ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Εισαγωγή στην Επιστήμη Δεδομένων και στην Εκπαίδευση μοντέλου Μηχανικής Όρασης"

Αναστασόπουλος Ζώης, 2773

Γεωργιόπουλος Θεόδωρος, 2792

ΠΑΤΡΑ 2023

Περίληψη

Στην παρούσα πτυχιακή εργασία θα γίνει αναφορά σε γενικά καλές πρακτικές δημιουργίας ενός βασικού μοντέλου (Base Model) μηχανικής μάθησης (Machine Learning). Ο τομέας που ασχολείται με την μηχανική μάθηση είναι αυτός της Επιστήμης Δεδομένων (Data Science) και η παρούσα πτυχιακή εργασία θα επεκταθεί στην υπό-κατηγορία της Μηχανικής Μάθησης: Τα νευρωνικά Δίκτυα (Neural Networks). Όπως θα δούμε παρακάτω, η εκπαίδευση των νευρωνικών δικτύων ανήκει στον τομέα της Μηχανικής Όρασης (Computer Vision).

Οι συγκεκριμένοι τομείς επιλέχθηκαν διότι υπάρχουν κάποιες προ εργασίες στην προετοιμασία των δεδομένων που είναι αναγκαίες και ευρέως επιλέξιμες στον επιστημονικό κλάδο που ασχολείται με την Μηχανική Όραση, έτσι ώστε να επιτευχθεί η δημιουργία ενός μοντέλου με σχετικά καλές επιδόσεις. Προφανώς θα εξηγήσουμε, τι εννοούμε καλές επιδόσεις, τι εννοούμε προ-εργασία δεδομένων και στο τέλος θα εκπαιδεύσουμε ένα νευρωνικό συνελικτικό μοντέλο μηχανικής όρασης με γνώμονα τις μετρικές αξιολόγησης των μοντέλων που υπάρχουν.

Είναι σημαντικό να αναφέρουμε, πως μία κατανόηση της θεωρίας στο τομέα της επιστήμης δεδομένων είναι αναγκαία, έστω εισαγωγικά, επομένως η παρούσα πτυχιακή εργασία θα καλύψει όλο το θεωρητικό υπόβαθρο αναγκαίο για την κατανόηση της εκπαίδευσης του νευρωνικού συνελικτικού μοντέλου που θα εισάγουμε στο τέλος.

Η γλώσσα προγραμματισμού που θα γραφτεί ο κώδικας θα είναι η Python, με την βιβλιοθήκη TensorFlow της Google. Ο λόγος επιλογής αυτών είναι το εύκολο συντακτικό της Python και η καλή υποστήριξη της TensorFlow στο Google Colab, που θα είναι η πλατφόρμα εφαρμογής του κώδικα και εκπαίδευσης των νευρωνικών δικτύων. Στο τέλος της πτυχιακής εργασίας θα δοθεί όλος ο κώδικας που γράφτηκε κατά την διάρκεια της εκπόνησής της.

Summary

In this thesis we will refer to general good practices for creating a Base Model for Machine Learning. The field that deals with machine learning is that of Data Science and this thesis will expand on the sub-category of Machine Learning: Neural Networks. As we will see below, the training of neural networks belongs to the field of Computer Vision.

These domains were chosen because there are some pre-tasks in data preparation that are necessary and widely selectable in the discipline dealing with Machine Vision, in order to achieve a model with relatively good performance. Obviously, we will explain, what we mean by good performance, what we mean by data pre-task and at the end we will train a neural convolutional machine vision model based on the evaluation metrics of existing models.

It is important to mention, that an understanding of theory in the field of data science is necessary, even at an introductory level, therefore this thesis will cover all the theoretical background necessary to understand the training of the neural convolutional model that we will introduce at the end.

The programming language that the code will be written in will be Python, with Google's TensorFlow library. The reason for choosing these is the easy syntax of Python and the good support of TensorFlow in Google Colab, which will be the platform for implementing the code and training the neural networks. At the end of the thesis all the code written during the thesis will be given.

Σύντομη Περιγραφή

Στο **κεφάλαιο 1** Εισαγωγή στην Επιστήμη Δεδομένων, εισάγουμε το θεωρητικό ορισμό της επιστήμης δεδομένων, δηλαδή τι μελετά, σε τι υποκατηγορίες διακρίνεται και τις διαφορετικές μεθόδους που συναντάμε σε μία εξόρυξη δεδομένων, για να προσεγγίσεις ένα πρόβλημα. Προφανώς, για λόγους πληρότητας αναφέρουμε και τον επηρεασμό που έχει η ανάπτυξη της επιστήμης των δεδομένων στην καθημερινή ζωή του ανθρώπου. Στην συνέχεια αναφέρουμε μερικά βασικά χαρακτηριστικά των δεδομένων και σε τι τύπους διακρίνονται, αριθμητικούς και κατηγορικούς, μαζί με τις υπό-κατηγορίες αυτών.

Στο **κεφάλαιο 2** Εισαγωγή στην Βαθιά Μάθηση, αναλύουμε τον ορισμό ενός νευρωνικού δικτύου στην μηχανική μάθηση, και εστιάζουμε κυρίως στον τομέα της επιστήμης δεδομένων, που μελετά τα νευρωνικά δίκτυα, απαραίτητα για την έμπρακτη μάθησή τους σε επόμενο κεφάλαιο. Επομένως, γίνεται αναφορά στους περιορισμούς και στις ικανότητες των διάφορων νευρωνικών δικτύων που μπορούμε να φτιάξουμε, και τέλος ερχόμαστε σε μία πρώτη επαφή με τα σύνολα δεδομένων, τους διαχωρισμούς που κάνει η επιστήμη δεδομένων σε αυτά, στις αρχές μάθησης που είναι αναγκαίες για να εκπαιδεύσεις ένα μοντέλο καθώς και τα βήματα που πρέπει να ακολουθήσει κανείς.

Στο **κεφάλαιο 3** Βασικές Αρχές, ξεκινάμε με κοινές πρακτικές στην μάθηση μοντέλων, όπως διαχωρισμό των συνόλων σε σύνολα εκπαίδευσης, επικύρωσης, αξιολόγησης και την σημασία τους. Δίνουμε μαθηματικά ορισμένες μετρικές έτσι ώστε να υπάρχει η κατανόηση της αξιολόγησης των επιδόσεων ενός νευρωνικού μοντέλου. Καθώς και αναφέρουμε, βασικές τακτικές επίλυσης προβλημάτων, όπως το φαινόμενο της υπερπροσαρμογής, με την διαμόρφωση των δεδομένων πριν εκπαιδευτεί το μοντέλο. Και τέλος, εξηγούμε την αρχιτεκτονική Tiny-VGG την οποία θα μιμηθούμε σε παρακάτω εκπαίδευση συνελικτικού νευρωνικού μοντέλου.

Στο **κεφάλαιο 4** Εκπαίδευση και Αποτελέσματα, προσπαθούμε να πράξουμε όλη την θεωρία και κατανόηση του τομέα της μηχανικής όρασης που μάθαμε σε προηγούμενα κεφάλαια. Αυτό γίνεται με την εισαγωγή του συνόλου δεδομένων από το διαδίκτυο, ετοιμασία και επεξεργασία του συνόλου δεδομένων με παράλληλη μάθηση συνελικτικών μοντέλων, για την παρατήρηση και παραγωγή διαφορετικών επιδόσεων, καθώς εφαρμόζουμε γνωστές και διαφορετικές τεχνικές στην προ εργασία των δεδομένων αυτών. Και τέλος δίνουμε τον κώδικα γραμμένο σε Python με την μορφή διαδικτυακού ιστοτόπου που παραπέμπει σε σημειωματάριο της Google Colab.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία εκπονήθηκε στο **Πανεπιστήμιο Πελοποννήσου** με σκοπό την ολοκλήρωση των προπτυχιακών μας σπουδών στο τμήμα **Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών**, της σχολής **Μηχανικών** κατά τη διάρκεια φοίτησης 2016-2023. Θα θέλαμε να ευχαριστήσουμε ιδιαιτέρως τον επιβλέποντα καθηγητή μας Χριστοδούλου Σωτήριο για την συμβουλευτική καθοδήγηση του, τον χρόνο του και τη άριστη συνεργασία που είχαμε κατά τη διάρκεια της συγγραφής της εργασίας αυτής, καθώς χωρίς αυτόν θα ήταν αδύνατη η ολοκλήρωση της. Χρωστάμε, επίσης, ένα μεγάλο ευχαριστώ σε όλους τους καθηγητές του τμήματος για την συμβολή τους στις γνώσεις και στην επιστημονική μας συγκρότηση κατά τα χρόνια την φοίτησης μας στο τμήμα. Οφείλουμε ακόμα, ένα μεγάλο ευχαριστώ σε όσους συναδέλφους συνέβαλαν στην ολοκλήρωση της εργασίας αυτής με τις γνώσεις και την βοήθεια του. Τέλος, θα θέλαμε να ευχαριστήσουμε τις οικογένειες και τους φίλους μας για την συμπαράσταση και την υποστήριξη τους, τόσο ηθική όσο και πρακτική, σε όλο το διάστημα των σπουδών μας.

Περιεχόμενα

1. Εισαγωγή στην Επιστήμη Δεδομένων

Τα τελευταία χρόνια η ραγδαία ανάπτυξη των επεξεργαστών (CPU) ενός ηλεκτρονικού υπολογιστή, επέτρεψε στην έμπρακτη εφαρμογή των Μεθόδων της Μηχανικής Μάθησης, που είναι γνωστές σε θεωρητικό επίπεδο από τα τέλη το 20ού αιώνα. Πλέον ένας υπολογιστής ενός μέσου νοικοκυριού μπορεί να διαχειριστεί τις απαραίτητες απαιτήσεις σε ταχύτητα που μπορεί να ζητήσει μία τέτοια έμπρακτη εφαρμογή της «θεωρίας», κάτι αδύνατον στα τέλη του 20ού αιώνα.

Αυτή η «θεωρία» έχει ονομαστεί ως Επιστήμη Δεδομένων (Data Science) και την τελευταία δεκαετία (2013 και μετά) έχει εξελιχθεί σε τέτοιο βαθμό που συναντάμε μέχρι και αυτοκίνητα που μπορούν να οδηγηθούν μόνα τους, χωρίς την ανάγκη κάποιου οδηγού (Tesla Inc.). Παρόλα αυτά όμως, δεν σημαίνει ότι είναι μια εύκολη διαδικασία η δημιουργία ενός τέτοιου μοντέλου μάθησης (ή στην προκειμένη περίπτωση: νευρωνικό δίκτυο) , αλλά απαιτεί ιδιαίτερη γνώση του αντικειμένου, με ικανότητα στην επεξεργασία και προετοιμασία των δεδομένων καθώς και πολλαπλές προσπάθειες βελτίωσης και παραμετροποίησης των παραμέτρων ενός τέτοιου μοντέλου, έτσι ώστε να ικανοποιεί ικανοποιητικά τις ανάγκες δημιουργίας του.

Όπως θα δούμε παρακάτω, το τελικό αποτέλεσμα ενός νευρωνικού μοντέλου (Neural Network), το οποίο είναι έτοιμο για χρήση και εκμετάλλευση, περνάει από μια «σειρά διαδικασιών» και επαναλήψεων. Αυτή η «διαδικασία» εμπεριέχει σαν βήματα την επεξεργασία και την προετοιμασία των δεδομένων που το μοντέλο θα εκπαιδευτεί. Πρώτα όμως πρέπει να γίνει μία εισαγωγή των εργαλείων που θα χρησιμοποιήσουμε, την αναφορά των τύπων των δεδομένων που μπορεί να συναντήσουμε και θα αναφέρουμε τα πιο γνωστά βήματα ή σειρά βημάτων που κάποιος πρέπει να διαβεί, έτσι ώστε να πετύχει μία σωστή εκπαίδευση ενός νευρωνικού δικτύου σύμφωνα με ειδικούς. [1]

1.1 Επιστήμη Δεδομένων και Καθημερινή Ζωή

Η καθημερινή μας ζωή μετά την ανακάλυψη του Internet και των social media έχει αλλάξει καθοριστικά. Άνθρωποι πλέον μπορούν να επικοινωνούν σε πραγματικό χρόνο από διαφορετικές άκρες του πλανήτη (Skype). Υπάρχει η δυνατότητα της αγοράς προϊόντων από το διαδίκτυο και η αποστολή αυτών στην πόρτα του καθενός, αυθημερόν (Amazon Inc.). Όλα αυτά είναι επιτεύξιμα χάρη στο διαδίκτυο και την μεταφορά δεδομένων. Όμως η ανάπτυξη της τεχνολογίας δεν σταματάει εκεί, χάρη στα δεδομένα έχουμε την δυνατότητα να μαθαίνουμε σε έναν υπολογιστή πως να μαθαίνει.

Η διαδικασία ένας υπολογιστής να μαθαίνει μέσα από δεδομένα και να ανακαλύπτει τυχαία μοτίβα αποκαλείται Μηχανική Μάθηση και η Επιστήμη Δεδομένων, που τόσο συχνά έχουμε αναφέρει, είναι η επιστήμη που μελετάει ακριβώς αυτό σε βάθος. Η επιστήμη Δεδομένων είναι ο κλάδος της τεχνολογίας των υπολογιστών (Computer Science) που συνδυάζει μαθηματικά, στατιστικά, ειδικό προγραμματισμό, προχωρημένη ανάλυση, Τεχνητή Νοημοσύνη και Μηχανική Μάθηση με συγκεκριμένη και ειδική τεχνογνωσία για την ανακάλυψη κρυφών μοτίβων που κρύβονται στα δεδομένα ενός οργανισμού. Αυτές οι γνώσεις μπορούν να χρησιμοποιηθούν για να καθοδηγήσουν σε μια σωστή λήψη αποφάσεων και σε έναν στρατηγικό σχεδιασμό [1].

Πολλοί συσχετίζουν τον όρο «Δεδομένα» με ζητήματα απορρήτου (privacy issues), διαρροές δεδομένων (data leaks) η ακόμα και τεχνικές εκμετάλλευσης πληροφοριών με σκοπό την αισχροκέρδεια. Θέτοντας, έτσι αν η ίδια η Επιστήμη Δεδομένων είναι κάτι που θα έπρεπε η ανθρωπότητα να ανησυχεί. Όπως θα δούμε αυτή δεν είναι η περίπτωση.

Στην πραγματικότητα η Επιστήμη Δεδομένων έχει κάνει περισσότερο «καλό» και έχει εγκατασταθεί περισσότερο στην καθημερινή μας ζωής με θετικό τρόπο, βελτιώνοντας την διαβίωσή μας, παρά το αντίθετο. Στην συνέχεια θα αναφέρουμε μερικές περιπτώσεις που η Επιστήμη Δεδομένων έχει επηρεάσει θετικά τις ζωές μας.

- Υγεία:

  Ο τομέας της υγείας ίσως είναι από τους πιο ραγδαία αναπτυσσόμενους κλάδους της Επιστημονικής Κοινότητας, έχοντας καταφέρει την δημιουργία εμβολίων πρόληψης σοβαρών παθήσεων και ιών μέχρι και την διόρθωση γενετικών προβλημάτων μέσω χειρουργείων. Έχοντας την πρώτη θέση στο κατά πόσο επηρεάζει την ανθρώπινη ζωή ο τομέας της υγείας, παρατηρούμε πως η Επιστήμη Δεδομένων, και τα Δεδομένα πιο συγκεκριμένα, έχουν συμβάλει σημαντικά μέσω κλινικών δοκιμών και στατιστικών

[1] https://www.ibm.com/topics/data-science

μεθόδων την παραγωγή συμπερασμάτων, όπως αυτής της μεθόδους της χορήγησης εικονικού φαρμάκου, με αποτέλεσμα την καλυτέρευση και πρόληψη της ανθρώπινης ζωής από κινδύνους. [2]

- Καιρός και Καταστροφές:

  Υπάρχουν δορυφόροι και αισθητήρες τοποθετημένοι σε όλες τις γωνιές του πλανήτη, με μοναδικό σκοπό την πρόβλεψη καιρικών συνθηκών ή και καταστροφών (σεισμοί, πλημμύρες). Από αυτά τα αισθητήρια όργανα αντλούνται άπειρου μεγέθους δεδομένα, κάτι το οποίο ένας επιστήμονας που ασχολείται με αυτά να είναι αδύνατον να τα επεξεργαστεί. Αλλά χάρη στην ανάπτυξη της Επιστήμης Δεδομένων, η επεξεργασία και η ανάλυση των δεδομένων που αντλούνται γίνεται με εύκολο και κατανοητό τρόπο. Θέτοντας έτσι την σοβαρότητα της ύπαρξης της επιστήμης αυτής, εφόσον η ασφάλεια ολόκληρων πληθυσμών εξαρτάται εξ ολοκλήρου από τις μεθόδους επεξεργασίας των ογκώδη δεδομένων. [2]

- Διασκέδαση:

  Χάρη στην επιστήμη που μελετά τα δεδομένα, δηλαδή η Επιστήμη των Δεδομένων, μπορούμε και παρέχουμε μια εμπειρία χρήσης τόσο πρωτότυπη όσο και καθοριστική για την ανθρώπινη καθημερινότητα. Η επιστήμη δεδομένων έχει και πάλι επηρεάσει τον τρόπο που αλληλεπιδράμε καθημερινώς ή ψυχαγωγούμαστε. Για παράδειγμα τον τρόπο που διαλέγουμε και ακούμε μουσική, προτείνοντας εξιδεικευμένα σε κάθε άτομο, νέα κομμάτια και προτιμήσεις (Spotify Inc.). Ακόμα, υπάρχει ανάπτυξη αλγορίθμων προτίμησης προϊόντων για κάθε άτομο ξεχωριστά όπως ταινίες που δεν ξέρουμε εξ αρχής ότι θα μας προσεγγίσουν το ενδιαφέρον, σε αντίθεση με τον αλγόριθμο που το είχε προβλέψει με μεγάλη πιθανότητα επιτυχίας. Παρόλα αυτά η επιστήμη αυτή δεν σταματάει μόνο εκεί αλλά συνεχώς νέες ιδέες και αντιλήψεις έρχονται συνεχώς στο παρασκήνιο για νέα προϊόντα ψυχαγωγίας. [2]

- Γενικότερα:

  Αξίζει να σημειωθεί πως η Επιστήμη Δεδομένων έχει επηρεάσει πολύ και το χρηματοοικονομικό κομμάτι των τραπεζών, αγορών, χρηματιστηρίου. Αυτό συμβαίνει διότι, αναπτύσσονται συνεχώς νέες τεχνολογικές εφαρμογές που αναλύουν δεδομένα με τέτοιο τρόπο ώστε μία απόφαση να βασίζεται πάνω σε κρίση του υπολογιστή. Για παράδειγμα, μία τράπεζα μπορεί να υπολογίσει πλέον με μεγάλη πιθανότητα εάν κάποιος ενδιαφερόμενος θα μπορέσει να αποπληρώσει το δάνειο ή όχι. Οι δυνατότητες, όμως δεν σταματάνε εκεί, καθημερινά σχεδόν, παρατηρείται μία ραγδαία αύξηση των

επαγγελμάτων που σκοπό έχουν την βελτίωση και συντήρηση τέτοιων εφαρμογών. Καταλαβαίνουμε λοιπόν πως η Επιστήμη των Δεδομένων ήρθε για να μείνει και πλέον η ανθρωπότητα βαδίζει σε νέα εδάφη. [2]

1.2 Επιστήμη Δεδομένων και Παρακλάδια της

Η Επιστήμη των δεδομένων δεν είναι ένας κλάδος που μπορεί να οριστεί εύκολα και απλά. Πρόκειται για έναν τομέα που περιλαμβάνει πληθώρα «διαδικασιών», υποκατηγοριών, και στόχους. Με λίγα λόγια μιλάμε για ένα ευρύ φάσμα δυνατοτήτων και τεχνογνωσίας. Παρόλα αυτά θα γίνει μία προσπάθεια εξήγησης κάποιων υποκατηγοριών που συχνότερα αναφέρονται σαν υποκατηγορίες της Επιστήμης Δεδομένων. Η επιστήμη δεδομένων χωρίζεται σε:

a) Σύνολα Δεδομένων που την αποτελούν
b) Υπολογιστικές και μαθηματικές μεθόδους της
c) Τεχνικές συλλογής, επεξεργασίας, αποθήκευσης και διανομής δεδομένων. [2]

1.2.1 Σύνολα Δεδομένων

Είναι λογικό να υποθέσουμε πως η επιστήμη των δεδομένων θα ασχολείται με δεδομένα. Εδώ θα γίνει αναφορά των συνόλων που η επιστήμη των δεδομένων διακρίνει και χωρίζει τα δεδομένα μεταξύ τους. Ο σκοπός είναι η καλύτερη αναγνώριση στο τύπο προβλημάτων που βρισκόμαστε, κάτι το οποίο η μορφή των δεδομένων του προβλήματος, μαρτυρά.

- Μεγάλα δεδομένα (**Big Data**):

  Προηγουμένως (κεφάλαιο 1.1) είχε αναφερθεί ένα παράδειγμα μεγάλων σε όγκο δεδομένων. Αυτό ήταν το παράδειγμα των αισθητήριων οργάνων που υπάρχουν σε διάφορες περιοχές του πλανήτη για να προλαμβάνουν μια φυσική καταστροφή. Τα μεγάλα δεδομένα είναι ένα σύνολο δεδομένων που συνήθως παράγονται σε μεγάλο όγκο σε πολύ μικρή χρονική διάρκεια. Προφανώς ο όγκος αυτός των δεδομένων δεν μπορεί να τα «διαβάσει» ένας άνθρωπος, για αυτό και η επιστήμη των δεδομένων έχει αναπτύξει ειδικά εργαλεία για δεδομένα τέτοιων περιπτώσεων. [2]
- Διάσπαρτα Δεδομένα (**Unstructured Data**):

  Παράλληλα με την δημιουργία των Μεγάλων Δεδομένων, δημιουργούνται και να Διάσπαρτα Δεδομένα. Διάσπαρτα Δεδομένα είναι εκείνα που συλλέγονται με διαφορετικούς τρόπους από διαφορετικές πηγές, και καταλήγουν σε διάφορους

τύπους δεδομένων. Για παράδειγμα η άντληση πληροφοριών από τα κοινωνικά δίκτυα, μπορεί να περιλαμβάνει βίντεο, αρχεία ήχου, αρχεία κειμένου κ.τ.λ. [2]

- Αμφίβολα και ελλιπή δεδομένα (**Uncertain and Missing Data**):

  Η συλλογή δεδομένων, πέρα από αναγκαία για να μπορέσει κάποιος να κάνει κάτι με αυτά, είναι και δύσκολη διαδικασία. Συνήθως τα δεδομένα που συλλέγονται, είτε από λάθος χειρισμού, είτε από έλλειψη πληροφορίας έρχονται με κενά και διαφοροποιήσεις. Αυτό δημιουργεί ένα πρόβλημα το οποίο είναι πως θα ετοιμάσουμε τα δεδομένα με τέτοιο τρόπο ώστε να μπορούμε να τα αξιοποιήσουμε. Παρόλα αυτά, τα δεδομένα με τέτοιες ιδιαιτερότητες ονομάζονται ελλιπή. Αμφίβολα ονομάζονται εκείνα τα οποία ο οργανισμός ή ο τρόπος συλλογής τους συνιστά την αμφιβολία και αξιοπιστία τους, είναι δηλαδή δεδομένα που δεν είναι αξιόπιστα. [2]

1.2.2 Υπολογιστικές Μεθόδους

Η επιστήμη δεδομένων πέρα από τα κατηγορίες δεδομένων που μελετάει, έχει αφιερώσει ένα μεγάλο κομμάτι της στην δημιουργία μαθηματικών και στατιστικών μεθόδων για την πλήρη αξιοποίηση των μοτίβων που μπορεί να υπάρχουν στα δεδομένα. Αυτοί οι μέθοδοι μπορεί να είναι, αλγόριθμοι που επιλύουν ένα πρόβλημα, Τεχνητή Νοημοσύνη, Μοντέλα Μηχανικής Μάθησης με σκοπό την δημιουργία ενός εκπαιδευμένου μοντέλου.

Πιο συγκεκριμένα, ένα τέτοιο **μοντέλο**, μαθαίνει τα δεδομένα με τέτοιο τρόπο έτσι ώστε να μην χρειαστεί η ανθρώπινη παρέμβαση. Για παράδειγμα, δεν μπορούμε να προγραμματίσουμε με τον κλασσικό τρόπο, έναν αλγόριθμο να αναγνωρίζει τα πρόσωπα σε μια εικόνα, επομένως η Μηχανική Μάθηση και γενικότερα οι Υπολογιστικές Μεθόδους, της Επιστήμης Δεδομένων, εκμεταλλεύονται με τέτοιο τρόπο τα μοτίβα που ανακαλύπτει στα σύνολα δεδομένων και σαν αποτέλεσμα δίνει ένα μοντέλο το οποίο πλέον μπορεί να αναγνωρίσει πρόσωπα σε μία εικόνα. Δηλαδή, έχει πλέον «εκπαιδευτεί».

Είναι σημαντικό να αναφέρουμε πως μέσα στις υπολογιστικές μεθόδους επίλυσης ενός προβλήματος, αντιμετωπίζουμε και το πρόβλημα παραγωγικότητας ενός «αλγορίθμου» ή αλλιώς μοντέλου. Με λίγα λόγια χρειάζεται να ορίσουμε μαθηματικές έννοιες που αξιολογούν πόσο καλά μία μέθοδος επιλύει ένα πρόβλημα, έτσι ώστε να διαχωρίσουμε τις καλύτερες μεταξύ όλων.

Εδώ εμφανίζονται μερικές μετρικές αξιολόγησης που βασίζονται σε μαθηματικούς τύπους, όπως το κέρδος πληροφορίας (**Information Gain**) και η εντροπία (**Entropy**).

Για καλύτερη κατανόηση, θα αναφερθούμε σε παρακάτω κεφάλαια, σε μεθόδους αξιολόγησης. Παρόλα αυτά, ξεκινάμε με την αναφορά των μεγαλύτερων μαθηματικών μεθόδων στην Επιστήμη Δεδομένων.

- Μηχανική Μάθηση (**Machine Learning**):

  Η Μηχανική Μάθηση εφαρμόζει κυρίως στατιστικές και μαθηματικές μεθόδους με σκοπό την κατηγοριοποίηση των δεδομένων σύμφωνα με ιδιαιτερότητες κάθε μίας γραμμής πληροφορίας, δηλαδή κάθε ενός στοιχείου που την απαρτίζουν. Για παράδειγμα, η κατηγοριοποίηση μανιταριών στο είδος τους, σύμφωνα με χαρακτηριστικά όπως το μέγεθος των μανιταριών, το σχήμα κ.τ.λ. (Iris Dataset). Η Μηχανική Μάθηση μαθαίνει πως να ξεχωρίζει μανιτάρια, με εκμάθηση των μοτίβων από ένα σύνολο δεδομένων με πολλές εισαγωγές μανιταριών. Αυτό κάνει με λίγα λόγια η Μηχανική Μάθηση. [2]

- Βαθιά Μάθηση (**Deep Learning**):

  Η μέθοδος Βαθιάς Μάθησης, βρίσκεται σαν υποκατηγορία της Μηχανικής Μάθησης, αλλά δημιουργεί την δική της «ανακάλυψη» μοτίβων με έναν διαφορετικό τρόπο αυτό της Μηχανικής Μάθησης. Η βαθιά Μάθηση χρησιμοποιεί τεχνητούς νευρώνες (Neural Networks) με την τεχνική του ζυγίσματος των βαρών ενός τεχνητού νευρώνα. Οι τεχνητοί νευρώνες αντιγράφουν την μάθηση των βιολογικών νευρώνων που συναντάμε σε ζωντανούς οργανισμούς. Είναι μία μέθοδος, μάθησης η οποία μέσω της εμπειρίας και των επαναλήψεων, βελτιώνει και «μαθαίνει» όπως ένας ανθρώπινος εγκέφαλος. Είναι μία μέθοδος που υπάρχει από τα μέσα του 20^{ου} αιώνα αλλά είναι εφικτή η λειτουργία μόνο τα τελευταία χρόνια, χάρη στην εξέλιξη του υλισμού ενός ηλεκτρονικού υπολογιστή. [2]

- Τεχνητή Νοημοσύνη (**Artificial Intelligence**):

  Η τεχνητή νοημοσύνη αναφέρεται στον κλάδο της επιστήμης δεδομένων παρόλο που ασχολείται με την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς. Αυτό συμβαίνει διότι χρησιμοποιεί δεδομένα για να «μιμηθεί» την ανθρώπινη συμπεριφορά, ακόμα και αν ο κλάδος την Τεχνητής Νοημοσύνης αγγίζει ένα ευρύ φάσμα επιστημονικών τομέων όπως αυτής της ψυχολογίας, φιλοσοφίας, νευρολογίας αλλά και της γλωσσολογίας. Φυσικά, τα μαθηματικά για την τεχνολογική ανάπτυξη της Τεχνητής Νοημοσύνης, είναι και πάλι παρόν. [2]

- Αλγόριθμοι (**Algorithms**):

  Οι αλγόριθμοι κατατάσσονται στον τομέα της πληροφορικής και των μαθηματικών. Ένας αλγόριθμος είναι μια σειρά από «εντολές» και διαδικασίες, με σκοπό την επιτυχή επίλυση ενός συγκεκριμένου τύπου προβλήματος. Τα μαθηματικά στον τομέα των αλγορίθμων, είναι εμφανώς υπαρκτά και συνδυάζονται εύκολα με την επιστήμη των υπολογιστών και κατά συνέπεια με την επιστήμη των δεδομένων. Η επιστήμη των δεδομένων, έχει αλγορίθμους που σχετίζονται συνήθως με την ταξινόμηση αριθμών, αρχείων, διαφοροποίηση κ.τ.λ. [2]

Αυτές είναι μερικές κατηγορίες στις οποίες η επιστήμη δεδομένων συνυπάρχει με τα μαθηματικά έτσι ώστε να αξιοποιεί τα δεδομένα με τέτοιο τρόπο ώστε να λύνει προβλήματα που δεν μπορούσαν να λυθούν στο παρελθόν. Θα συνεχιστεί η ανάλυσή μας στην επόμενη κατηγορία που διακρίνουμε την επιστήμη δεδομένων, στις τεχνικές συλλογής, επεξεργασίας, αποθήκευσης και διανομής δεδομένων.

1.2.3 Τεχνικές συλλογής, επεξεργασίας, αποθήκευσης και διανομής δεδομένων

Η επιστήμη δεδομένων πέρα από την αξιοποίηση των δεδομένων με μαθηματικές μεθόδους, όπως είδαμε, μελετάει και κάτι ακόμα πολύ πιο βασικό. Μελετάει τις μεθόδους συλλογής και επεξεργασίας των δεδομένων ώστε να υπάρξει η οποιαδήποτε εκμετάλλευσή τους. Η οποιαδήποτε αξιοποίηση των δεδομένων δεν υφίσταται αν τα δεδομένα δεν τα μετατρέψουμε σε συγκεκριμένες «μορφές».

