ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΚΑΤΕΥΘΥΝΣΗ ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ ΚΑΙ ΑΝΑΛΥΤΙΚΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάλυση CT-Scans με χρήση Συνελικτικών Νευρωνικών Δικτύων»

Μαρία Μάστορα

Επιβλέπων Καθηγητής: Μαγκλογιάννης Ηλίας

ΠΕΙΡΑΙΑΣ 2023

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου, Μαγκλογιάννη Ηλία, για την εμπιστοσύνη που μου έδειξε και τη πολύτιμη βοήθεια που μου παρείχε κατά τη διάρκεια της διπλωματικής μου εργασίας. Θέλω να ευχαριστήσω επίσης, την οικογένειά μου, και τους κοντινούς μου ανθρώπους για την κατανόηση και τη στήριξη που μου παρείχαν κατά την διάρκεια των σπουδών μου, με τον καλύτερο δυνατό τρόπο.

Περίληψη

Στο διάστημα έπειτα από την εμφάνιση και τη συνεχή εξάπλωση της ασθένειας COVID-19, πολλά κρούσματα και πολλοί θάνατοι εμφανίζονται παγκοσμίως σε διαφορετική κλίμακα σε κάθε χώρα ξεχωριστά. Λόγω αυτού του γεγονότος, γίνονται πολυάριθμα PCR tests αλλά και λήψεις ιατρικών εικόνων (X-Rays και CT-Scans) καθημερινά. Αξιοσημείωτο είναι το γεγονός ότι τα συστήματα υγειονομικής περίθαλψης είναι παγκοσμίως υπερφορτωμένα, προκαλώντας την νόσηση των εργαζομένων στα συστήματα αυτά ακόμη και με την τήρηση των απαραίτητων μέτρων ασφαλείας. Ιδιαίτερα χρήσιμη, θα ήταν η διάγνωση μέσω των CT-Scans γρηγορότερα και ακριβέστερα μιας και έχει παρατηρηθεί ότι η μέθοδος διάγνωσης με PCR δεν έχει ιδιαίτερα υψηλή ευαισθησία σε ήπιας μορφής μόλυνση από COVID-19.

Δεδομένου του γεγονότος ότι η ανάγνωση των CT-Scans από ειδικούς ακτινολόγους απαιτεί κάποιο υπολογίσιμο χρονικό διάστημα, θα ήταν ιδιαίτερα χρήσιμη η ανάπτυξη κάποιου αυτοματοποιημένου συστήματος τεχνητής νοημοσύνης όπου θα λειτουργούσε γρηγορότερα, και με υψηλή ακρίβεια με σκοπό την αποσυμπίεση του φόρτου εργασίας των ειδικών. Αποτέλεσμα αυτού θα είναι η αποσυμφόρηση των ειδικών αλλά και των υγειονομικών κέντρων και ο καλύτερος έλεγχος των περιστατικών COVID-19. Κατά συνέπεια, με αυτό τον τρόπο θα εφαρμόζεται καλύτερη οργάνωση ως προς τα κρίσιμα περιστατικά που χρίζουν άμεση περίθαλψη σε μονάδα εντατικής θεραπείας, αλλά και τα ήπια περιστατικά όπου αρκεί η ανάρρωση στο σπίτι. Στη παρούσα εργασία παρουσιάζονται και αναλύονται Deep Learning αλγόριθμοι, χρησιμοποιώντας τις Αξονικές Τομογραφίες (CT-Scans) και με τη βοήθεια αυτών προκύπτουν σημαντικά αποτελέσματα με υψηλή ακρίβεια.

Η παρούσα εργασία πραγματεύεται την ανάλυση Αξονικών Τομογραφιών για τη κατηγοριοποίηση των περιστατικών σε ασθενείς και μη-ασθενείς από COVID-19, δημιουργώντας έτσι ένα αυτοματοποιημένο σύστημα όπου μπορεί να οδηγήσει σε αποσυμφόρηση των συστημάτων υγειονομικής περίθαλψης και αποδοτικότερη οργάνωση των περιστατικών. Στην παρούσα εργασία, κατασκευάστηκαν 3D Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) και εκπαιδεύτηκαν με κατάλληλο τρόπο αφού πρώτα εφαρμόστηκε η προεπεξεργασία των εικόνων χρησιμοποιώντας πλήθος τεχνικών. Παρουσιάζονται χρήσιμα αποτελέσματα, συμπεράσματα καθώς επίσης και γραφικές παραστάσεις αλλά και εικόνες χρήσιμες για την κατανόηση του προβλήματος και τη σύγκριση των αποτελεσμάτων.

Abstract

In the period following the emergence and continuous spread of the COVID-19 disease, numerous cases and many deaths have been reported worldwide, each with varying degrees of impact in different countries. As a result of this situation, numerous PCR tests and medical image acquisitions (including CXRs and CT-Scans) are performed daily. Remarkably, healthcare systems globally are overwhelmed, leading to the infection of healthcare workers, even with the implementation of necessary safety measures. It would be particularly valuable to diagnose COVID-19 more quickly and accurately through CT-Scans, as it has been observed that the PCR diagnostic method lacks high sensitivity for mild COVID-19 infections.

Given that the interpretation of CT-Scans by specialized radiologists requires a significant amount of time, the development of an automated artificial intelligence system that operates more rapidly and with high accuracy would be extremely beneficial. The outcome of this development would be to alleviate the workload of specialists. Consequently, this would relieve the burden on both specialists and healthcare facilities and enable better control of COVID-19 cases. As a result, improved organization can be achieved, particularly in critical cases requiring immediate intensive care unit admission, as well as in mild cases suitable for home recovery.

This thesis statement presents and analyzes various types of Deep Learning algorithms, using axial tomography (CT-Scans), and demonstrates significant results with high accuracy. The current study focuses on the analysis of axial tomography images to categorize cases into COVID-19 patients and non-patients, thereby creating an automated system that could lead to the decongestion of healthcare systems and more efficient case management. In this work, different 3D Convolutional Neural Networks (CNNs) were constructed and appropriately trained following suitable image preprocessing techniques. Valuable results, conclusions, as well as graphical representations and images that aid in understanding the problem and comparing the results are presented.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1: Εισαγωγή

1.1 Ανάλυση του Προβλήματος

Ο SARS-CoV-2 έχει μολύνει συνολικά 273 εκατομμύρια και έχει προκαλέσει θάνατο σε 5,34 εκατομμύρια ανθρώπους συνολικά σε όλο τον κόσμο (τη στιγμή της συγγραφής της παρούσας διατριβής) και η μόλυνση αυτή συνεχίζει να επεκτείνεται. Είναι σημαντικό ο ιός αυτός να εντοπίζεται όσο το δυνατόν ταχύτερα και με ακρίβεια για τον έλεγχο της εξάπλωσης της νόσου και τη θεραπεία των ασθενών. Παρόλο που η αλυσιδωτή αντίδραση πολυμεράσης αντίστροφης μεταγραφής (RT-PCR) είναι βασικός τρόπος διάγνωσης του COVID-19, η ευαισθησία της RT-PCR δεν είναι αρκετά υψηλή για χαμηλό ιικό φορτίο που υπάρχει σε δείγματα δοκιμής [1]. Επιπλέον, η προμήθεια RT-PCR ποικίλλει από τόπο σε τόπο και σε ορισμένες αναπτυσσόμενες χώρες υπάρχει έλλειψη.

Κατά τη διάρκεια της πανδημίας COVID-19 στις περισσότερες χώρες προκλήθηκε υπερφόρτωση στα συστήματα υγειονομικής περίθαλψης, καθώς και στα ακτινολογικά τμήματα. Τη κατάσταση αυτή θα βοηθούσε η προσεκτική και ακριβώς υπολογισμένη χρήση οικονομικών και ανθρώπινων πόρων. Παρατηρήθηκε επίσης, πως τα προληπτικά μέτρα που εφαρμόστηκαν στις ιατρικές εγκαταστάσεις δεν ήταν αρκετά για την αποφυγή θανάτων των εργαζομένων. Η Αξονική Τομογραφία (Computed Tomography) θεωρείται βασικό εργαλείο για τη διάγνωση αλλά και αξιολόγηση της εξέλιξης του COVID-19. Με βάση την αξιολόγηση της εξέλιξης της νόσου καθορίζεται εάν ο ασθενής πρέπει να εισαχθεί σε μονάδα εντατικής θεραπείας ή να αναρρώσει στο σπίτι του. Η ολοένα αυξανόμενη χρήση της Αξονικής Τομογραφίας οδηγεί σε επιβάρυνση στο σύστημα υγείας. Για παράδειγμα, στη Μόσχα, στα δημοτικά και στα εξωτερικά κέντρα Αξονικής Τομογραφίας εκτελούνται περίπου 90 μελέτες ανά 1 αξονικό τομογράφο την ημέρα (το ρεκόρ είναι 163 μελέτες κατά τη διάρκεια μιας ημέρας) [2]. Για την γρηγορότερη διαδικασία διάγνωσης COVID-19 μέσω Αξονικών Τομογραφιών αναπτύχθηκαν Deep Learning μοντέλα με σκοπό την αυτοματοποιημένη διαδικασία διάγνωσης η οποία βελτιώνει την παραγωγικότητα και ελαχιστοποιεί τα λάθη στην κατηγοριοποίηση της σοβαρότητας των πνευμονικών ανωμαλιών.

Η Τεχνητή Νοημοσύνη (Artificial Intelligence) μπορεί να θεωρηθεί κατάλληλη για αυτή την πρόκληση. Με κατάλληλη τροφοδότηση από μεγάλα σύνολα δεδομένων και

εξελιγμένες GPUs, η Τεχνητή Νοημοσύνη καθώς και οι τεχνικές Deep Learning έχουν επιτύχει υψηλές επιδόσεις σε εργασίες υπολογιστικής όρασης (Computer Vision), όπως Image Classification και Object Detection. Πρόσφατη έρευνα [12] έχει δείξει ότι αλγόριθμοι Τεχνητής Νοημοσύνης μπορούν να επιτύχουν ή και να υπερβούν την απόδοση των ειδικών σε ορισμένες εργασίες διάγνωσης ιατρικών εικόνων, συμπεριλαμβανομένων των πνευμονικών ασθενειών. Στην έρευνα [11] παρουσιάζεται περίπτωση όπου δείγμα περιστατικού με καρκίνο του μαστού αγνοήθηκε από 6 ραδιολόγους και αναγνωρίστηκε από τα συστήματα Τεχνητής Νοημοσύνης. Ενώ αντίθετα, δείγμα με καρκίνου του μαστού αγνοήθηκε από τα συστήματα Τεχνητής Νοημοσύνης και αναγνωρίστηκε από τους 6 ραδιολόγους. Συνεπώς, οι ρόλοι των συστημάτων Τεχνητής Νοημοσύνης και των ανθρώπινων αναγνωστών καθίστανται συμπληρωματικοί στην εξαγωγή ακριβών συμπερασμάτων. Επιπλέον, λόγω της ύπαρξης διάφορων πνευμονικών παθήσεων (όπως καρκίνος του πνεύμονα, φυματίωση), η διαφοροποίηση του COVID-19 από αυτές, επειδή υπάρχει μεγάλη ομοιότητα μεταξύ τους, αποτελεί δύσκολο επίτευγμα και η χρήση της Τεχνητής Νοημοσύνης μπορεί να θεωρηθεί απαραίτητη.

Η παρούσα εργασία πραγματεύεται την ανίχνευση της νόσου COVID-19 μέσω Αξονικών Τομογραφιών (CT-scans). Οι εικόνες αυτές οι οποίες είναι σε μορφή NIfTI, φορτώνονται και επεξεργάζονται με τη βοήθεια του TorchIO. Στη συνέχεια δοκιμάζονται διάφορα Deep Learning μοντέλα CNN για 3D εικόνες. Εφαρμόζονται πειραματικές διαδικασίες που περιλαμβάνουν κατασκευή μοντέλου from scratch, καθώς επίσης και pre-trained μοντέλου. Τέλος, παρουσιάζονται διάφορα αποτελέσματα, αλλά και συγκρίσεις μεταξύ των μοντέλων αυτών.

1.2 Δομή της Διπλωματικής Εργασίας

Η δομή της παρούσας διπλωματικής εργασίας αναλύεται στα παρακάτω κεφάλαια.

Το πρώτο Κεφάλαιο, περιλαμβάνει την Εισαγωγή της διπλωματικής εργασίας. Παρουσιάζεται συνοπτικά η ανάλυση του προβλήματος καθώς και η διαχείρισή του, με τις κατάλληλες μεθόδους και τεχνικές.

Το δεύτερο Κεφάλαιο, περιλαμβάνει την σχετική βιβλιογραφία μαζί με τις τεχνικές όπου έχουν χρησιμοποιηθεί. Επιπλέον, παρουσιάζεται και αναλύεται το θεωρητικό υπόβαθρο, το οποίο είναι σχετικό με την παρούσα εργασία.

Στο τρίτο Κεφάλαιο, παρουσιάζεται η μεθοδολογία και οι τεχνικές που εφαρμόστηκαν για το πρόβλημα κατηγοριοποίησης σε ασθενείς και μη-ασθενείς από COVID-19. Αναλύονται τεχνικές που εφαρμόστηκαν κατά την προ-επεξεργασία των εικόνων, όπως επίσης και κάποια χρήσιμα πακέτα που χρησιμοποιήθηκαν, τα οποία βοήθησαν στην κατάλληλη διαχείριση του προβλήματος.

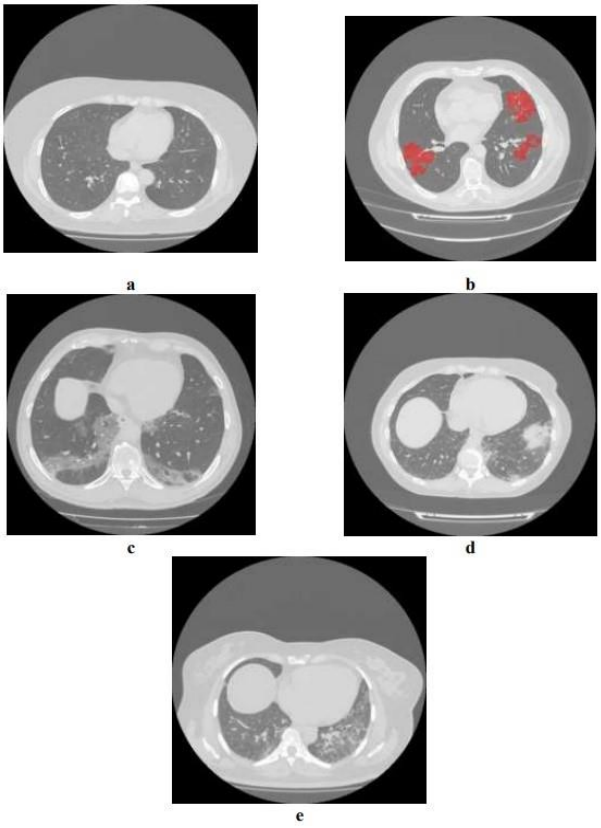
Το τέταρτο Κεφάλαιο, περιλαμβάνει την Περιγραφή του συνόλου δεδομένων, όπως επίσης και την Πειραματική διαδικασία. Σε αυτό το Κεφάλαιο αναλύεται το σύνολο δεδομένων πάνω στο οποίο εφαρμόστηκαν διάφορες επεξεργασίες, δηλαδή οι 3D CT-Scans εικόνες, οι οποίες είναι σε μορφή NIfTI. Παρουσιάζεται η αρχιτεκτονική των Συνελικτικών Νευρωνικών Δικτύων όπου κατασκευάστηκαν. Στη συνέχεια, αναλύονται οι πειραματικές δοκιμές στην εκπαίδευση των Συνελικτικών Νευρωνικών Δικτύων. Επιπρόσθετα, παρουσιάζονται γραφήματα και εικόνες για καλύτερη και απλούστερη κατανόηση των αποτελεσμάτων, εμφανίζοντας μετρικές απόδοσης, όπως Accuracy και Loss.

Στο πέμπτο Κεφάλαιο, παρουσιάζονται τα συμπεράσματα που προέκυψαν κατά το πέρας της εργασίας. Συνοψίζονται οι συγκρίσεις μεταξύ των Πειραματικών διαδικασιών και παρουσιάζονται κάποιες σημαντικές παρατηρήσεις.

Κεφάλαιο 2: Βιβλιογραφία & Θεωρητικό υπόβαθρο

2.1 Βιβλιογραφική Επισκόπηση

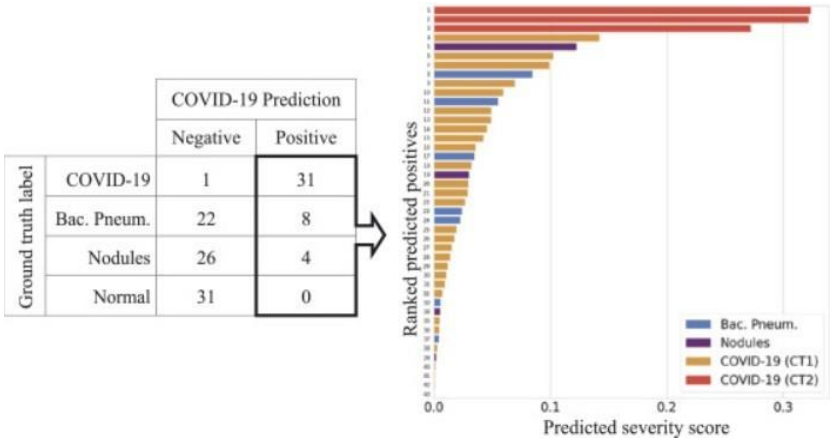
Το άρθρο [2] αφορά το σύνολο δεδομένων MOSMEDDATA, το οποίο περιλαμβάνει Αξονικές Τομογραφίες (CT-Scans) θώρακα σχετιζόμενες με ευρήματα COVID-19, και την αξιολόγηση αυτών με βάση τη σοβαρότητά τους. Οι Αξονικές Τομογραφίες στο σύνολο δεδομένων αποκτήθηκαν στο διάστημα μεταξύ 1ης Μαρτίου του 2020 και 25ης Απριλίου του 2020. Το σύνολο δεδομένων συλλέχθηκε από μία ομάδα ερευνητών από τα δημόσια νοσοκομεία της Μόσχας στη Ρωσία. Συνολικά υπάρχουν 1,110 μελέτες (studies) σε μορφή NIfTI, οι οποίες έχουν κατηγοριοποιηθεί σε 5 κατηγορίες, με βάση τη σοβαρότητα της ασθένειας. Οι κατηγορίες των εικόνων είναι οι ακόλουθες: CT-0: τα περιστατικά δεν εμφανίζουν καμία πνευμονική πάθηση ή COVID-19, CT-1: στα περιστατικά αυτά εμφανίζεται προσβολή πνευμονικού παρεγχύματος σε ποσοστό ≤ 25%, CT-2: στα περιστατικά αυτά εμφανίζεται προσβολή πνευμονικού παρεγχύματος σε ποσοστό 25% - 50%, CT-3: στα περιστατικά αυτά εμφανίζεται προσβολή πνευμονικού παρεγχύματος σε ποσοστό 50% - 75%, CT-4: στα περιστατικά αυτά εμφανίζεται προσβολή πνευμονικού παρεγχύματος σε ποσοστό 75% - 100%. Στην παρακάτω εικόνα [Εικόνα 1] απεικονίζονται 5 Αξονικές Τομογραφίες - μία για κάθε κατηγορία σοβαρότητας. Οι συγγραφείς ελπίζουν ότι το σύνολο δεδομένων αυτό θα βοηθήσει στη βελτίωση της διάγνωσης και της αξιολόγησης της σοβαρότητας του COVID-19 και θα συμβάλει στην ανάπτυξη νέων μεθόδων αναγνώρισης της νόσου. Πάνω στο σύνολο δεδομένων αυτό βασίζεται και η παρούσα έρευνα, και αποτελεί κομμάτι της προσπάθειας αυτής.



Εικόνα 1. Παράδειγμα Εικόνων: a - CT-0, b - CT-1, c - CT-2, d - CT-3, e - CT-4 [2]

Το άρθρο [10] χρησιμοποιεί το σύνολο δεδομένων MOSMEDDATA το οποίο αναλύεται παραπάνω [2]. Βασίζεται στο πρόβλημα διαλογής και χωρίζεται σε δύο σημαντικά προβλήματα. Το πρώτο πρόβλημα είναι ο προσδιορισμός των μελετών (εικόνων) των ασθενών με COVID-19 και η κατάλληλη προτεραιότητα αυτών από τους γιατρούς, ώστε να απομονωθούν οι μολυσμένοι ασθενείς όσο τον δυνατόν γρηγορότερα. Το δεύτερο πρόβλημα σχετίζεται με την ποσοτικοποίηση της σοβαρότητας, δίνοντας προτεραιότητα σε αυτούς που απαιτούν επείγουσα ιατρική περίθαλψη. Στην υλοποίηση που αναφέρεται στο άρθρο αυτό έγινε προ-επεξεργασία εφαρμόζοντας rescaling σε κάθε axial slice σε 2 x 2 mm και κανονικοποίηση της έντασης στο εύρος [0,1]. Σε όλα τα πειράματα όπως φαίνεται στο πρόβλημα identification του COVID-19 αλλά και στο segmentation εφαρμόστηκε περικοπή της μάσκας των πνευμόνων. Οι συγγραφείς για το πρόβλημα του segmentation των πνευμόνων εφάρμοσαν U-Net αρχιτεκτονική με 2D Συνελικτικά Επίπεδα για καθένα από τα axial slices και εκπαίδευσαν το μοντέλο πάνω στα σύνολα δεδομένων NSCLC-Radiomics και LUNA16, τα οποία περιέχουν εικόνες με βακτηριακή πνευμονία, και καρκινικούς όγκους. Στο μο-

ντέλο αυτό χρησιμοποιήθηκε ο Adam optimizer με τις default παραμέτρους και αρχικό learning rate ίσο με 0.001 το οποίο μειώθηκε σε 0.0001 έπειτα από 8K batches (από τα συνολικά 16K batches). Για την βελτίωση της απόδοσης του μοντέλου χρησιμοποιήθηκε επίσης 3 Fold Cross Validation και επιπλέον χρησιμοποιήθηκε το σύνολο δεδομένων MedSeg29 ως hold-out σύνολο. Αυτή η υλοποίηση προσέφερε τιμή Dice Score ίση με 0.976. Στην προτεινόμενη αρχιτεκτονική με ονομασία Multitask-Spatial 1, οι συγγραφείς χρησιμοποιούν ένα ενιαίο two-headed Συνελικτικό Νευρωνικό Δίκτυο για να εκτελούν ταυτόχρονα εργασίες τμηματοποίησης (segmentation) και ταξινόμησης (identification) για τη διαλογή COVID-19 χρησιμοποιώντας Αξονικές Τομογραφίες. Η εργασία τμηματοποίησης (segmentation task) εκτελείται χρησιμοποιώντας ένα 2D μοντέλο U-Net και η έξοδος αυτής της τμηματοποίησης χρησιμοποιείται για την αξιολόγηση της βαθμολογίας σοβαρότητας. Η κεφαλή ταξινόμησης μοιράζεται έναν κοινό ενδιάμεσο χάρτη χαρακτηριστικών (feature map) με το τμήμα τμηματοποίησης. Αυτοί οι χάρτες χαρακτηριστικών (feature maps) στοιβάζονται και συγκεντρώνονται σε ένα διάνυσμα χαρακτηριστικών (feature vector) χρησιμοποιώντας ένα pyramid pooling layer και το προκύπτων διάνυσμα χαρακτηριστικών (feature map) μετατρέπεται σε τιμή πιθανότητας για τον COVID-19 χρησιμοποιώντας δύο Fully Connected Layers έπειτα από την ενεργοποίηση σιγμοειδούς συνάρτησης (sigmoid activation). Συνεπώς, στο Multitask-Spatial 1, ο χάρτης χαρακτηριστικών (feature map) είναι κοινός από το τέλος της αρχιτεκτονικής U-Net και η εργασία αναγνώρισης επωφελείται από τη δομή των χαρακτηριστικών εισόδου. Από τα αποτελέσματα που παρουσιάζουν οι συγγραφείς φαίνεται πως η μέθοδος αυτή του Multitask είναι αυτή που παρουσιάζει και την μεγαλύτερη τιμή ακρίβειας με ROC AUC (0.87). Την μικρότερη τιμή ROC AUC παρουσιάζει το 3D U-Net (0.59). Οι συγγραφείς καταλήγουν στο συμπέρασμα ότι οι αλγόριθμοι ανάλυσης ιατρικών εικόνων που βασίζονται στο Deep Learning, είναι σπουδαίοι βοηθοί ευφυών ακτινολόγων σε μια γρήγορη και αξιόπιστη εκτίμηση χρονοβόρων διαδικασιών όπως η σοβαρότητα του COVID-19. Το κομμάτι της ταξινόμησης του συστήματος διαλογής τους μπορεί να χρησιμοποιηθεί ως εξαιρετικά σημαντικό εργαλείο πρώτης ανάγνωσης. Στην Εικόνα 2 παρουσιάζονται τα αποτελέσματα των προβλέψεων.



