



ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΑΝΘΡΩΠΙΣΤΙΚΕΣ ΚΑΙ ΚΟΙΝΩΝΙΚΕΣ ΕΠΙΣΤΗΜΕΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη Εφαρμογής στο Raspberry Pi 3 B+
για τον Χειρισμό Κάμερας με χρήση Αλγορίθμου Μηχανικής Όρασης**

Φοιτητής: Γιούρι Κιλινκαρίδης

Επιβλέπων Καθηγητής: Αυλωνίτης Μάρκος

ΘΕΣΣΑΛΟΝΙΚΗ, ΜΑΡΤΙΟΣ 2021

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη Εφαρμογής στο Raspberry Pi 3 B+
για τον Χειρισμό Κάμερας με χρήση Αλγορίθμου Μηχανικής Όρασης**

**Γιούρι Κιλινκαρίδης
Α.Μ. Π2016130**

Μέλη Συμβουλευτικής Επιτροπής

Επιβλέπων Καθηγητής: Αυλωνίτης Μάρκος

Μέλος Επιτροπής: Κάτια-Λήδα Κερμανίδου

Μέλος Επιτροπής: Μιχαήλ Στεφανιδάκης

Ανάπτυξη Εφαρμογής στο Raspberry Pi 3 B+ για τον Χειρισμό Κάμερας με χρήση Αλγορίθμου Μηχανικής Όρασης

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ

Υπεύθυνη Δήλωση: Βεβαιώνω υπεύθυνα ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης αναφέρω όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής του Ιονίου Πανεπιστημίου Κέρκυρας.

Όνομα:

Γιούρι Κιλινκαρίδης

Υπογεγραμμένος:

Γιούρι Κιλινκαρίδης

Ημερομηνία:

Πέμπτη, 11 Μαρτίου 2021

Ανάπτυξη Εφαρμογής στο Raspberry Pi 3 B+ για τον Χειρισμό Κάμερας με χρήση Αλγορίθμου Μηχανικής Όρασης

ΠΕΡΙΛΗΨΗ

Το αντικείμενο της πτυχιακής εργασίας ήταν η ανάπτυξη εφαρμογής χειρισμού κάμερας με την χρήση αλγορίθμου μηχανικής όρασης. Συγκεκριμένα, η εφαρμογή χρησιμοποιεί έναν αλγόριθμο υπεύθυνο για τον εντοπισμό σε πραγματικό χρόνο της θέσης και της αντίστοιχης κατηγορίας στην οποία ανήκει το αντικείμενο. Σε αυτήν την περίπτωση το αντικείμενο που αναγνωρίζεται απο τον αλγόριθμο τεχνητής νοημοσύνης είναι το χέρι του χρήστη πιο συγκεκριμένα τρεις κατηγορίες χειρονομιών. Η πρώτη κατηγορία είναι η γροθιά, η δεύτερη κατηγορία είναι το σήμα της νίκης και τέλος η ανοιχτή παλάμη με κλειστά δάχτυλα. Με βάση την κατηγορία της εντοπισμένης χειρονομίας απο τον αλγόριθμο, πραγματοποιούνται οι λειτουργίες όπως η λήψη και αποθήκευση φωτογραφίας, η προβολή των ληφθέντων φωτογραφιών και η περιήγηση των φωτογραφιών όπως η μετάβαση στην επόμενη ή την προηγούμενη εικόνα. Για την επίτευξη των λειτουργιών αυτών δημιουργήσα ένα σετ δεδομένων με 180 εικόνες, στο σύνολο 3 κατηγορίες απο μία για κάθε χειρονομία. Κάθε κατηγορία περιλαμβάνει απο 60 εικόνες οι οποίες επεξεργάστηκαν με το εργαλείο LabelImg για την δημιουργία του αντίστοιχου XML αρχείου. Το XML αρχείο περιλαμβάνει πληροφορίες σχετικά με την εικόνα όπως οι συντεταγμένες και την κατηγορία της χειρονομίας. Στη συνέχεια με την βοήθεια το TensorFlow Object Detection API ήταν δυνατόν να πραγματοποιηθεί η μετεκπαίδευση του αλγορίθμου τεχνητής νοημοσύνης για τον εντοπισμό των 3 κατηγοριών χειρονομιών.

Τέλος η ανάπτυξη της εφαρμογής έγινε με την χρήση της προγραμματιστικής γλώσσας Java και του Android Studio, επίσημο Περιβάλλον Ολοκληρωμένης Ανάπτυξης (IDE) για Android εφαρμογές.

Λέξεις Κλειδιά: μηχανικής όρασης, αλγόριθμος τεχνητής νοημοσύνης, γροθιά, παλάμη, σήμα της νίκης, συντεταγμένες, κατηγορίες, μετεκπαίδευση.

Camera Application Developed for Raspberry Pi 3 B+ using Computer Vision Algorithm

Abstract

The main goal was to develop a camera application controlled using gestures with the help of a computer vision algorithm. To locate in real time and classify each of three gestures SSD-MobilenetV2 was used. It is a fast and fairly accurate object detector made specifically for devices with low computational power. The three categories of gestures are a) fist, b) palm (fingers closed) and c) the sign of victory/peace. Based on the detected gesture specific commands execute such as capturing and saving photos, viewing captured photos and browsing among saved pictures. To achieve those functionalities a custom data set was created, 180 images in total for three gesture categories. Each category includes 60 images, for each image was created a XML file that contains information about the image. Information such as coordinates of the gesture and category label. Tensorflow Object Detection API was used for transfer-learning locally on a laptop. The trained model was exported as a tflite file to be used in an Android environment. Finally the application was developed using Java as the programming language and Android Studio, the official Integrated Development Environment (IDE) for Android applications.

Keywords: computer vision, neural network, fist, palm, victory sign, coordinates, categories, transfer-learning.

Εισαγωγή

Οι επιτυχίες στον τομέα της μηχανικής όρασης των τελευταίων ετών προσφέρουν νέες δυνατότητες στον τρόπο με τον οποίο αλληλεπιδρούν οι άνθρωποι με την τεχνολογία, δυνατότητες τις οποίες αποφάσισα να εξερευνήσω στα πλαίσια της πτυχιακής εργασίας. Έτσι το θέμα μελέτης της παρούσας εργασίας είναι η αποδοτική χρήση αλγορίθμου μηχανικής όρασης στην συσκευή Raspberry Pi 3 B+ για την ανάπτυξη τρόπου χειρισμού της κάμερας ανέπαφα.

Για να πετύχω τον στόχο αρχικά χρειάστηκε να μελετήσω σε βάθος τι ακριβώς είναι και πως λειτουργούν οι αλγόριθμοι μηχανικής όρασης. Δηλαδή τον τρόπο με τον οποίο έγινε η προσομοίωση του μηχανισμού της όρασης και της αναγνώρισης των αντικειμένων στον υπολογιστή. Μηχανισμοί οι οποίοι από τον άνθρωπο εκτελούνται απλά και άμεσα, θα μπορούσε κάποιος να ισχυριστεί στον βαθμό αυτοματισμού χωρίς να επιβάλλετε κάποια ιδιαίτερη προσπάθεια και ουτε να γίνεται αντιληπτή η όλη διαδικασία. Για τις μηχανές όμως τα πράγματα είναι διαφορετικά, απαιτείται μεγάλη προσοχή στον τρόπο προσεγγίσεις του προβλήματος για την ανάπτυξη ενός τέτοιου αλγορίθμου, καθώς ο ερευνητής είναι αυτός που πρέπει πρώτα να κατανοήσει όσο τον δυνατόν καλύτερα την διαδικασία της όρασης και της αναγνώρισης και στη συνέχεια να μεταφέρει τις τεχνικές αυτές στον υπολογιστή ώστε να μπορέσει με την σειρά της η μηχανή να αποκτήσει την ικανότητα της “όρασης” και της “κατανόησης” του περιεχομένου των ψηφιακών δεδομένων, όπως φωτογραφίες και βίντεο.

Το επόμενο πρόβλημα που έπρεπε να αντιμετωπίσω ήταν η αποτελεσματική εφαρμογή ενός τέτοιου αλγορίθμου στην συσκευή Raspberry Pi 3 B+. Το Raspberry Pi 3 B+ είναι ένας μικροϋπολογιστής μονής πλακέτας στο μέγεθος της παλάμης, παρέχει όλες τις λειτουργίες ενός κανονικού υπολογιστή, συνδέεται με οθόνη και περιφερειακές συσκευές όπως το πληκτρολόγιο και το ποντίκι. Εξαιτίας όμως του μικρού μεγέθους η υπολογιστική δύναμη της συσκευής είναι περιορισμένη με αποτέλεσμα για να λειτουργήσει μια εφαρμογή η οποία χρησιμοποιεί αλγόριθμο μηχανικής όρασης έπρεπε να γίνουν κάποιες αλλαγές σχετικά με το λειτουργικό σύστημα το οποίο είναι εγκατεστημένο στο Raspberry Pi 3 B+ αλλά και την μορφή του ίδιου του αλγορίθμου.

Επιπλέον επειδή το λειτουργικό σύστημα που χρησιμοποίησα ήταν το Android 9.0 η δημιουργία της εφαρμογής έγινε με την προγραμματιστική γλώσσα Java και του περιβάλλοντος ανάπτυξης εφαρμογών android, Android Studio. Ότι αφορά την εκπαίδευση του αλγορίθμου για την αναγνώριση των τριών κατηγοριών χειρονομιών το εργαλείο που αξιοποίησα ήταν το Tensorflow Object Detection API της Google και ένα σετ δεδομένων από εικόνες που δημιούργησα.

Τέλος η εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας εκτελεί ικανοποιητικά έως ένα βαθμό τις λειτουργίες που τέθηκαν ως στόχοι. Το τμήμα της εφαρμογής που χρειάζεται περαιτέρω μελέτη και βελτίωση είναι ο ίδιος ο αλγόριθμος μηχανικής όρασης, πιο συγκεκριμένα ο τρόπος με τον οποίο πραγματοποιείται η τεχνική της εκμάθησης για τον αποτελεσματικό εντοπισμό των επιθυμητών αντικειμένων και αποτελεί έναν από τους μελλοντικούς στόχους προς μελέτη και έρευνα.

ΠΕΡΙΕΧΟΜΕΝΑ

<i>Περίληψη</i>	i
<i>Abstract</i>	ii
<i>Εισαγωγή</i>	iii

Κεφάλαιο 1

Μηχανική Όραση

1.1 Μηχανική Όραση.....	10
1.2 Η Αρχή.....	11
1.3 Αλγόριθμοι Μηχανικής Όρασης.....	12
1.4 ANN (Artificial Neural Network).....	18
1.4.1 Μεθοδολογίες Εκπαίδευσης ANN.....	20
1.4.2 Κατηγορίες ANN.....	23
1.4.3 Συνάρτηση Ενεργοποίησης.....	24
1.4.4 Συνάρτηση Απώλειας.....	25
1.4.5 Εφαρμογές Τεχνητών Νευρωνικών Δικτύων.....	26
1.5 Convolutional Neural Network.....	27
1.6 Mobilenet V2.....	32

Κεφάλαιο 2

Τα Εργαλεία

2.1 Raspberry Pi 3 B+.....	34
2.2 Tensorflow Object Detection API.....	35
2.3 Android Studio και Java.....	36

Κεφάλαιο 3

Η Εφαρμογή

3.1 Επιλογή Λειτουργικού Συστήματος.....	37
3.2 Πρόσβαση στην USB Κάμερα.....	38

3.3 Επεξεργασία Δεδομένων Κάμερας.....	39
3.4 Εκπαίδευση Αλγορίθμου.....	40
3.5 Διαδικασία Αναγνώρισης.....	43
3.6 Πρόσβαση στα Αποτελέσματα.....	44
3.7 Λειτουργίες Εφαρμογής.....	45

Κεφάλαιο 4

Κώδικας Εφαρμογής

4.1 AndroidManifest.XML.....	48
4.2 MainActivity.Java.....	50
4.3 Επιλογή και Ενεργοποίηση Κάμερας.....	58
4.4 Συλλογή και Μετάδοση των Καρέ Κάμερας.....	61
4.5 Δημιουργία Φακέλου Αποθήκευσης.....	63
4.6 Αποθήκευση Φωτογραφίας.....	64
4.7 Διεργασίες του Background Thread.....	65
4.7.1 Λήψη Φωτογραφίας.....	65
4.7.2 Προβολή Φωτογραφιών.....	66
4.7.3 Περιήγηση Φωτογραφιών.....	68
4.8 Αναγνώριση Χειρονομίας.....	69
4.9 Ανάγνωση Αλγορίθμου και Ετικετών.....	72

<i>Παρατηρήσεις και Αποτελέσματα.....</i>	<i>73</i>
--	------------------

<i>Βιβλιογραφία.....</i>	<i>74</i>
---------------------------------	------------------

Κατάλογος με τις Εικόνες

Εικόνα 1	13
Εικόνα 2	14
Εικόνα 3	15
Εικόνα 4	16
Εικόνα 5	17
Εικόνα 6	18
Εικόνα 7	18
Εικόνα 8	23
Εικόνα 9	24
Εικόνα 10	27
Εικόνα 11	28
Εικόνα 12	29
Εικόνα 13	30
Εικόνα 14	30
Εικόνα 15	31
Εικόνα 16	31
Εικόνα 17	32
Εικόνα 18	35
Εικόνα 19	41
Εικόνα 20	41
Εικόνα 21	42
Εικόνα 22	45
Εικόνα 23	46
Εικόνα 24	47
Εικόνα 25	50
Εικόνα 26	51
Εικόνα 27	52
Εικόνα 28	53
Εικόνα 29	54
Εικόνα 30	55

Μηχανική Όραση

Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει η εισαγωγή στην έννοια της Μηχανικής Όρασης. Πως ξεκίνησε η ανάπτυξη του πεδίου, οι τεχνικές που χρησιμοποιούνται για την δημιουργία των αλγορίθμων και η εξέλιξη τους με το πέρασμα των ετών. Τέλος θα παρουσιαστεί ο αλγόριθμος μηχανικής όρασης που επιλέχθηκε για την διαδικασία της αναγνώρισης και του εντοπισμού των χειρονομιών.

1.1 Μηχανική Όραση

Η μηχανική όραση είναι ένα απο τα σημαντικότερα πεδία της επιστήμης των υπολογιστών, σαν έννοια εμφανιστηκε πρώτη φορά την δεκαετία του 1950 και αποτέλεσε μια απο τις δυσκολότερες προκλήσεις που είχαν να αντιμετωπίσουν οι ερευνητές της εποχής. Κάποιοι ερευνητές μάλιστα υποτίμισαν την πολυπλοκότητα του θέματος και θεώρησαν πως το πρόβλημα της “όρασης” στους υπολογιστές μπορεί να λυθεί στο διάστημα μερικων μηνων, κάτι που αποδείχτηκε αδύνατων εξαιτίας έλλειψης αρχικά τεχνολογίας και δευτερον κατανόησης του τρόπου λειτουργίας και της δομής του συστήματος όρασης. Ακόμα και σήμερα μετά απο 70 χρόνια έρευνας δεν έχει δοθεί ικανοποιητική λύση στο πρόβλημα αλλά παρόλα αυτά έχουν γίνει σημαντικά βήματα προς την κατεύθυνση της τελικής λύσης. Οι μελέτες του πεδίου επικεντρώνεται στην διαδικασία της προσομοίωσης του ανθρώπινου μηχανισμού όρασης, παρατήρησης και κατανόησης στους υπολογιστές και τις μηχανές για την καλύτερη αυτοματοποίηση της λειτουργίας τους σε πεδία όπως η ιατρική και ρομποτική.

1.2 Η Αρχή

Το 1963 ο Larry Roberts στο διδακτορικό[1] του πρότεινε στο MIT την ιδέα της εξαγωγής 3D γεωμετρικής πληροφορίας από 2D προβολές των μπλοκ, δηλαδή το πρόγραμμα που ανέπτυξε είχε ως στόχο την επεξεργασία των δισδιάστατων φωτογραφιών σε γραμμικά σχέδια και στην συνέχεια την δημιουργία και την προβολή των αντικειμένων της φωτογραφίας σε τρισδιάστατη μορφή. Η ιδέα του αυτή τον καθιέρωσε και ως τον πατέρα του πεδίου της μηχανικής όρασης. Το 1966 ο Marvin Minsky άλλος ένας ερευνητής του MIT θεώρησε ότι το πρόβλημα της μηχανικής όρασης δεν είναι τόσο περίπλοκο όσο φαίνεται και μπορεί να λυθεί σε μικρό χρονικό διάστημα για αυτό και ανέθεσε μια εργασία για την καλοκαιρινή περίοδο σε έναν από τους φοιτητές του τον Gerald Jay Sussman με θέμα την σύνδεση κάμερας στον υπολογιστή και την ικανότητα από την μηχανή να περιγράψει αυτά που έβλεπε. Βέβαια όπως αποδείχθηκε το πρόβλημα ήταν πολύ πιο δύσκολο για να λυθεί σε ένα καλοκαίρι και αποτελεί εδώ και 70 χρόνια πεδίο συνεχής έρευνας.

