ο αλγόριθμος της μάθησης. Επιπλέον, στην στοιβάδα συσσώρευσης μπορεί να χρησιμοποιηθεί πλήθος φίλτρων όσο και το πλήθος των καναλιών, ώστε η συνέλιξη να απεικονίζεται μόνο στις 2 διαστάσεις. Στις στοιβάδες συγκέντρωσης, χρησιμοποιούμε ως συνάρτηση συσσωμάτωσης το μέγιστο max, και ως βάρη τον σταθερό πίνακα τιμών $\mathbf{W} = \mathbf{1}_{k \times k \times 1}$. Ως συνέπεια, αφήνεται στον ερευνητή η επιλογή του μεγέθους του παραθύρου k σε κάθε στοιβάδα συνέλιξης και συγκέντρωσης, καθώς και η επιλογή του βήματος s. Επιπλέον, οφείλει να αποφασίσει εάν θα κάνει χρήση του παραγεμίσματος και να διαλέξει τις μη – γραμμικές συναρτήσεις ενεργοποίησης στις στοιβάδες συνελίξεως (Zaki & Meira, 2020). Δεν υπάρχει μία σίγουρη μεθοδολογία ακόμα για την επιλογή αυτών των υπερπαραμέτρων του συνελικτικού νευρωνικού δικτύου (Goodfellow, 2016).

Θεωρούμε την αρχιτεκτονική που αναγράφεται στο Σχήμα 26, η οποία περιγράφεται στη λεζάντα του, Με βάση αυτή την αρχιτεκτονική θα περιγράψουμε έπειτα την διαδικασία της μάθησης.



Σχήμα 26. Αρχιτεκτονική συνελικτικού νευρωνικού δικτύου. Έχουμε την στοιβάδα εισόδου ($l = 0$), μία στοιβάδα συνέλιξης ($l = 1$), μία στοιβάδα συγκέντρωσης ($l = 2$), μία στοιβάδα πλήρους σύνδεσης ($l = 3$) και τελικά την στοιβάδα εξόδου ($l = 4$), με τα αντίστοιχα φίλτρα $\boldsymbol{W}_i$ και τους όρους πόλωσης b_i. Επιπλέον, θεωρούμε τα εξής: το $\boldsymbol{X} \in \mathbb{R}^{n_0 \times n_0}$ έχει μόνο ένα κανάλι ($m = 1$), με πόλωση b_0 και φίλτρο $\boldsymbol{W}_0$ διαστάσεων $k_1 \times k_1$. Αυτά, θα έχουν ως αποτέλεσμα η συνέλιξή τους, $\boldsymbol{Z^1} \in \mathbb{R}^{n_1 \times n_1}$, όπου $n_1 = n_0 - k +$

1, με βηματισμό $s_1 = 1$. Θα έχει πόλωση $b_1 = 0$ και βάρος $\boldsymbol{W}_1 = \mathbf{1}_{k_1 \times k_1}$ (τα οποία δεν αναγράφονται στο σχήμα). Η στοιβάδα συγκεντρώσεως χρησιμοποιεί το μέγιστο ως συνάρτηση συσσωμάτωσης και βηματισμό $s = k_2$ όταν το φίλτρο $\boldsymbol{W}_2$ είναι διαστάσεων $k_2 \times k_2$. Τελικά, λόγο αυτών, το $\boldsymbol{Z}_2$ θα είναι διαστάσεων $n_2 \times n_2$. Επιπλέον, υποθέτουμε ότι $n_2 = n_1/k_1$. Τα n_2^2 στοιχεία του $\boldsymbol{Z}_2$ συνδέονται πλήρως με τα n_3 το πλήθος στοιχεία της στοιβάδας l=3, δηλαδή το φίλτρο $\boldsymbol{W}_2$ έχει διάσταση $n_2^2 \times n_3$. Τα n_3 στοιχεία συνδέονται πλήρως με τα p στοιχεία της στοιβάδας εξόδου, l=3, δηλαδή το $\boldsymbol{W}_3$ έχει διάσταση $n_3 \times p$. Οι όροι πόλωσης των στοιβάδων, έχουν μέγεθος ίσο με το μέγεθος των στοιβάδων, δηλαδή τα b_0, b_2, b_3 έχουν διάσταση n_0, n_2, n_3 αντίστοιχα. *Σχήμα από Zaki & Meira, 2020.*

Εμπρόσθια Διάδοση

Σε αντιστοιχία με τα νευρωνικά δίκτυα πολυεπίπεδων αντιλήπτρων, θεωρούμε το σύνολο των δεδομένων εκπαιδεύσεως $\mathbf{D} = \{\mathbf{X}_i, \boldsymbol{y}_i\}_{i=1}^n$ όπου $\mathbf{X}_i \in \mathbb{R}^{n_0 \times n_0 \times m_0}$ και $m_0 = 1$, και $\boldsymbol{y}_i \in \mathbb{R}^p$ τα αντίστοιχα διανύσματα απόκρισης (δηλαδή περιγράφουμε επιβλεπόμενη μάθηση). Με βάση τους Zaki & Meira (2020), δοθέντος σημείου $(\mathbf{X}, \mathbf{y}) \in \mathbf{D}$, έχουμε πως η απόκριση θα δίνεται από το ακόλουθο σύστημα:

$$\begin{aligned}
\mathbf{o} &= f^o(\mathbf{W}_o^T \mathbf{z}^3 + \mathbf{b}_o), \\
\mathbf{z}^3 &= f^3(\boldsymbol{W}_2^T \mathbf{z}^2 + \mathbf{b}_2), \\
\mathbf{Z}_2 &= \mathbf{Z}_1 *_{s_2,max} \mathbf{1}_{k_2 \times k_2}, \\
\mathbf{Z}_1 &= f^1\big((\mathbf{X} * \mathbf{W}_0) + b_0\big),
\end{aligned}$$

όπου $*_{s_2,max}$ αναπαριστά την συγκέντρωση με μέγιστο και βηματισμό $s = s_2$.

Οπίσθια Διάδοση

Θέλουμε να υπολογίσουμε τις σχέσεις:

$$\begin{aligned}
\mathbf{W}_l &= \mathbf{W}_l - \eta \cdot \nabla_{\mathbf{W}_l} \\
\mathrm{b}_l &= \mathrm{b}_l - \eta \cdot \nabla_{\mathbf{b}_l}
\end{aligned}$$

για $l = 4,3,2,1$, όπου με $l = 4$ ουσιαστικά συμβολίζουμε την στοιβάδα εξόδου. Σύμφωνα με τους Zaki & Meira (2020), δοθέντος της τιμής απόκρισης $\mathbf{o}$, και της συνάρτησης ελέγχου $E_{\boldsymbol{X}}$, υπολογίζουμε το διάνυσμα των κλίσεων δικτύου στην στοιβάδα εξόδου $\boldsymbol{\delta}^o$, ώστε να ανανεώσουμε τις τιμές από τα φίλτρα και τους όρους πόλωσης μέσω των της προς τα πίσω διάδοσης του σφάλματος στα διανύσματα των καθαρών κλίσεων των υπόλοιπων στοιβάδων, $\boldsymbol{\delta}^3, \boldsymbol{\delta}^2$ και $\boldsymbol{\delta}^1$. Μετά από μία σειρά πράξεων, προκύπτει:

$$\begin{aligned}
\boldsymbol{\delta}^o &= \partial \mathbf{f}^{\mathrm{o}} \odot \partial E_{\boldsymbol{X}} \\
\boldsymbol{\delta}^3 &= \partial \mathbf{f}^3 \odot (\mathbf{W}_0 \cdot \boldsymbol{\delta}^o) \\
\boldsymbol{\delta}^2 &= \partial \mathbf{f}^{\mathbf{2}} \odot (\mathbf{W}_2 \cdot \boldsymbol{\delta}^3) = \mathbf{W}_2 \cdot \boldsymbol{\delta}^3 \alpha
\end{aligned}$$

Όπου χρησιμοποιήσαμε ότι $\partial \mathbf{f}^{\mathbf{2}} = \mathbf{1}$. Με χρήση αυτών είναι εφικτός ο υπολογισμός των διαφορικών των φίλτρων και των όρων πόλωσης για τις στοιβάδες πέραν της $l = 1$. Για αυτήν, ισχύει πως:

$$\delta_{ij}^1 = \begin{cases} \delta_{ab}^2 \cdot \partial f_{ij}^1, & i = i^* \ \kappa\alpha\iota \ j = j^* \\ 0, & \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

Εδώ, έχουμε θέσει με i^* και j^* οι θέσεις από τους νευρώνες με την μεγαλύτερη τιμή στην στοιβάδα της ομαδοποίησης. Παρατηρούμε πως η τιμή της καθαρής κλίσης της στοιβάδας συνέλιξης θα είναι 0, εάν ο νευρώνας της στοιβάδας συνέλιξης δεν περιέχει στο παράθυρό του, που θα σαρώσει η συνάρτηση ομαδοποίησης, την μέγιστη τιμή. Με αυτό είμαστε έτοιμοι για τον υπολογισμό των τιμών των διαφορικών, οι οποίες προκύπτουν ως:

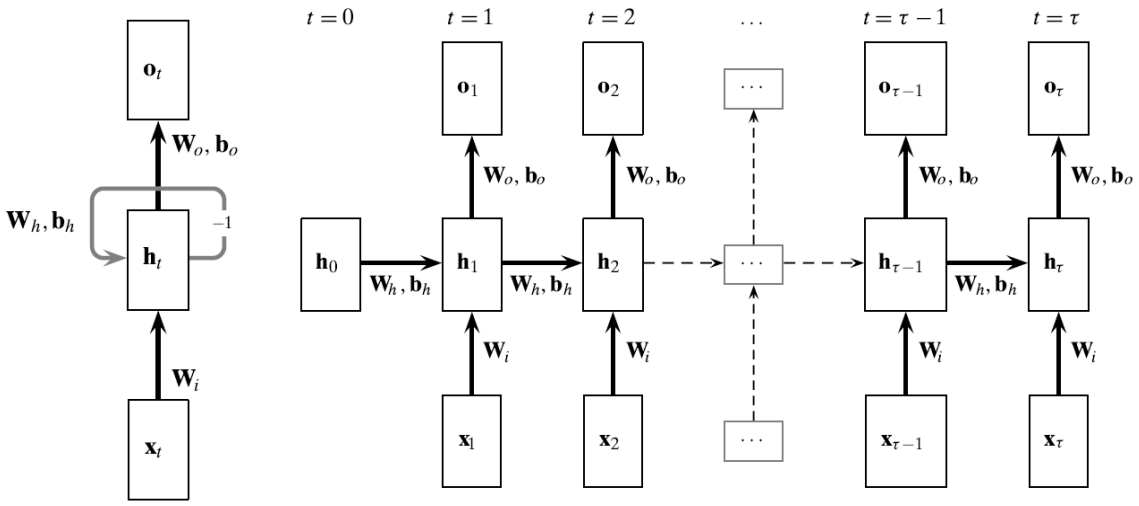
$$\begin{aligned}
\nabla_{\mathbf{W}_3} &= \mathbf{Z}^3 \cdot (\boldsymbol{\delta}^0)^T, \qquad \nabla_{\mathbf{b}_3} = \boldsymbol{\delta}^0 \\
\nabla_{\mathbf{W}_2} &= \mathbf{Z}^2 \cdot (\boldsymbol{\delta}^3)^T, \qquad \nabla_{\mathbf{b}_2} = \boldsymbol{\delta}^3
\end{aligned}$$

Το οποίο μας επιτρέπει να υπολογίσουμε τις κλίσεις στην στοιβάδα εισόδου:

$$\nabla_{\mathbf{W}_0} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} \mathbf{X}_{k_1}(i,j) \cdot \delta_{ij}^1, \qquad \nabla_{\mathbf{b}_0} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} \delta_{ij}^1$$

2.3.1 Αρχιτεκτονική

Για τα δεδομένα που ήταν συσχετισμένα στο επίπεδο, όπως οι φωτογραφίες, στραφήκαμε στα συνελικτικά νευρωνικά δίκτυα (CNN). Τώρα, για την μελέτη των συσχετισμένων δεδομένων σε μία διάσταση, θα αξιοποιήσουμε τα επαναλαμβανόμενα νευρωνικά δίκτυα. Τα *επαναλαμβανόμενα νευρωνικά δίκτυα* (Recurrent Neural Networks, RNN) είναι ικανά να διαχειριστούν δεδομένα χρονοσειρών, τα οποία δεν είναι ικανά να αναπαραστήσουν στα κυκλώματά τους τα υπόλοιπα νευρωνικά δίκτυα που έχουμε αναφέρει. Ένα παράδειγμα δεδομένων με συσχέτιση στον χρόνο είναι τα δεδομένα από λέξεις, καθώς η σειρά μίας λέξης σε μία πρόταση και η συχνότητα της χρήσης της, είναι άμεσα συσχετισμένη με τις υπόλοιπες πριν και μετά από αυτήν. Τα επαναλαμβανόμενα νευρωνικά δίκτυα, μέσω κατευθυνόμενων κύκλων (directed circles) δημιουργούν συνδέσεις μεταξύ των στοιβάδων τους, με χρήση της γνώσης από το μέλλον (ή και από το παρελθόν) ώστε να βελτιστοποιήσουν τις παραμέτρους τους, το οποίο αντιστοιχεί σε μάθηση χαρακτηριστικών μέσα από την «ανάμνηση» των προηγούμενων δεδομένων που εισάχθηκαν (Zhang et al. 2018). Επιπλέον, είναι δυνατό να κατασκευαστούν RNN τα οποία δέχονται μεταβαλλόμενο αριθμό δεδομένων και είναι ικανά να αντιμετωπίσουν πολύ μεγαλύτερες ακολουθίες από τιμές σε σύγκριση με τα MLP και τα CNN (Goodfellow, 2016).



Σχήμα 27. Απεικονίζεται η αρχιτεκτονική ενός RNN δύο φορές για καλύτερη κατανόηση.
Αριστερά, αυτή είναι «διπλωμένη» στον χρόνο, και αποτελεί την συχνή μορφή στην οποία απεικονίζονται τα RNN. Οι στοιβάδα εισόδου $\boldsymbol{x_t}$, η *κρυφή κατάσταση* (hidden state) $\boldsymbol{h_t}$ και η εξόδου $\boldsymbol{o_t}$ αλλάζουν για κάθε χρονικό βήμα t.
Δεξιά, είναι η ίδια αρχιτεκτονική όπως αριστερά, όμως είναι «ξεδιπλωμένη» στον χρόνο. Η διαφορά με τα κλασικά MLP νευρωνικά δίκτυα είναι πως κάθε κρυφή κατάσταση $\boldsymbol{h_t}$ πραγματοποιεί δύο ενέργειες: (α) παράγει το αποτέλεσμα εκείνη τη χρονική στιγμή $\boldsymbol{o_t}$ και (β) επηρεάζει την επόμενη κρυφή κατάσταση $\boldsymbol{h_{t+1}}$. Αυτό προφανώς δεν ισχύει για το βήμα $t = \tau$. Μπορούμε να παρατηρήσουμε το μοίρασμα των παραμέτρων (parameter sharing) καθώς οι πίνακες με τα βάρη είναι οι ίδιοι για τον κάθε νευρώνα της αντίστοιχης στοιβάδας εισόδου, κρυφής και εξόδου. *Σχήμα από Zaki & Meira, 2020.*

2.3.2 Εκπαίδευση

Εμπρόσθια διάδοση στον χρόνο

Ένα RNN έχει μία στοιβάδα εισόδου με διανύσματα τιμών τις $X = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_\tau),\ \boldsymbol{x}_t \in \mathbb{R}^d$, μία στοιβάδα εξόδου με διανύσματα τιμών $O = (\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_\tau),\ \boldsymbol{o}_t \in \mathbb{R}^p$, μία *κρυφή κατάσταση* (hidden state) με διανύσματα τιμών $S = (\boldsymbol{h}_0, \boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_\tau),\ \boldsymbol{h}_t \in \mathbb{R}^m$. Επιπλέον, θεωρούμε τα διανύσματα των πραγματικών τιμών απόκρισης, $Y = (\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_\tau),\ \boldsymbol{y}_t \in \mathbb{R}^p$. Τότε, σύμφωνα με τους Zaki & Meira (2020), ο σκοπός ενός RNN είναι η μάθηση μίας συνάρτησης η οποία προβλέπει τις τιμές στην στοιβάδα εξόδου Y, με βάση τις τιμές εισόδου X. Αλλά για την πρόβλεψη μίας απόκρισης y_t, την στιγμή t, η τιμή εισόδου $\boldsymbol{x}_t$ θα αξιοποιήσει επιπλέον και την στιγμή εισόδου $\boldsymbol{x}_{t-1}$ μέσω της τιμής στην κρυφή κατάσταση $\boldsymbol{h}_{t-1}$ (Zhang, 2018). Η τιμή της κρυφής στοιβάδας την χρονική στιγμή t, η οποία θα έχει συνάρτηση ενεργοποίησης f^h και πίνακα βαρών ίσο με $\boldsymbol{W}_h \in \mathbb{R}^{m \times m}$, θα είναι ίση με (Zaki & Meira, 2020):

$$\boldsymbol{h}_t = f^h(\boldsymbol{W}_i^T \boldsymbol{x}_t + \boldsymbol{W}_h^T \boldsymbol{h}_{t-1} + \boldsymbol{b}_t), \qquad t = 1, \dots, \tau$$

Στην παραπάνω εξίσωση έχουμε επιπλέον θεωρήσει ως $\boldsymbol{W_i} \in \mathbb{R}^{d \times n}$ τα βάρη μεταξύ της στοιβάδας εισόδου και των νευρώνων της κρυφής «στοιβάδας». Ως $\boldsymbol{b}_t$ είναι συμβολισμένα τα βάρη που σχετίζονται με τους νευρώνες στην κρυφή «στοιβάδα», ενώ μεταξύ της στοιβάδας εισόδου και της κρυφής δεν θεωρούμε πως υπάρχουν βάρη. Δοθέντος του $\boldsymbol{h}_t$, μπορούμε να υπολογίσουμε το διάνυσμα της απόκρισης $\boldsymbol{o}_t$ ως εξής, όπου με $\boldsymbol{W}_o \in \mathbb{R}^{m \times p}$ και με $\boldsymbol{b}_o$ συμβολίζουμε τον πίνακα με τα βάρη και το διάνυσμα πόλωσης, μεταξύ της κρυφής στοιβάδας και της στοιβάδας εξόδου:

$$\boldsymbol{o}_t = f^o(\boldsymbol{W}_o^T \boldsymbol{h}_t + \boldsymbol{b}_o), \qquad t = 1, \dots, \tau$$

Υπάρχει και στα RNN, όπως στα CNN, η επιλογή της κοινής χρήσης παραμέτρων. Για παράδειγμα, οι πίνακες με τα βάρη $\boldsymbol{W}_i,\ \boldsymbol{W}_h$ και $\boldsymbol{W}_o$ είναι κοινοί για όλους τους νευρώνες στις αντίστοιχες στοιβάδες. Αυτό μπορούμε να το δούμε στο ακόλουθο σχήμα. Με αντικατάσταση της $\boldsymbol{h}_t$ στην $\boldsymbol{o}_t$, σύμφωνα με τους Zaki & Meira (2020) καταλήγουμε ότι:

$$\boldsymbol{o}_t = f^o(\boldsymbol{W}_o^T f^h(\boldsymbol{W}_i^T \boldsymbol{x}_t + \boldsymbol{W}_h^T f^h(\cdots f^h(\boldsymbol{W}_i^T \boldsymbol{x}_1 + \boldsymbol{W}_h^T \boldsymbol{h}_0 + \boldsymbol{b}_h) + \cdots) + \boldsymbol{b}_h) + \boldsymbol{b}_o)$$

Παρατηρούμε πως, όπως αναμέναμε, η τιμή στον νευρώνα εξόδου την χρονική στιγμή t εξαρτάται από όλους τους νευρώνες εισόδου $\boldsymbol{x}_1, \dots, \boldsymbol{x}_t$, αλλά όχι σε μελλοντικούς νευρώνες.

Οπίσθια διάδοση στο χρόνο

Δοθέντος της ακολουθίας των τιμών εξόδου $\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_\tau$ είμαστε σε θέση να συνεχίσουμε την διαδικασία της οπίσθιας διάδοσης. Επιπλέον, εν συντομία παρουσιάζουμε τις τιμές των καθαρών κλίσεων (net gradients) για κάθε νευρώνα στην στοιβάδα εξόδου, δ_t^o, και για κάθε νευρώνα στην κρυφή κατάσταση, δ_h^o.

$$\delta_t^o = \left(\frac{\partial E_{\boldsymbol{x}_t}}{\partial net_{t1}^o}, \dots, \frac{\partial E_{\boldsymbol{x}_t}}{\partial net_{tm}^o}\right)^T, \qquad \delta_t^h = \left(\frac{\partial E_{\boldsymbol{x}_t}}{\partial net_{t1}^h}, \dots, \frac{\partial E_{\boldsymbol{x}_t}}{\partial net_{tm}^h}\right)^T$$

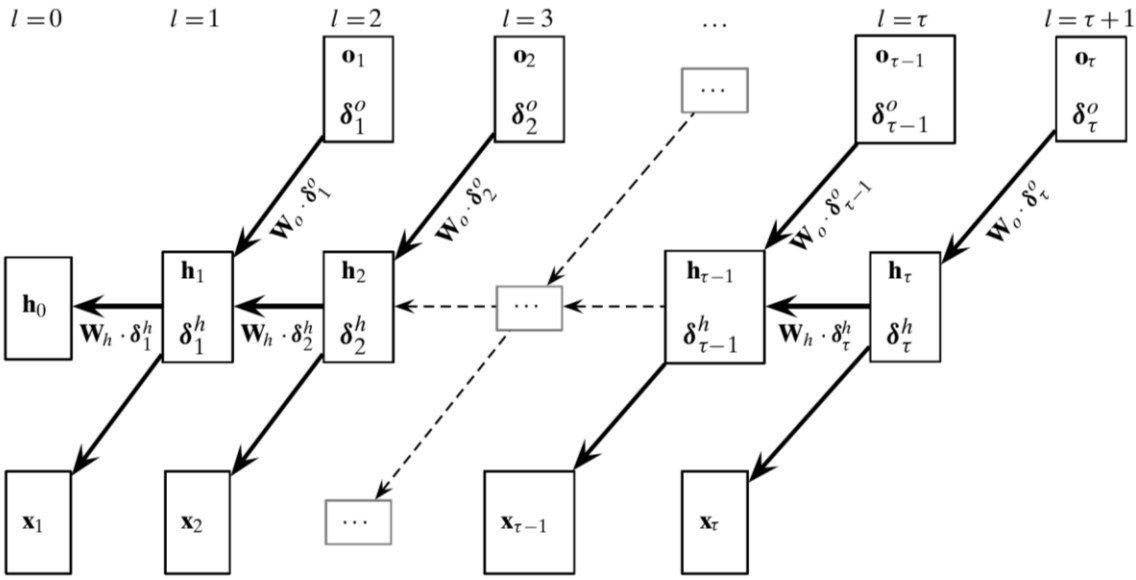
Είναι σημαντικό ότι, λόγο του μοιράσματος των παραμέτρων από κάθε βήμα, οι κλίσεις λαμβάνονται ως αθροίσματα των κλίσεων από κάθε χρονικό βήμα t. Τελικά, οι παράμετροι θα ανανεωθούν με βάση τις παρακάτω εξισώσεις, για την στοιβάδα εισόδου, την κρυφή και την εξόδου αντίστοιχα.

$$\boldsymbol{W}_i = \boldsymbol{W}_i - \eta \cdot \nabla_{\boldsymbol{W}_i} = \boldsymbol{W}_i - \eta \cdot \left(\sum_{t=1}^{\tau} \boldsymbol{x}_t \cdot \left(\delta_t^h\right)^T\right)$$

$$\boldsymbol{W}_h = \boldsymbol{W}_h - \eta \cdot \nabla_{\boldsymbol{W}_h} = \boldsymbol{W}_h - \eta \cdot \left(\sum_{t=1}^{\tau} \boldsymbol{h}_{t-1} \cdot \left(\delta_t^h\right)^T\right),\ \ \boldsymbol{b}_h = \boldsymbol{b}_h - \eta \cdot \nabla_{\boldsymbol{b}_h} = \boldsymbol{b}_h - \eta \cdot \left(\sum_{t=1}^{\tau} \delta_t^h\right)$$

$$\boldsymbol{W}_o = \boldsymbol{W}_o - \eta \cdot \nabla_{\boldsymbol{W}_o} = \boldsymbol{W}_o - \eta \cdot \left(\sum_{t=1}^{\tau} \boldsymbol{h}_t \cdot (\delta_t^o)^T\right), \qquad \boldsymbol{b}_o = \boldsymbol{b}_o - \eta \cdot \nabla_{\boldsymbol{b}_o} = \boldsymbol{b}_o - \eta \cdot \left(\sum_{t=1}^{\tau} \delta_t^o\right)$$

Η διαδικασία που ακολουθούμε για κάθε βήμα l είναι ορατή στο επόμενο σχήμα.