Εικόνα 2. Αποτελέσματα πρόβλεψης της [10]

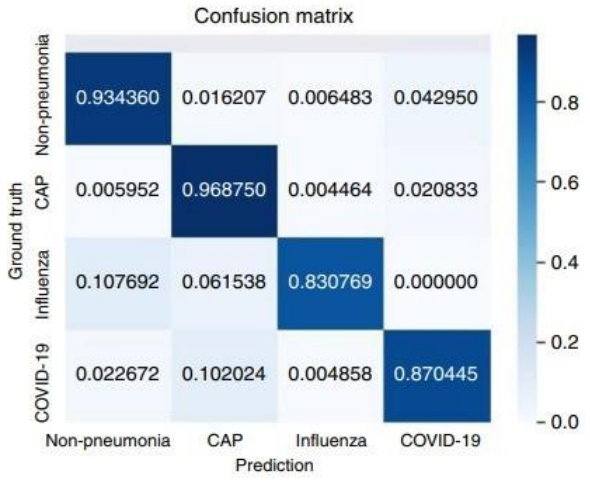
Το άρθρο [19] χρησιμοποιεί ένα σύνολο δεδομένων με 2,482 CT-Scans εικόνες από τις οποίες οι 1,252 εικόνες είναι από ασθενείς που έχουν μολυνθεί από COVID και οι 1,230 είναι από μη μολυσμένους ασθενείς από COVID-19, αλλά οι οποίοι παρουσιάζουν άλλες πνευμονικές παθήσεις. Τα δεδομένα συλλέχθηκαν από νοσοκομεία του Sao Paolo της Βραζιλίας. Το άρθρο αυτό στηρίζεται στην εφαρμογή του xDNN (Explanable Deep Neural Network). Στα αποτελέσματα φανερώνονται τα ποσοστά ακρίβειας ανά μέθοδο. Συγκεκριμένα, η xDNN παρουσιάζει accuracy ίσο με 97.38%, το ResNet 94.96%, το GoogleNet 91.73%, το VGG-16 94.96%, το AlexNet 93.75, το Decision Tree 79.44, το Ada Boost 95.16%. Τα πλεονεκτήματα της μεθόδου του xDNN είναι: Η υψηλή ακρίβεια συγκριτικά με τους υπόλοιπους αλγορίθμους που εφαρμόστηκαν, το υψηλό επίπεδο επεξήγησης και ο μη επαναληπτικός αλγόριθμος με συνεχή εκμάθηση. Στην Εικόνα 3 απεικονίζονται οι μετρικές απόδοσης των πειραματικών διαδικασιών.

Method \ Metric	Accuracy	Precision	Recall	F1 Score	AUC
xDNN	97.38%	99.16%	95.53%	97.31%	97.36%
ResNet	94.96%	93.00%	97.15%	95.03%	94.98%
GoogleNet	91.73%	90.20%	93.50%	91.82%	91.79%
VGG-16	94.96%	94.02%	95.43%	94.97%	94.96%
AlexNet	93.75%	94.98%	92.28%	93.61%	93.68%
Decision Tree	79.44%	76.81%	83.13%	79.84%	79.51%
AdaBoost	95.16%	93.63%	96.71%	95.14%	95.19%

Εικόνα 3. Αποτελέσματα μετρικών των προβλέψεων της [19]

Η [20] πραγματεύεται την ανίχνευση ασθενειών των πνευμόνων και συγκεκριμένα της γρίπης, του COVID-19, της λοίμωξης των πνευμόνων. Για τους σκοπούς της υλοποίησης συλλέχθηκαν δεδομένα από διάφορες πηγές. Τα σύνολα δεδομένων από τα οποία συλλέχθηκαν τα δεδομένα είναι τα ακόλουθα: LIDC–IDRI, Tianchi-Alibaba,

MosMedData33, και CC-CCII. Το σύνολο πάνω στο οποίο εφαρμόστηκαν τα μοντέλα της έρευνας προέκυψε από το συνονθύλευμα πολλών συνόλων δεδομένων. Αποτελείται από συνολικά 11,356 CT-Scans από 9,025 περιστατικά (subjects) και περιλαμβάνει τις ασθένειες του COVID19, της λοίμωξης των πνευμόνων, της γρίπης A-B, και της μη-πνευμονίας. Η μετρική AUC για τις 4 κατηγορίες βρέθηκε 0.9752 (για την μη-πνευμονία), 0.9804 (για την λοίμωξη των πνευμόνων), 0.9885 (για την γρίπη), και 0.9745 (για τον COVID-19). Οι συγγραφείς αναφέρονται στο γεγονός ότι η υλοποίηση εφαρμόστηκε σε 2D slices, για τον λόγο του ότι είναι ευκολότερο η εκπαίδευση να γίνει εντός του ορίου μνήμης για κοινές GPU (συνήθως 11 GB) αλλά και επειδή η διάγνωση στις 2D τομές μπορεί να χρησιμοποιηθεί για τον εντοπισμό μολυσματικών τμημάτων COVID-19. Συνοψίζοντας, λόγω της συλλογής πολλών δεδομένων από διαφορετικές πηγές, επιτεύχθηκε η υψηλή ακρίβεια των μοντέλων. Στην Εικόνα 4 παρουσιάζεται ο confusion matrix των προβλέψεων για καθεμία κατηγορία.



Εικόνα 4. Confusion Matrix αποτελεσμάτων των προβλέψεων της [20]

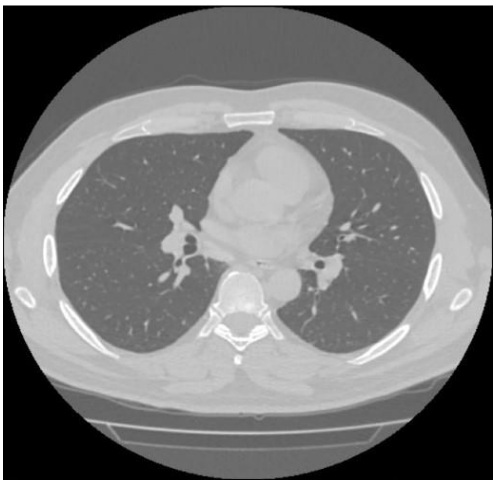
2.2 Θεωρητικό Υπόβαθρο

2.2.1 Φυσική στις Αξονικές Τομογραφίες CT-Scans

Η Αξονική Τομογραφία (CT-Scan) χρησιμοποιεί δέσμες ακτινών Χ για τη λήψη τρισδιάστατων εντάσεων εικονοστοιχείων του ανθρώπινου σώματος. Μια θερμαινόμενη κάθοδος απελευθερώνει δέσμες υψηλής ενέργειας (ηλεκτρόνια), οι οποίες με τη σειρά τους απελευθερώνουν την ενέργειά τους ως ακτινοβολία ακτινών Χ. Οι ακτίνες Χ περνούν μέσα από τους ιστούς του ανθρώπινου σώματος και χτυπούν έναν ανιχνευτή στην άλλη πλευρά [15].

Για την ανάγνωση μίας Αξονικής Τομογραφίας, λαμβάνονται υπόψη τα χρώματα λευκό, γκρι και μαύρο. Κάθε χρώμα αντιπροσωπεύει ένα ξεχωριστό μέρος του σώμα-

τος: μαλακούς ιστούς, λίπος, αέρας, οστά. Μία αλλαγή στο χρώμα σε μία συγκεκριμένη περιοχή του σώματος μπορεί να υποδηλώνει την παρουσία μίας ”ανωμαλίας”. Ένας πυκνός ιστός (όπως για παράδειγμα τα οστά) θα απορροφήσει περισσότερη ακτινοβολία από τους μαλακούς ιστούς (όπως για παράδειγμα το λίπος). Όταν οι ακτίνες Χ δεν απορροφώνται από το σώμα, δηλαδή στην περίπτωση του λίπους ή του αέρα, και φτάσουν στον ανιχνευτή, απεικονίζονται με σκούρο γκι ή μαύρο χρώμα. Αντίθετα, οι πυκνοί ιστοί, όπως τα οστά, απεικονίζονται με λευκό χρώμα. Οι μαλακοί ιστοί, καθώς και οποιοδήποτε υγρό, συμπεριλαμβανομένου του αίματός, θα εμφανιστούν με διάφορα χρώματα του γκρι. Στην Εικόνα 5 απεικονίζεται μία Αξονική Τομογραφία πνευμόνων όπου μπορεί να αναγνωστεί κατάλληλα από κάποιον ειδικό και με βάση την παραπάνω περιγραφή που έχει προηγηθεί.



Εικόνα 5. Αξονική Τομογραφία (CT-Scan)

2.2.2 Deep Learning

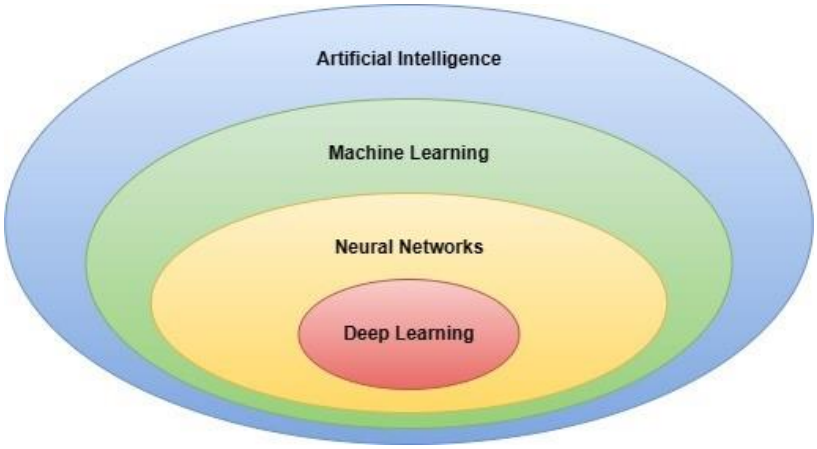
Στην κατηγορία την Τεχνητής Νοημοσύνης (Artificial Intelligence) ανήκουν η Μηχανική Μάθηση (Machine Learning) και το Deep Learning [16]. Η Μηχανική Μάθηση είναι η Τεχνητή Νοημοσύνη που μπορεί να προσαρμοστεί αυτόματα με ελάχιστη ανθρώπινη παρέμβαση. Το Deep Learning είναι ένα υποσύνολο της Μηχανικής Μάθησης που χρησιμοποιεί Τεχνητά Νευρωνικά Δίκτυα με σκοπό να μιμηθεί τη διαδικασία μάθησης του ανθρώπινου εγκεφάλου. Στην Εικόνα 6 παρουσιάζονται τα πεδία Deep Learning, Νευρωνικών Δικτύων, Μηχανικής Μάθησης και Τεχνητής Νοημοσύνης ως το ένα υποσύνολο του άλλου [13].

Το Deep Learning είναι ένα σύνολο αλγορίθμων της μηχανικής μάθησης με πολλαπλά επίπεδα Νευρωνικού Δικτύου. Τα επίπεδα αυτά των εκπαιδευμένων μοντέλων

Βαθιάς Μηχανικής Μάθησης εξάγουν γενικούς κανόνες και έννοιες (abstraction) στους οποίους υπακούν τα δεδομένα [14].

Το Deep Learning ένας τύπος μηχανικής μάθησης με πολυεπίπεδο (multilayered) Νευρωνικό Δίκτυο. Είναι μια από τις μεθόδους μηχανικής μάθησης που αφορά τη σύνθεση δεδομένων σε μια προγνωστική μορφή. Δύο εφαρμογές του Deep Learning είναι η παλινδρόμηση (regression) και η ταξινόμηση (classification). Σε κάθε περίπτωση, υπάρχουν δεδομένα προπόνησης (training data) που χρησιμοποιούνται για την προσαρμογή βαρών (άγνωστες παράμετροι) και ελαχιστοποιούν μια συνάρτηση απώλειας (αντικειμενική συνάρτηση) [17].

Το Deep Learning ανήκει στην οικογένεια της Μηχανικής Μάθησης (Machine Learning) που βασίζονται σε Τεχνητά Νευρωνικά Δίκτυα. Αρχιτεκτονικές Βαθιάς Μηχανικής Μάθησης όπως τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) έχουν εφαρμοστεί σε διάφορα πεδία συμπεριλαμβανομένων της Υπολογιστικής Όρασης (Computer Vision), Αναγνώριση Ομιλίας (Speech Recognition), Φυσική επεξεργασία Γλώσσας (Natural Language Processing), Βιοπληροφορικής (Bioinformatics), Ανάλυση Ιατρικών Εικόνων (Medical Image Analysis) κ.ά. [18].



Εικόνα 6. Από τη Τεχνητή Νοημοσύνη (Artificial Intelligence) στη Βαθιά Μάθηση (Deep Learning)

Το Deep Learning μπορεί να προσδιοριστεί ως υποσύνολο της Μηχανικής Μάθησης. Πιο συγκεκριμένα είναι ένα Νευρωνικό Δίκτυο με τρία ή περισσότερα επίπεδα. Αυτά τα Νευρωνικά Δίκτυα προσπαθούν να ”μοιάσουν” της συμπεριφορά του ανθρώπινου εγκεφάλου αποκτώντας την ικανότητα να ”μαθαίνουν” από μεγάλα σύνολα δεδομένων. Ξεκινώντας από το πρώτο επίπεδο (layer) όπου κάνει μία προσεγγιστική πρόβλεψη, τα επόμενα επίπεδα μπορούν να βοηθήσουν στην βελτιστοποίηση της ακρίβειας [5].

Το Deep Learning πρωτοεμφανίστηκε την δεκαετία του 1980, όμως παρόλα αυτά υπάρχουν δύο βασικοί λόγοι για τους οποίους τα τελευταία χρόνια έγινε πολύ χρήσιμο:

- Απαιτεί μεγάλα ποσά labeled δεδομένων. Για παράδειγμα για τα αυτοκινούμενα αυτοκίνητα απαιτούνται εκατομμύρια εικόνες και χιλιάδες ώρες βίντεο [6].

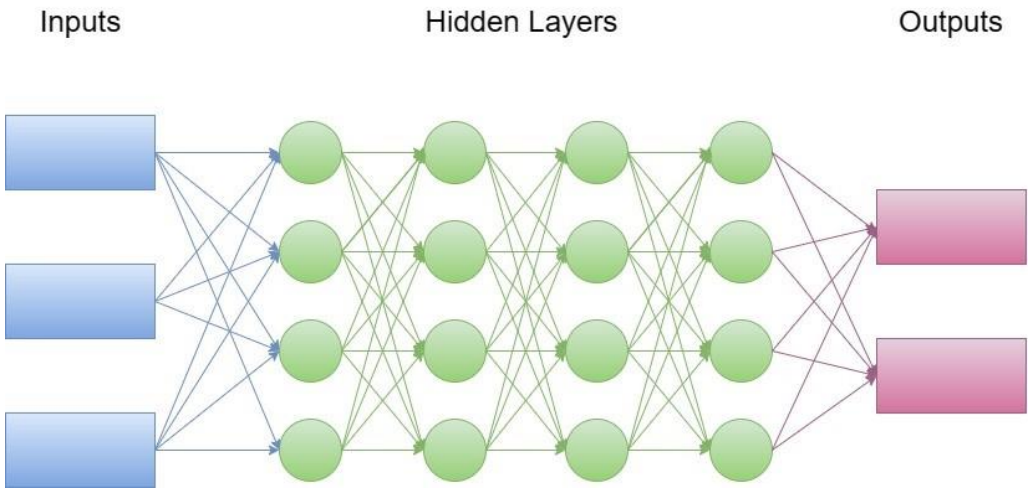
- Απαιτεί μεγάλους όγκους υπολογιστικής ισχύς. Τα GPUs υψηλής απόδοσης περιλαμβάνουν παράλληλες αρχιτεκτονικές οι οποίες κάνουν το Deep Learning χρήσιμο. Αντίστοιχο παράδειγμα αποτελούν τα συστήματα cloud ή τα clusters, με τη βοήθεια των οποίων οι υλοποιήσεις μειώνουν ιδιαίτερα σημαντικά τον χρόνο υπολογισμών από εβδομάδες σε ώρες ή και λιγότερο [6].

Τα δεδομένα εισόδου Νευρωνικού Δικτύου Deep Learning περνούν από πολλαπλά στρώματα επεξεργασίας τεχνητών νευρώνων το ένα μετά το άλλο για να παρέχουν την έξοδο. Στην Εικόνα 7 απεικονίζεται μία πολυ-επίπεδη (multi-layered) απεικόνιση ενός Νευρωνικού Δικτύου, το οποίο αποτελείται από:

- Ένα *Input Layer*. Το πρώτο επίπεδο όπου φορτώνει τα δεδομένα στο μοντέλο και τα διαβιβάζει χωρίς κανέναν υπολογισμό.

- Τα *Hidden Layers*. Πολλαπλά διασυνδεδεμένα επίπεδα που εκτελούν μαθηματικές πράξεις σε δεδομένα και εξάγουν χαρακτηριστικά.

- Τα *Outputs*. Το τελικό επίπεδο που παρέχει το αποτέλεσμα λαμβάνοντας ως εισόδους τα προηγούμενα hidden layers.



Εικόνα 7. Αρχιτεκτονική Νευρωνικού Δικτύου Βαθιάς Μηχανικής Μάθησης

2.2.3 Προκλήσεις στην επεξεργασία των ιατρικών εικόνων για Deep Learning

Αναπτύσσοντας αλγορίθμους Deep Learning στις ιατρικές εικόνες, φαίνεται πως πολλές προκλήσεις μπορούν να υπάρξουν και να προκύψουν στην πορεία. Βασικά ενδιαφέροντα σημεία είναι τα εξής [8]:

(i) Η διαχείριση των metadata που σχετίζονται με τη θέση και το μέγεθος της εικόνας.
(ii) Η έλλειψη των labels από αρκετά υψηλό μέγεθος δεδομένων.
(iii) Το υψηλό υπολογιστικό κόστος λόγω της πολυδιαστατικότητας των δεδομένων.
(iv) Η έλλειψη ομοφωνίας για τη καλύτερη μέθοδο κανονικοποίησης. Τέτοιες προκλήσεις είναι αρκετά συχνές στην επεξεργασία ιατρικών εικόνων και απαιτούν συγκεκριμένα χαρακτηριστικά, όπως το TorchVision.

Παρακάτω αναλύονται τα 4 ενδιαφέροντα σημεία με μεγαλύτερη επεξήγηση:

(i) Τα Metadata όπου κωδικοποιούν το σχήμα της εικόνας, την απόσταση από την οποία αποτυπώθηκε η εικόνα και με βάση την τοποθέτηση της κάμερας, τον προσανατολισμός των voxels, καθορίζουν την χωρική σχέση μεταξύ των voxels. Οι πληροφορίες των metadata μπορούν να δώσουν βαρυσήμαντες πληροφορίες.

(ii) Οι μέθοδοι Deep Learning απαιτούν μεγάλα ποσά δεδομένων, πράγμα το οποίο αποτελεί δύσκολο εγχείρημα λόγω του απόρρητου των ασθενών στα κλινικά κέντρα και στα νοσοκομεία. Στην δυσκολία της συλλογής μεγάλων δεδομένων επίσης επιδρούν οι οικονομικές και χρονικές επιβαρύνσεις, καθώς και η ανάγκη για σχολιασμούς από έμπειρους και υψηλά εκπαιδευμένους ειδικούς. Λύση στο εμπόδιο αυτό έρχεται να φέρει η τεχνική της Επαύξησης Δεδομένων (Data Augmentation). Η τεχνική αυτή αυξάνει το μέγεθος του training dataset με εφαρμογή διάφορων μετασχηματισμών (transformation) πάνω σε κάθε στοιχείο του training dataset, διατηρώντας τους κατάλληλους σχολιασμούς (annotations). Τέτοιες τεχνικές περιλαμβάνουν γεωμετρικούς μετασχηματισμούς όπως τυχαία περιστροφή (random rotation), μεγέθυνση (zoom), αλλά και αλλαγές στους χρωματικούς μετασχηματισμούς. Η τυχαία εναλλαγή καναλιών αποτελεί ένα τέτοιο παράδειγμα. Υπάρχουν όμως και περιπτώσεις όπου δεν εφαρμόζονται χρώματα στην εικόνα, όπως στις ιατρικές εικόνες, επειδή οι χρωματισμοί είναι στην κλίμακα του γκρι (grayscale). Η τεχνική του Data Augmentation συνήθως εφαρμόζεται την

στιγμή της φόρτωσης της εικόνας από τον δίσκο κατά τη διάρκεια της εκπαίδευσης. Επιπρόσθετα, αξίζει να σημειωθεί ότι, οι τεχνικές cropping και scaling είναι πιο δύσκολο να εφαρμοστούν στις ιατρικές εικόνες διότι υπάρχει ο κίνδυνος απώλειας χρήσιμου τμήματος.

(iii) Σε αντίθεση με τις 2D εικόνες οι οποίες χρησιμοποιούνται στο Deep Learning, και όπου σπανίως έχουν πάνω από ένα εκατομμύριο pixels, οι 3D εικόνες συχνά περιέχουν εκατοντάδες εκατομμύρια voxels. Στις περιπτώσεις αυτές η μείωση δειγματοληψίας (Downsampling) μπορεί να μην είναι αποδεκτή όταν θεωρούνται βαρυσήμαντες ακόμη και κάποιες μικρές λεπτομέρειες στην εικόνα. Στις εφαρμογές της υπολογιστικής όρασης, οι εικόνες όπου χρησιμοποιούνται στην εκπαίδευση ομαδοποιούνται σε batches των εκατοντάδων ή χιλιάδων training instances (ανάλογα με τη διαθεσιμότητα μνήμης της GPU). Τα δίκτυα εκπαιδεύονται με 2D slices όπου εξάγονται από τα 3D volumes.

(iv) Transfer Learning και κανονικοποίηση (normalization). Στο Transfer Learning γίνεται προ-εκπαίδευση Νευρωνικού Δικτύου πάνω σε ένα μεγάλο σύνολο δεδομένων με εικόνες. Παράδειγμα τέτοιο δικτύου είναι το ImageNet, το οποίο περιέχει πάνω από 14 εκατομμύρια εικόνες, και μπορεί να εκπαιδευτεί περαιτέρω σε μικρότερο σύνολο δεδομένων.

2.2.4 Παραδείγματα Εφαρμογών του Deep Learning

- *Αυτόματα αυτοκίνητα*: Οι ερευνητές των αυτόματων αυτοκινήτων χρησιμοποιούν Deep Learning για τον αυτοματοποιημένο εντοπισμό αντικειμένων όπως για παράδειγμα σημάνσεων stop, και φαναριών. Ιδιαίτερα χρήσιμο χαρακτηριστικό αποτελεί ο εντοπισμός πεζών, όπου μπορεί να συμβάλλει στην μείωση του αριθμού των ατυχημάτων σε σημαντικό βαθμό [23].

- *Αεροδιαστημική*: Το Deep Learning χρησιμοποιείται για την αναγνώριση αντικειμένων από δορυφόρους, σε συνδυασμό με τον εντοπισμό περιοχών ενδιαφέροντος, όπως επίσης και την αναγνώριση ασφαλών και μη-ασφαλών περιοχών [24].