Την ίδια περίοδο δημιουργήθηκε ένα ακόμα νέο πεδίο στον τομέα της επιστήμης των υπολογιστών και αυτό είναι η τεχνητή νοημοσύνη. Οι ερευνητές της εποχής πίστευαν πως η τεχνητή νοημοσύνη ήταν το κλειδί στο πρόβλημα της μηχανικής όρασης αλλά εξαιτίας της έλλειψης των κατάλληλων αλγορίθμων και του υπολογιστικού υλικού οι προσπάθειες ήταν ανεπιτυχής. Το 1962 ο Hubel και Wiesel δημοσίευσαν[2] μια πρωτοπόρα έρευνα σχετικά με την ιεραρχική δομή της οργάνωσης της όρασης στις γάτες, πάνω στη μελέτη αυτή βασίστηκε ο Kunihiko Fukushima το 1980 και παρουσίασε το Neocognitron[3] ένα ιεραρχικό πολυεπίπεδο τεχνητό νευρωνικό δίκτυο. Το δίκτυο εκπαιδεύεται χωρίς την ανάγκη εξωτερικής επέμβασης και αποκτά την ικανότητα να αναγνωρίζει μοτίβα με βάση την γεωμετρική ομοιότητα των σχημάτων τους χωρίς επηρεάζεται από την θέση τους. Αποτελείται από ένα στρώμα εισόδου το οποίο ακολουθείτε από ομοιόμορφα συνδεδεμένες δομές, η κάθε δομή περιλαμβάνει δύο στρώματα κόμβων, με το πρώτο να είναι τα “S-cells”, που αντιστοιχούν στα χαμηλότερου επιπέδου οπτικά νευρικά κύτταρα και το δεύτερο στρώμα τα “C-cells”, αντιστοιχούν στα υψηλότερου επιπέδου οπτικά νευρικά κύτταρα με βάση την έρευνα των Hubel και Wiesel. Στη συνέχεια το 1998 ο Yann LeCun μαζί με την ομάδα του σχεδίασε το πρώτο CNN (Convolutional Neural Network) το οποίο ονομάστηκε LeNet-5[4]. Το δίκτυο μπορούσε να αναγνωρίσει χειρόγραφους αριθμούς και χρησιμοποιήθηκε από κάποιες τράπεζες για την κατηγοριοποίηση των χειρόγραφων αριθμών στις αποδείξεις και τις επιταγές. Συνεπώς καταλήγουμε στο συμπέρασμα ότι η δομή του Convolutional Neural Network εμπνευσμένη από την έρευνα των Hubel και Wiesel, είναι αυτή που αποτελεί μέχρι και σήμερα τον κύριο τρόπο αναγνώρισης αντικειμένων σε εικόνες και βίντεο. Στην επόμενη ενότητα θα αναλύσουμε σε βάθος την αρχιτεκτονική και τον τρόπο λειτουργίας ενός CNN δικτύου.

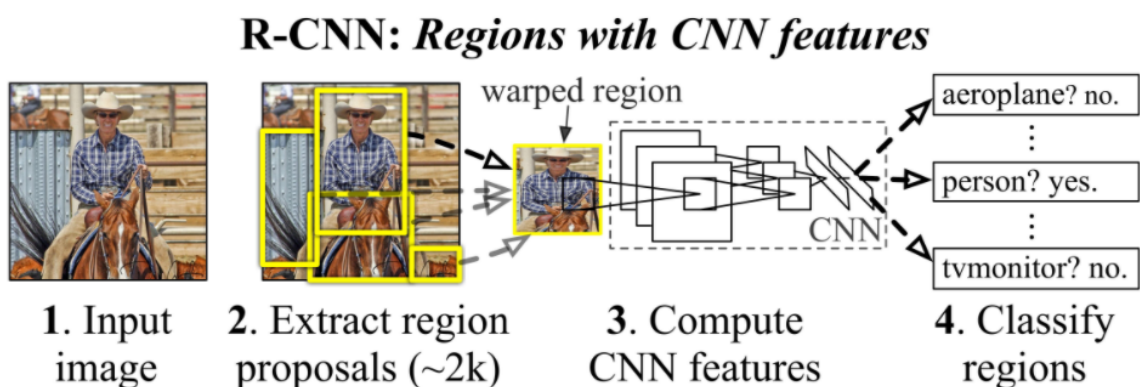
1.3 Αλγόριθμοι Μηχανικής Όρασης

AlexNet

Βασιζόμενοι στην αρχιτεκτονική του LeNet-5 σχεδιάστηκε το AlexNet[5] το 2012 από τους Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton. Η αρχιτεκτονική του AlexNet είναι αυτή που οδήγησε την ομάδα στην νίκη στο διαγωνισμό του ImageNet Large Scale Visual Recognition Challenge με ποσοστό σφάλματος κατηγοριοποίησης εικόνας στα 15.3% με τον δεύτερο καλύτερο αλγόριθμο να έχει ποσοστό σφάλματος στα 26%. Η επιτυχία κυρίως οφείλεται στην πολυεπίπεδη αρχιτεκτονική του μοντέλου και στον μεγάλο αριθμό υπολογισμών που πραγματοποιούνται κατά την εκπαίδευση και τον εντοπισμό από την μονάδα επεξεργασίας γραφικών. Για την εκπαίδευση, το AlexNet δέχεται ως είσοδο μια εικόνα διαστάσεων $227 \times 227 \times 3$ όπου το 3 συμβολίζει τα τρία κανάλια χρωμάτων της εικόνας (Red, Green, Blue). Η εικόνα στη συνέχεια περνάει από το πρώτο Convolutional layer το οποίο περιέχει 96 φίλτρα διαστάσεων $11 \times 11 \times 3$ με stride 4. Κατά την διάρκεια της διαδικασίας Convolution το κάθε φίλτρο διαστάσεων $11 \times 11 \times 3$ θα περάσει από ολη την επιφάνεια της εικόνας και οι τιμές του πίνακα και της εικόνας πολλαπλασιάζονται μεταξύ τους και προστίθενται δημιουργώντας έναν νέο πίνακα. Το $\text{stride} = 4$ συμβολίζει ότι το φίλτρο κατά την μετακίνηση του στην επιφάνεια της εικόνας θα αφήνει κενό τεσσάρων εικονοστοιχείων από την αρχική θέση στην οποία εφαρμόστηκε για πρώτη φορά. Μετά το πρώτο Convolutional layer δημιουργείται ο χάρτης χαρακτηριστικών διαστάσεων $55 \times 55 \times 96$. Πριν περάσει ο χάρτης χαρακτηριστικών στο επομενο ConvLayer2 ακολουθεί το Pooling layer διαστάσεων 3×3 και stride 2 στόχος του οποίου είναι να μικρύνει τις διαστάσεις του χάρτη χαρακτηριστικών από $55 \times 55 \times 96$ σε $27 \times 27 \times 96$. Στο ConvLayer2 τα φίλτρα πλέον είναι διαστάσεων 5×5 και στο σύνολο είναι 256 με padding = 2. Το padding = 2 είναι υπεύθυνο για τον αριθμό των εικονοστοιχείων που θα προστεθούν στις άκρες της εικόνας ώστε να διατηρηθούν οι διαστάσεις του χάρτη χαρακτηριστικών. Μετά το ConvLayer2 ο χάρτης είναι διαστάσεων $27 \times 27 \times 256$. Στο επόμενο στάδιο το Pooling είναι διαστάσεων 3×3 με stride 2 δηλαδή ένα Overlapping Pooling αυτό έχει ως αποτέλεσμα την μείωση των διαστάσεων του χάρτη χαρακτηριστικών σε $13 \times 13 \times 256$. Κατά το Pooling έχουμε απώλεια πληροφορίας των χαρακτηριστικών αλλά παρόλα αυτά το δίκτυο επωφελείται από την διαδικασία επειδή επιτρέπεται καλύτερη γενίκευση κατά την εξαγωγή χαρακτηριστικών. Το ConvLayer3 και ConvLayer4 διαθέτουν 384 φίλτρα διαστάσεων 3×3 με padding 1 και η έξοδος πλέον είναι της μορφής $13 \times 13 \times 384$. Το τελευταίο Convolution layer έχει 256 φίλτρα διαστάσεων 3×3 με padding 1, η έξοδος από το πέμπτο και τελευταίο Convolution layer έχει διαστάσεις $13 \times 13 \times 256$. Πλέον ακολουθεί το τελευταίο Max Overlapping Pooling το οποίο μετατρέπει τις διαστάσεις σε $6 \times 6 \times 256$ του χάρτη χαρακτηριστικών και έχουμε την αναδίπλωση του χάρτη σε μονοδιάστατο διάνυσμα για την είσοδο του στο πρώτο από τα τρία Fully Connected layer. Το πρώτο και το δεύτερο στρώμα έχουν το καθένα από 4096 νευρώνες και το τελευταίο το Output layer έχει 1000 νευρώνες που θα καθορίσουν και την κατηγορία στην οποία ανήκει η εικόνα.

R-CNN

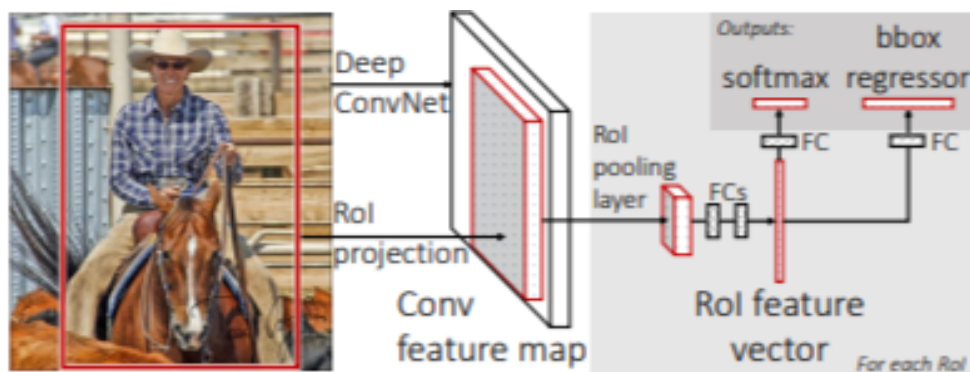
Το R-CNN [6] είναι το πρώτο μοντέλο της οικογένειας των R-CNN αλγορίθμων. Αποτελείται από δύο κύρια μέρη το πρώτο είναι α) η επιλογή περιοχών στην εικόνα που πιθανώς περιέχουν αντικείμενα και β) το CNN δίκτυο το οποίο είναι υπεύθυνο για την εξαγωγή χαρακτηριστικών από τις εικόνες για τον εντοπισμό των αντικειμένων (Εικ. 1). Η επιλογή περιοχών γίνεται με ξεχωριστό αλγόριθμο ο οποίος υπολογίζει την πιθανότητα να υπάρχει στην περιοχή ένα ή και περισσότερα αντικείμενα με βάση την σχέση που έχουν μεταξύ τους τα pixel, δηλαδή κατά πόσο ανήκουν στην ίδια ομάδα. Η σχέση μεταξύ των εικονοστοιχείων καθορίζεται με βάση χαρακτηριστικά όπως το χρώμα, την θέση και την απόστασή τους. Το αποτέλεσμα είναι ότι ο αλγόριθμος προτείνει στο τέλος 2000 περιοχές οι οποίες πρέπει να επεξεργαστούν και να ταξινομηθούν με βάση την κατηγορία στην οποία ανήκουν. Ότι αφορά το Convolutional Neural Network για την εξαγωγή των χαρακτηριστικών, αυτό είναι ίδιο με το AlexNet. Στο σύνολο υπάρχουν 5 Convolutional layers και 2 Fully Connected στρώματα. Για την κατηγοριοποίηση χρησιμοποιούνται SVMs (Support Vector Machines) για την κάθε κατηγορία. Το πρόβλημα που προκύπτει είναι ότι η για την εκπαίδευση των SVMs πρώτα πρέπει να εκπαιδευτεί το CNN τμήμα του μοντέλου, δηλαδή η διαδικασία χωρίζεται σε δύο τμήματα δίχως την δυνατότητα ταυτόχρονης εκπαίδευσης. Μια ακόμα μεθοδολογία που χρησιμοποιείται στο R-CNN είναι το Bounding Box Regression, όπου κάθε προτεινόμενο πλαίσιο για το αντικείμενο πρέπει να αντιστοιχεί σε ένα σίγουρο πλαίσιο με αποτέλεσμα με τα την σύγκριση των θέσεων τους να βελτιώνεται η τοποθέτηση του πλαισίου στην επόμενη επανάληψη. Το μειονέκτημα του R-CNN είναι ότι εξαιτίας της πολυπλοκότητας του μοντέλου δεν μπορεί να χρησιμοποιηθεί σε πραγματικό χρόνο για τον εντοπισμό αντικειμένων καθώς η διαδικασία εντοπισμού διαρκεί μερικά δευτερόλεπτα ακόμα και σε ένα σύστημα με ισχυρή μονάδα επεξεργασίας γραφικών.



Εικ. 1) R-CNN. Διαδικασία κατηγοριοποίησης.

Fast R-CNN

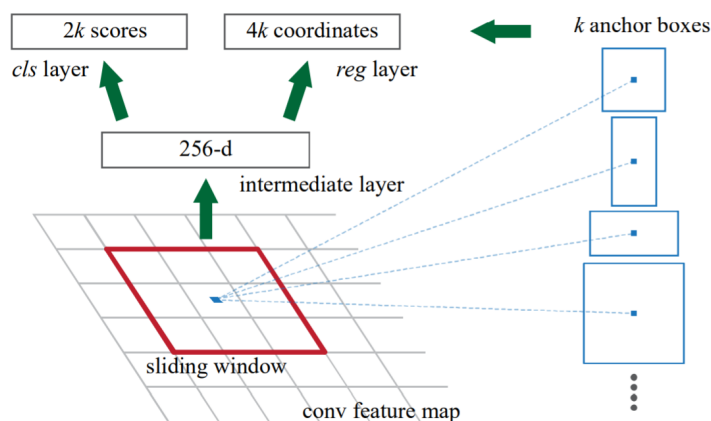
Βασισμένο στο R-CNN δημιουργήθηκε το Fast R-CNN [7] το οποίο φέρει μαζί του ένα σύνολο απο βελτιώσεις σε σχέση με το προηγούμενο μοντέλο. Αρχικά αυξήθηκε η ταχύτητα εκπαίδευσης και εντοπισμού και ταυτόχρονα αυξήθηκε το ποσοστό ευστοχίας του μοντέλου. Μια απο τις προσθήκες στο μοντέλο ήταν η χρήση RoI Pooling layer (Regions of Interest). Το RoI Pooling layer χρησιμοποιεί max pooling για την μετατροπή των χαρακτηριστικών κάθε προτεινόμενης περιοχής σε ένα μικρο χαρτη χαρακτηριστικών συγκεκριμένων διαστάσεων $H \times W$, έστω 7×7 . Κάθε RoI αποτελείται απο μια τετράδα συντεταγμένων (t, c, h, w) , οι πρώτες δύο συντεταγμένες είναι η πάνω αριστερή γωνία του πλαισίου και το h, w είναι το ύψος και το πλάτος του πλαισίου σε εικονοστοιχεία. Η διαδικασία είναι ότι το $(h \times w)$ πλαίσιο διαιρείται σε ένα σύνολο απο $(H \times W)$ πλαίσια και στο κάθε υποπλαίσιο πραγματοποιείται η διαδικασία του max pooling. Σε αντίθεση με το R-CNN το Fast R-CNN αξιοποιεί την αρχιτεκτονική του VGG16 για την εξαγωγή χαρακτηριστικών και όχι την ίδια δομή με το AlexNet. Το VGG16[8] αποτελείται απο 16 στρώματα, 13 Convolutional layers και 3 Fully Connected layers, η διαφορά με το AlexNet πέρα απο τον αριθμό των ConvLayers είναι το μέγεθος των φίλτρων. Το AlexNet έχει 11×11 φίλτρα στο πρώτο στρώμα ενώ το VGG16 διαθέτει 3×3 φίλτρα σε όλα του τα στρώματα. Ένα ακόμα πλεονέκτημα του Fast R-CNN είναι ότι η δομή του δικτύου είναι μονομερής, δεν πραγματοποιούνται ξεχωριστά οι διαδικασίες όπως πρόταση περιοχών ενδιαφέροντος, εξαγωγή χαρακτηριστικών και τέλος ο εντοπισμός και η κατηγοριοποίηση.