Σχήμα 28. Απεικονίζεται η διαδικασία της οπίσθιας διάδοσης σε ένα RNN, όπου αριθμούμε τα «βήματα» του αλγόριθμου στα αντίστοιχα βήματα l. Είναι και πάλι ορατό πως υπάρχουν κοινοί πίνακες από βάρη μεταξύ των νευρώνων εισόδου, των κρυφών καταστάσεων και των νευρώνων εξόδου. *Σχήμα από Zaki & Meira, 2020.*

2.3.3 Παραλλαγές των RNN

Η δομή του επαναλαμβανόμενου νευρωνικού δικτύου που αναφέραμε στην προηγούμενη ενότητα, έχει την πλέον βασική μορφή ενός RNN, ενώ συνηθίζεται να αποκαλείται *vanilla* RNN. Έχουν προταθεί πλήθος παραλλαγών σε αυτή την δομή, τις οποίες θα αναφέρουμε εν συντομία. Επιπλέον, δεν υπάρχει περιορισμός στους ερευνητές ως προς τους τρόπους που αυτοί μπορούν να συνδυάσουν τις παρακάτω τεχνικές, ή το πλήθος των κρυφών καταστάσεων. Αυτές οι εναλλακτικές δομές προέκυψαν για την βελτίωση των RNN καθώς και επειδή αυτά υπόκεινται εντονότερα στα προβλήματα των εξαφανιζόμενων και εκρηγνυόμενων κλίσεων κατά την εκπαίδευση (Goodfellow, 2016).

***Βαθιά RNN* (Deep RNN, DRNN)**
Είναι δυνατό να θεωρήσουμε ως την στοιβάδα εισόδου σε ένα RNN την στοιβάδα εξόδου από ένα άλλο. Με αυτό τον τρόπο, μπορούμε να δημιουργήσουμε βαθιά RNN (Schmidhuber, 1992). Επιπλέον, οι Pascanu et al. (2013) αναφέρουν πως η έννοια του «βάθους» είναι διαφορετική στα RNN, προτείνοντας μία εναλλακτική ερμηνεία των βαθιών RNN, με βάση (α) της συνάρτησης μεταξύ των δεδομένων εισόδου και των κρυφών καταστάσεων (β) των μεταβιβάσεων μεταξύ των κρυφών καταστάσεων (γ) των συναρτήσεων μεταξύ των κρυφών καταστάσεων και των νευρώνων εξόδου. Σύγχρονες τεχνικές ξεπερνούν τις τελευταίας τεχνολογίας επιδόσεις, και αφορούν την επιλογή του βάθους των DRNN μέσω γενετικών αλγορίθμων, καθώς η επιλογή της σωστής δομής από άνθρωπο απαιτεί εις βάθος γνώση του αντικειμένου και του προβλήματος (Viswambaran et al. 2020).

***RNN διπλής κατεύθυνσης* (bidirectional recurrent neural network, BRNN)**
Μία κρυφή κατάσταση στα RNN δεχόταν επιρροή μόνο από την προηγούμενή της, δηλαδή συνολικά οι κρυφές καταστάσεις δεχόντουσαν επιρροή από όλες τις παρελθοντικές κρυφές καταστάσεις. Στα RNN διπλής κατεύθυνσης, η κρυφή κατάσταση σε μία δεδομένη χρονική στιγμή δέχεται επιρροή από τις παρελθοντικές κρυφές καταστάσεις και επιπλέον από τις μελλοντικές κρυφές καταστάσεις, δηλαδή εξαρτάται από τις μελλοντικές τιμές όσο και από τις παρελθοντικές τιμές (Schuster & Paliwal, 1997). Ουσιαστικά είναι ένας συνδυασμός δύο RNN με ξεχωριστές κρυφές στοιβάδες, που αποφασίζουν για τις

αποκρίσεις από κοινού (Goodfellow, 2016). Παλιότερες εφαρμογές τους, αφορούσαν την ανάλυση συναισθήματος από μεγάλα δεδομένα, συγκεκριμένα μηνύματα στην ιστοσελίδα του Twitter, όπου εκπαιδεύτηκαν BRNN με χαρακτηριστικά που επιλέχτηκαν από έναν συνδυασμό εξελικτικών αλγορίθμων (Shazly et al. 2020). Σύγχρονες μελέτες αφορούν την παραγωγή περιγραφών για εικόνες (Thomas & Pillai, 2018).

Δίκτυα Μακράς Βραχύχρονης Μνήμης **(Long Short – term memory RNN, LSTM)**
Επιπλέον, είναι δυνατό να εισάγουμε επιπρόσθετες εργασίες στις κρυφές καταστάσεις, με σκοπό την βελτίωση της ικανότητας του RNN να αντιμετωπίζει το πρόβλημα των εξαφανιζόμενων και εκρηγνυόμενων κλίσεων. Αυτές περιλαμβάνουν τα *φραγμένα νευρωνικά δίκτυα* (gated NN), τα οποία αξιοποιούν φράγματα ή πύλες (gates) σε διάφορα σημεία του νευρωνικού δικτύου (Goodfellow, 2016).
Μία υποκατηγορία των φραγμένων νευρωνικών δικτύων, είναι τα δίκτυα μακράς βραχύχρονης μνήμης (LSTM) (Hochreiter & Schmidhuber, 2017) και είναι τα πιο πετυχημένα επαναλαμβανόμενα νευρωνικά δίκτυα της εποχής μας (Goodfellow, 2016). Αυτά περιλαμβάνουν συνήθως τρείς πύλες: Μία πύλη *εισόδου* (input gate), μία πύλη διανύσματος *απώλειας μνήμης* (forget gates) και μία πύλη στο διάνυσμα της εξόδου (output gate) (Zaki & Meira, 2020). Ουσιαστικά επιτρέπουν στο μοντέλο να θυμάται μεγάλες αλληλουχίες ενώ του επιτρέπεται να ξεχνάει κάποιες από αυτές σε βάθος χρόνου, με την δυνατότητα να τις χρησιμοποιήσει πάλι στην πρόβλεψη. Σύμφωνα με τους Greff et al (2016) τα καθοριστικά χαρακτηριστικά που οφείλουν την επιτυχία τους τα LSTM είναι η πύλη απώλειας μνήμης και η συνάρτηση στην στοιβάδα εξόδου, ενώ επιπλέον αναφέρουν πως στην έρευνά τους καμία άλλη παραλλαγή των LSTM δίνει σημαντική διαφορά στα αποτελέσματα. Η επίδραση του παραγεμίσματος στα LSTM και στα CNN απεικονίζεται συνοπτικά στο άρθρο των Dwarampudi & Reddy (2019).

Ενότητα 4.
Ιδιαιτερότητες στην Εκπαίδευση

Σε αυτή την ενότητα θα αναφέρουμε κάποιες τεχνικές οι οποίες μπορούν να βελτιώσουν την εκπαίδευση των νευρωνικών δικτύων, καθώς και κάποια προβλήματα που εμφανίζονται κατά την εκπαίδευση. Γενικά, αυτά που περιγράφονται μπορούν να αξιοποιηθούν σε οποιαδήποτε δομή νευρωνικού δικτύου, ενώ υπάρχουν έρευνες για συγκεκριμένες δομές.

Βελτιώσεις επίδοσης
Α) Προεπεξεργασία δεδομένων
Αναφέρεται πως η προεπεξεργασία των δεδομένων εκπαίδευσης βοηθάει στην σύγκλιση της εκπαίδευσης, συγκεκριμένα οι διαδικασίες κανονικοποίησης Ελαχίστου-Μεγίστου (Min-Max), όπου η ελάχιστη τιμή γίνεται 0, η μέγιστη τιμή γίνεται 1 και οι υπόλοιπες τιμές είναι δεκαδικοί, η κανονικοποίηση (Z-Score Normalization) κατά την οποία πραγματοποιούμε στα δεδομένα τον μετασχηματισμό $\frac{x-\mu}{\sigma}$, όπου μ είναι η μέση τιμή τους και σ είναι η τυπική απόκλιση, ενώ επιπλέον υπάρχει και η κανονικοποίηση δεκαδικής κλίμακας (Decimal Scaling Normalization), κατά την οποία μετασχηματίζουμε όλες τις τιμές μέσω του τύπου $x_i' = \frac{x_i}{10^j}$, φέρνοντάς τες στο ίδιο δεκαδικό σημείο (Nawi et al, 2013).

Β) Χρήση προ – εκπαιδευμένων μοντέλων νευρωνικών δικτύων
Ας υποθέσουμε πως θέλουμε να εκτελέσουμε μία διεργασία σε κατηγοριοποίηση εικόνας. αλλά έχουμε στην διάθεσή μας ένα ήδη εκπαιδευμένο μοντέλο νευρωνικών δικτύων (προ – εκπαιδευμένο, pre – trained). Τότε, μπορούμε να πάρουμε τα βάρη από τις πρώτες στοιβάδες του εκπαιδευμένου νευρωνικού δικτύου

και να τα αξιοποιήσουμε στο νέο νευρωνικό δίκτυο για την κατηγοριοποίηση εικόνας. Αυτό, αποτελεί μία εφαρμογή της *μάθησης μεταβίβασης* (transfer learning) (Geron, 2019).

Γ) Χρήση στοιβάδων απόσυρσης

Σε μία στοιβάδα απόσυρσης (dropout), τυχαία διαλέγονται κάποιοι νευρώνες από τις στοιβάδες εισόδου και τις κρυφές, οι οποίοι προσωρινά δεν συνεισφέρουν στην εκπαίδευση για την συγκεκριμένη εποχή (Geron, 2019). Πρώτη φορά τους πρότειναν οι Srivastava et al (2014).

Το πρόβλημα των Εξαφανιζόμενων & Εκρηγνυόμενων Κλίσεων στα MLP

Τα βαθιά νευρωνικά δίκτυα πολυεπίπεδων αντιλήπτρων (MLP) υπόκεινται στο πρόβλημα των *εξαφανιζόμενων κλίσεων* (vanishing gradients) και των *εκρηγνυόμενων κλίσεων* (exploding gradients), τα οποία μειώνουν εξαιρετικά το «βάθος» στο οποίο μπορούν να εκπαιδευτούν (Philipp et al. 2017). Αυτά, αναφέρονται στην εκθετική της μείωση ή αύξηση της κλίσης της συνάρτησης απώλειας καθώς αυτή *διαδίδεται* προς την στοιβάδα εισόδου. Εάν η κλίση γίνει πολύ μικρή στις στοιβάδες κοντά στην στοιβάδα εξόδου, στις πρώτες εποχές της μάθησης, τότε καθώς θα διαδίδεται προς τα πίσω η επίδρασή της θα γίνεται ολοένα και μικρότερη, με αποτέλεσμα να μην επιδρά σχεδόν καθόλου στα βάρη των πρώτων στοιβάδων. Αντίθετα, εάν η κλίση γίνει υπερβολικά μεγάλη, τότε τα βάρη μπορεί να γίνονται ολοένα και μεγαλύτερα καθώς θα διαδίδεται προς τα πίσω, με αποτέλεσμα την απόκλιση από την πραγματική λύση (Geron, 2019).

Α) Εξαφανιζόμενες Κλίσεις

Σε όσα ακολουθούν αναφέρουμε κάποιες μεθόδους που έχουν προταθεί για την αντιμετώπιση των εξαφανιζόμενων κλίσεων στα βαθιά MLP.

Συχνές προσεγγίσεις περιορισμού του προβλήματος αποτελούν οι συναρτήσεις ενεργοποίησης ReLU (Nair & Hinton, 2010) και η *παρτίδα κανονικοποίησης* (batch normalization) (Basodi et al. 2020), που πρότειναν οι Ioffe & Szegedy το 2015, ξεπερνώντας την ικανότητα των ανθρώπων στην κατηγοριοποίηση των εικόνων στο σύνολο δεδομένων ImageNet. Η παρτίδα κανονικοποίησης, σε κάθε εποχή της εκπαίδευσης *κανονικοποιεί* την στοιβάδα εισόδου, με αποτέλεσμα η κανονικότητα να γίνεται *μέρος της αρχιτεκτονικής του νευρωνικού δικτύου*, διατηρώντας κοινή την κατανομή των δεδομένων που εισέρχονται σε κάθε στοιβάδα, με αποτέλεσμα να μην χρειάζεται να την μάθει το νευρωνικό δίκτυο, το οποίο οδηγεί σε αυξημένους ρυθμούς μάθησης (Ioffe & Szegedy, 2015). Περισσότερες πληροφορίες μπορούν να βρεθούν στο άρθρο των Bjorck et al (2018).

Οι εξαφανιζόμενες κλίσεις είναι γνωστό πως συνδέονται με την χρήση των συναρτήσεων ενεργοποίησης όπως η σιγμοειδής, στα βαθιά MLP νευρωνικά δίκτυα (Roodschild et al 2020). Αυτό οδήγησε τους Roodschild et al (2020) στην πρόταση *μίας νέας μεθόδου οπίσθιας διάδοσης* κατά την οποία προστίθεται μία μικρή σταθερά στον υπολογισμό της παραγώγου της σιγμοειδής συνάρτησης. Αναφέρουν πως οδηγεί σε παρόμοια ακρίβεια του μοντέλου, με λιγότερες εποχές, ενώ επιτρέπει την μάθηση δύο ακόμη κρυφών στοιβάδων στα νευρωνικά δίκτυα εμπρόσθιας τροφοδότησης, δηλαδή «παραβλέπει» για δύο ακόμη στοιβάδες τις εξαφανιζόμενες κλίσεις.

Το 2021, οι Ko & Ko πρότειναν μία νέα μέθοδο για την αντιμετώπιση του προβλήματος των εξαφανιζόμενων κλίσεων στα βαθιά MLP, μέσω του ορισμού μίας παραμετρικής συνάρτηση ενεργοποίησης, η οποία *παρακάμπτει την ανάγκη για την ελαχιστοποίηση της συνάρτησης απώλειας μέσω της οπίσθιας διάδοσης* (Ko & Ko, 2021), ενώ το 2020 οι Basodi et al πρότειναν μία *μέθοδο ενίσχυσης της κλίσης* (gradient amplification) κατά την οποία αντιμετωπίζουν τις μικρές τιμές της κλίσης οι οποίες τελικά οδηγούν σε εξαφανιζόμενες κλίσεις, μέσω της στρατηγικής ενίσχυσης της κλίσης, σε συγκεκριμένες μόνο εποχές της εκπαίδευσης (Basodi et al, 2020).

Β) Εκρηγνυόμενες Κλίσεις στα MLP

Επιπλέον, είναι γνωστό στην κοινότητα πως οι εκρηγνυόμενες κλίσεις μπορούν να «τιθασευτούν» μέσω τεχνικών όπως οι ακόλουθες:

1. Η εισαγωγή στοιβάδων κανονικοποίησης (regularization layers) στα MLP, όπου οι τιμές που εισέρχονται, εξέρχονται με έναν μετασχηματισμό κανονικοποίησης (Geron, 2019), το οποίο

αναφέρεται ως *κανονικοποίηση παρτίδας* (batch normalization) όταν αναφερόμαστε στην στοιβάδα εισόδου, την οποία έννοια εισήγαγαν οι Ioffe & Szegedy το 2015.

2. Η αξιοποίηση των τεχνικών αρχικοποίησης των βαρών όπως αυτή των Glorot (αφορά την σιγμοειδή συνάρτηση ενεργοποίησης) και του He (Geron, 2019).
3. Η επιλογή εναλλακτικών συναρτήσεων ενεργοποίησης, όπως η Adam και η SELU (Scaled Exponential Unit), η οποία αποτελεί μία παραλλαγή της ELU (Philipp et al. 2017).
4. Η τεχνική περικοπής των κλίσεων (gradient clipping), όπου εάν η τιμή των κλίσεων ξεπεράσει ένα όριο, τότε μπορούμε είτε να παραλείψουμε τον όρο προς την κατεύθυνση που μεγαλώνει εξαιρετικά η κλίση, αλλά αυτό μπορεί να αλλάξει την φορά της κλίσης. Μία άλλη μέθοδος, είναι να διαιρέσουμε το διάνυσμα της κλίσης με την νόρμα της, με αποτέλεσμα της διατήρησης της φοράς της κλίσης (Geron, 2019).

Όμως, ο Hanin (2018) αναφέρει πως ακόμη και σε ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο με συνάρτηση ενεργοποίησης ReLU και τυχαία επιλογή στα βάρη κατά την εκκίνηση της εκπαίδευσης, θα υπάρχουν κλίσεις που προκύπτουν από την οπίσθια διάδοση, οι οποίες θα έχουν μεγάλη διασπορά στις τιμές τους εάν μία σταθερά β, η οποία προκύπτει από την αρχιτεκτονική του νευρωνικού δικτύου, έχει μεγάλη τιμή.

Επιπλέον, οι Philipp et al. (2017) προτείνουν πως αυτές οι τεχνικές αντιμετωπίζουν το πρόβλημα ως έναν μετασχηματισμών των τιμών από τις κλίσεις στις στοιβάδες των νευρωνικών δικτύων, ενώ αποτυγχάνουν να αντιληφθούν την πηγή που δημιούργησε τις εκρηγνυόμενες κλίσεις στα βαθιά MLP νευρωνικά δίκτυα. Επιπλέον, αναφέρουν πως οι δομές των βαθιών νευρωνικών δικτύων που δέχονται αυτές τις τεχνικές για την αντιμετώπιση του προβλήματος υπόκεινται στο «πρόβλημα του τομέα που καταρρέει» (collapsing domain problem), το οποίο υποσκάπτει την διαδικασία της εκπαίδευσης. Για την αντιμετώπιση του προβλήματος, προτείνουν τη χρήση *νευρωνικών δικτύων με κατάλοιπα* (Residual Neural Networks, ResNets) τα οποία απλοποιούν με την παράληψη συνδέσεων (skip connections) (Philipp et al. 2017).

Επιπλέον, στα CNN έγινε έρευνα πως η κανονικοποίηση καναλιών βοηθάει στην αποφυγή των εξαφανιζόμενων κλίσεων (Dai & Heckel, 2019)

Κεφάλαιο 3.
Εφαρμογές Νευρωνικών Δικτύων στην R.

Ενότητα 1.
Εισαγωγή

Σε αυτό το κεφάλαιο θα εξερευνήσουμε τις εφαρμογές των νευρωνικών δικτύων μέσω της γλώσσας προγραμματισμού R. Συγκεκριμένα για την μοντελοποίηση και την εκπαίδευση των νευρωνικών δικτύων θα χρησιμοποιήσουμε το πακέτο `keras`. Το πακέτο `keras` αποτελεί ένα *περιβάλλον εφαρμογών προγραμματισμού* (Application Programming Interface, API) στα νευρωνικά δίκτυα, το οποίο επιτρέπει εύκολο πειραματισμό σε δομές νευρωνικών δικτύων, ενώ επιτρέπει την επιλογή των διεργασιών να πραγματοποιηθούν είτε μέσω CPU είτε μέσω GPU από τον υπολογιστή. Αρχικά το πακέτο `keras` ήταν ένα API που αφορούσε μόνο την γλώσσα προγραμματισμού της Python (Chollet, 2015). Δύο χρόνια μετά, έγινε διαθέσιμο και στην R μέσω του πακέτου `keras` (Allaire, 2017). Για τις πράξεις μεταξύ των τανυστών (γενικευμένα διανύσματα) οι οποίες συμβαίνουν στις στοιβάδες των νευρωνικών δικτύων, θα χρησιμοποιήσουμε το περιβάλλον (back-end) του TensorFlow. Συγκεκριμένα, χρησιμοποιούμε την ενημερωμένη έκδοση 2.7 (Chollet & Allaire, 2021) με την χρήση GPU, ενώ επιπλέον αξιοποιείται η βιβλιοθήκη βαθιών νευρωνικών δικτύων cuDNN (Chetlur et al. 2014), έκδοσης 8.1 και η εργαλειοθήκη CUDA έκδοσης 11.2.0 (Vingelmann, 2021). Για τις ιδιαιτερότητες της έκδοσης 2.7 του keras στην R γίνεται αναφορά από τους Keydana & Kalinowski (2021).

Η χρήση της *μονάδας επεξεργασίας γραφικών* (Graphical Processing Unit, GPU) έναντι της *κεντρικής μονάδας επεξεργασίας* (Central Processing Unit, CPU) έχει αποδειχθεί πως επιταχύνει την διαδικασία της μάθησης στα νευρωνικά δίκτυα, με τις πρώτες εφαρμογές που εκμεταλλεύτηκαν τους παράλληλους υπολογισμούς της κάρτας γραφικών να αναφέρουν βελτίωση έως και 20 φορές με χρήση πολλαπλασιασμών πινάκων (Oh & Jung, 2004). Μεταγενέστερες εφαρμογές (Guzhva et al. 2009) αξιοποίησαν την τεχνολογία NVIDIA CUDA για την ταυτόχρονη μάθηση αντιλήπτρων (perceptrons) στην διαδικασία μάθησης με *οπίσθια διάδοση*, ενώ αναφέρουν πως αυτό οδήγησε σε 50 φορές μεγαλύτερη ταχύτητα σε σύγκριση με την χρήση του ταχύτερου CPU της εποχής. Σύγχρονες εφαρμογές αποσκοπούν στην περαιτέρω μείωση της ενέργειας που καταναλώνουν τα νευρωνικά δίκτυα, μέσω χρήσης επιταχυντών νευρωνικών δικτύων με πηγές ανοιχτού κώδικα (open-source code), συγκεκριμένα με το πρόγραμμα NVDLA της NVIDIA, ενώ βρέθηκε πως οι επιδόσεις του ξεπερνάνε αυτές από τους GPU και CPU επεξεργαστές (Noskova et al. 2022).

Όσα ακολουθούν, αφορούν μία εισαγωγή στα νευρωνικά δίκτυα με το πακέτο `keras` στην γλώσσα προγραμματισμού R, για την έκδοση TensorFlow 2.7, καθώς αυτή έχει εκφέρει αλλαγές στην υποκείμενη γλώσσα Python (Keydana & Kalinowski, 2021). Συνεπώς οι εφαρμογές στην R είναι δυσεύρετες, με αποτέλεσμα να οδηγείται ο χρήστης σε ξεπερασμένες εκδόσεις του TensorFlow 1.x, όπου φυσικά υπάρχει πλήθος βιβλιογραφίας και εφαρμογών, τις οποίες συνοψίζει στο βιβλίο των Chollet & Allaire, *Deep Learning with R* (2018), ο οποίος μετέφερε το πακέτο `keras` στην R. Μέσα από το βιβλίο αυτό βασίζονται κάποια από τα επόμενα παραδείγματα.