- *Ιατρικές έρευνες*: Ερευνητές πάνω στην ασθένεια του καρκίνου χρησιμοποιούν Deep Learning για την αυτόματη αναγνώριση των κυττάρων που σχετίζονται με τον καρκίνο. Η ομάδα του UCLA κατασκεύασε ένα προηγμένο μικροσκόπιο, το οποίο

αποδίδει ένα σύνολο δεδομένων όπου χρησιμοποιείται στην εκπαίδευση του Deep Learning για την αποδοτική αναγνώριση των καρκινικών κυττάρων [25].

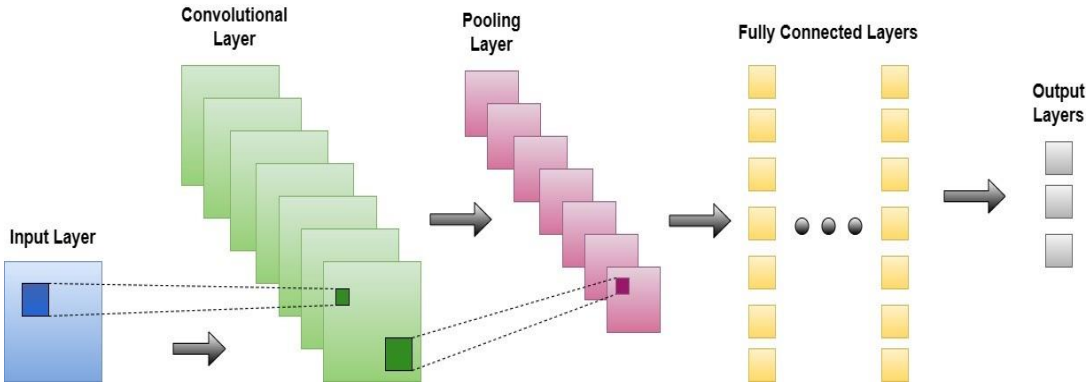
- *Βιομηχανικός Αυτοματισμός*: Το Deep Learning βοηθάει στην ασφάλεια των εργαζομένων όπου βρίσκονται γύρω από βαριά ή επικίνδυνα μηχανήματα. Αυτό επιτυγχάνεται με τον υπολογισμό του πότε τα μηχανήματα βρίσκονται σε μη-ασφαλή απόσταση από τους εργαζομένους [26].
- *Ηλεκτρονικές συσκευές*: Το Deep Learning χρησιμοποιείται στην αυτοματοποιημένη μετάφραση ακοής και ομιλίας. Παράδειγμα αποτελούν συσκευές οικιακής βοήθειας με ανταπόκριση στην ομιλία οι οποίες συνδέονται με εφαρμογές Deep Learning [27].

2.2.5 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network)

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network) διακρίνονται από άλλα νευρωνικά δίκτυα για την ανώτερη απόδοση στην εικόνα, την ομιλία, ή το ηχητικό σήμα. Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) έχουν τρείς βασικούς τύπους επιπέδων (layers). Αυτά τα επίπεδα είναι:

1. *Convolutional Layer*
2. *Pooling Layer*
3. *Fully-Connected (FC) Layer*

Στην Εικόνα 8 απεικονίζεται η αρχιτεκτονική του Συνελικτικού Νευρωνικού Δικτύου. Τα επίπεδα Input Layer, Convolutional Layer και Pooling Layer αποτελούν το κομμάτι της Εξαγωγής Χαρακτηριστικών (Feature Extraction) ενώ τα επίπεδα Fully Connected Layers και Output Layers αποτελούν το κομμάτι της Ταξινόμησης (Classification).



Εικόνα 8. Αρχιτεκτονική Συνελικτικού Νευρωνικού Δικτύου

Το Συνελικτικό Επίπεδο (Convolutional Layer) είναι το πρώτο επίπεδο του Συνελικτικού Νευρωνικού Δικτύου. Έπειτα από το Συνελικτικό Νευρωνικό Δίκτυο, ακολουθούν επιπρόσθετα Συνελικτικά Επίπεδα (Convolutional Layers) ή Pooling Layers. Το Fully-Connected Layer είναι το τελικό layer. Αυξάνοντας τα layers αυξάνεται η πολυπλοκότητα, αναγνωρίζοντας μεγαλύτερα τμήματα της εικόνας, με μεγαλύτερη λεπτομέρεια. Τα αρχικά layers εστιάζουν και αναγνωρίζουν βασικά απλά χαρακτηριστικά, όπως χρώματα, γωνίες, ευθείες και βασικά σχήματα. Όσο επεκτείνεται το δίκτυο προχωρώντας στα επόμενα layers, το Συνελικτικό Νευρωνικό Δίκτυο αναγνωρίζει πιο λεπτομερή στοιχεία και σχήματα του αντικειμένου, μέχρις ότου τελικά καταλήξει στην αναγνώριση του στόχου αντικειμένου [9].

Παρακάτω αναλύονται καθένα από τα layers του Συνελικτικού Νευρωνικού Δικτύου (Convolutional Neural Network):

1. *Convolutional Layer*

Το Convolutional Layer είναι το βασικό δομικό στοιχείο του Συνελικτικού Νευυρωνικού Δικτύου καθώς σε αυτό γίνεται η πλειοψηφία των υπολογισμών. Το Συνελικτικό Νευρωνικό Δίκτυο απαιτεί τα εξής στοιχεία: Τα δεδομένα εισόδου (Input Data), το φίλτρο (Filter), και τον πίνακα χαρακτηριστικών (Feature Map). Στο 2.2.6 παρουσιάζεται με παράδειγμα η διαδικασία της Συνέλιξης.

Έστω ότι έχουμε ως είσοδο μια εικόνα με χρώματα, η οποία αποτελείται από πίνακα με 3D pixels και συνεπώς με 3 διαστάσεις ύψος-πλάτος-βάθος όπου κάθε μία από τις διαστάσεις αντιστοιχεί σε καθένα από τα κανάλια RGB. Επιπλέον έχουμε έναν ανιχνευτή χαρακτηριστικών (feature detector) όπου ονομάζεται φίλτρο (filter) ή kernel. Το φίλτρο αυτό μετακινείται και διαπερνά όλα τα τμήματα της εικόνας, ελέγχοντας εάν το

φίλτρο αυτό εμφανίζεται σε τμήματα της. Η διαδικασία αυτή ονομάζεται Συνέλιξη (Convolution).

Το φίλτρο είναι πίνακας δύο ή τριών διαστάσεων με βάρη. Το φίλτρο λοιπόν, εφαρμόζεται σε μία περιοχή της εικόνας και γίνεται ο υπολογισμός του αθροίσματος του γινομένου μεταξύ των input pixels και των αντίστοιχων pixels του φίλτρου (σημειώνεται ότι αυτή η πράξη μεταξύ στοιχείων εικόνας και φίλτρου μπορεί να οριστεί οποιαδήποτε άλλη πέρα από τον πολλαπλασιασμό). Συνεπώς, το αποτέλεσμα του υπολογισμού αυτού, τροφοδοτείται σε έναν πίνακα εξόδου. Έπειτα, το φίλτρο μετακινείται κατά ένα stride, και επαναλαμβάνεται η ίδια διαδικασία μέχρις ότου το φίλτρο να έχει καλύψει όλη την εικόνα. Έπειτα από τον υπολογισμό και την εφαρμογή της παραπάνω διαδικασίας, δημιουργείται η τελική έξοδος όπου και ονομάζεται πίνακας χαρακτηριστικών (feature map ή activation map).

Έπειτα από την διαδικασία της συνέλιξης, στο Συνελικτικό Νευρωνικό Δίκτυο εφαρμόζεται ένας Rectified Linear Unit (ReLU) μετασχηματισμός στον πίνακα χαρακτηριστικών (feature map), εισάγοντας μη-γραμμικότητα στο μοντέλο.

Όπως αναφέρθηκε νωρίτερα, κάποιο άλλο Συνελικτικό Επίπεδο ακολουθεί το αρχικό Συνελικτικό Επίπεδο ή και το προηγούμενό του. Με την διαδικασία αυτή, η δομή του Συνελικτικού Νευρωνικού Δικτύου γίνεται ιεραρχική καθώς τα επόμενα layers ”βλέπουν” τα pixels των τμημάτων των προηγούμενων επιπέδων. Για παράδειγμα, έστω ότι ο στόχος είναι η αναγνώριση ύπαρξης ποδηλάτου σε μία εικόνα. Το ποδήλατο μπορεί να χωριστεί σε διάφορα τμήματα όπως τιμόνι, ρόδες, πετάλια. Καθένα από τα τμήματα του ποδηλάτου μπορεί να θεωρηθεί lower-level μοτίβο στο Νευρωνικό Δίκτυο, και ο συνδυασμός των τμημάτων αυτών ένα higher-level μοτίβο. Με τον τρόπο αυτό στο Συνελικτικό Νευρωνικό Δίκτυο δημιουργείται ιεραρχία χαρακτηριστικών.

2. Pooling Layer

Τα Pooling Layers, είναι υπεύθυνα για το Downsampling, γίνεται δηλαδή μείωση των διαστάσεων, μειώνοντας τον αριθμό των παραμέτρων της εισόδου. Πιο συγκεκριμένα, όπως στο Convolutional Layer, έτσι και στη διαδικασία Pooling, σαρώνεται ένα φίλτρο στον πίνακα εισόδου, με τη διαφορά ότι σε αυτή την περίπτωση το φίλτρο δεν έχει βάρη. Αντίθετα, το φίλτρο εφαρμόζει μία aggregation συνάρτηση στις τιμές εισόδου έχοντας ως αποτέλεσμα έναν πίνακα εξόδου. Υπάρχουν δύο τύποι Pooling:

(i) *Max Pooling*: Καθώς το φίλτρο μετακινείται κατά μήκος του πίνακα εισόδου, πάνω στο επικαλυπτόμενο τμήμα στοιχείων, επιλέγονται τα pixels με την μέγιστη τιμή για να συμπληρώσουν τον πίνακα εξόδου. Σημειώνεται ότι η τεχνική του Max Pooling χρησιμοποιείται πιο συχνά από ότι το Average Pooling.

(ii) *Average Pooling*: Καθώς το φίλτρο μετακινείται κατά μήκος του πίνακα εισόδου, υπολογίζεται η μέση τιμή πάνω στο επικαλυπτόμενο τμήμα στοιχείων, και συμπληρώνεται ο πίνακας εξόδου.

Κατά το Pooling Layer, χάνεται ένα μέρος της πληροφορίας, αλλά παρόλα αυτά επιφέρει πλεονεκτήματα στο Συνελικτικό Νευρωνικό Δίκτυο. Με την διαδικασία αυτή μειώνεται η πολυπλοκότητα, αυξάνεται η αποδοτικότητα και περιορίζεται ο κίνδυνος για υπερπροσαρμογή (overfitting).

3. Fully-Connected (FC) Layer

Όπως αναφέρθηκε και παραπάνω, οι τιμές pixel της εικόνας εισόδου δεν συνδέονται απευθείας με το επίπεδο εξόδου στα μερικώς συνδεδεμένα επίπεδα. Ωστόσο, στο Fully-Connected Layer, κάθε κόμβος στο επίπεδο εξόδου συνδέεται άμεσα με έναν κόμβο του προηγούμενου επιπέδου. Το Fully-Connected Layer είναι υπεύθυνο για την διαδικασία ταξινόμησης (classification) και βασίζεται στα χαρακτηριστικά που εξήχθησαν στα προηγούμενα επίπεδα. Ενώ το Convolutional Layer και το Pooling Layer χρησιμοποιούν συναρτήσεις ReLU, το Fully-Connected Layer χρησιμοποιεί την συνάρτηση ενεργοποίησης softmax για την κατάλληλη ταξινόμηση των εισόδων, παράγοντας μία πιθανότητα μεταξύ 0 και 1.

Βελτιωμένη αναγνώριση εικόνων μπορεί να επιτευχθεί μέσω προ-εκπαιδευμένων Συνελικτικών Νευρωνικών Δικτύων. Προ-εκπαιδευμένα Συνελικτικά Νευρωνικά Δίκτυα από δημόσια διαθέσιμους πόρους, όπως το ImageNet, μπορούν να χρησιμοποιηθούν σε πολλές εφαρμογές χωρίς αλλαγές στα βάρη τους, παρά μόνο στο τελικό επίπεδο ταξινόμησης. Αυτή η προσέγγιση επιτρέπει την αποτελεσματική αναγνώριση εικόνων, ακόμα και σε διαφορετικά σύνολα δεδομένων. Τα βάρη από τα πρώτα επίπεδα εξάγουν χρήσιμες πληροφορίες σχετικά με μοτίβα στις εικόνες, ενώ τα χαρακτηριστικά προ τελευταίου επιπέδου μπορούν να χρησιμοποιηθούν για unsupervised εφαρμογές, όπως η ανάκτηση εικόνων. Η σημασιολογική φύση των χαρακτηριστικών

καθιστά αυτήν την προσέγγιση δημοφιλή, με την εκπαίδευση από το μηδέν να είναι σπάνια απαραίτητη [29].

2.2.6 Παράδειγμα διαδικασίας Συνέλιξης

Στο σημείο αυτό θα παρουσιαστεί ένα απλό παράδειγμα εφαρμογής Συνελικτικών Νευρωνικών Δικτύων.

Κάθε εικόνα μπορεί να αναπαρασταθεί με έναν πίνακα από pixels, όπου καθένα από τα pixels αναπαριστά και την ένταση του χρώματος. Στην Εικόνα 9 φαίνεται η εικόνα εισόδου (Input Image) και το Φίλτρο (Filter). Το φίλτρο χρησιμοποιείται για την εξαγωγή χαρακτηριστικών από τις εικόνες.

Input Image

0	0	1	1	0	0
0	1	0	0	1	0
1	0	0	0	0	1
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0

Filter (Kernel)

0	0	1
0	1	0
1	0	0

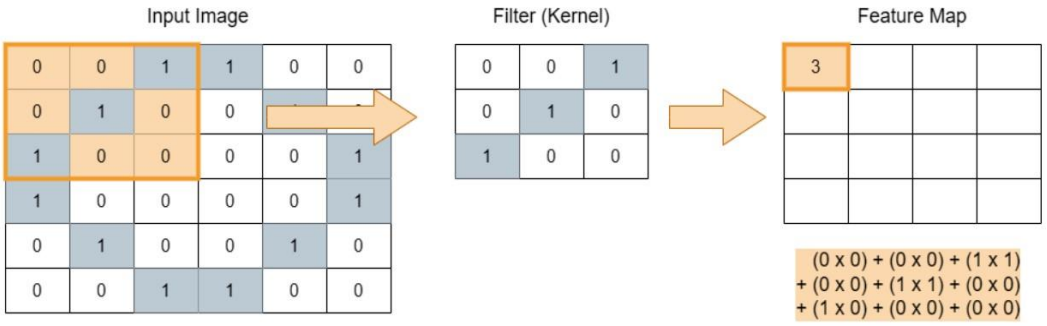
Εικόνα 9. Εικόνα εισόδου και Φίλτρο

Από την συνέλιξη της εικόνας εισαγωγής με το φίλτρο, προκύπτει η έξοδος (output). Στην εικόνα εισόδου εφαρμόζεται το φίλτρο, διαπερνώντας την σε όλα τα τμήματά της και εφαρμόζοντας του υπολογισμούς της συνέλιξης. Το Φίλτρο επιλέγεται να έχει μικρότερες διαστάσεις από τη εικόνα εισόδου. Στο παρόν παράδειγμα η εικόνα εισόδου έχει διαστάσεις 6 × 6 και το Φίλτρο έχει διαστάσεις 3 × 3. Με βάση τις διαστάσεις της εικόνας εισόδου και τις διαστάσεις του φίλτρου, μπορεί να υπολογιστούν οι διαστάσεις της εξόδου εφόσον το Stride είναι ίσο με 1 χρησιμοποιώντας τον τύπο:

- *dim (Output) = dim (InputImage) – dim (Filter) + 1*

Επικαλύπτοντας το φίλτρο πάνω στην εικόνα εισόδου εφαρμόζεται η συνέλιξη, δηλαδή υπολογίζεται το γινόμενο των επικαλυπτόμενων pixels, έπειτα εφαρμόζεται το άθροισμα των γινομένων και προκύπτει η τελική τιμή. Συνεπώς, εφαρμόζεται ο υπολογισμός του αθροίσματος γινομένου μεταξύ της εικόνας εισόδου και του φίλτρου.

Στην Εικόνα 10 απεικονίζεται ο υπολογισμός της συνέλιξης για το πρώτο pixel του Πίνακα Χαρακτηριστικών (Feature Map).



Εικόνα 10. Συνέλιξη μεταξύ εικόνας εισόδου και φίλτρου για τον υπολογισμό του πρώτου pixel στον Πίνακα Χαρακτηριστικών

Η διαδικασία της συνέλιξης εφαρμόζεται σε κάθε 3×3 επικάλυψη, με συγκεκριμένο βήμα όπου ορίζεται. Στο παρόν παράδειγμα το βήμα αυτό (Stride) ορίζεται ίσο με 1. Το Stride είναι μια παράμετρος του φίλτρου που τροποποιεί το μέγεθος της μετακίνησης πάνω από την εικόνα. Αν δώσουμε το βήμα (Stride) ίσο με 2, τότε θα γίνει παράκαμψη κάθε δεύτερου pixel. Ο τύπος υπολογισμού των διαστάσεων της εξόδου και εφόσον το Stride είναι μεγαλύτερο του ενός είναι ο ακόλουθος:

- *dim (Output) = ((dim (InputImage) – dim (Filter)) / Stride) + 1*

Επιπρόσθετα, αξίζει να αναφερθεί ότι ο υπολογισμός των διαστάσεων της εικόνας εξόδου είναι διαφορετικός εφόσον υπάρχει padding κατά την εφαρμογή του φίλτρου στην εικόνα εισόδου. Το padding είναι ένας όρος που αναφέρεται στον αριθμό των pixel που προστίθενται σε μια εικόνα όταν υποβάλλεται σε επεξεργασία κατά την διαδικασία της συνέλιξης. Για παράδειγμα, εάν η τιμή padding είναι 0, τότε κατά την χρήση του φίλτρου για τη σάρωση της εικόνας, το μέγεθος της εικόνας εξόδου θα είναι μικρότερο. Σε περίπτωση που θέλουμε να το αποφύγουμε και συνεπώς να διατηρήσουμε το αρχικό μέγεθος της εικόνας για να εξαγάγουμε ορισμένα χαρακτηριστικά χαμηλού επιπέδου, θα προσθέσουμε μερικά επιπλέον pixel εκτός της εικόνας. Η προσθήκη padding στην εικόνα εισόδου προσδίδει ακριβέστερη ανάλυση της εικόνας, και επιτρέπει την εξαγωγή χαρακτηριστικών χαμηλότερου επιπέδου. Στην Εικόνα 11 φαίνεται η εικόνα εισόδου με προσθήκη padding ίσο με 1. Ο υπολογισμός των διαστάσεων της εικόνας εξόδου εφόσον έχει προστεθεί padding, είναι ο ακόλουθος:

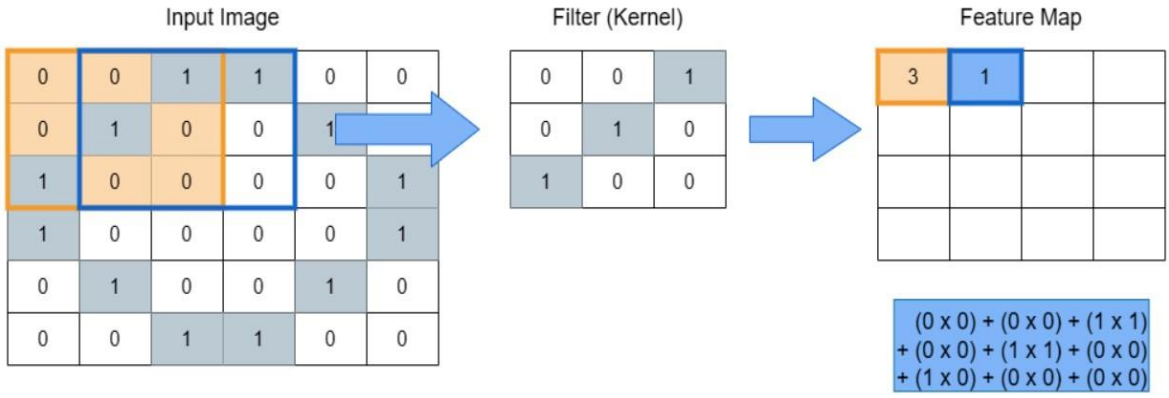
- *dim(Output) = (dim (InputImage) – dim (Filter) + 2 ∗Padding / Stride) + 1*

Input Image with padding equal to 1

0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

Εικόνα 11. Εικόνα εισόδου με προσθήκη padding ίσο με 1

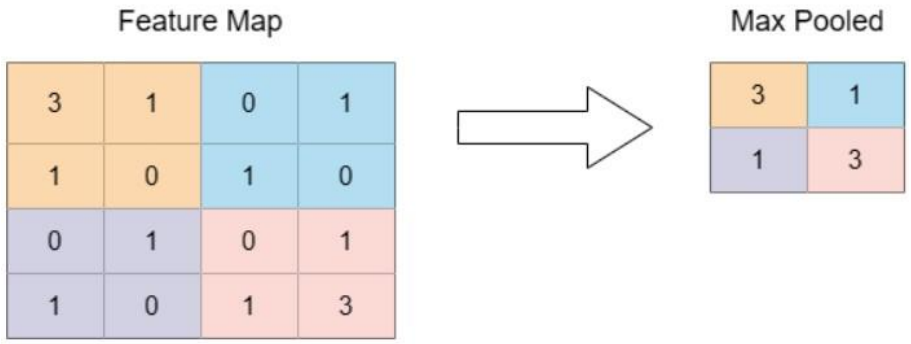
Στην Εικόνα 12 παρουσιάζεται η συνέλιξη της δεύτερης επανάληψης, και συνεχίζεται με τον ίδιο τρόπο μέχρι να επικαλυφθεί ολόκληρη η εικόνα.



Εικόνα 12. Συνέλιξη μεταξύ εικόνας εισόδου και φίλτρου για τον υπολογισμό του δεύτερου pixel στον Πίνακα Χαρακτηριστικών

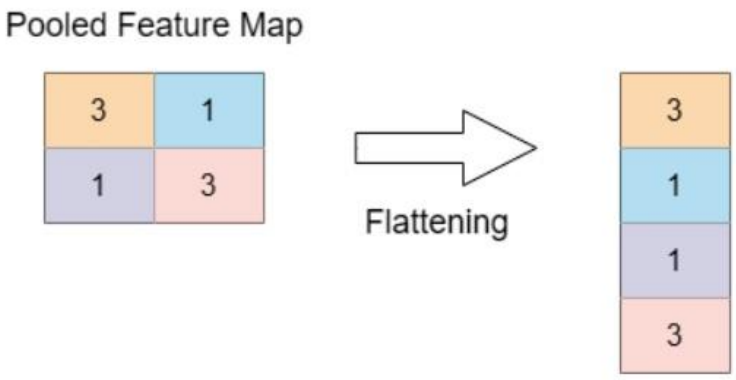
Έπειτα από την εφαρμογή της συνέλιξης προκύπτει ο Πίνακας Χαρακτηριστικών όπως φαίνεται στο αριστερό μέρος της Εικόνας 13. Σε επόμενο βήμα γίνεται εφαρμογή μίας ακόμη διαδικασίας στον Πίνακα Χαρακτηριστικών (Feature Map), όπως απεικονίζεται και στην Εικόνα 13. Γίνεται δηλαδή η επιλογή και διατήρησης της μεγαλύτερης τιμής, και το φίλτρο αυτό μετακινείται με τέτοιον τρόπο ώστε να μην υπάρχει επικάλυψη με τον εαυτό του. Για να γίνεται καλύτερη κατανόηση της επίδρασης του Max Pooling, μπορεί να παρατηρηθεί ότι στο πάνω αριστερά αλλά και στο κάτω δεξιά μέ-

ρος της εικόνας εισόδου (Εικόνα 9), υπάρχει πλήρης ταύτιση με το φίλτρο όπου έχει επιλεγεί. Έτσι, και στον Πίνακα Χαρακτηριστικών (Feature Map) (Εικόνα 13), στα αντίστοιχα μέρη πάνω αριστερά και κάτω δεξιά, παρατηρείται η εμφάνιση της μέγιστης τιμής. Στη συνέχεια, με την εφαρμογή του Max Pooling, μειώνονται οι διαστάσεις της κάθε περιοχής του Πίνακα Χαρακτηριστικών, και με αυτό τον τρόπο μπορεί να φανεί σε ποια μέρη της εικόνας έγινε καλύτερη ταύτιση με το φίλτρο που εφαρμόστηκε. Στο παρόν παράδειγμα εφαρμόζεται Max Pooling 2 × 2 διαστάσεων. Σε αυτό το σημείο να αναφερθεί ότι θα μπορούσε να εφαρμοστεί αντί για Max Pooling, η τεχνική Average Pooling.