Εικ. 2) Δομή Fast R-CNN

Faster R-CNN

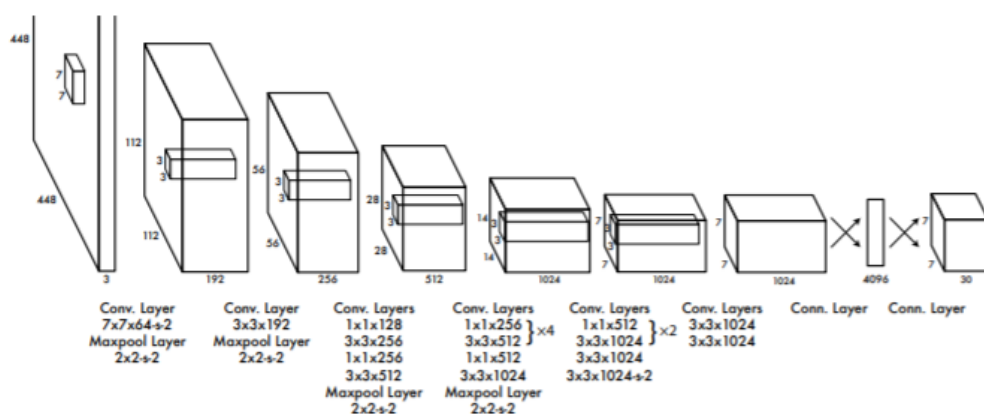
Το Faster R-CNN [9] είναι μια βελτιωμένη και πολύ πιο γρήγορη έκδοση του Fast R-CNN, αποτελείται από δύο τμήματα το deep fully convolutional network το οποίο στόχο έχει την πρόταση περιοχών για τον εντοπισμό των αντικειμένων και το δεύτερο τμήμα είναι το ίδιο το Fast R-CNN υπεύθυνο για τον εντοπισμό. Τα δύο τμήματα έχουν ενωθεί σε ένα ενιαίο δίκτυο με σκοπό τον εντοπισμό των αντικειμένων. Το πρώτο τμήμα ονομάζεται Region Proposal Network (RPN). Το RPN δέχεται ως είσοδο εικόνα οποιουδήποτε μεγέθους και το αποτέλεσμα είναι ένας αριθμός από περιοχές με σκόρ κατά πόσο πιθανόν και ποιά είναι τα αντικείμενα που μπορεί να υπάρχουν στην περιοχή. Οι περιοχές στη συνέχεια ελέγχονται από το Fast R-CNN και έτσι πραγματοποιείται ο εντοπισμός. Το χαρακτηριστικό που διαφοροποιεί το Faster R-CNN είναι ότι το RPN και το Fast R-CNN μοιράζονται Convolutional στρώματα μεταξύ των αρχιτεκτονικών τους. Με αποτέλεσμα η διαδικασία εκπαίδευσης να χρειάζεται λιγότερο χρόνο από το απλό Fast R-CNN. Επιπλέον στο Faster R-CNN εισάγεται ένας νέος όρος το Anchor Box. Τα Anchor Box είναι τα πλαίσια που παρέχει το RPN, δηλαδή για κάθε περιοχή της εικόνας παρέχεται ένας αριθμός από πλαίσια συνήθως είναι 9, τα πλαίσια έχουν διαφορετικά μεγέθη και οι διαστάσεις του πλάτους και του ύψους μπορούν να διαφέρουν. Για κάθε Anchor Box δίνεται η πιθανότητα αν περιέχει κάποιο αντικείμενο στο εσωτερικό του ή αν είναι κενό δηλαδή το φόντο της εικόνας. Με βάση το διαφορετικό μέγεθος των boxes για τον εντοπισμό των αντικειμένων πλέον απαιτείται μόνο η αρχική εικόνα και όχι παραλλαγές της σε περίπτωση που τα αντικείμενα διαφέρουν σε κλίμακα και μέγεθος.



Εικ. 3) RPN και τα Anchor Boxes που παρέχει για μια περιοχή της εικόνας

YOLO

Το YOLO [10] (You Only Look Once) είναι ένα καινοτόμο μοντέλο το οποίο συνδυάζει όλες τις ενέργειες ενός συστήματος εντοπισμού αντικειμένων σε ένα Convolutional Network το οποίο ταυτόχρονα προβλέπει bounding boxes και την πιθανότητα να περιέχουν κάποιο αντικείμενο. Το μοντέλο πραγματοποιεί εντοπισμό αντικειμένων κατευθείαν σε ολη την επιφάνεια της εικόνας, αυτό έχει ως αποτέλεσμα την ταχύτατη διαδικασία εντοπισμού ικανή άψογα να λειτουργήσει σε πραγματικό χρόνο. Το βασικό μοντέλο αποδίδει σε 45 καρε ανα δευτερόλεπτο σε μονάδα επεξεργασίας γραφικών Titan X ενώ η γρήγορη έκδοση φτάνει τα 150 καρέ το δευτερόλεπτο. Επιπλέον το YOLO μαθαίνει τα γενικά χαρακτηριστικά των αντικειμένων που έχει ως στόχο να εντοπίσει αλλά δυσκολεύεται στον εντοπισμό μικρών αντικειμένων. Όπως αναφέρθηκε προηγουμένως το YOLO εξετάζει όλη την εικόνα σαν σύνολο και χρησιμοποιεί όλα τα χαρακτηριστικά της εικόνας για να προβλέψει σε ολη την επιφάνεια τα bounding boxes μαζί με την κατηγορία του αντικειμένου ταυτόχρονα. Το μοντέλο αρχικά χωρίζει την εικόνα σε ένα πλήθος πλαισίων διαστάσεων $S \times S$ και αν το κέντρο του αντικειμένου βρίσκεται στο εσωτερικό του κελιού τότε το κελί αυτό είναι υπεύθυνο για τον εντοπισμό το αντικειμένου. Καθε bounding box αποτελείται απο 5 στοιχεία. Τις x και y συντεταγμένες του κέντρου του πλαισίου σε σύγκριση με την θέση του στο κελί. Το ύψος και το πλάτος του με βάση την συνολική εικόνα και τέλος το ποσοστό για το αν υπάρχει κάποιο αντικείμενο ή όχι στο εσωτερικό του bounding box. Η αρχιτεκτονική του βασικού μοντελου YOLO αποτελείται απο 24 Convolutional layers και 2 Fully Connected layers αρχιτεκτονική εμπνευσμένη απο το GoogLeNet [11]. Η γρήγορη έκδοση του YOLO αξιοποιεί 9 Convolutional layers αντι των 24αρων με το κάθε στρώμα να έχει και μικρότερο αριθμό φίλτρων. Η αποδοση του βασικού μοντελου στο ImageNet Validation set 2012 έφτασε το 88%.



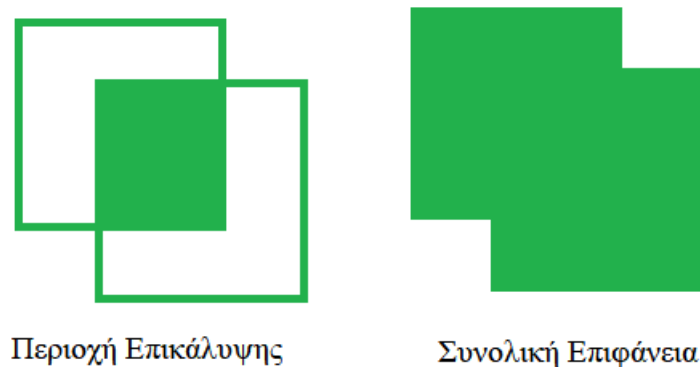
Εικ. 4) Αρχιτεκτονική YOLO 24 Convolutional layers και 2 Fully Connected layers

SSD (Single Shot Detector)

Το SSD[12] είναι ένα Convolutional δίκτυο τροφοδοσίας προς τα εμπρός το οποίο παρέχει καθορισμένους μεγέθους πλαίσια και αξιολογεί το κάθε ένα πλαίσιο για τις παρούσες κατηγορίες αντικειμένων που ενδεχομένως να περικλείουν. Το δίκτυο χρησιμοποιεί μια μέθοδο που ονομάζεται non-maximum suppression για τον τελικό εντοπισμό των αντικειμένων. Το non-maximum suppression επιτρέπει την επιλογή του κατάλληλου bounding box για το κάθε αντικείμενο από ένα σύνολο επικαλυπτόμενων boxes.

Συνήθως το κριτήριο για την επιλογή του σωστού πλαισίου αποτελεί το ποσοστό της πιθανότητας να υπάρχει στο πλαίσιο το ζητούμενο αντικείμενο και το IoU.

Το IoU είναι το Intersection over Union (Εικ. 5), και είναι ένα κλάσμα το οποίο περιγράφει το ποσοστό επικάλυψης ενός πλαισίου το οποίο πρόβλεψε το δίκτυο με τον πραγματικό πλαίσιο του αντικειμένου, συνήθως αν το ποσοστό ξεπερνάει το 0.5 θεωρείτε πως έγινε σωστή τοποθέτηση του πλαισίου που προέβλεψε το δίκτυο. Σε αντίθεση με το YOLO το SSD δεν δυσκολεύεται με τον εντοπισμό μικρών αντικειμένων. Για την επίλυση του προβλήματος αυτού στο τέλος του βασικού Convolution δικτύου εξαγωγής χαρακτηριστικών προστέθηκαν επιπλέον ConvLayers με στόχο την σταδιακή μείωση του μεγέθους των χαρτών χαρακτηριστικών. Η σταδιακή εμβάθυνση στα χαρακτηριστικά έχει ως αποτέλεσμα τον εντοπισμό αντικειμένων ανεξάρτητα του μεγέθους τους και των διαστάσεων τους.

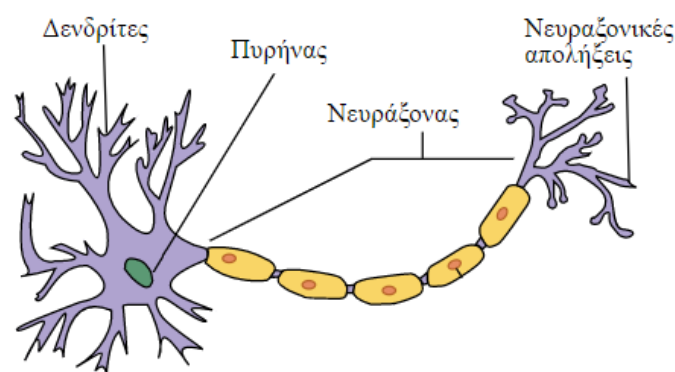


$$\text{IoU} = \frac{\text{Περιοχή Επικάλυψης}}{\text{Συνολική Επιφάνεια}}$$

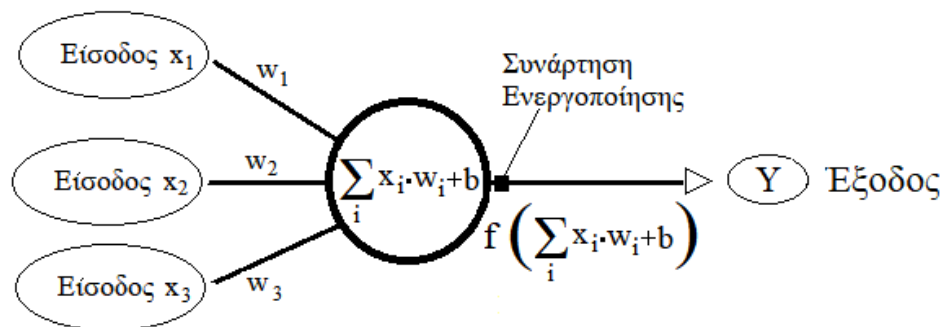
Εικ. 5) Τρόπος υπολογισμού IoU

1.4 ANN (Artificial Neural Network)

Η δομή ενός τεχνητού νευρωνικού δικτύου εμπνεύστηκε από το νευρωνικό δίκτυο του ανθρώπινου εγκεφάλου. Ένα νευρωνικό δίκτυο αποτελείται από δισεκατομμύρια νευρώνες, το καθένα αποτελείται από δενδρίτες, πυρήνα, τον νευράξονα και τις νευραξονικές απολήξεις (Εικ. 6). Τα ερεθίσματα γίνονται αντιληπτά από τους δενδρίτες με αποτέλεσμα να δημιουργείται ένας ηλεκτρονικός παλμός ο οποίος μεταδίδεται μέσω του άξονα στις απολήξεις οι οποίες συνδέονται με άλλα γειτονικά κύτταρα. Έτσι και το ANN (Εικ. 7) αποτελείται από ένα σύνολο κόμβων συνδεδεμένων μεταξύ που προσομοιάζουν τα νευρικά κύτταρα, δέχονται δεδομένα για είσοδο που επεξεργάζονται και προωθούν στα επόμενα κύτταρα.



Εικ. 6) Απεικόνιση νευρικού κυττάρου.



Εικ. 7) Απεικόνιση ενός κόμβου δικτύου ANN.

Τα (x) συμβολίζουν τα δεδομένα εισόδου και στον κόμβο στην προκειμένη περίπτωση έχουμε 3 δεδομένα εισόδου. Το (w) αντιπροσωπεύει τα βάρη, τα βάρη είναι κάποιες τιμές οι οποίες αντιστοιχούν με το πλήθος των δεδομένων εισόδου και ελέγχουν την ισχύ του δεσμού μεταξύ των κόμβων, επειδή ανάλογα με την τιμή τους επηρεάζεται η έξοδος του κόμβου. Με την είσοδο των δεδομένων πραγματοποιείται ο πολλαπλασιασμός τους με τα βάρη και τέλος προστίθεται μια σταθερά το bias πριν περασει το συνολικό αποτέλεσμα στην συνάρτηση ενεργοποίησης.

Τα βάρη και τα δεδομένα εισόδου είναι μονοδιάστατοι πίνακες.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n_x} \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_{n_x} \end{pmatrix}$$

(1) Η έξοδος του κόμβου πριν την επιρροή της συνάρτησης ενεργοποίησης.

$$(1) \quad z = \mathbf{x} * \mathbf{w} + \mathbf{b}$$

(2) Η έξοδος με την επιρροή της συνάρτησης ενεργοποίησης.

$$(2) \quad y = f(z) = f(\mathbf{x} * \mathbf{w} + \mathbf{b})$$

Όπου:

- y -> τελική έξοδος.
- $f(z)$ -> συνάρτηση ενεργοποίησης
- w -> διάνυσμα βαρών
- b -> σταθερά bias
- x -> δεδομένα εισόδου

1.4.1 Μεθοδολογίες Εκπαίδευσης ANN

Για να χρησιμοποιηθεί ένα τεχνητό νευρωνικό δίκτυο για την επίλυση κάποιου προβλήματος απαιτείται αρχικά η εκπαίδευσή του. Η εκπαίδευση μπορεί να γίνει με τους εξής τρεις τρόπους a) Supervised Learning (Εποπτευόμενη Μάθηση), b) Unsupervised Learning (Μη εποπτευόμενη Μάθηση) και c) Reinforcement Learning (Μάθηση με Ενίσχυση).

Εποπτευόμενη Μάθηση

Με την Εποπτευόμενη Μάθηση ο αλγόριθμος εκπαιδεύτε σε δεδομένα τα οποία έχουν ετικέτες, δηλαδή πληροφορίες σχετικές με το περιεχόμενο τους. Ο σκοπός κατά την εποπτευόμενη μάθηση είναι να μπορέσει ο αλγόριθμος καθώς εκπαιδεύτε με τα δεδομένα εισόδου να αναπροσαρμόζει τα βάρη ώστε η τιμή της τελικής πρόβλεψης να αντιστοιχεί σωστά στις ετικέτες. Αρχικά ο αλγόριθμος γνωρίζει τα δεδομένα εισόδου και όλες τους τις ετικέτες, για κάθε διάνυσμα δεδομένων που περνάει απο το δίκτυο, το δίκτυο προβλέπει μια τιμή και την συγκρίνει με την πραγματική τιμή που αντιστοιχεί στον συγκεκριμένο διάνυσμα. Η απόκλιση των δύο τιμών ονομάζεται το σφάλμα (error) και με βάση το σφάλμα γίνεται αναπροσαρμογή των τιμών των βαρών. Με αποτέλεσμα μετά απο πολλές τέτοιες επαναλήψεις το δίκτυο να μαθαίνει να προβλέπει τιμές για τα δεδομένα με ελάχιστη ή και μηδενική απόκλιση και όταν δοκιμαστεί σε νέα δεδομένα ανάλογα με αυτά που είχε εκπαιδευτεί χωρίς να γνωρίζει τις ετικέτες τους θα μπορέσει να προβλέψει σωστά το αποτέλεσμα.

Τα βήματα για την διεξαγωγή Εποπτευόμενης Μάθησης είναι:

1. Επιλογή του είδους των δεδομένων ενδιαφέροντος.
2. Συγκέντρωση δεδομένων για εκπαίδευση.
3. Προσθήκη ετικετών στα δεδομένα εκπαίδευσης
4. Εκπαίδευση του ANN και δοκιμή του σε άγνωστα δεδομένα για τον έλεγχο ποιότητας εκπαίδευσης.