Οι τεχνικές συλλογής και επεξεργασίας των δεδομένων, με οποιοδήποτε τρόπο, ονομάζεται Εξόρυξη Δεδομένων (**Data Mining**). Υπάρχουν δεδομένα που συλλέγονται από εταιρικές διεργασίες (**Business Processes**), δεδομένα από γραμμές παραγωγής, δεδομένα από αισθητήρες, δεδομένα από κοινωνικά δίκτυα που παράγονται συνεχώς, δεδομένα από συσκευές παρακολούθησης, κάμερες και ούτω καθεξής. Με λίγα λόγια τα δεδομένα συνεχώς δημιουργούνται και προέρχονται από διαφορετικούς πόρους και διαφορετικές μορφές, είναι συνετό λοιπόν να αναφέρουμε την προετοιμασία των δεδομένων ως ένα ξεχωριστό κλάδο της Επιστήμης Δεδομένων, και αυτός ο κλάδος δεν είναι κανείς άλλος από την Εξόρυξη Δεδομένων (**Data Mining**). [2]

Η συλλογή δεδομένων είναι από μόνη της μία διαδικασία ιδιαίτερα δύσκολη αφού δεν περιλαμβάνει μια καθολική διαδικασία συλλογής. Παρόλα αυτά πολλά δεδομένα, σύνολα δεδομένων, προσφέρονται δωρεάν, έτοιμα για μετέπειτα επεξεργασία και αξιοποίηση. Πολλές φορές κυβερνητικές ιστοσελίδες αναρτούν τέτοια δεδομένα, και πλατφόρμες όπως το Kaggle, είναι πηγές δεδομένων που μπορούν να θεωρηθούν αξιόπιστες.

Μόλις αποκτήσουμε τα δεδομένα, είναι απαραίτητο να τα αποθηκεύσουμε. Αυτό το πρόβλημα έχει παρουσιαστεί πολλά χρόνια πριν, και κατά συνέπεια έχουν αναπτυχθεί γλώσσες προγραμματισμού και βάσεις δεδομένων, που κάνουν εύκολη την διαχείριση και αποθήκευση των δεδομένων. Γλώσσες προγραμματισμού όπως η **SQL** και τα παρακλάδια της, έχουν κυρίαρχο ρόλο στην αποθήκευση και διαχείριση αυτών. Παρόλα αυτά, η παρόν εργασία δεν θα ασχοληθεί με την αποθήκευση των δεδομένων, επομένως οποιοδήποτε αναγνώστης επιθυμεί μπορεί να μεταφερθεί στο διαδίκτυο και σε πηγές όπως τον Walter Shields. Στο επόμενο κεφάλαιο θα αναφερθούμε στα μοντέλα εξόρυξης δεδομένων, και ο κύκλος που διανύουν μέχρι την τελική μορφή τους, που τα καθιστά ικανά για αξιοποίηση από υπολογιστικές μεθόδους. [2]

1.3 Μοντέλο Εξόρυξης Δεδομένων

Ερχόμενοι λοιπόν με το πρόβλημα της συλλογής και προετοιμασίας των δεδομένων, πολλοί ειδήμονες προσπάθησαν να δημιουργήσουν ένα καθολικό μοντέλο από «διαδικασίες» έτσι ώστε να επιτευχθεί μία επιτυχημένη εξόρυξη δεδομένων (Data Mining). Το Εθνικό Αμερικάνικο Ίδρυμα δημιούργησε το **Data Life Cycle** (Κύκλο ζωής των Δεδομένων) το οποίο αποτελείται από 8 βήματα, ενώ ο Gregory Piatetsky-Shapiro δημιούργησε το έτος 1989 το μοντέλο **Knowledge Discovery Databases** (Βάσεις δεδομένων ανακάλυψης γνώσης) το οποίο αποτελείται από 9 βήματα. Υπάρχουν πολλά μοντέλα που επιτυγχάνουν μία επιτυχημένη εξόρυξη δεδομένων, παρόλα αυτά όλα ακολουθούν τα παρακάτω 6 βήματα που θα αναφερθούν. [3]

ΒΗΜΑ 1: *Κατανόηση Επιχειρηματικότητας – Business Understanding*

Εδώ είναι η αρχική φάση της εξόρυξης δεδομένων. Η κατανόηση Επιχειρηματικότητας αποσκοπεί στην κατανόηση των απαιτήσεων και των δυσκολιών που θα εμφανιστούν με σκοπό την εκπλήρωση του στόχου, όποιος και αν είναι αυτός ο στόχος. Περιλαμβάνει κριτήρια επιτυχίας, επιχειρηματικά δεδομένα, τεχνολογίες ή ορολογίες που χρειάζονται περισσότερη κατανόηση. Επομένως το Βήμα 1, είναι να κατανοήσουμε τι και πως θα λύσουμε ένα συγκεκριμένο πρόβλημα. [2]

ΒΗΜΑ 2: *Κατανόηση Δεδομένων – Data Understanding*

Μετά την κατανόηση της επιχειρηματικότητας, το επόμενο βήμα είναι η συλλογή δεδομένων και ο έλεγχος ποιότητας τους. Εδώ ξεκινάμε την πρώτη μορφή εξερεύνησης των δεδομένων, όπως σε τι μορφή βρίσκονται, σε τι μορφή θέλουμε να τα μετατρέψουμε, κατανοούμε αν μπορούμε να τα συνδυάσουμε με δεδομένα άλλων πηγών ακόμα και αν χρειάζεται να συλλέξουμε καινούργια, περισσότερα ή διαφορετικού είδους δεδομένα. Καλή πρακτική θεωρείται και η απεικόνιση σε δισδιάστατη ή τρισδιάστατη μορφή των δεδομένων, έτσι ώστε να υποψιαζόμαστε τυχόν αλληλεξαρτήσεις ή μοτίβα που μπορεί να έχουν (π.χ. γραμμική εξάρτηση). Σημαντικό επίσης είναι να αναφέρουμε πως τα εργαλεία που μας βοηθάνε στο δεύτερο βήμα μιας επιτυχής εξόρυξης δεδομένων, είναι οι γλώσσες προγραμματισμού ή προγράμματα επιχειρηματικής νοημοσύνης (**Business Intelligence**) όπως η **Python**, **SQL** και **EXCEL**. [2]

ΒΗΜΑ 3: *Προετοιμασία Δεδομένων – Data Preparation (Preprocessing)*

Οι διεργασίες που γίνονται σε αυτή την φάση της εξόρυξης δεδομένων, περιλαμβάνουν κυρίως την επιλογή και προετοιμασία του τελικού συνόλου δεδομένων που έχουμε ανακτήσει. Αυτή η επιλογή του τελικού συνόλου, περιλαμβάνει την διαμόρφωση των χαρακτηριστικών που κατέχουμε (πληροφορίες), συγχώνευση πινάκων, διαμόρφωση των διαστάσεων των δεδομένων, καθώς και καθαρισμό του συνόλου δεδομένων (**Data Cleaning**), δηλαδή την διόρθωση τυχόν λαθών και ελλείψεων που υπάρχουν στο σύνολο δεδομένων. Παρατηρείται επίσης και η διαμόρφωση των χαρακτηριστικών με την δημιουργία καινούργιων, μια διαδικασία που ονομάζεται (**Feature Engineering**). Ο σκοπός της δημιουργίας ενός τελικού συνόλου δεδομένων, είναι η διαμόρφωσή του έτσι ώστε οι μαθηματικές μεθόδους που θα εφαρμοστούν πάνω σε αυτό, να είναι ικανές να «καταλαβαίνουν» με σκοπό να εκπαιδευτούν πάνω σε αυτό το σύνολο δεδομένων. [2]

ΒΗΜΑ 4: *Διαμόρφωση Μοντέλων – Modeling*

Στην Διαμόρφωση Μοντέλων πράττουμε μαθηματικές μεθόδους πάνω στα σύνολα δεδομένων που έχουμε προηγουμένως ετοιμάσει (Βήμα 3). Δηλαδή, εδώ γίνεται η δημιουργία και εκπαίδευση των διάφορων μοντέλων μηχανικής μάθησης. Σκοπός του συγκεκριμένου βήματος είναι να διεξαχθεί η ανάπτυξη ενός σχεδιασμού δοκιμής, εφαρμογής και αξιολόγησης μοντέλων έτσι ώστε ως τελικό προϊόν να δημιουργηθεί ένα μοντέλο μηχανικής μάθησης το οποίο ικανοποιεί τις προδιαγραφές που μας απασχολούν.

Είναι σημαντικό να αναφερθεί πως η μοντελοποίηση έχει στενή σχέση με το προηγούμενο βήμα της Εξόρυξης Δεδομένων (Βήμα 3) το οποίο προετοιμάζει δεδομένα με τέτοιο τρόπο ώστε η επίτευξη ενός μοντέλου μηχανικής μάθησης, να καθίσταται δυνατή. Παρόλα αυτά, όμως, υπάρχει η ανάγκη της δημιουργίας ενός μοντέλου με τις βέλτιστες επιδόσεις, αυτό φέρει σαν αποτέλεσμα την επινόηση ενός «κύκλου εκπαίδευσης» των μοντέλων, το οποίο δημιουργεί διαφορετικού τύπου επιδόσεων κάθε μοντέλου που δημιουργείται. Αυτό συμβαίνει είτε με την διαφοροποίηση των δεδομένων που εκπαιδεύεται είτε με τις ρύθμιση των υπερ-παραμέτρων του (**Hyper-Parameters**) είτε και με την δοκιμή διαφορετικών μαθηματικών μοντέλων. Όλα αποσκοπούν στην βελτιστοποίηση. [2]

ΒΗΜΑ 5: *Αξιολόγηση Μοντέλων – Evaluation*

Στην Αξιολόγηση Μοντέλων εστιάζουμε στην απόδοση των μοντέλων που εφαρμόστηκαν στο προηγούμενο βήμα (Βήμα 4) και αποφασίζουμε την ποιότητα των

αποτελεσμάτων σε σχέση με τα δεδομένα, τους εφαρμοσμένους αλγορίθμους και εάν το μοντέλο είναι σε θέση να επιτύχει τους επιχειρηματικούς στόχους που θέσαμε. Σε μία πολύ πιθανή εκδοχή, τα αποτελέσματα των μοντέλων να μην είναι να αποδεκτά, εφαρμόζουμε τον κύκλο εκπαίδευσης από την αρχή, και πολλές φορές ξεκινάμε πάλι από το Βήμα 3 ή το Βήμα 4 με διαφορετική προσέγγιση σε παραμέτρους. Στην ουσία ο κύκλος εκπαίδευσης προέρχεται μέσα από πειραματικές αλλαγές στις παραμέτρους των μοντέλων ή στην επεξεργασία των δεδομένων εκπαίδευσης. [2]

ΒΗΜΑ 6: *Απόδοση – Deployment*

Στο Βήμα 6 που είναι και το τελικό στάδιο της Εξόρυξης Δεδομένων, αναφέρουμε όλες τις γνώσεις που αποκτήθηκαν με σκοπό να παρουσιαστεί στο ενδιαφερόμενο κοινό, αν υπάρχει. Αυτό καταφέρνετε με την σύνταξη μιας τελευταίας αναφοράς/έκθεσης η οποία αναφέρει όλες τις λεπτομέρειες και προσεγγίσεις που ακολουθήθηκαν μέχρι την παράδοση του τελικού μοντέλου μηχανικής μάθησης. Επιπλέον, πολλές φορές γίνεται και παρουσίαση των αποτελεσμάτων όπως και παράδοση σχετικών εγγράφων (π.χ. αναθεώρηση έργου, παρακολούθηση και συντήρηση ακόμα και περαιτέρω σχέδιο δράσης). [2]

Συμπεράσματα:

Μάθαμε λοιπόν τα γνωστότερα βήματα που ακολουθούνται σε μία εξόρυξη δεδομένων, ένας κλάδος αρκετά μεγάλος της Επιστήμης Δεδομένων, ο οποίος συμβάλει στην δημιουργία των μαθηματικών Μοντέλων που προ αναφέραμε. Με αυτά τα μοντέλα, δίνεται η λύση σε προβλήματα που ο απλός προγραμματισμός δεν δίνατε να υλοποιηθεί, όπως αναγνώριση προσώπου σε μία εικόνα. Επίσης όπως είδαμε, κάποια βήματα επαναλαμβάνονται με σκοπό την βελτιστοποίηση των μοντέλων μάθησης σε απόδοση. Η επιστήμη δεδομένων δεν είναι μόνο ο δρόμος απόκτησης, επεξεργασίας, ανάλυσης δεδομένων αλλά είναι και η διαδικασία που θα λάβεις τα δεδομένα από το φυσικό περιβάλλον ή αλλιώς από τον έξω κόσμο, είναι η στατιστική και τα μαθηματικά που έχουν αναπτυχθεί για να μπορέσουν οι αλγόριθμοι και τα μοντέλα μάθησης να υπάρξουν εξ’ αρχής, και είναι ακόμα και η τελική παρουσίαση ή έκθεση που θα κάνεις στην τελική φάση της ανάλυσης σου, και φυσικά όλα τα ενδιάμεσα βήματα που αναφέραμε πιο πάνω.

1.4 Εργαλεία

Τώρα μπαίνουμε σε ένα κεφάλαιο που θα χρειαστεί να χρησιμοποιήσουμε κάποια εργαλεία έτσι ώστε να παρουσιάσουμε μία πετυχημένη εκπαίδευση ενός μοντέλου βαθιάς μάθησης (**Deep Learning Model**). Είναι σημαντικό να αναφέρουμε ότι για να το καταφέρουμε αυτό, θα χρησιμοποιήσουμε κυρίως γλώσσες προγραμματισμού και βιβλιοθήκες αυτών. Μία από τις πολλές γλώσσες προγραμματισμού ικανές για μία έμπρακτη εκπαίδευση μοντέλων, είναι η **Python**, την οποία και θα εντάξουμε σε αυτή την πτυχιακή εργασία.

Η Python είναι μία γλώσσα προγραμματισμού εύκολη στο συντακτικό της, ανοιχτού κώδικα, δηλαδή προσφέρετε και αναπτύσσεται δωρεάν και από την κοινότητα, και είναι ευρέως χρησιμοποιούμενη. Η **Python** γράφτηκε από τον Guido van Rossum ο οποίος για το όνομα εμπνεύστηκε από το ντοκιμαντέρ του BBC «Monty Python's Flying Circus». Προσφέρει συμβατικότητα με πολλές άλλες γλώσσες προγραμματισμού όπως C/C++/Java κ.τ.λ. και βρίσκεται σε πολύ υψηλό επίπεδο προγραμματισμού (**High Level Programming Language**). [4]

Στην επιστημονική κοινότητα που ασχολείται με την Επιστήμη Δεδομένων, είναι ευρέως καλή μέθοδος, ο κώδικας, να γράφετε σε διαδραστικά σημειωματάρια (**Notebooks**) όπως είναι το εργαλείο: **Jupyter** ή το **Google Colab** και όχι με την μορφή σειρών από γραμμές κώδικα (**script**). Αυτό συμβαίνει διότι, καθόλη την διαδικασία της προετοιμασίας των δεδομένων, θα χρειαστεί να τα μετατρέψουμε σε γραφήματα για κατανόηση ή να δοκιμάσουμε κομμάτια από κώδικα (**Block of code)** τα οποία θα κάνουν ένα συγκεκριμένο πράγμα αλλά δεν θέλουμε να ταράξει το υπόλοιπο αρχείο. Επομένως συνιστάται η εκμετάλλευση του Google Colab σαν πλατφόρμα εισαγωγής κώδικα. Επιπροσθέτως το Google Colab μας δίνει την δυνατότητα να τρέξουμε τον κώδικα μας με μεγάλη ταχύτητα εφόσον η υπολογιστική ισχύ είναι αυτής των server της Google. [4]

Επίσης, θα συναντήσουμε την ανάγκη της επεξεργασίας δεδομένων, αναπαράσταση των δεδομένων με γραφικό τρόπο, προετοιμασία των δεδομένων για τις μηχανές μάθησης καθώς και δημιουργία των μοντέλων μηχανών μάθησης. Θα χρησιμοποιήσουμε βιβλιοθήκες της Python όπως: **Pandas**, **Numpy**, **Matplotlib**, **Scikit-Learn**, **TensorFlow** κ.τ.λ. η οποίες θα λύνουν κάθε πρόβλημα που αναφέρθηκε, αντίστοιχα. Ο κώδικας που θα χρησιμοποιήσουμε, θα θεωρείται γνωστός διότι με μία απλή αναζήτηση στο διαδίκτυο και στην βιβλιογραφία μπορεί ο καθένας να ανακαλύψει την λειτουργία κάθε γραμμή κώδικα. Παρόλα αυτά, όλος ο

κώδικας που θα γραφτεί θα δοθεί στο τέλος της πτυχιακής εργασίας, με την μορφή ενός σημειωματάριου *.ipynb* δημιουργημένο από το Google Colab.

1.5 Τύποι Δεδομένων

Όπως είναι γνωστό τα δεδομένα χωρίζονται σε δύο μεγάλες κατηγορίες, τα αριθμητικά δεδομένα και τα κατηγορικά δεδομένα. Τα αριθμητικά δεδομένα είναι οι αριθμοί, και τα κατηγορηματικά δεδομένα ανήκουν στην κατηγορία η οποία περιγράφει οντότητες, καταστάσεις, κ.λ.π. με λέξεις. Παρόλα αυτά θα τα διαχωρίσουμε για λόγους ολοκλήρωσης και έτσι ώστε να έχουμε και μια βασική κατανόηση των ειδών των δεδομένων που μπορούμε να συναντήσουμε.

1.5.1 Κατηγορηματικά Δεδομένα (**Categorical Data**)

- Ονομαστικά Δεδομένα (**Nominal**)

  Τα ονομαστικά δεδομένα αντιπροσωπεύουν διακριτές μονάδες και χρησιμοποιούνται για την επισήμανση μεταβλητών που δεν έχουν ποσοτική τιμή. Με λίγα λόγια θα μπορούσαν να παραφραστούν ως οι λεγόμενες «ετικέτες». Επιπλέον, είναι σημαντικό να αναφέρουμε πως δεν διακρίνονται με τέτοιο τρόπο ώστε να παρουσιάζουν μια σειρά, δηλαδή ακόμα και αν αλλάζαμε την σειρά με την οποία εμφανίζονται το νόημα θα παρέμενε αναλλοίωτο. Ένα απλό παράδειγμα ονομαστικών δεδομένων είναι ο διαχωρισμός των δύο φύλων: Άνδρας, Γυναίκα. [5]

- Κανονικά Δεδομένα (**Ordinal Data**)

  Τα κανονικά δεδομένα αντιπροσωπεύουν διακριτές και διατεταγμένες τιμές. Συμπερασματικά, λοιπόν, είναι παρόμοια με τα ονομαστικά δεδομένα, εκτός από μία διαφορά: η σημασία τοποθέτησής στην διατύπωση τους έχει αλλοίωση στο νόημα. Για παράδειγμα:

  1 Δημοτικό
  2 Γυμνάσιο
  3 Λύκειο
  4 Πανεπιστήμιο/Κολλέγιο

  Είναι εύλογο να συμπεράνουμε πως η διατύπωση πρώτα του Γυμνασίου και μετέπειτα του Δημοτικού, για παράδειγμα, δεν θα έβγαζε λογική επαγωγή των Δεδομένων. Όμως, ένα πρόβλημα που δημιουργείται είναι πως η ποσοτική τους «σημασία» δεν ορίζεται παρά μονό η διαφοροποίηση με τα υπόλοιπα δεδομένα.

Δηλαδή, δεν μπορούμε να μετρήσουμε το Δημοτικό αλλά μπορούμε να πούμε ότι βρίσκεται πριν το Γυμνάσιο. Επομένως τον λόγο η εκμετάλλευσης των Κανονικών Δεδομένων συνήθως γίνεται για την μέτρηση μη αριθμητικών χαρακτηριστικών όπως η ευτυχία, η ικανοποίηση, ευχαρίστηση, κ.τ.λ. [5]

1.5.2 Αριθμητικά Δεδομένα (**Numerical Data**)

- Διακριτά Δεδομένα (Discrete Data)

  Τα διακριτά δεδομένα είναι αριθμητικές τιμές οι οποίες είναι ξεχωριστές η μία τιμή με την άλλη. Αυτές οι τιμές δεν υπολογίστηκαν αλλά μετρήθηκαν, όπως μετράμε για παράδειγμα πόσες φορές ήρθε κορώνα ή γράμματα σε 100 ρίψεις ενός νομίσματος. Είναι δηλαδή, πληροφορία που μπορεί να κατηγοριοποιηθεί σε μία κλάση. [5]

  Ένας τρόπος για να τα διακρίνουμε είναι να κάνουμε τις εξής ερωτήσεις:

  α) Μπορούμε να το μετρήσουμε?

  β) Αν ναι, τότε μπορούμε να κάνουμε αριθμητικές πράξεις πάνω σε αυτά?

  Τότε βρισκόμαστε σε τύπο δεδομένων που είναι διακριτά αριθμητικά δεδομένα.

- Συνεχή Δεδομένα (**Continuous Data**)

  Τα συνεχή δεδομένα είναι αριθμητικές τιμές οι οποίες έχουν υπολογιστεί και όχι μετρηθεί, όπως αντίθετα οι διακριτές. Εύκολα μπορούμε να τα διακρίνουμε διότι παίρνουν τιμές αποδεκτές μόνο ανάμεσα σε κάποια όρια, όπως είναι το ύψος ενός ανθρώπου, που δεν μπορεί να έχει αρνητική τιμή, το βάρος του, κ.τ.λ.

  Είναι σημαντικό να αναφέρουμε όμως ότι μπορούν να πάρουν οποιαδήποτε τιμή μέσα σε αυτά τα όρια, δηλαδή μπορεί το βάρος να είναι 80.1 ή και 80.25 σε αντίθεση με τα διακριτά τα οποία στο παράδειγμα με το νόμισμα, δεν μπορούν να πάρουν δεκαδικές τιμές όπως 49.5 φορές ήρθε κορώνα. [5]

  Στα Συνεχή δεδομένα διακρίνουμε άλλες δύο υποκατηγορίες δεδομένων:

  - Τιμές Διαστήματος (**Interval Data**)

    Οι τιμές διαστήματος, είναι τιμές διατεταγμένες αλλά έχουν μεταξύ τους την ίδια «απόσταση». Για παράδειγμα οι τιμές στην θερμοκρασία του περιβάλλοντος τις οποίες όμως έχουμε ορίσει, για κάποιο λόγο σε κάποιο πρόβλημα, να έχουν αποστάσεις 5 C. [5]

-5	0	5	10	15

- Τιμές Αναλογίας (**Ratio Data**)

  Στα συνεχή δεδομένα συναντάμε και τις τιμές αναλογίας, οι οποίες έχουν την ίδια φιλοσοφία αυτής της τιμές διαστήματος, με κύρια διαφορά όμως την ύπαρξη του 0 ως μηδενικό στοιχείο στην πράξη της πρόσθεση. Με διαφορετικά λόγια, εννοούμε πως οι τιμές αναλογίας θα έχουν το νούμερο που συμβολίζει την μη ύπαρξη αυτού που μετράει, για παράδειγμα το μήκος ενός αντικειμένου, ένα έχει 0 εκατοστά σημαίνει ότι το αντικείμενο στην ουσία δεν έχει μήκος. Σε αντίθεση με τις τιμές Διαστήματος που το 0, όπως 0 βαθμούς θερμοκρασίας δεν συνεπάγεται την μη ύπαρξη θερμοκρασίας. [5]

2. Εισαγωγή στην Βαθιά Μάθηση (Deep Learning)

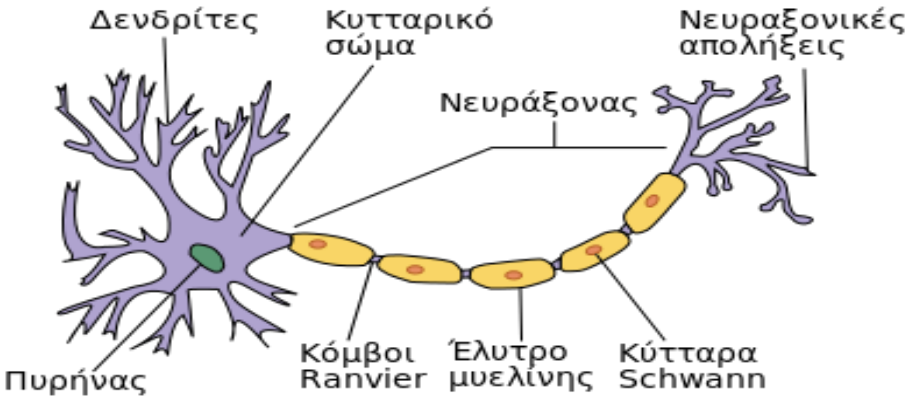
Ο εγκέφαλος είναι το πιο καταπληκτικό όργανο σε έναν ζωντανό οργανισμό όπως ο άνθρωπος. Μας βοηθά να δηλώσουμε αισθήσεις τους περιβάλλοντος όπως η όραση, το άκουσμα, την μυρωδιά, την γεύση και την υφή. Μας επιτρέπει να αποθηκεύσουμε και να βιώσουμε αναμνήσεις, συναισθήματα ακόμα και να ονειρευτούμε. Χωρίς αυτό θα υπήρχε ανικανότητα στην αντίληψη, σαν γενικότερη έννοια, έχοντας μόνο τις βασικές αρχές των αντανακλαστικών. Επομένως, ο εγκέφαλος είναι το σπουδαιότερο όργανο μας και αυτό που μας δίνει την νοημοσύνη μας. [6]

Ο εγκέφαλος ενός νεογνού ζυγίζει μόλις μισό κιλό περίπου, αλλά παρόλα αυτά βρίσκει λύσεις σε προβλήματα που ο πιο δυνατός υπερ-υπολογιστής μας θεωρεί αδύνατα. Μόλις σε μερικούς μήνες μετά την γέννα, έχουν την ικανότητα να αναγνωρίσουν πρόσωπα όπως αυτών των γονιών και των συγγενών τους, να ξεχωρίζουν αντικείμενα από το περιβάλλον τους ακόμα και να ξεχωρίζουν φωνές. Μέσα στο πρώτο έτος, έχουν καταλάβει τις βασικές αρχές της φυσικής, όπως την βιώνουν, και μέσα στα επόμενα χρόνια εξελίσσουν την ικανότητα να πράττουν γραμματική και το λεξιλόγιό τους πάνω στην μητρική γλώσσα. [6]

Για δεκαετίες, η ονειροπόληση της δημιουργίας ευφυών συστημάτων παρόμοια του ανθρώπινου εγκεφάλου οργίαζε. Όμως η δημιουργία τέτοιων ευφυών συστημάτων απαιτεί την επίλυση των δυσκολότερων και των περιπλοκότερων υπολογιστικών προβλημάτων που έχει έρθει ποτέ η ανθρωπότητα αντιμέτωπη. Αυτός ο τομέα θεωρείται από τους πιο ενεργούς της τεχνητής νοημοσύνης, συχνά αποκαλούμενος ως Βαθιά Μάθηση (**Deep Learning**). [6]

2.1 Εισαγωγή στα Νευρωνικά Δίκτυα

Το θεμελιώδες μοναδιαίο μέτρο ενός ανθρώπινου εγκεφάλου είναι ένας νευρώνας. Μια πολύ μικρή περιοχή του εγκεφάλου στο μέγεθος ενός κόκκου ρυζιού περιέχει πάνω από 10.000 νευρώνες κάθε ένας από τους οποίους έχει κατά μέσο όρο 6.000 συνδέσεις με τους υπόλοιπους νευρώνες. Στην παρακάτω εικόνα 1 παρατηρούμε την βιολογική μορφή ενός νευρώνα, αυτό το ενοποιημένο σύστημα είναι υπεύθυνο για την λειτουργία του εγκεφάλου ως έχει. Στόχος μας είναι η παρομοίωση αυτού του ενοποιημένου συστήματος στους υπολογιστές έτσι ώστε να δημιουργηθούν αντίστοιχα μηχανικά μοντέλα κατασκευασμένα από εμάς με σκοπό να «λύνουν» αντίστοιχα προβλήματα. [6]



Εικόνα 1: Σχηματικό διάγραμμα ενός τυπικού βιολογικού νευρώνα.

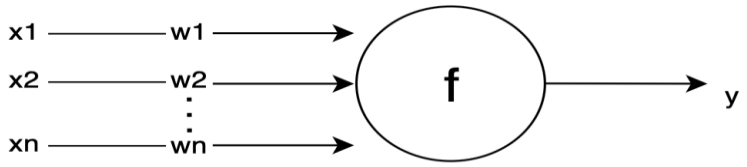
Στην εικόνα 2 προσφέρετε έτσι η γενικότερη ιδέα στην λειτουργία των νευρώνων:

Εικόνα 2: Περιγραφική λειτουργία ενός βιολογικού νευρώνα, με βήματα. [6]

Καθίσταται πλέον εφικτό να παρομοιάσουμε την λειτουργία ενός βιολογικού νευρώνα με τέτοιο τρόπο έτσι ώστε ένας υπολογιστής να μπορεί να «μαθαίνει» όπως ένας ανθρώπινος εγκέφαλος. Μία δημιουργία ενός τέτοιου μοντέλου, με μαθηματικό ορισμό, προτάθηκε για πρώτη φορά το 1943 στην εργασία των Warren S. McCulloch και Walter H. Pitts.[2] Όπως οι βιολογικοί νευρώνες έτσι και οι τεχνικοί λαμβάνουν κάποια αριθμητικά εισαγωγικά δεδομένα ή αλλιώς μεταβλητές $x_1, x_2, \dots, x_n$, οι οποίες μετέπειτα πολλαπλασιάζονται με κάποια προϋπάρχοντα αριθμητικά «βάρη» $w_1, w_2, \dots, w_n$, [6]. Αυτές οι καινούργιες τιμές προστίθενται μεταξύ τους για να παράγουν ένα τελικό αποτέλεσμα $z = \sum_{i=0}^{n} w_i\, x_i$ το οποίο

[2] McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The Bulletin of Mathematical Biophysics. 5.4 (1943): 115-133.

εισέρχεται σε μία συνάρτηση $f(x)$, η οποία ορίζεται από εμάς, παράγοντας έτσι το τελικό προϊόν $y = f(z)$. Αυτή η τιμή y μεταφέρεται σε άλλους τεχνητούς νευρώνες. [6]



Εικόνα 3: Σχηματική αναπαράσταση ενός τεχνητού νευρώνα σε ένα τεχνητό νευρωνικό δίκτυο.