Ενότητα 2.
Κατηγοριοποίηση εικόνας σε MLP

3.2.1 Συλλογή και Αποθήκευση Δεδομένων

Τα δεδομένα που θα χρησιμοποιήσουμε σε αυτό το παράδειγμα αφορούν την απλούστερη περίπτωση, όπου μπορούμε να τα λάβουμε μόνο με μία εντολή, μέσω του πακέτου `keras`. Είναι το σύνολο δεδομένων `fashion_mnist` (Xiao et al. 2017) και αφορά μία παραλλαγή στο γνωστό σύνολο `mnist` (Deng, 2012) το οποίο η κοινότητα αποκαλεί το `hello world` της κατηγοριοποίησης εικόνας μέσω νευρωνικών δικτύων. Το σύνολο `fashion_mnist` αποτελείται από εικόνες 28x28 στην κλίμακα του γκρι, ενώ έχει 10 κατηγορίες από ρουχισμό και ενδύματα. Περιέχονται 60.000 εικόνες στα δεδομένα εκπαίδευσης και 10.000 εικόνες στα δεδομένα ελέγχου. Δημιουργούμε το σύνολο δεδομένων μέσω της εντολής `dataset_fashion_mnist()` και αντιστοιχούμε τα δεδομένα εκπαίδευσης και τα δεδομένα ελέγχου από τα επόμενα:

```
library(keras)
fashion_mnist <- dataset_fashion_mnist()
c(c(train_images, train_labels),
  c(test_images, test_labels)) %<-% fashion_mnist
```

Το σύμβολο `%<-%` προέρχεται από το πακέτο `zealot` (Teetor & Teetor, 2018) και αφορά την πολλαπλή αντιστοίχιση τιμών. Αυτό περιλαμβάνεται στο πακέτο `keras`. Αν όλα είναι επιτυχή, σε αυτό το σημείο θα έχουμε μόνο το εξής μήνυμα από την κονσόλα της R:

```
Loaded Tensorflow version 2.7.0
```

Δίνοντας στην κονσόλα την εντολή `str(train_images)` περιμένουμε να μας εμφανίσει την διάσταση των εικόνων εκπαίδευσης. Δίνει ως αποτέλεσμα:

```
int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
```

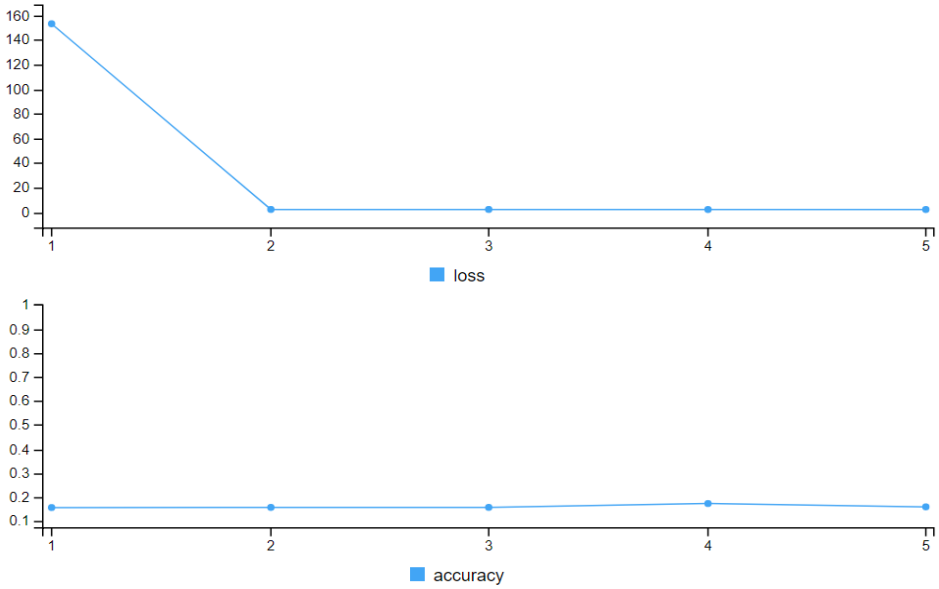
Συνεπώς όντως το σύνολο εκπαίδευσης περιέχει 60.000 εικόνες 28x28 pixel, με τιμές στους φυσικούς αριθμούς (*int* από *integer*), ενώ αυτές οι τιμές ανήκουν στο σύνολο [0,255]. Σε αυτό το σημείο να επισημάνουμε πως πολύχρωμη εικόνα, όπως θα δούμε στα επόμενα παραδείγματα, θα είχε επίσης μία ακόμα διάσταση με μέγεθος 3. Αυτό συμβαίνει γιατί οι εικόνες με χρώμα έχουν 3 κανάλια, όπου το κάθε ένα παίρνει τιμές από το 0 έως το 255. Οι εικόνες του συνόλου `fashion_mnist()` από την άλλη είναι στην κλίμακα του γκρι και έχουν προεπεξεργαστεί ώστε να μην έχουν άλλα κανάλια. Ως συνέπεια, μία εικόνα στην περίπτωση του `fashion_mnist()` είναι ήδη σε ψηφιακή μορφή. Συνεπώς, αν αντιστοιχίσουμε την ένταση με την τιμή από το 0 έως το 255, μπορούμε να δούμε το σχήμα της εικόνας:

```
library(tidyr)
library(ggplot2)
image_1 <- as.data.frame(train_images[1, , ])
colnames(image_1) <- seq_len(ncol(image_1))
image_1$y <- seq_len(nrow(image_1))
image_1 <- gather(image_1, "x", "pixel_value", -y)
image_1$x <- as.integer(image_1$x)
par(mfrow = c(3,3))
ggplot(image_1, aes(x = x, y = y, fill = pixel_value)) +
    geom_tile() +
    scale_fill_gradient(low = "white", high = "black")+
    scale_y_reverse() +
    theme_minimal() +
    theme(panel.grid = element_blank()) +
    theme(aspect.ratio = 1) +
    xlab("") +
    ylab("") +
    theme(axis.text.x=element_blank(),
          axis.ticks.x=element_blank(),
          axis.text.y=element_blank(),
          axis.ticks.y=element_blank())
```

Το οποίο επιστρέφει το αποτέλεσμα:

3.2.2 Προεπεξεργασία των Δεδομένων

Σε αυτό το σημείο, θα ξεκινήσουμε την προεπεξεργασία των δεδομένων ώστε αυτά να μπορούν να τροφοδοτηθούν στο νευρωνικό δίκτυο. Είναι συνήθης τακτική οι τιμές που εισάγονται στο νευρωνικό δίκτυο να ανήκουν σε ένα μικρό εύρος τιμών όπως το $[0,1]$, γιατί έτσι μία (μεγάλη) τιμή όπως το 255, που στην περίπτωσή μας αναφέρει πως το pixel της εικόνας θα είναι μαύρο, θα οδηγήσει σε πολύ μεγάλες τιμές κλίσης κατά την οπίσθια διάδοση, με πιθανά προβλήματα όπως αυτά των εκρηγνυόμενων κλίσεων. Για παράδειγμα, στο νευρωνικό δίκτυο που θα παρουσιάσουμε στην επόμενη ενότητα, εάν δεν πραγματοποιήσουμε τον μετασχηματισμό των δεδομένων $f: [0,255] \rightarrow [0,1]$, η ακρίβεια που θα πετύχει αν χρησιμοποιήσουμε ελαχιστοποίηση με SGD θα είναι, σταθερά, *10%*. Δηλαδή θα βρίσκει 1 σωστό αποτέλεσμα στα 10, ακριβώς ότι δεν θέλουμε από ένα μοντέλο πρόβλεψης. Αυτό φαίνεται στο επόμενο σχήμα. *Συγκεντρωτικά τα μοντέλα είναι στο τέλος του κεφαλαίου.*



Σχήμα 29. Τα διαγράμματα απώλειας και της μετρικής που παράγονται κατά την μάθηση. Στο συγκεκριμένο παράδειγμα δεν πραγματοποιήθηκε η μετατροπή $f: [0,255] \rightarrow [0,1]$, και σε αυτό οφείλεται η εξαιρετικά χαμηλή τιμή ακρίβειας του μοντέλου (0.1). Στον άξονα x υπάρχουν οι *εποχές* της εκπαίδευσης, ενώ στον άξονα y υπάρχουν οι τιμές (κάτω διάγραμμα) της ακρίβειας και οι τιμές (πάνω διάγραμμα) της κατηγορικής σταυρωτής εντροπίας. Ως αλγόριθμος βελτιστοποίησης χρησιμοποιήθηκε η SGD.

Συνεπώς, αυτός είναι ένας απλός λόγος που θέλουμε να φέρουμε τις τιμές εισόδου σε ένα σύνολο γύρω από το 0. Στο δικό μας πρόβλημα, θα μετασχηματίσουμε τις τιμές ώστε με 0 να είναι λευκό και με 1 να

είναι μαύρο. Επιπλέον, θα αλλάξουμε τις διαστάσεις τους ώστε να μπορούν να εισαχθούν στο νευρωνικό δίκτυο. Αυτά γίνονται με τις παρακάτω εντολές:

```
train_images <- array_reshape(train_images, c(60000, 28*28))
test_images <- array_reshape(test_images, c(10000, 28*28))
train_images <- train_images/255
test_images <- test_images/255
train_labels <- to_categorical(train_labels)
test_labels <- to_categorical(test_labels)
```

Η εντολή `to_categorical()` ουσιαστικά πραγματοποιεί την διαδικασία του *one – hot encoding*, όπου για παράδειγμα η πρώτη κατηγορία από τις δέκα, όπου πριν αντιστοιχούσε στον αριθμό $c_1 = 0$, αντιστοιχίζεται στο διάνυσμα $c_1 = (1,0,0,0,0,0,0,0,0,0)$, η δεύτερη κατηγορία από τις δέκα, η οποία αντιστοιχούσε στον αριθμό $c_2 = 1$, αντιστοιχίζεται στο διάνυσμα $c_2 = (0,1,0,0,0,0,0,0,0,0)$, και ούτω καθεξής, μέχρι την δέκατη κλάση. Η εντολή `to_categorical()` μας δημιουργεί 10 κλάσεις που επισημαίνονται με έναν αριθμό, για παράδειγμα η πρώτη κλάση αντιστοιχεί στο 0, η δεύτερη κλάση στο 1, η τρίτη στο 2, ενώ η δέκατη αντιστοιχεί στο 9. Διαπιστώνουμε πως πραγματοποιήθηκε σωστά η διαδικασία μέσω των εντολών `str(train_images)` και `str(train_labels)`, οι οποίες επιστρέφουν:

```
int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
num [1:60000, 1:10] 0 1 1 0 1 0 0 0 0 0 ...
```

Είμαστε πλέον έτοιμοι να προχωρήσουμε στην μοντελοποίηση του νευρωνικού δικτύου.

3.2.3 Μοντελοποίηση Νευρωνικού Δικτύου

Η μοντελοποίηση ενός νευρωνικού δικτύου με το πακέτο `keras` αποτελείται από δύο βήματα. Αρχικά, τοποθετούμε σε σειρά τις στοιβάδες με τις συναρτήσεις ενεργοποίησης και τους νευρώνες που θέλουμε, ενώ αυτό πραγματοποιείται κυρίως με συναρτήσεις που ακολουθούν την `keras_model_sequential()`, ενώ υπάρχει ξεχωριστή διαδικασία σε περίπτωση που ο χρήστης θέλει να χρησιμοποιήσει δικές του στοιβάδες, όπου οι νευρώνες συνδέονται με όποιον τρόπο αυτός επιθυμεί. Έπειτα, περνάμε στο μοντέλο τις υπερπαραμέτρους που απαιτεί η μάθηση και η αποτίμηση του μοντέλου πρόβλεψης, δηλαδή την συνάρτηση απώλειας, τον αλγόριθμο βελτιστοποίησης και την μετρική. Αυτές δίνονται ως παράμετροι στην συνάρτηση `compile()`. Για παράδειγμα, ας θεωρήσουμε το MLP νευρωνικό δίκτυο, με μία στοιβάδα εισόδου, μία κρυφή στοιβάδα και μία στοιβάδα εξόδου, Η στοιβάδα εισόδου θα πρέπει να δέχεται $28 * 28 = 784$ τιμές, συνεπώς θα έχει τόσους κόμβους εισόδου. Θα είναι πλήρως συνδεδεμένη με την κρυφή στοιβάδα, η οποία ας υποθέσουμε πως έχει πλήθος κόμβων μία δύναμη του 2, έστω $2^9 = 512$ κόμβους, και έστω πως έχει την συνάρτηση ενεργοποίησης ReLU. Στην στοιβάδα εξόδου, εφόσον έχουμε πρόβλημα κατηγοριοποίησης με δέκα κλάσεις, θα θεωρήσουμε δέκα νευρώνες, και συνάρτηση ενεργοποίησης την κατηγορική σταυρωτή εντροπία για πολλές κλάσεις. Αυτά γίνονται με τις ακόλουθες εντολές.

```
model <- keras_model_sequential() %>%
    layer_dense(units = 512, activation = 'relu', input_shape = c(28*28)) %>%
    layer_dense(units = 10, activation = 'softmax')

model %>% compile(
    optimizer = 'sgd',
    loss = 'categorical_crossentropy',
    metrics = c('accuracy')
)
```

Επιπλέον, θα αναφερθούμε την χρήση του *pipe operator* `%>%`, ο οποίος είναι μέρος του πακέτου `magrittr` (Bache et al. 2017). Ο *pipe operator* `%>%` αντικαταστεί αυτό που βρίσκεται αριστερά του ως στοιχείο της συνάρτησης που βρίσκεται δεξιά του, ταυτόχρονα ανανεώνοντας την τιμή του. Δηλαδή, η εντολή `x %>% f` είναι ισοδύναμη με την `f(x)` (Bache & Wickham, 2019).

3.2.4 Εκπαίδευση και Επικύρωση Νευρωνικού Δικτύου

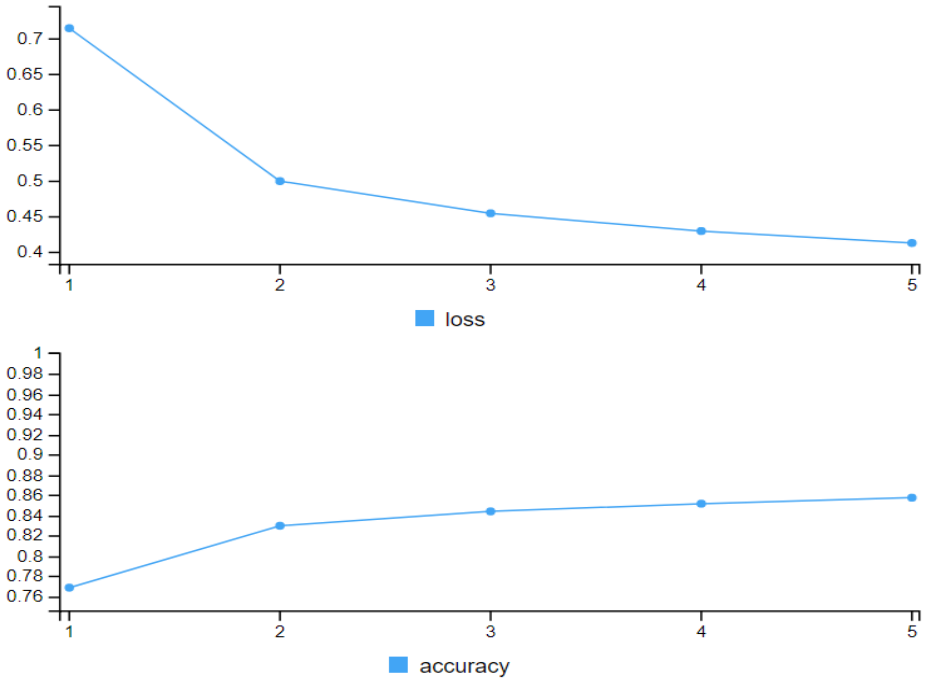
Σε αυτό το σημείο, έχουμε λάβει όλες τις ενέργειες ώστε να είναι δυνατή η εκκίνηση της εκπαίδευσης. Ας υποθέσουμε πως θέλουμε να πραγματοποιηθούν 5 εποχές εκπαίδευσης, με παρτίδες από 32 εικόνες την φορά. Αυτό σημαίνει πως τα βάρη του νευρωνικού δικτύου δεν θα ανανεώνονται μετά από την κάθε μία εικόνα, αλλά πως με τα ίδια βάρη θα γίνεται ταυτόχρονα ο υπολογισμός της συνάρτησης κόστους σε 32 εικόνες, και τότε η ανανέωση των βαρών του νευρωνικού δικτύου θα γίνεται με βάρη την μέση τιμή της συνάρτησης κόστους. Συνεπώς, αναμένουμε πως ο αλγόριθμος για κάθε εποχή της εκπαίδευσης θα πραγματοποιήσει $1875 = \frac{60.0000}{32} = \frac{Δεδομένα\ Εκπαίδευσης}{Μέγεθος\ Παρτίδας}$ ανανεώσεις στα βάρη. Εάν δεν προσδιορίσουμε το μέγεθος βήματος, τότε επιλέγεται αυτόματα μία τιμή στις δυνάμεις του 2 καθώς αυτό επιτρέπει καλύτερα την αξιοποίηση των παράλληλων υπολογισμών στις κάρτες γραφικών NVIDIA (Vanhoucke et al. 2011) ενώ με παρτίδες μικρότερες ή ίσες του $32 = 2^5$ έχουν παρατηρηθεί καλά αποτελέσματα (Masters & Luschi, 2018). Για να υλοποιηθούν αυτά, αρκεί η εντολή:

```
model %>% fit(train_images, train_labels, epochs = 5, batch_size = 32)
```

η οποία θα ξεκινήσει μία διαδικασία η οποία απεικονίζει τις τιμές καθώς ανανεώνονται τα βάρη στην κονσόλα και έπειτα τις απεικονίζει σε ένα διαδραστικό γράφημα στην καρτέλα "Viewer" στο Rstudio. Η κονσόλα θα έχει μορφή με τιμές κοντά στις παρακάτω:

```
Epoch 1/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.7132 - accuracy: 0.7687
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.4993 - accuracy: 0.8298
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.4543 - accuracy: 0.8440
Epoch 4/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.4295 - accuracy: 0.8516
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.4128 - accuracy: 0.8575
```

Ενώ το διάγραμμα στον *Viewer*, με την ακρίβεια και την τιμή της συνάρτησης απώλειας, σε αυτή την περίπτωση είναι το ακόλουθο:



Τελικά, πετύχαμε την πέμπτη εποχή ακρίβεια ίση με 85.75%, ενώ η τιμή της συνάρτησης απώλειας είναι 0.4128. Καθόλου άσχημα μόνο για πέντε εποχές εκπαίδευσης, με ένα τόσο απλό νευρωνικό δίκτυο. Επιπλέον, με την συνάρτηση `evaluate()` θα διαπιστώσουμε την εγκυρότητα του μοντέλου στα δεδομένα ελέγχου:

```
model %>% evaluate(test_images, test_labels)
```

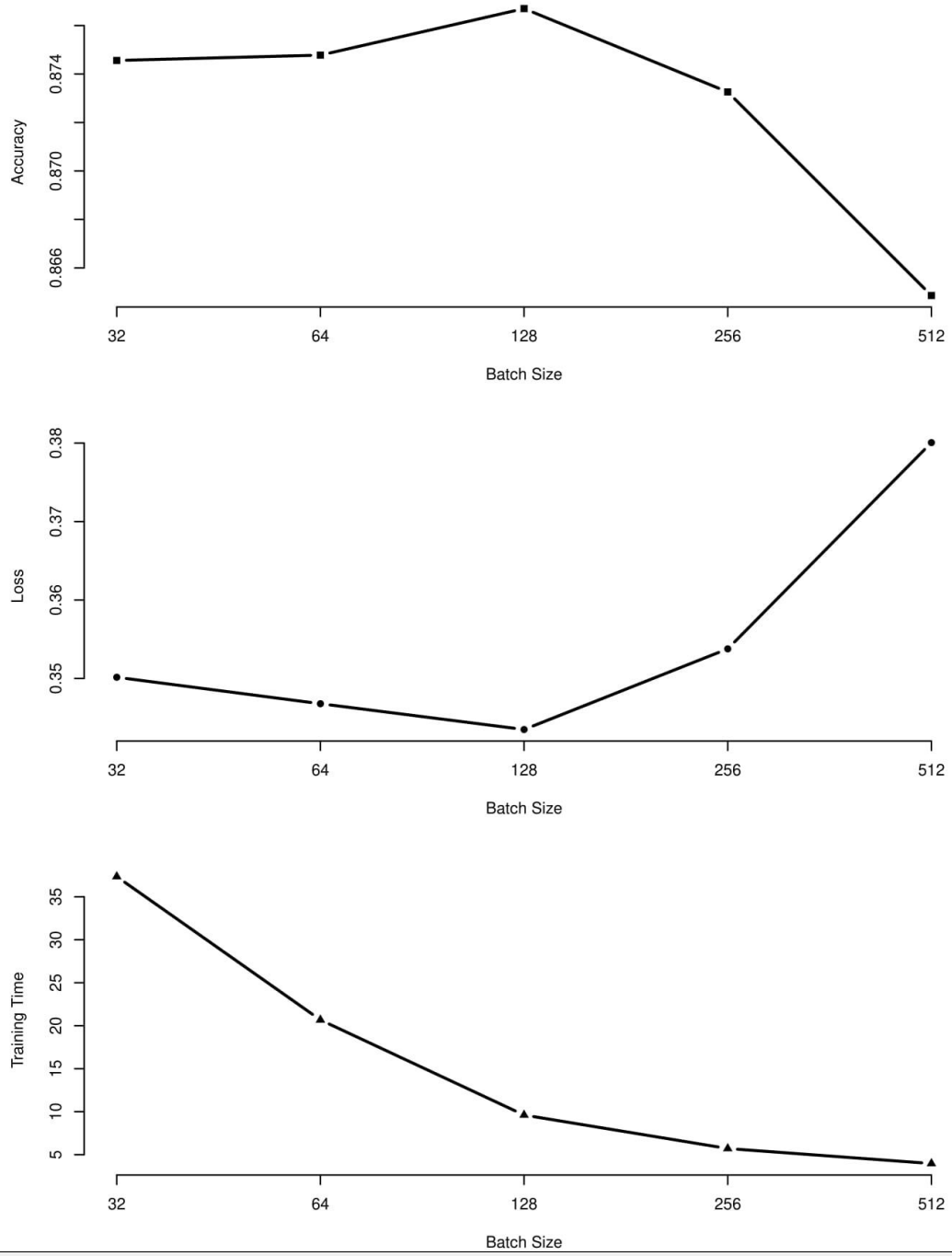
Το οποίο επιστρέφει:

```
313/313 - 1s - loss: 0.4412 - accuracy: 0.8430 - 1s/epoch - 3ms/step
```

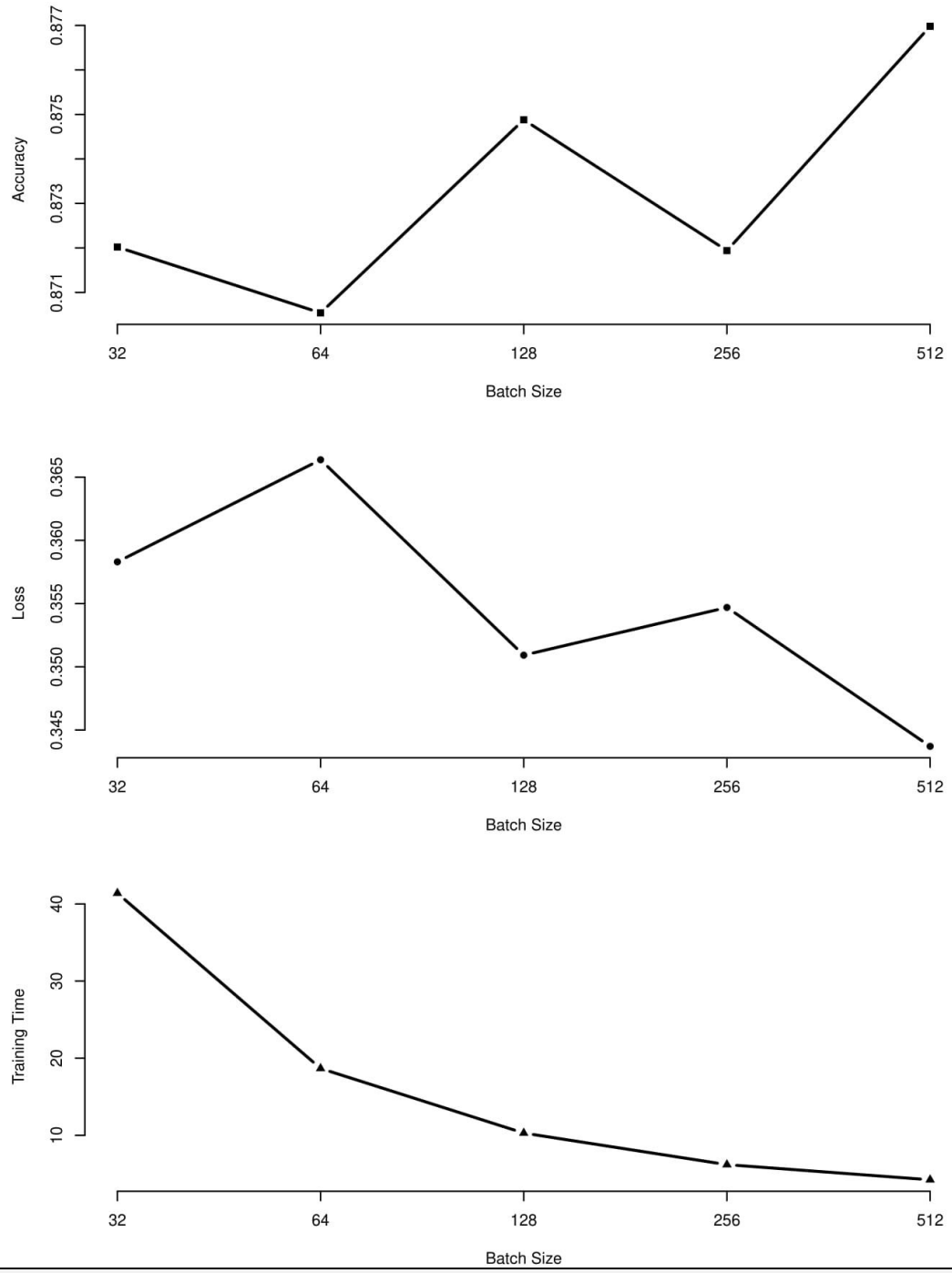
Δηλαδή, είχε επιτυχία 84.3%. Επίσης, εάν χρησιμοποιήθηκε GPU, η κονσόλα ενδέχεται να βγάλει το παρακάτω προειδοποιητικό:

```
WARNING : tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time
(batch time: 0.0038s vs `on_train_batch_end` time: 0.0045s). Check your callbacks.
```

Το προειδοποιητικό της κονσόλας αναφέρεται στο ότι δεν αξιοποιήσαμε πλήρως τις δυνατότητες της ταχύτητας που προσφέρει η παράλληλη επεξεργασίας του GPU, μπορούμε δηλαδή να το αποφύγουμε εάν χρησιμοποιήσουμε μεγαλύτερο μέγεθος παρτίδας, 2^N, $N > 5$, $N \in \mathbb{N}$. Αλλά με μεγαλύτερο μέγεθος παρτίδας θα έχουμε λιγότερες ανανεώσεις των βαρών μέσω της οπίσθιας διάδοσης : συνεπώς αναμένεται πως θα έχουμε μείωση της απόδοσης του μοντέλου για το ίδιο πλήθος εποχών. Αυτά φαίνονται συνοπτικά στον επόμενο πίνακα, όπου θεωρήσαμε τον μέσο όρο από 10 επαναλήψεις του μοντέλου, με 5 εποχές εκπαίδευσης. Μετά, ακολουθεί αντίστοιχο σχήμα για το ίδιο μοντέλο, αλλά με 10 εποχές εκπαίδευσης. Έπειτα, συνοψίζονται τα αποτελέσματα στον *Πίνακα 4*.



Σχήμα 30. Αφορά τις μέσες τιμές από 5 επαναλήψεις, για το *Μοντέλο 1,* το οποίο έχει μία κρυφή στοιβάδα με 512 νευρώνες. Χρησιμοποιήθηκε ο αλγόριθμος βελτιστοποίησης ADAM. Παρατηρούμε πως η παρτίδα με μέγεθος $128 = 2^7$ δίνει την καλύτερη ακρίβεια ως προς τα δεδομένα ελέγχου του συνόλου, καθώς και την μικρότερη απώλεια.