Μη εποπτευόμενη Μάθηση

Όπως το αναφέρει και ο τίτλος κατά την Μη εποπτευόμενη Μάθηση δεν υπάρχουν ετικέτες που να αντιστοιχούν στα δεδομένα εκπαίδευσης αρα δεν επιβλέπετε απο κάποιον παράγοντα το δίκτυο. Το δίκτυο κατα την εκπαίδευση απο μόνο του ανακαλύπτει και μαθαίνει μοτίβα τα οποία υπάρχουν στο σετ δεδομένων. Η Μη εποπτευόμενη Μάθηση χρησιμοποιείται σε μοντέλα ANN για την επίλυση προβλημάτων κυρίως όπως το a) clustering, b) συσχέτισης και c) μείωσης χαρακτηριστικών δεδομένων στο στάδιο της προεπεξεργασίας.

a) Clustering

Το clustering χρησιμοποιείται για την ομαδοποίηση μεγάλου όγκου δεδομένων με βάση τις ομοιότητες και τις διαφορές τους. Τα δεδομένα που εισέρχονται σε έναν αλγόριθμο clustering δεν έχουν υποστεί προεπεξεργασία, με την εκπαίδευση ο αλγόριθμος εντοπίζει δομές και μοτίβα τα οποία επιτρέπουν την ομαδοποίηση τους.

b) Συσχέτιση

Η συσχέτιση είναι μια μέθοδος για τον εντοπισμό των σχέσεων εξάρτησης μεταξύ των μεταβλητών των δεδομένων. Χρησιμοποιείται κυρίως στο μάρκετινγκ απο τις εταιρείες για τον εντοπισμό των σχέσεων που αναπτύσσονται μεταξύ των προϊόντων απο τις συνθήκες της αγοράς. Βοηθάει στην κατανόηση των συνηθειών των καταναλωτών και την ανάπτυξη κατάλληλων τεχνικών για την μεγιστοποίηση των πωλήσεων.

c) Μείωση Χαρακτηριστικών Δεδομένων

Συνήθως όσο μεγαλύτερο είναι ένα σετ δεδομένων για εκπαίδευση τόσο το καλύτερο για το δίκτυο τεχνητής νοημοσύνης, δημιουργείται όμως ένα πρόβλημα, αν τα δεδομένα περιέχουν μεγάλο αριθμό χαρακτηριστικών τα οποία θα λάβει υπόψη του το δίκτυο τότε η ευστοχία πρόβλεψης μειώνεται δραματικά. Για τον λόγο αυτό δημιουργήθηκαν αλγόριθμοι οι οποίοι επεξεργάζονται τα δεδομένα και εξάγουν περιττά χαρακτηριστικά χωρίς όμως να αλλοιώνουν την ουσία τους.

Μάθηση με Ενίσχυση

Η Μάθηση με Ενίσχυση αποτελεί έναν ιδιαίτερο τρόπο εκπαίδευσης δικτύων τεχνητής νοημοσύνης. Κατά την διαδικασία της μάθησης με ενίσχυση το μοντέλο δεν δέχεται δεδομένα επεξεργασμένα με ετικέτες όπως γίνεται στην περίπτωση της εποπτευόμενης μάθησης. Τα δεδομένα εισόδου είναι ερεθίσματα απο το περιβάλλον στο οποίο δραστηριοποιείται ο αλγόριθμος, δηλαδή το ίδιο το μοντέλο παράγει τα δεδομένα.

Με βάση τις πράξεις που επιλέγει να εκτελέσει ενεργοποιείται ένας μηχανισμός ανταμοιβής ο οποίος και καθορίζει αν οι επιλογές του μοντέλου ήταν σωστές ή όχι. Ένα απο τα προβλήματα σε αυτόν τον τύπο εκπαίδευσης είναι ότι για να αποκτήσει γνώση το μοντελο πρέπει συνέχεια να προκαλεί δράση και να μαθαίνει με βάση το αποτέλεσμα, αυτο προϋποθέτει συνεχόμενες δοκιμές ακόμα και αν φέρουν αποτυχημένο αποτέλεσμα αρα και κατανάλωση χρόνου. Επίσης το μοντελο πρέπει να μάθει να συνδυάζει την γνώση που έχει για να δημιουργήσει νέα και να μην αρκείτε σε ένα συγκεκριμένο μοτίβο.

Μάθηση με Ενίσχυση έχει χρησιμοποιηθεί σε τομείς όπως:

- a) Αυτόνομη κίνηση ρομποτ.
- b) Εξατομικευμένες Προτάσεις (π.χ. Διαφημισεις, Ειδησεις, Προϊόντα)
- c) NLP (Neural Language Processing)
- d) Παιχνίδια (π.χ. Σκάκι και Go)

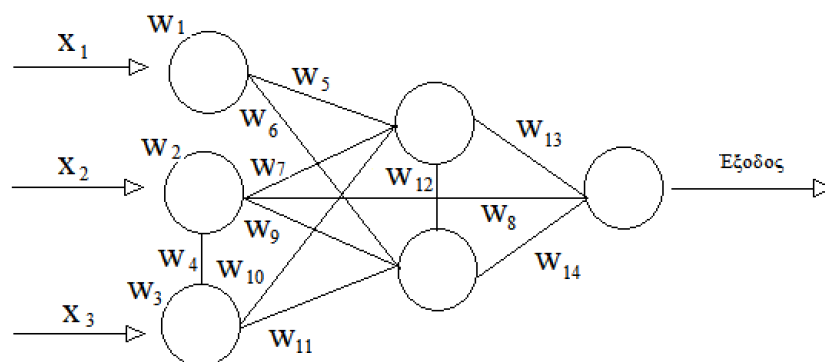
1.4.2 Κατηγορίες ANN

FeedForward

Η σύνδεση των κόμβων σε ένα Feedforward δίκτυο τεχνητής νοημοσύνης είναι γραμμική προς τα εμπρός, δεν σχηματίζονται κάποιες κυκλικές ενώσεις, επίσης δεν είναι δυνατή η ροή πληροφορίας στην ίδια σύνδεση και προς τις δύο κατευθύνσεις. Αποτελούν το πρώτο είδος δικτύων που εφευρέθηκαν και έχουν την πιο απλή δομή. Η πληροφορία εισερχεται στο δίκτυο από το πρώτο στρώμα κόμβων που ονομάζονται κόμβοι εισόδου (input nodes) στη συνέχεια η πληροφορία μεταδίδεται στο επόμενο στρώμα κρυφών κόμβων (hidden layer), σε ένα δίκτυο μπορούν να υπάρχουν περισσότερα από 1 hidden layers και τέλος οι κρυφοί κόμβοι ενώνονται με τους κόμβους εξόδου (output nodes) οι οποίοι και προβάλλουν το τελικό αποτέλεσμα. Τα Feedforward δίκτυα χρησιμοποιούνται κυρίως σε περιπτώσεις όπου τα δεδομένα δεν έχουν κάποια διαδοχική σχέση μεταξύ τους και για την εκπαίδευσή τους εφαρμόζεται εποπτευόμενη μάθηση.

FeedBack

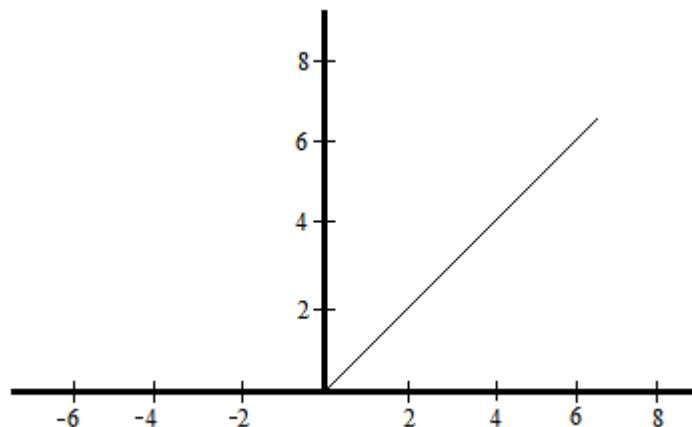
Ένα feedback (Εικ. 8) δίκτυο τεχνητής νοημοσύνης σε αντίθεση με το feedforward δίκτυο μπορεί να μεταφέρει πληροφορίες από κόμβο σε κόμβο και στις δύο κατευθύνσεις. Στο εσωτερικό του δικτύου σχηματίζονται πολύπλοκες συνδέσεις και οι εξόδους των κόμβων μπορεί να μεταδίδεται κυκλικά δημιουργώντας μια κατάσταση που προσομοιάζει την μνήμη. Το δίκτυο είναι δυναμικό και η κατάστασή του συνεχώς μεταβάλλεται έως ότου βρεθεί σε ισορροπία, μόλις βρεθεί σε κατάσταση ισορροπίας παραμένει σε αυτήν, αν όμως εξαιτίας κάποιας αλλαγής στα δεδομένα εισόδου χρειαστούν περαιτέρω υπολογισμοί το δίκτυο μεταβάλλεται και αναπροσαρμόζεται μέχρι να βρεθεί πάλι σε κατάσταση ισορροπίας.



Εικ. 8) FeedForward Artificial Neural Network

1.4.3 Συνάρτηση Ενεργοποίησης

Κάθε στρώμα ενός Convolutional δικτύου χρησιμοποιεί μια συνάρτηση ενεργοποίησης, από τις πιο διαδεδομένες συναρτήσεις για την επίλυση του προβλήματος του εντοπισμού των αντικειμένων είναι η ReLU (Rectified Linear Unit). Μαθηματικά ορίζεται ως $f(x) = \max(0, x)$. Η συνάρτηση είναι γραμμική, για θετικές τιμές το αποτέλεσμα είναι θετικό ενώ για αρνητικές τιμές μηδενίζεται (Εικ. 9). Ο σκοπός της συνάντησης ενεργοποίησης είναι να βοηθήσει το δίκτυο να μάθει περίπλοκα μοτίβα από τα δεδομένα εισόδου για την καλύτερη διαδικασία αναγνώρισης και εντοπισμού. Κάθε νευρώνας δέχεται μια είσοδο δεδομένων (input) την οποία επεξεργάζεται με ένα σύνολο βαρών W , τέλος προστίθεται μια σταθερά το bias (b), δηλαδή έχουμε $x = \text{input} * W + b$. Το αποτέλεσμα αυτό αποτελεί την είσοδο της συνάρτησης ενεργοποίησης και το αποτέλεσμα της συνάρτησης είναι πλέον η είσοδος στον επόμενο κόμβο. Η διαδικασία ονομάζεται forward propagation και σκοπό έχει τον υπολογισμό της απόκλισης της πραγματικής τιμής εντοπισμού με αυτήν που προέβλεψε το δίκτυο. Με βάση την απόκλιση ρυθμίζονται οι τιμές των βαρών (W) για την επόμενη επανάληψη της διαδικασίας, έτσι το δίκτυο μαθαίνει να αναγνωρίζει σωστά τα δεδομένα εισόδου.



Εικ. 9) Γραφική αναπαράσταση της ReLU.

1.4.4 Συνάρτηση Απώλειας.

Προηγουμένως έγινε αναφορά ότι για να λυθεί ένα πρόβλημα με την χρήση κατάλληλου δικτύου τεχνητής νοημοσύνης πρώτα πρέπει να συγκεντρώσουμε δεδομένα τα οποία ταιριάζουν στις απαιτήσεις του προβλήματος και στη συνέχεια να εκπαιδεύσουμε το μοντέλο. Ας πάρουμε για παράδειγμα ένα πρόβλημα κατηγοριοποίησης, έστω έχουμε ένα σετ δεδομένων με δύο κατηγορίες και κάθε εικόνα ανήκει είτε στην Α είτε στην Β κατηγορία. Κατά την εκπαίδευση ο αλγόριθμος δέχεται τις εικόνες και ύστερα απο επεξεργασία προσπαθεί να προβλέψει την κατηγορία στην οποία ανήκει η κάθε εικόνα. Μετά απο κάθε πρόβλεψη η συνάρτηση απώλειας υπολογίζει την διαφορά μεταξύ της πραγματικής τιμής που αντιστοιχεί στα δεδομένα με αυτήν που προέβλεψε ο αλγόριθμος. Με βάση το αποτέλεσμα της απόκλισης δημιουργείται ένα σήμα το οποίο ενημερώνει το μοντέλο σχετικά με την επίδοση του δηλαδή αξιολογεί πόσο καλά καταφέρνει και προβλέπει τις κατηγορίες. Υπάρχουν δύο κύριες κατηγορίες συναρτήσεων απώλειας a) Classification και b) Regression.

a) Classification Losses

- ❖ Cross-entropy loss (Binary, Categorical, Sparse)
- ❖ Logarithmic loss
- ❖ Exponential loss
- ❖ Hinge loss

b) Regression Losses

- ❖ Mean Square Error loss
- ❖ Mean Absolute Error loss
- ❖ Huber loss
- ❖ Mean Absolute Percentage Error
- ❖ Mean Squared Logarithmic Error
- ❖ Log Cosh

1.4.5 Εφαρμογές Τεχνητών Νευρωνικών Δικτύων

Η ραγδαία ανάπτυξη στο χώρο της τεχνητής νοημοσύνης επιτρέπει την επίλυση πολύπλοκων προβλημάτων με την χρήση των υπολογιστών. Τα μοντέλα τεχνητής νοημοσύνης προσομοιάζουν τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου με αποτέλεσμα οι μηχανές να αποκτούν ικανότητες όπως πρόβλεψεις και μοντελοποίησης περίπλοκων μοτίβων. Παρακάτω παρουσιάζονται μερικοί από τους τομείς στους οποίους αξιοποιούνται οι δυνατότητες των ANN.

1. Ομιλία: Αναγνώριση ομιλίας, Μετατροπή ομιλίας σε κείμενο και το αντιστρόφο, Κατηγοριοποίηση ομιλίας.
2. Συμπίεση και Φιλτράρισμα Δεδομένων.
3. Βιομηχανία: Έλεγχος διαδικασιών κατασκευής προϊόντων, Έλεγχος ποιότητας, Γενική διαχείριση.
4. Ρομποτική: Αυτονομία κίνησης, Οπτική αναγνώριση.
5. Ιατρική: Ανάλυση καρκινικών κυττάρων, Προσθετική, Ανάλυση ΗΚΓ.
6. Αναγνώριση προσώπων, χαρακτήρων και μοτίβων.
7. Επικοινωνία και Εξυπηρέτηση.
8. Αξιολόγηση ακίνητης περιουσίας.
9. Χρηματοοικονομική ανάλυση.
10. Επεξεργασία και Κατηγοριοποίηση σημάτων.

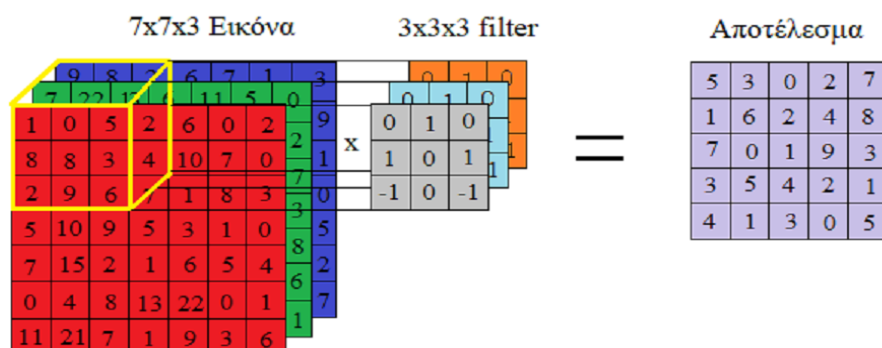
1.5 Convolutional Neural Network

Ας αρχίσουμε πρώτα με την απάντηση στην ερώτηση “Τι είναι ένα Convolutional Neural Network”. Το CNN είναι ένας αλγόριθμος βαθιάς μάθησης και ανήκει στην κατηγορία των τεχνητών νευρωνικών δικτύων, μπορεί να δεχτεί ως είσοδο μια εικόνα και ύστερα απο την επεξεργασία της να περιγράψει τα είδη και την κατηγορία στην οποία ανήκουν τα αντικείμενα της εικόνας. Δηλαδή το μοντέλο έχει την δυνατότητα να μαθαίνει και να αναγνωρίζει οπτικά μοτίβα απο τις εικόνες.

Αρχιτεκτονική CNN

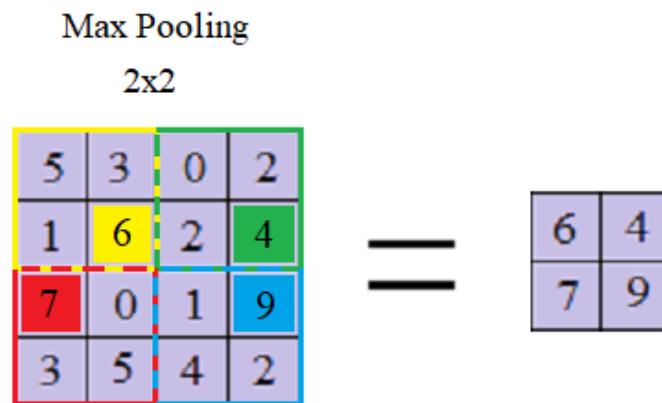
Η δομή ενός Convolutional Neural Network αποτελείται κυρίως απο τρία στρώματα a) Convolutional layer, b) Pooling layer and c) Fully Connected layer. Το κάθε στρώμα έχει συγκεκριμένο ρόλο και λειτουργίες.