Εικόνα 13. Εφαρμογή Max Pooling

Σε επόμενο βήμα, το Pooled Layer μετατρέπεται σε μία στήλη από Input Nodes. Η ισοπέδωση (Flatterning) χρησιμοποιείται για να μετατρέψει όλους τους δισδιάστατους πίνακες που προκύπτουν από Pooled Feature Maps σε ένα ενιαίο μεγάλο συνεχές γραμμικό διάνυσμα. Ο ισοπεδωμένος (Flattened) πίνακας τροφοδοτείται ως είσοδος στο Fully Connected Layer για την ταξινόμηση της εικόνας. Στην Εικόνα 14 απεικονίζεται η εφαρμογή Ισοπέδωσης (Flattening).



Εικόνα 14. Εφαρμογή Ισοπέδωσης (Flattening)

Κεφάλαιο 3: Μεθοδολογία και Τεχνικές

Σε αυτή την ενότητα θα αναλυθούν η μεθοδολογία και οι τεχνικές που εφαρμόστηκαν κατά την υλοποίηση του αλγορίθμου της μηχανικής μάθησης, καθώς επίσης και οι τεχνολογίες που χρησιμοποιήθηκαν.

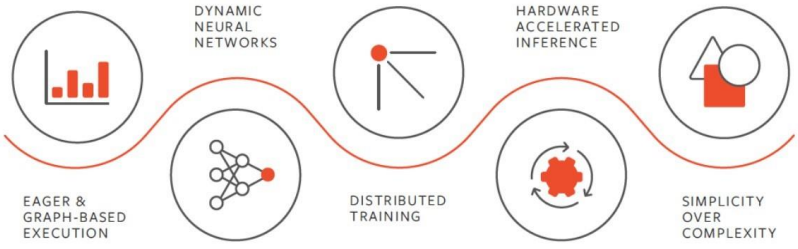
3.1 PyTorch

Το PyTorch είναι μία βιβλιοθήκη ανοιχτού κώδικα. Χρησιμοποιείται ευρέως για τη δημιουργία Deep Learning μοντέλων. Αναπτύχθηκε από το AI research lab της Facebook, και έχει γίνει ιδιαίτερα δημοφιλής από την κοινότητα του Machine Learning λόγω της απλότητας, της ευελιξίας και της ικανότητας να τρέχει σε CPUs και GPUs.

Ένα από τα σημαντικότερα χαρακτηριστικά της PyTorch είναι η βιβλιοθήκη tensor, όμοια με την NumPy με την επιπλέον ικανότητα να τρέχει σε GPUs, κάνοντας πολύ γρηγορότερους τους υπολογισμούς σε μεγάλα σύνολα δεδομένων. Η PyTorch επίσης περιλαμβάνει πολλά εργαλεία για το χτίσιμο και την εκπαίδευση Νευρωνικών Δικτύων, συμπεριλαμβανομένων προ-εκπαιδευμένων μοντέλων, τους Data Loaders, και ποικιλία από loss functions και optimizers.

Εκτός των προαναφερόμενων βασικών χαρακτηριστικών, η PyTorch επίσης περιλαμβάνει και άλλα εργαλεία και βιβλιοθήκες για διάφορες εργασίες όπως Natural Language Processing, Computer Vision και άλλα. Έχει σχεδιαστεί για να είναι εύκολα χρησιμοποιούμενη και ευέλικτη με έμφαση στη γρήγορη κατασκευή μοντέλων [22].

Συνοψίζοντας, η PyTorch είναι ισχυρό και δημοφιλές εργαλείο για την ανάπτυξη και την κατασκευή Deep Learning μοντέλων, και είναι ευρέως χρησιμοποιήσιμη από τους ερευνητές και ασκούμενους στο πεδίο αυτό. Στην Εικόνα 15 απεικονίζονται τα χαρακτηριστικά της PyTorch.



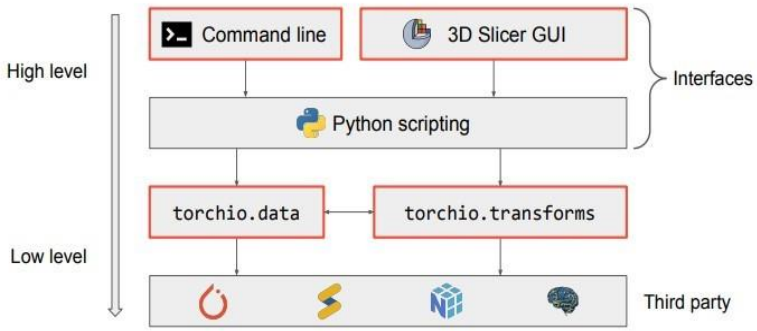
Εικόνα 15. Χαρακτηριστικά της Pytorch [28]

3.2 Torchio

Το TorchIO είναι open-source βιβλιοθήκη της Python όπου προσφέρει αποδοτική φόρτωση, προ-επεξεργασία, επαύξηση (augmentation), patch-based sampling στις 3D ιατρικές εικόνες. Το TorchIO στηρίζεται στον σχεδιασμό του PyTorch [7]. Περιλαμβάνει πολλαπλούς μετασχηματισμούς έντασης (intensity) καθώς και χωρικούς (spatial) μετασχηματισμούς, για επαύξηση δεδομένων (data augmentation) και προ-επεξεργασία δεδομένων (data preprocessing).

Το TorchIO καθώς είναι modular βιβλιοθήκη μπορεί να χρησιμοποιηθεί στα πλαίσια higher-level Deep Learning στις ιατρικές εικόνες, όπως το Medical Open Network for AI (MONAI). Ένα από τα σημαντικότερα χαρακτηριστικά του TorchIO είναι ότι απαλλάσσει τους ερευνητές από την ανάγκη για γραφή του κώδικα στο κομμάτι της προ-επεξεργασίας. Επιπρόσθετα, επιτρέπει στους ερευνητές στην εστίαση σε διάφορα πειράματα, στην υποστήριξη της αναπαραγωγιμότητας του πειράματος και στην τυποποίηση των μεθόδων που χρησιμοποιούνται για την επεξεργασία των ιατρικών εικόνων [8].

Όπως προαναφέρθηκε, το TorchIO χρησιμοποιεί πλατφόρμες λογισμικού ιατρικής απεικόνισης ανοιχτού κώδικα. Τα πακέτα επιλέχθηκαν για να μειωθεί ο αριθμός των απαιτούμενων εξωτερικών εξαρτήσεων αλλά και η ανάγκη της εκ νέου υλοποίησης βασικών λειτουργιών επεξεργασίας ιατρικής απεικόνισης (φόρτωση εικόνας, επαναδειγματοληψία κ.λπ.). Τα χαρακτηριστικά TorchIO χωρίζονται σε δύο κατηγορίες: δομές δεδομένων και εισαγωγή/εξαγωγή (torchio.data) και μετασχηματισμοί για προ-επεξεργασία και επαύξηση (torchio.transforms). Η Εικόνα 16 αντιπροσωπεύει ένα διάγραμμα των διαφορετικών διεπαφών της βιβλιοθήκης. Τα πλαίσια με κόκκινο περίγραμμα αντιπροσωπεύουν τα στοιχεία που υλοποιούνται στο TorchIO [8]. Τα λογότυπα υποδεικνύουν βιβλιοθήκες της Python χαμηλότερου επιπέδου που χρησιμοποιούνται από το TorchIO. Στην εικόνα αναπαρίστανται με την εξής σειρά: PyTorch, SimpleITK, NumPy, NiBabel.



Εικόνα 16. Διάγραμμα του TorchIO, οι εξαρτήσεις του και οι διεπαφές του. [8]

3.3 Προ-επεξεργασία δεδομένων με χρήση TorchIO

Αρχικά διαβάστηκε το αρχείο excel με στήλες το *study_id* (από 1 έως 1.110), το *category* (από CT-0 έως CT-4), και το *study_file* (τα paths των αρχείων με τις εικόνες των ασθενών). Τα αρχεία με τις εικόνες είναι σε τρισδιάστατη μορφή NIfTI, με διαστάσεις εικόνων 512×512×(36-41). Παρακάτω αναλύεται και επεξηγείται η μορφή αρχείου NIfTI.

Το NIfTI (Neuroimaging Informatics Technology Initiative) είναι μια μορφή αρχείου που δημιουργήθηκε στις αρχές της δεκαετίας του 2000 από μια επιτροπή που εδρεύει στα Εθνικά Ινστιτούτα Υγείας με σκοπό να δημιουργήσει μια μορφή για νευροαπεικόνιση διατηρώντας τα πλεονεκτήματα της μορφής Analyze, αλλά επιλύοντας τις αδυναμίες [38].

Η μορφή αρχείου NIfTI χρησιμοποιείται για την αποθήκευση δεδομένων ιατρικής απεικόνισης, όπως Μαγνητική Τομογραφία (MRI) και Αξονική Τομογραφία (CT-Scan). Προσφέρει πολλά χαρακτηριστικά, όπως η αποθήκευση raw δεδομένων σε τρισδιάστατη εικόνα που περιλαμβάνει δύο συντεταγμένες για τη συσχέτιση του δείκτη voxel με τον χωρικό δείκτη. Ένα πλεονέκτημα του NIFTI είναι η αποθήκευση δύο αρχείων ανά μία 3D σάρωση αντί να διαχειρίζεται πολλά αρχεία Analyze. Επιπλέον, το αρχείο NIfTI μπορεί να χρησιμοποιηθεί για την αποθήκευση πρόσθετων παραμέτρων, όπως σημαντικές παράμετροι απόκτησης και πειραματικός σχεδιασμός [21].

Η μορφή NIfTI σε αντίθεση με την μορφή DICOM, αναπτύχθηκε για να διευκολύνει την ανταλλαγή δεδομένων νευροαπεικόνισης μεταξύ διαφορετικών πακέτων λογισμικού και να παρέχει μια βάση για δια-λειτουργικότητα μεταξύ διαφορετικών κέντρων νευροαπεικόνισης. Αυτή η μορφή αρχείου, βασίζεται στην ευρέως χρησιμοποιούμενη μορφή αρχείου Analyze, με ορισμένες πρόσθετες δυνατότητες και περιορισμούς [31]. Διάφορες βιβλιοθήκες έχουν αναπτυχθεί από την κοινότητα για να διαβάζουν και να

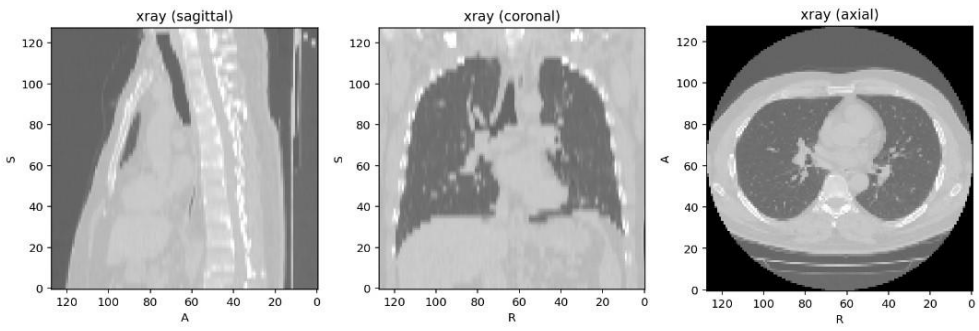
γράφουν αρχεία NIfTI, πράγμα που σημαίνει ότι οι προγραμματιστές δεν χρειάζεται να εφαρμόσουν ξανά υποστήριξη σε αυτήν τη μορφή (η οποία είναι χρονοβόρα, αλλά δίνει επίσης την ευκαιρία για εισαγωγή σφαλμάτων) [21].

Καθώς οι κατηγορίες των εικόνων είναι 5, από CT-0 έως CT-4, το πρόβλημα αποφασίζεται να μετατραπεί σε πρόβλημα δυαδικής ταξινόμησης, με labels 0 και 1. Συνεπώς, πλέον η κατηγορία CT-0 κατατάσσεται στο label 0 (αρνητικά περιστατικά Covid-19) και οι κατηγορίες CT-1, CT-2, CT-3, CT-4 κατατάσσονται στο Label 1 (θετικά περιστατικά Covid-19). Επομένως, συνολικά από τις εικόνες που χρησιμοποιήθηκαν στην παρούσα έρευνα τα 856 περιστατικά ασθενών είναι θετικά στον Covid-19 και τα 254 περιστατικά ασθενών είναι αρνητικά στον Covid-19. Η κατανομή αυτή των περιστατικών παρουσιάζεται στον Πίνακα 1.

Πίνακας 1. Κατανομή δεδομένων στις δύο κλάσεις

Πλήθος των Subjects	**Κατηγορία Ταξινόμησης**
254	0
856	1

Έχοντας τα paths και τα labels των εικόνων, χρησιμοποιήθηκε το TorchIO για την δημιουργία των subject, ένα για κάθε ασθενή. Το Subject είναι μία δομή δεδομένων που χρησιμοποιείται για την αποθήκευση των εικόνων που σχετίζονται με ένα subject και οποιοδήποτε άλλο metadata απαραίτητο για την προ-επεξεργασία. Ένα subject περιέχει 36 έως 41 axial εικόνες, αναλόγως την περίπτωση του subject. Στην Εικόνα 17 απεικονίζεται ένα από τα axial slices ενός τυχαίου subject. Στον κώδικα της παρούσας έρευνας, παρήχθησαν επιπλέον και οι άλλες 2 πλευρές των πνευμόνων (sagital και coronal), έχοντας το τρισδιάστατο αρχείο του ασθενή.



Εικόνα 17. Παράδειγμα ενός Subject - μία εικόνα από κάθε πλευρά

Οι αρχικές διαστάσεις των εικόνων ήταν 512×512×(36−41). Κατά την προ-επεξεργασία μέσω του TorchIO έγινε αλλαγή μεγέθους των εικόνων σε 256×256×41. Το *batch_size* τέθηκε ίσο με 8.

Στο PyTorch, η κλάση DataLoader παρέχει έναν εύκολο τρόπο φόρτωσης δεδομένων παράλληλα και σε παρτίδες. Η κλάση DataLoader χειρίζεται αυτόματα τη φόρτωση, την ταξινόμηση των δεδομένων και είναι υπεύθυνοι για την προετοιμασία των δεδομένων για εκπαίδευση.

Κατά τη διαδικασία εκπαίδευσης ενός μοντέλου Deep Learning, το σύνολο δεδομένων πρέπει να διαβαστεί από τη μνήμη και να υποβληθεί σε προ-επεξεργασία, προτού μεταδοθεί ως είσοδος στο μοντέλο. Αυτή η λειτουργία απαιτεί τη φόρτωση των δεδομένων στη μνήμη ταυτόχρονα. Στις περισσότερες περιπτώσεις και, ειδικά, με μεγάλα σύνολα δεδομένων, υπάρχει το πρόβλημα έλλειψης μνήμης και συνεπώς του χρόνου απόκρισης του συστήματος. Αυτό το μειονέκτημα αντιμετωπίζεται με βιβλιοθήκες Deep Learning χρησιμοποιώντας τους Data Loaders. Αυτή η δομή παρέχει έναν τρόπο επανάληψης του συνόλου δεδομένων αξιοποιώντας την παράλληλη επεξεργασία, την ανάκτηση, τη δέσμευση και άλλες τεχνικές για τη μείωση του χρόνου φόρτωσης δεδομένων όπως επίσης και της επιβάρυνσης της μνήμης. Ο κύριος στόχος του DataLoader είναι να εκτελέσει τις ενέργειες που είναι υπεύθυνες για τη μεταφορά δεδομένων από μια θέση αποθήκευσης στη μνήμη για εκπαίδευση, έτσι ώστε να σχηματιστεί μια παρτίδα δειγμάτων που θα τροφοδοτηθούν στο μοντέλο. Με τη χρήση των DataLoaders μπορεί να διαπιστωθεί μεγάλη επίδραση στο συνολικό χρόνο που απαιτείται για την ολοκλήρωση της εκπαίδευσης [30].

Κατά την εκπαίδευση του μοντέλου, τα δείγματα (samples) περνούν σε ”minibatches”, γίνεται αναπροσαρμογή των δεδομένων σε κάθε epoch για μείωση της πιθανότητας της υπερπροσαρμογής (overfitting) του μοντέλου και χρησιμοποιείται η Python για γρηγορότερη ανάκτηση των δεδομένων.

Πιο αναλυτικά, παρακάτω επεξηγούνται οι παράμετροι του DataLoader όπου χρησιμοποιήθηκαν:

- *batch_size*: Ο αριθμός των δειγμάτων ανά batch. Ο default αριθμός είναι 1.
- *shuffle*: Εάν πρόκειται να αναμειχθούν τα δεδομένα πριν από κάθε epoch. Η default τιμή είναι True.

- *num_workers*: Ο αριθμός των worker threads που χρησιμοποιούνται για την φόρτωση των δεδομένων. Η default τιμή ειναι 0.

3.4 Cross Validation

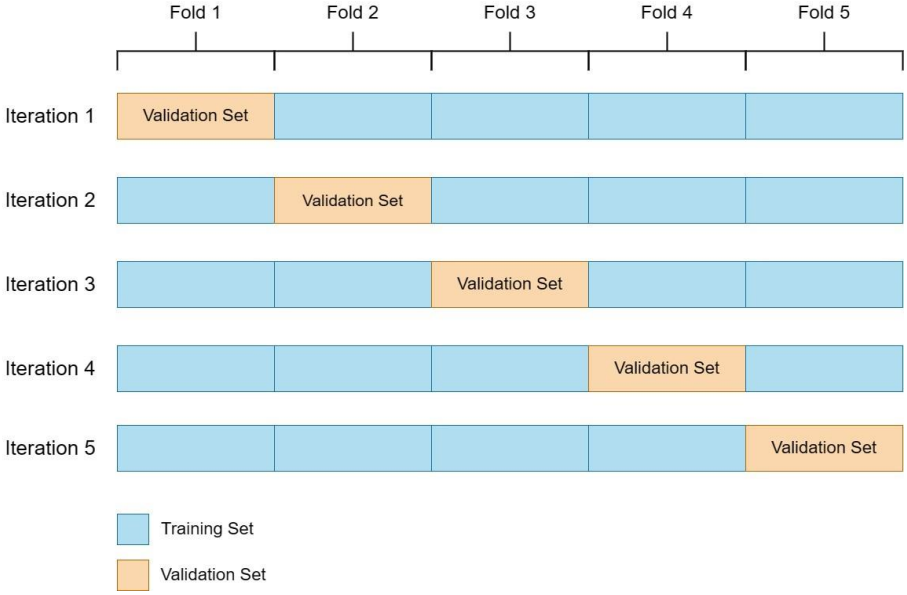
Η Τεχνική του Cross Validation είναι μια από τις πιο ευρέως χρησιμοποιούμενες μεθόδους επαναδειγματοληψίας δεδομένων και βοηθάει στην γενίκευση του μοντέλου αλλά και την αποφυγή του overfitting.

Η τεχνική του K-Fold Cross Validation χρησιμοποιεί διαφορετικά τμήματα δεδομένων για να εκπαιδεύσει και να τεστάρει ένα μοντέλο σε διαφορετικά τμήματα και σε διαφορετικές επαναλήψεις [40]. Η τεχνική Cross-Validation είναι παρόμοια με τη μέθοδο επαναλαμβανόμενης τυχαίας δειγματοληψίας, αλλά η δειγματοληψία γίνεται με τέτοιο τρόπο ώστε να μην επικαλύπτονται δύο σύνολα. Στη μέθοδο αυτή K-Fold, το διαθέσιμο σύνολο εκμάθησης χωρίζεται σε K υποσύνολα περίπου ίσου μεγέθους. Το Fold αναφέρεται στα υποσύνολα που προκύπτουν. Αυτός ο διαχωρισμός εκτελείται με τυχαία δειγματοληψία από το σύνολο εκμάθησης χωρίς αντικατάσταση. Το μοντέλο εκπαιδεύεται χρησιμοποιώντας K-1 υποσύνολα, τα οποία όλα μαζί αντιπροσωπεύουν το σύνολο εκπαίδευσης (training set). Αυτή η διαδικασία επαναλαμβάνεται έως ότου καθένα από τα K υποσύνολα χρησιμοποιηθεί ως σύνολο επικύρωσης (test set). Ο μέσος όρος των K μετρικών απόδοσης (performances) στα K σύνολα επικύρωσης είναι η τελική Cross-Validated απόδοση [39]. Το πώς λειτουργεί η τεχνική αυτή εξηγείται επακολούθως με παράδειγμα.

Για παράδειγμα, έστω ότι γίνεται εφαρμογή της τεχνικής 5 Folds Cross Validation. Στην περίπτωση αυτή γίνεται τυχαίος διαχωρισμός του συνόλου εκπαίδευσης σε 5 ίσα (ή σχεδόν ίσα) τμήματα. Έτσι έχουμε τα εξής τμήματα: F_1, F_2, F_3, F_4, F_5. Καθένα από τα F_k, k = 1,...,5 περιέχει το 20% των δεδομένων του συνόλου εκπαίδευσης. Συνεπώς, 5 μοντέλα εκπαιδεύονται με τον ακόλουθο τρόπο. Για την εκπαίδευση του πρώτου μοντέλου, f_1 χρησιμοποιούνται όλα τα υποσύνολα από το F_2 έως και το F_5 ως σύνολο εκπαίδευσης (training set) και το υποσύνολο F_1 ως σύνολο επικύρωσης (validation set). Για την εκπαίδευση του δεύτερου μοντέλου, f_2 τα υποσύνολα F_1, F_3, F_4, F_5 χρησιμοποιούνται ως σύνολο εκπαίδευσης (training set) και το υποσύνολο F_2 ως σύνολο επικύρωσης (validation set). Για την εκπαίδευση του τρίτου μοντέλου, f_3 τα υποσύνολα F_1, F_2, F_4, F_5 χρησιμοποιούνται ως σύνολο εκπαίδευσης (training set) και το υποσύνολο F_3 ως σύνολο επικύρωσης (validation set). Για την εκπαίδευση του τέταρτου μοντέλου, f_4 τα υποσύνολα F_1, F_2, F_3, F_5 χρησιμοποιούνται ως σύνολο εκπαί-

δευσης (training set) και το υποσύνολο F_4 ως σύνολο επικύρωσης (validation set). Τέλος, για την εκπαίδευση του πέμπτου μοντέλου, f_5 τα υποσύνολα F_1, F_2, F_3, F_4 χρησιμοποιούνται ως σύνολο εκπαίδευσης (training set) και το υποσύνολο F_5 ως σύνολο επικύρωσης (validation set). Καθώς η διαδικασία αυτή γίνεται επαναληπτικά, υπολογίζεται μία τιμή μετρικής, για καθένα από τα σύνολα επικύρωσης από το F_1 έως το F_5 (validation sets). Στη συνέχεια υπολογίζεται ο μέσος όρος των 5 αυτών τιμών, έτσι ώστε να προκύψει τελικά, μία τιμή ως μετρική του μοντέλου. Μέσω της Εικόνας 18 μπορεί να γίνει ευκολότερη η κατανόηση της τεχνικής του Cross-Validation.

Ο στόχος της τεχνικής του Cross-Validation είναι να ελεγχθεί η ικανότητα του μοντέλου να προβλέπει νέα δεδομένα που δεν χρησιμοποιήθηκαν για την εκτίμησή του. Κατά αυτό τον τρόπο περιορίζεται το πρόβλημα του overfitting [41]. Επιπλέον, η τεχνική αυτή συνεισφέρει στην γενίκευση του μοντέλου σε κάποιο άγνωστο σύνολο δεδομένων, με σκοπό την ακριβέστερη πρόβλεψη.