Ολοκληρώνοντας την μαθηματική επεξήγηση ενός τεχνητού νευρώνα, μένει να εξηγήσουμε την λειτουργία του χρησιμοποιώντας διανύσματα. Εάν παραμετροποιήσουμε τις εισαγόμενες τιμές (inputs) σαν ένα διάνυσμα, δηλαδή $\bar{\mathbf{x}} = [\mathrm{x}_1\ \mathrm{x}_2\ \ldots\ \mathrm{x}_\mathrm{n}]$ και τα βάρη ως $\bar{\boldsymbol{w}} = [\mathrm{w}_1\ w_2\ \ldots w_n]$. Τότε μπορούμε να ξανά ορίσουμε την έξοδο (**output**) του νευρώνα ως $y = f(\bar{\mathbf{x}} * \bar{\mathbf{w}} + b)$, όπου b θεωρείται ως ο όρος μεροληψίας. Με απλά λόγια υπολογίζοντας το τελικό αποτέλεσμα του αθροίσματος του όρου μεροληψίας με το εσωτερικό γινόμενο των διανυσμάτων των εισαγόμενων τιμών με το διάνυσμα των βαρών και τελικώς, εισάγοντας αυτό το τελικό αποτέλεσμα στην συνάρτηση $f(x)$ έχουμε το εξαγόμενο αποτέλεσμα y το οποίο πολύ πιθανό είναι να τροφοδοτεί άλλον νευρώνα. [6]

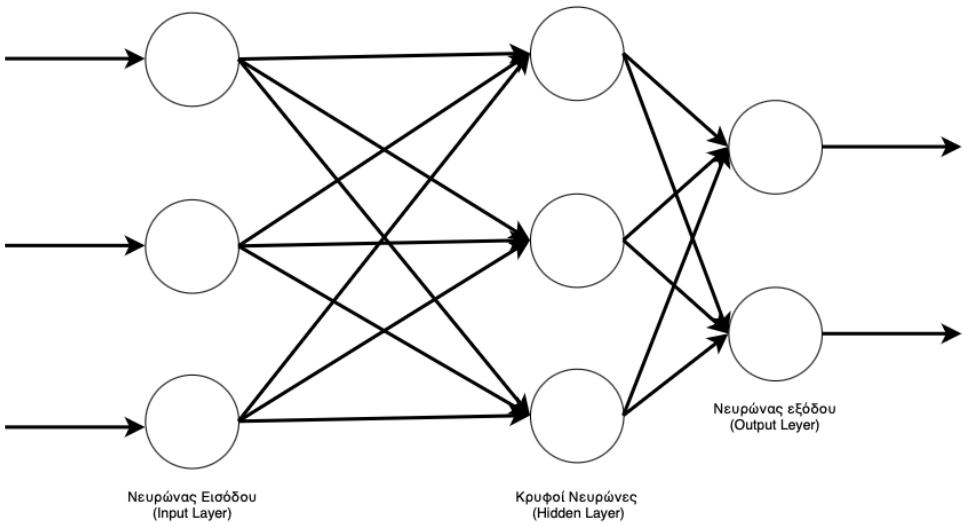
2.1.1 Νευρωνικά Δίκτυα τροφοδότησης προς τα εμπρός

Όπως αναφέραμε πιο πάνω, είναι αξιοσημείωτο ότι ένας εγκέφαλος έχει χιλιάδες συνδέσεις με άλλους χιλιάδες νευρώνες σε όγκο ενός κόκκου ρυζιού και αυτό μας φέρνει σε μία μεγάλη παρατήρηση. Υπάρχει λόγος που ο εγκέφαλος μας έχει πάνω από έναν νευρώνα, και αυτό διότι είναι αδύνατο για έναν μοναδικό νευρώνα να λύσει περίπλοκα προβλήματα όπως η αναγνώριση αριθμών. Έτσι, και οι τεχνητοί νευρώνες, ακολουθούν την ίδια τακτική και δημιουργούμε νευρωνικά δίκτυα που περιέχουν πάνω από έναν νευρώνα για την επίλυση περίπλοκων προβλημάτων. [6]

Παρόλα αυτά συναντάμε και το φαινόμενο των «επιπέδων» από νευρώνες ακόμα και σε έναν βιολογικό εγκέφαλο. Πιο αναλυτικά, ο ανθρώπινος εγκεφαλικός φλοιός (cerebral cortex), ο οποίος είναι υπεύθυνος για το μεγαλύτερο μέρος της ανθρώπινης νοημοσύνης, αποτελείται από 6 επίπεδα στρώσεων από νευρώνες [3]. Στην όραση η πληροφορία που εισάγεται από τα

[3] Mountcastle, Vernon B. "Modality and topographic properties of single neurons of cat's somatic sensory cor- tex." *Journal of Neurophysiology* 20.4 (1957): 408-434.

μάτια περνάει και επεξεργάζεται από «επίπεδο σε επίπεδο» μέχρι το 6 και τελευταίο επίπεδο όπου πλέον η εισακτέα τιμή (πληροφορία από τα μάτια) μπορεί πλέον να μεταφραστεί σε κατανόηση αυτού που βλέπουμε. Στην εικόνα 4, έχουμε μια γραφική απεικόνιση ενός τεχνητού νευρωνικού δικτύου με «επίπεδα». Πρώτα έχουμε το επίπεδο εισόδου όπου νευρώνες λαμβάνουν την εισαγωγική τιμή τους, έπειτα έχουμε τα κρυφά επίπεδα στα οποία γίνεται η όλη «μαγεία» των νευρώνων, ο αριθμός των κρυφών επιπέδων μπορεί να είναι 1 ή και παραπάνω. Τελευταίο επίπεδο είναι πάντα το επίπεδο εξόδου το οποίο είναι εκείνο που μας δίνει την «απάντηση» ή το αποτέλεσμα με την μορφή αριθμητικής τιμής, συνήθως τιμή πιθανότητας. Επιπλέον, υπάρχει η πιθανότητα να είναι πάνω από μία έξοδο, συνήθως είναι τόσοι έξοδοι όσες και οι πιθανές επιλογές απάντησης ενός προβλήματος, για παράδειγμα αν μία εικόνα περιέχει μία γάτα και όχι έναν σκύλο τότε για τις δύο επιλογές αυτές θα έχουμε δύο εξόδους από το σύστημα μας, το οποίο θα δίνεται σε μορφή διανύσματος με 2 τιμές: [1, 0]. Αν περιέχει γάτα και σκύλο τότε θα είχαμε έξοδο το διάνυσμα [1, 1] και αντιστοίχως αν δεν είχε κανένα από τα δύο, το διάνυσμα [0, 0]. [6]



Εικόνα 4: Γραφική παρουσίαση ενός προς-τα-εμπρός νευρωνικού δικτύου ανατροφοδότησης με 3 επίπεδα (ενός επιπέδου εισόδου από 3 νευρώνες, ενός κρυφού επιπέδου από 3 νευρώνες και ενός επιπέδου εξόδου από 2 νευρώνες).

2.1.2 Γραμμικοί Νευρώνες και περιορισμοί

Η πιο απλή μορφή ενός τεχνητού νευρώνα είναι ένας νευρώνας Perceptron. Ο λεγόμενος νευρώνας είναι ένας γραμμικός νευρώνας δηλαδή η συνάρτηση $f(x)$, που έχουμε δει ότι είναι βασική λειτουργία ενός νευρώνα, είναι της μορφής $f(z) = az + b$ με $a, b = αριθμοί$.

Οι Γραμμικοί Νευρώνες (**Linear Neurons**) παρόλο που είναι γρήγοροι στον υπολογισμό τους, διατρέχουν ένα μεγάλο μειονέκτημα. Έχει αποδειχθεί πως τα νευρωνικά δίκτυα με προς-τα-εμπρός νευρώνες τροφοδότησης (Κεφάλαιο 2.1.1) με κάθε νευρώνας να είναι γραμμικής μορφής, μπορούν να αναπαρασταθούν ή αλλιώς να αντικατασταθούν με έναν αντίστοιχο ο οποίος δεν έχει κρυφά επίπεδα (**Hidden Layers**). Αυτό, είναι και το μεγαλύτερο μειονέκτημα, διότι όπως έχουμε δει προηγουμένως η όλη λειτουργία των νευρώνων εξαρτάται κυρίως από τα κρυφά επίπεδα που έχει, με αποτέλεσμα έτσι να χάνουμε σε πολυπλοκότητα των νευρώνων. Συμπερασματικά, λοιπόν, οι μη γραμμικοί νευρώνες θα μας δώσουν καλύτερες επιδόσεις, τουλάχιστον σε θεωρητικό επίπεδο, και όπως θα δούμε και σε πρακτικό. [6]

2.1.3 Μη γραμμικές συναρτήσεις για Νευρωνικά

Είδαμε πως τα γραμμικά νευρωνικά δίκτυα, δηλαδή προς τα εμπρός δίκτυα τροφοδότησης με γραμμικούς νευρώνες, έχουν μεγάλο μειονέκτημα στην πολυπλοκότητα εφόσον μπορούν να αναπαρασταθούν ως δίκτυα με νευρώνες χωρίς κρυφά επίπεδα μάθησης. Παρόλα αυτά υπάρχει λύση με το να αντικαταστήσουμε της συναρτήσεις των νευρώνων με μη γραμμικές.

Μερικές από τις πιο γνωστές μη γραμμικές συναρτήσεις που η επιστημονική κοινότητα χρησιμοποιεί είναι η σιγμοειδής (**sigmoid**), η υπερβολική εφαπτομένη **tanh** ($\boldsymbol{x}$) αλλά και η **ReLU**. Προφανώς υπάρχουν και άλλες αλλά θα αναφερθούμε για αρχή σε αυτές τις 3. [6]

ReLU:Είναι η συνάρτηση που δίνει τον ίδιο αριθμό αν η είσοδος του είναι θετικός, αλλιώς δίνει το 0.

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

Sigmoid (Σιγμοειδής):

Η σιγμοειδής χρησιμοποιείται κυρίως από στατιστικολόγους και μαθηματικούς ως μία συνάρτηση πιθανότητας. Δηλαδή, έχει για έξοδο μία πιθανότητα με διάστημα 0 έως και 1. Επιπλέον αυτό σημαίνει πως όταν οι τιμές εισόδου είναι μικρές τότε το αποτέλεσμα της πιθανότητας είναι επίσης μικρό, και αντίστοιχα όταν είναι μεγάλες το αποτέλεσμα της σιγμοειδής συνάρτησης πλησιάζει την μονάδα. [6]

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

Tanh (Υπερβολική εφαπτομένη):

Οι νευρώνες που χρησιμοποιούν την υπερβολική εφαπτομένη δίνουν στα μοντέλα τους την μη γραμμικότητα σχήματος S (**S-shaped nonlinearity**). Η διαφορά της με την σιγμοειδής

είναι ότι το διάστημα αποτελεσμάτων της είναι από -1 έως και 1. Προτιμάται κυρίως από την σιγμοειδή διότι το 0 είναι το κεντρικό σημείο του διαστήματος που δίνει (**zero-centered**). [6]

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

2.1.4 Συνάρτηση SoftMax Στα Επίπεδα Εξόδου (Output Layers)

Στα επίπεδα εξόδου, οι νευρώνες θα έχουν για έξοδο ένα διάνυσμα με τιμές. Σε παρακάτω κεφάλαιο θα μάθουμε τα προβλήματα ταξινόμησης (Κεφάλαιο 2.3 Αρχές Μάθησης), τα οποία είναι εκείνα που θέλουμε τα νευρωνικά δίκτυα να ταξινομούν σε κατηγορίες τα δεδομένα μας. Στα προβλήματα αυτά είναι αναγκαία η ύπαρξη πιθανότητας διότι έτσι θα γνωρίζουμε ποια είναι η πιθανότητα σε μία εικόνα, για παράδειγμα, να απεικονίζεται σκύλος ή γάτα. Δηλαδή να υπάρχει μία κατανομή πιθανότητας σε ένα σύνολο αμοιβαία αποκλειστικών ετικετών.

Αυτός όμως, είναι δύσκολο να επιτευχθεί σωστά χωρίς την ύπαρξη της κατάλληλης συνάρτησης, έτσι ώστε αντί για αριθμητικές τιμές χωρίς σημασία να δίνει τιμές που θα απεικονίζουν πιθανότητες αποκλειστικών ετικετών. Έτσι θα έχουμε πιθανότητες για το τι θεωρεί πιο πιθανό σε μία εικόνα να υπάρχει. Η συνάρτηση που το καταφέρνει εύκολα αυτό λέγεται **SoftMax**. Αυτή η συνάρτηση χρησιμοποιείται για την δημιουργία ενός επιπλέον στρώματος εξόδου έτσι ώστε θα μετατρέπει τις τιμές του νευρωνικού δικτύου σε διάνυσμα πιθανοτήτων. Μία ισχυρή πιθανότητα μίας ετικέτας θα πλησιάζει την τιμή 1 ενώ μία λιγότερο έως απίθανη ετικέτα θα πλησιάζει την τιμή 0. [6]

2.2 Εισαγωγή στην Μηχανική Όραση

Η μηχανική όραση ανήκει στον τομέα της Επιστήμης Δεδομένων και συγκεκριμένα ανήκει στις υπολογιστικές μεθόδους της Επιστήμης Δεδομένων. Βρίσκεται στην υποκατηγορία αυτή της Μηχανικής Μάθησης και χρησιμοποιεί κυρίως νευρωνικά δίκτυα για να μπορέσει να επιτευχθεί.

Όπως έχουμε ήδη αναφέρει για τα νευρωνικά δίκτυα και την φιλοσοφία τους, να αντιγράφουν την λογική ενός βιολογικού νευρώνα στον εγκέφαλο ενός οργανισμού, έτσι και η μηχανική όραση δεν είναι τίποτα παραπάνω από νευρωνικά δίκτυα ειδικά σχεδιασμένα για να μπορέσει ένας υπολογιστής να «δει». Όμως ένας βιολογικός εγκέφαλος μπορεί να αντιληφθεί αντικείμενα, μέσω της όρασης, σε κλάσματα του δευτερολέπτου χωρίς ενδοιασμούς, μπορεί να τα διακρίνει από το περιβάλλον τους και να ξεχωρίσει το βάθος και το σχήμα τους στιγμιαία. [6]

Η πορεία της πληροφορίας που εισέρχεται από τα μάτια μας και μετατρέπεται σε νόηση αυτού που βλέπουμε είναι αξιοθαύμαστη. Παρόλα αυτά, όσο εύκολο είναι για εμάς να το κάνουμε άλλο τόσο δύσκολο είναι για έναν υπολογιστή. Όμως, όπως η όραση μετατρέπει αυτές τις εκπομπές φωτός σε πληροφορία η οποία επεξεργάζεται με δίκτυα νευρώνων στον

εγκέφαλο[4], μας προϊδεάζει πως με την εξέλιξη των νευρωνικών δικτύων που έχουμε, υπάρχει η πιθανότητα να δημιουργήσουμε ένα μοντέλο τόσο κατάλληλο που θα επιτρέπει την μηχανική όραση ως έναν βαθμό. [6]

2.2.1 Εξαγωγή Χαρακτηριστικών – Η λάθος Προσέγγιση

Το 2001 η Paul Viola και ο Michael Jones έδωσαν μια παρουσίαση που παρουσίασαν μία εικόνα ενός τυχαία επιλέξιμου ανθρώπου , και ζήτησαν από το κοινό να αναγνωρίσει τα πρόσωπα σε αυτή την εικόνα [5]. Προφανώς για έναν άνθρωπο αυτό θα ήταν ιδιαίτερα εύκολη εργασία αλλά για έναν υπολογιστή σχεδόν αδύνατο τότε.

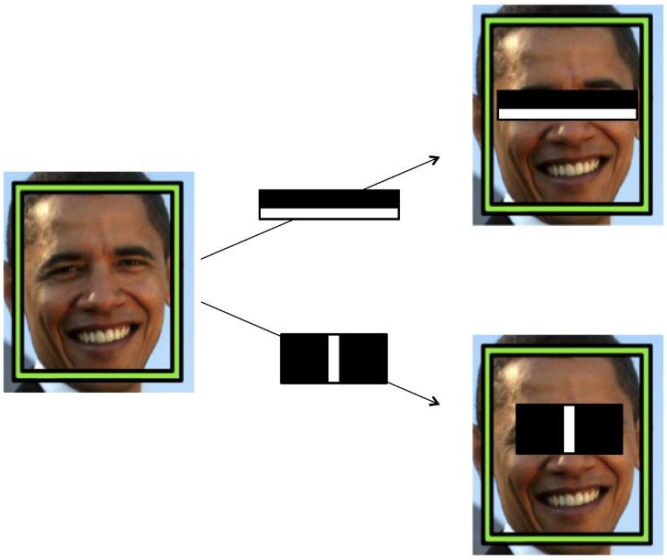
Πως θα μπορούσαμε να αναγνωρίσουμε πρόσωπα σε μία εικόνα με αλγορίθμους της μηχανικής μάθησης; Συμπερασματικά, αποφάνθηκε ότι δεν είναι υλοποιήσιμο διότι τα απτά δεδομένα από τα pixel της εικόνας έδιναν υψηλά ποσά από «θόρυβο» στο πρόβλημα ταξινόμησης αφού δεν «ήξερε» ο αλγόριθμος τι ακριβώς έπρεπε να μάθει. [6]

Επομένως, για επίλυση του υψηλού θορύβου που παρουσιαζόταν, κάνανε επιλογή των χαρακτηριστικών που θέλανε να κρατήσουν μειώνοντας έτσι τις διαστάσεις που συναντούσανε, μία τεχνική που ονομάζεται **feature extraction** και από αυτές συνέχιζαν με κάποιον αλγόριθμο ταξινόμησης. Καλό είναι να αναφέρουμε πως με τον όρο «αλγόριθμοι ταξινόμησης» εννοούμε μεθόδους μηχανικής μάθησης που δεν χρησιμοποιούν τεχνικές νευρωνικών δικτύων. Στην εικόνα 5 θα δούμε μία αναπαράσταση της ιδέας των Viola και Jones όπου αποφάσισαν να διαλέξουν ως σημαντικό χαρακτηριστικό την διαφορά της έντασης του φωτός σε ένα πρόσωπο ανάμεσα στα μάτια με την μύτη ή ανάμεσα των ματιών με τα μάγουλα. [6]

---

4 Cohen, Adolph I. "Rods and Cones." *Physiology of Photoreceptor Organs*. Springer Berlin Heidelberg, 1972. 63-110.

5 Viola, Paul, and Michael Jones. "Rapid Object Detection using a Boosted Cascade of Simple Features." Com- puter Vision and Pattern Recognition, 2001. CVPR 2001. *Proceedings of the 2001 IEEE Computer Society Con- ference* on. Vol. 1. IEEE, 2001.



Εικόνα 5: Παρουσίαση της ιδέα των Viola-Jones με τους ανιχνευτές έντασης (Intensity detectors) [6]

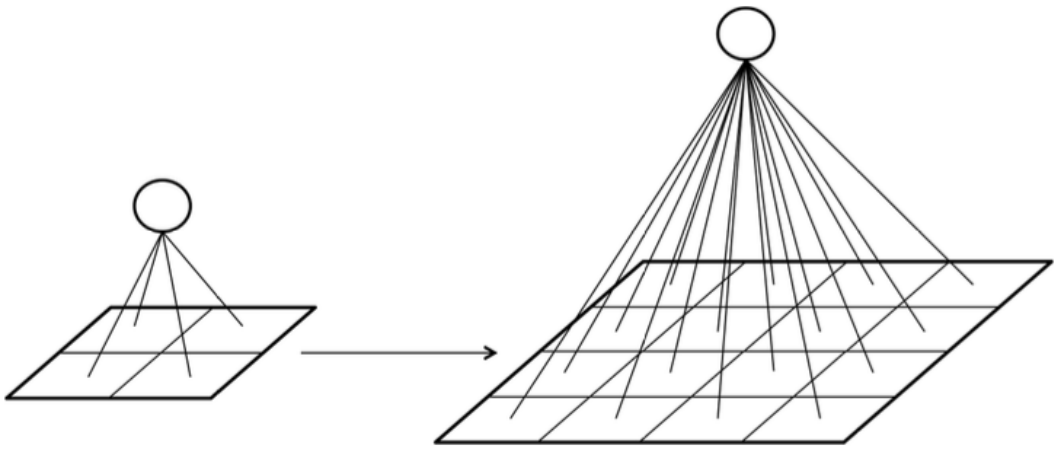
Αυτή η τεχνική ήταν αρκετά υποσχόμενη με καλύτερες επιδόσεις από ότι υπήρχε εκείνη την εποχή. Όμως, υπήρχε ένα μεγάλο ελάττωμα, σε περίπτωση που σε μία εικόνα εμφανίζεται ένα πρόσωπο στο οποίο το φως να πέφτει από διαφορετικές γωνίες, δημιουργώντας ή εξαφανίζοντας σκιές, έκανε την ιδέα των ανιχνευτών έντασης (intensity detectors) μη βιώσιμη. Επομένως, όπως θα δούμε, δημιουργήθηκαν διαφορετικές τεχνικές μες την πάροδο των χρόνων για την επίτευξη της μηχανικής όρασης και αναγνώρισης προσώπων σε μία εικόνα.

2.2.2 Απλά Νευρωνικά Δίκτυα – Η λάθος Προσέγγιση

Παρόλο που η πτυχιακή εργασία αφοσιώνεται στα νευρωνικά δίκτυα και κατά συνέπεια στην μηχανική όραση, είναι σημαντικό να εξηγήσουμε για ποιον λόγο τα απλά νευρωνικά δίκτυα θεωρούνται λάθος προσέγγιση, από το ευρύ επιστημονικό κοινό.

Στην ενότητα 2.1 στην οποία αναλύουμε τα νευρωνικά δίκτυα προς τα εμπρός, αναφέραμε τα κρυφά επίπεδα αυτών. Εκεί υπάρχει μία σημαντική λεπτομέρεια, κάθε νευρώνας συνδέεται με όλους τους επόμενους νευρώνες δημιουργώντας κατά την μάθηση του τα κατάλληλα βάρη, τότε λέμε και ότι ο νευρώνας έχει εκπαιδευτεί. Όμως, στην μηχανική όραση, δηλαδή όταν προσπαθήσουμε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο να μάθει από εικόνες, του δίνουμε για τιμή εισόδου όλα τα pixel της εικόνας. Είναι εύλογο κανείς να παρατηρήσει πως με μία εικόνα μεγέθους 200 x 200 pixels το διάνυσμα εισόδου θα είναι μεγέθους 40.000, και άμα λάβουμε υπόψιν και τα χρώματα κόκκινο, πράσινο, μπλε (**RGB**) τότε μόνο ο νευρώνας εισόδου έχει ένα διάνυσμα εισόδου μεγέθους 40.000 x 3 = 120.000. Ο υπολογισμός αυτός είναι μόνο για το διάνυσμα εισόδου, σκεφτείτε τι μπορεί να γίνει αν στο

νευρωνικό δίκτυο δημιουργήσουμε και άλλους νευρώνες στα κρυφά επίπεδα. Στην εικόνα 6 θα δώσουμε ένα παράδειγμα ενός νευρώνα που καθώς αυξάνεται η εικόνα σε pixels τόσο αυξάνονται και οι ενώσεις του νευρώνα. [6]



Εικόνα 6: Αναπαράσταση της αύξησης των ενώσεων ενός νευρωνικού νευρώνα καθώς η εικόνα μεγαλώνει σε ανάλυση. [6]

Με αυτό τον τρόπο συμπεραίνουμε πως εύκολα, μία απλή και σχετικά μικρή σε μέγεθος εικόνα θα χρησιμοποιεί απίστευτα μεγάλη υπολογιστική ισχύ μόνο για την εκπαίδευση ενός νευρωνικού δικτύου και αυτό καθιστά τα απλά νευρωνικά δίκτυα μη βιώσιμα. Όμως, την απάντηση σε αυτή την δυσκολία την δίνουμε στο επόμενο κεφάλαιο με τα Συνελικτικά Νευρωνικά Δίκτυα (**Convolutional Neural Networks**). [6]

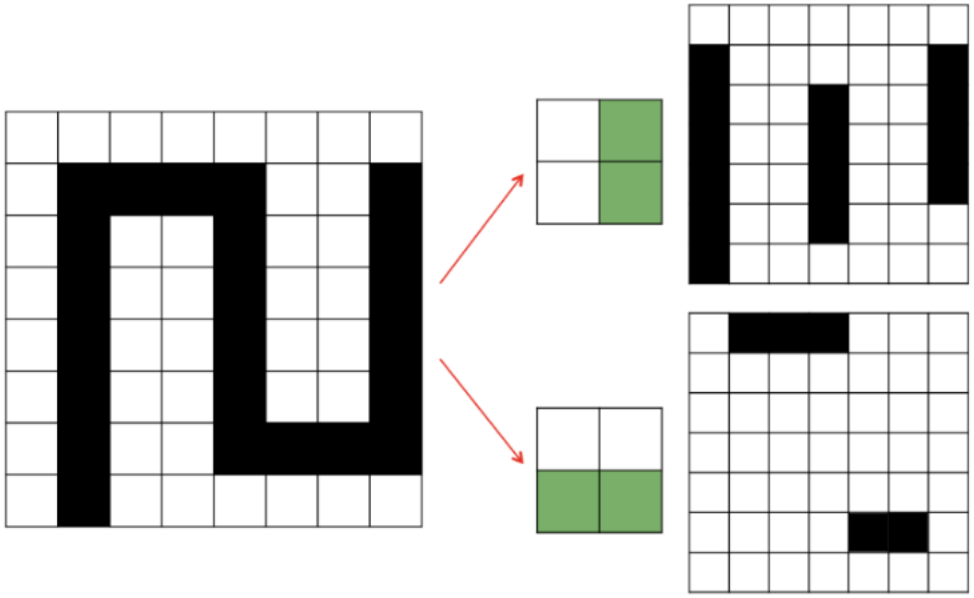
2.2.3 Συνελικτικά Νευρωνικά Δίκτυα (**Convolutional Neural Networks**)

Μία από τις πιο ενδιαφέρουσες εργασίες πάνω στην ανθρώπινη όραση και κατανόηση αυτής, ήρθε στην διαφάνεια από τους David Hubel και Torsten Wiesel το 1959 [6], όπου ανακάλυψαν συνδέοντας ηλεκτρόδια στον εγκέφαλο μίας γάτας που έβλεπε σχήματα, πως υπάρχει μέρος του βιολογικού εγκεφάλου όπου αναγνωρίζει μόνο την γεωμετρία των σχημάτων και συγκεκριμένα τις γωνίες αυτών. Σε συνέχεια της έρευνάς τους ανακάλυψαν πως ο οπτικός φλοιός ήταν οργανωμένος σε στρώματα που κάθε στρώμα ήταν υπεύθυνο για την ανάπτυξη των χαρακτηριστικών που ανιχνεύονται σε προηγούμενα στρώματα από γραμμές, περιγράμματα, σχήματα μέχρι ολόκληρα αντικείμενα! Αυτό μαζί με το γεγονός πως οι ίδιοι ανιχνευτές χαρακτηριστικών αναπαράγονται σε ολόκληρη την εικόνα προκειμένου να

[6] Hubel, David H., and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." *The Journal of Physiology* 148.3 (1959): 574-591.

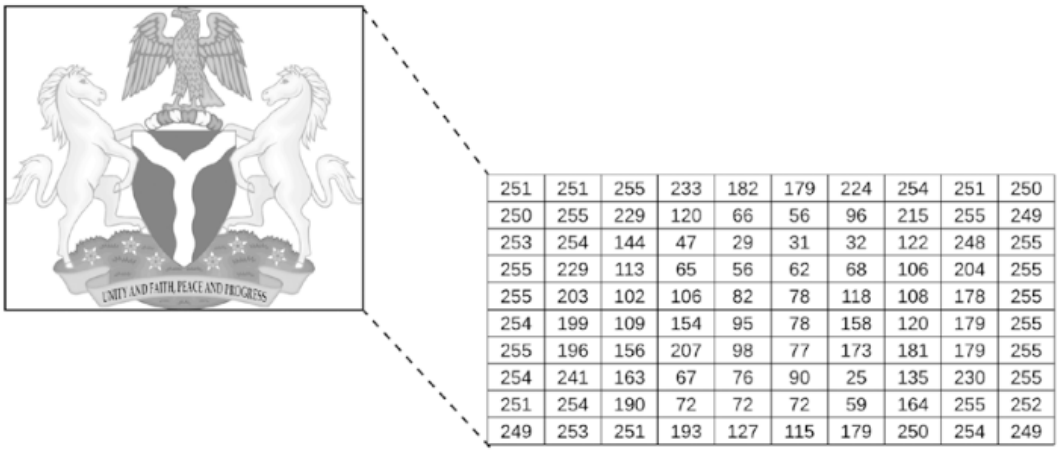
ανιχνεύσουν αυτά τα χαρακτηριστικά, έθεσε τα θεμέλια για την ανάπτυξη των συνελικτικών δικτύων που κάνουν ακριβώς αυτό. [6]

Αυτό επιτυγχάνεται με την εισαγωγή φίλτρων στα κρυφά επίπεδα των νευρωνικών δικτύων. Με τον όρο «φίλτρα» εννοούμε ένα στρώμα του νευρωνικού δικτύου το οποίο θα ανιχνεύει χαρακτηριστικά στην εικόνα και κατά συνέπεια η έξοδός του θα δοθεί στο επόμενο στρώμα του νευρωνικού δικτύου για περαιτέρω εκμάθηση. Αυτά τα χαρακτηριστικά θα μπορούσαν να είναι κάθετες, οριζόντιες, λοξές γραμμές και οτιδήποτε άλλο θεωρούμε ως γεωμετρικό και σημαντικό χαρακτηριστικό για τις εικόνες μας. Για παράδειγμα στην εικόνα 7 οι κάθετες γραμμές τις εικόνας εμφανίζονται μετά την εφαρμογή του φίλτρου, όπως και οι οριζόντιες γραμμές μετά από εφαρμογή διαφορετικού φίλτρου υπεύθυνου για την εμφάνιση οριζόντιων γραμμών.