Σχήμα 31. Αφορά τις μέσες τιμές από 5 επαναλήψεις, για το *Μοντέλο 2*, το οποίο έχει δύο κρυφές στοιβάδες: την πρώτη με $512 = 2^9$ νευρώνες και την δεύτερη με $256 = 2^8$. Χρησιμοποιήθηκε ο αλγόριθμος βελτιστοποίησης ADAM. Παρατηρούμε πως η παρτίδα με μέγεθος $512 = 2^9$ δίνει την καλύτερη ακρίβεια ως προς τα δεδομένα ελέγχου του συνόλου, καθώς και την μικρότερη απώλεια. Αυτό ενδέχεται να συμβαίνει διότι το *Μοντέλο 2* είχε περισσότερες παραμέτρους ως προς μάθηση, συνεπώς με τα μικρότερα μεγέθη των παρτίδων επήλθε γρηγορότερα η υπερπροσαρμογή του μοντέλου στα δεδομένα εκπαίδευσης, με αποτέλεσμα μικρότερη απόδοση στο σύνολο ελέγχου.

MLP Νευρωνικό Δίκτυο (1)	**Μέγεθος Παρτίδας**	**Μέση Ακρίβεια**	**Μέση Τιμή Συνάρτησης Απώλειας**	**Μέσος Χρόνος Εκπαίδευσης (Δευτερόλεπτα)**
Μία κρυφή στοιβάδα με 512 κόμβους. 407.050 παραμέτρους.	32	0.87456	0.3501550	37.343588
	64	0.87478	0.3467780	20.676972
	128	**0.87670**	**0.3434892**	**9.613424**
	256	0.87326	0.3537729	5.713548
	512	0.86486	0.3800647	3.971018
MLP Νευρωνικό Δίκτυο (2)	**Μέγεθος Παρτίδας**	**Μέση Ακρίβεια**	**Μέση Τιμή Συνάρτησης Απώλειας**	**Μέσος Χρόνος Εκπαίδευσης (Δευτερόλεπτα)**
Δύο κρυφές στοιβάδες με 512 και 256 κόμβους. 535,818 παραμέτρους.	32	0.87202	0.3583018	41.391279
	64	0.87054	0.3663741	18.669194
	128	0.87488	0.3509189	10.291016
	256	0.87194	0.3546943	6.194819
	512	**0.87698**	**0.3437101**	**4.233850**

Πίνακας 4. Σύγκριση μοντέλων

Τέλος, θα θεωρήσουμε το καλύτερο μοντέλο (με βάση τα παραπάνω αποτελέσματα) και θα απεικονίσουμε τα αποτελέσματα που είχαμε σωστά και που είχαμε λάθος. Δίνουμε στην R τον κώδικα:

```
model <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = 'relu', input_shape = c(28*28)) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')
model %>% compile(optimizer = 'adam', loss = 'categorical_crossentropy',
  metrics = c('accuracy'))
model %>% fit(train_images, train_labels,
              epochs = 10, verbose = 2,
              batch_size = 512)
predictions <- model %>% predict(test_images)

class_names = c('T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat',
                'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Boot')
pdf(file = "ch1_bestmodel2_true_false.pdf", width = 13, height = 10)
par(mfcol=c(10,10), mar=c(0, 0, 1.2, 0))

for (i in 1:100) {
    img <- matrix(test_images[i, ], ncol = 28, nrow = 28)
    img <- t(apply(img, 2, rev))
    img <- t(apply(img, 2, rev))
    predicted_label <- which.max(predictions[i, ])
    true_label <- which.max(test_labels[i,])
    if (predicted_label == true_label) {
        color <- '#008800'
    } else {
        color <- '#bb0000'
    }
    image(1:28, 1:28, img, col = gray(1-0:255/255), xaxt = 'n', yaxt = 'n',
          main = paste0(class_names[predicted_label], " (",
                        class_names[true_label], ")"), col.main = color)
}
dev.off()
```

Αυτό αποθηκεύει στον υπολογιστή μας το αρχείο *ch1_bestmodel2_true_false.pdf*, το οποίο περιέχει το σχήμα της επόμενης σελίδας.

Σχήμα 32. Τα 100 πρώτα στοιχεία του συνόλου ελέγχου από το `fashion_dist`. Πάνω από κάθε εικόνα είναι με πράσινο χρώμα είναι οι επιτυχείς προβλέψεις, ενώ με κόκκινο είναι οι αποτυχίες, ενώ η αριστερή λέξη είναι η πρόβλεψη της κατηγορίας από το νευρωνικό δίκτυο και η δεξιά, σε παρενθέσεις, είναι η πραγματική. Δηλαδή την εικόνα κάτω δεξιά την πρόβλεψε σωστά, ως ένα πουλόβερ, ενώ την αριστερή εικόνα από αυτή την πρόβλεψε λάθος ως πουλόβερ (Pullover) ενώ πραγματικά είναι φούστα (skirt).

Ενότητα 3.
Κατηγοριοποίηση εικόνας σε CNN

3.3.1 Συλλογή και Αποθήκευση Δεδομένων

Τα δεδομένα που θα επεξεργαστούμε σε αυτή την ενότητα δεν θα είναι προ – επεξεργασμένα όπως αυτά που περιέχει το πακέτο `keras`, αλλά αποτελούν πραγματικά δεδομένα που σύλλεξε η Microsoft Research, για την διεξαγωγή του διαγωνισμού «Dogs vs. Cats» στο kaggle το 2014. Τα δεδομένα μπορούν να κατεβούν από την ακόλουθη ιστοσελίδα, αλλά απαιτείται πρώτιστος η δημιουργία ενός δωρεάν λογαριασμού: https://www.kaggle.com/c/dogs-vs-cats/data.

Ο κώδικας που ακολουθεί, μπορεί να αναπαρασταθεί πλήρως εάν η εξαγωγή των δύο φακέλων *train* και *test* που περιέχουν τα δεδομένα εκπαίδευσης και ελέγχου, αντίστοιχα, από την ιστοσελίδα του kaggle γίνει σε *έναν* φάκελο που ονομάζεται *kaggle_data*, ο οποίος είναι τοποθετημένος στον προκαθορισμένο χώρο που διαβάζει αρχεία η R (εάν δεν έχει αλλάξει, αυτός ο χώρος είναι τα *Έγγραφα (Documents)*, σε έναν υπολογιστή με Windows 10).

Συνολικά, στους φακέλους *train* και *test* μέσα στον *kaggle_data,* υπάρχουν 12.500 εικόνες έκαστος. Με τις ακόλουθες γραμμές κώδικα, θα διαβάσουμε και θα αντιγράψουμε ένα υποσύνολο των εικόνων αυτών, τις οποίες θα τοποθετήσουμε σε ξεχωριστούς φακέλους που θα δημιουργήσουμε μέσα στον φάκελο *subset_small*. Συγκεκριμένα, θα δημιουργήσουμε 3 φακέλους μέσα στον φάκελο *subset_small*: τους *train, validation* και *test*. Κάθε ένας από αυτούς τους φακέλους, θα περιέχει 2 ακόμα φακέλους, *cats* και *dogs*, οι οποίοι θα περιέχουν τις εικόνες που θα αντιγράψουμε. Ο φάκελος *training* θα έχει συνολικά 1000 σκύλους και 1000 γάτες, ο φάκελος *validation* θα έχει συνολικά 500 σκύλους και 500 γάτες, ομοίως και ο φάκελος *test*. Όλα αυτά γίνονται αυτόματα με τις παρακάτω εντολές:

```
# Making the folders
original_dataset_dir <- "~/kaggle_data"
original_train_dir <- file.path(original_dataset_dir, "train")
original_test_dir <- file.path(original_dataset_dir, "test")
base_dir <- "~/subset_small"
dir.create(base_dir)
train_dir <- file.path(base_dir, "train")
dir.create(train_dir)
validation_dir <- file.path(base_dir, 'validation')
dir.create(validation_dir)
test_dir <- file.path(base_dir, 'test')

dir.create(test_dir)
train_cats_dir <- file.path(train_dir, "cats")
dir.create(train_cats_dir)
train_dogs_dir <- file.path(train_dir, "dogs")
dir.create(train_dogs_dir)
validation_cats_dir <- file.path(validation_dir, "cats")
dir.create(validation_cats_dir)
validation_dogs_dir <- file.path(validation_dir, "dogs")
dir.create(validation_dogs_dir)
test_cats_dir <- file.path(test_dir, "cats")
dir.create(test_cats_dir)
test_dogs_dir <- file.path(test_dir, "dogs")
dir.create(test_dogs_dir)

# Copying the data
fnames <- paste0("cat.", 1:1000, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(train_cats_dir))
fnames <- paste0("cat.", 1001:1500, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(validation_cats_dir))

fnames <- paste0("cat.", 1501:2000, ".jpg")
```

```
file.copy(file.path(original_train_dir, fnames),
          file.path(test_cats_dir))
fnames <- paste0("dog.", 1:1000, ".jpg")
file.copy(file.path(original train dir, fnames),
          file.path(train_dogs_dir))
fnames <- paste0("dog.", 1001:1500, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(validation_dogs_dir))
fnames <- paste0("dog.", 1501:2000, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(test_dogs_dir))
```

Κυρίως έχει αξιοποιηθεί η εντολή `dir.create()` η οποία δημιουργεί έναν φάκελο στην «διεύθυνση» που θα δώσουμε μέσω του directory. Για παράδειγμα, το directory `"~/subset_small"` αντιστοιχεί στον φάκελο που θέλουμε να αποθηκεύσουμε τους φακέλους με τις εικόνες, και η εντολή `dir.create("~/subset_small")` θα δημιουργήσει αυτόν τον φάκελο.

Ως αποτέλεσμα, τώρα έχουμε δημιουργήσει αρκετούς φακέλους με δεδομένα που θα χρησιμοποιήσουμε. Αυτή η μέθοδος διαχωρισμού των δεδομένων έχει διπλό σκοπό: πρώτων, με το μικρότερο σύνολο δεδομένων είναι εφικτό ακόμη και για κάποιους με CPU να ακολουθήσουν το παράδειγμα δίχως μεγάλο πρόβλημα. Επιπλέον, μετά από αυτό το παράδειγμα θα αυξήσουμε τον όγκο των δεδομένων και θα δοκιμάσουμε το ίδιο μοντέλο, ώστε να συγκρίνουμε την ικανότητα μάθησης του νευρωνικού δικτύου και πώς αυτό μαθαίνει. Με την ακόλουθη εντολή, μπορούμε να γνωρίζουμε εάν ολοκληρώσαμε με επιτυχία την διαδικασία.

```
for (c in c(train cats dir, train dogs dir,
            validation cats dir, validation dogs dir,
            test_cats_dir, test_dogs_dir)){
    cat('\n', as.character(c), "has ", length(list.files(c)), "files in total.")}
```

Τα αποτελέσματα θα πρέπει να είναι όπως τα ακόλουθα:

```
~/subset_small/train/cats has  1000 files in total.
 ~/subset_small/train/dogs has  1000 files in total.
 ~/subset_small/validation/cats has  500 files in total.
 ~/subset_small/validation/dogs has  500 files in total.
 ~/subset_small/test/cats has  500 files in total.
 ~/subset_small/test/dogs has  500 files in total.
```

Είμαστε έτοιμοι να περάσουμε στην προεπεξεργασία των δεδομένων.

3.3.2 Προεπεξεργασία των δεδομένων

Σε αυτό το σημείο προκύπτει το ερώτημα του πως μπορούμε να μετασχηματίσουμε τις εικόνες JPG που έχουμε στον υπολογιστή μας σε μία μορφή ώστε να μπορούν να τροφοδοτηθούν σε ένα νευρωνικό δίκτυο. Μία μέθοδος είναι η αντιστοίχιση μίας πολύχρωμης εικόνας μεγέθους $n \times m$, σε έναν τανυστή με διαστάσεις $(1, n, m, 3)$, όπου η τέταρτη διάσταση αντιστοιχεί στα τρία κανάλια από το χρώμα της εικόνας, δηλαδή σε εντάσεις RBG $\in [0,255]$. Αυτό μπορεί να πραγματοποιηθεί στην R μέσα σε λίγες γραμμές κώδικα. Όμως, εάν αντιστοιχούσαμε όλους τους N τανυστές $(N, n, m, 3)$ σε μία μεταβλητή `training_data` τότε σύντομα θα αντιλαμβανόμασταν το μεγάλο κόστος της αποθήκευσης αυτής της μεταβλητής στην μνήμη του υπολογιστή (RAM), την οποία ενδέχεται να μην είναι ικανό το σύστημά μας να διαχειριστεί, ενώ βρισκόμαστε ακόμα πριν την εκπαίδευση κατά την οποία ο υπολογιστής μας πρέπει να χειριστεί πράξεις χιλιάδων παραμέτρων. Ως αποτέλεσμα, η εκπαίδευση με πολλά δεδομένα όπως οι εικόνες, απαιτεί διαφορετική προσέγγιση. Το πακέτο `keras` διαθέτει `generators` με τους οποίους μόνο μία εικόνα (ή μία παρτίδα εικόνων) τροφοδοτείται στο νευρωνικό δίκτυο κατά την εμπρόσθια τροφοδότηση, δίχως να υπάρχουν οι υπόλοιπες εικόνες στην μνήμη του υπολογιστή. Όταν ολοκληρωθεί η οπίσθια διάδοση, τότε ο `generator` θα αναπαράγει την επόμενη εικόνα (ή την παρτίδα εικόνων) και η

διαδικασία της εμπρόσθιας τροφοδότησης και οπίσθιας διάδοσης θα επαναληφθεί. Όταν περάσουν όλες οι εικόνες που έχουμε στα δεδομένα εκπαίδευσης, θα έχει περάσει μία *εποχή* (epoch).

Μπορούμε να θεωρήσουμε έναν generator ο οποίος δρα στα δεδομένα εκπαίδευσης, στα δεδομένα επικύρωσης και στα δεδομένα ελέγχου, με τις ακόλουθες εντολές:

```
library(keras)
train_datagen <- image_data_generator(rescale = 1/255)
validation_datagen <- image_data_generator(rescale = 1/255)
test_datagen <- image_data_generator(rescale = 1/255)
```

Οι συγκεκριμένοι generators αυτόματα μετατρέπουν όλες τις εικόνες σε τανυστές μεγέθους $(1, N, M, 3)$, όπου N, M είναι οι αρχικές διαστάσεις της εικόνας. Έπειτα, μπορούμε να εκτελέσουμε την αλλαγή των εικόνων σε ένα συγκεκριμένο μέγεθος και να επιτρέψουμε την ροή των εικόνων από τους φακέλους όπου βρίσκονται με τις παρακάτω εντολές:

```
train_generator <- flow_images_from_directory(
    train_dir,
    train_datagen,
    target size = c(150, 150),
    batch_size = 32,
    class mode = 'binary')

validation_generator <- flow_images_from_directory(
    validation dir,
    validation_datagen,
    target size = c(150,150),
    batch_size = 32,
    class_mode = 'binary')

test_generator <- flow_images_from_directory(
    test dir,
    test_datagen,
    target size = c(150, 150),
    batch_size = 32,
    class_mode = 'binary')
```

3.3.3 Μοντελοποίηση, εκπαίδευση και επικύρωση

Συνεπώς, τώρα μπορούμε να θεωρήσουμε το μοντέλο μας. Θα ξεκινήσουμε από ένα απλό μοντέλο, το οποίοι θα ονομάσουμε `model_1`, το οποίο είχαμε θεωρήσει στο πρώτο σχήμα των συνελικτικών νευρωνικών δικτύων που συναντήσαμε, στο κεφάλαιο 3. Αυτό αποτελείται από 2 κομμάτια: την *συνελικτική βάση* και το μοντέλο πρόβλεψης που συνδέεται με την συνελικτική βάση. Επιπλέον, στην στοιβάδα εξόδου θα υπάρχει ένας κόμβος, καθώς αυτό είναι ένα πρόβλημα κατηγοριοποίησης 2 κλάσεων.

```
model <- keras_model_sequential() %>%
    layer conv 2d(filters = 64, kernel size = c(3,3), activation = 'relu',
                  input_shape = c(150, 150, 3)) %>%
    layer max pooling 2d(pool size = c(2, 2)) %>%
    layer flatten() %>%
    layer_dense(units = 64, activation = 'relu') %>%
    layer_dense(units = 1, activation = 'sigmoid')
```

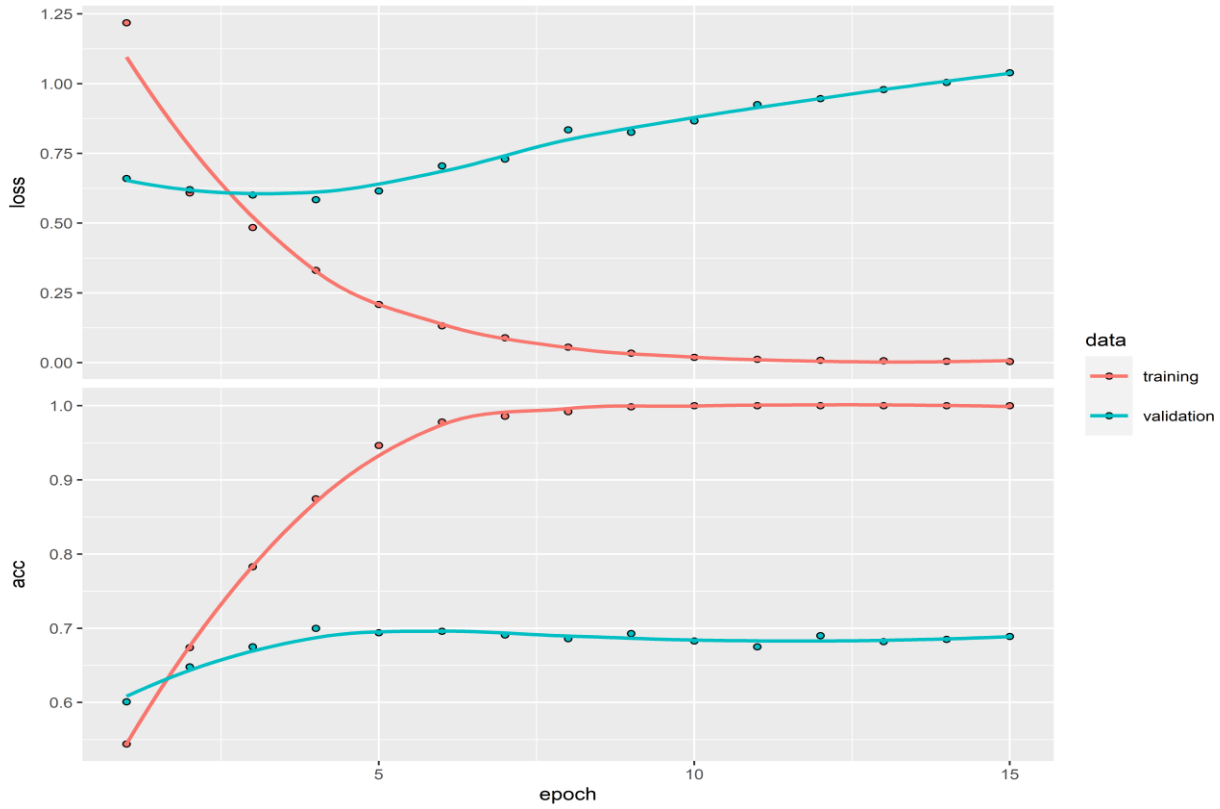
Με την εντολή `summary()` μπορούμε να δούμε την δομή του νευρωνικού δικτύου καθώς και τις διαστάσεις των στοιβάδων και τον πλήθος των παραμέτρων του:

```
> summary(model)
Model: "sequential_12"
_____________________________________________________________________________
Layer (type)                     Output Shape                  Param
=============================================================================
conv2d_12 (Conv2D)               (None, 148, 148, 64)          1792
```

```
max_pooling2d_1 (MaxPooling2D)          (None, 74, 74, 64)                0
flatten_12 (Flatten)                    (None, 350464)                    0
dense_25 (Dense)                        (None, 64)                        22429760
dense_24 (Dense)                        (None, 1)                         65
================================================================================
    Total params: 22,431,617
Trainable params: 22,431,617
Non-trainable params: 0
_________________________________________________________________________________
```

Παρατηρούμε πως ένα τόσο απλό μοντέλο, τελικά καταλήγει να έχει πάρα πολλές παραμέτρους, δίχως να είναι βαθύ. Είναι συχνό φαινόμενο να μην αρκεί η μνήμη του GPU για να πραγματοποιήσει όλες τις παράλληλες πράξεις που επιθυμούμε. Για παράδειγμα, εάν θεωρήσουμε το ίδιο μοντέλο αλλά με 65 νευρώνες στην κρυφή στοιβάδα αντί για 64, είναι πολύ πιθανό πως θα υπάρξει σφάλμα κατά την εκτέλεση της εντολής. Επίσης είναι σημαντικό πως το `batch_size` ουσιαστικά θεωρεί ταυτόχρονους υπολογισμούς από `batch_size` το πλήθος νευρωνικά δίκτυα στην κάρτα γραφικών μας, το οποίο δυσχεραίνει τις συνθήκες κατά την εκπαίδευση όταν εργαζόμαστε με τόσες πολλές παραμέτρους. Αυτός είναι ένας λόγος που, καθώς μεγαλώνουν τα μοντέλα σε πλήθος παραμέτρων, αναγκαζόμαστε να χρησιμοποιήσουμε μικρότερο `batch_size` στα νευρωνικά δίκτυα. Ας παρατηρήσουμε την μάθηση που θα πραγματοποιήσει το μοντέλο, δίνοντας τις παρακάτω εντολές.

```
model %>% compile(
    loss = 'binary_crossentropy',
    optimizer = 'adam',
    metrics = c('acc'))
history <- model %>% fit(
    train_generator,
    epochs = 15,
    validation_data = validation_generator,
    verbose = 2)
plot(history)
```



Παρατηρούμε πως η εντολή μας επιστρέφει ένα ενιαίο σχήμα, το οποίο αποτελείται από δύο γραφήματα: αυτό όπου στα αριστερά του γράφει «loss» είναι το γράφημα του σφάλματος της επικύρωσης (validation) (με την πράσινη γραμμή) μαζί με το σφάλμα της εκπαίδευσης (κόκκινη γραμμή). Παρατηρούμε πως το σφάλμα της επικύρωσης αρχίζει μετά την 5^{η} εποχή να μεγαλώνει, δηλαδή πλέον το μοντέλο έχει αρχίσει την υπερκάλυψη (overfit) των δεδομένων εκπαίδευσης. Όπως είναι αναμενόμενο, το σφάλμα της εκπαίδευσης (training), μειώνεται με ταχύ ρυθμό.

Το γράφημα όπου στα αριστερά του γράφει «acc» είναι το γράφημα της ακρίβειας του μοντέλου πρόβλεψης του νευρωνικού δικτύου στα δεδομένα επικύρωσης (πράσινη γραμμή) μαζί με την ακρίβεια στα δεδομένα εκπαίδευσης (κόκκινη γραμμή). Παρατηρούμε πως η ακρίβεια της επικύρωσης είναι σταθερή στο 0.7 και δεν βελτιώνεται πλέον μετά την 5^{η} εποχή, ενώ το μοντέλο πολύ γρήγορα φτάνει να έχει ακρίβεια 1 στα δεδομένα. Τα αποτελέσματα συνοψίζονται στον πίνακα στο τέλος της ενότητας. Αυτό το μοντέλο πλέον μπορεί να αποτελέσει μία *κατώτατη βάση* (baseline) με την οποία θα κρίνουμε τα επόμενα μοντέλα. Ένας εύκολος τρόπος να το κάνουμε, είναι με την *προσαύξηση των δεδομένων* (data augmentation), Αυτή, περιλαμβάνει μετασχηματισμούς στις εικόνες όπως το (τυχαίο) τράβηγμα, η επιμήκυνση, η μεγέθυνση και η περιστροφή των εικόνων, νέες εικόνες που δεν έχει ξανά έρθει σε επαφή το νευρωνικό δίκτυο (Chollet & Allaire, 2018). Για να το κάνουμε αυτό, θα χρησιμοποιήσουμε τους `generators` όπως αυτοί ορίζονται στα ακόλουθα, και θα συνεχίσουμε να χρησιμοποιούμε αυτούς τους `generators` και στα επόμενα μοντέλα νευρωνικών δικτυών που ακολουθούν, εκτός αν οριστούν άλλοι.

```
train_datagen <- image_data_generator(
    rescale = 1/255,
    rotation_range = 20,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = TRUE,
```

```
    fill_mode = 'nearest')
validation_datagen <- image_data_generator(rescale = 1/255)
test_datagen <- image_data_generator(rescale = 1/255)
train_generator <- flow_images_from_directory(
    train_dir,train_datagen, target_size = c(150, 150),
    batch_size = 64,class_mode = 'binary')
validation_generator <- flow_images_from_directory(
    validation_dir, validation_datagen, target_size = c(150,150),
    batch_size = 64, class_mode = 'binary')
test_generator <- flow_images_from_directory(
    test_dir, test_datagen, target_size = c(150, 150),
    batch_size = 64, class_mode = 'binary')
```

Στο σχήμα που ακολουθεί απεικονίζονται κάποιες τιμές από τον `train_generator` με επαύξηση, από τις εντολές:

```
par(mfrow = c(3, 3),mar = c(1, 0, 1, 0))
for (i in 1:9) { batch <- generator_next(train_generator)
    plot(as.raster(batch[[1]][32, , , ]))}
```

Σχήμα 33. Εικόνες των δεδομένων με προσαύξηση κατά ένα τυχαίο ποσοστό, όπως περιγράφονται στην μεταβλητή `train_datagen()`.