Εικόνα 18. Cross Validation με 5 Folds

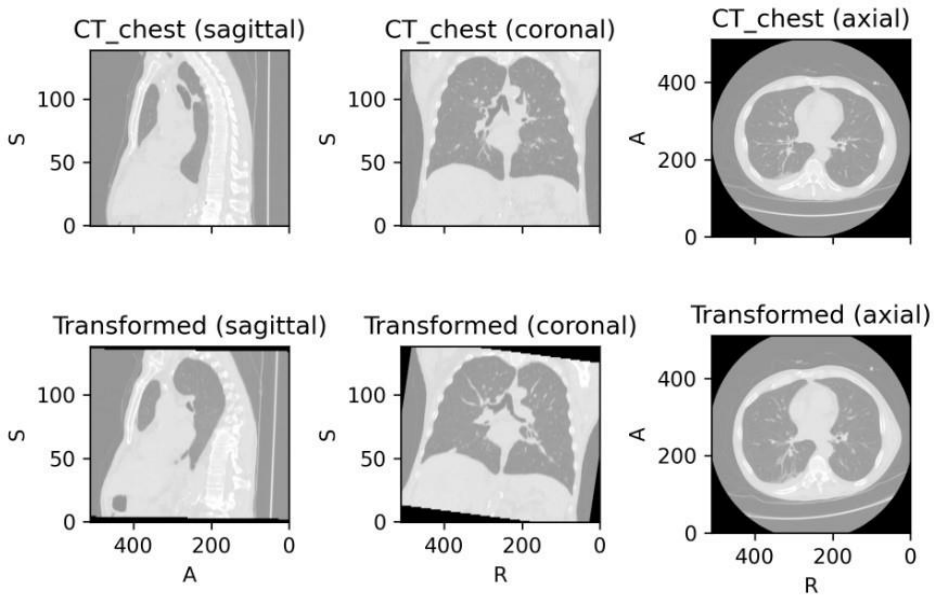
3.5 Επαύξηση Δεδομένων (Data Augmentation)

Ένα κλασικό πρόβλημα της Βαθιάς Μηχανικής Μάθησης είναι η περιορισμένη ποσότητα δεδομένων. Μία λύση στο πρόβλημα αυτό φέρνει η Επαύξηση των Δεδομένων (Data Augmentation), η οποία περιλαμβάνει μία σειρά από τεχνικές που βελτιώνουν το μέγεθος και την ποιότητα των συνόλων εκπαίδευσης. Οι αλγόριθμοι επαύξησης εικόνας περιλαμβάνουν γεωμετρικούς μετασχηματισμούς, επαύξηση του χώρου χρώματος, φίλτρα, ανάμειξη διαφορετικών εικόνων, τυχαία διαγραφή. Η Επαύξηση

Δεδομένων μπορεί να βελτιώσει την απόδοση των μοντέλων και να επεκτείνει τα περιορισμένα σύνολα δεδομένων για να εκμεταλλευτούν τις δυνατότητες των μεγάλων συνόλων δεδομένων [32].

Επιπλέον, η τεχνική της επαύξησης δεδομένων βρίσκει λύση στο πρόβλημα του overfitting. Αυτό συμβαίνει διότι η εικόνα επιφέρει διάφορους μετασχηματισμούς και τροποποιήσεις και έτσι το μοντέλο δεν μαθαίνει αποκλειστικά και μόνο τι εικόνες εισόδου. Με αυτό τον τρόπο είναι καλύτερη η πρόβλεψη σε άλλες πιθανώς διαφορετικές εικόνες εισόδου. Το πρόβλημα του overfitting είναι συχνό σε μικρού πλήθους δεδομένα.

Στην Εικόνα 19, παρουσιάζεται η Αξονική Τομογραφία ενός ασθενή. Στη πρώτη γραμμή της εικόνας απεικονίζεται η αυθεντική εικόνα και στην δεύτερη γραμμή φαίνεται η εικόνα έπειτα από την τεχνική augmentation.



Εικόνα 19. Αξονική Τομογραφία original και augmented [33]

3.6 Μεταφορά Μάθησης (Transfer Learning)

Η ”Μεταφορά Μάθησης” (Transfer Learning) είναι ένα άλλο παράδειγμα που συνεισφέρει στην αποφυγή overfitting. Ένα Συνελικτικό Νευρωνικό Δίκτυο απαιτεί μεγάλη ποσότητα επισημασμένων δεδομένων εκπαίδευσης για να είναι αποτελεσματικό, τα οποία ενδέχεται να μην είναι διαθέσιμα. Έτσι, χρησιμοποιείται ένας τρόπος μεταφοράς μάθησης με την εκπαίδευση ενός Συνελικτικού Νευρωνικού Δικτύου χρησιμοποιώντας διαθέσιμα επισημασμένα δεδομένα προέλευσης και στη συνέχεια την εξα-

γωγή των εσωτερικών στρωμάτων του Συνελικτικού Νευρωνικού Δικτύου (τα οποία αντιπροσωπεύουν μια γενική αναπαράσταση ενδιάμεσου επιπέδου) [34]. Η Μεταφορά Μάθησης λειτουργεί εκπαιδεύοντας ένα δίκτυο σε ένα μεγάλο σύνολο δεδομένων, όπως το ImageNet, και στη συνέχεια χρησιμοποιεί αυτά τα βάρη ως τα αρχικά βάρη σε ένα νέο πρόβλημα κατηγοριοποίησης. Συνήθως, αντί για ολόκληρο το δίκτυο συμπεριλαμβανομένων των πλήρως συνδεδεμένων επιπέδων, αντιγράφονται μόνο τα βάρη στα Convolutional Layers. Αυτή η τεχνική είναι αρκετά αποτελεσματική, διότι πολλά σύνολα δεδομένων εικόνων μοιράζονται τα ίδια ή παρόμοια χαρακτηριστικά χαμηλού επιπέδου με άλλα μεγαλύτερα σύνολα δεδομένων εικόνων, πάνω στα οποία έχουν ήδη προ-εκπαιδευτεί κάποια μοντέλα, και έχουν χρησιμοποιηθεί με τον καλύτερο δυνατό τρόπο [35].

3.7 Προ-εκπαίδευση (Pretraining)

Η ”Προ-εκπαίδευση” (Pretraining) είναι έννοια ανάλογη με τη Μεταφορά Μάθησης. Στην Προ-εκπαίδευση, η αρχιτεκτονική του δικτύου ορίζεται και στη συνέχεια εκπαιδεύεται σε ένα μεγάλο σύνολο δεδομένων, όπως το ImageNet.

Το ImageNet έχεις ως στόχο να παρέχει μια πολύ υψηλή και με μεγάλη ποικιλία κάλυψη εικόνων στον κόσμο. Τα 12 υποδέντρα που βρίσκονται σε λειτουργία αυτή τη στιγμή αποτελούνται από συνολικά 3,2 εκατομμύρια εικόνες, οι οποίες είναι κατηγοριοποιημένες σε 5,247 διαφορετικές κατηγορίες. Σε ό,τι αφορά το μέγεθος και την πολυπλοκότητα των δεδομένων, αυτή η συλλογή αποτελεί ήδη το μεγαλύτερο διαθέσιμο σύνολο δεδομένων εικόνων στην ερευνητική κοινότητα της υπολογιστικής όρασης [36].

Η προ-εκπαίδευση είναι μια τεχνική που εφαρμόζεται σε διάφορες έρευνες και σε εφαρμογές των Συνελικτικών Νευρωνικών Δικτύων. Σε ερευνητικά περιβάλλοντα, η προ-εκπαίδευση εφαρμόζεται παντού στην ανίχνευση και την τμηματοποίηση αντικειμένων τελευταίας τεχνολογίας [37]. Η προ-εκπαίδευση περιορίζει την μάθηση σε ένα υποσύνολο του χώρου των παραμέτρων μέσω του fine-tuning σε ένα supervised πρόβλημα. Πρόσφατη εργασία έδειξε επίσης κάποια στοιχεία που υποδηλώνουν ότι η βελτίωση των προ-εκπαιδευμένων μοντέλων γλώσσας δεν αποκλίνει σημαντικά από τα προ-εκπαιδευμένα βάρη. Με άλλα λόγια, τα τελικά βάρη προκαθορίζονται ως επί το πλείστων από την προ-εκπαίδευση. Η προ-εκπαίδευση δίνει τη δυνατότητα εκκίνησης των βαρών χρησιμοποιώντας μεγάλα σύνολα δεδομένων, ενώ εξακολουθεί να επιτρέπει την ευελιξία στον σχεδιασμό της αρχιτεκτονικής του δικτύου.

Κεφάλαιο 4: Περιγραφή Συνόλου Δεδομένων & Πειραματική Διαδικασία

4.1 Περιγραφή Συνόλου δεδομένων

Στην παρούσα έρευνα χρησιμοποιήθηκε το σύνολο δεδομένων MosMedData το οποίο περιλαμβάνει 1,110 CT volumes τα οποία συλλέχθηκαν από περιστατικά στη Μόσχα από την 1η Μαρτίου του 2020 έως την 25η Απριλίου του 2020. Αυτό το σύνολο δεδομένων περιέχει Αξονικές Τομογραφίες ανθρώπινου πνεύμονα με και χωρίς ραδιολογικά ευρήματα που σχετίζονται με τον COVID-19. Στην αρχή της πανδημίας, η Αξονική Τομογραφία χρησίμευσε ως βασικό εργαλείο για τη διάγνωση και την παρακολούθηση της εξέλιξης του COVID-19 στη Μόσχα. Ειδικοί εμπειρογνώμονες βαθμολόγησαν τη σοβαρότητα του COVID-19. Κατέταξαν τη σοβαρότητα της κατάστασης των περιστατικών με βάση τα ευρήματα σε 5 κατηγορίες.

Από όλες τις Αξονικές Τομογραφίες, σε ένα σύνολο μελετών και συγκεκριμένα 50 σε αριθμό, έχει γίνει annotation από τους ειδικούς της έρευνας και του Practical Clinical Center for Diagnostics and Telemedicine Technologies της Μόσχας του τμήματος υγείας. Κατά τη διάρκεια των annotations για κάθε ground-glass εικόνα, οι περιοχές αδιαφάνειας και ενοποίησης επιλέχθηκαν ως θετικά (λευκά) pixels στο αντίστοιχο binary pixel mask. Καθένα από τα περιστατικά (studies) αποθηκεύτηκε σε NIfTI format και έπειτα μετατράπηκε σε αρχείο Gzip.

4.1.1 Κατηγορίες περιστατικών σχετικές με τον COVID-19

Ο αριθμός των περιστατικών ανά κατηγορία είναι ο εξής: CT-0 (254 - 22.8%), CT-1 (684 - 61.6%), CT-2 (125 - 11.3%), CT-3 (45 - 4.1%), CT-4 (2 - 0.2%).

- Στην κατηγορία CT-0 (Zero) κατατάχθηκαν τα περιστατικά τα οποία δεν εμφανίζουν κάποια πνευμονική πάθηση σχετική με τον COVID-19 (και τη πνευμονία).
- Στην κατηγορία CT-1 (Mild) κατατάχθηκαν τα περιστατικά στα οποία εμφανίστηκε προσβολή πνευμονικού παρεγχύματος σε ποσοστό ≤ 25%. Στη περίπτωση αυτή της ήπιας μορφής COVID-19 αποφασίζεται παρακολούθηση στο σπίτι με χρήση τεχνολογιών τηλεϊατρικής.

- Στη κατηγορία CT-2 (Moderate) κατατάχθηκαν τα περιστατικά τα οποία εμφάνισαν πνευμονική παρεγχυματική προσβολή σε ποσοστό 25% − 50%. Στη περίπτωση αυτή αποφασίζεται παρακολούθηση στο σπίτι από ιατρό.
- Στην κατηγορία CT-3 (Severe) κατατάχθηκαν τα περιστατικά που εμφάνισαν πνευμονική παρεγχυματική προσβολή σε ποσοστό 50% − 75%. Η προσβολή των πνευμόνων αυξήθηκε σε 24 – 48 ώρες κατά 50% στην αναπνευστική ανεπάρκεια ανά μελέτες παρακολούθησης. Στη περίπτωση αυτή απαιτείται άμεση εισαγωγή σε εξειδικευμένο COVID νοσοκομείο. Στο νοσοκομείο γίνεται άμεση μεταφορά στη μονάδα εντατικής θεραπείας και ανάνηψης.
- Τέλος, στην κατηγορία CT-4 (Critical) κατατάχθηκαν τα περιστατικά στα οποία εμφανίστηκαν διάχυτες αδιαφάνειες και πνευμονική παρεγχυματική προσβολή σε ποσοστό ≥ 75%. Στη περίπτωση αυτή, η οποία και είναι περισσότερο κρίσιμη σε σχέση με τις υπόλοιπες απαιτείται επείγουσα ιατρική περίθαλψη, καθώς υπάρχει αναπνευστική ανεπάρκεια αλλά και ανεπάρκεια πολλαπλών οργάνων. Γίνεται άμεση εισαγωγή σε εξειδικευμένο νοσοκομείο για ασθενείς με διάγνωση COVID-19, με άμεση μεταφορά στη μονάδα εντατικής θεραπείας και ανάνηψης.

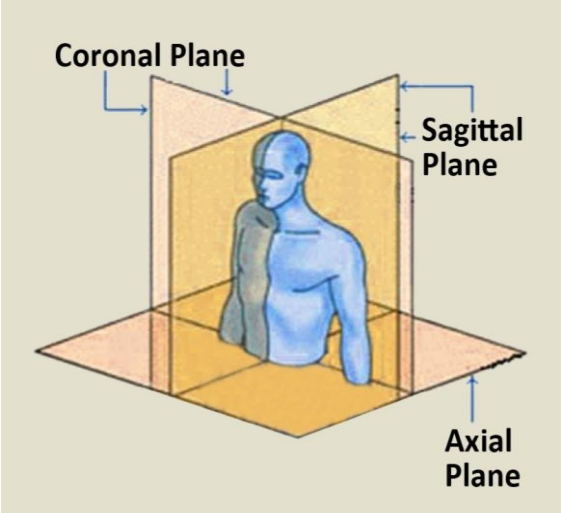
Ο συνολικός πληθυσμός των ανθρώπων από τους οποίους λήφθηκαν τα CT-Scans είναι 1,110. Από αυτούς το 42% είναι Άνδρες, το 56% είναι Γυναίκες και το 2% είναι άλλο/unknown. Οι ηλικίες των ανθρώπων είναι από 18 έως 97 χρόνων, με μέσο όρο ηλικίας 47 χρόνων. Σε πρώτο βήμα, όλα τα περιστατικά (n=1110) κατανεμήθηκαν σε 5 κατηγορίες σύμφωνα με την ταξινόμηση. Στον Πίνακα 2 απεικονίζονται οι 5 κατηγορίες μαζί με τα κλινικά δεδομένα και τις παροτρύνσεις/αποφάσεις των ειδικών για καθεμία από αυτές.

4.1.2 Περιγραφή μορφής και μεγέθους των εικόνων

Το μέγεθος κάθε εικόνας είναι 512×512×(36−41). Για καθέναν από τους ανθρώπους, υπάρχει ένα volume και συνεπώς μία Αξονική Τομογραφία. Κάθε μία από αυτές με βάση το dataset έχει από 36 έως 41 slices. Αυτά τα slices είναι στο axial επίπεδο, παρόλο που θα μπορούσαν να ληφθούν και από τα δύο υπόλοιπα επίπεδα δηλαδή το sagittal και το coronal. Σε σύγκριση με τις X-Rays, οι CT-Scans είναι πιο ισχυρό και εξελιγμένο είδος ακτινογραφίας που λαμβάνει μια εικόνα 360 μοιρών των εσωτερικών οργάνων.

Η Εικόνα 20 αφορά ένα συγκεκριμένο περιστατικό (ένα volume) και απεικονίζονται 40 slices ξεκινώντας από το πάνω μέρος το άξονα και προχωρώντας προς τα κάτω. Μπορούν να παρατηρηθούν κάθε μορφής διαφάνειες στα slices σε καθένα από τα επίπεδα, καθώς καθένα από αυτά μπορεί να φανερώσει κάποιας μορφής ανωμαλία-αδιαφάνεια και ένδειξη προσβολής από COVID-19.

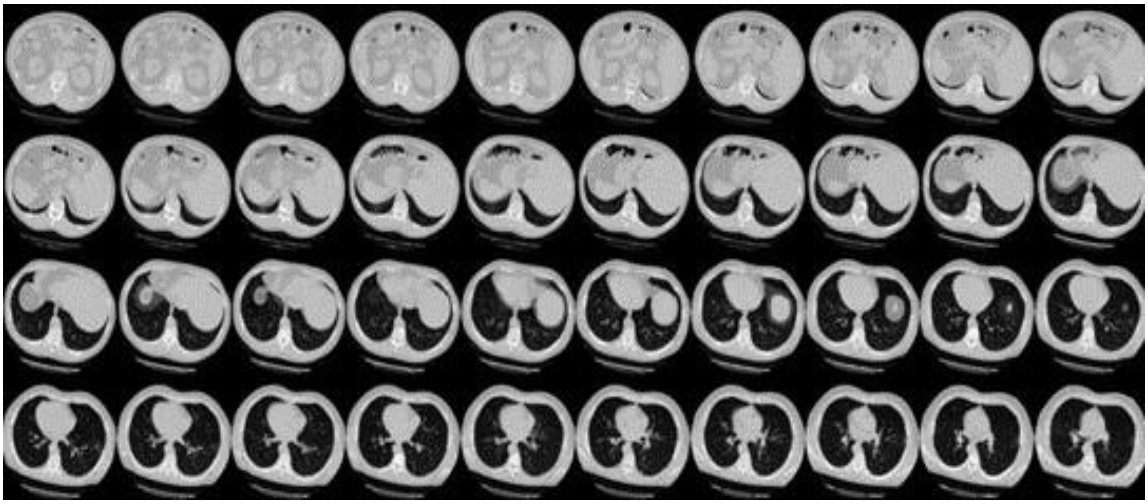
Όπως προαναφέρθηκε, το σύνολο δεδομένων της παρούσας έρευνας, αποτελείται από subjects όπου μπορούν να αναλυθούν σε 2D CT-Scans στο axial επίπεδο. Παρόλα αυτά δεν είναι το μόνο επίπεδο στο οποίο μπορούν να ληφθούν CT-Scans εικόνες. Όπως φαίνεται στην Εικόνα 21, μπορεί να παρατηρηθεί με βάση τη τοποθέτηση του ανθρώπου, πως αναπαρίστανται τα τρία 2D επίπεδα στον τρισδιάστατο χώρο. Αυτά τα επίπεδα ονομάζονται axial, coronal και sagittal.



Εικόνα 20. Τα 3 επίπεδα στο χώρο με βάση το ανθρώπινο σώμα [4]

Πίνακας 2. Κατηγορίες COVID-19

Σοβαρότητα	**CT**	**Clinical Data**	**Decision**
Μηδέν	CT-0 Μη σχετική με COVID-19	-	Ενημερώστε τον θεράποντα ιατρό
Ήπια	CT-1 Αδιαφάνεια. Προσβολή πνευμονικού παρεγχύματος ≤ 25%	A. $t^0 < 38.0^0$C B. *RR* < 2/min C. $SpO_2 > 95\%$	Παρακολούθηση στο σπίτι με χρήση τεχνολογιών τηλεϊατρικής (υποχρεωτική τηλε-παρακολούθηση)
Μέτρια	CT-2 Αδιαφάνεια. Πνευμονική παρεγχυματική προσβολή 25 − 50%.	A. t 0 > 38.5 0C B. RR20 − 30/min C. SpO295%	Παρακολούθηση στο σπίτι από ιατρό πρωτοβάθμιας περίθαλψης
Σοβαρή	CT-3 Αδιαφάνεια γυαλιού. Πνευμονική ενοποίηση. Πνευμονική παρεγχυματική προσβολή 50 − 75%. Η προσβολή των πνευμόνων αυξήθηκε σε 24-48 ώρες κατά 50% στην αναπνευστική ανεπάρκεια ανά μελέτες παρακολούθησης	A. t 0 > 38, 5 ◦C B. RR ≥ 30/min C. SpO2 ≤ 95% D. Partial pressure of oxygen (PaO2) / Fraction of inspired oxygen (FiO2) ≤ 300mmHg (1 mmHg = 0, 133kPa)	Άμεση εισαγωγή σε εξειδικευμένο νοσοκομείο COVID. Σε νοσοκομειακό περιβάλλον: άμεση μεταφορά στη μονάδα εντατικής θεραπείας και ανάνηψης. Επείγουσα Αξονική Τομογραφία (αν δεν έχει γίνει πριν)
Κρίσιμη	CT-4 Διάχυτες αδιαφάνειες από εσμυρισμένο γυαλί με ενοποιήσεις και δικτυωτές αλλαγές. Υδροθώρακας (αμφίπλευρα, περισσότερο αριστερά). Πνευμονική παρεγχυματική προσβολή ≥ 75%.	Ανεπάρκεια πολλαπλών οργάνων, αναπνευστική ανεπάρκεια.	Επείγουσα ιατρική περίθαλψη. Άμεση εισαγωγή σε εξειδικευμένο νοσοκομείο για ασθενείς με διάγνωση COVID-19. Άμεση μεταφορά στη μονάδα εντατικής θεραπείας και ανάνηψης.



Εικόνα 21. 40 slices

Ένα voxel, είναι ένα pixel με όγκο (κύβος μικρών διαστάσεων). Οι εικόνες των slices είναι σε grayscale κλίμακα (μαύρο για την ελάχιστη τιμή και λευκό για τη μέγιστη τιμή) και καθένα από τα pixels αναπαριστά ένα voxel διότι αυτό το 2D slice αντιπροσωπεύει ένα κομμάτι από την 3D εικόνα με συγκεκριμένο βάθος. Ο πίνακας 3D είναι επομένως επίσης ένας πίνακας voxel. Όπως για κάθε πίνακα, μπορούμε να επιλέξουμε συγκεκριμένες τιμές με indexing.

Εστιάζοντας στη λέξη χώρος, ορίζεται από ένα διατεταγμένο σύνολο αξόνων. Για τον τρισδιάστατο χώρο μας, είναι ένα σύνολο τριών ανεξάρτητων αξόνων. Μπορούμε να αποφασίσουμε τι χώρο θέλουμε να χρησιμοποιήσουμε επιλέγοντας αυτούς τους άξονες. Επιλέγουμε την αρχή των αξόνων, την κατεύθυνση των αξόνων και τις μονάδες τους.

- Αρχικά ορίζεται ένα σύνολο τριών ορθογώνιων αξόνων σαρωτή. Η αρχή των αξόνων βρίσκεται στο ισόκεντρο του μαγνήτη. Αυτή η συντεταγμένη θα βρίσκεται στο (0,0,0) στο χώρο αναφοράς μας και οι τρείς άξονες εισέρχονται από το ισόκεντρο.
- Οι μονάδες και για τους τρεις άξονες είναι χιλιοστά.
- Ας φανταστούμε έναν παρατηρητή να στέκεται πίσω από τον σαρωτή και να κοιτάζει μέσα από τη μαγνητική οπή προς το τέλος του κρεβατιού του σαρωτή. Ας φανταστούμε μια γραμμή του ταξιδεύει προς τον παρατηρητή μέσω του κέντρου της οπής μαγνήτη, παράλληλη προς το κρεβάτι, με το σημείο μηδέν στο ισόκεντρο του μαγνήτη και τις θετικές τιμές πιο κοντά στον παρατηρητή. Αυτή η γραμμή ονομάζεται άξονας οπής σαρωτή.

- Δημιουργούμε μια γραμμή που κινείται από το δάπεδο του δωματίου του σαρωτή προς τα πάνω μέσω του ισοκεντρικού μαγνήτη προς την οροφή, σε ορθή γωνία προς τον άξονα της οπής του σαρωτή. Το 0 βρίσκεται στο ισόκεντρο και οι θετικές τιμές είναι προς το ανώτατο όριο. Ας ονομάσουμε αυτό τον άξονα σαρωτή-δαπέδου/οροφής.
- Δημιουργούμε μια ευθεία σε ορθή γωνία με τις άλλες δύο ευθείες γραμμές, που να κινείται από τα αριστερά του παρατηρητή, παράλληλα με το πάτωμα, και μέσω του ισοκεντρικού μαγνήτη προς τα δεξιά του παρατηρητή. Το 0 βρίσκεται στο ισόκεντρο και οι θετικές τιμές είναι δεξιά. Ας ονομάσουμε αυτό τον άξονα σαρωτής-αριστερά/δεξιά [3].