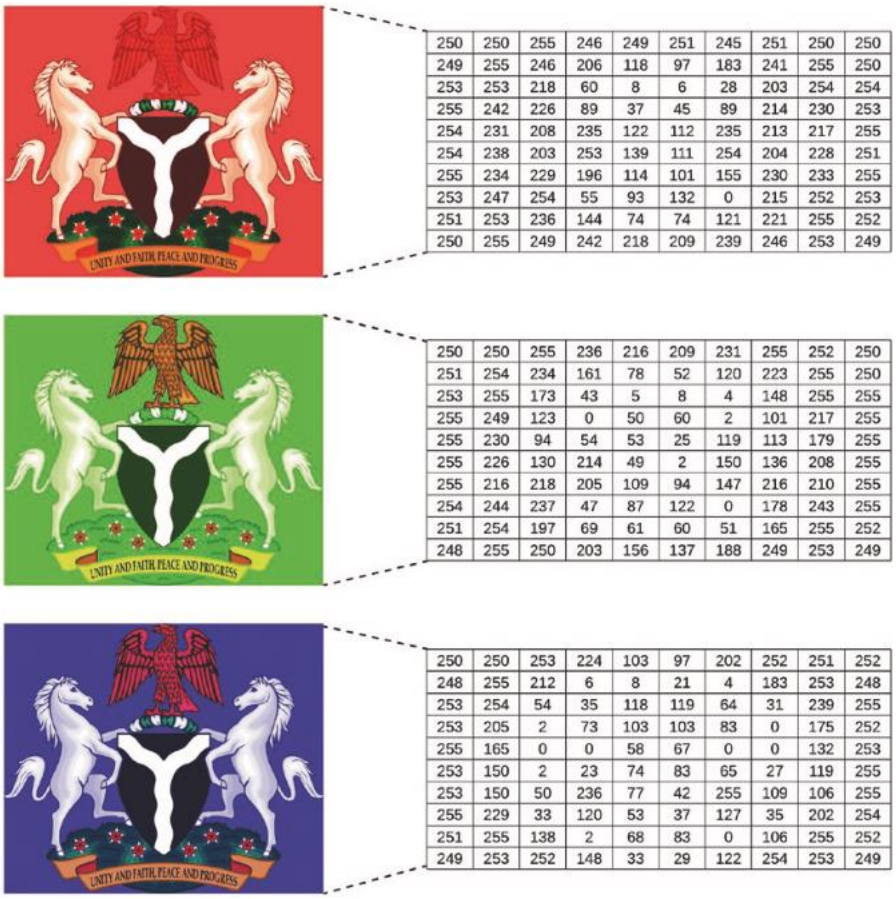
Εικόνα 7: Εφαρμογή φίλτρων για την εμφάνιση των κάθετων γραμμών και οριζόντιων γραμμών. Πάνω είναι οι κάθετες και κάτω είναι οι οριζόντιες. [6]

Όπως γνωρίζουμε μία ασπρόμαυρη εικόνα αναπαρίσταται ως ένα δυσδιάστατο διάνυσμα [ύψος, πλάτος] από τιμές από το 0 έως το 255 και σε περίπτωση που είναι έγχρωμη η εικόνα αυτό το διάνυσμα είναι 3 διαστάσεων τιμών από 0 έως 255 και αναπαρίσταται ως εξής [ύψος, πλάτος, χρωματικό κανάλι] . Η επόμενη εικόνα μας δείχνει ένα παράδειγμα αυτού.



Εικόνα 8: παράδειγμα ασπρόμαυρης εικόνας η οποία αναπαρίσταται ως ένα διάνυσμα τιμών από 0 έως 255. [7]

Και μία έγχρωμη εικόνα είναι ένα τριών διαστάσεων διάνυσμα με κάθε μία διάσταση να αναπαριστά το ύψος, το πλάτος και τέλος τις τιμές του χρωματικού καναλιού. Η επόμενη εικόνα 9 μας δίνει μία αναπαράσταση και αυτού. [7]

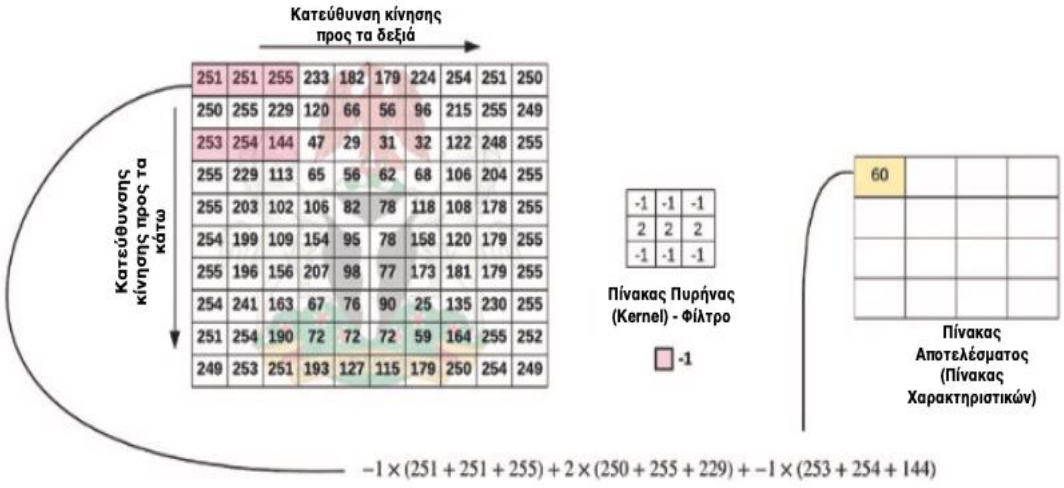


Εικόνα 9: Αναπαράσταση του διανύσματος κάθε χρωματικού καναλιού. [7]

Πάμε λοιπόν να εξηγήσουμε τα μαθηματικά για την εμφάνιση αυτών των χαρακτηριστικών των εικόνων με τα λεγόμενα φίλτρα των συνελικτικών νευρωνικών

δικτύων. Παίρνοντας τις τιμές της εικόνας που έχουμε σε ένα δυσδιάστατο διάνυσμα, πολλαπλασιάζουμε ένα μέρος του πίνακα αυτού, που το ορίζουμε εμείς το μέγεθος, με έναν πίνακα λεγόμενο πυρήνα (**kernel**). Αυτός ο πυρήνας είναι υπεύθυνος για την «ανακάλυψη» αυτών των χαρακτηριστικών που μιλήσαμε πριν, όπως οριζόντιες ευθείες ή κάθετες και πάει λέγοντας. [6]

Αφού γίνει ο εσωτερικός πολλαπλασιασμός πινάκων για κάθε συνδυασμό που μπορεί να δημιουργηθεί με κατεύθυνση από τα αριστερά προς τα δεξιά και μετά προς τα κάτω, τότε βγάζουμε ένα διάνυσμα ο οποίος είναι πλέον πολύ μικρότερος του αρχικού διανύσματος της εικόνας, και. ταυτόχρονα είναι παραμετροποιημένος έτσι που αναδεικνύει το χαρακτηριστικό που επιλέξαμε (οριζόντιες γραμμές, κάθετες κ.τ.λ.). Η εικόνα 10 θα μας δώσει μία ιδέα του τι ακριβώς γίνεται γραφικά. [6]



Εικόνα 10: Παράδειγμα εφαρμογής ενός πολλαπλασιασμού πίνακα πυρήνα με ένα 3x3 πίνακα του διανύσματος μίας εικόνας για την εφαρμογή του φίλτρου στην εικόνα. [7]

Αυτό ο πολλαπλασιασμός γίνεται για κάθε 3x3 πίνακα υπό-πίνακα του διανύσματος που εκπροσωπεί την εικόνα. Επομένως το αποτέλεσμα όπως θα δούμε είναι ένας πολύ μικρός σε μέγεθος πίνακας σε σχέση με αυτού του αρχικού πίνακα της εικόνας. Με αυτό τον τρόπο εξασφαλίζουμε πως τα Συνελικτικά δίκτυα (**convolutional networks**), που περιέχουν αυτού του είδους τα φίλτρα στα κρυφά επίπεδα, δεν θα αυξήσουν την υπολογιστική ισχύ όπως θα έκαναν τα απλά νευρωνικά δίκτυα με όλες τους τις συνδέσεις με κάθε στρώμα από νευρώνων. Καλό είναι να αναφέρουμε πως υπάρχουν και υπερ-παράμετροι που εξασφαλίζουν το μέγεθος του υπό-πίνακα που πολλαπλασιάζεται με τον πίνακα πυρήνα, ή τον ρυθμό με τον

οποίο μεταβαίνει στον επόμενο συνδυασμό και όλα αυτά αλλάζουν και ρυθμίζουν τα φίλτρα με σκοπό την βελτιστοποίηση των επιδόσεων των συνελικτικών νευρωνικών δικτύων. [6] [7]

2.3 Αρχές Μάθησης

Οι υπολογιστικές μέθοδοι της Επιστήμης δεδομένων και κατά συνέπεια η μηχανική μάθηση, είναι κατά κύριο λόγο, χωρισμένη σε 3 υποκατηγορίες με βάση την προσέγγιση του προβλήματος και την μάθηση των μοντέλων μηχανικής μάθησης. Οι 3 επικρατέστερες κατηγορίες μάθησης είναι η μάθηση με επίβλεψη (**Supervised Learning**), μάθηση χωρίς επίβλεψη (**Unsupervised Learning**) και τα σχήματα ενίσχυσης μάθησης (**reinforcement Learning Schemes**). [7]

Σε αυτή την εργασία θα αναφέρουμε απλά αναφορικά τις διαφορές της μάθησης υπό επίβλεψη και αυτής της μάθησης χωρίς επίβλεψη. Έπειτα θα συνεχίσουμε με τα σύνολα δεδομένων και τα προβλήματα παλινδρόμησης και ταξινόμησης, έτσι ώστε να καταλάβουμε σε τι κατηγορία εμπίπτει η μηχανική όραση και το πρόβλημα της «αναγνώρισης» αντικειμένων σε μία εικόνα.

2.3.1 Γνωριμία με τα Σύνολα Δεδομένων

Αφού μάθαμε την γενική συμπεριφορά ενός βιολογικού νευρώνα και πως μπορούμε με μαθηματικό τρόπο να δημιουργήσουμε δικά μας τεχνητά δίκτυα, είναι καιρός να εκπαιδεύσουμε ένα τέτοιο μοντέλο για την καλύτερη κατανόηση του τρόπου λειτουργίας του. Όμως πριν εκπαιδεύσουμε ένα τέτοιο μοντέλο είναι αρκετά σημαντικό να υπάρχει κατανόηση στα δεδομένα που εισάγουμε, στην μορφή που βρίσκονται ακόμα και την κατηγορία προβλήματος στην οποία βρισκόμαστε (υπό επίβλεψη ή χωρίς).

Στην Επιστήμη Δεδομένων για να έχουμε μία πετυχημένη εκπαίδευση ενός μοντέλου μηχανικής μάθησης, μετατρέπουμε τα δεδομένα που έχουμε αντλήσει, σε μορφή πινάκων και πιο συγκεκριμένα σε ένα σύνολο δεδομένων. Στην παρακάτω εικόνα 5 αναπαριστούμε δεδομένα 5 ηλεκτρονικών αλληλογραφιών που κρίθηκαν ανεπιθύμητα ή όχι.

Κάθε γραμμή του πίνακα είναι μία ένδειξη ή αλλιώς κάθε ένα ηλεκτρονικό μήνυμα (email) που έχει καταγραφεί προηγουμένως, ονομάζεται επίσης και Δείγμα (**Sample**). Το πεδίο Χαρακτηριστικών (**Features**) είναι τα χαρακτηριστικά που είχαν τα συγκεκριμένα email (δηλαδή πόσες φορές περιλάμβαναν την λέξη Άρθρο, ML, κ.λ.π.) και οι τιμές στόχων (**Target Values**) είναι η κατηγορία που εντάχθηκε κάθε ηλεκτρονικό μήνυμα δηλαδή αν ήταν ανεπιθύμητη ή όχι το κάθε δείγμα. [7]

	$$$	Άρθρο (article)	ML	Δικαιούχος (Beneficiary)	Στόχος (Target)
email1	5	1	0	6	Ανεπιθύμητη (Spam)
email3	0	4	7	0	Επιθυμητή
email3	0	3	5	0	Επιθυμητή
email4	4	0	0	4	Ανεπιθύμητη (Spam)
email5	8	1	0	5	Ανεπιθύμητη (Spam)

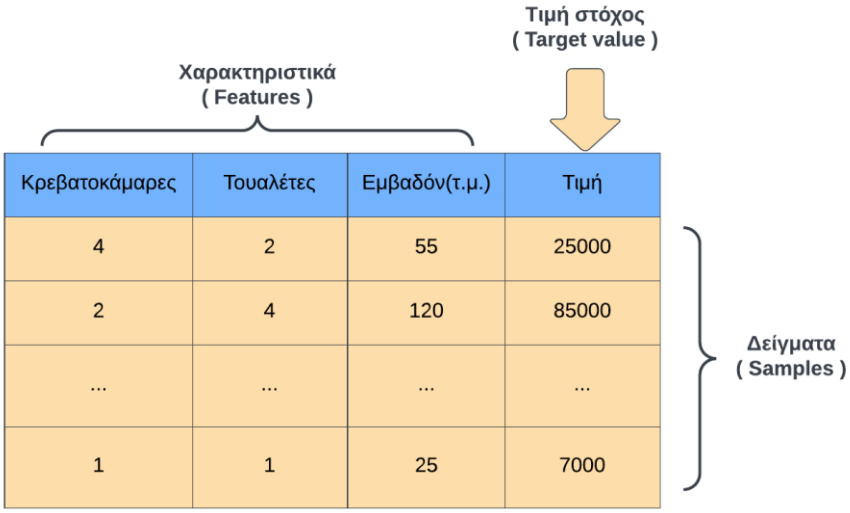
Εικόνα 11: Πίνακας αναπαράστασης ανεπιθύμητων ηλεκτρονικών αλληλογραφιών (Spam Emails) με την ένδειξη Χαρακτηριστικά, Δείγματα και Τιμές Στόχοι.

Η θεμελιώδης ιδέα πάνω στην μηχανική μάθηση είναι πως κάθε δείγμα δεδομένων έχει μία τιμή στόχο (target value). Σκοπός μας λοιπόν είναι να «μάθουμε» στον υπολογιστή να βρίσκει και να μαθαίνει μοτίβα με την βοήθεια των χαρακτηριστικών των δεδομένων που του δίνουμε να εκπαιδευτεί, με σκοπό να μας δίνει μία πιθανότητα πρόβλεψης της τιμής στόχου. Δηλαδή, στο παράδειγμα του συνόλου δεδομένων της εικόνας 5, ο στόχος μας είναι να εκπαιδεύσουμε ένα μοντέλο πάνω σε αυτό το σύνολο δεδομένων έτσι ώστε να προβλέπει καινούργια email αν είναι ανεπιθύμητη ή όχι αλληλογραφία. [7]

2.3.2 Παλινδρόμηση εναντίον Ταξινόμησης

Τον διαχωρισμό των προβλημάτων παλινδρόμησης και ταξινόμησης τον ανακαλύπτουμε με την μορφή του συνόλου δεδομένου που έχουμε και συγκεκριμένα την τιμή της τιμής στόχου. Για παράδειγμα στην εικόνα 6, παρατηρούμε ένα ακόμη παράδειγμα συνόλου δεδομένου που μπορεί να χρησιμοποιηθεί για εκπαίδευση ενός μοντέλου μηχανικής μάθησης για την πρόβλεψη τιμών σπιτιών που έχουν παρόμοια ή όχι χαρακτηριστικά. Είναι σημαντικό να αναφέρουμε πως πλέον η πρόβλεψη του μοντέλου που σκεφτόμαστε να δημιουργήσουμε, είναι μία τιμή η οποία αντικατοπτρίζει το κόστος πώλησης του σπιτιού ενώ στο προηγούμενο σύνολο δεδομένων προέβλεπε αν ένα email ήταν ανεπιθύμητο ή όχι. Δηλαδή, στην πρώτη περίπτωση έχουμε για τιμή στόχο κατηγορίες που θέλουμε να εντάξουμε τα δείγματά μας ενώ στην δεύτερη θέλουμε μία ενδεχόμενη τιμή πώλησης. Εδώ παρουσιάζουμε την διαφορά των

προβλημάτων ταξινόμησης και παλινδρόμησης, με τα προβλήματα ταξινόμησης να είναι αυτά παρόμοια με το σύνολο της εικόνας 5, και ταξινομούν τα δείγματα σε κατηγορίες. Σε αντίθεση τα προβλήματα παλινδρόμησης στοχεύουν και έχουν την τιμή στόχος (**target value**) σαν ένα πραγματικό αριθμητικό δεδομένο. [7]



Εικόνα 12: Σύνολο δεδομένων σπιτιών με τα χαρακτηριστικά τους και τις τιμές πώλησής τους για τιμή στόχο. Παράδειγμα ενός συνόλου προβλήματος παλινδρόμησης

2.3.3 Μάθηση χωρίς και υπό επίβλεψη

Στην Επιστήμη Δεδομένων και στις Μαθηματικές μεθόδους της, δηλαδή στην μηχανική μάθηση και κατά συνέπεια στα νευρωνικά δίκτυα έχουμε δύο κύριες κατηγορίες μοντέλων εκπαίδευσης, αυτή της **μάθησης υπό επίβλεψη (supervised learning)** και αυτή της **μάθησης χωρίς επίβλεψη (unsupervised learning)**. Θα αναφέρουμε λίγα και εισαγωγικά κομμάτια της μηχανικής μάθησης χωρίς επίβλεψη. Ο σκοπός της μάθησης χωρίς επίβλεψη είναι να δημιουργήσει ένα μοντέλο μάθησης που καταγράφει την υποκείμενη κατανομή του συνόλου δεδομένων. Το σύνολο δεδομένων δεν έχει μεταβλητές στόχους (target values), σε αντίθεση με την μάθηση υπό επίβλεψη, και η έλλειψη μεταβλητής στόχος την καθιστά αδύνατον να υπάρξει μία συνάρτηση που δημιουργεί σχέσεις των δεδομένων με την μεταβλητή στόχος, κάτι το οποίο συμβαίνει στην εκπαίδευση των νευρωνικών δικτύων. Αντιθέτως, χρησιμοποιεί αλγόριθμους ομαδοποίησης (**Clustering**) που ομαδοποιεί «παρόμοια» δεδομένα μεταξύ τους, που είναι μία προσπάθεια το μοντέλο να «καταλάβει» τα δεδομένα αντί να προβλέψει τιμές σύμφωνα με αυτά. [7]

Με λίγα λόγια η μάθηση υπό επίβλεψη περιέχει σύνολα δεδομένων που περιέχουν τα δείγματα (**samples**) τα χαρακτηριστικά αυτών (**Features**) και την τιμή στόχος (**target value**) ενώ η μάθηση χωρίς επίβλεψη δεν περιέχει την τιμή στόχο. Συμπεραίνουμε λοιπόν πως τα

σύνολα δεδομένων των εικόνων 5 και 6 ανήκουν και τα δύο στην μάθηση υπό επίβλεψη (supervised learning) άσχετα με το αν είναι προβλήματα **παλινδρόμησης** ή **ταξινόμησης**.

2.4 Γνωριμία με TensorFlow

Παρόλου που θα μπορούσαμε να αφιερώσουμε όλη την πτυχιακή εργασία για τα νευρωνικά δίκτυα και την μηχανική μάθηση σε θεωρητικό επίπεδο, καλό είναι να εκπαιδεύσουμε μόνοι μας κάποια μοντέλα και να δούμε στην πράξη την διαφορά των επιδόσεών τους με τις αλλαγές που θα τους εφαρμόζουμε.

Το κύριο εργαλείο που θα χρησιμοποιήσουμε, πέρα από την γλώσσα Python, είναι **TensorFlow.** Η TensorFlow, είναι μία ανοιχτού κώδικα βιβλιοθήκη της Python, η οποία δημιουργήθηκε από την Google το 2015 για ευκολότερη δημιουργία μοντέλων μηχανικής μάθησης, εκπαίδευσης και εφαρμογής αυτών. Στην αρχή ξεκίνησε ως ένα εργαλείο της εταιρεία της ίδιας και χρησιμοποιούνταν μόνο από τους ίδιους. Παρόλο που υπάρχουν πολλές βιβλιοθήκες και γλώσσες προγραμματισμού που κάνουν επίσης εύκολη την δημιουργία, εκπαίδευση και απόδοση σε πραγματικά δεδομένα μοντέλων μηχανικής μάθησή, επιλέξαμε την TensorFlow διότι σε σχέση με τις άλλες μεθόδους, οι πράξεις πινάκων γίνονται σαφέστατα πιο γρήγορα διότι χρησιμοποιούν την «δύναμη» ή αλλιώς επεξεργαστική ισχύ, μίας κάρτας γραφικών και όχι του επεξεργαστή όπως συνηθίζεται. Επιπλέον το Google Colab που θα είναι και η πλατφόρμα που θα τρέχει ο κώδικας, έχει καλή υποστήριξη με τα Tensors της TensorFlow. Τα Tensors είναι οι αντίστοιχοι «πίνακες» ή αλλιώς διανύσματα που θα υπάρχουν στα μοντέλα μηχανικής μάθησης. [6] [8]

2.4.1 Απλό Πρόγραμμα στην TensorFlow

Ας ξεκινήσουμε εισάγοντας την βιβλιοθήκη tensorflow μέσω της γλώσσας Python.

```
# import tensorflow
import tensorflow as tf

# Quadratic expression: x**2 + 3x - 4 = 0
a = tf.constant(1.0)
b = tf.constant(3.0)
c = tf.constant(-4.0)

print(a)
```

```
print(b)
print(c)

'Output':
tf.Tensor(1.0, shape=(), dtype=float32)
tf.Tensor(3.0, shape=(), dtype=float32)
tf.Tensor(-4.0, shape=(), dtype=float32)
```

Η `tf.constant` είναι μία μέθοδος της βιβλιοθήκης, όπου δημιουργεί ένα διάνυσμα Tensor για αποθήκευση μίας σταθερής τιμής. Στο συγκεκριμένο παράδειγμα δημιουργούμε σταθερές τιμές, επομένως μένει τώρα να υπολογίσουμε τις τετραγωνικές ρίζες της συνάρτησης μας.

```
x1 = (-b + tf.math.sqrt(b**2 - (4*a*c))) / 2**a
x2 = (-b - tf.math.sqrt(b**2 - (4*a*c))) / 2**a

print(x1)
print(x2)
'Output':
tf.Tensor(1.0, shape=(), dtype=float32)
tf.Tensor(-4.0, shape=(), dtype=float32)
```

Επομένως βρήκαμε τις δύο ρίζες της εξίσωσης $x^2 + 3x - 4 = 0$ οι οποίες είναι οι τιμές 1 και -4. Είναι σημαντικό να παρατηρήσουμε πως οι εσωτερικές συναρτήσεις της TensorFlow τρέχουν πρώτα και αμέσως, όπως ακριβώς και στην γλώσσα προγραμματισμού Python. [6]

2.5 Δημιουργία Μοντέλου Μηχανικής Μάθησης - ΒΗΜΑΤΑ

Η δημιουργία ενός μοντέλου μηχανικής μάθησης και κατά συνέπεια μοντέλου νευρωνικού δικτύου, απαιτεί τον προγραμματισμό του. Αποφασίσαμε να ασχοληθούμε με την βιβλιοθήκη της Python, TensorFlow και είναι καιρός να φτιάξουμε ένα μοντέλο μέσο αυτών των εργαλείων. Πρώτα όμως να δώσουμε κάποια βήματα που πρέπει να ακολουθήσουμε για την επιτυχημένη δημιουργία του μοντέλου από προγραμματιστικής απόψεως.

Βήμα 1: Δημιουργία της Αρχιτεκτονικής.

Το πρώτο βήμα στον προγραμματισμό ενός μοντέλου μηχανικής μάθησης, είναι να το συντάξουμε με τα κατάλληλα επίπεδα όπως το επίπεδο εισόδου (**Input Layer**) που θα αποτελείται από αριθμό νευρώνων που εμείς θα καθορίσουμε, τα κρυφά επίπεδα (**Hidden Layers**), επίσης με αριθμό νευρώνων που εμείς καθορίζουμε, που όπως έχουμε δει είναι και το σημαντικότερο μέρος ενός νευρωνικού δικτύου και το επίπεδο εξόδου (**Output Layer**) το οποίο συνηθίζεται να είναι έτσι διαμορφωμένο με τόσες τιμές εξόδου όσες είναι και οι επιλογές που δίνουμε στο μοντέλο να μας απαντήσει στο συγκεκριμένο πρόβλημα. [6] [7]

Ο τρόπος που θα γίνει αυτό θα είναι με την βοήθεια της **Keras** που είναι βασισμένη στον τρόπο λειτουργίας της TensorFlow αλλά ασχολείται κυρίως με τα νευρωνικά δίκτυα. Μέσω αυτής θα δημιουργήσουμε τα επίπεδα, τον αριθμό νευρώνων και γενικότερα την αρχιτεκτονική του μοντέλου μας.

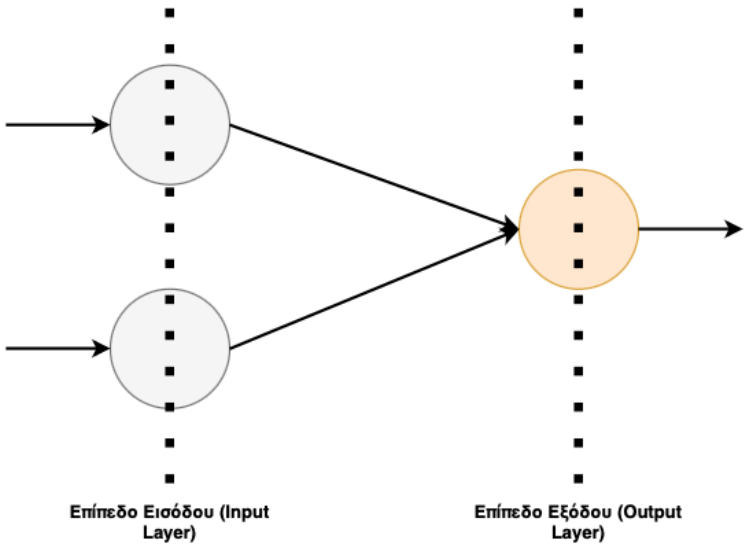
Πρώτα κάνουμε τις κατάλληλες εισαγωγές.

```
# Import TensorFlow
import tensorflow as tf
```

Έπειτα εισάγουμε την μέθοδο `.Sequential` της βιβλιοθήκης **Keras** όπου θα εισάγουμε ένα διάνυσμα από τα επίπεδα που θα αποτελείται το μοντέλο μας. Εμείς δημιουργούμε δύο επίπεδα, το πρώτο το οποίο θα αποτελείται από 2 νευρώνες και θα έχει τον ρόλο του επιπέδου εισόδου με δύο νευρώνες και τέλος έχουμε ένα ακόμα επίπεδο με έναν νευρώνα ο οποίος έχει τον ρόλο του επιπέδου εξόδου με έναν νευρώνα. Δίνεται ο κώδικας:

```
# Creation of model architecture
model = tf.keras.Sequential([
    tf.keras.layers.Dense(2),
    tf.keras.layers.Dense(1)
])
```

Αυτό γραφικά μπορούμε ένα το αναπαραστήσουμε όπως στην επόμενη εικόνα 13:



Εικόνα 13: Γραφική αναπαράσταση του μοντέλου που προγραμματίσαμε, με δύο επίπεδα. Το πρώτο επίπεδο εισόδου με δύο νευρώνες και το δεύτερο με έναν νευρώνα εξόδου.

Προφανώς προσθέτοντας και άλλα επίπεδα στην μέθοδο της Keras `.Sequential` μπορούμε να δημιουργήσουμε και άλλα επίπεδα, τα οποία θα έχουν και τον αριθμό των νευρώνων που επιθυμούμε. Αφού δημιουργήσουμε την αρχιτεκτονική του νευρωνικού δικτύου, επόμενο είναι να το ρυθμίσουμε με παραμέτρους που έχει η βιβλιοθήκη **Keras** διαθέσιμες.

Βήμα 2: Ρύθμιση παραμέτρων του μοντέλου

Δυο αρχιτεκτονικά δίκτυα που έχουν την ίδια αρχιτεκτονική δεν σημαίνει ότι είναι και ίδια δίκτυα, αυτό διότι υπάρχουν και συγκεκριμένες ιδιαιτερότητες που δεν έχουμε λάβει ακόμα υπόψιν και έχουν σημαντική επίπτωση στην εκμάθηση και τον τρόπο εκμάθησης του νευρωνικού δικτύου.

Αυτές μπορεί να είναι παράμετροι που ρυθμίζουν τον τρόπο ή την ταχύτητα εκμάθησης του νευρωνικού δικτύου, μπορεί να είναι η ρύθμιση ή αλλαγή της συνάρτησης $f(x)$ με την οποία έχουμε έρθει προηγουμένως σε επαφή. Να τονίσουμε πως η $f(x)$ των νευρωνικών δικτύων λέγεται και συνάρτηση ενεργοποίησης (**activation function**). Και άλλες παράμετροι οι οποίες η Keras έχει φροντίσει να συμπεριλάβει και να ορίσει μία βασική τιμή στην μέθοδο `.compile`.

Δίνουμε τον κώδικα:

```
# Compiling the model
model.compile(loss=tf.keras.losses.mae,
              optimizer=tf.keras.optimizers.Adam(),
```

```
        metrics=['mae'])
```

Παρατηρούμε πως στην μέθοδο `.compile` έχουμε αλλάξει το **loss** το οποίο ορίζεται ως η μετρική η οποία θα υπολογίζεται σε κάθε επανάληψη και θα προσπαθεί το μοντέλο να μειώσει. Για παράδειγμα εμείς θέσαμε να είναι το μέσο απόλυτο σφάλμα (**mean absolute error**). [6] [8] [7]

Επίσης ρυθμίσαμε την παράμετρο **optimizer** η οποία είναι υπεύθυνη για τον τρόπο με τον οποίο το νευρωνικό δίκτυο θα διορθώνει τα βάρη του έτσι ώστε να ελαχιστοποιήσει την μετρική loss που ορίσαμε πιο πάνω. Συνήθως δοκιμάζεται η σταδιακή κάθοδος (**gradient decent**) αλλά υπάρχουν και άλλες όπως η τεχνική Adam η οποία αλλάζει τα βάρη με «ρυθμό» δηλαδή σταδιακά. Παρόλα αυτά εμείς θα ασχοληθούμε κυρίως με Adam. [6] [8] [7]

Και τέλος η τελευταία παράμετρος που ορίσαμε είναι η metrics η οποία μας δίνει στο αποτέλεσμα κάθε επανάληψης και άλλες μετρικές που ίσως θέλουμε να υπολογίσουμε, συνήθως επιλέγεται η ίδια με αυτή της loss παράμετρο. Φυσικά αφού έχουμε δημιουργήσει το μοντέλο μας, και το έχουμε ρυθμίσει με τις κατάλληλες παραμέτρους, είναι καιρός και να το εκπαιδεύσουμε, το επόμενο βήμα είναι η εκπαίδευση του νευρωνικού δικτύου.

Βήμα 3: Εκπαίδευση του Μοντέλου

Έχοντας υλοποιήσει όλα τα παραπάνω, και εφόσον έχουμε ετοιμάσει τα δεδομένα μας είναι καιρός να εκπαιδεύσουμε το μοντέλο μας πάνω σε αυτά. Στην εκπαίδευση του νευρωνικού δικτύου υπάρχουν και εδώ κάποιες παράμετροι που πρέπει να δώσουμε, έτσι ώστε να ξεκινήσει η εκπαίδευση του. Αυτές είναι, τα δεδομένα εκπαίδευσης, τα δεδομένα ταυτοποίησης δηλαδή στα προβλήματα ταξινόμησης αυτές είναι οι ταμπέλες/κατηγορίες των δεδομένων εκπαίδευσης και τις επαναλήψεις που θέλουμε να κάνει το νευρωνικό δίκτυο πάνω σε όλα τα δεδομένα μας για να εκπαιδευτεί πάνω σε αυτά.