- a) Convolutional layer: Η κάθε πολύχρωμη εικόνα που εισέρχεται στο δίκτυο στην πραγματικότητα αποτελείται απο τρεις δισδιάστατους πίνακες το περιεχόμενο των οποίων είναι οι χρωματικές τιμές των εικονοστοιχείων. Οι πίνακες είναι τρεις επειδή μια πολύχρωμη εικόνα έχει τρια χρωματικά κανάλια (RGB), απο ένα κανάλι για το κάθε χρώμα κόκκινο (Red), πράσινο (Green) και μπλε (Blue). Στο στρώμα αυτό υπάρχουν πίνακες ή αλλιώς φίλτρα, ο αριθμός των φίλτρων σε καθε convolutional layer μπορεί να διαφέρει αλλά το μεγεθος τους όχι. Συνήθως τα μεγέθη τους είναι 3x3, 5x5, 7x7 ακομα και 9x9 . Το κάθε φίλτρο περιέχει κάποιες τυχαίες τιμές, όταν η εικόνα εισέρχεται στο στρώμα το φίλτρο πολλαπλασιάζεται με την αντίστοιχη επιφάνεια της εικόνας και το συνολικό άθροισμα αποθηκεύεται σε έναν νέο πίνακα. Η διαδικασία επαναλαμβάνεται έως ότου το φίλτρο θα περάσει απο ολη την επιφάνεια της εικόνας και αυτό θα γίνει για καθε φίλτρο (Εικ. 10). Οι νέοι πίνακας που δημιουργείται απο την διαδικασία ονομάζονται χάρτες χαρακτηριστικών και αποτελούν την εισοδο για τα επόμενα στρώματα.



Εικ. 10) Απεικόνιση διαδικασίας Convolution.

- b) Pooling layer: Το στρώμα αυτό είναι υπεύθυνο για την μείωση του μεγέθους των δεδομένων εισόδου, δηλαδή την μείωση του ύψους και του πλάτους. Βεβαία η διαδικασία σμίκρυνσης οδηγεί στην απώλεια δεδομένων αλλά μειώνει τον αριθμό των υπολογισμών και βοηθάει στην εξαγωγή από την εικόνα των σημαντικότερων χαρακτηριστικών των αντικειμένων. Χαρακτηριστικά τα οποία δεν επηρεάζονται από την θέση και την γωνία τους στην εικόνα. Υπάρχουν τρεις κατηγορίες Pooling, το Max Pooling το Average Pooling και το Global Pooling. Όπως αναφέρεται προηγουμένως η έξοδος του Convolutional layer αποτελεί την είσοδο του Pooling layer. Στην περίπτωση του Max Pooling οι χάρτες χαρακτηριστικών αποτελούνται από πολλούς πίνακες, για να μειωθούν οι διαστάσεις τους επιλέγεται ένα ακόμα κενό φίλτρο μεγέθους συνήθως 2x2 ή 3x3 και σαρώνει όλη την επιφάνεια του πίνακα, σε κάθε 2x2 ή 3x3 παράθυρο αντιστοιχούν 4 ή 9 ψηφία του ίδιου του πίνακα, από αυτά επιλέγεται το μεγαλύτερο και αποθηκεύεται σε νέο πίνακα (Εικ. 11).



Εικ. 11) Απεικόνιση διαδικασίας Max Pooling: Από κάθε 2x2 παράθυρο επιλέγεται ο μεγαλύτερος αριθμός.

Η διαδικασία είναι παρόμοια στο Average Pooling με την διαφορά ότι αναλογα με το μεγεθος του φίλτρου προστίθενται ολοι οι αριθμοί και το αποτελεσμα διαιρείται με το πλήθος των ψηφίων, ο αριθμος που προκυπτει αποθηκεύεται σε νέο πίνακα (Εικ. 12).

Average Pooling 2x2

2	3	0	2
1	6	2	4
7	0	1	9
3	2	4	2

=

3	2
3	4

Εικ. 12) Απεικόνιση διαδικασίας Average Pooling: Σε κάθε 2x2 παράθυρο τα τέσσερα ψηφία αθροίζονται και διαιρούνται με το πλήθος τους, ετσι προκύπτει ο νέος πίνακας.

Υπολογισμός περιεχομένου του νέου πίνακα.

Παραθυρο 1: $2 + 3 + 1 + 6 = 12 \Rightarrow 12 / 4 = 3$

Παραθυρο 2: $0 + 2 + 2 + 4 = 8 \Rightarrow 8 / 4 = 2$

Παραθυρο 3: $7 + 0 + 3 + 2 = 12 \Rightarrow 12 / 4 = 3$

Παραθυρο 4: $1 + 9 + 4 + 2 = 16 \Rightarrow 16 / 4 = 4$

Τέλος το Global Pooling διαθέτει δύο μεθόδους, το Global Max Pooling (Εικ. 13) και το Global Average Pooling (Εικ. 14). Ανάλογα το είδος του Global Pooling που θα χρησιμοποιηθεί κάθε πίνακας του χάρτη χαρακτηριστικών μειώνεται σε μια τιμή.

Global Max Pooling

2	3	0	2
1	6	2	4
7	0	1	9
3	2	4	2

=

9

Εικ. 13) Απεικόνιση διαδικασίας Global Max Pooling: Επιλέγεται μόνο το μεγαλύτερο ψηφίο.

Global Average Pooling

2	3	0	2
1	6	2	4
7	0	1	9
3	2	4	2

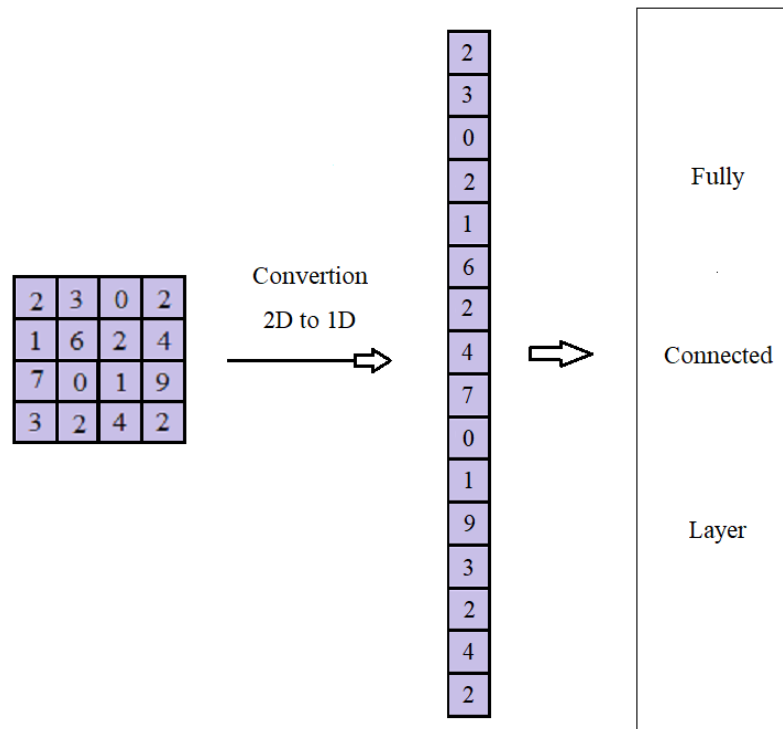
=

3

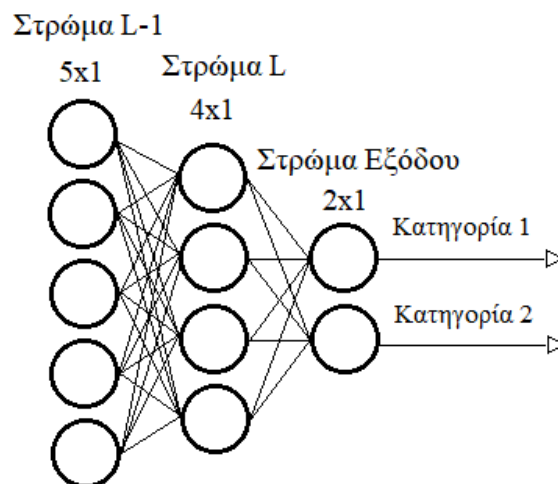
Εικ. 14) Απεικόνιση διαδικασίας Global Average Pooling: Όλα τα ψηφία του πίνακα προστίθενται και το σύνολο διαιρείται με το πλήθος τους.

Global Average Pooling: $2 + 3 + 0 + 2 + 1 + 6 + 2 + 4 + 7 + 0 + 1 + 9 + 3 + 2 + 4 + 2 = 48$
 $48 / 16 = 3$

- c) Fully Connected layer: Οι χάρτες χαρακτηριστικών αποτελούνται από δισδιάστατους πίνακες, πριν περάσουν τα δεδομένα των πινάκων στο πλήρως συνδεδεμένο στρώμα κόμβων πραγματοποιείται η μετατροπή τους σε μονοδιάστατα διανύσματα (Εικ. 15). Το στρώμα αυτό είναι ένα απλό τεχνητό νευρωνικό δίκτυο τροφοδοσίας προς τα εμπρός (Εικ. 16).



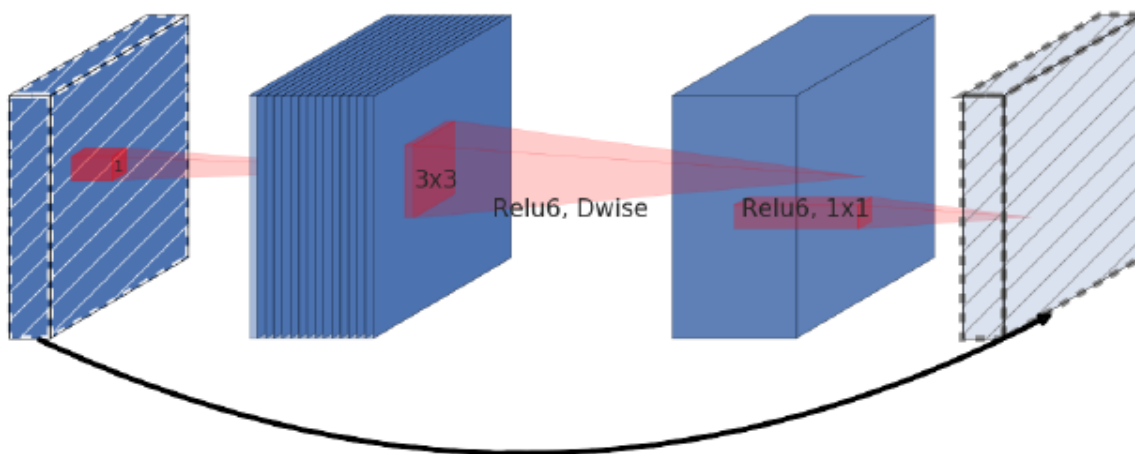
Εικ. 15) Απεικόνιση διαδικασίας μετατροπής πίνακα σε γραμμικό διάνυσμα.



Εικ. 16) Απεικόνιση ενός πλήρως συνδεδεμένου δικτύου.

1.6 Mobilenet V2

Το Mobilenet V2 [13] ανήκει στην οικογένεια των δικτύων τεχνίτης νοημοσύνης Mobilenet αναπτυγμένων ειδικά για τον σκοπό του εντοπισμού αντικειμένων, κατηγοριοποίησης και σημασιακού καταμερισμού σε συσκευές με περιορισμένη υπολογιστική ισχύς, όπως έξυπνα τηλέφωνα, ταμπλέτες και μικροϋπολογιστές. Εξαιτίας της αρχιτεκτονικής του δικτύου ο αριθμός των απαιτούμενων υπολογισμών μειώνεται σημαντικά με αποτέλεσμα και η ποσότητα μνήμης που χρειάζεται για να λειτουργήσει το δίκτυο να μειώνεται αντίστοιχα. Η βασική δομή του δικτύου αποτελείται από Bottleneck Depth-Separable Convolution μπλοκ (Εικ. 17). Σε αντίθεση με την κλασική διαδικασία που πραγματοποιείται σε ένα Convolution μπλοκ το νέο μπλοκ αποτελείται από ένα στρώμα επέκτασης διαστάσεων 1×1 και το Depthwise Separable Convolution το οποίο διαθέτει δύο στρώματα. Το πρώτο στρώμα επέκτασης με το 1×1 convolution έχει ως στόχο να μεγαλώσει τις διαστάσεις των χαρτών χαρακτηριστικών που αποτελούν τα δεδομένα εισόδου έτσι ώστε να χρησιμοποιηθεί μη γραμμική συνάρτηση ενεργοποίησης ReLU6. Το Depthwise Separable Convolution αποτελείται από το Depthwise Convolution όπου μόνο ένα φίλτρο αντιστοιχεί σε κάθε κανάλι εισόδου για την διαδικασία του convolution και το δεύτερο στρώμα convolution με φίλτρο διαστάσεων 1×1 το οποίο δημιουργήσει καινούργια χαρακτηριστικά μέσω των γραμμικών συνδυασμών των καναλιών εισόδου επαναφέροντας τις διαστάσεις του νέου χάρτη χαρακτηριστικών στο μέγεθος του αρχικού. Ένα ακόμα χαρακτηριστικό των Inverted Residual μπλοκ είναι η σύνδεση μεταξύ του πρώτου convolution στρώματος και του τελευταίου. Δηλαδή των στρωμάτων με τις λιγότερες διαστάσεις καναλιών, κάτι που επιτρέπει στο δίκτυο την δυνατότητα πρόσβασής στις προηγούμενες ενεργοποιήσεις νευρώνων.



Εικ. 17) Απεικόνιση ενός Bottleneck Depth-Separable Convolution μπλοκ [13].

Τέλος με την εξαιρετικά αποδοτική αρχιτεκτονική του το Mobilenet V2 μπορεί να χρησιμοποιηθεί ως βάση σε ένα δίκτυο με σκοπό τον εντοπισμό αντικειμένων. Το Mobilenet V2 λειτουργεί ως εξαγωγέας χαρακτηριστικών και συνδυάζεται με μια εναλλακτική έκδοση του SSD (Single Shot Detector) όπου κάθε convolutional στρώμα αντικαταστάθηκε από ένα Depthwise Separable στρώμα μειώνοντας ακόμα περισσότερο το σύνολο των υπολογισμών κατά την διαδικασία της εκπαίδευσης και του εντοπισμού σε σύγκριση με το κλασικό SSD μοντέλο.

Εισαγωγή

Στο Κεφάλαιο 2 θα γίνει αναφορά στα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής όπως το Android Studio, το Tensorflow Object Detection API, η συσκευή πάνω στην οποία χτίστηκε και δοκιμάστηκε το σύστημα αλλά και η γλώσσα προγραμματισμού που αξιοποιήθηκε για τον σκοπο της εργασιας.

2.1 Raspberry Pi 3 B+

Το Raspberry Pi 3 B+ είναι ένας μικροϋπολογιστής μονής πλακέτας μεγέθους παλάμης, παρόλο το μικρό μέγεθος του είναι ικανό να τρέξει λειτουργικά συστήματα όπως Linux και Android. Το 2012 κυκλοφόρησε η πρώτη έκδοση το Raspberry Pi 1 Model B+ με συχνότητα επεξεργαστή στα 700 MHz, χαμηλής κατανάλωσης ενεργειακά μονάδας επεξεργασίας γραφικών VideoCore IV και συνολική μνήμη τα 512 Mb. Επιπλέον η συσκευή διαθέτει καρφίτσες εισόδου/εξόδου γενικής χρήσης (GPIO) και υποδοχές USB και HDMI που επιτρέπουν στον χρήστη να συνδέση εύκολα άλλες εξωτερικές συσκευές όπως πληκτρολόγιο, ποντίκι και οθόνη ενώ μέσω των GPIO καρφιτσών μπορούν να συνδεθούν διάφορη αισθητήρες όπως ανίχνευσης κίνησης, θερμοκρασίας, υγρασίας και υπέρυθρου φωτός. Το χαμηλό κόστος και το πλήθος των δυνατοτήτων που προσφέρει μια συσκευή στο μέγεθος πιστωτική κάρτας αποδείχτηκαν το κλειδί της επιτυχίας για την εξαπλωση του Raspberry Pi στην αγορά αλλά και στον κόσμο της τεχνολογίας. Έτσι η εταιρεία Raspberry Pi Foundation συνέχισε το έργο της δημιουργώντας νέα μοντέλα με βελτιωμένα χαρακτηριστικά. Το 2018 κυκλοφόρησε στην αγορά το Raspberry Pi 3 B+ με κύρια χαρακτηριστικά τον τετραπύρηνο επεξεργαστή στα 1,4GHz 64 bit και το 1 GB μνήμης RAM που επιτρέπουν στην συσκευή να ανταπεξέλθει ικανοποιητικά στις ανάγκες μιας εφαρμογής που χρησιμοποιεί αλγόριθμο εντοπισμού αντικειμένων ειδικά σχεδιασμένο για συσκευές με περιορισμένη υπολογιστική ισχύς.