Στην Εικόνα 21, απεικονίζονται τα τρία επίπεδα στο χώρο με βάση το ανθρώπινο σώμα στα οποία στηρίζονται οι Αξονικές Τομογραφίες.

4.2 Πειράματα

Εν’ όψη της πρόβλεψης του COVID-19 μέσω των subjects όπου περιλαμβάνουν 3D CT-Scans, στο παρόν Κεφάλαιο παρουσιάζονται τα αποτελέσματα των υλοποιήσεων όπου έχουν πραγματοποιηθεί.

4.2.1 Πείραμα 1 - Convolutional Neural Network from scratch

Στο Πείραμα 1 κατασκευάστηκε το μοντέλο ταξινόμησης βασισμένο στο Συνελικτικό Νευρωνικό Δίκτυο που προσφέρει η PyTorch. Παρακάτω παρουσιάζονται τα επίπεδα από τα οποία αποτελείται το Συνελικτικό Νευρωνικό Δίκτυο:

- Convolutional Layer 1:
    - Το πρώτο επίπεδο είναι το Συνελικτικό Επίπεδο όπου παίρνει ως είσοδο έναν 3D πίνακα (tensor) με ένα κανάλι (channel) και έχει ως έξοδο έναν 3D πίνακα με 32 κανάλια. Το Συνελικτικό αυτό Επίπεδο εφαρμόζει μία 3D Συνελικτική Διαδικασία με μέγεθος φίλτρου (kernel size) (3,3,3) και με σκοπό τον υπολογισμό γινομένου της εικόνας εισόδου. Επίσης ορίζεται Stride ίσο με (1,1,1), δηλαδή το φίλτρο εφαρμόζεται σε κάθε τμήμα της εικόνας εισόδου με βήμα (1,1,1). Το padding τίθεται ίσο με 0, που σημαίνει ότι δεν εφαρμόζεται padding στην εικόνα εισόδου πριν από την διαδικασία της συνέλιξης.
    - Έπειτα από την διαδικασία της συνέλιξης, ο πίνακας εξόδου περνάει από μία LeakyReLU συνάρτηση ενεργοποίησης (activation function) η οποία εισάγει

μια μικρή αρνητική κλίση στην τυπική συνάρτηση ReLU για την πρόληψη νεκρών νευρώνων.

- Έπειτα, η έξοδος περνάει από ένα 3D max pooling επίπεδο με μέγεθος φίλτρου (kernel size) (2,2,2), μειώνοντας κατά αυτό τον τρόπο τις χωρικές διαστάσεις του πίνακα εξόδου κατά 2 σε καθεμιά από τις 3 διαστάσεις. Η ίδια Συνελικτική Διαδικασία δηλαδή με το μέγεθος του φίλτρου, τον αριθμό Stride, την LeakyReLU συνάρτηση ενεργοποίησης με τη συγκεκριμένη αρνητική κλίση, και το επίπεδο Max Pooling με τις συγκεκριμένες παραμέτρους, εφαρμόζονται και στα επόμενα δύο Συνελικτικά Επίπεδα.

Τα επόμενα 2 Συνελικτικά Επίπεδα αποτελούνται από ένα Συνελικτικό Επίπεδο στο οποίο αλλάζουν οι διαστάσεις εισόδου και εξόδου, από το επίπεδο ReLU και από το Max Pooling επίπεδο (με Kernel size 2×2×2 και Stride ίσο με 2). Καθώς τα επίπεδα ReLU και Max Pooling είναι τα ίδια με αυτά που περιεγράφηκαν στο Convolutional Layer 1, στα επίπεδα που ακολουθούν, Convolutional Layer 2 και Convolutional Layer 3, περιγράφονται μόνο τα Συνελικτικά Επίπεδα.

- Convolutional Layer 2:

  Το δεύτερο επίπεδο είναι το Συνελικτικό Επίπεδο όπου παίρνει ως είσοδο έναν 3D πίνακα με 32 κανάλια και συνεπώς 32 χάρτες χαρακτηριστικών (feature maps), και παράγει 64 κανάλια εξόδου. Αυξάνοντας τον αριθμό των καναλιών εξόδου στα επόμενα Συνελικτικά Επίπεδα, το Νευρωνικό Δίκτυο μαθαίνει όλο και περισσότερο πολύπλοκα χαρακτηριστικά από τα δεδομένα εισόδου.

- Convolutional Layer 3:

  Το τρίτο επίπεδο είναι το Συνελικτικό Επίπεδο όπου παίρνει ως είσοδο έναν 3D πίνακα με 64 κανάλια και συνεπώς 64 χάρτες χαρακτηριστικών (feature maps), και παράγει 128 κανάλια εξόδου.

- Flattening επίπεδο:
  - Ο ισοπεδωμένος (Flattened) πίνακας τροφοδοτείται ως είσοδος στο Fully Connected Layer για την ταξινόμηση της εικόνας.

- Fully Connected Layer:

- Επίπεδο Dropout: Το Dropout επίπεδο με p=0,1 δηλαδή με πιθανότητα 0,1 ορίζει την απόρριψη των στοιχείων του πίνακα εισόδου κατά τη διάρκεια της εκπαίδευσης. Αυτό σημαίνει πως κατά τη διάρκεια του forward pass του Νευρωνικού Δικτύου, το επίπεδο Dropout θα απορρίπτει τυχαία το 10% των τιμών του πίνακα που μεταβιβάζεται σε αυτό. Ο σκοπός της χρήσης του Dropout είναι να αποτραπεί η υπερπροσαρμογή (overfitting) στο μοντέλο, με την τυχαία απόρριψη ορισμένων από τους νευρώνες του δικτύου κατά τη διάρκεια της εκπαίδευσης, και συνεπώς αποτρέπεται οποιοσδήποτε νευρώνας να γίνει πολύ σημαντικός για την τελική πρόβλεψη. Είναι σημαντικό να σημειωθεί ότι το Dropout εφαρμόζεται μόνο κατά τη διάρκεια της εκπαίδευσης και όχι κατά τη διάρκεια δοκιμών (testing). Κατά τη διάρκεια της εξαγωγής συμπερασμάτων, το πλήρες σύνολο των νευρώνων χρησιμοποιείται για την πραγματοποίηση προβλέψεων.

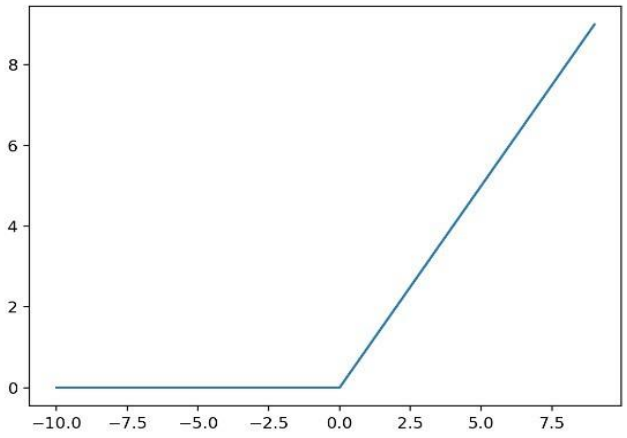
- Πρώτο Επίπεδο Γραμμικού Συνδέσμου: Το πρώτο επίπεδο Linear είναι ένα πλήρως συνδεδεμένο επίπεδο. Παίρνει ως είσοδο την έξοδο από τα προηγούμενα Συνελικτικά Επίπεδα. Ο αριθμός των χαρακτηριστικών εισόδου είναι 2.621.440, το οποίο καθορίζεται από την έξοδο των προηγούμενων Συνελικτικών Επιπέδων. Ο αριθμός των χαρακτηριστικών εξόδου είναι 128, πράγμα που σημαίνει ότι αυτό το επίπεδο θα έχει 128 νευρώνες. Σκοπός αυτού του επιπέδου είναι να εφαρμόσει μια γραμμική μετασχηματιστική συνάρτηση στα δεδομένα εισόδου, ακολουθούμενη από μια συνάρτηση ενεργοποίησης.

- Ενεργοποίηση ReLU: Το επίπεδο ReLU εφαρμόζει την συνάρτηση ενεργοποίησης Rectified Linear Unit (ReLU) στην έξοδο του πρώτου επιπέδου γραμμικού συνδέσμου. Η ενεργοποίηση ReLU θέτει όλες τις αρνητικές τιμές ίσες με το μηδέν και διατηρεί τις θετικές τιμές αναλλοίωτες. Εισάγει μη γραμμικότητα στο δίκτυο και βοηθά το μοντέλο να μάθει περίπλοκα πρότυπα και αναπαραστάσεις. Στην Εικόνα 22 απεικονίζεται η ReLU συνάρτηση ενεργοποίησης.

- Δεύτερο Επίπεδο Γραμμικού Συνδέσμου: Το δεύτερο Linear επίπεδο είναι το τελικό επίπεδο του επιπέδου Fully Connected. Παίρνει την 128-διάστατη έξοδο από την προηγούμενη ενεργοποίηση ReLU και παράγει την τελική έξοδο. Ο αριθμός των χαρακτηριστικών εξόδου σε αυτό το επίπεδο καθορίζεται από τον αριθμό των κλάσεων, που αντιπροσωπεύει τον αριθμό των κατηγοριών στο πρόβλημα ταξινόμησης που το Συνελικτικό Δίκτυο έχει σχεδιαστεί να λύσει. Οι τιμές στην έξοδο

αυτού του επιπέδου χρησιμοποιούνται για την κατηγοριοποίηση, και συνήθως εφαρμόζεται μια συνάρτηση softmax για να ληφθούν οι πιθανότητες των κατηγοριών.

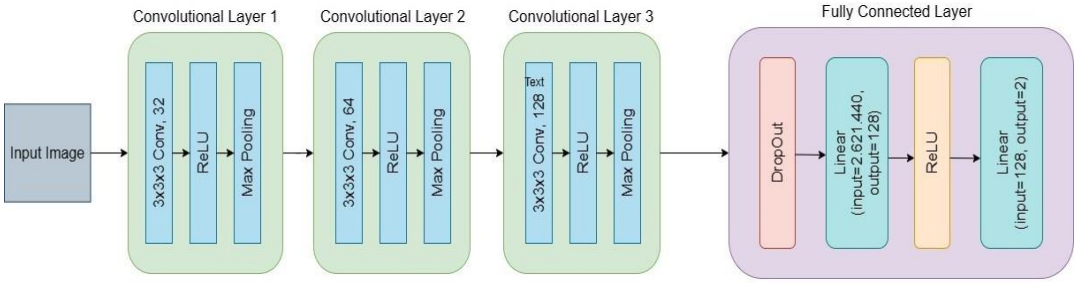
Στην Εικόνα 23 και Εικόνα 24 παρουσιάζονται τα επίπεδα από τα οποία αποτελείται το μοντέλο που κατασκευάστηκε κατά το Πείραμα 1.



Εικόνα 22. Συνάρτηση ενεργοποίησης ReLU

```
Simple3DCNN(
  (conv_layer): Sequential(
    (0): Conv3d(1, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): ReLU(inplace=True)
    (2): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv3d(32, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv3d(64, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (7): ReLU(inplace=True)
    (8): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layer): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=2621440, out_features=128, bias=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=128, out_features=2, bias=True)
  )
)
```

Εικόνα 23. Πείραμα 1 - Τα επίπεδα του μοντέλου που κατασκευάστηκε

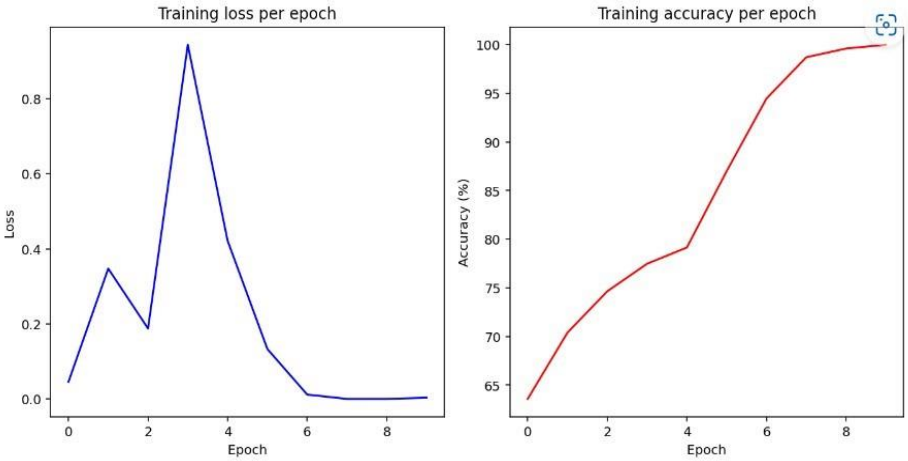


Εικόνα 24. Αρχιτεκτονική του Συνελικτικού Δικτύου του Πειράματος 1

Σε αυτό το Πείραμα, το σύνολο δεδομένων χωρίστηκε σε 70% των subjects για training, το 15% για validation και το 15% για test. Κατά την εκπαίδευση ορίστηκε batch size ίσο με 8, και εφαρμόστηκε τροποποίηση του μεγέθους των εικόνων σε 512×512×40. Το Συνελικτικό Νευρωνικό Δίκτυο εκπαιδεύτηκε σε 10 epochs και με learning rate ίσο με 0.001. Στον Πίνακα 3 φαίνονται οι τιμές Accuracy που καταγράφηκαν σε καθένα από τα 10 epochs στο training και validation set. Η τιμή Accuracy στο test set καταγράφηκε ίση με 74.09%. Στην Εικόνα 25 φαίνονται τα διαγράμματα τιμών Accuracy και Loss για κάθε epoch στο training set. Από το διάγραμμα αυτό μπορεί να παρατηρηθεί ότι η τιμή Accuracy αυξάνεται συνεχώς με την αύξηση των epochs. Αντίθετα, υπάρχει μία αυξομείωση της τιμής Loss έως και το 4ο epoch. Από το 4ο epoch και έπειτα παρατηρείται συνεχής μείωση της τιμής Loss.

Πίνακας 3. Πείραμα 1 - Τιμές Accuracy στο Training και Validation Set

Epochs	Training Set	Validation Set
Epoch 1	63.57%	79.51%
Epoch 2	70.39%	79.51%
Epoch 3	74.64%	78.91%
Epoch 4	77.47%	78.31%
Epoch 5	79.15%	63.85%
Epoch 6	87.00%	73.49%
Epoch 7	94.46%	77.71%
Epoch 8	98.71%	78.31%
Epoch 9	99.61%	73.49%
Epoch 10	100.0%	74.09%



Εικόνα 25. Πείραμα 1 - Τιμές Accuracy και Loss

4.2.2 Πείραμα 2 - Convolutional Neural Network from scratch με χρήση Cross Validation

Στο Πείραμα 2 εφαρμόστηκε η Τεχνική του Cross Validation. Οι εικόνες εισόδου έχουν μέγεθος 256×256×40. Το Συνελικτικό Νευρωνικό Δίκτυο αποτελείται από τα ακόλουθα επίπεδα:

- Συνελικτικό Νευρωνικό Δίκτυο τριών διαστάσεων με ένα κανάλι εισόδου και 32 κανάλια εξόδου και τιμή padding ίση με 1.
- Επίπεδο με τη συνάρτηση ReLU.
- Επίπεδο Max Pooling με kernel size ίσο με 2 και Stride ίσο με 2.

Η σειρά αυτή των επιπέδων επαναλαμβάνεται άλλες δύο φορές, με τα ίδια layers και τις ίδιες παραμέτρους στη ReLU και στο Max Pooling, με τη διαφορά των παραμέτρων εισόδου και εξόδου στο Συνελικτικό Επίπεδο. Έπειτα ακολουθεί το Flattening επίπεδο, και στη συνέχεια το Fully Connected επίπεδο όπως αναλύεται με τα παρακάτω επίπεδα:

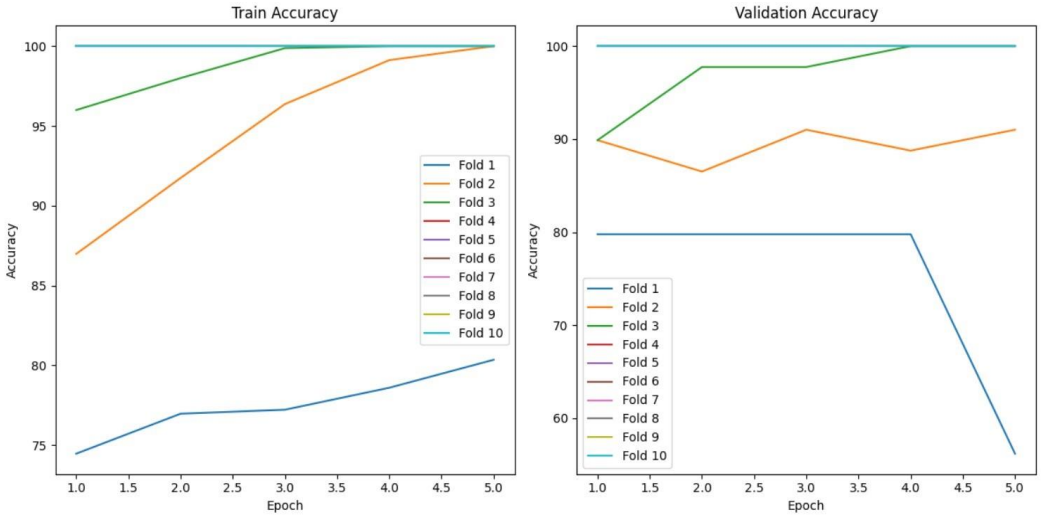
- Το επίπεδο Dropout, με πιθανότητα 0.1.
- Το επίπεδο Linear, με 655.360 χαρακτηριστικά εισόδου και 128 χαρακτηριστικά εξόδου.
- Το επίπεδο με τη συνάρτηση ReLU.
- Το επίπεδο Linear με παράμετρο εισόδου ίση με 128, και παράμετρο εξόδου ίση με 2 ίση με τον αριθμό των κλάσεων.

```
Simple3DCNN(
  (conv_layer): Sequential(
    (0): Conv3d(1, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (1): ReLU(inplace=True)
    (2): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv3d(32, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv3d(64, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1), padding=(1, 1, 1))
    (7): ReLU(inplace=True)
    (8): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layer): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=655360, out_features=128, bias=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=128, out_features=2, bias=True)
  )
)
```

Εικόνα 26. Πείραμα 2 - Αρχιτεκτονική μοντέλου που κατασκευάστηκε

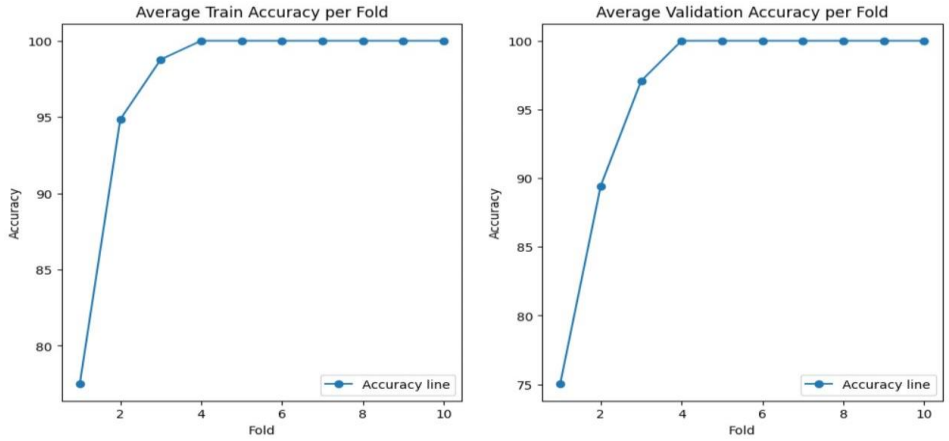
Η εκπαίδευση του μοντέλου έγινε σε 10 Folds Cross Validation. Δεδομένου ότι το σύνολο δεδομένων έχει 1,110 εικόνες, από αυτές το 80% (δηλαδή 888) χρησιμοποιήθηκε για το στάδιο της εκπαίδευσης και το 20% (δηλαδή 222) για το στάδιο του test. Κατά το στάδιο της εκπαίδευσης, ο διαχωρισμός των δεδομένων σε K τμήματα εφαρμόστηκε με τη χρήση της βιβλιοθήκης *sklearn.model_selection.KFold* της *scikitlearn*. Με τη χρήση της παραπάνω συνάρτησης επιτυγχάνεται ο διαχωρισμός του συνόλου δεδομένων σε training και test σε K διαδοχικά κομμάτια (χωρίς shuffling από προεπιλογή). Κάθε κομμάτι χρησιμοποιείται, στη συνέχεια, μία φορά ως validation, ενώ τα εναπομείναντα K-1 κομμάτια αποτελούν το σύνολο εκπαίδευσης (training set).

Κάθε ένα από τα 10 Folds εκπαιδεύτηκε σε 5 epochs, και με learning rate ίσο με 0.001. Επίσης, ο αριθμός batch size ορίστηκε ίσος με 4. Η τιμή Accuracy στο test set όπου καταγράφηκε είναι ίση με 76.57%. Στην Εικόνα 27, απεικονίζονται οι τιμές Accuracy στο training set (αριστερά της εικόνας) και στο validation set (δεξιά της εικόνας). Οι τιμές Accuracy αναπαρίστανται για καθένα από τα 10 Folds. Αξίζει να σημειωθεί ότι στα 5 τελευταία Folds η τιμή Accuracy είναι ίση με 100%, και για αυτό τον λόγο μπορούν να γίνουν διακριτές οι 5 από τις 10 γραμμές. Μία ακόμη παρατήρηση είναι ότι σε κάθε επόμενο epoch καταγράφονται υψηλότερες τιμές Accuracy από το προηγούμενό του.



Εικόνα 27. Πείραμα 2 - Τιμές Accuracy για κάθε Fold στο training και στο validation

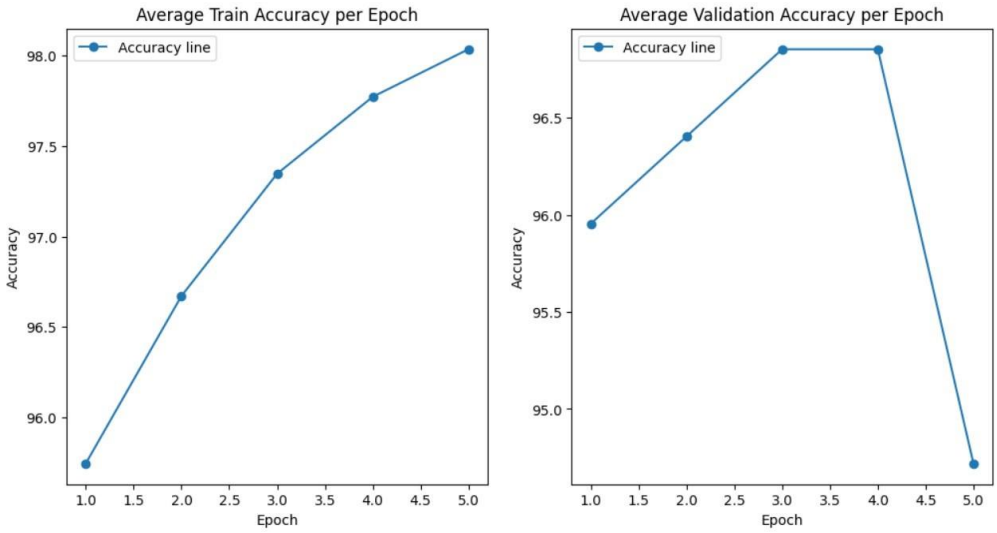
Στην Εικόνα 28, παρουσιάζονται οι μέσοι όροι των τιμών Accuracy στο training set (αριστερά της εικόνας) και στο validation set (δεξιά της εικόνας), για καθένα από τα 10 Folds. Από αυτή την εικόνα μπορεί να διαπιστωθεί ότι από το 4ο Fold και έπειτα, οι τιμές Accuracy είναι ίσες με 100%.