Αρχικοποίηση δεδομένων

Θα δημιουργήσουμε για χάρη παραδείγματος τα δεδομένα μας τα οποία θα ξέρουμε πως ακολουθούν την συνάρτηση y = x +2 δηλαδή θα είναι γραμμικής μορφής. Θα προσπαθήσουμε να αναπαραστήσουμε τα δεδομένα μας γραφικά πριν τα εκπαιδεύσουμε για να έχουμε μία αντίληψη και έπειτα θα προχωρήσουμε στην εκπαίδευση του μοντέλου μας.

Εισάγουμε την TensorFlow και δημιουργούμε τα σημεία του x άξονα.

```
# Create sample dataset
X = tf.range(-100, 100, 4)
X
```

```
'Output':
<tf.Tensor: shape=(50,), dtype=int32, numpy= array([-100, -96, -92, -
88, -84, -80, -76, -72, -68, -64, -60, -56, -52, -48, -44, -40, -36, -
32, -28, -24, -20, -16, -12, -8, -4, 0, 4, 8, 12, 16, 20, 24, 28, 32,
36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96],
dtype=int32)>
```

Και τώρα θα δημιουργήσουμε τα σημεία του y άξονα.

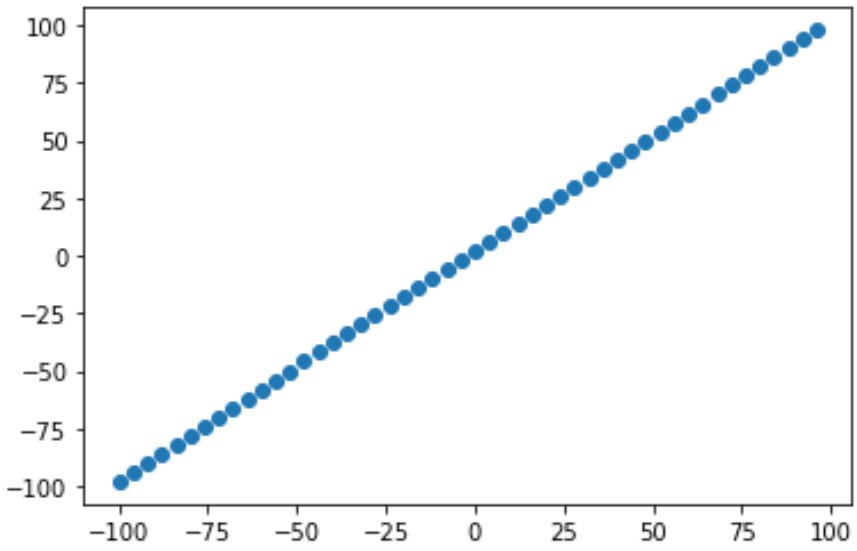
```
# Create the y data
y = X + 2
y
```

```
'Output':
<tf.Tensor: shape=(50,), dtype=int32, numpy= array([-98, -94, -90, -86,
-82, -78, -74, -70, -66, -62, -58, -54, -50, -46, -42, -38, -34, -30, -
26, -22, -18, -14, -10, -6, -2, 2, 6, 10, 14, 18, 22, 26, 30, 34, 38,
42, 46, 50, 54, 58, 62, 66, 70, 74, 78, 82, 86, 90, 94, 98],
dtype=int32)>
```

Και τώρα θα τα αναπαραστήσουμε γραφικά μέσο της βιβλιοθήκης matplotlib.

```
# Visualize the data
import matplotlib.pyplot as plt
plt.scatter(X,y);
```

Το οποίο μας δίνει το γράφημα:



Εικόνα 14: γραφικά αναπαράσταση των δεδομένων που δημιουργήσαμε στο δυσδιάστατο χώρο.

Αυτά τα δεδομένα θα προσπαθήσουμε να εκπαιδεύσουμε το νευρωνικό μας δίκτυ έτσι ώστε για τιμές x που δεν έχει ξαναδεί να μπορέσει να κάνει μία πρόβλεψη που πλησιάζει την συνάρτηση y = x +2.

Δίνουμε τον κώδικα για την εκπαίδευση του δικτύου:

```
# Fit the model
model.fit(tf.expand_dims(X, axis=-1), y, epochs=5)
```

```
'Output':
Epoch 1/5
2/2 [==============================] - 1s 18ms/step - loss: 74.9092 -
mae: 74.9092
Epoch 2/5
2/2 [==============================] - 0s 11ms/step - loss: 33.5844 -
mae: 33.5844
Epoch 3/5
2/2 [==============================] - 0s 9ms/step - loss: 18.6887 -
mae: 18.6887
```

```
Epoch 4/5
2/2 [==============================] - 0s 7ms/step - loss: 21.3022 -
mae: 21.3022
Epoch 5/5
2/2 [==============================] - 0s 7ms/step - loss: 16.1774 -
mae: 16.1774
<keras.callbacks.History at 0x7f332c689a00>
```

Παρατηρούμε πως δώσαμε για εκπαίδευση τα δεδομένα μας X και y, αλλά στα δεδομένα της μεταβλητής X αναγκαστήκαμε να αλλάξουμε τις διαστάσεις διότι έτσι όπως τις φτιάξαμε ήταν θεωρημένο χωρίς διαστάσεις και αυτό δημιουργούσε πρόβλημα στην εκπαίδευση του μοντέλου μας. Επίσης δώσαμε και μία τιμή 5 στην μεταβλητή epochs, αυτό σημαίνει πως το νευρωνικό μας δίκτυο εκπαιδεύτηκε 5 φορές πάνω στα δεδομένα μας, με την ελπίδα να τα «μάθει» καλύτερα. Τέλος βλέπουμε πως στην έξοδο παρουσιάζεται το **loss** και το **mae** το οποίο και στις δύο περιπτώσεις είναι το μέσο απόλυτο σφάλμα, όπως το ορίσαμε και πως μειώνεται σε κάθε μας επανάληψη.

Αυτή ήταν μία εκπαίδευση ενός μοντέλου νευρωνικού δικτύου, αναφέρουμε πως τα επίπεδα στην μηχανική όραση δεν είναι `.dense` αλλά όπως θα δούμε αλλάζουν και γίνονται της μορφής επίπεδα σύντηξης (**convolutional layers**) διότι όπως εξηγήσαμε είναι πιο παραγωγικά από τα απλά νευρωνικά δίκτυα, όπως αυτό που φτιάξαμε τώρα.

Στο επόμενο κεφάλαιο θα ασχοληθούμε με τον τρόπο που ξεχωρίζουμε τις επιδόσεις των νευρωνικών δικτύων, πως αλλάζουν ανάλογα με την κατηγορία προβλήματος που ανήκουμε καθώς και την προετοιμασία των δεδομένων. Και στο τέλος θα μιλήσουμε για τις απαραίτητες διεργασίες στην προετοιμασία των δεδομένων αφού πειραματικά συγκρίνουμε μοντέλα τις οποίες τις ακολουθούν με αυτές που δεν τις ακολουθούν.

3. Βασικές αρχές

Για να μπορέσουμε να ορίσουμε κάποιες βασικές αρχές, γενικότερα, θα πρέπει να ακολουθήσουμε μία τεχνική που θα μας εξασφαλίζει την μία προσέγγιση καλύτερη της άλλης. Ένας από τους τρόπους να συγκρίνουμε τεχνικές είναι σύμφωνα με τα αποτελέσματα που μας φέρνει κάθε μία. Επομένως, και στην επιστήμη δεδομένων, χρειαζόμαστε μία μορφή σύγκρισης «τεχνικών» ή στην συγκεκριμένη περίπτωση, σύγκριση μοντέλων μάθησης. Ο καταλληλότερος τρόπος να επιτευχθεί αυτό είναι με την δημιουργία μετρικών που υπολογίζουν «πόσο καλά» ένα μοντέλο έχει προβλέψει στοιχεία δεδομένων που δεν έχει εκπαιδευτεί. [7]

3.1 Σύνολα εκπαίδευσης, αξιολόγησης, επικύρωσης

Στην επιστήμη δεδομένων και συγκεκριμένα στις μαθηματικές μεθόδους της, συναντάμε πολλές φορές των διαχωρισμό των δεδομένων σε 3 σύνολα. Αυτά τα 3 σύνολα είναι τα σύνολα εκπαίδευσης, αξιολόγησης και επικύρωσης. Κάθε ένα από αυτά τα σύνολα έχει τον δικό του σκοπό στην εκπαίδευση των μοντέλων μάθησης.

Σύνολο Εκπαίδευσης (Train Data):

Όπως το ορίζει και το όνομα τους είναι το σύνολο το οποίο δίνεται στα μοντέλα μάθησης και κατά συνέπεια στα νευρωνικά δίκτυα, έτσι ώστε να εκπαιδευτούν πάνω σε αυτά. Με τον όρο εκπαίδευση, εννοούμε προφανώς την εκμάθηση των μοτίβων των δεδομένων από τα μοντέλα, για την εκμάθηση περίπλοκων διεργασιών, όπως αυτή της όρασης. [7]

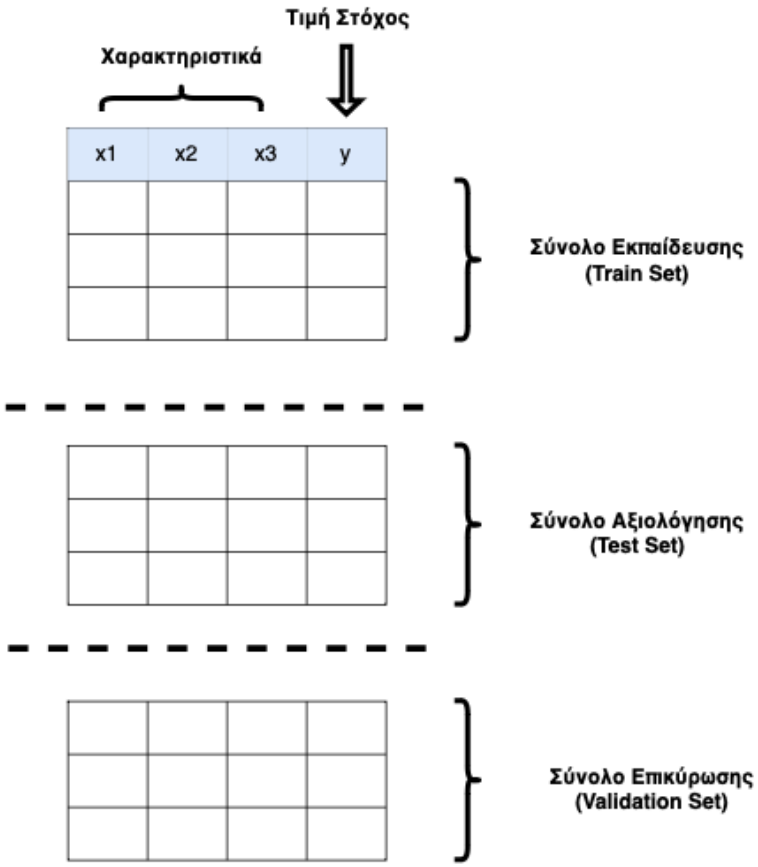
Σύνολο Αξιολόγησης (Test Data):

Έχοντας πλέον εκπαιδεύσει το μοντέλο μας, με το σύνολο εκπαίδευσης, είναι καιρός να δούμε πως θα συμπεριφερθεί σε δεδομένα που δεν έχει ξαναδεί, όπως για παράδειγμα μία καινούργια εικόνα. Αυτό το σύνολο που περιέχει δεδομένα μη γνωστά για το μοντέλο μας, θεωρείται το σύνολο αξιολόγησης. [7]

Σύνολο Επικύρωσης (Validation Data):

Κατά την διάρκεια της εκπαίδευσης, ο αλγόριθμος χρειάζεται έναν «οδηγό» για την αλλαγή των βαρών του αν πρόκειται για νευρωνικό δίκτυο, ή τις παραμετροποιήσεις του αν πρόκειται για ένα μοντέλο μάθησης γενικότερα. Σε αυτό το στάδιο το μοντέλο μαθαίνει αν τα πηγαίνει καλά με το σύνολο Επικύρωσης, στην ουσία μοιάζει με το σύνολο αξιολόγησης, δηλαδή δεν έχει εκπαιδευτεί πάνω σε αυτό αλλά χρησιμοποιείται κατά την διάρκεια της εκπαίδευσής του και όχι μετά όπως στο σύνολο αξιολόγησης. [7]

Μία σύνηθες στρατηγική είναι να χωρίζουμε όλα τα δεδομένα μας σε ποσοστά του συνολικού. Στο σύνολο εκπαίδευσης (Train Set) χρησιμοποιούμε περίπου το 60% , 20% σε σύνολο επικύρωσης (Validation Set) και 20% σε σύνολο αξιολόγησης (Test Set). Αυτή η στρατηγική είναι γνωστή ως ο κανόνας των 60/20/20. Παρόλα αυτά υπάρχουν πολλές διαφορετικές μέθοδοι να χωρίσουμε το αρχικό σύνολο μας, πολλές από τις οποίες δεν υπάρχει το σύνολο επικύρωσης αλλά μόνο το σύνολο εκπαίδευσης και αξιολόγησης.



Εικόνα 15: Σύνολο Δεδομένων το οποίο έχει διαχωριστεί σε σύνολο εκπαίδευσης, σύνολο αξιολόγησης και σύνολο επικύρωσης.

Έχοντας πλέον αναλύσει τον διαχωρισμό των συνόλων, που συνηθίζεται, στην μηχανική μάθηση ή την μοντελοποίηση γενικότερα, είναι εύλογο να προχωρήσουμε σε έννοιες όπως την αξιολόγηση των μοντέλων. Παρακάτω θα αναλύσουμε πως κάνουμε μία αξιολόγηση μοντέλων, πως συνδέεται με τον τύπο προβλημάτων που βρισκόμαστε και θα δώσουμε τον ορισμό μερικών μετρικών που υπάρχουν.

3.2 Αξιολόγηση Μοντέλων

Πως ξέρουμε εάν το μοντέλο μας έχει εκπαιδευτεί; Η εκπαίδευσή του είναι η καλύτερη που μπορούμε να δημιουργήσουμε; Υπάρχει τρόπος να συγκρίνουμε δύο μοντέλα ως προς την απόδοσή τους; Την απάντηση σε αυτά τα ερωτήματα θα δώσουμε σε αυτό το κεφάλαιο.

Τα μοντέλα εκπαίδευσης, και κατά συνέπεια και τα νευρωνικά δίκτυα, είναι στην ουσία αλγόριθμοι. Δηλαδή, τα δεδομένα μας είναι στην ουσία μια σειρά από εντολές που σε συνεργασία με μαθηματικά μας δίνουν και μας οδηγούν στην λύση του αποτελέσματος, επομένως μπορούμε να τα αξιολογήσουμε και κατά συνέπεια να μετρήσουμε πόσο καλά αποδίδουν με δημιουργία άλλων μαθηματικών λεγόμενων μετρικών.

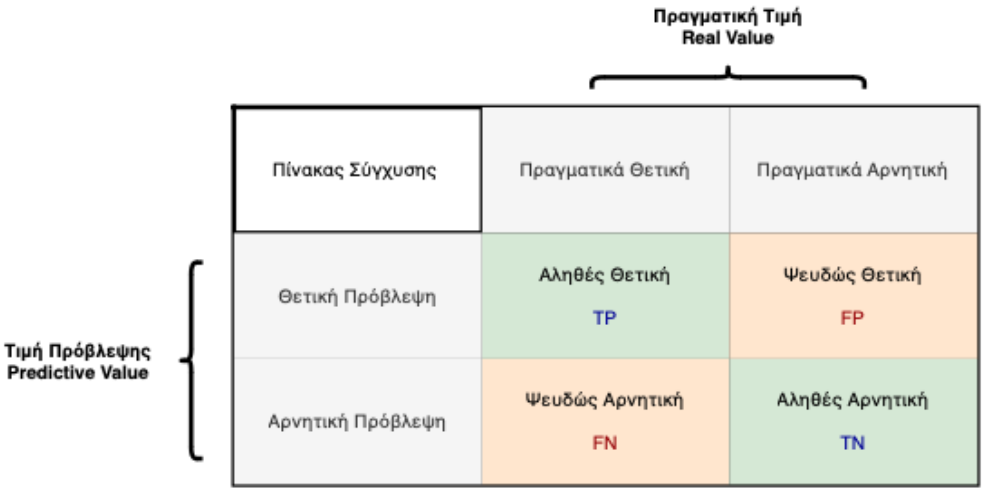
Η απόδοση του μοντέλου στα δεδομένα εκπαίδευσης αξιολογείται για να ληφθεί η ακρίβεια του συνόλου εκπαίδευσης λεγόμενο ως **Precision,** ενώ η απόδοσή στα δεδομένα αξιολόγησης (**Test Set**) γίνεται για να ληφθεί η ακρίβεια των δεδομένων αξιολόγησης όταν το μοντέλο προβλέπει τους στόχους των δειγμάτων που δεν έχει έρθει προηγουμένως σε επαφή. Η αξιολόγηση των μοντέλων σε δεδομένα που δεν έχει εκπαιδευτεί δηλαδή σε δεδομένα αξιολόγησης (**Test Set**), είναι το πραγματικό μέτρο απόδοσης των μοντέλων. [7]

Είναι σημαντικό να αναφέρουμε πως η μέτρηση της απόδοσης του κάθε μοντέλου μας, εξαρτάτε άμεσα από τον τύπο προβλήματος που αντιμετωπίζουμε καθώς διαφορετικές μετρικές υπάρχουν για τα προβλήματα ταξινόμησης και διαφορετικές για τα προβλήματα παλινδρόμησης. Θα δώσουμε επιγραμματικά μερικές μετρικές και την αντιστοιχία τους ανά κατηγορία προβλημάτων. [7]

3.2.1 Μετρικές Αξιολόγησης στα Προβλήματα Ταξινόμησης

3.2.1.1 Πίνακας Σύγχυσης:

Μία από τις πιο γνωστές τακτικές στα προβλήματα ταξινόμησης είναι εκείνες που λαμβάνουν υπόψιν τον πίνακας σύγχυσης (**Confusion Matrix)**. Ο πίνακας σύγχυσης είναι ένα εργαλείο μέτρησης της απόδοσης ενός μοντέλου μηχανικής μάθησης υπό επίβλεψη. Παρουσιάζεται ως ένας πίνακας με κελιά που μοιάζουν με πλέγμα. Σε περίπτωση ενός προβλήματος ταξινόμησης με δύο κλάσεις, οι στήλες του πίνακα σύγχυσης δίνουν την πραγματική τιμή της κλάσης είτε αρνητική είτε θετική, ενώ οι γραμμές του πίνακα είναι οι προβλεπόμενες τιμές του μοντέλου ταξινόμησης αυτών των δύο κλάσεων. Κάθε γραμμή του πίνακα αναφέρεται σε ψευδή ή αληθή προβλεπόμενη τιμή, καλύτερη κατανόηση στην επόμενη εικόνα 16. [7]



*Εικόνα 16: Πίνακας Σύγχυσης όπου οι γραμμές του είναι οι τιμές πρόβλεψης του μοντέλου μας και οι στήλες αναπαριστούν τις πραγματικές τιμές στόχος (**Target Value**) των δεδομένων.*

Θα αναλύσουμε κάθε κελί του πίνακα σύγχυσης ως εξής:

- **Αληθώς Θετικές / TP:** Ο αριθμός των δειγμάτων που το μοντέλο πρόβλεψε την μεταβλητή στόχο ως θετική και είναι πράγματι θετική τιμή.
- **Ψευδώς Θετικές / FP:** Ο αριθμός των δειγμάτων που το μοντέλο πρόβλεψε την μεταβλητή στόχο ως θετική αλλά στην πραγματικότητα είναι αρνητική τιμή.
- **Αληθώς Αρνητικές / TN:** Ο αριθμός των δειγμάτων που το μοντέλο πρόβλεψε την μεταβλητή στόχο ως αρνητική και είναι πράγματι αρνητική τιμή.
- **Ψευδώς Αρνητικές / FN :** Ο αριθμός των δειγμάτων που το μοντέλο πρόβλεψε την μεταβλητή στόχο ως αρνητική αλλά στην πραγματικότητα ήταν θετική.

Από τον πίνακα σύγχυσης μπορούμε να εισάγουμε κάποιες μετρικές οι οποίες μας δίνουν μία μαθηματική έννοια του πόσο καλά πηγαίνει το μοντέλο μας. [7]

Αυτές είναι:

- **Ορθότητα / Accuracy:** Η ορθότητα είναι ένα κλάσμα που σαν αριθμητή έχει το σύνολο των προβλέψεων που ο αλγόριθμος/μοντέλο πρόβλεψε σωστά, δηλαδή τις τιμές αληθώς θετικές και αληθώς αρνητικές, με παρονομαστή να έχει το συνολικό πλήθος των προβλέψεων. Δηλαδή:

$$\boldsymbol{accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{FP} + \boldsymbol{FN} + \boldsymbol{TN}}$$

- **Ακρίβεια / Precision:** Η ακρίβεια ή αλλιώς η θετική προγνωστική τιμή εκφράζει το κλάσμα που σαν αριθμητή έχει τις αληθώς θετικές τιμές TP που προέβλεψε το

μοντέλο και παρονομαστή έχει το άθροισμα των αληθώς θετικών τιμών TP και των ψευδών θετικών τιμών FP. Με άλλα λόγια η ακρίβεια είναι το κλάσμα των αποτελεσμάτων που ο αλγόριθμος έδωσε θετική πρόβλεψη προς το άθροισμα των θετικών προβλέψεων ανεξαρτήτως αν ήταν σωστή ή όχι η πρόβλεψη. Η εξίσωση είναι:

$$\boldsymbol{precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}}$$

- **Ανάκληση / Ευαισθησία / Recall:**

  Η ανάκληση ή αλλιώς ευαισθησία είναι ξανά ένα κλάσμα όπως και οι προηγούμενες δύο μετρικές που εισάγαγε, αλλά αυτή την φορά με αριθμητή την τιμή των αληθώς θετικών τιμών TP, αλλά για παρονομαστή έχουμε το άθροισμα των αληθώς θετικών τιμών με των ψευδών αρνητικών τιμών. Έτσι καταλαβαίνουμε ότι για παρονομαστή έχουμε το άθροισμα για όλες τις τιμές του συνόλου δεδομένων με θετική μεταβλητή στόχο, ανεξάρτητα εάν το μοντέλο τις πρόβλεψε σωστά ή όχι. Το άθροισμα του παρονομαστή λέγεται και αλλιώς ως θετική κατάσταση (**Condition Positive**).

$$\boldsymbol{recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}$$

3.2.2 Μετρικές Αξιολόγησης στα προβλήματα Παλινδρόμησης

Στην συνέχεια θα ασχοληθούμε με τους τρόπους μέτρησης ενός μοντέλου μηχανικής μάθησης υπό επίβλεψη στα προβλήματα παλινδρόμησης. θα ξεκινήσουμε να μιλάμε για το μέσο τετραγωνικό σφάλμα θα δώσουμε τον τύπο υπολογισμό του και στην συνέχεια θα αναφερθούμε σε ένα τεχνητό παράδειγμα για τον υπολογισμό του.

3.2.2.1 Ριζικό μέσο τετραγωνικό σφάλμα (RMSE)

Το μέσο τετραγωνικό σφάλμα γνωστό και ως RMSE για συντομία , είναι ένα πολύ σημαντικό εργαλείο αξιολόγησης μοντέλων μηχανικής μάθησης στα προβλήματα παλινδρόμησης. Υπολογίζει το σφάλμα της πραγματικής τιμής του δείγματος με την προβλεπόμενη τιμή του μοντέλου. Μαθηματικά μπορεί να παρουσιαστεί ως το άθροισμα των διαφόρων της πραγματικής τιμής με την προβλεπόμενη τιμή κάθε δείγματος που υπολογίστηκε. [7]

Η εξίσωση είναι:

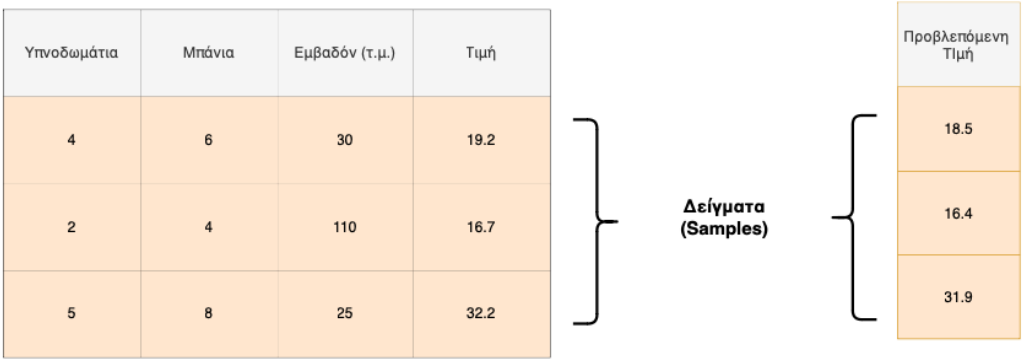
$$\boldsymbol{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\boldsymbol{y_i} - \boldsymbol{u_i})^2}{\boldsymbol{n}}}$$

Όπου:

- n = ο αριθμός των δειγμάτων στο σύνολο δεδομένων μας
- y_i = η πραγματική τιμή του δείγματος i.
- u_i = η προβλεπόμενη τιμή του δείγματος i από το μοντέλο μηχανικής μάθησης.

Έτσι με το μέσο όρο του αθροίσματος των διαφορών των τετραγώνων $(y_i - u_i)^2$ παίρνουμε την θετική απόκλιση των τιμών της πραγματικής με της προβλεπόμενης τιμής η οποία έχει μονάδες του προβλήματος στο τετράγωνο. Με την εισαγωγή του ριζικού όμως παίρνουμε σαν απάντηση την επίδοση του μοντέλου μας με τις σωστές μονάδες του επίσης. Έτσι το RMSE είναι και μία πολύ καλή μέθοδος υπολογισμού απόδοσης των μοντέλων. [7]

Παρακάτω θα δώσουμε ένα παράδειγμα χρήσης αυτής της μετρικής RMSE. Ας υποθέσουμε πως θέλουμε να προβλέψουμε τις τιμές κατοικιών, σε χιλιάδες ευρώ, και πως έχουμε το παρακάτω σύνολο δεδομένων το οποίο θα το παρουσιάσουμε γραφικά:



Εικόνα 17: Παράδειγμα ενός συνόλου δεδομένων με τις προβλεπόμενες και πραγματικές τιμές σπιτιών σε χιλιάδες μονάδες.

Και εφόσον έχουμε στην εικόνα 17 το σύνολο που τεχνικά δημιουργήσαμε, τις πραγματικές τιμές των σπιτιών και τις προβλεπόμενες μέσω ενός φανταστικού μοντέλου, μπορούμε να υπολογίσουμε το RMSE αυτού του μοντέλου.

$$RMSE = \sqrt{\frac{(19.2 - 18.5)^2 + (16.7 - 16.4)^2 + (32.2 + 31.9)^2}{3}} =$$

$$RMSE = \sqrt{\frac{(0.7)^2 + (0.3)^2 + (0.3)^2}{3}} = \sqrt{\frac{0.49 + 0.09 + 0.09}{3}}$$

$$RMSE = \sqrt{0.223} = 0.47$$

Επομένως το RMSE του μοντέλου του παραδείγματος που αναφέραμε έχει την τιμή 0.47 και πλέον μπορεί να συγκριθεί με κάποιο άλλο μοντέλο για να υπολογιστούν οι αποδόσεις τους. Στην συνέχεια θα προσπαθήσουμε να δημιουργήσουμε ένα σύνολο δεδομένων το οποίο περιέχει εικόνες από φαγητά και θα προσπαθήσουμε να εκπαιδεύσουμε ένα νευρωνικό μοντέλο σύντηξης ώστε να μάθει από αυτές τις εικόνες. Όμως καλό είναι να έρθουμε σε πρώτη επαφή με το σύνολο δεδομένων που θα ασχοληθούμε.

3.3 Προετοιμασία και Επεξεργασία Δεδομένων

Σε αυτό το κεφάλαιο θα ασχοληθούμε με μεθόδους προετοιμασία Δεδομένων, που όπως αναφέραμε στο Κεφάλαιο 1 είναι ένα από τα βήματα μιας ολοκληρωμένης εξόρυξης δεδομένων. Παρόλα αυτά είναι πλέον λογικό ότι τα δεδομένα μας πρέπει να είναι σε μορφή κατάλληλη έτσι ώστε τα νευρωνικά δίκτυα που αναπτύσσουμε` να τα καταλαβαίνουν. Ξεκινάμε με την τεχνική της δυαδοποίησης(binarization).

3.3.1 Δυαδοποίηση δεδομένων (Binarization)

Η τεχνική της δυαδοποίησης μετατρέπει ένα σύνολο δεδομένων με τιμές σε ένα σύνολο δυαδικό. Ορίζοντας μία παράμετρο που ονομάζουμε κατώφλι (**threshold**), θέτουμε παράλληλα όλες τις τιμές πάνω από αυτή την παράμετρο στην τιμή 1, ενώ όλες οι άλλες τιμές κάτω από αυτή την παράμετρο κατώφλι, θέτονται στην τιμή 0. Αυτή η τεχνική χρησιμοποιείται κυρίως σε πιθανοτικά σύνολα ή σε σύνολα που θέλουμε να κατηγοριοποιήσουμε σε δύο κλάσεις (προβλήματα ταξινόμησης με δύο ενδεχόμενα τιμής στόχου). Αυτή η τεχνική μπορεί να αναγνωριστεί και ως **στρογγυλοποίηση** των αριθμών, και η **tensorflow** έχει μία μέθοδο στην βιβλιοθήκη της **tensorflow.math.round()** η οποία τις τιμές πάνω από 0.5 τις μετατρέπει σε 1 και τις τιμές κάτω από 0.5 τις μετατρέπει σε 0. Αυτό είναι χρήσιμο όταν θα έχουμε τις προβλέψεις των μοντέλων μας και θέλουμε να δούμε που

κατατάσσεται κάθε ένα δείγμα. Θα δούμε παραδείγματα παρακάτω, όταν θα δημιουργήσουμε νευρωνικά Συνελικτικά μοντέλα. [7]

Δίνουμε τον κώδικα Python και την βιβλιοθήκη tensorflow ώστε να επιτευχθεί η διαδικασία της δυαδοποίησης / στρογγυλοποίησης.

```
# Import libraries and Initializing array
import tensorflow as tf
array = tf.random.uniform(
    shape=[10, 1],
    minval=0,
    maxval=1,
    dtype=tf.dtypes.float32
)

print(array)
```

'Output'

```
tf.Tensor(
[[0.8661202 ]
 [0.06596005]
 [0.6543921 ]
 [0.9861572 ]
 [0.50255   ]
 [0.13801932]
 [0.7633536 ]
 [0.9069402 ]
 [0.05497003]
 [0.16668987]], shape=(10, 1), dtype=float32)
```

Δημιουργήσαμε ένα διάνυσμα το οποίο πλέον έχει τυχαίες τιμές μεταξύ του 0 και 1. Τώρα θα προσπαθήσουμε να το δυαδοποιήσουμε .

```
# Binarization of the array
binarized_array = tf.math.round(array)
print(binarized_array)
```

'Output'

```
tf.Tensor(
[[1.]
 [0.]
 [1.]
 [1.]
 [1.]
 [0.]
 [1.]
 [1.]
 [0.]
 [0.]], shape=(10, 1), dtype=float32)
```

Κάτι το οποίο μέσο της μεθόδους **tf.math.round()** μπορούμε να το μετατρέψουμε σε τιμές 0 και 1. Καταλαβαίνουμε επίσης πως το **threshold** / κατώφλι, είναι στην προκειμένη το 0.5.