Το Raspberry Pi 3 B+ (Εικ. 18) είναι η συσκευή που επιλέχθηκε για την ανάπτυξης της εφαρμογής έξυπνης κάμερας, ο τρόπος λειτουργίας και η δομή της θα αναλυθούν στο επόμενο κεφάλαιο.



Εικ. 18) Απεικόνιση του Raspberry Pi 3 B+.

2.2 Tensorflow Object Detection API

Στο προηγούμενο κεφάλαιο έγινε αναφορά στον αλγόριθμο τεχνητής νοημοσύνης Mobilenet V2, ο οποίος αποτελεί έναν υπολογιστικά οικονομικό εξαγωγέα χαρακτηριστικών και ο συνδυασμός του με την τροποποιημένη έκδοση του SSD (Single Shot Detector) έχουν ως αποτέλεσμα την δημιουργία ενός αλγορίθμου ικανού να εντοπίζει αντικείμενα σε πραγματικό χρόνο και να επιστρέφει την κατηγορία στην οποία ανήκει το αντικείμενο, το ποσοστό ακρίβειας του εντοπισμού αλλά και της συντεταγμένες του. Για να μπορέσει όμως ο SSD-Mobilenet V2 να αναγνωρίσει τις τρεις κατηγορίες χειρονομιών που ήταν απαραίτητες για την εργασία, έπρεπε να γίνει η εκπαίδευση του με εικόνες των αντίστοιχων χειρονομιών και η διαδικασία αυτή επιτεύχθηκε με την βοήθεια του Tensorflow Object Detection API.

Το Tensorflow [14] είναι μια βιβλιοθήκη αριθμητικών υπολογισμών και μηχανικής μάθησης, ο κώδικας της οποίας είναι διαθέσιμος στο διαδίκτυο για κοινή χρήση. Αναπτύχθηκε από την ομάδα Google Brain της Google με σκοπό την δημιουργία μοντέλων μηχανικής μάθησης για την επίλυση πολύπλοκων προβλημάτων όπως της μηχανικής όρασης, αναγνώρισης ομιλίας, πρόβλεψης καταστάσεων και ανάλυσης δεδομένων. Το Tensorflow Object Detection API χρησιμοποιεί ως βάση την βιβλιοθήκη Tensorflow και επιπλέον παρέχει έναν αριθμό από έτοιμους προ-εκπαιδευμένα μοντέλα για τον σκοπό του εντοπισμού αντικειμένων με το κάθε έναν να διαφοροποιείται στην αρχιτεκτονική του, την ταχύτητα και το ποσοστό ευστοχίας πρόβλεψης. Επίσης επιτρέπει στους χρήστες τοπικά στον υπολογιστή τους να πραγματοποιήσουν την εκπαίδευση των αλγορίθμων με δικά τους δεδομένα για την προσαρμογή του στις συνθήκες του προβλήματος που θέλουν να αντιμετωπίσουν.

2.3 Android Studio και Java

Η ανάπτυξη της εφαρμογής έγινε με την χρήση του περιβάλλοντος Android Studio και της προγραμματιστικής γλώσσας Java. Το Android Studio είναι ειδικά διαμορφωμένο για την αποτελεσματική και άμεση δημιουργία εφαρμογών για συσκευές οι οποίες λειτουργούν με το λειτουργικό σύστημα Android, όπως έξυπνα τηλέφωνα, τηλεοράσεις, ταμπλέτες και φορητές συσκευές. Επιπλέον επιτρέπει στον χρήστη καθόλη την διάρκεια δημιουργίας της εφαρμογής την δοκιμή της είτε σε φυσική συσκευή είτε σε εικονική συσκευή της επιλογής του. Δημιουργήθηκε από την κοινή συνεργασία της Google και της JetBrains, η πρώτη επίσημη έκδοση του Android Studio κυκλοφόρησε στην αγορά το 2014.

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Java. Είναι αντικειμενοστραφής γλώσσα γενικού σκοπού με κύριο χαρακτηριστικό την ευελιξία της. Δηλαδή κώδικας γραμμένος σε Java λειτουργεί σε όλες τις πλατφόρμες ενώ η σύνταξη της γλώσσας είναι παρόμοια με την C++ και C. Δημιουργός της Java είναι ο James Arthur Gosling και κυκλοφόρησε το 1995.

Η Εφαρμογή

Εισαγωγή

Στο Κεφάλαιο 3 περιγράφονται οι διαδικασίες εκπαίδευσης του αλγορίθμου μηχανικής όρασης, η προετοιμασία του σετ εκπαίδευσης, η πρόσβαση στην usb κάμερα και ο τρόπος επεξεργασίας των δεδομένων της κάμερας. Επίσης γίνεται αναφορά στην διαδικασία αναγνώρισης των χειρονομιών και στην πρόσβαση των αποτελεσμάτων. Τέλος θα αναλυθούν οι λειτουργίες της εφαρμογής και πως επιτεύχθηκαν.

3.1 Επιλογή Λειτουργικού Συστήματος

Το επίσημο λειτουργικό σύστημα του Raspberry Pi είναι το Raspbian, πλέον ονομάζεται Raspberry Pi OS. Η βάση του Raspberry Pi OS είναι το Debian ένα δωρεάν λειτουργικό σύστημα το οποίο χρησιμοποιεί το Linux kernel. Το Raspberry Pi OS είναι ειδικά σχεδιασμένο για τα τεχνικά χαρακτηριστικά που διαθέτουν τα Raspberry Pi, κυρίως την ARM αρχιτεκτονική της κεντρικής μονάδας επεξεργασίας. Παρόλο που το Pi OS είναι το επίσημο λειτουργικό σύστημα, υπάρχει πλήθος άλλων που μπορούν να λειτουργήσουν στο Raspberry Pi όπως το Ubuntu Mate, RiSC OS, DietPi, Lakka Linux και πολλά ακόμη με το καθένα να αποσκοπεί σε έναν διαφορετικό σκοπό. Το πρόβλημα που δημιουργείται όμως είναι ότι το Raspberry Pi 3 B+ δεν διαθέτει την ίδια υπολογιστική δύναμη με ένα σταθερό ή φορητό υπολογιστή εξαιτίας του μικρού μεγέθους με αποτέλεσμα το λειτουργικό σύστημα να δεσμεύει το μεγαλύτερο ποσοστό των πόρων για την λειτουργία του. Συνεπώς ένα πρόγραμμα το οποίο χρησιμοποιεί έναν αλγόριθμο μηχανικής όρασης δεν θα καταφέρει να λειτουργήσει ομαλά σε πραγματικό χρόνο και θα χρειαστεί ο συνδυασμός του Raspberry Pi με μια επιπλέον συσκευή όπως το Coral USB Accelerator της Google. Με την αξιοποίηση όμως του Lineage OS, το οποίο είναι ένα λειτουργικό σύστημα βασισμένο στο Android, απελευθερώνονται υπολογιστικοί πόροι και το πρόγραμμα μπορεί να λειτουργεί χωρίς προβλήματα σε πραγματικό χρόνο.

3.2 Πρόσβαση στην USB Κάμερα

Η εφαρμογή που αναπτύχθηκε χρησιμοποιεί κάμερα, μόνο που το Raspberry Pi 3 B+ δεν διαθέτει μια ενσωματωμένη, παρόλαυτα η σύνδεση εξωτερικής κάμερας μπορεί να γίνει με τρεις τρόπους. Η πρώτη επιλογή είναι το Raspberry Pi Camera Module v2 το οποίο συνδέεται με μέσω καλωδιωταινίας στην υποδοχή σειριακής διεπαφής κάμερας (CSI). Στην δεύτερη περίπτωση ανήκουν η εξωτερικές κάμερες usb και τέλος είναι οι ασύρματες κάμερες, η σύνδεση με τις οποίες πραγματοποιεί με το wifi. Στην εφαρμογή χρησιμοποιήθηκε η usb κάμερα. Για την δημιουργία της διεπαφής με την usb κάμερα αρχικά έπρεπε να γίνει ο έλεγχος υποστηριζόμενου επιπέδου των δυνατοτήτων της συσκευής. Απο τον έλεγχο προέκυψε ότι το Raspberry Pi 3 B+ υποστηρίζει περιορισμένες λειτουργίες κάμερας πρόσβαση στις οποίες μπορεί να γίνει μέσω του Camera 1 API.

Αρχικά στο στο AndroidManifest.xml αρχείο δηλώθηκε πως η εφαρμογή απαιτεί πρόσβαση στην κάμερα και στον εξωτερικό χώρο αποθήκευσης για εγγραφή και ανάγνωση δεδομένων. Το AndroidManifest.xml είναι αναγκαίο δομικό αρχείο καθώς προσφέρει σημαντικές πληροφορίες της εφαρμογής στο σύστημα Android για την λειτουργία της. Στη συνέχεια σε ξεχωριστό αρχείο Java δημιουργήθηκε η επιφάνεια στην οποία θα προβάλλεται η ζωντανή ροή της κάμερας και ο έλεγχος για τον αριθμό των συνδεδεμένων καμερών στην συσκευή. Οι usb κάμερες ανήκουν στην κατηγορία καμερών με προσανατολισμό φακού προς τα πίσω, το χαρακτηριστικό αυτό αποτέλεσε τον τρόπο επιλογής της σωστής κάμερας. Η ανάλυση μετάδοσης της εικόνας επιλέχθηκε στα 480p για την καλύτερη απόδοση της εφαρμογής. Επίσης πραγματοποιείται ο έλεγχος της περιστροφής της συσκευής για την ομαλή διατήρηση της αναλογίας απεικόνισης της εικόνας στο πλάτος και ύψος. Ο έλεγχος αυτός όμως δεν επηρεάζει το Raspberry Pi καθώς δεν διαθέτει ενσωματωμένο αισθητήρα για την μέτρηση περιστροφής, αλλά είναι απαραίτητος για άλλες android συσκευές όπως κινητά και ταμπλέτες.

3.3 Επεξεργασία Δεδομένων Κάμερες

Η κάμερα διαρκώς μεταδίδει ροή δεδομένων στην επιφάνεια προβολής της εφαρμογής τα οποία πρέπει κατάλληλα να επεξεργαστούν πριν την προώθηση τους στον αλγόριθμο μηχανικής όρασης. Για την πρόσβαση στην ροή των δεδομένων της κάμερας χρησιμοποιήθηκε η συνάρτηση `onPreviewFrame()` η οποία κάθε αντίγραφο καρτέ της κάμερας το αποθηκεύει σε έναν πίνακα bytes. Ο κάθε πίνακας των bytes αποτελεί μια εικόνα κωδικοποιημένης μορφής YUV. Το 'Y' αντιπροσωπεύει την φωτεινότητα που έχει το κάθε εικονοστοιχείο ενώ οι το 'U και V' περιέχουν τις τιμές των χρωμάτων των εικονοστοιχείων. Για τον αλγόριθμο όμως η κωδικοποίηση της εικόνας πρέπει να είναι της μορφής RGB. Αρχικά τα YUV bytes μετατρέπονται με την βοήθεια της συνάρτησης `cnvtYUVtoARGB8888()` σε μορφή ARGB8888 όπου η πληροφορία κάθε εικονοστοιχείου αποθηκεύεται σε 4 bytes, η κωδικοποίηση αυτή παρέχει την καλύτερη ποιότητας εικόνας. Το 'A' αντιπροσωπεύει την τιμή της διαφάνειας του εικονοστοιχείου. Στη συνέχεια μέσω της συνάρτησης `YUVtoRGB()` πραγματοποιείται η τελική μετατροπή των bytes σε μορφή RGB. Τέλος οι τρεις RGB τιμές κάθε εικονοστοιχείου αποθηκεύονται σε ένα πίνακα Bitmap διαστάσεων 640x480. Ουσιαστικά το Bitmap είναι μια δομή η οποία όταν επεξεργαστεί από το σύστημα θα παρέχει τις διαστάσεις της εικόνας και τις τιμές των εικονοστοιχείων της τα οποία θα πρέπει να προβληθούν στην οθόνη. Τέλος το Bitmap με τις RGB τιμές μετατρέπεται σε έναν πίνακα bytes πριν περάσει στον αλγόριθμο για τον εντοπισμό. Η μετατροπή αυτή είναι σημαντική καθώς έτσι οι υπολογισμοί πραγματοποιούνται πολύ πιο γρήγορα

3.4 Εκπαίδευση Αλγορίθμου

Για τον χειρισμό της εξωτερικής usb κάμερας ανέπαφα έπρεπε να χρησιμοποιηθεί ένα μοντέλο μηχανικής όρασης. Υπάρχουν πολλά μοντέλα ικανά να αντιμετωπίσουν το πρόβλημα του εντοπισμού των αντικειμένων το πρόβλημα όμως είναι πιο απο όλα μπορεί αποδοτικά να τρέξει σε μια μικρή συσκευή όπως το Raspberry Pi 3 B+ με περιορισμένη υπολογιστική ισχύς.

Στο προηγούμενο κεφάλαιο έγινε αναφορά στο Tensorflow Object Detection API το οποίο προσφέρει ένα σύνολο απο μοντέλα μηχανικής όρασης με το κάθε ένα να διαφέρει σε αρχιτεκτονική και στην μέθοδο με την οποία λύνει το πρόβλημα του εντοπισμού των αντικειμένων σε πραγματικό και μη χρόνο. Τα κριτήρια για την επιλογή του κατάλληλου αλγορίθμου ήταν ο χρόνος εντοπισμού και το ποσοστό ευστοχίας. Στην βιβλιοθήκη με τα μοντέλα του Tensorflow για το κάθε μοντέλο αναγράφεται ο χρόνος με τον οποίο πραγματοποιεί τον εντοπισμό αλλά και τον μέσο όρο ευστοχίας του εντοπισμού σε ειδικά διαμορφωμένα σετ δεδομένων όπως είναι το COCO, Kitti και Open Images. Ύστερα απο πειραματισμούς ο αλγόριθμος που επιλέχθηκε ήταν ο SSD-Mobilenet V2 άλλα η quantized έκδοση. Η συγκεκριμένη έκδοση δεν διαφέρει σε αρχιτεκτονική απο την κανονική, η μόνη διαφορά είναι ότι η υπολογισμοί στα στρώματα του δικτύου πραγματοποιούνται με ακέραιους αριθμούς και όχι δεκαδικούς κάτι που αυξάνει την ταχύτητα αλλά επηρεάζει ελαφρά αρνητικά την ευστοχία. Η quantized έκδοση του αλγορίθμου έχει μέσο όρο ευστοχίας στο COCO σετ δεδομένων 80 κατηγοριών αντικειμένων τις 22 μονάδες και την ταχύτητα εντοπισμού στα 29 χιλιοστά του δευτερολέπτου, χαρακτηριστικά αρκετά ικανοποιητικά για τον σκοπό της εργασίας.

Για να μπορέσει ο επιλεγμένος αλγόριθμος να εντοπίσει τις τρεις κατηγορίες χειρονομιών πρώτα πρέπει να εκπαιδευτεί πάνω σε αυτές, δηλαδή να κατασκευαστεί ένα σύνολο δεδομένων απο εικόνες των αντίστοιχων χειρονομιών. Οι κατηγορίες χειρονομιών είναι τρεις (Εικ. 19), η πρώτη κατηγορία είναι εικόνες της γροθιάς, η δεύτερη κατηγορία είναι εικόνες παλάμης με ενωμένα τα δάχτυλα και τέλος το σήμα της νίκης. Συνολικά ο αριθμός των εικόνων ανέρχεται στις 180, απο 60 φωτογραφίες για κάθε κατηγορία.



Εικ. 19) Απεικόνιση των τριών κατηγοριών χειρονομιών του σετ δεδομένων.