Εικόνα 28. Πείραμα 2 - Τιμές Accuracy για το training και το validation set για κάθε Fold

Στην Εικόνα 29, παρουσιάζονται οι μέσοι όροι των τιμών Accuracy στο training set (αριστερά της εικόνας) και validation set (δεξιά της εικόνας), για καθένα από τα 5 epochs. Από την εικόνα αυτή μπορεί να παρατηρηθεί ότι οι τιμές Accuracy για όλα τα epochs παραμένουν σε υψηλές τιμές, με αυξητική τάση. Εξαίρεση στην αυξητική τάση αποτελεί το Accuracy στο validation set κατά το τελευταίο epoch.

Η τιμή accuracy που καταγράφηκε στο test set είναι ίση με 76.57%.



Εικόνα 29. Πείραμα 2 - Τιμές Accuracy για το training και το validation set για κάθε epoch

4.2.3 Πείραμα 3 - MC3-18 pretrained Convolutional Neural Network

Επόμενο μοντέλο που εφαρμόστηκε είναι το Συνελικτικό Νευρωνικό δίκτυο όπου περιέχει το προ-εκπαιδευμένο μοντέλο MC3_18 του PyTorch. Το μοντέλο MC3-18 έχει προ-εκπαιδευτεί στο σύνολο δεδομένων Kinetics [43]. Το σύνολο δεδομένων Kinetics είναι ένα μεγάλο σύνολο δεδομένων, με υψηλής ποιότητας για αναγνώριση ανθρώπινης δράσης στα Video. Αυτό το σύνολο δεδομένων, περιέχει περίπου 500 χιλιάδες video clips και καλύπτει 600 κλάσεις ανθρώπινης δράσης με τουλάχιστον 600 clips για κάθε κλάση. Κάθε video clip διαρκεί περίπου 10 δευτερόλεπτα και κατηγοριοποιείται σε μία μοναδική κλάση. Το MC3_18 βασίζεται στο VideoResNet.

Αρχικά, για καλύτερη κατανόηση του VideoResNet γίνεται επεξήγηση του Συνελικτικού Δικτύου ResNet. Το Residual Network (ResNet) είναι ένα Deep Learning μοντέλο, όπου χρησιμοποιείται σε εφαρμογές υπολογιστικής όρασης. Αποτελεί μία αρχιτεκτονική Συνελικτικού Νευρωνικού Δικτύου που έχει σχεδιαστεί για να υποστηρίζει εκατοντάδες ή χιλιάδες Συνελικτικά Επίπεδα. Άλλες αρχιτεκτονικές Συνελικτικών Νευρωνικών Δικτύων δεν μπορούν να κλιμακωθούν σε μεγάλο αριθμό επιπέδων, γεγονός που είχε ως αποτέλεσμα περιορισμένη απόδοση. Ωστόσο, κατά την προσθήκη περισσότερων επιπέδων, οι ερευνητές αντιμετώπισαν το πρόβλημα εξαφανιζόμενης κλίσης (vanishing gradient).

Τα Νευρωνικά Δίκτυα εκπαιδεύονται μέσω της διαδικασίας του backpropagation, όπου στηρίζεται στο gradient descent. Εάν υπάρχουν πολλά επίπεδα, οι επαναλαμβανόμενοι πολλαπλασιασμοί μειώνουν αρκετά την κλίση (gradient) μέχρι τελικά να

εξαφανιστεί. Το γεγονός αυτό έχει ως αποτέλεσμα να παρουσιάζεται κορεσμός και επιδείνωση στην απόδοση με κάθε επιπλέον επίπεδο που προστίθεται.

Το ResNet παρέχει μία καινοτόμο λύση στο πρόβλημα της εξαφάνισης της κλίσης (gradient) η οποία είναι η παράβλεψη συνδέσεων (skip connections). Το ResNet πραγματοποιεί πολλαπλές αντιστοιχίσεις (Συνελικτικά Επίπεδα που δεν κάνουν τίποτα στην αρχή), παρακάμπτει αυτά τα επίπεδα και επαναχρησιμοποιεί τις ενεργοποιήσεις του προηγούμενου επιπέδου. Η παράλειψη επιταχύνει την αρχική εκπαίδευση συμπιέζοντας το δίκτυο σε λιγότερα επίπεδα.

Στη συνέχεια, όταν το δίκτυο ξανά εκπαιδεύεται, όλα τα επίπεδα επεκτείνονται και τα υπόλοιπα μέρη του δικτύου - γνωστά ως υπολειπόμενα μέρη (residual parts) - επιτρέπεται να εξερευνήσουν περισσότερο τον χώρο χαρακτηριστικών της εικόνας εισόδου. Τα περισσότερα ResNet μοντέλα παρακάμπτουν δύο ή τρία επίπεδα κάθε φορά με μη γραμμικότητα και batch normalization ενδιάμεσα. Πιο προηγμένες αρχιτεκτονικές ResNet, μπορούν να μάθουν την παράλειψη βαρών, τα οποία καθορίζουν δυναμικά τον αριθμό των επιπέδων που πρέπει να παραλειφθούν [42].

Έχοντας αναλύσει το ResNet, γίνεται επεξήγηση του VideoResNet, πάνω στο οποίο βασίζεται το MC3_18. Το Video ResNet είναι ένα Συνελικτικό Νευρωνικό Δίκτυο που χρησιμοποιείται σε προβλήματα ταξινόμησης σε βίντεο. Βασίζεται στο αρχικό ResNet αρχιτεκτονικής που αναπτύχθηκε για την επίλυση του προβλήματος για ταξινόμηση εικόνας.

Η βασική ιδέα του Video ResNet είναι να χρησιμοποιήσει την εισαγωγή μιας επιπλέον διάστασης που αντιπροσωπεύει τον χρόνο στα βίντεο. Συνεπώς, κάθε πλαίσιο βίντεο προβάλλεται σε έναν διαφορετικό χρόνο, πράγμα που επιτρέπει στο δίκτυο να ανιχνεύει κινούμενα αντικείμενα και να αναγνωρίζει δράσεις.

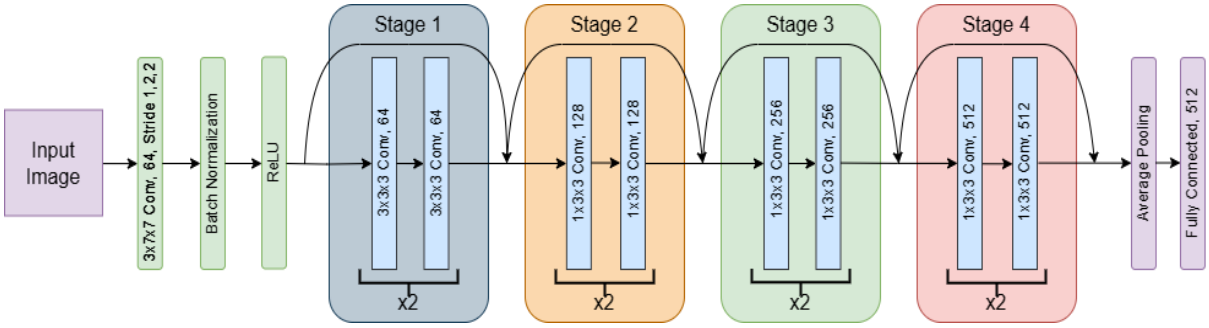
Το Video ResNet χρησιμοποιεί μια σειρά από 3D Συνελικτικά Επίπεδα για την εξαγωγή χαρακτηριστικών από τα βίντεο. Κατόπιν, ακολουθείται από ένα πλήρως συνδεδεμένο επίπεδο που επιτρέπει στο δίκτυο να προβλέπει την κλάση πάνω στη συγκεκριμένη δράση.

Η αρχιτεκτονική του Video ResNet είναι παρόμοια με αυτήν του αρχικού ResNet, με την προσθήκη επιπλέον διάστασης για το χρόνο στα Συνελικτικά Επίπεδα. Μια άλλη διαφορά του Video ResNet από το ResNet είναι η χρήση του 3D pooling αντί του 2D pooling που χρησιμοποιείται στο ResNet. Αυτό επιτρέπει στο δίκτυο να καταγράφει τόσο χωρικά όσο και χρονικά χαρακτηριστικά στο βίντεο εισόδου.

Τέλος, η έξοδος του τελευταίου Συνελικτικού Επιπέδου διέρχεται από μερικά πλήρως συνδεδεμένα στρώματα για ταξινόμηση. Το τελικό επίπεδο χρησιμοποιεί την ενεργοποίηση softmax για να δημιουργήσει πιθανότητες κλάσης. Συνολικά, το Συνελικτικό Δίκτυο Video ResNet είναι μια ισχυρή αρχιτεκτονική βαθιάς εκμάθησης για εργασίες ταξινόμησης βίντεο και έχει επιτύχει κορυφαίες επιδόσεις σε πολλά σύνολα δεδομένων αναφοράς.

Παρομοίως όπως εφαρμόζεται το Video ResNet σε προβλήματα ταξινόμησης σε βίντεο, με παρόμοιο τρόπο μπορεί να εφαρμοστεί και σε προβλήματα ταξινόμησης 3D εικόνων, όπως ακριβώς έγινε και στην παρούσα διατριβή.

Στην Εικόνα 30 απεικονίζεται η αρχιτεκτονική του Video ResNet μοντέλου που κατασκευάστηκε και χρησιμοποιήθηκε.



Εικόνα 30. Αρχιτεκτονική του Video ResNet Συνελικτικού Νευρωνικού Δικτύου

Σε αυτό το Πείραμα, χρησιμοποιήθηκε το pre-trained μοντέλο MC3-18 της PyTorch. Το σύνολο δεδομένων όπως περιγράφεται και στο Κεφάλαιο 4.1, περιλαμβάνει 856 subjects θετικά στον COVID-19 και 254 subjects αρνητικά στον COVID-19 [Πίνακας 4].

Πίνακας 4. Κατηγοριοποίηση θετικών και αρνητικών περιπτώσεων σε COVID-19

Θετικές περιπτώσεις σε COVID-19	**Αρνητικές περιπτώσεις σε COVID-19**
856	254

Στο σύνολο δεδομένων των εικόνων, πραγματοποιήθηκε διαχωρισμός των δεδομένων σε 70% για training set, 15% για validation set και 15% για test set. Πριν την

φόρτωση των συνόλων αυτών στους Data Loaders, εφαρμόστηκαν μετασχηματισμοί (transformations) στις ιατρικές εικόνες.

- **ToCanonical()**

  Ο πρώτος μετασχηματισμός που εφαρμόστηκε εξασφαλίζει ότι ο προσανατολισμός της εικόνας είναι κανονικοποιημένος, καθιστώντας ευκολότερη τη σύγκριση και την επεξεργασία εικόνων. Εφαρμόζει περιστροφή, αναστροφή ή αλλαγή αξόνων για την ευθυγράμμιση της εικόνας με έναν κοινό προσανατολισμό αναφοράς.

- **RescaleIntensity((0, 1))**

  Ο δεύτερος μετασχηματισμός που εφαρμόστηκε αλλάζει την κλίμακα στις τιμές έντασης της εικόνας σε ένα καθορισμένο εύρος μεταξύ 0 και 1.

- **ZNormalization(masking_method=torchio.ZNormalization.mean)**

  Ο τρίτος μετασχηματισμός εκτελεί κανονικοποίηση Z στις τιμές έντασης της εικόνας. Η κανονικοποίηση Z κανονικοποιεί τις εντάσεις αφαιρώντας τη μέση τιμή και διαιρώντας με την τυπική απόκλιση.

- **Resize((512, 512, 41))**

  Ο τελευταίος μετασχηματισμός που χρησιμοποιήθηκε αλλάζει το μέγεθος της ιατρικής εικόνας, έτσι ώστε να έχει νέο μέγεθος (512, 512, 41).

  Οι τρεις αυτές τιμές αναπαριστούν τις διαστάσεις κατά μήκος των τριών αξόνων:

  - Η πρώτη τιμή (512) υποδεικνύει το νέο μέγεθος της τρισδιάστατης εικόνας κατά μήκος του άξονα x.
  - Η δεύτερη τιμή (512) υποδεικνύει το νέο μέγεθος της τρισδιάστατης εικόνας κατά μήκος του άξονα y.
  - Η τρίτη τιμή (41) υποδεικνύει το νέο μέγεθος της τρισδιάστατης εικόνας κατά μήκος του άξονα z.

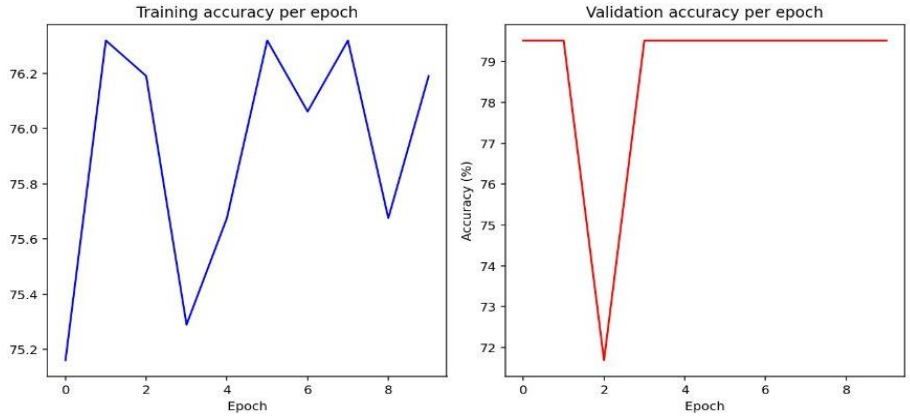
Έπειτα από την προ-επεξεργασία των subjects των εικόνων και την φόρτωσή στους Data Loaders όπως προαναφέρθηκε, χρησιμοποιήθηκε η βάση του pre-trained μοντέλου MC3-18 της PyTorch.

Στο παρόν Πείραμα, ορίστηκε μια νέα αρχιτεκτονική Νευρωνικού Δικτύου και προσαρμόστηκε κατάλληλα στη συγκεκριμένη εργασία αλλάζοντας το πρώτο Συνελικτικό Επίπεδο καθώς και το Πλήρες Συνδεδεμένο Επίπεδο (Fully Connected Layer). Πιο αναλυτικά, εφαρμόστηκε τροποποίηση του πρώτου 3D Συνελικτικού Επιπέδου έτσι ώστε το kernel size να είναι ίσο με (3, 7, 7), το stride ίσο με (1, 2, 2) και το padding ίσο με (1, 3, 3). Επιπλέον, έγινε προσθήκη του Fully Connected Layer με input size ίσο με 512 και αριθμό κλάσεων 2. Στα υπόλοιπα επίπεδα δεν εφαρμόστηκαν τροποποιήσεις, που σημαίνει πως προέρχονται αυτούσια από το προ-εκπαιδευμένο μοντέλο MC3-18.

Η προσέγγιση αυτή, επιτρέπει την αξιοποίηση των προ-εκπαιδευμένων βαρών για την εξαγωγή χαρακτηριστικών, ενώ παράλληλα γίνεται προσαρμογή της αρχιτεκτονικής ώστε να ταιριάζει κατάλληλα στο συγκεκριμένο πρόβλημα. Για την εκπαίδευση του μοντέλου χρησιμοποιούνται 10 epochs. Επιπλέον, η τιμή learning rate τέθηκε ίση με 0.001, και το batch size ϊσο με 8. Στον Πίνακα 5 παρουσιάζονται οι τιμές Accuracy στο training και στο validation set για καθένα από τα epochs. Επιπλέον στη Εικόνα 31, απεικονίζονται μέσω γραφικών παραστάσεων οι τιμές Accuracy στο training και στο validation set για καθένα από τα epochs. Η τιμή Accuracy στο training set παρουσιάζει μικρές διακυμάνσεις μεταξύ 75.16% και 76.31%. Η τιμή Accuracy στο validation set είναι σταθερή στο 79.51%, με εξαίρεση το 3ο epoch στο οποίο η τιμή Accuracy είναι ίση με 71.68%. Η τιμή Accuracy στο test set είναι ίση με 79.51%.

Πίνακας 5. Τιμές Accuracy στο Training και Validation Set

Epochs	Training Set	Validation Set
Epoch 1	75.16%	79.51%
Epoch 2	76.31%	79.51%
Epoch 3	76.19%	71.68%
Epoch 4	75.28%	79.51%
Epoch 5	75.67%	79.51%
Epoch 6	76.31%	79.51%
Epoch 7	76.06%	79.51%
Epoch 8	76.31%	79.51%
Epoch 9	75.67%	79.51%
Epoch 10	76.19%	79.51%



Εικόνα 31. Πείραμα 3 - Τιμές Accuracy κατά το training και το validation set

4.2.4 Σύγκριση Πειραμάτων

Στον Πίνακα 6 παρουσιάζονται τα 3 Πειράματα που εφαρμόστηκαν στην παρούσα διατριβή για καλύτερη σύγκριση μεταξύ τους. Μέσω του Πίνακα αυτού μπορεί να διαπιστωθεί ότι η χρήση της τεχνικής 10 Fold Cross Validation προσέφερε αύξηση της τιμής Accuracy της τάξεως των δυόμιση περίπου μονάδων σε σχέση με τον απλό διαχωρισμό των δεδομένων. Επιπρόσθετα, η χρήση του προ-εκπαιδευμένου μοντέλου MC3-18 της PyTorch προσέφερε αύξηση της τιμής Accuracy της τάξεως των τριών περίπου μονάδων.

Πίνακας 6. Σύγκριση τιμών Accuracy μεταξύ των τριών Πειραμάτων

Πείραμα	**Accuracy**
Πείραμα 1 - CNN from scratch	74.09 %
Πείραμα 2 - CNN from scratch με 10 Fold Cross Validation	76.57 %
Πείραμα 3 - MC3-18 pretrained	79.51 %

Κεφάλαιο 5: Συμπεράσματα

Στην παρούσα διατριβή, η οποία αφορά την πρόβλεψη εμφάνισης ή όχι της νόσου από COVID-19, αποτυπώνοντας και έχοντας επεξεργαστεί Αξονικές Τομογραφίες (CT-Scans), υπήρξαν διάφορες προκλήσεις και εμπόδια. Το σύνολο δεδομένων των Αξονικών Τομογραφιών, MOSMED το οποίο συλλέχθηκε από νοσοκομεία της Μόσχας στη Ρωσία, αποτελείται από 3D ιατρικές εικόνες, οι οποίες είναι αποθηκευμένες σε μορφή NIfTI. Λόγω της τρισδιάστατης μορφής των εικόνων, και της χρήσης των 3D Συνελικτικών Νευρωνικών Δικτύων, στην παρούσα διατριβή έγιναν δοκιμές και προσπάθειες επίλυσης των εμποδίων που συναντήθηκαν.

Μέρος των προκλήσεων αυτών, αφορά την υψηλή απαίτηση χώρου και μνήμης για την εκπαίδευση των μοντέλων. Μία επιπλέον πρόκληση, είναι το γεγονός ότι το πλήθος των δεδομένων δεν ήταν ιδιαίτερα μεγάλο για την εκπαίδευση Deep Learning μοντέλου που επιδίωκε τα καλύτερα δυνατά αποτελέσματα. Συγκεκριμένα χρησιμοποιήθηκαν 1,110 περιπτώσεις ασθενών με μέγεθος εικόνων (512, 512, (36-41)), για κάθε ασθενή. Συνεπώς, το γεγονός ότι οι περιπτώσεις δεν ήταν υπερπληθείς, συντέλεσε στην μεγαλύτερη προσπάθεια και επίτευξη υψηλής τιμής Accuracy.

Μία επιπρόσθετη παρατήρηση, είναι ότι το σύνολο δεδομένων των εικόνων είναι imbalanced. Πιο συγκεκριμένα, η κατανομή των κλάσεων (έπειτα από ανακατάταξη των 5 κλάσεων σε 2) στις δύο κλάσεις, δεν είναι ίση (ή περίπου ίση). Συγκεκριμένα, με 254 περιπτώσεις στην κλάση 0 (Αρνητικές στον COVID-19), και 856 περιπτώσεις στην κλάση 1 (Θετικές στον COVID-19), το σύνολο των δεδομένων δεν μπορεί να θεωρηθεί ισοκατανεμημένο. Για την διαχείριση της κατάστασης αυτής έγινε πειραματική δοκιμή, με χρήση της τεχνικής Upsampling στα δεδομένα με τη βοήθεια της βιβλιοθήκης TorchIO. Παρόλα αυτά, παρατηρήθηκε ότι δεν καταγράφηκε υψηλότερη τιμή ακρίβειας στην πρόβλεψη του μοντέλου έπειτα από την εκπαίδευσή του πάνω στο επαυξημένο σύνολο δεδομένων.

Παρά τις παραπάνω προκλήσεις, για την κατασκευή ενός δομημένου 3D Συνελικτικού Δικτύου το οποίο θα μπορούσε να συνεισφέρει στη λύση του προβλήματος του COVID-19, με τόσες επιπτώσεις στην κοινωνία, εφαρμόστηκε πλήθος τεχνικών και δοκιμάστηκαν διάφορα πειράματα. Χρησιμοποιήθηκε το πακέτο της PyTorch, όπως και η βιβλιοθήκη TorchIO, η οποία δημιουργήθηκε από ομάδα ερευνητών και θεωρείται κατάλληλη για τέτοιου είδους προβλήματα Deep Learning, όπως του προ-

βλήματος της παρούσας διατριβής, για επεξεργασία εικόνων αλλά και βίντεο με έτοιμα πακέτα προ-επεξεργασίας για βελτίωση της απόδοσης των μοντέλων. Η βιβλιοθήκη της TorchIO, έδωσε λύση στην εισαγωγή των εικόνων που ήταν σε μορφή NIfTI, αλλά και στην καλύτερη και πιο αυτοματοποιημένη επεξεργασία τους. Επιπρόσθετα, χρησιμοποιήθηκαν τεχνικές, όπως K-Fold Cross Validation, Επαύξηση Δεδομένων (Data Augmentation) και Επαύξηση Δειγματολειψίας (Upsampling) των δεδομένων.

Στο στάδιο των Πειραμάτων, δοκιμάστηκε πλήθος μοντέλων και προέκυψαν οι ακόλουθες παρατηρήσεις. Αρχικά, κατασκευάστηκε 3D Συνελικτικό Νευρωνικό Δίκτυο, με layers τα οποία δομήθηκαν from scratch. Η τιμή Accuracy που καταγράφηκε στο μοντέλο αυτό είναι ίση με 74.09%. Στη συνέχεια, με αλλαγή της τεχνικής του διαχωρισμού των δεδομένων (80% για training και 20% για test) και αντικατάσταση αυτής με την τεχνική του Cross Validation, επιτεύχθηκε μεγαλύτερη ακρίβεια (Accuracy) στο μοντέλο ίση με 76.5%. Συνεπώς, διαπιστώθηκε η σημαντικότητα της τεχνικής Cross Validation στην εκπαίδευση του μοντέλου, και την επίτευξη μεγαλύτερης ακρίβειας.

Στη συνέχεια, σε επόμενο πείραμα δοκιμάστηκε το προ-εκπαιδευμένο μοντέλο MC318, με προσθήκη σε αυτό επιπρόσθετων επιπέδων, διαπιστώθηκε τιμή ακρίβειας 79.51%, η οποία είναι και η υψηλότερη ακρίβεια που καταγράφηκε μεταξύ όλων των πειραμάτων της παρούσας διατριβής. Συμπερασματικά, διαπιστώθηκε αντίστοιχα η σημαντικότητα της χρήσης των προ-εκπαιδευμένων μοντέλων, τα οποία εμφανίζουν πλήθος πλεονεκτημάτων. Ένα από τα πλεονεκτήματα της χρήσης των προ-εκπαιδευμένων μοντέλων, είναι η εξοικονόμηση σημαντικού χρόνου, λαμβάνοντας γενικά υπόψη τον χρόνο που απαιτεί η εκπαίδευση ενός Deep Learning μοντέλου. Επιπλέον, το γεγονός ότι ένα προ-εκπαιδευμένο μοντέλο έχει εκπαιδευτεί ήδη σε ένα τεράστιο πλήθος δεδομένων, συνεισφέρει στην καλύτερη απόδοσή του και στην μεγαλύτερη ακρίβεια στις προβλέψεις του. Θα μπορούσε να ειπωθεί ότι θα ήταν αδύνατη η χρήση χιλιάδων εικόνων Αξονικών Τομογραφιών, γνώση η οποία περιέχεται στα ήδη προ-εκπαιδευμένα μοντέλα (πάνω σε εικόνες γενικού περιεχομένου). Αυτό συμβαίνει λόγω της δυσκολίας συγκέντρωσης τόσο μεγάλου πλήθους Αξονικών Τομογραφιών, καθώς εμπλέκονται άλλα ζητήματα, όπως των προσωπικών δεδομένων, αλλά και η δυσκολία της εξέτασης τόσου μεγάλου πλήθους εικόνων από ιατρούς.