Θεωρούμε το *δεύτερο μοντέλο*, το οποίο αν και πιο περίπλοκο σε αρχιτεκτονική, έχει λιγότερες παραμέτρους.

```
model <- keras_model_sequential() %>%
    layer_conv_2d(filters = 128, kernel_size = c(3,3), activation = 'relu',
                  input_shape = c(150, 150, 3)) %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
```

```
    layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu') %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_flatten() %>%
    layer_dense(units = 256, activation = 'relu') %>%
    layer_dense(units = 128, activation = 'relu') %>%
    layer_dense(units = 1, activation = 'sigmoid')
model %>% save_model_hdf5(filepath = '~/ch3_model2.h5') # run after training
summary(model)
```

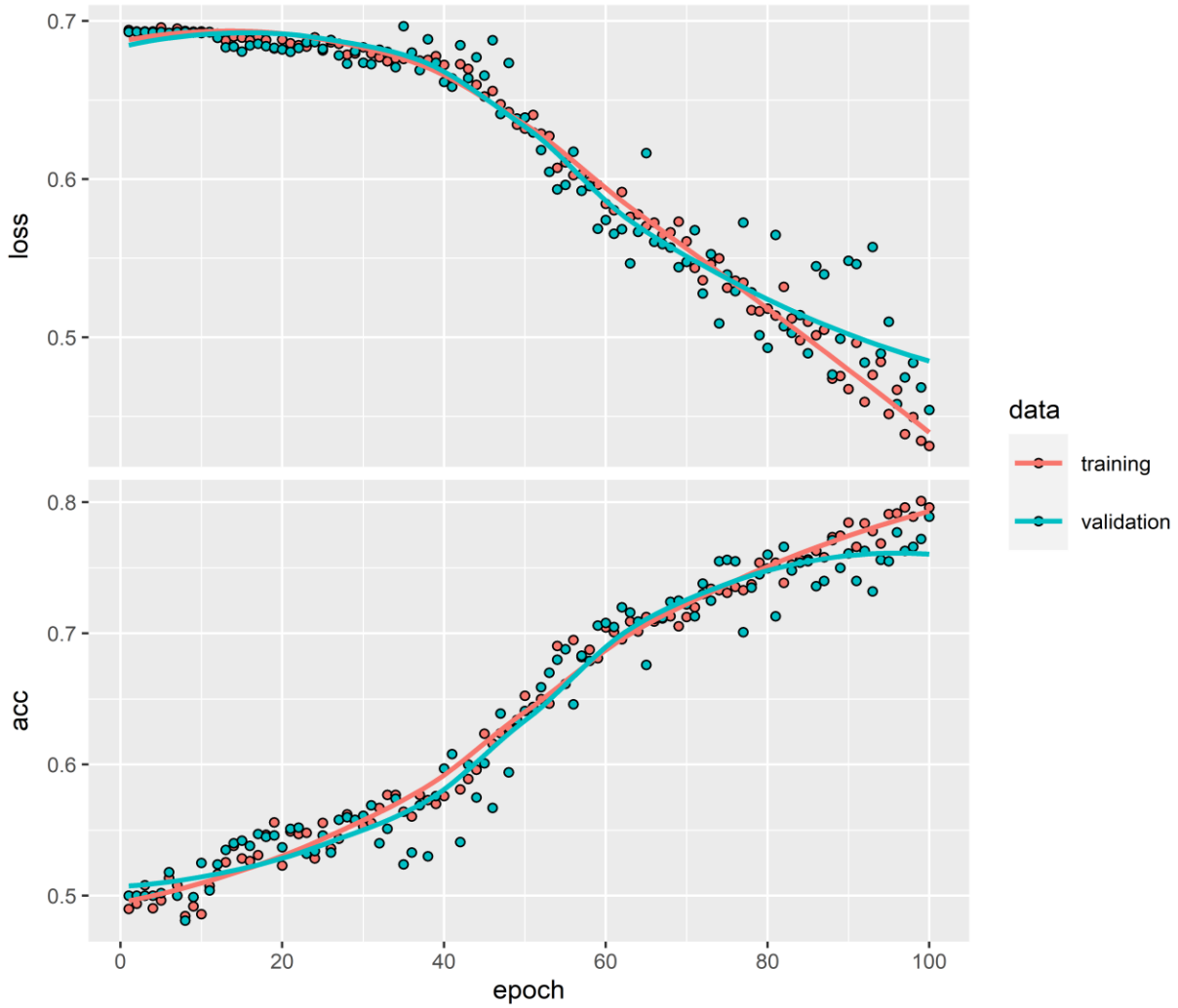
Με την εντολή `save_model_hdf5(…)`, μπορούμε να καλέσουμε αργότερα το μοντέλο 2 για περαιτέρω ανάλυση μέσω της εντολής `load_model_hdf5(…)`. Η εντολή `summary(model)` επιστρέφει:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 148, 148, 128)     3584
max_pooling2d_17 (MaxPooling2D) (None, 74, 74, 128)    0
conv2d_16 (Conv2D)           (None, 72, 72, 64)        73792
max_pooling2d_16 (MaxPooling2D) (None, 36, 36, 64)     0
conv2d_15 (Conv2D)           (None, 34, 34, 64)        36928
max_pooling2d_15 (MaxPooling2D) (None, 17, 17, 64)     0
conv2d_14 (Conv2D)           (None, 15, 15, 32)        18464
max_pooling2d_14 (MaxPooling2D) (None, 7, 7, 32)       0
conv2d_13 (Conv2D)           (None, 5, 5, 32)          9248
max_pooling2d_13 (MaxPooling2D) (None, 2, 2, 32)       0
flatten_3 (Flatten)          (None, 128)               0
dense_8 (Dense)              (None, 512)               66048
dense_7 (Dense)              (None, 1)                 513
=================================================================
Total params: 208,577
Trainable params: 208,577
Non-trainable params: 0
```

Στην αρχιτεκτονική του δεύτερου μοντέλου, εναλλάσσονται τέσσερις στοιβάδες συνέλιξης και ομαδοποίησης με σκοπό την εξαγωγή των κυριότερων χαρακτηριστικών των εικόνων και την μείωση των παραμέτρων, οι οποίες συνδέονται έπειτα σε ένα νευρωνικό δίκτυο MLP με μία κρυφή στοιβάδα (dense_8) και μία στοιβάδα εξόδου (dense_7), μέσω διανυσματοποίησης (*στοιβάδα: flatten_3*). Δίνοντας επιπλέον τις παρακάτω εντολές, λαμβάνουμε το γράφημα στην επόμενη σελίδα (*αυτές οι εντολές θα παραμείνουν ίδιες και στα επόμενα μοντέλα*):

```
model %>% compile(
    loss = 'binary_crossentropy',
    optimizer = 'adam',
    metrics = c('acc'))
history <- model %>% fit(
    train_generator,
    epochs = 100,
    validation_data = validation_generator,
    verbose = 2)
```

```
plot(history)
```



Παρατηρούμε πως στις 100 εποχές ακόμα δεν έχουμε φαινόμενα υπερπροσαρμογής, καθώς το σφάλμα της επικύρωσης ακόμα μειώνεται. Συνεπώς, καταφέραμε να αντιμετωπίσουμε την υπερπροσαρμογή που υπήρχε στο πρώτο μοντέλο. Όμως, αντιμετωπίζοντας την πρώιμη υπερπροσαρμογή, τελικά το μοντέλο δεν κατάφερε να φτάσει σε μία αξιοπρεπή ακρίβεια επικύρωσης (78.9%). Συνεχίζοντας, θα θεωρήσουμε το ίδιο μοντέλο αλλά θα προσπαθήσουμε να δυναμώσουμε τις στοιβάδες του δίχως να το κάνουμε βαθύτερο. Για να το πετύχουμε αυτό, μετά από κάθε στοιβάδα ομαδοποίησης του μοντέλου 2 θα προσθέσουμε μία στοιβάδα *απόσυρσης* (dropout), με ρυθμό απόσυρσης ίσο με 20%. Αυτό έχει ως αποτέλεσμα σε κάθε στοιβάδα συνέλιξης να επιλέγεται τυχαία το 20% των νευρώνων και να θεωρείται ως ανενεργό για εκείνη την εποχή μάθησης, με αποτέλεσμα να μην εξαρτάται το νευρωνικό δίκτυο σε μία ή δύο αναπαραστάσεις για να βγάλει τα αποτελέσματα, αλλά να αναγκάζεται να ενισχύσει ομοιόμορφα τους υπόλοιπους νευρώνες του δικτύου. Αυτά μπορούν να γίνουν με τις εντολές:

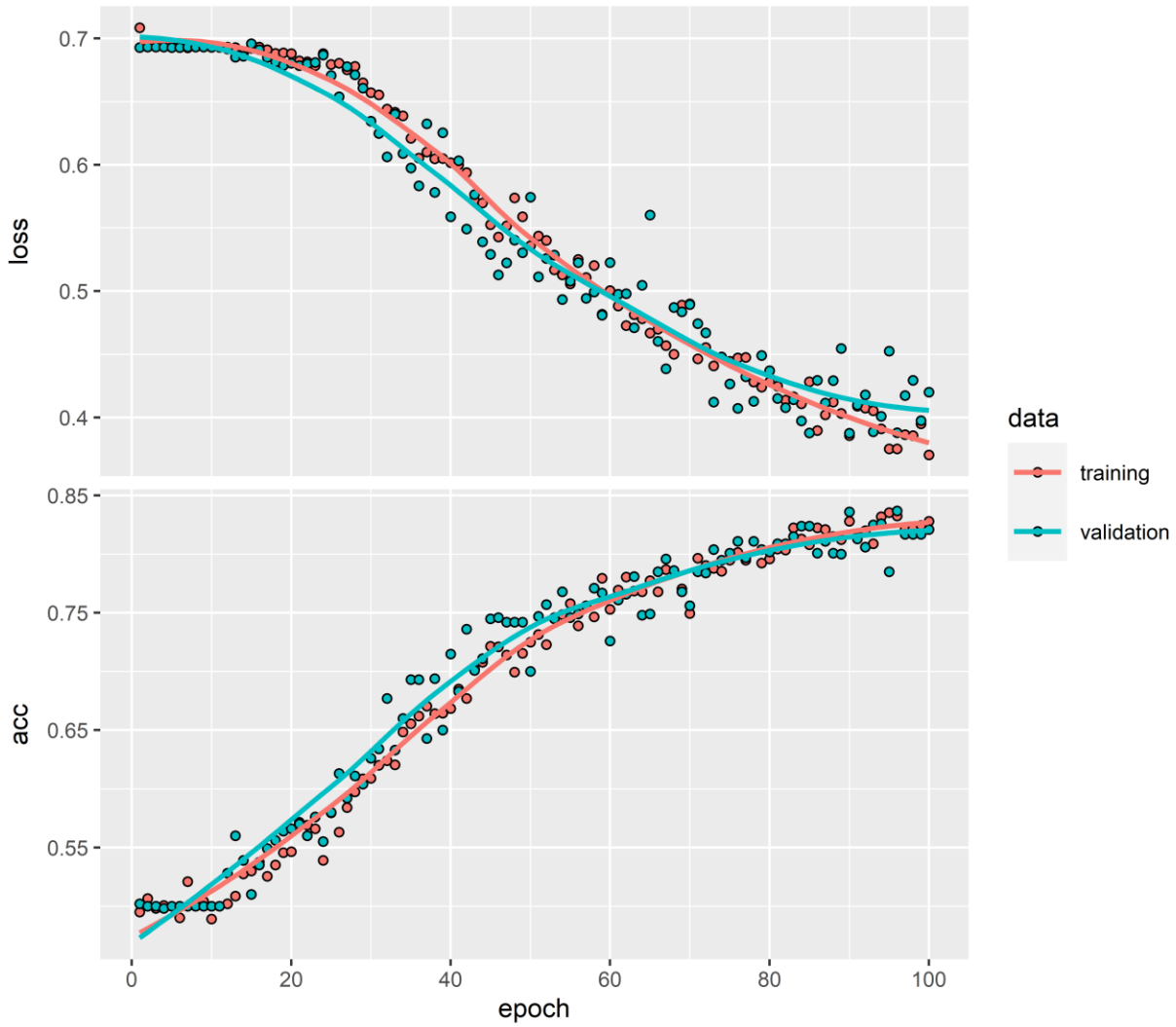
```
model <- keras_model_sequential() %>%
    layer_conv_2d(filters = 128, kernel_size = c(3,3), activation = 'relu',
                  input_shape = c(150, 150, 3)) %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
```

```
    layer_dropout(0.2) %>%
    layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_dropout(0.2) %>%
    layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_dropout(0.2) %>%
    layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu') %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_dropout(0.2) %>%
    layer_flatten() %>%
    layer_dense(units = 512, activation = 'relu') %>%
    layer_dense(units = 1, activation = 'sigmoid')
model %>% save_model_hdf5(filepath = '~/ch3_model3.h5') # run after training
summary(model)
```

Το οποίο επιστρέφει:

```
Layer (type)                         Output Shape                    Param #
================================================================================
conv2d_40 (Conv2D)                   (None, 148, 148, 128)           3584

max_pooling2d_40 (MaxPooling2D)      (None, 74, 74, 128)             0

dropout_7 (Dropout)                  (None, 74, 74, 128)             0

conv2d_39 (Conv2D)                   (None, 72, 72, 64)              73792

max_pooling2d_39 (MaxPooling2D)      (None, 36, 36, 64)              0

dropout_6 (Dropout)                  (None, 36, 36, 64)              0

conv2d_38 (Conv2D)                   (None, 34, 34, 64)              36928

max_pooling2d_38 (MaxPooling2D)      (None, 17, 17, 64)              0

dropout_5 (Dropout)                  (None, 17, 17, 64)              0

conv2d_37 (Conv2D)                   (None, 15, 15, 32)              18464

max_pooling2d_37 (MaxPooling2D)      (None, 7, 7, 32)                0

dropout_4 (Dropout)                  (None, 7, 7, 32)                0

flatten_8 (Flatten)                  (None, 1568)                    0

dense_18 (Dense)                     (None, 512)                     803328

dense_17 (Dense)                     (None, 1)                       513

================================================================================
Total params: 936,609
Trainable params: 936,609
Non-trainable params: 0
```

Παρατηρούμε πως οι περισσότερες παράμετροι του τρίτου μοντέλου βρίσκονται μεταξύ της στοιβάδας διανυσματοποίησης και της πρώτης κρυφής στοιβάδας του νευρωνικού δικτύου πολυστρωματικών αντιλήπτρων. Δίνοντας τις ίδιες εντολές για την εκπαίδευση με αυτές του μοντέλου 2, προκύπτουν τα ακόλουθα.

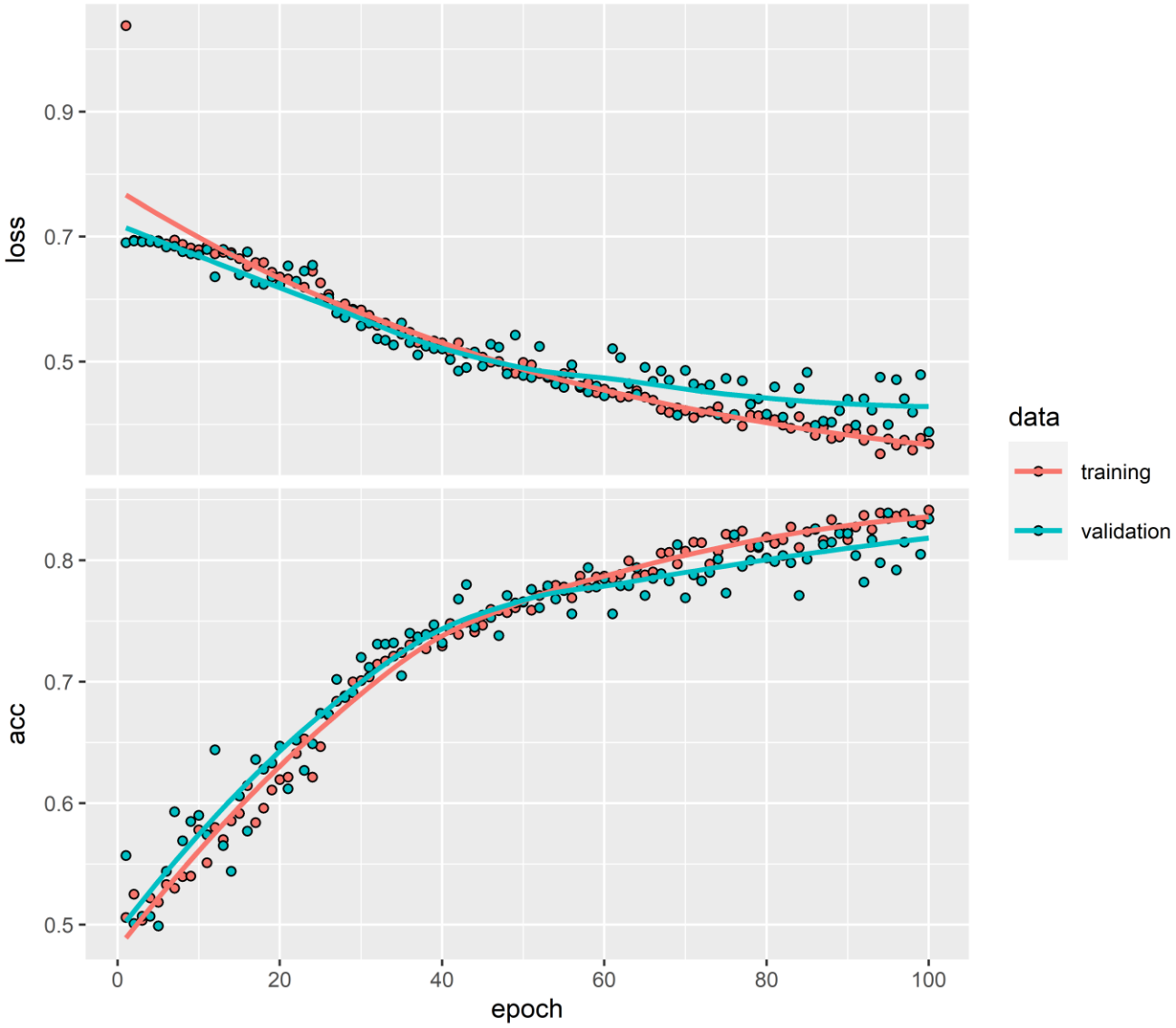


Μετά την εντολή `plot(history)` για τις ίδιες παραμέτρους όπως προηγουμένως, παρατηρούμε πως η ακρίβεια όντως αυξήθηκε στο σύνολο επικύρωσης, δίχως να υπάρχει υπερπροσαρμογή ή υποπροσαρμογή των δεδομένων. Συνεπώς, με το επόμενο μοντέλο, θα προσπαθήσουμε να αυξήσουμε την χωρητικότητα του μοντέλου. Αυτό θα το κάνουμε δίχως να μεταβάλλουμε τις στοιβάδες του. Θα δοκιμάσουμε τις τεχνικές του παραγεμίσματος (padding) σε συνδυασμό με τον βηματισμό (striding), ώστε να διατηρούνται οι διαστάσεις στις στοιβάδες συνέλιξης. Επιπλέον, επειδή το μοντέλο που προκύπτει έχει πολλές παραμέτρους (1.460.897) θα πραγματοποιήσουμε και αρχικοποίηση στα βάρη ώστε να αποφευχθούν τα τοπικά ελάχιστα κατά την εκκίνηση. Θα πραγματοποιήσουμε εκκίνηση βαρών σε κάθε συνελικτική στοιβάδα από κανονική κατανομή με μέση τιμή 0 και τυπική απόκλιση 0.5.

```
model <- keras_model_sequential() %>%
    layer_conv_2d(filters = 128, kernel_size = c(3,3), activation = 'relu',
                  input_shape = c(150, 150, 3),
                  padding = "same", strides = 1,
                  kernel_regularizer = tf$random_normal_initializer(stddev = 0.5)) %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_dropout(0.2) %>%
    layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu',
                  padding = "same", strides = 1,
                  kernel_regularizer = tf$random_normal_initializer(stddev = 0.5)) %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
```

```
    layer_dropout(0.2) %>%
    layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu',
                  padding = "same", strides = 1,
                  kernel_regularizer = tf$random_normal_initializer(stddev = 0.5)) %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_dropout(0.2) %>%
    layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu',
                  padding = "same", strides = 1,
                  kernel_regularizer = tf$random_normal_initializer(stddev = 0.5)) %>%
    layer_max_pooling_2d(pool_size = c(2,2)) %>%
    layer_dropout(0.2) %>%
    layer_flatten() %>%
    layer_dense(units = 512, activation = 'relu') %>%
    layer_dense(units = 1, activation = 'sigmoid')
model %>% save_model_hdf5(filepath = '~/ch3_model4.h5') # run after training
summary(model)
```

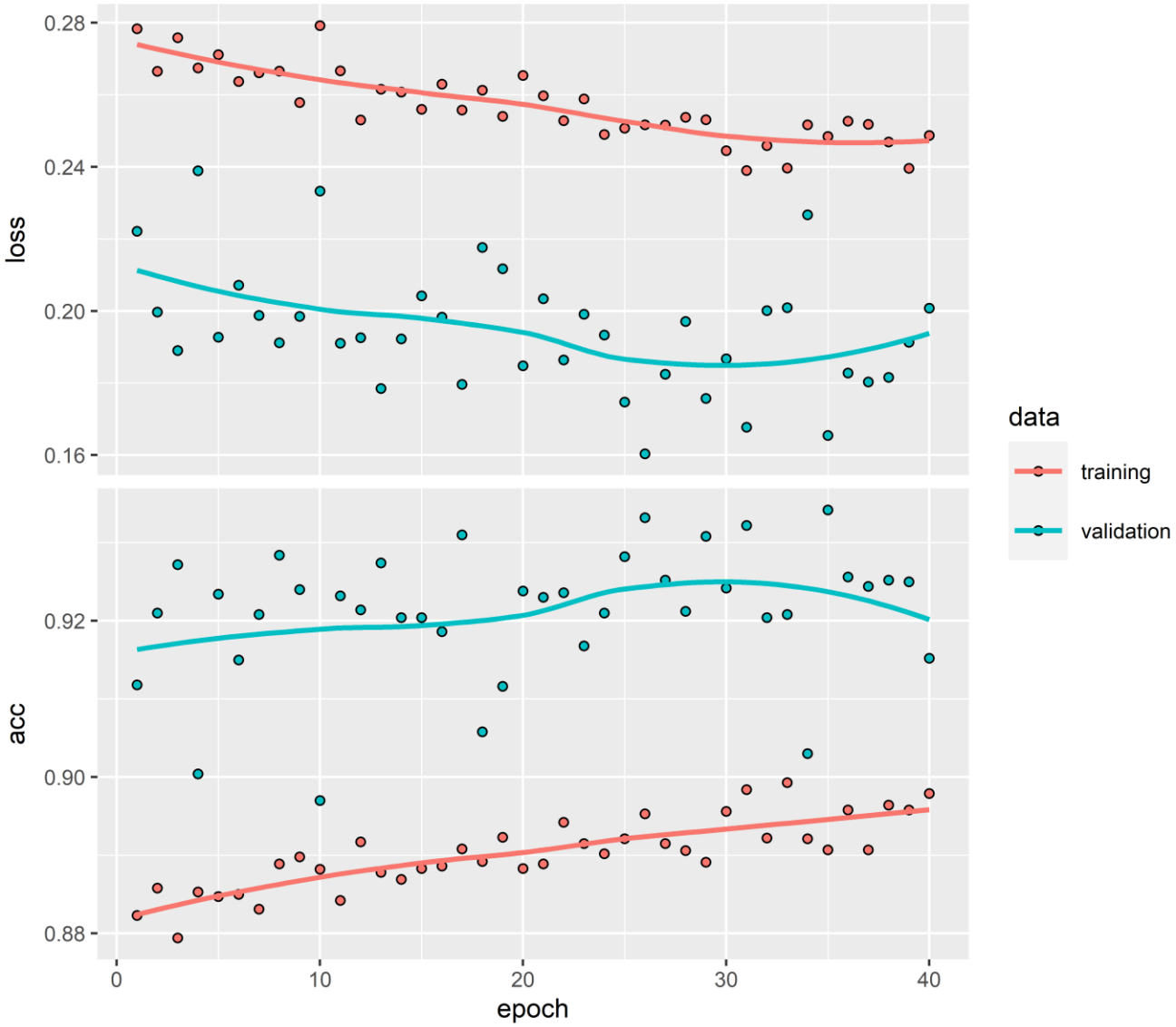
```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)            (None, 150, 150, 128)     3584