Για την διαδικασία της εκπαίδευσης δεν αρκούν μόνο οι εικόνες, κάθε εικόνα πρέπει να συνοδεύεται απο ένα XML αρχείο (Εικ. 20). Το XML περιέχει τις εξής σημαντικές πληροφορίες α) το όνομα του φακέλου στο οποίο βρίσκεται η εικόνα, b) την ονομασία της, c) την πλήρη θέση της στο σύστημα, d) τις διαστάσεις του υψους και του πλάτους της, e) αν είναι πολύχρωμη, f) την κατηγορία στην οποία ανήκει και g) τις διαστάσεις του ορθογώνιου πλαισίου που περιλαμβάνει το αντικείμενο για εντοπισμό.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

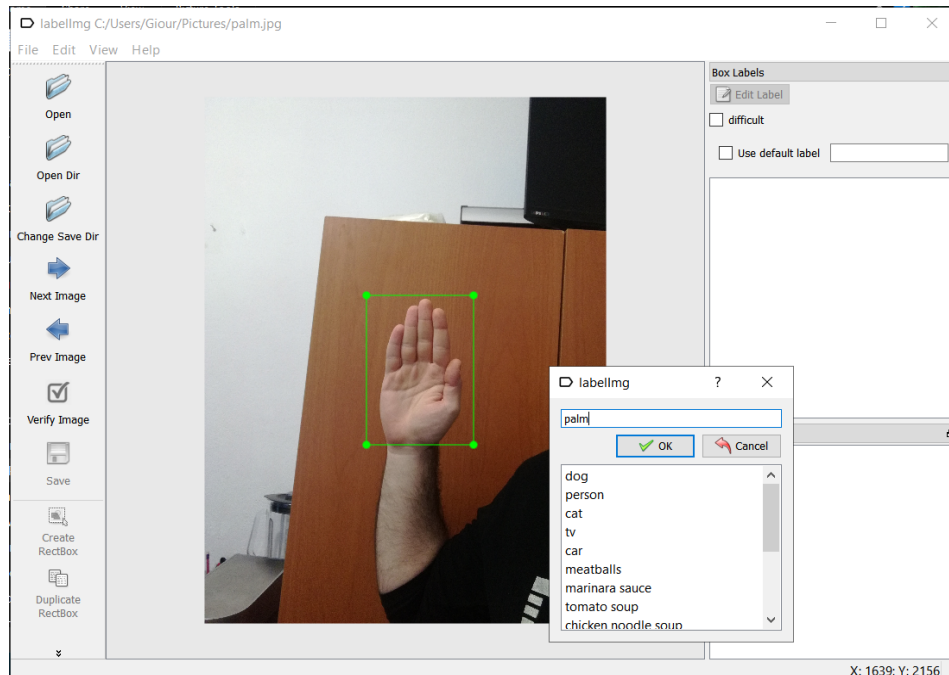
```

▼<annotation>
  <folder>file1</folder>
  <filename>IMG_20210202_160802_1.jpg</filename>
  <path>C:\Users\Giour\Desktop\file1\IMG_20210202_160802_1.jpg</path>
  ▼<source>
    <database>Unknown</database>
  </source>
  ▼<size>
    <width>2448</width>
    <height>3264</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  ▼<object>
    <name>fist</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    ▼<bndbox>
      <xmin>961</xmin>
      <ymin>1710</ymin>
      <xmax>1740</xmax>
      <ymax>2353</ymax>
    </bndbox>
  </object>
</annotation>

```

Εικ. 17) Απεικόνιση δομής του XML αρχείου.

Το XML αρχείο δημιουργήθηκε ξεχωριστά για κάθε εικόνα με το πρόγραμμα LabelImg. Στο κεντρικό παράθυρο του LabelImg φορτώνονται μια την φορά οι εικόνες από τον φάκελο με το όνομα της κατηγορίας του αντικειμένου, στη συνέχεια ο χρήστης με το ποντίκι δημιουργεί το ορθογώνιο πλαίσιο γύρω από το αντικείμενο της επιλογής του και αποθηκεύει το XML (Εικ. 21).



Εικ. 21) Απεικόνιση διαδικασίας δημιουργίας του XML.

Οι πληροφορίες του XML αρχείου είναι σημαντικές για την εκπαίδευση διότι αποτελούν σημείο αναφοράς για τον εύστοχο εντοπισμό του αντικειμένου.

Η εκπαίδευση έγινε με το Tensorflow Object Detection API και διήρκεσε οκτώ περίπου ώρες και τερματίστηκε ύστερα από 150000 βήματα σε φορητό υπολογιστή με 16GB ram και κάρτα γραφικών την Nvidia GTX 1060 6GB. Ακολούθησε η εξαγωγή του νέου μετεκπαιδευμένου μοντέλου και μετατροπή σε μορφή κατάλληλη για την λειτουργία σε Android περιβάλλον.

3.5 Διαδικασία Αναγνώρισης

Πλέον με τον αλγόριθμο μηχανικής όρασης εκπαιδευμένο στα δεδομένα χειρονομιών, έτοιμη την διεπαφή με την usb κάμερα και κατάλληλα προετοιμασμένων δεδομένων εικόνας ακολουθεί το βήμα αναγνώρισης των χειρονομιών σε πραγματικό χρόνο. Μετα την εκπαίδευση και εξαγωγή του μοντέλου προκύπτει το αρχείο detect.tflite το οποίο περιέχει τον αλγόριθμο, επιπλέον δημιουργήθηκε άλλο ένα αρχείο το label.txt με περιεχόμενο τις τρεις κατηγορίες χειρονομιών για αναγνώριση. Το πρώτο βήμα είναι η ανάγνωση του περιεχομένου των δύο αρχείων και φόρτωση τους στο σύστημα της εφαρμογής. Οι συνάντησης υπεύθυνες για το βήμα αυτό είναι η loadModelFile() και loadLablesList(), το μοντέλο φορτώνεται στην μεταβλητή τύπου Interpreter της Tensorflow Lite βιβλιοθήκης και πλέον είναι έτοιμο για χρήση. Το μοντέλο για την αναγνώριση δέχεται RGB εικόνες διαστάσεων 300x300x3 ο αριθμός 3 συμβολίζει τα κανάλια χρωμάτων 'R' για την τιμή του κόκκινου χρώματος, το 'G' για το πράσινο και 'B' για το μπλε , προηγουμένως έγινε αναφορά ότι η κάμερα παρέχει δεδομένα τα οποία ύστερα απο επεξεργασία αποθηκεύονται σε ένα Bitmap διαστάσεων 640x480 που ξεπερνούν τις ανάγκες του αλγορίθμου. Για την επίλυση του προβλήματος δημιουργήθηκε η συνάρτηση getResizedBitmap() η οποία δέχεται κάθε φορά το νέο Bitmap και το συρρικνώνει σε διαστάσεις 300x300, το Bitmap στη συνέχεια μετατρέπεται σε πίνακα bytes για την γρηγορότερη εκτέλεση των υπολογισμών απο τον αλγόριθμο με την συνάρτηση convertBitmapToByteBuffer(). Πλέον το μόνο που μένει είναι η δημιουργία της δομής που θα συγκροτήσει το αποτέλεσμα της αναγνώρισης. Η έξοδος του αλγορίθμου είναι η κατηγορία στην οποία ανήκει το αντικείμενο, το ποσοστό επιτυχίας της αναγνώρισης, η θέση του αντικειμένου και ο αριθμός των αντικειμένων. Για τον σκοπό της εργασίας ο αριθμός των αντικειμένων για αναγνώριση περιορίστηκε στη μια χειρονομία ανά είσοδο εικόνας. Η δομή στην οποία αποθηκεύονται τα αποτελέσματα είναι το HashMap το οποίο αποτελείται απο τέσσερις δυάδες κλειδιού και τιμής. Το πρώτο κλειδί αντιστοιχεί στις συντεταγμένες του αντικειμένου το δεύτερο στην κατηγορία, το τρίτο στο ποσοστό επιτυχίας αναγνώρισης και το τέταρτο στον αριθμό των αντικειμένων.

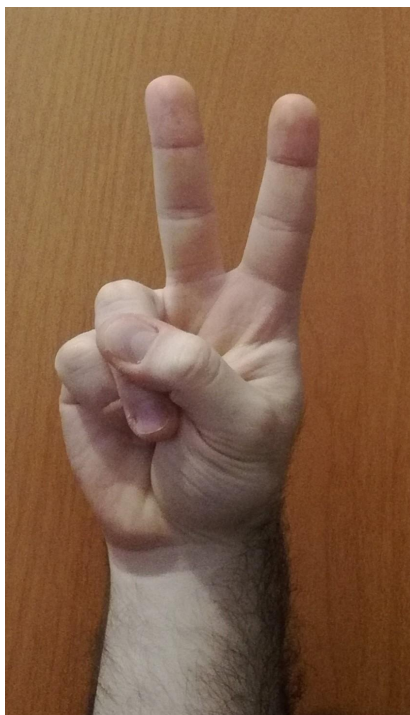
3.6 Πρόσβαση στα Αποτελέσματα

Κάθε καρέ της κάμερας περνάει απο ειδική επεξεργασία πριν την είσοδο του στον αλγόριθμο και το αποτέλεσμα αποθηκεύεται στην δομή HashMap, για την πρόσβαση στο πεδία της δομής δημιουργήθηκαν τέσσερις συναρτήσεις μια για κάθε πεδίο. Οι συναρτήσεις είναι η `getObjPosition()` τύπου `RectF` για τις συντεταγμένες του αντικειμένου, η `getObjlabel()` τύπου `String` για το όνομα κατηγορίας, η `getObjConf()` τύπου `Float` για το ποσοστό επιτυχίας αναγνώρισης και η `getNumDetections()` τύπου `Float` για τον συνολικό αριθμό των εντοπισμένων αντικειμένων. Για την μέγιστη απόδοση του συστήματος τέθηκε όριο προβολής της εντοπισμένης κατηγορίας χειρονομίας με βάση το ποσοστό του βαθμού ευστοχίας εντοπισμού. Μετά από δοκιμές το όριο με την καλύτερη απόδοση επιλέχθηκε στα 90% με τον αλγόριθμο πρακτικά να μην εμφανίζει λάθος κατηγορία χειρονομίας. Για την βέλτιστη λειτουργία της εφαρμογής η διαδικασία εντοπισμού και η πρόσβαση στα αποτελέσματα του εντοπισμού πραγματοποιούνται σε ξεχωριστό background thread. Η χρήση των threads επιτρέπει στην εφαρμογή να μοιράσει τις λειτουργίες της στους πόρους που διαθέτει αλλά και να εκτελεί τις ενέργειές της παράλληλα. Έτσι δεν επιβαρύνεται το κυρίως thread το οποίο εκτελεί λειτουργίες όπως αποθήκευση, προβολή και περιήγηση στον φάκελο των ληφθέντων φωτογραφιών με αποτέλεσμα η ανταπόκριση της εφαρμογής να αποδίδει στο μέγιστο ακόμα και σε μια συσκευή περιορισμένης ισχύος όπως το Raspberry Pi 3 B+.

3.7 Λειτουργίες Εφαρμογής

Εφαρμογή που αναπτύχθηκε επιτρέπει στον χρήστη τον χειρισμό της κάμερας ανέπαφα με την βοήθεια αλγόριθμου τεχνητής νοημοσύνης. Προς το παρόν οι λειτουργίες της εφαρμογής είναι οι εξής: a) η λήψη φωτογραφιών, b) η αποθήκευση τους σε έναν φάκελο στην εξωτερική μνήμη της συσκευής, c) η προβολή των ληφθέντων φωτογραφιών και d) η περιήγηση στον φάκελο με τις φωτογραφίες η την διακοπή προβολής.

Με την εγκατάσταση της εφαρμογής στην συσκευή και την πρώτη εκκίνηση ζητείται απο τον χρήστη οι άδειες πρόσβασης στην κάμερα και στον χώρο αποθήκευσης. Με το δοθεί η πρόσβαση η εφαρμογη δημιουργει εναν φακελο στον οποιο θα αποθηκευονται οι φωτογραφίες που θα επιλέξει να τραβήξει με την κάμερα ο χρήστης. Η συνάρτηση υπεύθυνη για την δημιουργία του φακέλου είναι η `createDirectoryPath()`, σε περίπτωση που αποτύχει η δημιουργία του φακέλου ο χρήστης ενημερώνεται με κατάλληλο μήνυμα και τον τερματισμό της εφαρμογής. Εφόσον δημιουργηθεί ο φάκελος χωρίς κάποιο πρόβλημα ο χρήστης είναι ελεύθερος να τραβαι φωτογραφίες με το να σχηματίσει με το χέρι του μπροστά απο την κάμερα την χειρονομία της νίκης (Εικ. 22).



Εικ. 22) Η χειρονομία που σηματοδοτεί την λήψη φωτογραφίας.

Σε περίπτωση που ο χρήστης θέλει να δει την φωτογραφία που έβγαλε αρκεί να δείξει την παλάμη του με κλειστά τα δάχτυλα (Εικ. 23). Με την χειρονομία αυτή εμφανίζεται στην οθόνη η φωτογραφία που έβγαλε. Για την λειτουργία αυτή είναι υπεύθυνες οι συναρτήσεις `accessImg()`, `loadImages()`, `openImages()` και `displayImages()`. Η `accessImg()` απενεργοποιεί την επιφάνεια προβολής σε πραγματικό χρόνο της ροής της κάμερας και καλεί την `openImages()` η οποία ρυθμίζει ανάλογα αν πρέπει να προβάλει η όχι την φωτογραφία μέσω της `displayImages()` που στόχο έχει την φόρτωση του `Bitmap` της φωτογραφίας στην επιφάνεια προβολής. Τέλος η `loadImages()` φορτώνει απο την μνήμη της εικόνες σε μια λίστα ώστε να είναι έτοιμες για προβολή.



Εικ 23) Η χειρονομία υπεύθυνη για την προβολή της πρώτης φωτογραφίας

Τέλος αν χρήστης θέλει να περιηγηθεί στον φάκελο με τις ληφθέντες φωτογραφίες αρκεί να δείξει στην κάμερα την χειρονομία της γροθιάς (Εικ. 24) και να την μετακίνηση προς τα αριστερά είτε τα δεξιά για να περάσει στην επόμενη ή την προηγούμενη εικόνα αντίστοιχα . Με τον εντοπισμό της γροθιάς σε έναν πίνακα έξι θέσεων αποθηκεύεται οι συντεταγμένες της θέσης της γροθιάς. Αν ο χρήστης μετακινεί το χέρι του απο τα δεξιά προς τα αριστερά οι συντεταγμένες στον πίνακα αποθηκεύονται σε αύξουσα σειρά με αποτέλεσμα η αφαίρεση της αρχικής θέσης απο την τελική να έχει αρνητικο αποτέλεσμα έτσι δίνεται η εντολή της προβολής της επόμενης εικόνας. Αν η γροθιά μετακινείται απο δεξιά προς τα αριστερά η αφαίρεση έχει θετικό αποτέλεσμα και προβάλλεται η προηγούμενη εικόνα. Σε περίπτωση που ο χρήστης δεν επιθυμεί άλλο να βλέπει τις φωτογραφίες και θελει να τραβήξει καινούργιες μπορεί να κλείσει την προβολή με την χειρονομία της παλάμης οπως δηλαδή και την άνοιξε.



Εικ. 24) Η χειρονομία της γροθιας υπεύθυνη για την μετάβαση είτε στην επόμενη είτε στην προηγούμενη εικόνα.

Κώδικας Εφαρμογής

Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται ο κώδικας της εφαρμογής. Η εφαρμογή αναπτύχθηκε στο περιβάλλον ανάπτυξης εφαρμογών Android Studio για συσκευές που λειτουργούν αξιοποιώντας τις δυνατότητες που προσφέρει το λειτουργικό σύστημα Android.

4.1 AndoridManifest.XML

Το xml αρχείο περιγράφει σημαντικές πληροφορίες της εφαρμογής στο λειτουργικό σύστημα και στο εργαλείο ανάπτυξης Android Studio. Μεταξύ άλλων περιέχει επίσης τις άδειες που χρειάζεται η εφαρμογή για να λειτουργήσει, όπως πρόσβαση σε αρχεία, στο υλικό τμήμα της συσκευής όπως η κάμερα και το μικρόφωνο ή κάποιου άλλου είδους αισθητήρα. Επιπλέον στο xml φαίνονται η προϋποθέσεις υλικού και λογισμικού της εφαρμογής για να μπορέσει να λειτουργήσει στην συσκευή στην οποία γίνεται η εγκατάσταση της. Ακόμη δηλώνονται τα τμήματα από τα οποία αποτελείται η εφαρμογή όπως activities και services. Το κάθε αρχείο έχει όνομα και καταληξη για την αναγνώριση της ιδιότητας του και της γλώσσας προγραμματισμού που χρησιμοποιήθηκε για την δημιουργία του. Τέλος δηλώνεται το όνομα του πακέτου της εφαρμογής ώστε κατά την δοκιμή του πρότζεκτ το Android Studio να γνωρίζει πια αρχεία και φακέλους πρέπει να λάβει υπόψη του για να γίνει σωστά η κατασκευή. Επίσης κατά την εξαγωγή της εφαρμογής το όνομα του πακέτου αντικαθιστάτε με ένα ειδικό αναγνωριστικό ID για την διασφάλιση της ασφάλειας.