3.3.2 Κωδικοποίηση Κατηγορικών Μεταβλητών

Στο κεφάλαιο 1, μιλήσαμε για τον τύπο δεδομένων που μπορεί να συναντήσουμε, ένα από αυτά είναι τα κατηγορικά δεδομένα. Οι αλγόριθμοι μάθησης και κατά συνέπεια τα νευρωνικά δίκτυα, δέχονται κυρίως αριθμητικές τιμές για να. μπορέσουν να λειτουργήσουν όπως είναι φτιαγμένα. Αυτό υπονοεί πως οποιαδήποτε μεταβλητή που δεν είναι αριθμητική, υπάρχει μεγάλη περίπτωση να δημιουργεί λανθασμένη λειτουργία του αλγορίθμου μάθησης. Για αυτό, στην Python έχουμε εξελίξει τεχνικές και μεθόδους που μετατρέπουν αυτά τα κατηγορικά δεδομένα σε αριθμητικά. Η τεχνική αυτή ονομάζεται κωδικοποίηση κατηγορικών δεδομένων (**Encoding Categorical Variables – Label Encoding**). [7]

Η tensorflow μας δίνει μία μέθοδο **.StringLookup** που δημιουργεί ένα επίπεδο κατηγοριοποίησης μεταβλητών ανά αριθμό και μπορούμε να δώσουμε ένα διάνυσμα με κατηγορικές μεταβλητές το οποίο θα μας τις μετατρέψει σε έναν πίνακα με αριθμητικές τιμές. Για παράδειγμα δίνεται ο κώδικας:

```
import tensorflow as tf

# Define your categorical variable
colors = tf.constant(['red', 'green', 'blue', 'green', 'red', 'blue'])

# Remove duplicates from the categorical variable
unique_colors, _ = tf.unique(colors)

# Create a StringLookup layer
color_encoder = tf.keras.layers.StringLookup(vocabulary=unique_colors,
mask_token=None)

# Encode the categorical variable using the StringLookup layer
encoded_colors = color_encoder(colors)

# Print the encoded values
print(encoded_colors)
```

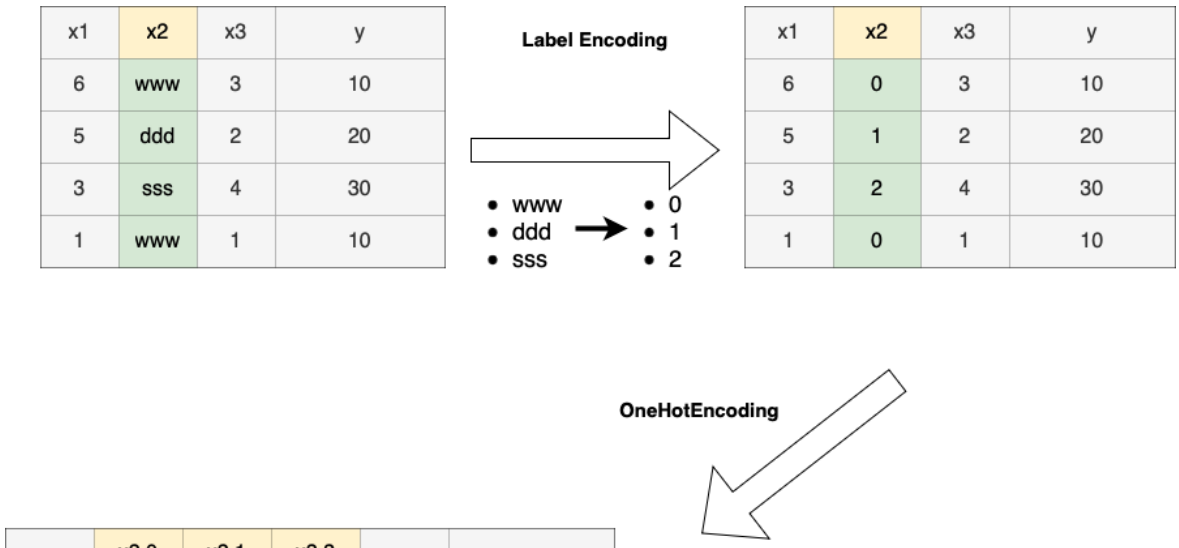
‘Output’

```
tf.Tensor([1 2 3 2 1 3], shape=(6,), dtype=int64)
```

Το οποίο μετέτρεψε κάθε χρώμα σε μία αριθμητική τιμή. Παρόλα αυτά ο πίνακας που έδωσε σαν αποτέλεσμα περιέχει τιμές από το 1 έως και το 3. Υπάρχει μία τεχνική που πηγαίνει την κωδικοποίηση των κατηγορικών μεταβλητών ένα βήμα πιο πέρα και μετατρέπει

τις κατηγορικές μεταβλητές σε στήλες όπου κάθε μία είναι αυτή η μεταβλητή και περιέχει είτε 0 είτε 1 εάν βρισκόμαστε σε αυτή την μεταβλητή. [7]

Η τεχνική αυτή ονομάζεται **OneHotEncoding** και προτιμάται στην περίπτωση που θέλουμε να δώσουμε δεδομένα για εκπαίδευση σε έναν αλγόριθμο και υπάρχουν κατηγορικές τιμές. Θα δώσουμε την παρακάτω εικόνα η οποία μας δίνει μία αίσθηση του τι γίνεται σε μία OneHotEncoding διαδικασία. [7]



x1	x2	x3	y
6	www	3	10
5	ddd	2	20
3	sss	4	30
1	www	1	10

x1	x2	x3	y
6	0	3	10
5	1	2	20
3	2	4	30
1	0	1	10

x1	x2.0 ή 'www'	x2.1 ή 'ddd'	x2.2 ή 'sss'	x3	y
6	1	0	0	3	10
5	0	1	0	2	20
3	0	0	1	4	30
1	1	0	0	1	10

*Εικόνα 18: Μετατροπή του συνόλου δεδομένων και τις κατηγορηματικές μεταβλητές της στήλης **x2** με Label Encoding και OneHotEncoding αντίστοιχα.*

Μπορούμε να πράξουμε την τεχνική της OneHotEncoding με κώδικα επίσης. Η Python και η βιβλιοθήκη tensorflow έχουν φροντίσει για αυτό, και με την μέθοδο **tf.one_hot().**

```
import tensorflow as tf

# Define input labels
labels = [1, 2, 3, 2, 0]

# Define depth of the one-hot vector
depth = 4

# Use tf.one_hot() to perform one-hot encoding
one_hot_encoded = tf.one_hot(labels, depth)
```

```
# Print the one-hot encoded labels
print(one_hot_encoded)
```

‘Output’

```
tf.Tensor(
[[0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]], shape=(5, 4), dtype=float32)
```

Το οποίο είναι το αναμενόμενο διάνυσμα.

3.3.3 Επανακλιμάκωση και Κανονικοποίηση Δεδομένων

Σε αυτό το στάδιο προσπαθούμε να βελτιώσουμε τα δεδομένα μας έτσι ώστε να είναι πιο παραγωγική, εύκολη και ταχύτερη η εκμάθηση των δεδομένων από το μοντέλο που δημιουργούμε. Με την διαδικασία της επανακλιμάκωσης και της κανονικοποίησης των τιμών του συνόλου δεδομένων, πετυχαίνουμε ακριβώς αυτό, ταχύτερη και ευκολότερη μάθηση των δεδομένων από το μοντέλο μας.

Πολύ συχνά συναντάμε σύνολα δεδομένων να περιέχουν τιμές διαφορετικού εύρους και μονάδων. Για παράδειγμα θα μπορούσε στην στήλη Α να είχαμε αριθμητικά δεδομένα με εύρος από 1 έως και 10.000 ενώ στην στήλη Β να έχουμε τιμές διαστήματος από 0 έως και 1. Αυτό δυσκολεύει την εύρεση των κατάλληλων βαρών που θα πρέπει να δώσει ο αλγόριθμος στην στήλη Α και Β έτσι ώστε να ισορροπούνται από άποψη «σημαντικότητας». Επομένως χρησιμοποιούμε τεχνικές για να προσαρμόσουμε τις τιμές αυτές σε αντίστοιχες μικρότερης κλίμακας με την ιδιότητα ότι όλες οι στήλες είναι στην ίδια κλίμακα. [7]

Η tensorflow έχει μία μέθοδο στην βιβλιοθήκη της η οποία δημιουργεί ένα αντικείμενο της το οποίο όταν εκπαιδευτεί με τα δεδομένα μας, μπορεί να μετατρέψει τα δεδομένα έτσι ώστε να έχουν ένα διάστημα από -1 έως 1 με κέντρο το 0, το οποίο προσομοιώνει την κανονική κατανομή πιθανότητας. Θεωρείται καλή πρακτική. [7]

θα εισάγουμε το σύνολο δεδομένων Iris το οποίο είναι προ εγκαταστημένο στην βιβλιοθήκη sklearn της Python.

```
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load iris dataset
data = load_iris()
X = data.data
```

```
y = data.target

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create normalization layer
normalization_layer = tf.keras.layers.Normalization()

# Fit normalization layer to training data
normalization_layer.adapt(X_train)

# Rescale training and test data using normalization layer
X_train_scaled = normalization_layer(X_train).numpy()
X_test_scaled = normalization_layer(X_test).numpy()

# Output
print(X_train_scaled[:10])
```

‘Output’

```
[[-1.4739367   1.2036583  -1.5625348  -1.3126028 ]
 [-0.13307074  2.9923766  -1.2760065  -1.0456327 ]
 [ 1.0858984   0.08570959  0.38585827  0.28921762]
 [-1.2301425   0.7564792  -1.2187006  -1.3126028 ]
 [-1.7177303   0.30929965 -1.3906178  -1.3126028 ]
 [ 0.5983112  -1.2558286   0.7296923   0.9566428 ]
 [ 0.720208    0.30929965  0.4431639   0.42270267]
 [-0.7425553   0.98006874 -1.2760065  -1.3126028 ]
 [-0.9863489   1.2036583  -1.3333122  -1.3126028 ]
 [-0.7425553   2.3216069  -1.2760065  -1.4460877 ]]
```

Το οποίο είναι κανονικοποιημένο.

Να σημειώσουμε πως η διαφορά την κλιμάκωση με την κανονικοποίησης είναι πως η κλιμάκωση (**rescaling**) αλλάζει την περιοχή των δεδομένων για παράδειγμα αν έχουμε δεδομένα από 0 έως 1 να τα κλιμακώσουμε έτσι ώστε να είναι από 0 έως 100 πολλαπλασιάζοντας κάθε τιμή με το 100.

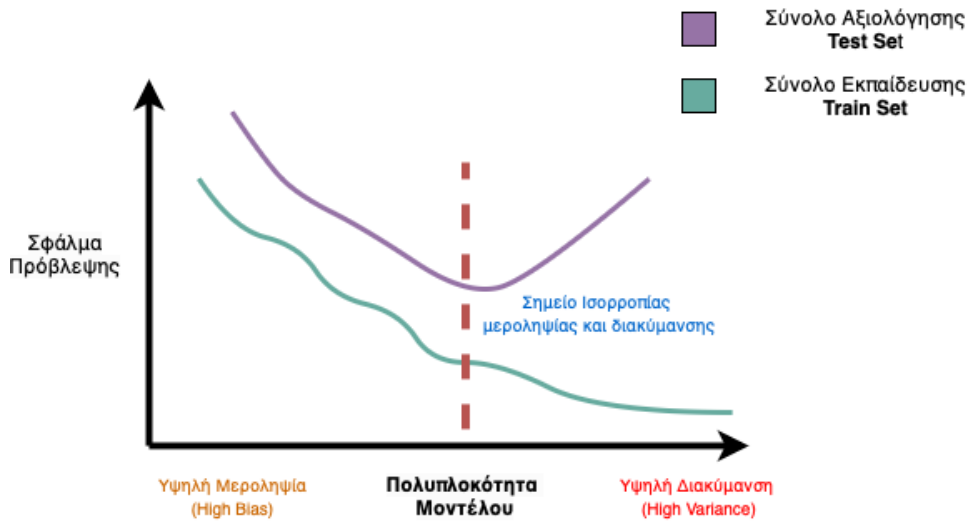
Ενώ η κανονικοποίηση (**normalization**) αλλάζει το σχήμα των δεδομένων. Η κλιμάκωση μπορεί να βοηθήσει στην αποφυγή υπερχείλισης ή υποχείλισης των δεδομένων και να βελτιώσει την αριθμητική σταθερότητα, ενώ η κανονικοποίηση μπορεί να βοηθήσει να διασφαλιστεί ότι οι χαρακτηριστικά ή οι μεταβλητές έχουν συγκρίσιμες κλίμακες, να κεντράρουν τα δεδομένα γύρω από το μηδέν και να αποτρέψουν την υπερεκπαίδευση. Η επιλογή της συγκεκριμένης τεχνικής κλιμάκωσης ή κανονικοποίησης εξαρτάται από τη φύση των δεδομένων και τις απαιτήσεις του μοντέλου μηχανικής μάθησης, όπως η επιλογή ενός

ανθεκτικού μεθόδου κλιμάκωσης για δεδομένα με ακραίες τιμές ή μια διαφορετική τεχνική κανονικοποίησης για μη κανονικά κατανεμημένα δεδομένα. [7] [1] [6]

3.4 Φαινόμενο Υπερπροσαρμογής (Overfitting)

Για να εξηγηθεί το φαινόμενο της υπερπροσαρμογής, είναι αναγκαίο να έχουμε εξηγήσει τι είναι υψηλή μεροληψία (**High Bias**) καθώς και υψηλή διακύμανση (**High Variance**). Η υψηλή μεροληψία παρατηρείται όταν το μοντέλο δεν έχει καλή απόδοση στο σύνολο εκπαίδευσης του (**train set**). Φυσικά, αυτό έπεται πως θα αποδίδει τόσο χαμηλά και σε δεδομένα στα οποία δεν έχει εκπαιδευτεί, όπως το σύνολο αξιολόγησης (**Test Set**). Η υψηλή μεροληψία δημιουργεί το λεγόμενο φαινόμενο της υπολειτουργίας του μοντέλου και λέμε, στην ουσία, πως το μοντέλο μας υπολειτουργεί στα δεδομένα μας (**Underfit**). [7]

Η υψηλή διακύμανση (**High Variance**) όμως, συμβαίνει όταν το μοντέλο μας έχει «μάθει» τόσο καλά τα δεδομένα εκπαίδευσης (**Train Set**) που φτάνει σε ένα σημείο να μην μπορεί να προβλέπει σωστά σε νέα δεδομένα, όπως το σύνολο αξιολόγησης (**Test Set**). Όταν έχουμε, υψηλή διακύμανση, λέμε πως το μοντέλο μας έχει υπερπροσαρμοστεί στα δεδομένα μας (**Overfit**). Στο παρακάτω διάγραμμα/εικόνα θα αποδώσουμε γραφικά την ισορροπία μεταξύ διακύμανσης και μεροληψίας σε συνάρτηση με την απόδοση του μοντέλου. Υπάρχει ένα σημείο που είναι πάντα επιθυμητό να βρούμε ή πετύχουμε στην εκπαίδευση του μοντέλου μας το οποίο έχει αρκετή καλή απόδοση και στο σύνολο εκπαίδευσης και στο σύνολο αξιολόγησης. [7]



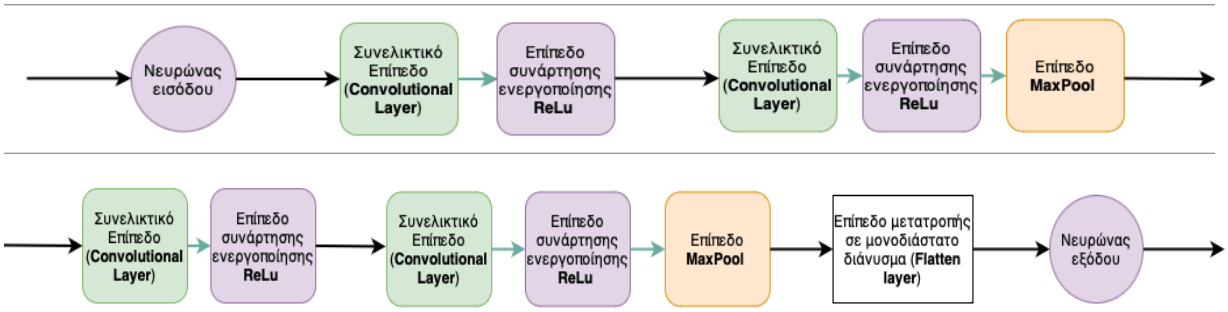
Εικόνα 19: Διάγραμμα πολυπλοκότητας μοντέλου σε σχέση με το σφάλμα του, καθώς και απεικόνιση υψηλής μεροληψίας και υψηλής διακύμανσης.

Στην εικόνα 19 αποτυπώνεται και το σημείο ισορροπίας το οποίο είναι επιθυμητό να πετύχουμε σε μία οποιαδήποτε μάθηση ενός μοντέλου. Συνήθως όταν πρόκειται για νευρωνικά δίκτυα, συναντάμε και το φαινόμενο της υψηλής μεροληψίας στην οποία περίπτωση χρειάζεται να κάνουμε πιο περίπλοκο το μοντέλο μας αλλά και το φαινόμενο της υψηλής διακύμανσης που θεωρείται η υπερπροσαρμογή του στα δεδομένα εκπαίδευσης (**Overfitting**). Σε παρακάτω κεφάλαιο θα αναλύσουμε, έναν τρόπο μέσω των επιπέδων ενός νευρωνικού δικτύου, πως μπορούμε να μειώσουμε την υπερπροσαρμογή με τη χρήση του επιπέδου **MaxPool**.

3.5 Αρχιτεκτονική Tiny-VGG

Θέλοντας να δημιουργήσουμε ένα νευρωνικό συνελικτικό μοντέλο (**convolutional network**) για βαθιά μάθηση πάνω στην μηχανική όραση, το οποίο θα επηρεάζεται μόνο από τις τιμές εισόδου, δηλαδή τα δεδομένα εκπαίδευσης, θα χρειαστεί να έχουμε εξ αρχής μία αρχιτεκτονική ενός τέτοιου μοντέλου που θεωρείται καλή στον επιστημονικό τομέα. Έχοντας πλέον εξηγήσει τον τρόπο σκεπτικής μας, αφιερωνόμαστε στην υιοθέτηση της Tiny-VGG αρχιτεκτονικής η οποία έχει υιοθετηθεί και στα μαθήματα του πανεπιστημίου του Stanford. Επίσης, το 2021 αναπτύχθηκε ιστοσελίδα και κώδικας με σκοπό την εκμάθηση της μηχανικής όρασης και των συνελικτικών νευρωνικών δικτύων σε συνεργασία με πολλούς διακεκριμένους επιστήμονες. [9]

Μένει λοιπόν να εξηγήσουμε την αρχιτεκτονική Tiny-VGG όπου θα ακολουθήσουμε στο επόμενο κεφάλαιο. Δίνεται η παρακάτω εικόνα η οποία απεικονίζει τα διαφορετικά είδη επιπέδων στο νευρωνικό δίκτυό ενός μοντέλου Tiny-VGG. [9]



Εικόνα 20: Εικόνα που δείχνει γραφικά την αρχιτεκτονική του μοντέλου Tiny-VGG, με τα αντίστοιχα επίπεδα νευρώνων που έχει και τους νευρώνες εισόδου, εξόδου.

Έχοντας αυτή την αρχιτεκτονική, η οποία θεωρείται γενικά «καλή» μπορούμε να αφοσιωθούμε στις διαφορετικές επιδόσεις των επόμενων μοντέλων πάνω σε διαφορετικές καταστάσεις; των δεδομένων εκπαίδευσης. Στο επόμενο κεφάλαιο θα ασχοληθούμε με την

προγραμματιστική ανάπτυξη των μοντέλων και τις διαφορετικές τεχνικές στην προετοιμασία των δεδομένων εκπαίδευσης καθώς και τις διαφορετικές επιδόσεις των μοντέλων.

4. Εκπαίδευση και Αποτελέσματα

Σε αυτό το κεφάλαιο θα δημιουργήσουμε μοντέλα νευρωνικών συνελικτικών δικτύων (convolutional networks) τα οποία θα ακολουθούν σταδιακά κάποιες παραμετροποιήσεις των δεδομένων που μαθαίνουν έτσι ώστε να διευκολυνθεί η σημαντικότητα αυτών των αλλαγών που γίνονται στα δεδομένα εκπαίδευσης και αξιολόγησης. Ο σκοπός είναι να καταλάβουμε την σημαντικότητα των αλλαγών που γίνονται έτσι ώστε να μην παραλείπονται αυτές οι παραμετροποιήσεις σε μία ενδεχόμενη εκμάθηση συνελικτικών δικτύων. Ξεκινάμε με τα απολυτός απαραίτητα, δηλαδή με την εισαγωγή των δεδομένων που θα ασχοληθούμε παρακάτω.

4.1 Εισαγωγή του Συνόλου Δεδομένων – Εικόνες από φαγητά (Food Dataset)

Σε αυτό το σημείο θα προσπαθήσουμε να έρθουμε σε επαφή με το σύνολο δεδομένων που θα χρησιμοποιούμε σε όλες τις μετέπειτα εκπαιδεύσεις μοντέλων που θα γίνουν. Το σύνολο αυτό θα περιέχει φωτογραφίες από φαγητά ταξινομημένα. Επομένως συνεπάγεται ότι θα βρισκόμαστε σε προβλήματα ταξινόμησης καθώς το μοντέλο θα προσπαθεί να αναγνωρίσει και να κατηγοριοποιήσει τα ενδεχόμενα είδη φαγητών.

Το αρχείο που θα χρησιμοποιήσουμε έχει δημιουργηθεί από το πανεπιστήμιο της Ζυρίχης στην εργασία τους το 2014 [7]. Μόλις κατεβάσουμε το πακέτο δεδομένων που υπάρχει στον σύνδεσμο της ιστοσελίδας που συσχετίζεται με την εργασία του πανεπιστημίου της Ζυρίχης θα έρθουμε σε επαφή με ένα αρχείο συνολικού όγκου 5GB. Με το άνοιγμα του αρχείου θα προσπαθήσουμε να καταλάβουμε τα περιεχόμενά του με κώδικα, θα δημιουργήσουμε έναν άλλον φάκελο το οποίο θα διαχωρίζει τα περιεχόμενά μας σε σύνολα εκπαίδευσης και αξιολόγησης, και στο τέλος θα εισάγουμε κώδικα ο οποίος θα διαβάζει τυχαίες εικόνες από αυτούς τους φακέλους. Επομένως με την χρήση του Google Colab εισάγουμε τα αρχεία ως εξής.

```
# Downloading the data
!wget http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz
```

Το οποίο θα αρχίσει την διαδικασία για να κατέβει το αρχείο από τον υπεύθυνο σύνδεσμο. Μόλις κατέβει θα δημιουργηθεί ένα αρχείο `'food-101.tar.gz'`. Αυτό θα χρειαστεί να το αποσυμπιέσουμε για να εμφανίσουμε το εσωτερικό του, ως εξής.

[7] Food-101 – Mining Discriminative Components with Random Forests, Bossard, Lukas and Guillaumin, Matthieu and Van Gool, Luc, European Conference on Computer Vision, 2014

```
# Import the libraries
import tarfile

# Open the File
file = tarfile.open('food-101.tar.gz')

# Extracting the File
file.extractall('./')

# Close the File
file.close()
```

Έπειτα θα δημιουργηθεί ένας φάκελος με όνομα 'food-101' το οποίο θα περιέχει τις εικόνες μας στην διεύθυνση 'food-101/images' όπου θα εμφανίσουμε το περιεχόμενό ως εξής.

```
# View the inside of "images" folder
!ls food-101/images
```

Το οποίο θα μας εμφανίσει τους φακέλους με τις εικόνες κάθε κατηγορίας φαγητού:

```
apple_pie            eggs_benedict     onion_rings
baby_back_ribs       escargots         oysters
baklava              falafel           pad_thai
beef_carpaccio       filet_mignon      paella
beef_tartare         fish_and_chips    pancakes
beet_salad           foie_gras         panna_cotta
beignets             french_fries      peking_duck
bibimbap             french_onion_soup pho
bread_pudding        french_toast      pizza
breakfast_burrito    fried_calamar     pork_chop
bruschetta           fried_rice        poutine
caesar_salad         frozen_yogurt     prime_rib
cannoli              garlic_bread      pulled_pork_sandwich
caprese_salad        gnocchi           ramen
carrot_cake          greek_salad       ravioli
ceviche              grilled_cheese_sandwich  red_velvet_cake
cheesecake           grilled_salmon     risotto
cheese_plate         guacamole         samosa
chicken_curry        gyoza             sashimi
chicken_quesadilla   hamburger         scallops
chicken_wings        hot_and_sour_soup  seaweed_salad
chocolate_cake       hot_dog           shrimp_and_grits
chocolate_mousse     huevos_rancheros  spaghetti_bolognese
churros              hummus            spaghetti_carbonara
clam_chowder         ice_cream         spring_rolls
club_sandwich        lasagna           steak
crab_cakes           lobster_bisque     strawberry_shortcake
creme_brulee         lobster_roll_sandwich    sushi
croque_madame        macaroni_and_cheese      tacos
cup_cakes            macarons                takoyaki
deviled_eggs         miso_soup               tiramisu
donuts               mussels           tuna_tartare
```

```
dumplings            nachos              waffles
edamame              omelette
```

Εμείς θα κρατήσουμε μόνο δύο φακέλους από φαγητά και αυτό διότι θέλουμε να έρθουμε σε μία ταξινόμηση δυαδική με δύο πιθανές επιλογές, για απλούστευση. Επιλέγουμε την κατηγορία φαγητού «πίτσα» (**pizza**) και «μπριζόλα» (**steak**). Ο κώδικας είναι ο εξής:

```
# Keeping only two classes/folders

# Driver function
import os
import shutil

for (root, dirs, files) in os.walk('food-101/images/', topdown=True):
  # Print current directory
  print(root)

  # so we keep only two classes
  if not root.endswith(('pizza', 'steak')):

    # Skip because is the root directory
    if root == 'food-101/images/':
      continue

    shutil.rmtree(root)
```

Το οποίο θα εμφανίζει τους φακέλους που θα διαγράφει, και με την επόμενη εντολή θα επιβεβαιώσουμε ότι πράγματι έχουμε μόνο δύο φακέλους στο αρχείο μας.

```
# View insides of "images" folder
!ls food-101/images
```

Με έξοδο:

```
pizza  steak
```

Κάτι το οποίο είναι και το ζητούμενο. Προχωράμε στο επόμενο βήμα το οποίο είναι να διαμορφώσουμε αυτά τα δεδομένα/εικόνες σε φακέλους έτσι ώστε να λαμβάνουν τον ρόλο των συνόλων εκπαίδευσης, επικύρωσης και αξιολόγησης που έχουμε αναφέρει ότι είναι πολύ σημαντικό να γίνει.

4.1.1 Χωρισμός σε φακέλους

Μένει πλέον τώρα να αντιγράψουμε τα εσωτερικά των φακέλων αυτών και να τα χωρίσουμε σε σύνολα εκπαίδευσης (**Train Set**), σύνολα αξιολόγησης (**Test Set)** και σε σύνολα επικύρωσης (**Validation Set**).