ΠΑΡΆΡΤΗΜΑ 1

Κώδικας Πειράματος 1

```
# Iterate through rows in the DataFrame and create subjects
for (idx, row) in df.iterrows():
    # Get the path to the study file and concatenate it with the base path
    extended_path = row.loc['study_file']
    filenames = f"{path}{extended_path}"

    # Create a subject using the tio.Subject class
    subject = tio.Subject(
        ct=tio.ScalarImage(filenames),
        label=row.loc['Label']
    )

    # Append the subject to the subjects list
    subjects.append(subject)

# Shuffle the subjects randomly
subjects = shuffle(subjects, random_state=42)

# Calculate the sizes for 70% training, 15% validation, and 15% testing sets
train_size = int(0.7 * len(subjects))
val_size = (len(subjects) - train_size) // 2
test_size = len(subjects) - train_size - val_size

# Split the subjects into training, validation, and testing datasets
train_subjects, val_subjects, test_subjects = torch.utils.data.random_split(subjects,
[train_size, val_size, test_size])

batch_size = 8

# Define data transformations for the training set
train_transform = transforms.Compose([
    tio.transforms.ToCanonical(),  # Convert image to canonical orientation
    tio.transforms.RescaleIntensity((0, 1)),  # Rescale intensity values between 0 and 1
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),  # Apply Z nor-
malization
    tio.transforms.Resize((512, 512, 40))  # Resize the images to (512, 512, 40)
])

# Create a training dataset and dataloader
train_dataset = tio.SubjectsDataset(train_subjects, transform=train_transform)
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

# Define data transformations for the validation set
```

```
val_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((512, 512, 40))
])

# Create a validation dataset and dataloader
val_dataset = tio.SubjectsDataset(val_subjects, transform=val_transform)
val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

# Define data transformations for the test set
test_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((512, 512, 40))
])

# Create a test dataset and dataloader
test_dataset = tio.SubjectsDataset(test_subjects, transform=test_transform)
test_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

class Simple3DCNN(nn.Module):
    def __init__(self, num_classes=2):
        super(Simple3DCNN, self).__init__()

        # Define the convolutional layers
        self.conv_layer = nn.Sequential(
            nn.Conv3d(in_channels=1, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),  # Apply ReLU activation function
            nn.MaxPool3d(kernel_size=2, stride=2),  # Apply 3D max pooling

            nn.Conv3d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(kernel_size=2, stride=2),

            nn.Conv3d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(kernel_size=2, stride=2),
        )

        # Define the fully connected layers
        self.fc_layer = nn.Sequential(
            nn.Dropout(p=0.1),  # Apply dropout for regularization
            nn.Linear(2621440, 128),  # Dimension depends on the output of the conv_layer
            nn.ReLU(inplace=True),  # Apply ReLU activation function
            nn.Linear(128, num_classes),  # Output layer with num_classes classes
        )
```

```
    def forward(self, x):
        x = self.conv_layer(x)  # Pass input through convolutional layers
        x = x.view(x.size(0), -1)  # Flatten the tensor for the fully connected layers
        x = self.fc_layer(x)  # Pass through fully connected layers
        return x

    model = Simple3DCNN(num_classes=2)

# Define the loss function
criterion = nn.CrossEntropyLoss()

# Set the learning rate and initialize the optimizer
learning_r = 0.001
optimizer = torch.optim.Adam(model.parameters(), lr=learning_r)

# Initialize lists to store training and validation metrics
train_losses = []
train_accuracies = []

val_losses = []
val_accuracies = []

test_accuracies = []

# Loop through training epochs
for epoch in range(10):
    running_loss = 0.0
    correct = 0
    total = 0

    # Set the model to training mode
    model.train()

    # Loop through the training data
    for i, data in tqdm(enumerate(train_dataloader)):

        # Extract input data and labels from the batch
        inputs = data["ct"]["data"].float()
        labels = data["label"]

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward pass through the model
        outputs = model(inputs)

        # Calculate the loss
        loss = criterion(outputs, labels)
```

```
        # Backpropagation and optimization
        loss.backward()
        optimizer.step()

        # Update running loss and calculate accuracy
        running_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

        # Print and reset running loss every 20 mini-batches
        if i % 20 == 19:
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 20))
            running_loss = 0.0

    # Append training loss and accuracy to lists
    train_losses.append(loss.item())
    train_accuracies.append(100 * correct / total)

    # Set the model to evaluation mode
    model.eval()

    # Validation loop (no gradient computation)
    with torch.no_grad():
        running_loss = 0.0
        correct = 0
        total = 0

        # Loop through the validation data
        for i, data in enumerate(val_dataloader):

            # Extract input data and labels from the batch
            inputs = data["ct"]["data"].float()
            labels = data["label"]

            # Forward pass through the model
            outputs = model(inputs)

            # Calculate the loss
            loss = criterion(outputs, labels)

            # Update running loss and calculate accuracy
            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        # Append validation loss and accuracy to lists
```

```
        val_losses.append(loss.item())
        val_accuracies.append(100 * correct / total)

# Test data evaluation
with torch.no_grad():
    correct = 0
    total = 0

    # Loop through the test data
    for i, data in enumerate(test_dataloader, 0):

        # Extract input data and labels from the batch
        inputs = data["ct"]["data"].float()
        labels = data["label"]

        # Forward pass through the model
        outputs = model(inputs)

        # Calculate accuracy on test data
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    # Append test accuracy to the list
    test_accuracies.append(100 * correct / total)

# Print the final test accuracy
print('Accuracy of the network on the test images: %d %%' % (100 * correct / total))
```

ΠΑΡΆΡΤΗΜΑ 2

Κώδικας Πειράματος 2

```
# Create an empty list to store subject data
subjects = []

# Loop through rows in the DataFrame
for (idx, row) in df.iterrows():
    # Extract file path information
    extended_path = row.loc['study_file']
    filenames = f"{path}{extended_path}"

    # Create a subject instance with CT image and label
    subject = tio.Subject(
```

```
        ct=tio.ScalarImage(filenames),
        label=row.loc['Label']
    )

    subjects.append(subject)

# Set batch size for data loading
batch_size = 4

# Define transformations for training, validation, and testing data
train_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((256, 256, 40))
])

val_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((256, 256, 40))
])

test_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((256, 256, 40))
])

# Define a Simple3DCNN class for the model
class Simple3DCNN(nn.Module):
    def __init__(self, num_classes=2):
        super(Simple3DCNN, self).__init__()

        self.conv_layer = nn.Sequential(
            nn.Conv3d(in_channels=1, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(kernel_size=2, stride=2),

            nn.Conv3d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(kernel_size=2, stride=2),

            nn.Conv3d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(kernel_size=2, stride=2),
        )
```

```
        self.fc_layer = nn.Sequential(
            nn.Dropout(p=0.1),
            nn.Linear(655360, 128),
            nn.ReLU(inplace=True),
            nn.Linear(128, num_classes),
        )

    def forward(self, x):
        x = self.conv_layer(x)
        x = x.view(x.size(0), -1)
        x = self.fc_layer(x)
        return x

# Create an instance of the Simple3DCNN model
model = Simple3DCNN(num_classes=2)

# Define the loss function, learning rate, and optimizer
criterion = nn.CrossEntropyLoss()
learning_r = 0.001
optimizer = torch.optim.Adam(model.parameters(), lr=learning_r)

# Define a function to train and evaluate the model
def train_and_evaluate_model(model, train_dataloader, val_dataloader, criterion, optimizer,
num_epochs):
    train_losses = []
    train_accuracies = []

    val_losses = []
    val_accuracies = []

    for epoch in range(num_epochs):
        running_loss = 0.0
        correct = 0
        total = 0
        model.train()
        for i, data in tqdm(enumerate(train_dataloader)):
            inputs = data["ct"]["data"].float().cuda()
            labels = data["label"].cuda()

            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
```

```
            correct += (predicted == labels).sum().item()

            if i % 20 == 19:
                print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 20))
                running_loss = 0.0

        train_losses.append(loss.item())
        train_accuracies.append(100 * correct / total)

        model.eval()
        with torch.no_grad():
            running_loss = 0.0
            correct = 0
            total = 0
            for i, data in enumerate(val_dataloader):
                inputs = data["ct"]["data"].float().cuda()
                labels = data["label"].cuda()

                outputs = model(inputs)
                loss = criterion(outputs, labels)

                running_loss += loss.item()
                _, predicted = torch.max(outputs.data, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()

            val_losses.append(loss.item())
            val_accuracies.append(100 * correct / total)

    print('Accuracy of the network on the test images: %d %%' % (100 * correct / total))
    return train_losses, train_accuracies, val_losses, val_accuracies

# Set the number of epochs for training
num_epochs = 5

# Set the number of folds for cross-validation
num_folds = 10

# Split the entire dataset into a training set and a separate test set
train_subjects, test_subjects = train_test_split(subjects, test_size=0.2, random_state=42)

# Create the cross-validator
cross_validator = KFold(n_splits=num_folds, shuffle=True, random_state=42)

# Initialize lists to store training and validation accuracies for each fold
fold_train_losses = []
fold_train_accuracies = []
fold_val_losses = []
fold_val_accuracies = []
```

```
# Loop over the folds
for fold, (train_index, val_index) in enumerate(cross_validator.split(train_subjects)):
    print(f"Fold: {fold + 1}/{num_folds}")

    # Split the training set into training and validation sets for the current fold
    fold_train_subjects = [train_subjects[i] for i in train_index]
    fold_val_subjects = [train_subjects[i] for i in val_index]

    # Create the dataloaders for the current fold
    train_dataset = tio.SubjectsDataset(fold_train_subjects, transform=train_transform)
    train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

    val_dataset = tio.SubjectsDataset(fold_val_subjects, transform=val_transform)
    val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

    # Train and evaluate the model for the current fold
    train_losses, train_accuracies, val_losses, val_accuracies =
train_and_evaluate_model(model,train_dataloader, val_dataloader, criterion, optimizer,
num_epochs)

    # Store the accuracies for the current fold
    fold_train_losses.append(train_losses)
    fold_train_accuracies.append(train_accuracies)
    fold_val_losses.append(val_losses)
    fold_val_accuracies.append(val_accuracies)

# Now create dataloader for the test set
test_dataset = tio.SubjectsDataset(test_subjects, transform=test_transform)
test_dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
test_accuracies = []

# Evaluate the model on the test set
with torch.no_grad():
    correct = 0
    total = 0
    for i, data in enumerate(test_dataloader, 0):
        inputs = data["ct"]["data"].float().cuda()
        labels = data["label"].cuda()

        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
    test_accuracies.append(100 * correct / total)

print(f"Test accuracies: {test_accuracies}")
```

ΠΑΡΆΡΤΗΜΑ 3

Κώδικας Πειράματος 3

```
# Create an empty list to store subjects
subjects = []

# Iterate over rows in DataFrame (assuming 'df' is defined elsewhere)
for (idx, row) in df.iterrows():
    extended_path = row.loc['study_file']
    filenames = f"{path}{extended_path}"

    # Create 'Subject' objects and append them to 'subjects' list
    subject = tio.Subject(
        ct=tio.ScalarImage(filenames),
        label=row.loc['Label']
    )
    subjects.append(subject)

# Shuffle the 'subjects' list with a fixed random state
subjects = shuffle(subjects, random_state=42)

# Calculate sizes for 70% training, 15% validation, and 15% test sets
train_size = int(0.7 * len(subjects))
val_size = (len(subjects) - train_size) // 2
test_size = len(subjects) - train_size - val_size

# Split the subjects into training, validation, and test sets
train_subjects, val_subjects, test_subjects = torch.utils.data.random_split(subjects,
[train_size, val_size, test_size])

batch_size = 8

# Define transformations for training, validation, and test datasets
train_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((512, 512, 41))

val_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((512, 512, 41))
])
```

```
test_transform = transforms.Compose([
    tio.transforms.ToCanonical(),
    tio.transforms.RescaleIntensity((0, 1)),
    tio.transforms.ZNormalization(masking_method=tio.ZNormalization.mean),
    tio.transforms.Resize((512, 512, 41))
])

# Define model (MyMC3_18) and criterion
class MyMC3_18(nn.Module):
    def __init__(self, num_classes, in_channels=1):
        super(MyMC3_18, self).__init__()
        self.model = models.mc3_18(pretrained=True)
        self.model.stem[0] = nn.Conv3d(
            in_channels=in_channels,
            out_channels=64,
            kernel_size=(3, 7, 7),
            stride=(1, 2, 2),
            padding=(1, 3, 3),
            bias=False
        )
        self.model.fc = nn.Linear(512, num_classes)

    def forward(self, x):
        x = self.model(x)
        return x

# Create an instance of your custom model
model = MyMC3_18(num_classes=2)

# Define loss criterion and optimizer
criterion = nn.CrossEntropyLoss()
learning_r = 0.001
optimizer = torch.optim.Adam(model.parameters(), lr=learning_r)

# Initialize lists to store training and validation results
train_losses = []
train_accuracies = []
val_losses = []
val_accuracies = []

# Iterate over training epochs
for epoch in range(10):
    running_loss = 0.0
    correct = 0
    total = 0
    model.train()  # Set the model in training mode
    for i, data in tqdm(enumerate(train_dataloader)):
        inputs = data["ct"]["data"].float()
```

```
        labels = data["label"]

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

        if i % 20 == 19:
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 20))
            running_loss = 0.0

    train_losses.append(loss.item())
    train_accuracies.append(100 * correct / total)

    model.eval()  # Set the model in evaluation mode
    with torch.no_grad():
        running_loss = 0.0
        correct = 0
        total = 0
        for i, data in enumerate(val_dataloader):
            inputs = data["ct"]["data"].float()
            labels = data["label"]

            outputs = model(inputs)
            loss = criterion(outputs, labels)

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        val_losses.append(loss.item())
        val_accuracies.append(100 * correct / total)

# Initialize a list to store test accuracies
test_accuracies = []

# Iterate over the test dataset
with torch.no_grad():
    correct = 0
    total = 0
    for i, data in enumerate(test_dataloader, 0):
```

```
        inputs = data["ct"]["data"].float()
        labels = data["label"]

        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    test_accuracies.append(100 * correct / total)

# Print the accuracy of the network on the test images
print('Accuracy of the network on the test images: %d %%' % (100 * correct / total))
```

Βιβλιογραφία

[1] Jin, C., Chen, W., Cao, Y. et al. Development and evaluation of an artificial intelligence system for COVID-19 diagnosis. Nat Commun 11, 5088 (2020). https://doi.org/10.1038/s41467-020-18685-1. Pages: 2

[2] S.P. Morozov, A.E. Andreychenko, N.A. Pavlov, A.V. Vladzymyrskyy, N.V.Ledikhova, V.A. Gombolevskiy, I.A. Blokhin, P.B. Gelezhe, A.V. Gonchar, V.Yu. Chernina MOSMEDDATA: CHEST CT-SCANS WITH COVID-19 RELATED FINDINGS DATASET Research and Practical Clinical Center of Diagnostics and Telemedicine Technologies, Department of Health Care of Moscow, Russian Federation, https://doi.org/10.48550/arXiv.2005.06465

[3] https://nipy.org/nibabel/coordinate_systems.html (Accessed 06/06/2023)

[4] https://www.ipfradiologyrounds.com/hrct-primer/image-reconstruction/ (Accessed 20/06/2023)

[5] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539. Pages: 438-440

[6] Deep Learning Limitations and Flaws. Bahman Zohuri1, Masoud Moghaddam. DOI: 10.32474/MAMS.2020.02.000138. Pages: 248-249.

[7] https://torchio.readthedocs.io/ (Accessed 12/06/2023)

[8] Pérez-García F, Sparks R, Ourselin S. TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. Comput Methods Programs Biomed. 2021 Sep;208:106236. doi: 10.1016/j.cmpb.2021.106236. Epub 2021 Jun 17. PMID: 34311413; PMCID: PMC8542803.

[9] T. N. Sainath, O. Vinyals, A. Senior and H. Sak, "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 2015, pp. 4580-4584, doi: 10.1109/ICASSP.2015.7178838.

[10] Goncharov M, Pisov M, Shevtsov A, Shirokikh B, Kurmukov A, Blokhin I, Chernina V, Solovev A, Gombolevskiy V, Morozov S, Belyaev M. CT-Based COVID-19 triage: Deep multitask learning improves joint identification and severity quantification. Med Image Anal. 2021 Jul;71:102054. doi: 10.1016/j.media.2021.102054. Epub 2021 Apr 1. PMID: 33932751; PMCID: PMC8015379.

[11] McKinney, S.M., Sieniek, M., Godbole, V. et al. International evaluation of an AI system for breast cancer screening. Nature 577, 89–94 (2020). https://doi.org/10.1038/s41586-019-1799-6

[12] Neha Baranwal, Preethi Doravari, Renu Kachhoria. Classification of Histopathology Images of Lung Cancer Using Convolutional Neural Network (CNN). https://doi.org/10.48550/arXiv.2112.13553. Pages:3-4.

[13] P. Ongsulee, "Artificial intelligence, machine learning and deep learning," 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, Thailand, 2017, pp. 1-6, doi: 10.1109/ICTKE.2017.8259629. Pages: 3.

[14] Li Deng; Dong Yu, Deep Learning: Methods and Applications , now, 2014. Pages: 5.

[15] Jarrett, Chris. (2007). Computed Tomography: Fundamentals, System Technology, Image Quality, Applications [Book Review]. Engineering in Medicine and Biology Magazine, IEEE. 26. 12-12. 10.1109/MEMB.2007.335578.

[16] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, "Deep Learning for Computer Vision: A Brief Review", Computational Intelligence and Neuroscience, vol. 2018, Article ID 7068349, 13 pages, 2018. https://doi.org/10.1155/2018/7068349

[17] Miotto, Riccardo & Wang, Fei & Wang, Shuang & Jiang, Xiaoqian. (2017). Deep learning for healthcare: review, opportunities and challenges. Briefings in bioinformatics. 19. 10.1093/bib/bbx044.

[18] Benuwa, Ben & Zhan, Yong & Ghansah, Benjamin & wornyo, Dickson & Banaseka, Frank. (2016). A Review of Deep Machine Learning. International Journal of Engineering Research in Africa. 24. 124-136. 10.4028/www.scientific.net/JERA.24.124. Pages: 18.

[19] Soares, Eduardo & Angelov, Plamen & Biaso, Sarah & Froes, Michele & Abe, Daniel. (2020). SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification. 10.1101/2020.04.24.20078584.

[20] Jin C, Chen W, Cao Y, Xu Z, Tan Z, Zhang X, Deng L, Zheng C, Zhou J, Shi H, Feng J. Development and evaluation of an artificial intelligence system for COVID-19 diagnosis. Nat Commun. 2020 Oct 9;11(1):5088. doi: 10.1038/s41467-020-18685-1. PMID: 33037212; PMCID: PMC7547659.

[21] Padmanaban, Sriramakrishnan & Thiruvenkadam, Kalaiselvi & T., Padmapriya & Sivasakthivel, Ramkumar & Nagarajan, Kalaichelvi. (2020). Medical Image File Formats and Digital Image Conversion. International Journal of Engineering and Advanced Technology. 9. 74-78. 10.35940/ijeat.A1093.1291S419.

[22] https://pytorch.org/ (Accessed 02/0/2023)

[23] Iftikhar S, Zhang Z, Asim M, Muthanna A, Koucheryavy A, Abd El-Latif AA. Deep Learning-Based Pedestrian Detection in Autonomous Vehicles: Substantial Issues and Challenges. Electronics. 2022; 11(21):3551. https://doi.org/10.3390/electronics11213551

[24] Massimi F, Ferrara P, Benedetto F. Deep Learning Methods for Space Situational Awareness in Mega-Constellations Satellite-Based Internet of Things Networks. Sensors (Basel). 2022 Dec 23;23(1):124. doi: 10.3390/s23010124. PMID: 36616722; PMCID: PMC9824630.

[25] Chen, Claire & Mahjoubfar, Ata & Tai, Li-Chia & Blaby, Ian & Huang, Allen & Niazi, Kayvan & Jalali, Bahram. (2016). Deep Learning in Label-free Cell Classification. Scientific Reports. 6. 21471. 10.1038/srep21471.

[26] Fang, Weili & Ding, Lieyun & Zhong, Botao & Luo, Hanbin. (2018). Automated Detection of Workers and Heavy Equipment on Construction Sites: A Convolutional Neural Network Approach. Advanced Engineering Informatics. 37. 10.1016/j.aei.2018.05.003.

[27] Ambuj Mehrish, Navonil Majumder, Rishabh Bhardwaj, Rada Mihalcea, Soujanya Poria. A Review of Deep Learning Techniques for Speech Processing. https://doi.org/10.48550/arXiv.2305.00359

[28] https://www.nvidia.com/en-us/glossary/data-science/pytorch/ (Accessed 12/06/2023)

[29] Neural Networks and Deep Learning Charu C. Aggarwal. ISBN: 978-3-319-94462-3

[30] Ofeidis, Iason & Kiedanski, Diego & Tassiulas, Leandros. (2022). An Overview of the Data-Loader Landscape: Comparative Performance Analysis. 10.48550/arXiv.2209.13705.

[31] Li, Xiangrui & Morgan, Paul & Ashburner, John & Smith, Jolinda & Rorden, Chris. (2016). The first step for neuroimaging data analysis: DICOM to NIfTI conversion. Journal of Neuroscience Methods. 264. 10.1016/j.jneumeth.2016.03.001.

[32] Shorten, Connor & Khoshgoftaar, Taghi. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data. 6. 10.1186/s40537-019-0197-0.

[33] https://torchio.readthedocs.io/transforms/augmentation.htmlrandomaffine (Accesseed 15 July 2023)

[34] Karl, R., Weiss., Taghi, M., Khoshgoftaar., Dingding, Wang. (2016). A survey of transfer learning. Journal of Big Data, 3(1):9-. doi: 10.1186/S40537-016-0043-6. Pages: 14.

[35] L. Shao, F. Zhu and X. Li, "Transfer Learning for Visual Categorization: A Survey," in IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 5, pp. 1019-1034, May 2015, doi: 10.1109/TNNLS.2014.2330900. Pages:1027.

[36] Deng, Jia & Dong, Wei & Socher, Richard & Li, Li-Jia & Li, Kai & Li, Fei-Fei. (2009). ImageNet: a Large-Scale Hierarchical Image Database. IEEE Conference on Computer Vision and Pattern Recognition. 248-255. 10.1109/CVPR.2009.5206848. Pages: 1-2.

[37] Hendrycks, Dan & Lee, Kimin & Mazeika, Mantas. (2019). Using Pre-Training Can Improve Model Robustness and Uncertainty.

[38] Larobina, M., Murino, L. Medical Image File Formats. J Digit Imaging 27, 200–206 (2014). https://doi.org/10.1007/s10278-013-9657-9. Pages: 4.

[39] Berrar, Daniel. (2018). Cross-Validation. 10.1016/B978-0-12-809633-8.20349-X. Pages: 3.

[40] Burkov, A. (2019) The Hundred-Page Machine Learning Book. 159.

[41] Cawley, Gavin & Talbot, Nicola. (2010). On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. Journal of Machine Learning Research. 11. 2079-2107.

[42] Yoo S, Gujrathi I, Haider MA, Khalvati F. Prostate Cancer Detection using Deep Convolutional Neural Networks. Sci Rep. 2019 Dec 20;9(1):19518. doi: 10.1038/s41598-019-55972-4. PMID: 31863034; PMCID: PMC6925141.

[43] Gökçe, Çağrı & Özdemir, Oğulcan & Kındıroğlu, Ahmet & Akarun, Lale. (2020). Score-Level Multi Cue Fusion for Sign Language Recognition. 10.1007/978-3-030-66096-3_21.