 max_pooling2d_3 (MaxPooling2D)  (None, 75, 75, 128)    0

 dropout_3 (Dropout)          (None, 75, 75, 128)       0

 conv2d_2 (Conv2D)            (None, 75, 75, 64)        73792

 max_pooling2d_2 (MaxPooling2D)  (None, 37, 37, 64)     0

 dropout_2 (Dropout)          (None, 37, 37, 64)        0

 conv2d_1 (Conv2D)            (None, 37, 37, 64)        36928

 max_pooling2d_1 (MaxPooling2D)  (None, 18, 18, 64)     0

 dropout_1 (Dropout)          (None, 18, 18, 64)        0

 conv2d (Conv2D)              (None, 18, 18, 32)        18464

 max_pooling2d (MaxPooling2D)  (None, 9, 9, 32)         0

 dropout (Dropout)            (None, 9, 9, 32)          0

 flatten (Flatten)            (None, 2592)              0

 dense_1 (Dense)              (None, 512)               1327616

 dense (Dense)                (None, 1)                 513

=================================================================
Total params: 1,460,897
Trainable params: 1,460,897
Non-trainable params: 0
```



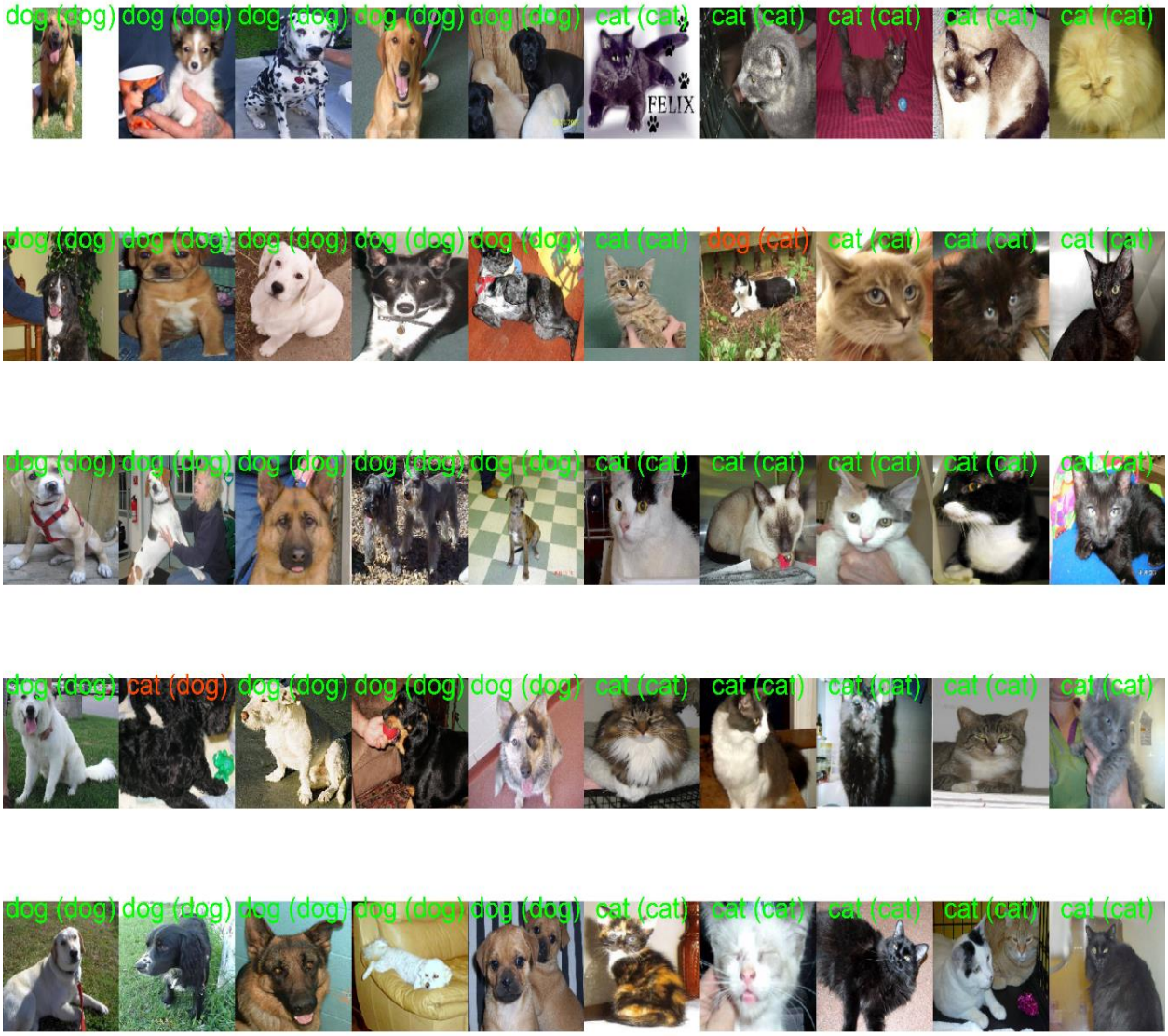
Σε αυτό το σημείο παρατηρούμε πως μέσω της συνεχόμενης αλλαγής των υπερπαραμέτρων, καταφέραμε τελικά στο μοντέλο 4 να λάβουμε ακρίβεια επικύρωσης ίση με 0.834 (πίνακας στο τέλος της ενότητας). Όμως η ακρίβεια στα δεδομένα ελέγχου είναι η ίδια με την ακρίβεια που είχε το μοντέλο 3. Συνεπώς, ενδέχεται το μοντέλο πρόβλεψης να προσαρμόστηκε στα δεδομένα επικύρωσης, μέσω της αλλαγής των υπερπαραμέτρων του σε αυτά τα παραδείγματα. Για αυτό, συνεχίζοντας θα προτιμήσουμε το μοντέλο 3, καθώς έχει λιγότερες παραμέτρους από το μοντέλο 4. Θα συνεχίσουμε την εκπαίδευσή του για 100 εποχές, με ακόμα 5000 διαφορετικά δεδομένα εκπαίδευσης, και θα θεωρήσουμε ακόμα 2500 διαφορετικά δεδομένα για επικύρωση και έλεγχο. Αυτά γίνονται με τις ακόλουθες εντολές:

```
original_dataset_dir <- "~/kaggle_data"
original_train_dir <- file.path(original_dataset_dir, "train")
original_test_dir <- file.path(original_dataset_dir, "test")
base_dir <- "~/subset_medium"
dir.create(base_dir)

train_dir <- file.path(base_dir, "train")
dir.create(train_dir)
validation_dir <- file.path(base_dir, 'validation')
```

```
dir.create(validation_dir)
test_dir <- file.path(base_dir, 'test')
dir.create(test_dir)
train cats dir <- file.path(train dir, "cats")
dir.create(train_cats_dir)
train dogs dir <- file.path(train dir, "dogs")
dir.create(train_dogs_dir)
validation_cats_dir <- file.path(validation_dir, "cats")
dir.create(validation cats dir)
validation_dogs_dir <- file.path(validation_dir, "dogs")
dir.create(validation dogs dir)
test_cats_dir <- file.path(test_dir, "cats")
dir.create(test_cats_dir)
test_dogs_dir <- file.path(test_dir, "dogs")
dir.create(test_dogs_dir)

# Copying the data
fnames <- paste0("cat.", 2001:7000, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(train_cats_dir))
fnames <- paste0("cat.", 7001:9500, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(validation cats dir))
fnames <- paste0("cat.", 9501:12000, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(test cats dir))
fnames <- paste0("dog.", 2001:7000, ".jpg")
file.copy(file.path(original train dir, fnames),
          file.path(train_dogs_dir))
fnames <- paste0("dog.", 7001:9500, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(validation_dogs_dir))
fnames <- paste0("dog.", 9501:12000, ".jpg")
file.copy(file.path(original_train_dir, fnames),
          file.path(test dogs dir))
for (c in c(train_cats_dir, train_dogs_dir,
            validation_cats_dir, validation_dogs_dir,
            test cats dir, test dogs dir)){
    cat('\n', as.character(c), "has ", length(list.files(c)), "files in total.")}
model <- load model hdf5(filepath = '~/ch3 model3.h5') # trained model