Ο τρόπος που θα δημιουργήσουμε τα σύνολα αυτά θα είναι με την βιβλιοθήκη `split-folders' όπου θα τα διανέμει σε φακέλους που έχουν τον ρόλο κάθε συνόλου που αναφέραμε. Ο λόγος που υιοθετούμε αυτή την τακτική είναι διότι μετέπειτα, όταν θα χρειαστεί να προ επεξεργαστούμε τα δεδομένα μας θα βρίσκονται ήδη στην μορφή που οι μέθοδοι που θα εφαρμόσουμε, θα αντιλαμβάνονται για να συνεχίσουμε στην εκπαίδευση του μοντέλου μας. Είναι λοιπόν για δικούς μας ευνόητους λόγους. Συνεχίζουμε όμως με την διαδικασία της δημιουργίας των συνόλων εκπαίδευσης, επικύρωσης και αξιολόγησης με ποσοστά των 70%, 10% και 20% αντίστοιχα του συνολικού συνόλου κάθε μία. Ο κώδικας είναι ο εξής:

```
# Import the nesseccary libraries
!pip install split-folders
import splitfolders
```

Έχοντας εισάγει πλέον την κατάλληλη βιβλιοθήκη, δημιουργούμε τα αρχεία μας/φακέλους/σύνολα μας ως εξής:

```
# Create folders that take up 70% Train Set, 10% Validation Set and 20%
Test Set, of total.
splitfolders.ratio('food-101/images/', seed=42, ratio=(0.7, 0.10,
0.20))
```

Όπου έτσι δημιουργείται ένα αρχείο `output` όπου περιέχει μέσα του 3 φακέλους ονομαζόμενα ως εξής:

```
# View the inside files of 'output' folder
!ls output
```

Με έξοδο:

```
test  train  val
```

Όπου κάθε ένας φάκελος από αυτούς περιέχει τις κατηγορίες των εικόνων που αφήσαμε σε προηγούμενο βήμα. Θα προσπαθήσουμε τώρα να λάβουμε ως τιμές ενός πίνακα τα ονόματα των κατηγοριών των εικόνων που έχουμε για εκπαίδευση. Ο κώδικας θα είναι ως εξής:

```
# get the classnames programmatically
import pathlib
import numpy as np
data_dir = pathlib.Path("output/train")
class_names = np.array(sorted([item.name for item in
data_dir.glob("*")])) # Created a list of class_names from the
subdirectories
print(class_names)
```

Με έξοδο το ζητούμενο:

```
['pizza' 'steak']
```

4.1.2 Απεικόνιση τυχαίας εικόνας από το σύνολο εκπαίδευσης μας

Σε αυτή την φάση της προετοιμασίας των εικόνων μας, καλό είναι να καταλάβουμε το σύνολο μας βλέποντας μερικές τυχαίες εικόνες από το σύνολο εκπαίδευσης μας. Έχουμε δημιουργήσει την συνάρτηση `view_random_image` η οποία λαμβάνει την διεύθυνση του φακέλου με τις εικόνες που έχουμε αποθηκεύσει και την αντίστοιχη κλάση ονομαστικά. Ο κώδικας είναι ο εξής:

```
# Import nesseccary libraries
from typing import Mapping
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random

# Create function that views random images
def view_random_image(target_dir, target_class):
  # Setup the target directory ( we'll view images from here)
  target_folder = target_dir +"/"+ target_class
  # Get a random image path
  random_image = random.sample(os.listdir(target_folder), 1)
  print("Random Image:",random_image)
  # Read in the image and plot it using matplotlib
  img = mpimg.imread(target_folder + "/" + random_image[0])
  plt.imshow(img)
  plt.title(target_class)
  plt.axis("off");

  print(f"Image shape: {img.shape}") # Show the shape of the image

  return img
```

Έχοντας πλέον δημιουργήσει την κατάλληλη συνάρτηση, μπορούμε να την καλέσουμε με τα κατάλληλα ορίσματα και έχουμε:

```
# View a random image from the training dataset
img = view_random_image(target_dir="output/train",
                        target_class="steak")
```

Όπου σαν έξοδο λαμβάνουμε:

```
Random Image: ['290406.jpg']
Image shape: (306, 512, 3)
```

Και κάθε φορά που θα καλούμε την συνάρτηση θα διαλέγει τυχαία μία εικόνα από μπριζόλα. Σε αυτό το βήμα καλό είναι να καταλάβουμε το σύνολό μας και συγκεκριμένα τις εικόνες μας παραπάνω, επομένως ας δούμε τις διαστάσεις και πως αναπαρίστανται σαν διάνυσμα **tensor** της βιβλιοθήκης **tensorflow**.

```
# View dimensions of the image returned.
img.shape
```

`Output`:

```
(384, 512, 3)
```

Το οποίο υπονοεί πως έχουμε ένα 3 διαστάσεων πίνακα με την πρώτη διάσταση να έχει 384 τιμές, το δεύτερο 512 και το 3ο να έχει 3 τιμές το οποίο είναι στην ουσία οι τιμές του συστήματος R.G.B. το οποίο δίνει χρώμα στην εικόνα. Η εικόνα μας σαν τιμή ενός tensor πίνακα είναι ως εξής:

```
# Make the image a tensor array.
import tensorflow as tf
tf.constant(img)
```

Με έξοδο:

```
<tf.Tensor: shape=(384, 512, 3), dtype=uint8, numpy= array([[[ 88, 72,
47], [ 88, 72, 47], [ 90, 74, 49], ..., [ 55, 58, 89], [ 57, 62, 84], [
60, 66, 82]], [[ 88, 72, 47], [ 89, 73, 48], [ 90, 74, 49], ..., [ 57,
61, 90], [ 58, 63, 85], [ 62, 68, 84]], [[ 89, 73, 50], [ 90, 74, 51],
[ 90, 74, 51], ..., [ 61, 62, 92], [ 63, 65, 88], [ 64, 70, 86]], ...,
[[133, 121, 125], [140, 123, 131], [144, 122, 134], ..., [ 87, 52, 10],
[ 88, 54, 8], [ 88, 54, 8]], [[132, 116, 119], [136, 115, 122], [137,
114, 124], ..., [ 88, 54, 9], [ 87, 53, 7], [ 88, 54, 6]], [[137, 121,
122], [135, 114, 119], [135, 109, 120], ..., [ 88, 51, 7], [ 86, 49,
4], [ 85, 49, 1]]], dtype=uint8)>
```

Εύκολα καταλαβαίνει κανείς ότι οι τιμές κάθε πίνακα είναι από 0 έως και 255 και οι διαστάσεις αναπαράστασης των εικόνων είναι έτσι όπως τις εξηγήσαμε σε προηγούμενο κεφάλαιο. Παρακάτω θα δώσουμε κώδικα για την δημιουργία του πρώτου μας συνελικτικού νευρωνικού δικτύου (**Convolutional Neural network**) κατάλληλο για αναγνώριση εικόνων.

4.2 Δημιουργία μοντέλου – Πειραματικά 1

4.2.1 Επεξήγηση Κώδικα

Σε πρώτη φάση θα δώσουμε τον κώδικα κατάλληλο να φτιάξει το πρώτο μας μοντέλο και στην συνέχεια θα εξηγήσουμε τον κώδικα στα διάφορα μέρη που τον αποτελούν. Θα ονομάσουμε το πρώτο μας μοντέλο ως **model_1** και σε κάθε αλλαγή που θα κάνουμε στα δεδομένα εκπαίδευσής του θα αλλάζουμε την ονομασία σε **model_1_1** για την πρώτη αλλαγή και σε **model_1_2** για την δεύτερη αλλαγή και πάει λέγοντας.

Να θυμίσουμε ο σκοπός μας είναι να δούμε τις διαφορετικές επιδόσεις που θα λάβουμε κατά την αξιολόγησή του μοντέλου μας, αλλάζοντας μόνο τα δεδομένα εκπαίδευσης, με μεθόδους προ-επεξεργασίας αυτών. Η βιβλιοθήκη της tensorflow έχει στην διάθεσή της συναρτήσεις και αντικείμενα οι οποίες βοηθάνε στην προ-επεξεργασία των δεδομένων που βρίσκονται στην μορφή εικόνων. Εμείς θα χρησιμοποιήσουμε αυτές τις τεχνικές, ανήκουν στην βιβλιοθήκη της **tensorflow.preprocessing.image.ImageDataGenerator** και δημιουργούν τα αντίστοιχα αντικείμενα, όπως θα δούμε. Δίνεται ο παρακάτω κώδικας:

```
# Make the data suitable for fitting in the upcoming model
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Δημιουργούμε τα αντικείμενα της κλάσης **ImageDataGenerator** που θα είναι στις μεταβλητές

train_datagen, **test_datagen**, **validation_datagen** όπου κάθε μία μεταβλητή θα είναι υπεύθυνη για το αντίστοιχο σύνολο εκπαίδευσης, αξιολόγησης και επικύρωσης, όπως ορίζεται στην ονοματολογία τους.

```
# Preprocess data (get all of the pixel values between 1 and 0, also
called scaling/normalization)
train_datagen = ImageDataGenerator()
test_datagen = ImageDataGenerator()
validation_datagen = ImageDataGenerator()
```

Έπειτα με την μέθοδο **.flow_from_directory()** της κλάσης **ImageDataGenerator** ορίζουμε το μέγεθος στο οποίο θέλουμε κάθε εικόνα να μετατραπεί στην προκειμένη θέλουμε να έχουμε εικόνες μεγέθους 224 x 224 pixels και αυτό θα επιτευχθεί με το argument **target_size.** Επίσης με την επιλογή του **"binary"** στο argument class_mode διαλέγουμε τον τύπο προβλήματος που θα αντιμετωπίσουμε, και μίας και έχουμε μόνο δύο κλάσεις ταξινόμησης, βρισκόμαστε σε δυαδική μορφή. Τέλος, ορίζουμε το argument **shuffle** στην τιμή **False** έτσι ώστε όταν θα εκπαιδευτεί το μοντέλο να μην λαμβάνει τυχαίες εικόνες από το σύνολό μας αλλά να τις επιλέγει την μία μετά την άλλη, αυτό συμβαίνει για να δείξουμε την σημαντικότητα της τυχαιοποίησης των δεδομένων μάθησης στις επιδόσεις των μοντέλων που εκπαιδεύουμε.

```
# Setup the train and test directories
train_dir = "output/train/"
test_dir = "output/test/"
validation_dir = "output/val/"

# Import data from directories and turn it into batches
train_data = train_datagen.flow_from_directory(train_dir,
                                               target_size=(224, 224),
                                               class_mode='binary',
                                               shuffle=False)

test_data = test_datagen.flow_from_directory(test_dir,
                                             target_size=(224, 224),
                                             class_mode='binary',
                                             shuffle=False)

validation_data = test_datagen.flow_from_directory(validation_dir,
                                                   target_size=(224, 224),
                                                   class_mode='binary',
                                                   shuffle=False)
```

Έχοντας λοιπόν ολοκληρώσει το μέρος της προετοιμασίας των εικόνων μας μέσω της κλάσης **ImageDataGenerator** θα δώσουμε τον κώδικα της δημιουργίας του πρώτου μας μοντέλου με τα αντίστοιχα επίπεδα της αρχιτεκτονικής Tiny-VGG που ορίσαμε παραπάνω. Ο κώδικας είναι ο εξής:

```
# Train the model
# Import libraries
from tensorflow.keras import Sequential
from tensorflow.keras.layers import  Conv2D, MaxPool2D, Flatten, Dense

# Create a CNN model (same as Tiny VGG but for binary classification -
https://poloclub.github.io/cnn-explainer/ )

model_1 = Sequential([

  Conv2D(filters=10, kernel_size=3, activation='relu',
  input_shape=(224, 224, 3)),
  Conv2D(filters=10, kernel_size=3, activation='relu' ),
  MaxPool2D(),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Flatten(),
  Dense(1, activation='sigmoid')
])
```

Όπως έχουμε μάθει στην δημιουργία του μοντέλο μας, το πρώτο βήμα είναι η αρχιτεκτονική. Παρατηρούμε πως έχουμε προσθέσει τα επίπεδα **Conv2D**, **MaxPool2D**, **Flatten** και **Dense**. Τα επίπεδα **Conv2D** είναι τα Συνελικτικά επίπεδα του μοντέλου μας και είναι αυτά υπεύθυνα που το μετατρέπουν το μοντέλο μας σε ένα συνελικτικό μοντέλο. Το επίπεδο **MaxPool2D** είναι το επίπεδο αναγκαίο για την μείωση των Pixels κατά την διάρκεια της μάθηση από το μοντέλο μας, για την αποφυγή της υπερπροσαρμογής. Το επίπεδο **Flatten** του μοντέλου μας προσαρμόζει όλες τις τιμές σε ένα μονοδιάστατο διάνυσμα κατάλληλο να εισαχθεί στο επόμενο επίπεδο του μοντέλου μας **Dense** το οποίο θεωρείται ένα απλό επίπεδο νευρώνων του μοντέλου μας έτσι ώστε να αναπαράγει μετά μία τιμή εξόδου μέσα από την σιγμοειδής συνάρτηση. Όλα αυτά είναι αποσκοπούν στην παρομοίωση της Tiny-VGG αρχιτεκτονικής έτσι ώστε να έχουμε ένα μοντέλο σχετικά «καλό» και η μόνη διαφορά ανάμεσα στις επόμενες εκπαιδεύσεις είναι η παραμετροποίηση των δεδομένων που

εκπαιδεύεται το μοντέλο μας. Σε συνέχεια θα δοθεί εξήγηση του κώδικα για να παραμετροποιήσουμε το μοντέλο μας και μετά στο να το εκπαιδεύσουμε, τα οποία είναι τα βήματα 2 και 3 όπως έχουν αναφερθεί στην Ενότητα 2.5.

```
# Compile the model
model_1.compile(loss="binary_crossentropy",
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])

# Fit the model
history_1 = model_1.fit(train_data,
                        epochs=5,
                        steps_per_epoch=len(train_data),
                        validation_data=validation_data,
                        validation_steps=len(validation_data)
                        )
```

Παρατηρούμε πως στην μέθοδο **.compile()** η τιμή της argument **loss** είναι η «**binary_crossentropy**» η οποία είναι μία συνάρτηση χρησιμοποιούμενη έτσι ώστε να μετρά την διαφορά της πραγματικής με προβλεπόμενης τιμής σε προβλήματα δυαδικής ταξινόμησης όπως σε αυτό που βρισκόμαστε τώρα. Το argument **optimizer** έχει τεθεί ώς **Adam** δηλαδή ο τρόπος με τον οποίο το μοντέλο θα «διορθώνει» τον εαυτό του ανάλογα με την τιμή της **loss** είναι με την τεχνική Adam. Και στο argument **metrics** θέσαμε την μετρική accuracy για feedback στον χρήστη.

Κατά την εκπαίδευση του μοντέλου μας ορίζουμε την μεταβλητή **history_1** η οποία θα έχει τον ρόλο της ιστορικού του μοντέλου μας, για να μπορέσουμε μετέπειτα να παρουσιάσουμε γραφικά τις επιδόσεις του καθόλη την διάρκεια εκπαίδευσής του. Προφανώς έχουμε εισάγει τα δεδομένα μας train_data, έπειτα η τιμή **epochs** αναφέρει πόσες επαναλήψεις των δεδομένων το μοντέλο μας θα κάνει, εμείς διαλέξαμε 5. Έχουμε επίσης και τα δεδομένα επικύρωσης που φτιάξαμε προηγουμένως, τα οποία θα λειτουργούν σαν μία μέτρηση των επιδόσεων κατά την διάρκεια του μοντέλου μας, σε σύνολα που δεν έχει εκπαιδευτεί.

Τρέχοντας όλο τον προηγούμενο κώδικα, θα λάβουμε για τιμές εξόδου τις εξής:

```
Found 1400 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
Epoch 1/5
44/44 [==============================] - 15s 258ms/step - loss: 9.0246
- accuracy: 0.5564 - val_loss: 2.3213 - val_accuracy: 0.5000
Epoch 2/5
```

```
44/44 [==============================] - 9s 194ms/step - loss: 0.7853 -
accuracy: 0.6386 - val_loss: 1.0824 - val_accuracy: 0.4800
Epoch 3/5
44/44 [==============================] - 8s 170ms/step - loss: 0.6756 -
accuracy: 0.6257 - val_loss: 0.7234 - val_accuracy: 0.5650
Epoch 4/5
44/44 [==============================] - 9s 194ms/step - loss: 0.5149 -
accuracy: 0.7379 - val_loss: 0.7370 - val_accuracy: 0.6100
Epoch 5/5
   44/44 [==============================] - 8s 192ms/step - loss: 0.4073
- accuracy: 0.8307 - val_loss: 0.7064 - val_accuracy: 0.6250
```

Το οποίο μας φανερώνει πληροφορίες για τα σύνολα δεδομένων μας και τέλος την εκπαίδευση του αλγορίθμου καθώς και στατιστικά επίδοσης της εκπαίδευσης. Μπορούμε να δούμε την πορεία της μάθησης ανά επανάληψη (τιμή που δώσαμε στο argument **epoch**) και να δούμε πως μαθαίνει τα δεδομένα μας κάθε επανάληψη το μοντέλο μας. Στην συνέχεια θα δούμε μία συνάρτηση η οποία θα μας εμφανίζει τις επιδόσεις του μοντέλου μας στο σύνολο επικύρωσης, κατά την εκπαίδευσή του. Αυτό θα συμβεί με χρήση της μεταβλητής **history** που θέσαμε κατά την μέθοδο **.fit()** του μοντέλου μας.

4.2.2 Συνάρτηση εμφάνισης επιδόσεων στο σύνολο επικύρωσης

Θα δώσουμε μία συνάρτηση δημιουργημένη στην Python από εμάς, όπου με χρήση της βιβλιοθήκης matplotlib θα έχουμε την ικανότητα να εμφανίζουμε με όμορφο και γραφικό τρόπο τις επιδόσεις των μοντέλων μας κατά την διάρκεια της εκπαίδευσή τους. Ξεκινάμε δίνοντας τον κώδικα και στην συνέχεια θα τον εξηγήσουμε.

```
import matplotlib.pyplot as plt
# Plot the validation and training data separately
def plot_loss_curves(history):
  """
  Returns separate loss curves for training and validation metrics.
  """
  loss = history.history['loss']
  val_loss = history.history['val_loss']

  accuracy = history.history['accuracy']
  val_accuracy = history.history['val_accuracy']

  epochs = range(len(history.history['loss']))

  # Plot loss
  plt.plot(epochs, loss, label='training_loss')
  plt.plot(epochs, val_loss, label='val_loss')
  plt.title('Loss')
```

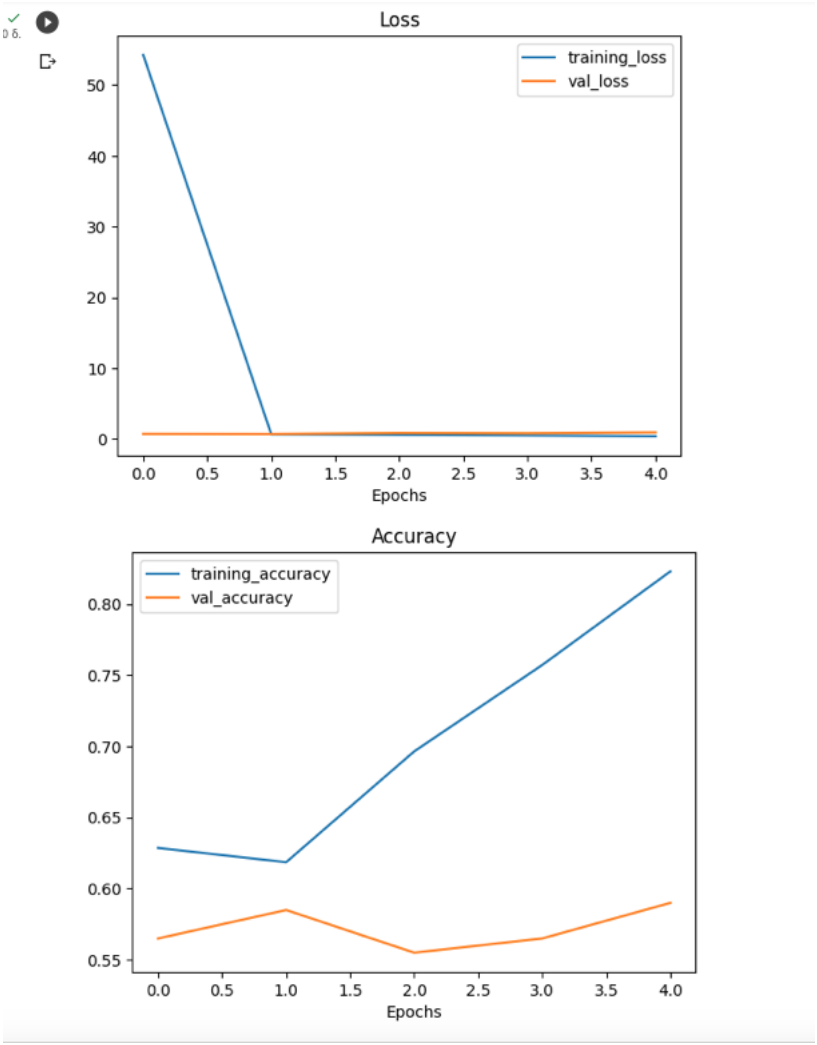
```
  plt.xlabel('Epochs')
  plt.legend()

  # Plot accuracy
  plt.figure()
  plt.plot(epochs, accuracy, label='training_accuracy')
  plt.plot(epochs, val_accuracy, label='val_accuracy')
  plt.title('Accuracy')
  plt.xlabel('Epochs')
  plt.legend();
```

Σαν πρώτο όρισμα της συνάρτησης **plot_loss_curves()** δίνουμε την μεταβλητή **history** που θέσαμε κατά την εκπαίδευση του αλγορίθμου με την μέθοδο **.fit()**. Αυτή η μεταβλητή περιέχει μέσα της τις πληροφορίες που χρειαζόμαστε για κάθε μία επανάληψη μάθησης που το μοντέλο μας κάνει, αυτές οι επαναλήψεις είναι σε αριθμό όσο θέσαμε στην μεταβλητή **epoch** κατά την εκπαίδευση του μοντέλου μας σε προηγούμενο βήμα.

Έπειτα με την βιβλιοθήκη της **matplotlib** δημιουργούμε δύο γραφήματα ένα για την μεταβλητή **loss** που θέσαμε στην εκπαίδευση του μοντέλου μας και ένα γράφημα για την ακρίβεια του μοντέλο μας. Βέβαια θα διαφοροποιήσουμε την ακρίβεια στο σύνολο εκπαίδευσης και την ακρίβεια που έχει στο σύνολο επικύρωσης, που θυμίζουμε είναι δεδομένα που δεν έχει έρθει σε επαφή ξανά. Δίνουμε ένα παράδειγμα, με την μεταβλητή **history_1** η οποία εκπροσωπεί το **model_1** που εκπαιδεύσαμε προηγουμένως.

```
# Reveal loss and accuracy plots
plot_loss_curves(history_1)
```

Με έξοδο:



*Εικόνα 21: Εμφάνιση εξόδου, της συνάρτησης **plot_loss_curves()** με τις επιδόσεις του μοντέλου 1 (model_1) που φτιάξαμε.*

Παρατηρούμε πως το μοντέλο μας συμπεριφέρεται αρκετά με χαμηλές επιδόσεις, μιας και η ακρίβεια του μοντέλου μας έφτασε τόσο χαμηλά όσο και 0.55. Πάντα λαμβάνουμε την ακρίβεια του συνόλου επικύρωσης μιας και δίνεται να μαντέψει τιμές από σύνολο που δεν έχει εκπαιδευτεί. Αυτό συμβαίνει για πολλούς λόγους, αλλά μπορούμε να αρχίσουμε να τους διορθώνουμε σταδιακά. Στην προετοιμασία των δεδομένων ας δοκιμάσουμε αυτή την φορά να ανακατέψουμε τα δεδομένα των δύο κλάσεων σε όλα τα σύνολα εκπαίδευσης, επικύρωσης και αξιολόγησης.

4.2.3 Τυχαιοποίηση Δεδομένων (Shuffle)

Στην πρώτη μας εκπαίδευση του μοντέλου μας, θέσαμε την μεταβλητή **shuffle=False** κατά την προετοιμασία των δεδομένων μας. Αυτό συνέβη και στα 3 σύνολα που φτιάχναμε δηλαδή σύνολο εκπαίδευσης (**Train Set**), σύνολο επικύρωσης (**Validation Set**) και σύνολο αξιολόγησης (**Test Set**). Αυτό είχε σαν αποτέλεσμα το μοντέλο μας να μην εκπαιδευτεί σωστά, εφόσον δεν έχουν ανακατευτεί οι εικόνες εκπαίδευσης κατά την εκπαίδευσή του. Επομένως δεν ήταν μεροληπτική η διαδικασία εκπαίδευσή του. Ένας τρόπος να το διορθώσουμε αυτό είναι να θέσουμε την μεταβλητή **shuffle=True**. Επομένως ένας το κάνουμε αυτό και το ονομάσουμε το μοντέλο μας **model_1_1**. Έχουμε τον κώδικα:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Preprocess data
train_datagen = ImageDataGenerator()
test_datagen = ImageDataGenerator()
validation_datagen = ImageDataGenerator()

# Setup the train and test directories
train_dir = "output/train/"
test_dir = "output/test/"
validation_dir = "output/val/"

# Import data from directories
train_data = train_datagen.flow_from_directory(train_dir,
                                               target_size=(224, 224),
                                               class_mode="binary",
                                               shuffle=True,
                                               )

test_data = test_datagen.flow_from_directory(test_dir,
                                             #  batch_size=None,
                                              target_size=(224, 224),
                                              class_mode='binary',
                                              shuffle=True)

validation_data = test_datagen.flow_from_directory(validation_dir,
                                             #  batch_size=None,
                                              target_size=(224, 224),
                                              class_mode='binary',
                                              shuffle=True)

# Train the model
from tensorflow.keras import Sequential
```

```
from tensorflow.keras.layers import  Conv2D, MaxPool2D, Flatten, Dense

# Create a CNN model (same as Tiny VGG but for binary classification -
https://poloclub.github.io/cnn-explainer/ )
model_1_1 = Sequential([
  Conv2D(10, 3, activation='relu', input_shape=(224, 224, 3)), # same
input shape as our images
  Conv2D(10, 3, activation='relu'),
  MaxPool2D(),
  Conv2D(10, 3, activation='relu'),
  Conv2D(10, 3, activation='relu'),
  MaxPool2D(),
  Flatten(),
  Dense(1, activation='sigmoid')
])

# Compile the model
model_1_1.compile(loss="binary_crossentropy",
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])

# Fit the model
history_2 = model_1_1.fit(train_data,
                        epochs=5,
                        steps_per_epoch=len(train_data),
                        validation_data=validation_data,
                        validation_steps=len(validation_data)
                        )
```

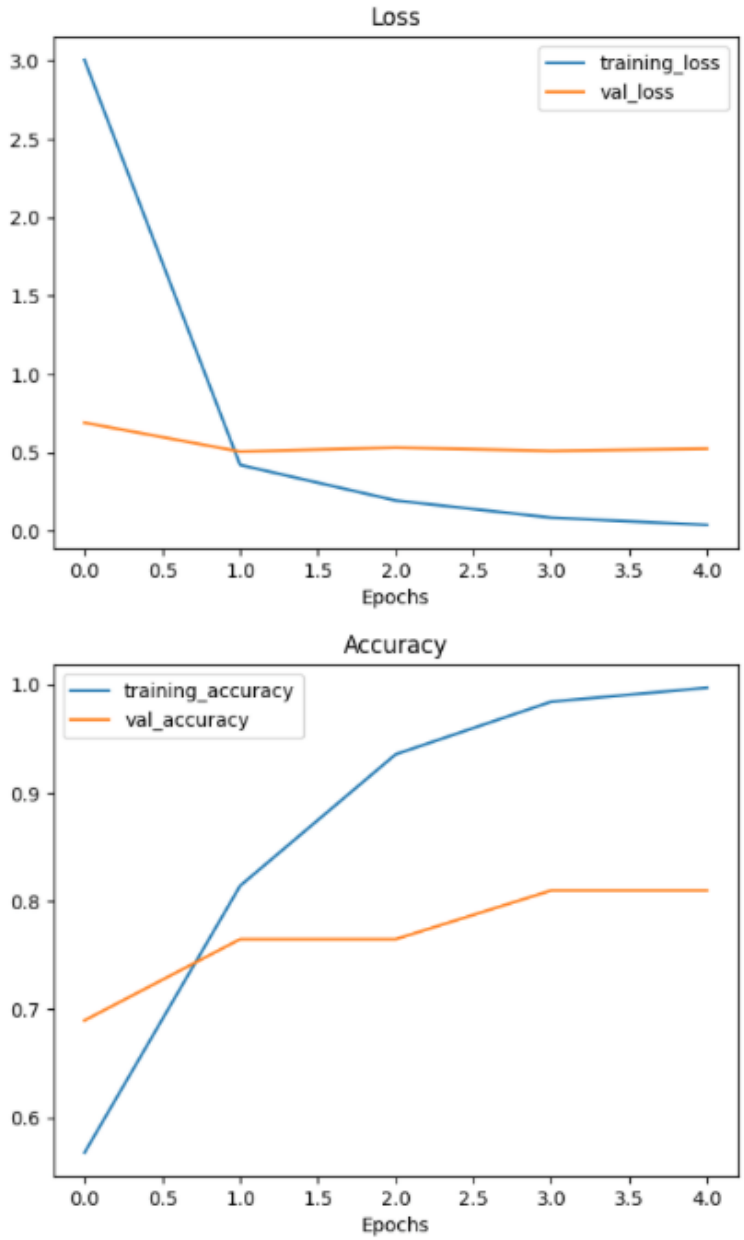
Με έξοδο:

```
Found 1400 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
Epoch 1/5
44/44 [==============================] - 14s 277ms/step - loss: 3.0040
- accuracy: 0.5679 - val_loss: 0.6903 - val_accuracy: 0.6900
Epoch 2/5
44/44 [==============================] - 11s 252ms/step - loss: 0.4206
- accuracy: 0.8143 - val_loss: 0.5058 - val_accuracy: 0.7650
Epoch 3/5
44/44 [==============================] - 9s 212ms/step - loss: 0.1938 -
accuracy: 0.9357 - val_loss: 0.5315 - val_accuracy: 0.7650
Epoch 4/5
44/44 [==============================] - 8s 179ms/step - loss: 0.0842 -
accuracy: 0.9843 - val_loss: 0.5100 - val_accuracy: 0.8100
Epoch 5/5
44/44 [==============================] - 8s 191ms/step - loss: 0.0377 -
accuracy: 0.9971 - val_loss: 0.5241 - val_accuracy: 0.8100
```

Εύκολα κανείς με την συνάρτηση **plot_loss_curves()** λαμβάνει γραφικά τις επιδόσεις του μοντέλου ως εξής:

```
# Plot loss, accuracy
plot_loss_curves(history_2)
```

Με έξοδο:



Εικόνα 22: Γραφική αναπαράσταση των επιδόσεων του μοντέλου model_1_1 με τυχαιοποίησης των δεδομένων στην προετοιμασία τους.

Παρατηρούμε ότι από θέμα απλών αριθμητικών το μοντέλο **model_1_1** σε αυτή την περίπτωση ξεπέρασε το 0.6 σε ακρίβεια του συνόλου επικύρωσης (**val_accuracy**) που θεωρούταν οριακή τιμή στο **model_1**. Αυτό συμβαίνει λόγο της τυχαιοποίησης των εικόνων όταν εκπαιδευόταν το μοντέλο. Παρατηρούμε πως επίσης, η ακρίβεια στο σύνολο εκπαίδευσης, είναι πάρα πολύ υψηλή, όμως μην σας ξεγελά αυτό το φαινόμενο, φαίνεται πως το μοντέλο μας έχει την τάση της υπερπροσαρμογής του στα μοντέλα εκπαίδευσης

(**overfitting**), θα δούμε παρακάτω μερικές τεχνικές στην προετοιμασία των δεδομένων ώστε να το αποφύγουμε κατά έναν βαθμό.

4.2.4 Κανονικοποίηση Δεδομένων (Normalization)

Είχαμε αναφέρει σε προηγούμενο κεφάλαιο πως το σύνολο δεδομένων μας μπορεί να περιέχει τιμές από ένα διάστημα τιμών. και είχαμε επίσης αναφέρει πως μία καλή τεχνική είναι να μειώνουμε αυτό το διάστημα σε ένα διάστημα κοινό για όλα τα χαρακτηριστικά, όπως η απόσταση τιμών από το 0 έως και το 1. Αυτή η τεχνική ονομάζεται επανακλιμάκωση, και διαφέρει της τεχνικής κανονικοποίησης που επίσης αναφέραμε σε προηγούμενο κεφάλαιο.

θα προσπαθήσουμε να επανακλιμακώσουμε τις τιμές των δεδομένων μας έτσι ώστε να έχουν το ίδιο διάστημα. Είναι σημαντικό να αναφέρουμε, πως κάθε τιμή ενός pixel μίας εικόνας παίρνει τιμές από το 0 έως και το 255. Αυτό μας ευνοεί από την άποψη πως με μια απλή διαίρεση των τιμών με την μέγιστη τιμή τους, το 255 στην προκειμένη, μπορούμε να πάρουμε δεκαδικές τιμές από το 0 έως και το 1.