model %>% compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = c('acc'))
history <- model %>% fit(train_generator, epochs = 100, validation_data = validation_generator)
plot(history)
```



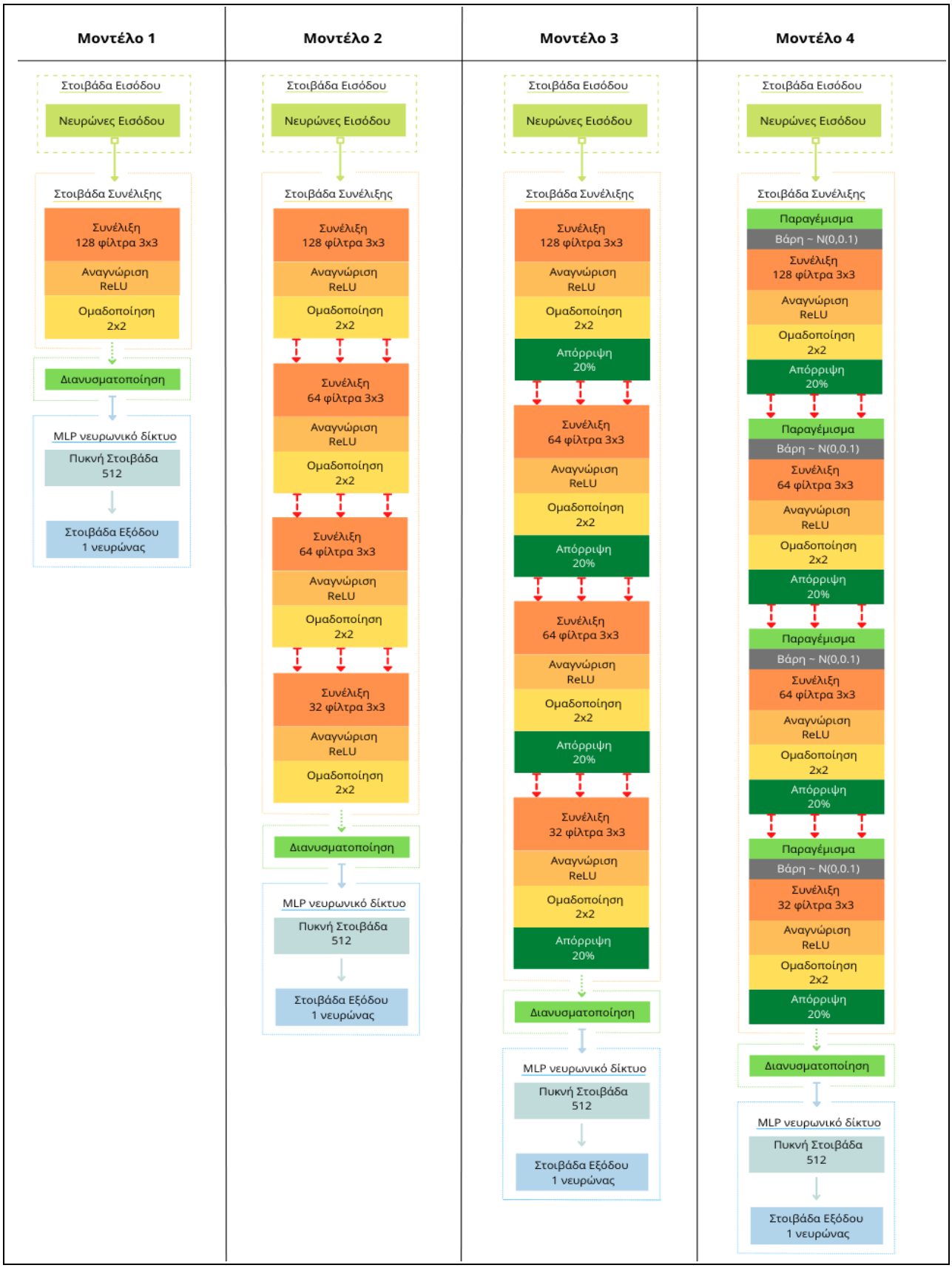
Είναι αναμενόμενο πως η ακρίβεια στα δεδομένα εκπαίδευσης θα είναι μικρότερη από αυτήν στα δεδομένα επικύρωσης, καθώς το νευρωνικό δίκτυο δεν έχει ξανασυναντήσει ούτε τα 5000 δεδομένα εκπαίδευσης ούτε τα 2500 δεδομένα επικύρωσης. Ταυτόχρονα, στο σύνολο εκπαίδευσης είναι ευκολότερο να πραγματοποιήσει λάθος πρόβλεψη, λόγω του μεγάλου πλήθος των εικόνων, συνεπώς και η μικρότερη ακρίβεια. Στο γράφημα εστιάζουμε κυρίως στις 40 τελευταίες εποχές μέχρι να φτάσουμε στην 200^{η} (από την αρχή) εποχή εκπαίδευσης. Παρατηρούμε πως το μοντέλο έχει γενική ικανότητα κατηγοριοποίησης σκύλων και γάτων, πέραν του συνόλου εκπαίδευσης. Εφόσον το μοντέλο δεν έχει ακόμα εμφανίσει υπερπροσαρμογή, οι επιδόσεις του αναμένεται να βελτιωθούν ακόμα. Παραθέτουμε κάποιες εικόνες από ένα σύνολο το οποίο δεν περικλείεται στα δεδομένα εκπαίδευσης μαζί με την αντίστοιχη κατηγοριοποίηση του μοντέλου 3. Επίσης, παραθέτουμε τον *Πίνακα 5*, όπου συνοψίζονται τα αποτελέσματα των μοντέλων, καθώς και ένα σχήμα όπου παρουσιάζονται γραφικά όλα τα μοντέλα.



Σχήμα 34. Προβλέψεις μέσω του μοντέλου 3, το οποίο εκπαιδεύσαμε σε 6000 δεδομένα. Με πράσινο απεικονίζονται οι σωστές περιπτώσεις και με κόκκινο οι λανθασμένες. Παρατηρούμε πως στις περιπτώσεις των λαθών, δεν είναι άμεσα ευδιάκριτο εάν πρόκειται για σκύλο ή για γάτα.

Μοντέλο 1 (1000 δεδομένα εκπαίδευσης)	**Ακρίβεια Εκπαίδευσης (15 εποχές)**	**Ακρίβεια Επικύρωσης (15 εποχές)**		**Χρόνος εκπαίδευσης (λεπτά)**	**Σχόλια (15 εποχές)**
22,431,617 παράμετροι. Χωρίς επαύξηση δεδομένων. Παρτίδα μεγέθους 64.	1.0	0.688		1	Υπόδειγμα υπερπροσαρμογής μοντέλου με πολλές παραμέτρους και απλή αρχιτεκτονική.
Μοντέλο 2 (1000 δεδομένα εκπαίδευσης)	**Ακρίβεια Εκπαίδευσης (100 εποχές)**	**Ακρίβεια Επικύρωσης (100 εποχές)**		**Χρόνος εκπαίδευσης (λεπτά)**	**Σχόλια (100 εποχές)**
208.577 παράμετροι. Επαύξηση Δεδομένων. Παρτίδα μεγέθους 32.	0.796	0.789		18	Αργή Εκκίνηση, χαμηλή ακρίβεια, **δίχως υπερπροσαρμογή.**
Μοντέλο 3 (1000 δεδομένα εκπαίδευσης)	**Ακρίβεια Εκπαίδευσης (100 εποχές)**	**Ακρίβεια Επικύρωσης (100 εποχές)**	**Γενικευμένη Ακρίβεια (100 εποχές)**	**Χρόνος εκπαίδευσης (λεπτά)**	**Σχόλια (100 εποχές)**
936.609 παράμετροι. Επαύξηση Δεδομένων. Παρτίδα μεγέθους 16.	**0.828**	**0.821**	**0.805**	18	Αργή Εκκίνηση. Καλή ακρίβεια γενίκευσης. Καλή προσαρμογή.
Μοντέλο 4 (1000 δεδομένα εκπαίδευσης)	**Ακρίβεια Εκπαίδευσης (100 εποχές)**	**Ακρίβεια Επικύρωσης (100 εποχές)**	**Γενικευμένη Ακρίβεια (100 εποχές)**	**Χρόνος εκπαίδευσης (λεπτά)**	**Σχόλια (100 εποχές)**
1.460.897 παράμετροι. (Επαύξηση) Παρτίδα μεγέθους 8	0.8415	0.834	0.8	19	Καλή Εκκίνηση. Μέτρια ακρίβεια γενίκευσης. Καλή προσαρμογή.
Μοντέλο 3 (1000 + 5000 δεδομένα εκπαίδευσης)	**Ακρίβεια Εκπαίδευσης (200 εποχές)**	**Ακρίβεια Επικύρωσης (200 εποχές)**	**Γενικευμένη Ακρίβεια (200 εποχές)**	**Χρόνος εκπαίδευσης (λεπτά)**	**Σχόλια (200 εποχές)**
936.609 παράμετροι. Επαύξηση Δεδομένων. Παρτίδα μεγέθους 16	**0.8979**	**0.9152**	**0.9124**	92	Υπάρχει ακόμα περιθώριο μάθησης. Ικανοποιητικά αποτελέσματα. Μεγάλος χρόνος εκπαίδευσης.

Πίνακας 5. Σύγκριση μοντέλων


Μοντέλο 1
Στοιβάδα Εισόδου
Νευρώνες Εισόδου
Στοιβάδα Συνέλιξης
Συνέλιξη 128 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Διανυσματοποίηση
MLP νευρωνικό δίκτυο
Πυκνή Στοιβάδα 512
Στοιβάδα Εξόδου 1 νευρώνας
Μοντέλο 2
Στοιβάδα Εισόδου
Νευρώνες Εισόδου
Στοιβάδα Συνέλιξης
Συνέλιξη 128 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Συνέλιξη 64 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Συνέλιξη 64 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Συνέλιξη 32 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Διανυσματοποίηση
MLP νευρωνικό δίκτυο
Πυκνή Στοιβάδα 512
Στοιβάδα Εξόδου 1 νευρώνας
Μοντέλο 3
Στοιβάδα Εισόδου
Νευρώνες Εισόδου
Στοιβάδα Συνέλιξης
Συνέλιξη 128 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Συνέλιξη 64 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Συνέλιξη 64 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Συνέλιξη 32 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Διανυσματοποίηση
MLP νευρωνικό δίκτυο
Πυκνή Στοιβάδα 512
Στοιβάδα Εξόδου 1 νευρώνας
Μοντέλο 4
Στοιβάδα Εισόδου
Νευρώνες Εισόδου
Στοιβάδα Συνέλιξης
Παραγέμισμα
Βάρη ~ N(0,0.1)
Συνέλιξη 128 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Παραγέμισμα
Βάρη ~ N(0,0.1)
Συνέλιξη 64 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Παραγέμισμα
Βάρη ~ N(0,0.1)
Συνέλιξη 64 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Παραγέμισμα
Βάρη ~ N(0,0.1)
Συνέλιξη 32 φίλτρα 3x3
Αναγνώριση ReLU
Ομαδοποίηση 2x2
Απόρριψη 20%
Διανυσματοποίηση
MLP νευρωνικό δίκτυο
Πυκνή Στοιβάδα 512
Στοιβάδα Εξόδου 1 νευρώνας

Σχήμα 35. Δομή των νευρωνικών δικτύων που αναλύθηκαν σε αυτή την ενότητα. Το κάθε μοντέλο διαβάζεται από τα πάνω προς τα κάτω, ξεκινώντας από την στοιβάδα εισόδου, μετά στην στοιβάδα συνέλιξης (ή στην *συνελικτική βάση*) η οποία περιλαμβάνει τα στάδια συνέλιξης, αναγνώρισης και ομαδοποίησης, μετά συνεχίζοντας έχουμε την διανυσματοποίηση των τιμών από την στοιβάδα συνέλιξης και τελικά την τροφοδότηση σε ένα νευρωνικό δίκτυο πολυεπίπεδων αντιλήπτρων (MLP) το οποίο δρα ως ένα μοντέλο πρόβλεψης από τα εξαγμένα χαρακτηριστικά της στοιβάδας συνέλιξης. Συγκεκριμένα, το μοντέλο πρόβλεψης είναι ίδιο και για τα τέσσερα συνελικτικά νευρωνικά δίκτυα (CNN).

Συνεπώς, καταλήγουμε πως η εκπαίδευση των νευρωνικών δικτύων όταν υπάρχουν πολλά διαθέσιμα δεδομένα είναι χρονοβόρα και υπόκειται στην προκατάληψη που εισέρχεται στο μοντέλο κατά την αλλαγή των υπερπαραμέτρων του προβλήματος. Ακόμα, οι προτεινόμενες μέθοδοι για την καταπολέμηση προβλημάτων κατά την μάθηση ενδέχεται να κοστίσουν τελικά στην απόδοση του μοντέλου. Καταλήγουμε πως η δημιουργία ενός νευρωνικού δικτύου το οποίο να έχει καλά αποτελέσματα σε άγνωστα δεδομένα είναι δύσκολη διαδικασία, ενώ το χρονικό πλαίσιο που απαιτείται, υποσκάπτει τον πειραματισμό. Αυτό έχει οδηγήσει πολλούς ερευνητές στην χρήση προεκπαιδευμένων νευρωνικών δικτύων για την επίλυση προβλημάτων (Shanmugam et al. 2022), (Bagherzadeh et al. 2021), με μοντέλα όπως το VGG16 (Debnath et al. 2022) ενώ σύγχρονες εφαρμογές προτείνουν νέα προεκπαιδευμένα δίκτυα, όπως το δίκτυο DCASENet (Jung et al. 2021).

Κεφάλαιο 4.
Μεγάλα δεδομένα με Νευρωνικά Δίκτυα

Ενότητα 1.
Αξιοποίηση Μεγάλων Δεδομένων μέσω των Νευρωνικών Δικτύων

Μία αδυναμία των κλασικών αλγόριθμων της μηχανικής μάθησης είναι η μικρή τους επεκτασιμότητα, η οποία αποτρέπει την εκμετάλλευση των μεγάλων δεδομένων (Al-Jarrah et al. 2015), καθώς και η ανάγκη για την μηχανική των χαρακτηριστικών (feature engineering) από τον ερευνητή (Najafabadi et al. 2015). Όμως τα νευρωνικά δίκτυα αποτελούν την εφαρμογή όπου η αναλυτική των μεγάλων δεδομένων λαμβάνει τη μεγαλύτερη προσοχή της ακαδημαϊκής έρευνας (Chiroma et al. 2019). Οι αλγόριθμοι των νευρωνικών δικτύων προσφέρουν μεθόδους αντιμετώπισης σε κάποιες από τις προκλήσεις στην αξιοποίηση της ανάλυσης των μεγάλων δεδομένων, οι οποίες προκύπτουν από τα χαρακτηριστικά τους. Συνήθως τα νευρωνικά δίκτυα εφαρμόζονται σε τομείς των μεγάλων δεδομένων όπως η αναγνώριση ομιλίας, η υπολογιστική όραση και η επεξεργασία φυσικής γλώσσας (Najafabadi et al. 2015). Επιπλέον, βρίσκουν πλήθος εφαρμογών σε τομείς όπως η πρόβλεψη άφιξης τουριστών (Hopken et al. 2021), η πρόγνωση του καιρού (Fathi et al. 2021), η πρόβλεψη ασθενειών (Talasila et al. 2020), η δημιουργία ψηφιακών διδύμων (Rathore et al. 2021), ενώ πλήθος άλλων εφαρμογών συνεχίζουν να παράγονται.
Παρακάτω, θα εστιάσουμε στους διαφορετικούς τύπους των αλγόριθμων από τα νευρωνικά δίκτυα που αξιοποιούνται σε μεγάλα δεδομένα. Δύο μεγάλες κατηγορίες των νευρωνικών δικτύων, αποτελούν τα Νευρωνικά Δίκτυα Πολυστρωματικών Αντιλήπτρων (Multilayered Perceptron Neural Network) και τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks). Τα πρώτα, περιλαμβάνουν τα Νευρωνικά Δίκτυα Οπίσθιας Διάδοσης, τα Συνελικτικά Νευρωνικά Δίκτυα, τα Νευρωνικά Δίκτυα με Συναρτήσεις Ακτινικής Βάσης και τα Νευρωνικά Δίκτυα με Μάθηση Διανυσμάτων Κβαντισμού (Chiroma et al. 2019). Επισημαίνεται πως είναι σημαντική η προ – επεξεργασία των δεδομένων πριν την εισαγωγή στα νευρωνικά δίκτυα, καθώς αυτά περιέχουν πολλές τιμές οι οποίες είναι πιθανό να μην συνεισφέρουν στο πρόβλημα προς επίλυση (Lopez-Miguel, 2021).

Νευρωνικά Δίκτυα οπίσθιας διάδοσης
Τα νευρωνικά δίκτυα οπίσθιας διάδοσης (Back Propagation Neural Networks, BPNN) βρίσκουν πολλές εφαρμογές στον τομέα των μεγάλων δεδομένων, αλλά αυτές συνήθως αφορούν παραλλαγές και υβρίδιά τους, ώστε να ανταποκρίνονται καλύτερα στο πρόβλημα και τα εξειδικευμένα μεγάλα δεδομένα που έχουν να αντιμετωπίσουν. Για παράδειγμα, η πρόβλεψη άφιξης τουριστών (Hopken et al. 2021) χρησιμοποιεί έναν συνδυασμό αυτοπαλίνδρομου κινητού μέσου (ARIMA) με Νευρωνικά Δίκτυα οπίσθιας διάδοσης.

Βαθιά Νευρωνικά Δίκτυα
Τα βαθιά νευρωνικά δίκτυα είναι χρήσιμα στην εύρεση των σύνθετων και μη – γραμμικών αναπαραστάσεων που υπάρχουν στα μεγάλα δεδομένα, το οποίο είναι ανέφικτο από παραδοσιακούς αλγόριθμους της μηχανικής μάθησης, οι οποίοι μαθαίνουν με «ρηχή μάθηση» (Shallow Learning). Τα βαθιά νευρωνικά δίκτυα συνδράμουν με μεγάλη επιτυχία στις περιπτώσεις όπου έχουμε ακατέργαστα δεδομένα, δίχως επισήμανση και κατηγοριοποίηση (Najafabadi et al. 2015) (Liu et al. 2017), δηλαδή αντιμετωπίζουν επιτυχώς τον όγκο και την ποικιλομορφία από τα χαρακτηριστικά των μεγάλων δεδομένων.
Οι πιο συνηθισμένοι αλγόριθμοι των βαθιών νευρωνικών δικτύων (Deep Neural Networks, DNN) που βρίσκουν εφαρμογή σε μεγάλα δεδομένα αφορούν τους στοιβαγμένους αυτόματους κωδικοποιητές (Stacked Auto-Encoders), τα δίκτυα βαθιάς πεποίθησης (Deep Belief Networks) τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks) καθώς και τα επαναλαμβανόμενα νευρωνικά δίκτυα (Recurrent Neural Networks) (Zhang et al. 2018). Επίσης, είναι συνηθισμένο να χρησιμοποιούνται παραλλαγές των παραπάνω ή και συνδυασμοί τους με άλλους αλγόριθμους της μηχανικής μάθησης (Chiroma et al. 2019).

Αναφέρονται σε ικανοποιητικά αποτελέσματα στην βιβλιογραφία, σε διεργασίες όπως η δημιουργία σημασιολογικών δεικτών (Semantic Indexing) των δεδομένων, η αυτοματοποιημένη επισήμανση των δεδομένων, η γρήγορη ανάκτηση πληροφορίας, η εξαγωγή σύνθετων μοτίβων και η απλοποίηση προβλημάτων διαχωρισμού (Najafabadi et al. 2015). Πρόσφατα, οι αλγόριθμοι της βαθιάς μάθησης βρίσκουν εφαρμογή στην πρόβλεψη της χρήσης ηλεκτρικής ενέργειας σε ηλεκτρικά πλέγματα (Devaraj et al. 2021).

Συνελικτικά Νευρωνικά Δίκτυα

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks, CNN) χρησιμοποιούν την δομή των νευρωνικών δικτύων οπίσθιας διάδοσης, αλλά περιλαμβάνουν στοιβάδες όπου γίνεται περαιτέρω επεξεργασία των δεδομένων όπως η συνέλιξη, η ομαδοποίηση, το παραγέμισμα και ο βηματισμός (convolution, pooling, padding, striding) με σκοπό την διευκόλυνσης της μάθησης και την μείωση του υπολογιστικού κόστους. Έχουν πολλές σύγχρονες εφαρμογές στα μεγάλα δεδομένα. Για παράδειγμα, προτείνεται η ανάλυση μεγάλων δεδομένων για το ίντερνετ των πραγμάτων, με ψηφιακά δίδυμα μίας έξυπνης πόλης, από συνελικτικό νευρωνικό δίκτυο με χρήση παράλληλης μάθησης (Li et al. 2022). Επιπλέον, έχει προταθεί η ανάλυση συναισθήματος μεταξύ τομέων με βάση ένα συνελικτικό νευρωνικό δίκτυο που αξιοποιεί μάθηση με ήμι – επίβλεψη (Lee et al. 2022), ενώ μία ακόμα εφαρμογή της ανάλυσης συναισθήματος για μεγάλα δεδομένα αφορά τον συνδυασμό με LSTM νευρωνικά δίκτυα για κριτικές σε ποικιλόμορφους τομείς (Behera et al. 2021).

Επαναλαμβανόμενα Νευρωνικά Δίκτυα

Τα συνηθισμένα μοντέλα βαθιάς μάθησης δεν λαμβάνουν υπόψη την πιθανή χρονική συσχέτιση των δεδομένων, όπως για παράδειγμα σε δεδομένα που προέρχονται από μία χρονοσειρά (Najafabadi et al. 2015). Τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks, RNN) είναι ένα μοντέλο πρόβλεψης το οποίο χρησιμοποιείται κυρίως για δεδομένα σε αλληλουχία, με εφαρμογές στην ανάλυση συναισθήματος, στην κατηγοριοποίηση κειμένου, στην μετατροπή κειμένου σε ομιλία και σε μεταφράσεις μεταξύ διάφορων γλωσσών (Hammou et al. 2020). Ταυτόχρονα όμως, η χρήση τους θα ήταν συνετή σε περιπτώσεις όπου το παρελθόν μπορεί να αξιοποιηθεί για την πρόβλεψη του μέλλοντος, όπως στην πρόβλεψη της θερμοκρασίας. Έχουν τις περισσότερες εφαρμογές τους στην βαθιά μάθηση μέσω του «κεντροποιημένου» μοντέλου. (Chiroma et al. 2019). Η πρόβλεψη ασθενειών που πραγματοποίησαν οι Talasila et al., χρησιμοποιεί RNN αφού πρώτα έκαναν επιλογή των χαρακτηριστικών με διαφορετική μέθοδο. Οι Hammou et al. (2020) προτείνουν μία παραλλαγή των RNN για μεγάλα δεδομένα σε μορφή κειμένου που παράγονται από τα μέσα κοινωνικής δικτύωσης, με σκοπό την ανάλυση συναισθήματος σε πραγματικό χρόνο. Εφαρμογές στα μεγάλα δεδομένα περιλαμβάνουν την μοντελοποίηση των πιστωτικών μονάδων (credit score) για πελάτες τράπεζας, όπου οι Ala'raj et al (2021) αξιοποίησαν ένα διπλής κατεύθυνσης LSTM για την πρόβλεψη μίας ή πολλών αποτυχημένων πληρωμών από τους πελάτες. Εφαρμογή των διπλής κατεύθυνσης LSTM υπάρχει και για την απόδοση των καλλιέργειών βασισμένη σε μεγάλα δεδομένα που περιλάμβαναν την ποσότητα άρδευσης, δεδομένα για το κλίμα, και την περιεκτικότητα του εδάφους σε νερό, πετυχαίνοντας $R^2 \in (97{,}99)$ (Alibabaei et al. 2021).

Νευρωνικά Δίκτυα με Συναρτήσεις Ακτινικής Βάσης

Τα νευρωνικά δίκτυα με συναρτήσεις ακτινικής βάσης (Radical Basis Function, RBF) διαφέρουν ως προς την επιλογή της συνάρτησης ενεργοποίησης στις κρυφές στοιβάδες. Αυτή είναι μία συνάρτηση ακτινικής βάσης, η οποία εξαρτάται μόνο από την απόσταση μεταξύ ενός σημείου του διανυσματικού χώρου $x \in V$ και ενός σταθερού σημείου c, δηλαδή μία συνάρτηση $f \in [0, +\infty)$ σε συνδυασμό με μία μετρική $\|\cdot\| \in V \to [0, +\infty)$ τέτοια ώστε

$$f(x) = f_0(||x - c||)$$

Έχει αρκετές εφαρμογές σε μεγάλα δεδομένα, για παράδειγμα στην έρευνα του He (2021) αξιοποιήθηκε ένας συνδυασμός από RBFNN και DNN για την πρώιμη πρόληψη τραυματισμών σε αθλήματα. Στην έρευνα του Chen (2017), χρησιμοποιείται μία βελτιστοποίηση των RBF NN, για την πρόβλεψη της κίνησης

των αυτοκινήτων. Στην έρευνα των Jiang & Li (2019), προτείνεται ένας συνδυασμός RBF NN και συσταδοποίησης για την κατηγοριοποίηση μεγάλων δεδομένων σχετικά με την υγεία. Οι Lu et al. (2016) χρησιμοποίησαν ένα RBF νευρωνικό δίκτυο για την βραχυπρόθεσμη πρόβλεψη φόρτωσης σε έξυπνα πλέγματα. Ο Wu (2021) χρησιμοποίησε ένα RBF νευρωνικό δίκτυο για την βελτιστοποίηση και προσομοίωση διαχείρισης από πόρους εταιριών.

Νευρωνικά Δίκτυα με μάθηση διανυσμάτων κβαντισμού

Η μάθηση διανυσμάτων κβαντισμού (Learning Vector Quantization, LVQ) στα νευρωνικά δίκτυα είναι μία υποκατηγορία των νευρωνικών δικτύων πολυστρωματικών αντιλήπτρων. Χρησιμοποιεί μία ανταγωνιστική στοιβάδα και έπειτα μία γραμμική στοιβάδα. Η ανταγωνιστική στοιβάδα λειτουργεί σε μεγάλη ομοιότητα με τον αυτό – οργανωμένου χάρτη του Kohonen. (Kohonen, 1995). Το 2021, οι Nilashi et al., χρησιμοποίησαν LVQ Νευρωνικά Δίκτυα για την ανάλυση μεγάλων κοινωνικών δεδομένων από την ποιότητα του φαγητού στην ικανοποίηση των ταξιδιωτών σε ξενοδοχεία. Οι Sanober και Rani (2021) αξιοποίησαν ένα νευρωνικό δίκτυο με μαθητευόμενα διανύσματα κβαντισμού για να προβλέψουν την ποιότητα του αέρα. Ο Tongke (2021) χρησιμοποίησε LVQ νευρωνικά δίκτυα για την ανίχνευση ασθενειών και παρασίτων σε εικόνες από φύλλα φυτών. Οι Zheng (2021) χρησιμοποίησαν τα LVQ νευρωνικά δίκτυα για τον εντοπισμό απάτης στην λογιστική.

Βελτιστοποίηση Νευρωνικών Δικτύων μέσω αλγόριθμων νέφους, εξέλιξης και γενετικών αλγόριθμων.

Οι Chiroma et al. (2019), αναφέρουν την ανάγκη για την αξιοποίηση βιολογικά – εμπνευσμένων αλγορίθμων για την βελτιστοποίηση των παραμέτρων, καθώς αυτοί είναι ανθεκτικοί και μεταβάλλονται κατά την μάθηση, το οποίο προτείνει πως δρουν καλά σε προβλήματα όπως οι εκρηγνυόμενες κλίσεις. Επιπλέον αυτοί μπορούν να αξιοποιηθούν για την επιλογή των υπερπαραμέτρων των νευρωνικών δικτύων, οι οποίες περιλαμβάνουν την αρχιτεκτονική (πλήθος κόμβων, πλήθος στοιβάδων) καθώς και τους διάφορους τρόπους μάθησης. Στην έρευνά τους, είχε βρεθεί και ένας μικρός αριθμός ερευνών οι οποίοι αξιοποίησαν αλγόριθμους σμήνους και εξελικτικούς αλγόριθμους για την βελτιστοποίηση των παραμέτρων των νευρωνικών δικτύων, με εφαρμογές σε μεγάλα δεδομένα (Chiroma et al. 2019). Όμως υπάρχουν σύγχρονες μελέτες οι οποίες εκμεταλλεύονται εξελικτικούς αλγόριθμους για την βελτιστοποίηση των παραμέτρων, για την επιλογή των υπερπαραμέτρων των νευρωνικών δικτύων (Zhang et al. 2021) καθώς και για την επιλογή χαρακτηριστικών των δεδομένων στα οποία θα εκπαιδευτεί το νευρωνικό δίκτυο. Για παράδειγμα, ο αλγόριθμος σμήνους βελτιστοποιεί τα νευρωνικά δίκτυα που αξιοποιούνται για τα μεγάλα δεδομένα, μέσα από την κατηγοριοποίηση των σταδίων στα όνειρα (Surantha et al. 2021), από την ανάλυση συναισθήματος μέσω του Twitter (Sreenivasulu & Jayakarthik, 2021) και την πρόβλεψη καρδιακών νόσων (Nandy et al. 2021). Επιπλέον, αξιοποιήθηκαν LSTM για την πρόβλεψη της θερμοκρασίας, όπου η επιλογή των παραμέτρων έγινε με τον εξελικτικό αλγόριθμο του νέφους (Patil et al. 2021), καθώς και RBF νευρωνικά δίκτυα με βελτιστοποίηση σμήνους σωματιδίων για την πρόβλεψη της κυκλοφορίας σε μεγάλα δεδομένα διαδικτύου (He, 2016).

Κβαντικά Νευρωνικά Δίκτυα

Τα κβαντικά νευρωνικά δίκτυα (quantum neural networks, QNN) (Kak, 1995) είχαν δύσκολη υλοποίηση. Έγιναν νέες προτάσεις (Schuld et al. 2014) και πλέον μπορούν να εκμεταλλευτούν τους σύγχρονους υπολογιστές και ως συνέπεια να βγουν στην παραγωγή (Cong et al, 2019). Αυτά έχουν αυξημένη ταχύτητα και επιδόσεις σε σύγκριση με τα συμβατικά, μειώνοντας την απαίτηση από πράξεις, με εφαρμογές στην κατηγοριοποίηση εικόνας (Lu et al. 2021). Επιπλέον, έχουν βρει εφαρμογές σε κατηγοριοποίηση μεγάλων δεδομένων Ακόμα, ο συνδυασμός συνελικτικών και κβαντικών νευρωνικών δικτύων βρέθηκε να έχει εξαιρετικές επιδόσεις, όπως η ανθεκτικότητα σε θόρυβο στα δεδομένα, ο οποίος θεωρείται πως θα επιτρέψει την τα κβαντικά συνελικτικά νευρωνικά δίκτυα (QCNN) να συνδυαστούν με μεγάλα δεδομένα (Wei et al. 2021). Επιπλέον, ένας συνδυασμός συνελικτικών νευρωνικών δικτύων με κβαντικά νευρωνικά δίκτυα για κατηγοριοποίηση εικόνας παρουσιάζεται στους Zheng, 2021. Μία εισαγωγή της κβαντικής μηχανικής μάθησης βρίσκεται στο άρθρο των Martin-Guerrero & Lamata (2022).

Ενότητα 2.

Προκλήσεις στην Αξιοποίηση των Μεγάλων Δεδομένων μέσω των Νευρωνικών Δικτύων

Γενικά, οι προκλήσεις των μεγάλων δεδομένων αφορούν άμεσα τα χαρακτηριστικά τους όπως ο όγκος, η ταχύτητα, η ποικιλομορφία η αξία και η ακρίβεια, αλλά δεν περιορίζονται μόνο σε αυτά: απαιτούνται επιπλέον οι τεχνικές για την λήψη, τον μετασχηματισμό, την ενσωμάτωση και την μοντελοποίηση των δεδομένων, ενώ απαιτούνται επιπλέον ενέργειες για την διασφάλιση της ασφάλειας και του απόρρητου (Czarnowski et al. 2018). Αυτά απαιτούν νέες τεχνολογίες και εργαλεία, τα οποία με μία βλάβη μπορεί να αναπαράγουν λάθος δεδομένα συνεπώς και αναλύσεις.

Το 2019 οι Chiroma et al. πραγματοποίησαν μία επισκόπηση σχετικά με την προσέγγιση των νευρωνικών δικτύων στην αναλυτική των μεγάλων δεδομένων. Ανέφεραν γενικά προβλήματα, όπως η ανάγκη και η δυσκολία της προεπεξεργασίας των μεγάλων δεδομένων, ώστε να γίνει η εκμετάλλευση της αξίας τους, πριν την τροφοδότησή τους στα νευρωνικά δίκτυα, με μεθόδους όπως η μείωση της διάστασης, η κανονικοποίηση, η αντιμετώπιση των ελλειπουσών τιμών των δεδομένων, η διεξαγωγή αναλύσεων συσχέτισης και άλλα. Με αυτό τον τρόπο, αναφέρουν πως μπορεί να αποφευχθεί η χαμηλή ποιότητα των μεγάλων δεδομένων. Επιπλέον, αναφέρουν πως η έρευνα για την παραγωγή νέων μοντέλων νευρωνικών δικτύων συνηθίζεται να πραγματοποιείται με μικρά σύνολα δεδομένων, το οποίο περιορίζει τις δυνατότητες της έκτασης των νευρωνικών δικτύων στον τομέα των μεγάλων δεδομένων. Επιπρόσθετα, οι περισσότερες *τελευταίας τεχνολογίας* εφαρμογές (το 2019) νευρωνικών δικτύων σε μεγάλα δεδομένα, αφορούσαν βαθιά MLP και CNN, τα οποία αξιοποίησαν κυρίως τον αλγόριθμο της οπίσθιας διάδοσης για την μάθηση. Αναφέρουν πως η εφαρμογή τους στην αναλυτική των μεγάλων δεδομένων περιορίζεται από τα ακόλουθα προβλήματα:

- Το μεγάλο πλήθος των παραμέτρων που υπόκεινται σε μάθηση, το οποίο οδηγεί σε μεγάλο κόστος σε μνήμη κατά την διάρκεια των ταυτόχρονων υπολογισμών.
- Την αργή σύγκλιση λόγω των πολλών τοπικών ελαχίστων που προκύπτουν.
- Την δυσκολία αξιοποίησης παράλληλης μάθησης λόγω της διαδοχικής φύσης που πραγματοποιείται ο αλγόριθμος της οπίσθιας διάδοσης.
- Τις εξαφανιζόμενες / εκρηγνυόμενες κλίσεις που μπορεί να προκύψουν στην μάθηση.
- Την ανάγκη και το κόστος από ηλεκτρική ενέργεια κατά την διεξαγωγή των υπολογισμών. Επίσης αναφέρεται πως δεν είχαν αναφερθεί ενδεχόμενες ενέργειες για την μείωση των αναγκών σε ηλεκτρισμό σε καμία έρευνα από αυτές της επισκόπησης.

Επιπρόσθετα, στην έρευνά τους το 2021, οι Ikhlasse et al. βρήκαν πως μόλις το 10% των συμμετεχόντων χρησιμοποιούσαν μετρικές για την ενέργεια που καταναλώνουν οι αλγόριθμοι των νευρωνικών δικτύων τους καθώς δούλευαν σε μεγάλα δεδομένα, ενώ προτείνουν μία μεθοδολογία πράξεων για την καλύτερη αξιοποίηση του λογισμικού και των δεδομένων (Ikhlasse et al. 2021).

Είναι αξιοσημείωτο πως ο Chollet (2018) αναφέρει στο βιβλίο του *Deep Learning with R* σε συνεργασία με τον Allaire, πως υπάρχουν 2 βασικές προϋποθέσεις πίσω από κάθε εφαρμογή της μηχανικής μάθησης:

1. *Η πληροφορία που υπάρχει στα δεδομένα εξόδου είναι ικανή να βρεθεί από τα δεδομένα εισόδου.*
2. *Τα δεδομένα εισόδου περιέχουν αρκετή πληροφορία ώστε να μπορεί να είναι δυνατή η μάθηση της σχέσης μεταξύ των δεδομένων εισόδου και των δεδομένων εξόδου*.

Συνεπώς, μία συνεχόμενη πρόκληση της εφαρμογής των νευρωνικών δικτύων στα μεγάλα δεδομένα, είναι η πληρότητα των χαρακτηριστικών των μεγάλων δεδομένων. Οφείλουν να διατηρούν την αξία και την ακρίβειά τους, παρά την ποικιλομορφία, τον όγκο, την ταχύτητα, την αστάθεια και την μεταβλητότητά τους, ειδάλλως τα νευρωνικά δίκτυα θα μαθαίνουν μονάχα τον θόρυβο στα δεδομένα, δίχως να απαντούν σε κάποιο ερευνητικό ερώτημα.

Ενότητα 3.

Συμπεράσματα

Σε αυτή τη διπλωματική εργασία, σκοπός είναι η παρουσίαση της σύνδεσης των νευρωνικών δικτύων με τα μεγάλα δεδομένα. Για να το πετύχουμε αυτό, αρχικά αναφέραμε το θεωρητικό υπόβαθρο των μεγάλων δεδομένων, με εφαρμογές και τα χαρακτηριστικά τους, τα οποία τα καθιστούν δύσκολα στον χειρισμό και την αξιοποίηση. Έπειτα, αναφέραμε τις βασικές έννοιες της μηχανικής μάθησης και το θεωρητικό πλαίσιο που γέννησε την πιο απλή μορφή από τα σύγχρονα νευρωνικά δίκτυα. Μετά εστιάσαμε στις πιο γνωστές και δημοφιλείς αρχιτεκτονικές νευρωνικών δικτύων, όπως τα MLP, τα CNN και τα RNN, αναλύοντας τις αρχιτεκτονικές τους και την διαδικασία με την οποία αυτά «μαθαίνουν», ενώ αναφέραμε τις δυσκολίες που συναντούν κατά τη μάθηση, όπως το πρόβλημα των εξαφανιζόμενων και εκρηγνυόμενων κλίσεων, αλλά και στις τεχνικές οι οποίες αξιοποιούνται για την αντιμετώπισή τους. Δείξαμε με εφαρμογές τις δυσκολίες της εκπαίδευσης και της τελικής επιλογής ενός μοντέλου νευρωνικών δικτύων με την χρήση της γλώσσας R. Τότε, έχοντας αναφέρει ατομικά τις δυσκολίες στην αξιοποίηση των μεγάλων δεδομένων και στην μάθηση των νευρωνικών δικτύων, παρουσιάσαμε τα νευρωνικά δίκτυα και την σύνδεσή τους με τα μεγάλα δεδομένα μέσα από κάποιες εφαρμογές τους, και προσθέσαμε τις δυσκολίες που προκύπτουν. Καταλήγοντας, τα μεγάλα δεδομένα αποτελούν ένα τομέα στον οποίο μόλις τώρα αρχίζουμε να εξερευνούμε την πλήρη έκταση των δυνατοτήτων του. Οι εφαρμογές των νευρωνικών δικτύων στα μεγάλα δεδομένα συνεχίζουν να πληθαίνουν με την μέρα, ενώ προτείνονται συνεχώς νέες ιδέες οι οποίες καλύπτουν προηγούμενες δυσκολίες, όπως τα κβαντικά νευρωνικά δίκτυα και η βελτιστοποίηση μέσω εξελικτικών αλγόριθμων. Ταυτόχρονα, τα νευρωνικά δίκτυα βρίσκουν εφαρμογές σε όλους τους τομείς των μεγάλων δεδομένων όπως οι έξυπνες πόλεις, η πρόβλεψη των καιρικών φαινομένων και η δημιουργία ψηφιακών διδύμων. Έως σήμερα, ο συνδυασμός των νευρωνικών δικτύων με τα μεγάλα δεδομένα αποφέρει μία ιδιαίτερα αποτελεσματική και καρποφόρα μέθοδο για την εκμετάλλευσή τους.

Ενότητα 4.

Μελλοντική Έρευνα

Κλείνοντας, υπενθυμίζουμε πως τα νευρωνικά δίκτυα αποτελούν μία σειρά από συνεχείς γεωμετρικούς μετασχηματισμούς πάνω στα δεδομένα εισόδου, με σκοπό να βρουν μία απεικόνιση σε έναν χώρο που είναι παρόμοιος με αυτόν των δεδομένων εξόδου. Αποτελούν, δηλαδή, μία απεικόνιση $f: X \to Y$. Μπορεί να πετυχαίνουν μεγάλα ποσοστά ακρίβειας ως προς την κατηγοριοποίηση μίας εικόνας, όμως *δεν έχουν την ικανότητα της αφηρημένης σκέψης*, δεν μπορούν να μάθουν οδηγίες για να τρέξουν ένα πρόγραμμα ή να οδηγήσουν ένα αυτοκίνητο με την πρώτη προσπάθεια. Θα πρέπει να επαναλάβουν την διαδικασία πολλές φορές μέχρι να οδηγήσουν το αυτοκίνητο, ενώ εάν αλλάξουν μέρος οδήγησης η διαδικασία θα πρέπει να ξεκινήσει από την αρχή (Chollet, 2018).

Μελλοντικές εφαρμογές των νευρωνικών δικτύων θα πρέπει να εστιάσουν προς την εισαγωγή της αφηρημένης σκέψης στα νευρωνικά δίκτυα, το οποίο θα αυξήσει τις πιθανές διεργασίες που μπορούν να πραγματοποιηθούν στα μεγάλα δεδομένα. Επιπλέον, αρκετές από τις έρευνες που αναφέραμε, αξιοποιούν έναν συνδυασμό μεθόδων ή τομέων, ώστε να έχουν αυξημένη απόδοση. Συνεπώς, προτείνουμε στους ερευνητές να συνεχίσουν την συνεργασία με άτομα σε διαφορετικούς τομείς εξειδίκευσης. Επιπρόσθετα, θα πρέπει να συνεχίσουν οι έρευνες για την επίλυση των ατομικών προβλημάτων των νευρωνικών δικτύων και των μεγάλων δεδομένων, ώστε να διευκολυνθεί η κοινή τους χρήση. Τέλος, για να είναι δυνατό να προχωρήσει η έρευνα των μεγάλων δεδομένων με τα νευρωνικά δίκτυα, θα πρέπει να δημιουργηθούν νέες τεχνολογίες και λογισμικά, οι οποίες να επιτρέπουν ευκολότερη επεξεργασία και πρόσβαση των μεγάλων δεδομένων στον κάθε ερευνητή.

Βιβλιογραφία

Ala'raj, M., Abbod, M. F., & Majdalawieh, M. (2021). Modelling customers credit card behaviour using bidirectional LSTM neural networks. *Journal of Big Data*, *8*(1), 1-27. https://doi.org/10.1186/s40537-021-00461-7

Alibabaei, K., Gaspar, P. D., & Lima, T. M. (2021). Crop Yield Estimation Using Deep Learning Based on Climate Big Data and Irrigation Scheduling. *Energies*, *14*(11), 3004. https://doi.org/10.3390/en14113004

Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., & Taha, K. (2015). Efficient machine learning for big data: A review. *Big Data Research*, *2*(3), 87-93. https://doi.org/10.1016/j.bdr.2015.04.001

Allaire JJ. (2017). *RStudio AI Blog: Keras for R.* https://blogs.rstudio.com/tensorflow/posts/2017-09-06-keras-for-r/

Allaire JJ. & Chollet F. (2021). *keras: R Interface to 'Keras'*. R package version 2.7.0. https://CRAN.R-project.org/package=keras

Bache S. M., Wickham H., Lionel H. (2017). https://CRAN.R-project.org/package=magrittr

Bagherzadeh, S., Maghooli, K., Shalbaf, A. *et al.* (2021). Emotion recognition using effective connectivity and pre-trained convolutional neural networks in EEG signals. *Cogn Neurodyn*. https://doi.org/10.1007/s11571-021-09756-0

Barrett, D. G. T., Morcos, A. S., Macke, J. H. (2019). Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current Opinion in Neurobiology*, *55*, 55–64. https://doi.org/10.1016/j.conb.2019.01.007

Basodi, S., Ji, C., Zhang, H., & Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, *3*(3), 196-207. https://doi.org/10.26599/BDMA.2020.9020004

Behera, R. K., Jena, M., Rath, S. K., & Misra, S. (2021). Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data. *Information Processing & Management*, 58(1), 102435. https://doi.org/10.1016/j.ipm.2020.102435

Bjorck, J., Gomes, C., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *arXiv preprint arXiv:1806.02375*.

Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, *14*, 045-076. https://doi.org/10.28945/4184

Cao, J., Cui, H., Shi, H., & Jiao, L. (2016). Big Data: A Parallel Particle Swarm Optimization-Back-Propagation Neural Network Algorithm Based on MapReduce. *PLoS One. 11*(6), e0157551. https://doi.org/10.1371/journal.pone.0157551

Ceci L. (2021). *Hours of video uploaded to YouTube every minute 2007-2020.* Λήφθηκε 27 Γενάρη, 2022 από https://www.statista.com/

Chen, D. (2017). Research on Traffic Flow Prediction in the Big Data Environment Based on the Improved RBF Neural Network. *In IEEE Transactions on Industrial Informatics*, *13*(4),2000-2008. https://doi.org/10.1109/TII.2017.2682855

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

Chiroma, H., Abdullahi, U. A., Abdulhamid, S. M., Abdulsalam Alarood, A., Gabralla, L. A., Rana, N., Shuib, L., Targio Hashem, I. A., Gbenga, D. E., Abubakar, A. I., Zeki, A. M., & Herawan, T. (2019). Progress on Artificial Neural Networks for Big Data Analytics: A Survey. *IEEE Access*, *7*. https://doi.org/10.1109/ACCESS.2018.2880694

Chollet, F., & Others. (2015). Keras. https://keras.io

Chollet, F., & Allaire, J. J. (2018). Deep learning with R. *Manning Publications*. ISBN: 9781617295546

Cong, I., Choi, S. & Lukin, M.D. (2019) Quantum convolutional neural networks. *Nat.Phys.* **15,** 1273–1278. https://doi.org/10.1038/s41567-019-0648-8

Czarnowski, I., Jedrzejowicz, P., Chao, K. M., & Yildirim, T. (2018). Overcoming "Big Data" Barriers in Machine Learning Techniques for the Real-Life Applications. *Complexity, 2018*. https://doi.org/10.1155/2018/1234390

Dai, Z., & Heckel, R. (2019). Channel normalization in convolutional neural network avoids vanishing gradients. *arXiv preprint arXiv:1907.09539*.

Debnath, S., Roy, R., & Changder, S. (2022). Photo classification based on the presence of diagonal line using pre-trained DCNN VGG16. *Multimedia Tools and Applications*, 1-22. https://doi.org/10.1007/s11042-021-11557-w

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6), 141–142. https://doi.org/10.1109/MSP.2012.2211477

Dertat A. (2017) "Applied Deep Learning - Part 4: Convolutional Neural Networks" Λήφθηκε στις 20 Γενάρη 2022, από την ιστοσελίδα towardsdatascience.com.

Devaraj, J., Madurai Elavarasan, R., Shafiullah, G. M., Jamal, T., & Khan, I. (2021). A holistic review on energy forecasting using big data and deep learning models. *International Journal of Energy Research*. https://doi.org/10.1002/er.6679

Dwarampudi, M., & Reddy, N. V. (2019). Effects of padding on LSTMs and CNNs. *arXiv preprint arXiv:1903.07288*.

Essiet, I., Sun, Y., & Wang, Z. (2019). Big data analysis for gas sensor using convolutional neural network and ensemble of evolutionary algorithms. *Procedia Manufacturing, 35*, 629- 634. https://doi.org/10.1016/j.promfg.2019.06.005

Fan, T. (2021). Research on Edge Detection of Agricultural Pest and Disease Leaf Image Based on LVQ Neural Network. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 14*(6), 1903-1911. https://doi.org/10.2174/2666255813666191218112436

Fathi, M., Haghi Kashani, M., Jameii, S. M., & Mahdipour, E. (2021). Big data analytics in weather forecasting: A systematic review. *Archives of Computational Methods in Engineering*, 1-29. https://doi.org/10.1007/s11831-021-09616-4

Feng, J., & Lu, S. (2019). Performance analysis of various activation functions in artificial neural networks. Journal of Physics Conference Series, 1237(2), 0022030. https://doi.org/10.1088/1742-6596/1237/2/022030

Funahashi, K-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks, 2*(3), 183-192.https://doi.org/10.1016/0893-6080(89)90003-8

Geczy, P. (2014). Big Data Characteristics. *The Macrotheme Review: A Multidisciplinary Journal of Global Macro Trends, 3*(6), 94-104.

Geron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. *O'Reilly Media*. ISBN: 9781492032595

Ghorbani, B., Mei, S., Misiakiewicz, T., & Montanari, A. (2021). When do neural networks outperform kernel methods? *Journal of Statistical Mechanics: Theory and Experiment*, *2021*(12), 124009. https://doi.org/10.1088/1742-5468/ac3a81

Goodfellow, I., & Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems, 28*(10), 2222-2232. https://doi.org/10.1109/tnnls.2016.2582924

Guzhva, A., Dolenko, S., & Persiantsev, I. (2009, September). Multifold acceleration of neural network computations using GPU. In *International Conference on Artificial Neural Networks* (pp. 373 -380). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04274-4_39

Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, *18*(4), 2715-2743.

https://www.archives.palarch.nl/index.php/jae/article/view/6705
Hammou, B. A., Lahcen, A. A., & Mouline, S. (2020). Towards a real-time processing framework based on improved distributed recurrent neural network variants with fastText for social big data analytics. *Information Processing & Management, 57*(1), 102122 https://doi.org/10.1016/j.ipm.2019.102122
Hanin, B. (2018). Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*.
He, F. (2021). Early Warning Model of Sports Injury Based on RBF Neural Network Algorithm. *Complexity, 2021*. https://doi.org/10.1155/2021/6622367
He, T., Dan, T., Wei, Y., Li, H., Chen X., & Qin, G. (2016). Particle Swarm Optimization RBF Neural Network Model for Internet Traffic Prediction. *2016 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pp. 521-524, https://doi.org/10.1109/ICITBS.2016.146
Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. N*eural computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
Hopken, W., Eberle, T., Fuchs, M., & Lexhagen, M. (2021). Improving tourist arrival prediction: a big data and artificial neural network approach. Journal of Travel Research, 60(5), 998-1017. https://doi.org/10.1177%2F0047287520921244
Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*(2), 1. https://doi.org/10.5121/ijdkp.2015.5201
Hui, L., & Belkin, M. (2020). Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*.
IBM (2020). *Data Analytics and AI are empowering transformation, Communication Service Providers*. Λήφθηκε 28 Δεκεμβρίου, 2021 από https://www.ibm.com/downloads/cas/NDDP7VAP
IBM (2020). *It's a new era in advanced analytics and AI, Financial Services Companies*. Λήφθηκε 28 Δεκεμβρίου, 2021 από https://www.ibm.com/downloads/cas/BKGDLGP8
IBM (2020). *Welcome to the revolution in healthcare, Big Data, Advanced Analytics and AI*. Λήφθηκε 28 Δεκεμβρίου, 2021 από https://www.ibm.com/downloads/cas/VGP23NDZ
Ikhlasse, H., Benjamin, D., Vincent, C., & Hicham, M. (2021). Recent implications towards sustainable and energy efficient AI and big data implementations in cloud-fog systems: A newsworthy inquiry. *Journal of King Saud University-Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2021.11.002
Inoue, M., Futamura, M., & Ninomiya, H. (2020). No one-hidden-layer neural network can represent multivariable functions. *arXiv preprint arXiv:2006.10977*.
Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR: 37:448-456
Jiang, C., & Li, Y. (2019). Health Big Data Classification Using Improved Radial Basis Function Neural Network and Nearest Neighbor Propagation Algorithm. In *IEEE Access, 7*, 176782-176789. https://doi.org/10.1109/ACCESS.2019.2956751
Jung, J. W., Shim, H. J., Kim, J. H., & Yu, H. J. (2021). DCASENet: An integrated pretrained deep neural network for detecting and classifying acoustic scenes and events. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 621-625). IEEE. https://doi.org/10.1109/ICASSP39728.2021.9414406
Kaggle DogVCat Competition: http://www.kaggle.com/c/dogs-vs-cats
Kak, S. C. (1995). Quantum neural computing. *Advances in imaging and electron physics, 94*, 259-313. https://doi.org/10.1016/S1076-5670(08)70147-2

Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems, 1*(4), 111–122.

Keydana & Kalinowski (2021, Nov. 18). *RStudio AI Blog: Keras for R is back!* Λήφθηκε στις 2 Ιανουαρίου, 2022, από https://blogs.rstudio.com/tensorflow/posts/2021-11-18-keras-updates/

Kleinberg, B., Li, Y., & Yuan, Y. (2018). An alternative view: When does SGD escape local minima? In *International Conference on Machine Learning* (pp. 2698-2707). PMLR 80:2698-2707.

Ko, Y. M., & Ko, S. W. (2021). Alleviation of Vanishing Gradient Problem Using Parametric Activation Functions. *KIPS Transactions on Software and Data Engineering*, *10*(10), 407–420. https://doi.org/10.3745/KTSDE.2021.10.10.407

Kohonen, T. (1995), Learning vector quantization. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 537–540), Cambridge, MA: MIT Press. https://doi.org/10.1007/978-3-642-97610-0_6

Lanham, M. (2018). *Learn ARCore-Fundamentals of Google ARCore: Learn to build augmented reality apps for Android, Unity, and the web with Google ARCore 1.0*. Packt Publishing Ltd. ISBN: 9781788833639

Le Cun, Y., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., & Henderson, D. (1990). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems 2*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(1), 2278-2324. https://doi.org/10.1109/5.726791

Lee, L. K., Chui, K. T., Wang, J., Fung, Y. C., & Tan, Z. (2022). An Improved Cross-Domain Sentiment Analysis Based on a Semi-Supervised Convolutional Neural Network. In *Data Mining Approaches for Big Data and Sentiment Analysis in Social Media* (pp. 155-170). IGI Global. https://doi.org/10.4018/978-1-7998-8413-2.ch007

Li, X., Liu, H., Wang, W., Zheng, Y., Lv, H., & Lv, Z. (2022). Big data analysis of the internet of things in the digital twins of smart city based on deep learning. *Future Generation Computer Systems*, *128*, 167-177. https://doi.org/10.1016/j.future.2021.10.006

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11-26. https://doi.org/10.1016/j.neucom.2016.12.038

Lopez-Miguel, I. D. (2021). Survey on Preprocessing Techniques for Big Data Projects. *Engineering Proceedings*, *7*(1), 14. https://doi.org/10.3390/engproc2021007014

Lu, Y., Gao, Q., Lü, J., Ogorzałek, M., & Zheng, J. (2021). A Quantum Convolutional Neural Network for Image Classification. In *2021 40th Chinese Control Conference (CCC)* (pp. 6329-6334). IEEE. https://doi.org/10.23919/CCC52363.2021.9550027

Lu, Y., Zhang, T., Zeng, Z., & Loo, J. (2016). An improved RBF neural network for short-term load forecast in smart grids. *2016 IEEE International Conference on Communication Systems (ICCS)*, pp. 1-6. https://doi.org/10.1109/ICCS.2016.7833643

Macpherson, T., Churchland, A., Sejnowski, T., DiCarlo, J., Kamitani, Y., Takahashi, H., & Hikida, T. (2021). Natural and Artificial Intelligence: A brief introduction to the interplay between AI and neuroscience research. *Neural Networks*, *144*, 603-613. https://doi.org/10.1016/j.neunet.2021.09.018

Martin-Guerrero, J. D., & Lamata, L. (2022). Quantum Machine Learning: A tutorial. *Neurocomputing*, *470*, 457-461. https://doi.org/10.1016/j.neucom.2021.02.102

Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. arXiv preprint arXiv:1804.07612.

Mitchell T. (1997) Machine Learning. *McGraw-Hill Education*. ISBN: 9780071154673

MSR Asirra: http://research.microsoft.com/en-us/um/redmond/projects/asirra/

Muller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python*. O'Reilly. ISBN: 9781449369415

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *In Proceedings of the 27th International Conference on Machine Learning* (pp. 807-814), Haifa, Israel. https://icml.cc/Conferences/2010/papers/432.pdf

Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M. *et al.* (2015) Deep learning applications and challenges in big data analytics. *Journal of Big Data* 2, 1. https://doi.org/10.1186/s40537-014-0007-7

Nandy, S., Adhikari, M., Balasubramanian, V. *et al.*(2021) An intelligent heart disease prediction system based on swarm-artificial neural network. *Neural Comput & Applic*. https://doi.org/10.1007/s00521-021-06124-1

Narkhede, M.V., Bartakke, P.P. & Sutaone, M.S. A review on weight initialization strategies for neural networks. *Artif Intell Rev* **55,** 291–322 (2022). https://doi.org/10.1007/s10462-021-10033-z

Nawi, N. M., Atomi, W. H., & Rehman, M. Z. (2013). The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technology*, *11*, 32-39. https://doi.org/10.1016/j.protcy.2013.12.159

Nilashi, M., Abumalloh, R. A., Almulihi, A., Alrizq, M., Alghamdi, A., Ismail, M. Y., ... & Asadi, S. (2021). Big social data analysis for impact of food quality on travelers' satisfaction in eco-friendly hotels. *ICT Express*. https://doi.org/10.1016/j.icte.2021.11.006

Noskova, E. S., Zakharov, I. E., Shkandybin, Y. N., & Rykovanov, S. G. (2022). Towards energy-efficient neural network calculations. *Computer Optics*, 46(1), 160-166. https://doi.org/10.18287/2412-6179-CO-914

NVIDIA, Vingelmann, P., & Fitzek, F. H. P. (2021). *CUDA, release: 11.2.0*. Λήφθηκε στις 2 Ιανουαρίου, 2022, από https://developer.nvidia.com/cuda-toolkit

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

Oh, K. S., & Jung, K. (2004). GPU implementation of neural networks. *Pattern Recognition*, *37*(6), 1311-1314. https://doi.org/10.1016/j.patcog.2004.01.013

Oracle, (2021) *22 Big Data Use Cases You Want to Know*, Λήφθηκε στις 20 Δεκεμβρίου, 2021, από την ιστοσελίδα https://www.oracle.com/a/ocom/docs/dc/top22usecasesforbigdatav2.pdf

Panigrahi, A., Shetty, A., & Goyal, N. (2019). Effect of activation functions on the training of overparametrized neural nets. *arXiv preprint arXiv:1908.05660*.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Patel, D., & Sastry, P. S. (2021). Memorization in Deep Neural Networks: Does the Loss Function Matter? In: Karlapalem K. et al. (Eds.), *Advances in Knowledge Discovery and Data Mining*. *PAKDD 2021. Lecture Notes in Computer Science*, *12713*. Springer, Cham. https://doi.org/10.1007/978-3-030-75765-6_11

Patil, M. M., Rekha, P. M., Solanki, A., Nayyar, A., & Qureshi, B. (2022). Big Data Analytics Using Swarm-Based Long Short-Term Memory for Temperature Forecasting. *CMC-COMPUTERS MATERIALS & CONTINUA*, *71*(2), 2347-2361. http://dx.doi.org/10.32604/cmc.2022.021447

Philipp, G., Song, D., & Carbonell, J. G. (2017). The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions. *arXiv preprint arXiv:1712.05577*.

Polanco, C. (2018). Transformations: A Mathematical Approach-Fundamental Concepts. *Bentham Science Publishers*. ISBN: 978-1-681088-711-5

Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, *2016*(1), 1-16. https://doi.org/10.1186/s13634-016-0355-x

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/

Rathore, M. M., Shah, S. A., Shukla, D., Bentafat, E., & Bakiras, S. (2021). The Role of AI, Machine Learning, and Big Data in Digital Twinning: A Systematic Literature Review, Challenges, and Opportunities. *IEEE Access*, *9*, 32030-32052. https://doi.org/10.1109/ACCESS.2021.3060863

RStudio Team (2021). RStudio: Integrated Development Environment for R. RStudio, PBC, Boston MA URL http://www.rstudio.com/.

Roodschild, M., Gotay Sardiñas, J. & Will, A. (2020). A new approach for the vanishing gradient problem on sigmoid activation. *Progress in Artificial Intelligence*, *9*, 351–360. https://doi.org/10.1007/s13748-020-00218-y

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y

Sanober S., & Usha Rani, K. (2021). A Substantial Approach to Predict Air Quality Using LVQ Neural Network. In: Jyothi S., Mamatha D. M., Zhang Y. D., & Raju K. S. (Eds.), *Proceedings of the 2nd International Conference on Computational and Bio Engineering. Lecture Notes in Networks and Systems, 215*. Springer, Singapore. https://doi.org/10.1007/978-981-16-1941-0_52

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210-229. https://doi.org/10.1147/rd.116.0601

Sarker, I.H. (2021) Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN COMPUT. SCI.* **2,** 160. https://doi.org/10.1007/s42979-021-00592-x

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, *4*(2), 234–242. https://doi.org/10.1162/neco.1992.4.2.234

Schroeck, M., Shockley, R., Smart, J., Romero-Morales, D. & Tufano, P. (2012). *Analytics: The Real-World Use of Big Data* (IBM Institute for Business Value - Executive Report). IBM Institute for Business Value. https://www.bibsonomy.org/bibtex/2bdbc7773e69483bdac56ea7bb187189a/sb3000

Schuld, M., Sinayskiy, I. & Petruccione, F. (2014) The quest for a Quantum Neural Network. *Quantum Inf Process* **13,** 2567–2586. https://doi.org/10.1007/s11128-014-0809-8

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*(11), 2673-2681. https://doi.org/10.1109/78.650093

See A. (2021). *Amount of data created, consumed, and stored 2010-2025*. Λήφθηκε 27 Γενάρη, 2022 από https://www.statista.com/

Shahrivari S. (2014). Beyond Batch Processing: Towards Real-Time and Streaming Big Data. *Computers*, *3*(4), 117-129. https://doi.org/10.3390/computers3040117

Shanmugam, J. V., Duraisamy, B., Simon, B. C., & Bhaskaran, P. (2022). Alzheimer's disease classification using pre-trained deep networks. *Biomedical Signal Processing and Control*, *71*, 103217. https://doi.org/10.1016/j.bspc.2021.103217

Shazly, K., Eid, M., & Salem, H. (2020). An Efficient Hybrid Approach for Twitter Sentiment Analysis based on Bidirectional Recurrent Neural Networks. *International Journal of Computer Applications*, *175*(17), 32-36.

Shlens, J. (2014). Notes on Kullback-Leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000.*

Sreenivasulu, T., & Jayakarthik, R. (2021). Intelligent Deep Neural Network integrated with Chaotic Particle Swarm Intelligence based Sentiment Analysis in Big Data Paradigm. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)* (Vol. 1, pp. 1721-1726). IEEE. https://doi.org/10.1109/ICACCS51430.2021.9441976

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*, 1929-1958.

Surantha, N., Lesmana, T.F. & Isa, S.M. (2021). Sleep stage classification using extreme learning machine and particle swarm optimization for healthcare big data. *J Big Data* **8,** 14. https://doi.org/10.1186/s40537-020-00406-6

Talasila, V., Madhubabu, K., Madhubabu, K., Mahadasyam, M., Atchala, N., & Kande, L. (2020). The prediction of diseases using rough set theory with recurrent neural network in big data analytics. *International Journal of Intelligent Engineering and Systems*, *13*(5), 10-18. https://doi.org/10.22266/ijies2020.1031.02

Taleb, I., Serhani, M. A., & Dssouli, R. (2018). Big data quality: A survey. In *2018 IEEE International Congress on Big Data (BigData Congress)*, *166-173*. IEEE. https://doi.org/10.1109/BigDataCongress.2018.00029

Teetor N., & Teetor P. (2018) https://cran.r-project.org/web/packages/zeallot/index.html

Thomas, J. J., & Pillai, N. (2018). A deep learning framework on generation of image descriptions with bidirectional recurrent neural networks. *In International Conference on Intelligent Computing & Optimization* (pp. 219-230). Springer, Cham. https://doi.org/10.1007/978-3-030-00979-3_22

Tongke, F. (2021). Research on edge detection of agricultural pest and disease leaf image based on LVQ neural network. *Recent advances in computer science and communications*, *14*(6), 1903-1911. https://doi.org/10.2174/2666255813666191218112436

Uddin, M. F., & Gupta, N. (2014). Seven V's of Big Data understanding Big Data to extract value. In: *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*, Bridgeport, Connecticut, USA, pp. 1-5. https://doi.org/10.1109/ASEEZone1.2014.6820689

Vanhoucke, V., Senior, A., & Mao, M. Z. (2011). Improving the speed of neural networks on CPUs. *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*

Vijendra, S. (2016). Big data characteristics, challenges, architectures, analytics and applications: A review. *International Journal of ocial Computing and Cyber-Psysical Systems, 1*(4), 356-386. ISSN: 2040-0721

Viswambaran, R. A., Chen, G., Xue, B., & Nekooei, M. (2020). Evolving Deep Recurrent Neural Networks Using A New Variable-Length Genetic Algorithm. *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1-8, https://doi.org/10.1109/CEC48606.2020.9185851

Vuckovic, A., Popovic, D., & Radivojevic, V. (2002). Artificial neural network for detecting drowsiness from EEG recordings. In *6th Seminar on Neural Network Applications in Electrical Engineering* (pp. 155-158). IEEE. https://doi.org/10.1109/NEUREL.2002.1057990

Wallisch, P., Lusignan, M. E., Benayoun, M. D., Baker, T. I., Dickey, A. S., & Hatsopoulos, N. G. (2014). MATLAB for neuroscientists: an introduction to scientific computing in MATLAB. *Academic Press*. ISBN: 978-0-12-383836-0

Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2020). A Comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science*, 1-26. https://doi.org/10.1007/s40745-020-00253-5

Wei, S., Chen, Y., Zhou, Z., & Long, G. (2021). A Quantum Convolutional Neural Network on NISQ Devices. *arXiv preprint arXiv:2104.06918*.

Weiss, K., Khoshgoftaar, T.M. & Wang, D. (2016). A survey of transfer learning. *J Big Data* **3,** 9. https://doi.org/10.1186/s40537-016-0043-6

Wickham H. (2016). ggplot2: Elegant Graphics for Data Analysis. *Springer-Verlag New York*. ISBN: 978-0-387-98141-3

Wu, Y., & Sun, X. (2021). Optimization and Simulation of Enterprise Management Resource Scheduling Based on the Radial Basis Function (RBF) Neural Network. *Computational Intelligence and Neuroscience*, *2021*, ID 6025492, 10 pages, 2021. https://doi.org/10.1155/2021/6025492

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv [cs.LG]*. http://arxiv.org/abs/1708.07747

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Yang, X. S. (2019). Introduction to algorithms for data mining and machine learning. *Academic press*. ISBN: 978-0-12-817216-2

Yoo, H., & Chung, K. (2020). Deep learning-based evolutionary recommendation model for heterogeneous big data integration. *KSII Transactions on Internet and Information Systems (TIIS)*, *14*(9), 3730-3744. https://doi.org/10.3837/tiis.2020.09.009

Zaki, M. J., & Meira, Jr., W. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2nd Ed.), Cambridge University Press. ISBN: 978-1108473989.

Zhang, M., Li, H., Pan, S., Lyu, J., Ling, S., & Su, S. (2021). Convolutional neural networks based lung nodule classification: a surrogate-assisted evolutionary algorithm for hyperparameter optimization. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2021.3060833

Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion*, *42*, 146-157. https://doi.org/10.1016/j.inffus.2017.10.006

Zhang XD. (2020). Machine Learning. In: A Matrix Algebra Approach to Artificial Intelligence. *Springer, Singapore*. https://doi.org/10.1007/978-981-15-2770-8_6

Zheng, Y., & Ye, X., & Wu, T. (2021). Using an Optimized Learning Vector Quantization- (LVQ-) Based Neural Network in Accounting Fraud Recognition. *Computational Intelligence and Neuroscience* , ID 4113237, 10 pages. https://doi.org/10.1155/2021/4113237