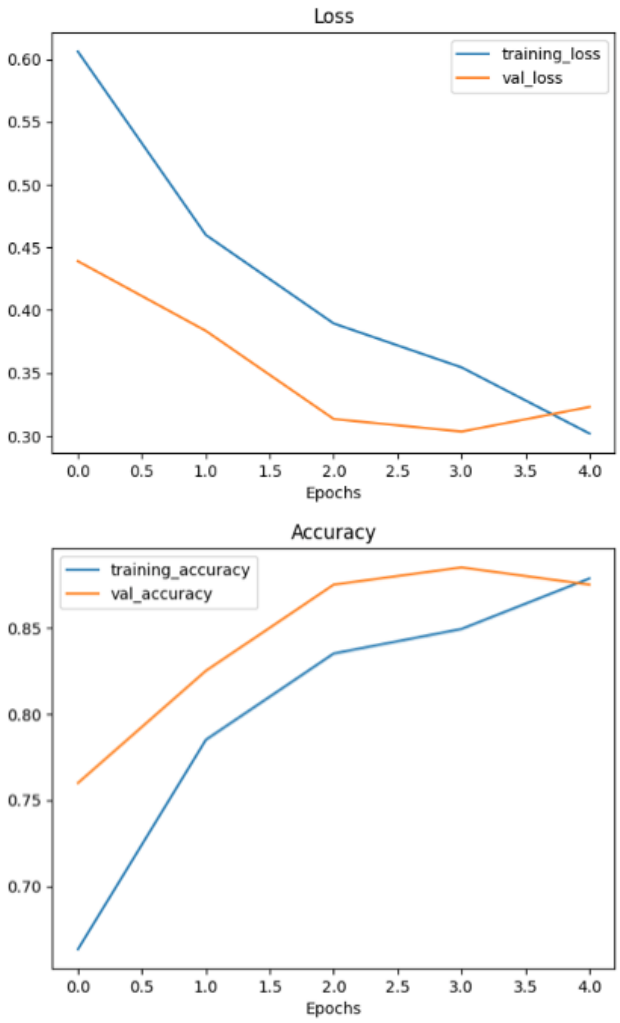
Ευτυχώς, η tensorflow έχει ένα argument κατά την δημιουργία των αντικειμένων της κλάσης **ImageDataGenerator()**, η οποία μπορεί να επανακλιμακώσει όλες τις τιμές που λαμβάνει από το σύνολό μας χωρίς να χρειαστεί να κάνουμε περαιτέρω ενέργειες εμείς οι ίδιοι. Αυτό το argument είναι το argument **rescale.** Κατά την συνέχεια θα προσπαθήσουμε να κανονικοποιήσουμε τα δεδομένα, προσθέτοντας στην αρχιτεκτονική του μοντέλου μας ένα επίπεδο κανονικοποίησης, το οποίο θα εισάγουμε με την βιβλιοθήκη της tensorflow. Δίνεται ο κώδικας που κάνει τα παραπάνω:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Preprocess data (get all of the pixel values between 1 and 0, also
called scaling/normalization)
train_datagen = ImageDataGenerator(rescale=1/255.)
test_datagen = ImageDataGenerator(rescale=1/255.)
validation_datagen = ImageDataGenerator(rescale=1/255.)

# Setup the train and test directories
train_dir = "output/train/"
test_dir = "output/test/"
validation_dir = "output/val/"

# Import data from directories and turn it into batches
train_data_rescaled = train_datagen.flow_from_directory(train_dir,
                                                        target_size=(224, 224),
```

```
                                                  class_mode="binary",
                                                  shuffle=True)

test_data_rescaled = test_datagen.flow_from_directory(test_dir,
                                                  target_size=(224, 224),
                                                  class_mode='binary',
                                                  shuffle=True)

validation_data_rescaled =
test_datagen.flow_from_directory(validation_dir,
                                                  target_size=(224, 224),
                                                  class_mode='binary',
                                                  shuffle=True)

# Train the model
# Import libraries
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense,
Normalization

# Create a CNN model (same as Tiny VGG but for binary classification -
https://poloclub.github.io/cnn-explainer/ )
model_1_2 = Sequential([
  Normalization(axis=None),
  Conv2D(filters=10, kernel_size=3, activation='relu',
input_shape=(224, 224, 3)),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Flatten(),
  Dense(1, activation='sigmoid')
])

# Compile the model
model_1_2.compile(loss="binary_crossentropy",
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])

# Fit the model
history_3 = model_1_2.fit(train_data_rescaled,
                        epochs=5,
                        steps_per_epoch=len(train_data_rescaled),
                        validation_data=validation_data_rescaled,
                        validation_steps=len(validation_data_rescaled)
                        )
```

Όπως παρατηρούμε έγινε και η επανακλιμάκωση στην προετοιμασία των δεδομένων και προστέθηκε το επίπεδο κανονικοποίησης των δεδομένων στην αρχιτεκτονική του μοντέλου μας **model_1_2.** Εφαρμόζοντας πάλι την συνάρτηση εμφάνισης των επιδόσεων του μοντέλου μας παίρνουμε τα εξής γραφήματα.

```
# Plot Accuracy, loss
plot_loss_curves(history_3)
```



*Εικόνα 23: Γραφήματα του μοντέλου **model_1_2** στο οποίο έχει εφαρμοστεί η μέθοδος της τυχαιοποίησης, επανακλιμάκωσης και κανονικοποίησης των δεδομένων.*

Παρατηρούμε πως έχουμε καλύτερες επιδόσεις στην εκπαίδευση του μοντέλου μας με αυτές τις αλλαγές που κάναμε. Παρόλα αυτά είναι καλό να το αξιολογήσουμε και με τα δεδομένα αξιολόγησης. Έχουμε τον κώδικα:

```
# Evaluation of model_1_2
from sklearn.metrics import classification_report
predictions_3 = model_1_2.predict(test_data_rescaled)
predicted_3 = tf.math.round(predictions_3)
```

```
report = classification_report(test_data_rescaled.classes, predicted_3)
print(report)
```

Το οποίο μας δίνει την έξοδο:

```
13/13 [==============================] - 3s 192ms/step
              precision    recall  f1-score   support

           0       0.47      0.54      0.50       200
           1       0.45      0.38      0.41       200

    accuracy                           0.46       400
   macro avg       0.46      0.46      0.46       400
weighted avg       0.46      0.46      0.46       400
```

Παρατηρούμε ότι σε δεδομένα στα οποία δεν έχει έρθει καθόλου σε επαφή, το μοντέλο μας δεν γενικεύει καλά τις προβλέψεις του, επομένως έχει υπερπροσαρμοστεί στα δεδομένα εκπαίδευσης. Αυτό μπορούμε να το συμπεράνουμε και με την υψηλή ακρίβεια κατά την εκπαίδευσή του. Ένας τρόπος να το λύσουμε αυτό, είναι να δημιουργήσουμε κατά την εκπαίδευσή του, επαυξημένα δεδομένα. Εξηγούμε σε παρακάτω ενότητα.

4.2.5 Μάθηση με ομάδες δεδομένων (Batches of Data)

Κατά την διάρκεια της εκπαίδευσης, το μοντέλο μας, «βλέπει» όλες τις εικόνες ταυτόχρονα. Αυτό έχει σαν αποτέλεσμα η μνήμη και η ταχύτητα της εκπαίδευσής του να αυξάνονται δραματικά. Σε προηγούμενες εκπαιδεύσεις, είχαμε ρυθμίσει να αξιοποιούμε μία κάρτα γραφικών που προσφέρεται από το εργαλείο Google Colab. Αυτό όμως, δεν θα είναι πάντα διαθέσιμος τρόπος εκπαίδευσης, επομένως υπάρχουν περιπτώσεις που το μοντέλο μας δεν θα μπορεί να εκπαιδευτεί λόγο μειωμένης ικανότητας του υπολογιστή. Ένας τρόπος να το αντιμετωπίσουμε αυτό είναι με την δημιουργία των ομάδων από δεδομένα (**batches of data**), δηλαδή στην προκειμένη περίπτωση, ομάδες από εικόνες. Ένας πολύ συνηθισμένος αριθμός από στοιχεία στις ομάδες των δεδομένων, είναι ο αριθμός 32. Δηλαδή, αν τον εφαρμόσουμε και εμείς, το μοντέλο μας θα φορτώνει στην μνήμη του 32 εικόνες την φορά και όχι όλες μαζί ταυτόχρονα. [7] [8] [1]

Έχοντας αυτό στο νου δίνουμε την παραμετροποίηση του κώδικα ως εξής:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Preprocess data (get all of the pixel values between 1 and 0, also
called scaling/normalization)
train_datagen = ImageDataGenerator(rescale=1/255.)
test_datagen = ImageDataGenerator(rescale=1/255.)
validation_datagen = ImageDataGenerator(rescale=1/255.)

# Setup the train and test directories
train_dir = "output/train/"
test_dir = "output/test/"
validation_dir = "output/val/"

# Import data from directories and turn it into batches
train_data_rescaled = train_datagen.flow_from_directory(train_dir,
                                                        batch_size=32,
                                                        target_size=(224, 224),
                                                        class_mode="binary",
                                                        shuffle=True)

test_data_rescaled = test_datagen.flow_from_directory(test_dir,
                                                      batch_size=32,
                                                      target_size=(224, 224),
                                                      class_mode='binary',
                                                      shuffle=True)

validation_data_rescaled =
test_datagen.flow_from_directory(validation_dir,
```

```
                                                batch_size=32,
                                                target_size=(224, 224),
                                                class_mode='binary',
                                                shuffle=True)

# Train the model
# Import libraries
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense,
Normalization

# Create a CNN model (same as Tiny VGG but for binary classification -
https://poloclub.github.io/cnn-explainer/ )
model_1_3 = Sequential([
  Normalization(axis=None),
  Conv2D(filters=10, kernel_size=3, activation='relu',
input_shape=(224, 224, 3)),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Flatten(),
  Dense(1, activation='sigmoid')
])

# Compile the model
model_1_3.compile(loss="binary_crossentropy",
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])

# Fit the model
history_4 = model_1_3.fit(train_data_rescaled,
                        epochs=5,
                        steps_per_epoch=len(train_data_rescaled),
                        validation_data=validation_data_rescaled,
                        validation_steps=len(validation_data_rescaled)
                        )
```

Μπορούμε να δούμε τις επιδόσεις του μοντέλου μας στην ακρίβεια του, με τον εξής κώδικας:

```
# Evaluate the model_1_3
model_1_3.evaluate(test_data_rescaled)
```

Όπου μας βγάζει για το σύνολο αξιολόγησης:

```
13/13 [==============================] - 2s 141ms/step - loss: 0.3934 - accuracy: 0.8425
[0.39336055517196655, 0.8424999713897705]
```

Όπου το μοντέλο εκπαιδεύεται με 32 εικόνες την φορά φορτωμένες στην μνήμη του. Το οποίο το κάνει πιο παραγωγικό από άποψη χρόνου εκπαίδευσης, και μνήμης. [7]

Σε επόμενη ενότητα, θα προσπαθήσουμε να διορθώσουμε το πρόβλημα της υπερπροσαρμογής με την επαύξηση δεδομένων.

4.2.6 Επαύξηση Δεδομένων (Data Augmentation)

Η επαύξηση των δεδομένων είναι μία τεχνική για την δημιουργία νέων δειγμάτων από τα υπάρχοντα, έτσι ώστε να ξεπεραστούν περιορισμοί των μικρών συνόλων δεδομένων και να αποφευχθεί η υπερβολική προσαρμογή του μοντέλου (overfitting) με τα δεδομένα εκπαίδευσης. Χρησιμοποιείται ευρέως τα τελευταία χρόνια, και ειδικότερα στην βαθιά μάθηση, όπως και κατά συνέπεια στην μηχανική όραση. [7]

Αυτή η τεχνική περιλαμβάνει την εφαρμογή ενός συνόλου μετασχηματισμών στα αρχικά δεδομένα, όπως η περιστροφή, κλιμάκωση, περικοπή και αναστροφή, για την δημιουργία νέων παραλλαγών των εικόνων ή άλλων τύπων δεδομένων. [7]

Θα ξεκινήσουμε δίνοντας τον κώδικα δημιουργίας επαυξημένων δεδομένων και στην συνέχεια θα προσπαθήσουμε να εμφανίσουμε αυτές τις επαυξημένες εικόνες, πριν εκπαιδεύσουμε το μοντέλο μας με αυτές. Δίνεται ο κώδικας προετοιμασίας των επαυξημένων εικόνων:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Preprocess data (get all of the pixel values between 1 and 0, also
called scaling/normalization)
# Add rotation, zoom_range, width/height shift and horizontal flip to
the images.
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=0.5,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)

# Setup the train and test directories
train_dir = "output/train/"
test_dir = "output/test/"
validation_dir = "output/val/"
```

```
# Import data from directories and turn it into batches
train_data_rescaled = train_datagen.flow_from_directory(train_dir,
                                                        batch_size=32,
                                                        target_size=(224, 224),
                                                        class_mode="binary",
                                                        shuffle=True)

test_data_rescaled = test_datagen.flow_from_directory(test_dir,
                                                      batch_size=32,
                                                      target_size=(224, 224),
                                                      class_mode='binary',
                                                      shuffle=True)

validation_data_rescaled =
test_datagen.flow_from_directory(validation_dir,
                                                      batch_size=32,
                                                      target_size=(224, 224),
                                                      class_mode='binary',
                                                      shuffle=True)
```

Ο κώδικας αυτός, δημιουργεί τα σύνολα δεδομένων μας με τις αντίστοιχες ιδιότητες της μεγέθυνσης, περιστροφής, μετατόπισης και αναστροφής των εικόνων σε ποσοστό του 20%. θα προσπαθήσουμε να εμφανίσουμε την πρώτη ομάδα εικόνων, δηλαδή 31 εικόνες, όπως ακριβώς θέσαμε το **batch_size** προηγουμένως.

```
# get the first batches of augmented images
images, labels = next(train_data_rescaled)

# Create the subplots of the images
fig, axes = plt.subplots(4, 8, figsize=(20, 10))
axes = axes.ravel()

for i in range(32):
    axes[i].imshow(images[i])
    axes[i].axis('off')
    axes[i].set_title(f'Image: {i}\nLabel: {labels[i]}')

plt.subplots_adjust(hspace=0.5)
plt.show()
```

Αυτός ο κώδικας μας δίνει για έξοδο:

Εικόνα 24: Έξοδος του κώδικα εμφάνισης των εικόνων της πρώτης ομάδας εικόνων, μετά από επαύξηση των εικόνων αυτών.

Παρατηρούμε πως οι εικόνες φαίνονται επεξεργασμένες στις άκρες τους. Αυτό συμβαίνει λόγο της διαδικασίας της επαύξησης αυτών. Όπως είναι γνωστό, αυτή η διαδικασία επιτρέπει τις συνθήκες στο να καταλάβει το μοντέλο μας τα δεδομένα με λιγότερες πιθανότητες να υπερπροσαρμοστεί σε αυτά. [7]

Συνεχίζουμε με την εκπαίδευση του μοντέλου μας, πάνω σε δεδομένα τα οποία έχουν υποστεί και επαύξηση αλλά και όλα τα προηγούμενα στάδια στα οποία αναφερθήκαμε.

ξεκινάμε δίνοντας τον κώδικα:

```
# Train the model
# Import libraries
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense,
Normalization

# Create a CNN model (same as Tiny VGG but for binary classification -
https://poloclub.github.io/cnn-explainer/ )
model_1_4 = Sequential([
  Normalization(axis=None),
  Conv2D(filters=10, kernel_size=3, activation='relu',
input_shape=(224, 224, 3)),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  Conv2D(filters=10, kernel_size=3, activation='relu'),
  MaxPool2D(),
  Flatten(),
```

```
  Dense(1, activation='sigmoid')
])

# Compile the model
model_1_4.compile(loss="binary_crossentropy",
                  optimizer=tf.keras.optimizers.Adam(),
                  metrics=["accuracy"])

# Fit the model
history_5 = model_1_4.fit(train_data_augmented,
                          epochs=7,
                          steps_per_epoch=len(train_data_augmented),
                          validation_data=validation_data,
                          validation_steps=len(validation_data)
                          )
```

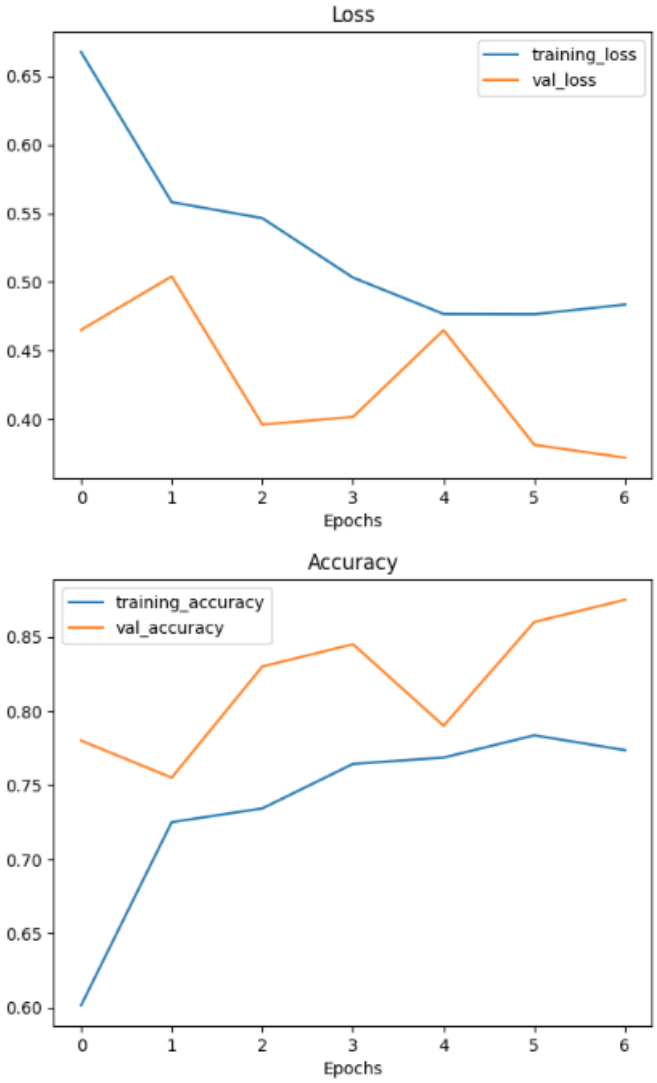
Το οποίο μας δίνει για έξοδο:

```
Epoch 1/7
44/44 [==============================] - 24s 493ms/step - loss: 0.6673
- accuracy: 0.6014 - val_loss: 0.4653 - val_accuracy: 0.7800
Epoch 2/7
44/44 [==============================] - 22s 494ms/step - loss: 0.5582
- accuracy: 0.7250 - val_loss: 0.5041 - val_accuracy: 0.7550
Epoch 3/7
44/44 [==============================] - 22s 495ms/step - loss: 0.5465
- accuracy: 0.7343 - val_loss: 0.3964 - val_accuracy: 0.8300
Epoch 4/7
44/44 [==============================] - 22s 490ms/step - loss: 0.5033
- accuracy: 0.7643 - val_loss: 0.4020 - val_accuracy: 0.8450
Epoch 5/7
44/44 [==============================] - 22s 494ms/step - loss: 0.4768
- accuracy: 0.7686 - val_loss: 0.4651 - val_accuracy: 0.7900
Epoch 6/7
44/44 [==============================] - 22s 511ms/step - loss: 0.4767
- accuracy: 0.7836 - val_loss: 0.3817 - val_accuracy: 0.8600
Epoch 7/7
44/44 [==============================] - 21s 478ms/step - loss: 0.4837
- accuracy: 0.7736 - val_loss: 0.3724 - val_accuracy: 0.8750
```

Φτάνοντας μέχρι στιγμής στα υψηλότερα ποσοστά που έχουμε δει. Ο κώδικας για την εκτύπωση των επιδόσεών του κατά την εκπαίδευση μας επιστρέφει:

```
# Plot Accuracy, loss
plot_loss_curves(history_5)
```

Με έξοδο:



Εικόνα 25: Γράφημα της εκπαίδευσής του μοντέλου model_1_4 μετά από επαύξηση των εικόνων του.

Στο οποίο παρατηρούμε πολύ καλές μετρήσεις, και το φαινόμενο της υπερπροσαρμογής φαίνεται να μην ξεχωρίζει. Παρατηρήστε επίσης, πως το εκπαιδεύσαμε για 7 επαναλήψεις (**epochs**) μίας και εφόσον εφαρμόσαμε τεχνική η οποία μειώνει την πολυπλοκότητα του αλγορίθμου για να αφαιρέσουμε την εμφάνιση της υπερπροσαρμογής, χρειάζεται να το εκπαιδεύσουμε λίγο παραπάνω για να καλύψει το κενό που δημιουργείται. Θα δώσουμε και τον κώδικα εμφάνισης της ακρίβειας του μοντέλου στο σύνολο αξιολόγησης:

```
# Evaluate the model_1_4
model_1_4.evaluate(test_data)
```

Με έξοδο:

```
13/13 [==============================] - 2s 136ms/step - loss: 0.3944 - accuracy: 0.8450
```

[0.3943915069103241, 0.8450000286102295]

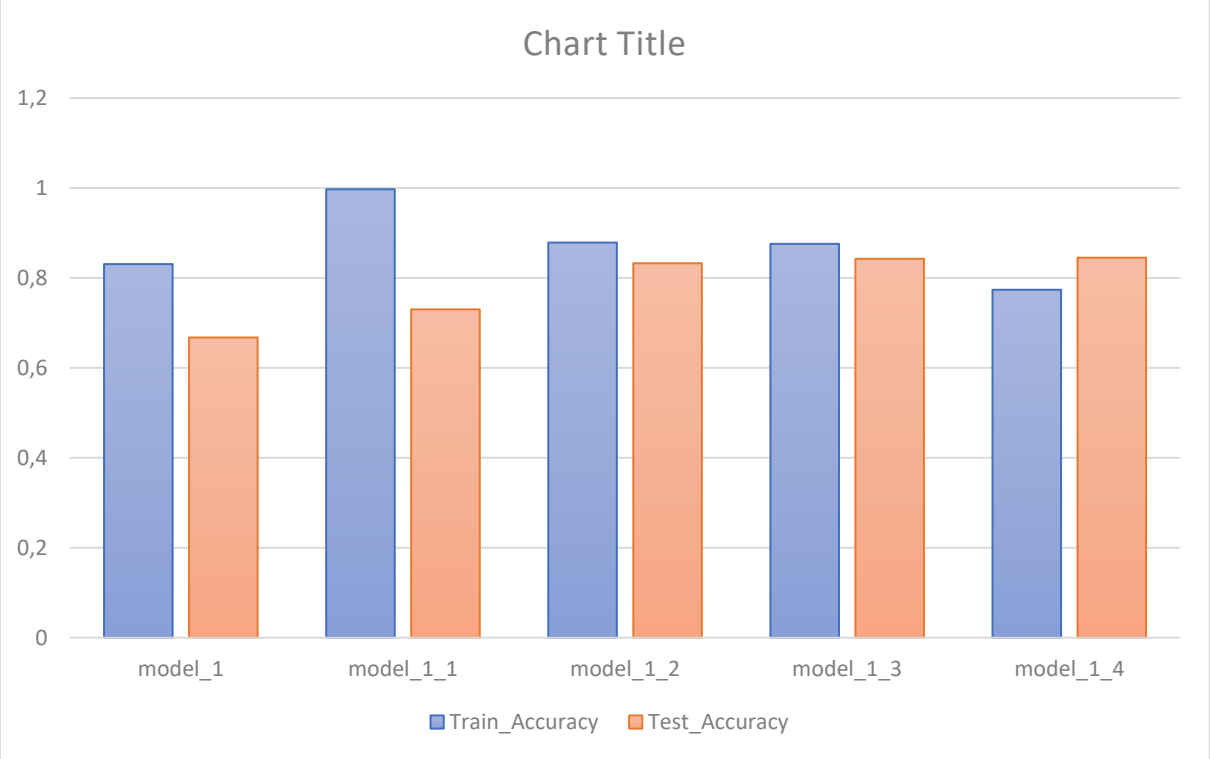
Το οποίο, μας φανερώνει πως το μοντέλο μας, σε εικόνες που δεν έχει ξανά έρθει σε επαφή, φτάνει στο 85% σωστής πρόβλεψης.

4.3 Συμπεράσματα και Στατιστικά

Σε αυτή την ενότητα θα προσπαθήσουμε να καταλάβουμε τα αποτελέσματα των προσπαθειών μας. Σίγουρα αντιλαμβανόμαστε, ότι πλέον η εκπαίδευση ενός μοντέλου δεν ακολουθεί μία συγκεκριμένη μεθοδολογία, η οποία θεωρείται σωστή ή λάθος. Αυτό συμβαίνει διότι, όπως είδαμε, το μοντέλο μας και χωρίς την επαύξηση των εικόνων που του δώσαμε να εκπαιδευτεί πάνω σε αυτές, είχε ήδη από πριν μία καλή επίδοση μόνο με την κανονικοποίηση και επανακλιμάκωση των εικόνων.

Εισάγοντας έννοιες όπως υψηλή μεροληψία και υψηλή διακύμανση, που σημαίνουν ότι το μοντέλο μας υπολειτουργεί ή έχει υπερπροσαρμοστεί στα δεδομένα εκπαίδευσής μας, μπορούν να μας καθοδηγήσουν, μετά από αλλεπάλληλες διαφορετικές προσεγγίσεις και παραμετροποιήσεις των μοντέλων μας, σε ένα μοντέλο που «λειτουργεί» καλά για ένα συγκεκριμένο πρόβλημα. Μέσα από τον πειραματισμό και τις διαφορετικές μεθόδους επεξεργασίας των δεδομένων, ανακαλύψαμε πως η εκπαίδευση σε ανακατεμένα δεδομένα, όπως κάναμε σε προηγούμενη ενότητα, καθώς και η επανακλιμάκωση των δεδομένων, έχουν την κυρίαρχη επιρροή στις αποδόσεις των μοντέλων μας. Χωρίς αυτό να σημαίνει παράλληλα πως, μεθοδολογίες όπως η επαύξηση των εικόνων και της ομαδοποίησης των εικόνων κατά την εκπαίδευση του μοντέλου μας, πρέπει να λαμβάνονται αμελητέες.

Κάθε πρόβλημα, αντιμετωπίζει διαφορετικά δεδομένα, επομένως υπάρχουν εργαλεία και μεθοδολογίες για διαφορετικά προβλήματα που μπορούμε να έρθουμε σε μία εκπαίδευση ενός μοντέλου στην μηχανική όραση, αλλά και στα νευρωνικά δίκτυα γενικότερα. Παρακάτω δίνονται τα στατιστικά για κάθε μοντέλο που εκπαιδεύσαμε, έπειτα θα δοθεί ο κώδικας για όλα όσα φτιάξαμε και τέλος θα δοθούν τα βάρη του μοντέλου model_1_4 το οποίο θεωρείται κατά την άποψή μας από το αποδοτικότερο.



Εικόνα 26: Γράφημα που αναπαριστά τις διαφορετικές επιδόσεις κάθε μοντέλου που δημιουργήθηκε.

Καταλαβαίνουμε πως με κάθε αλλαγή που θέσαμε στα μοντέλα μας, η ακρίβεια στο σύνολο εκπαίδευσης μειωνόταν και ερχόταν στα ίδια ύψη με την ακρίβεια που έχει στο σύνολο αξιολόγησης. Αυτό γενικότερα είναι καλό σημάδι, ότι το μοντέλο μας δεν έχει υπερπροσαρμοστεί (overfitting) και ότι δεν έχει ούτε υπό προσαρμοστεί (underfitting) στα δεδομένα εκπαίδευσης. Κατά συνέπεια, έχουμε βρει μία ισορροπία στην πολυπλοκότητα και τις επιδόσεις του μοντέλου μας, όπως δείξαμε στο κεφάλαιο που εξηγούμε το φαινόμενο της υπερπροσαρμογής.

Γενικότερα, αυτός είναι ο στόχος σε μία εκπαίδευση μοντέλου, όχι μόνο στην μηχανική όραση, αλλά γενικότερα στην επιστήμη των δεδομένων. Έχοντας ένα μοντέλο που αρχικά δεν έχει τις επιδόσεις που αναζητάμε, ή εμφανίζει το φαινόμενο της υπερ-προσαρμογής, τότε με πειραματικές τεχνικές αλλαγής της μορφής των δεδομένων εκπαίδευσης (pre process) μπορούμε να αντιμετωπίσουμε παρόμοια φαινόμενα σε μία άλλη εκπαίδευση μοντέλου μηχανικής όρασης.

Έχοντας αυτό στο νου, αποδεχόμαστε την καλή επίδοση του μοντέλου model_1_4 το οποίο δεν αφήνει περιθώρια υπερπροσαρμογής, και με πειραματισμό στις ρυθμίσεις της αρχιτεκτονικής του μπορούμε να βελτιώσουμε ακόμα περισσότερο τις επιδόσεις του. Κατά συνέχεια, η πτυχιακή εργασία ήρθε στο τέλος της, εφόσον έδειξε τα βήματα και τις δυσκολίες

που συναντάει κανείς σε μία εκπαίδευση ενός μοντέλου μηχανικής όρασης, και στην συνέχεια δοθεί ο κώδικας που γράφτηκε κατά την διάρκεια της εκπόνησής της.

4.4 Παράδοση κώδικα και βάρη Τελικού Μοντέλου

https://colab.research.google.com/drive/1NmppKJmSPnRi5PY0Gs5su6x_jGQS3lJe

Βιβλιογραφία

[1] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[2] C. S. Johannes Magenheim, Encyclopedia of Education and Information Technologies, Springer, 2020.

[3] M. Kaufmann, In The Morgan Kaufmann Series in Data Management Systems, Data Mining (Third Edition), J. Han, M. Kamber and J. Pei, Eds., 2012.

[4] P. S. Foundation, The Python Tutorial, 2022.

[5] S. S. Stevens, "On the Theory of Scales of Measurement," *Science, New Series*, pp. 677-680, 2009.

[6] N. B. J. P. Nithin Buduma, Fundamentals of Deep Learning, 2nd Edition, O'Reilly Media, Inc., 2022.

[7] E. Bisong, Building Machine Learning and Deep Learning Models on Google Cloud Platform, Apress Berkeley, CA, 2019.

[8] G. R. K. Kolla Bhanu Prakash, Programming with TensorFlow, Springer Cham, 2021.

[9] Z. J. W. a. R. T. a. O. S. a. H. P. a. N. D. a. F. H. a. M. K. a. D. H. P. Chau, "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization," *Transactions on Visualization and Computer Graphics*, vol. 27, pp. 1396--1406, 2021.