Περιεχόμενο αρχείου AndroidManifest.xml

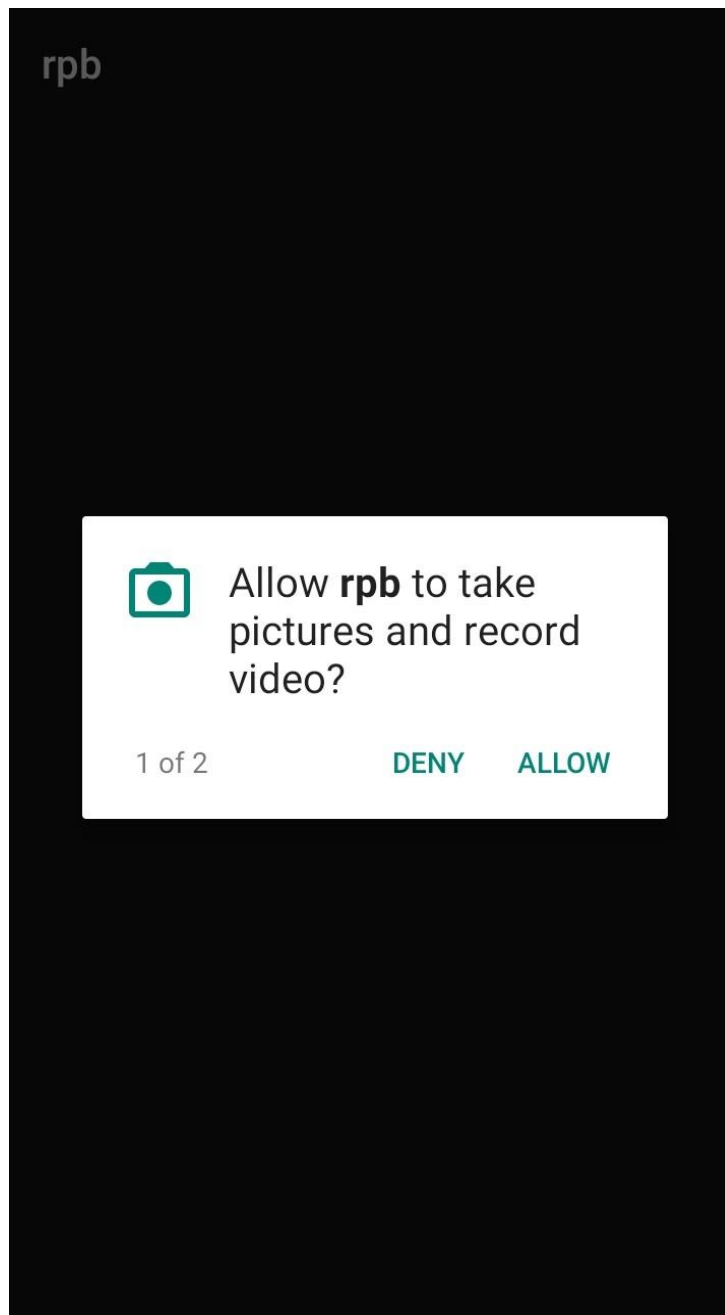
```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.rpb">

    <uses-permission android:name = "android.permission.CAMERA"/>
    <uses-feature android:name = "android.hardware.camera" />
    <uses-permission android:name =
"android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name =
"android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Rpb">
        <activity android:name=".CameraActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

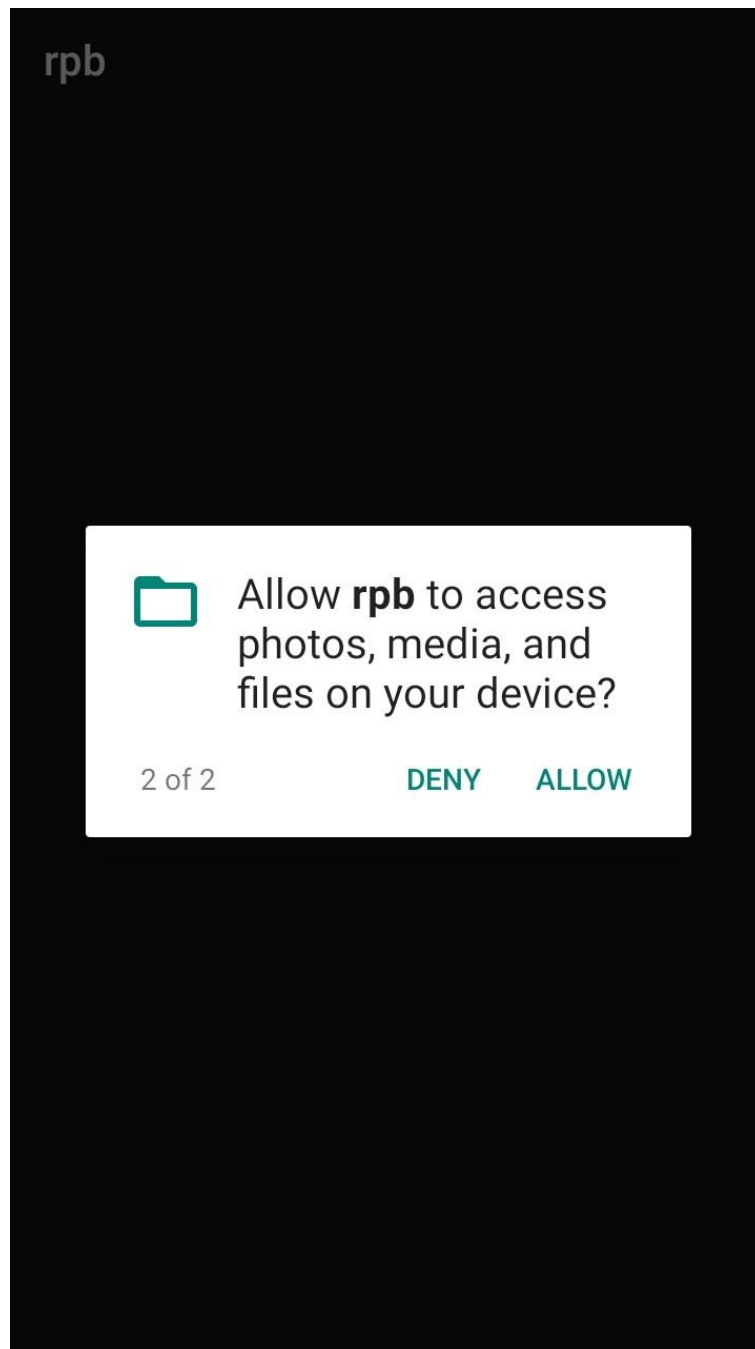
4.2 MainActivity.Java

Το αρχείο MainActivity.Java αποταίλει την πρώτη δραστηριότητα της εφαρμογής. Όταν ο χρήστης ανοίγει για πρώτη φορά την εφαρμογή του εμφανίζονται δύο μηνύματα διαδοχικά, το πρώτο ζητάει από τον χρήστη άδεια πρόσβασης για την λήψη φωτογραφιών και βίντεο (Εικ. 25).



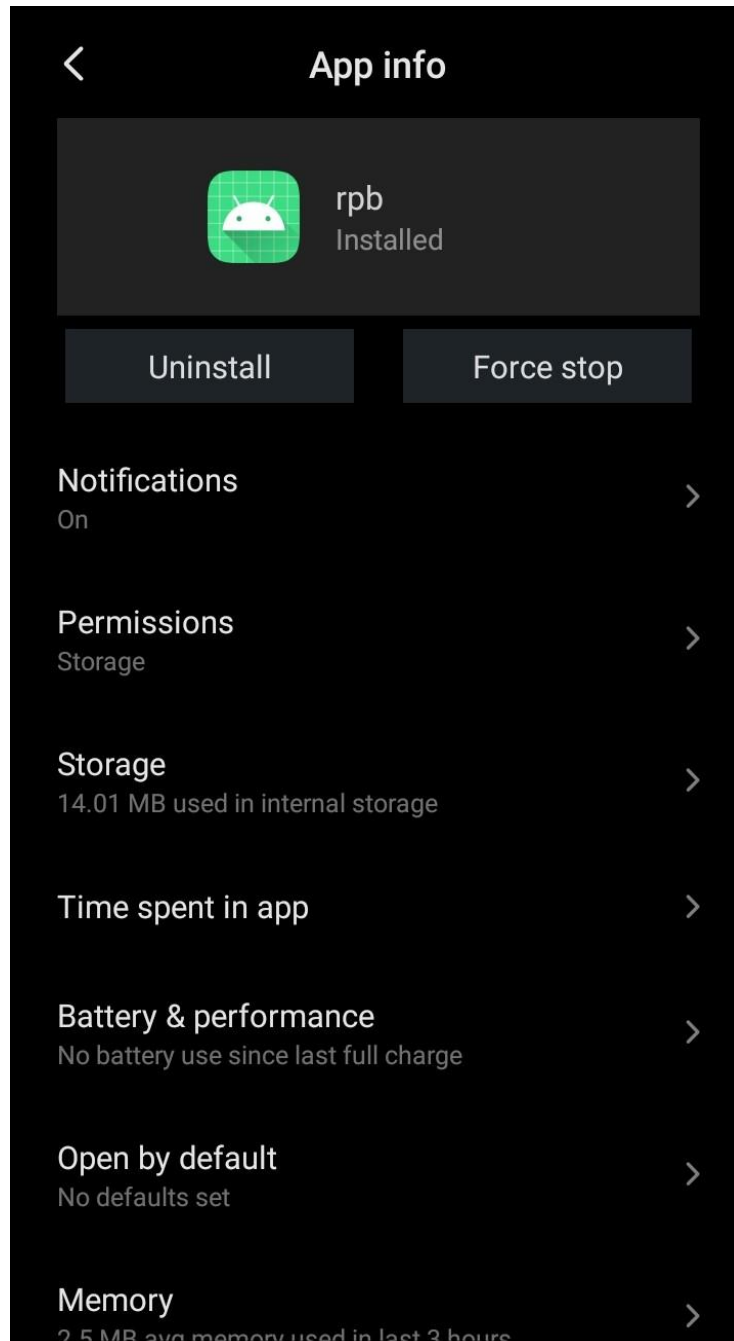
Εικ. 25) Ερώτηση για την αποδοχής άδειας λήψης φωτογραφιών και βίντεο.

Το επόμενο μήνυμα ζητάει από τον χρήστη την παροχή άδειας στην εφαρμογή για την πρόσβαση στα αρχεία της συσκευής δηλαδή τον αποθηκευτικό χώρο (Εικ. 26).



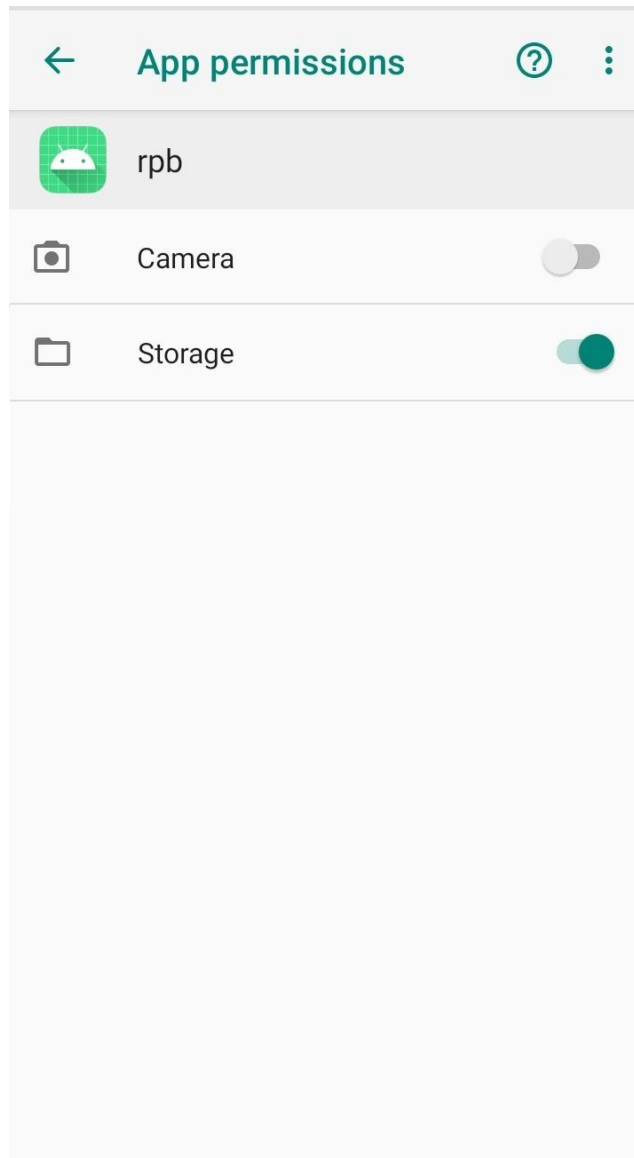
Εικ. 26) Ερώτηση για την αποδοχής άδειας για την πρόσβαση στον αποθηκευτικό χώρο.

Στην περίπτωση που ο χρήστης αρνηθεί την παροχή άδειας είτε στο πρώτο μήνυμα είτε στο δεύτερο ή αν κατά λάθος πατήσει το κουμπί άρνησης η εφαρμογή ελαχιστοποιείται και ο χρήστης μεταφέρεται στις ρυθμίσεις (Εικ. 27) όπου μπορεί επιλέγοντας το μενού με τις άδειες της εφαρμογής να της ενεργοποιήσει.



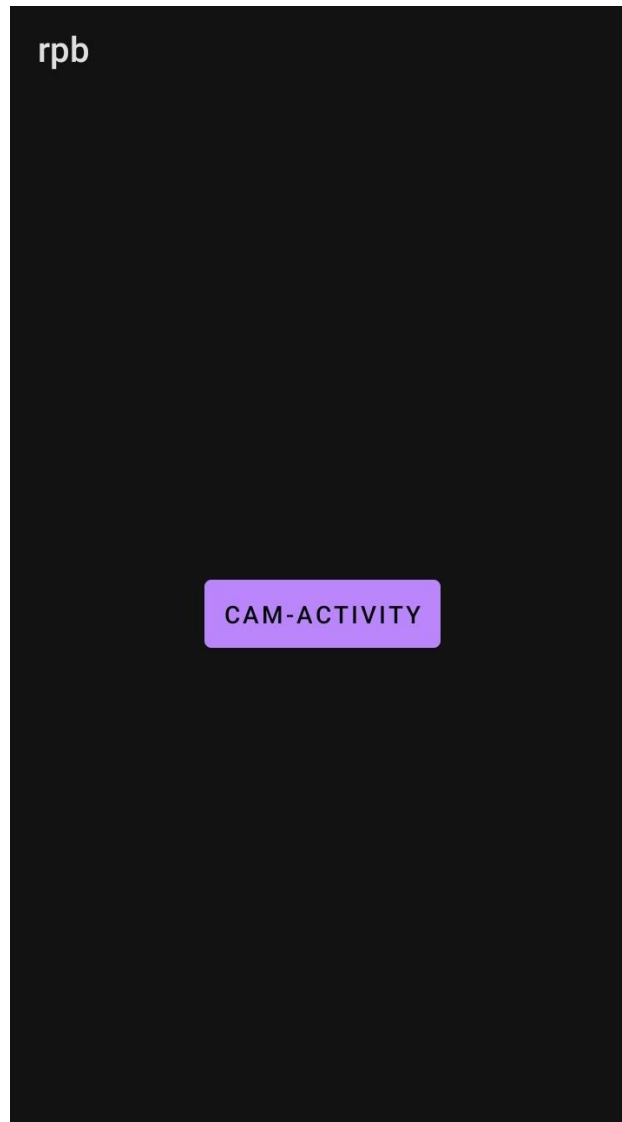
Εικ. 27) Ρυθμίσεις εφαρμογής.

Για να μεταβεί στις άδειες πρέπει να πιάσει την επιλογή που ονομάζεται Permissions (Εικ. 28). Εκεί εμφανίζεται ένα νέο μενού στο οποίο φαίνονται οι απαραίτητες άδειες για την λειτουργία της εφαρμογής. Στη συνέχεια μπορεί να επιλέξει την αποδοχή τους και να μεταβεί πίσω στην εφαρμογή.



Εικ. 28) Μενού με τις άδειες της εφαρμογής.

Με την αποδοχή των αδειών πλέον ο χρήστης μπορεί ελεύθερα να χρησιμοποιήσει την εφαρμογή. Με την είσοδο του στην εφαρμογή εμφανίζεται ένα κουμπί στη μέση της οθόνης το οποίο ονομάζεται CAM-ACTIVITY (Εικ. 29). Αν το επιλέξει ξεκινάει η προβολή ζωντανής ροής από την κάμερα (Εικ. 30).



Εικ. 29) Αρχική οθόνη της εφαρμογής.



Εικ. 30) Προβολή της ζωντανής ροής της κάμερας.

Σε περίπτωση που ο χρήστης δεν επιθυμεί άλλο να χρησιμοποιήσει την εφαρμογή μπορεί απλά να την κλείσει, δεν απαιτούνται κάποιες έξτρα ενέργειες. Επίσης αν θέλει να διαγράψει τις εικόνες που τράβηξε θα πρέπει να μεταβεί στον φακελο Συλλογη/Gallery της συσκευής.

Ο κώδικας υπεύθυνος για την συμπεριφορά της MainActivity.Java παρουσιάζεται παρακάτω.

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG ="MSG:";
    private final int CODE = 1010;
    private final String[] RequiredPermissions = new
        String[]{Manifest.permission.CAMERA,
Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.WRITE_EXTERNAL_STORAGE};
    private Button cameraActivity;
```

Αρχικοποίηση της MainActivity.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    cameraActivity = findViewById(R.id.openCameraActivity);

    if(allPermissionsGranted()){
        Log.d("TAG", "Permissions Granted");
    } else {

        ActivityCompat.requestPermissions(this,RequiredPermissions,
                                           CODE);

    }

    cameraActivity.setOnClickListener(v -> {
        Intent cameraIntent = new Intent(this, CameraActivity.class);
        startActivity(cameraIntent);
    });

}
```

Έλεγχος αδειών που αποδέχτηκε ο χρήστης.

```
private boolean allPermissionsGranted(){
    for(String permissions : RequiredPermissions){
        if(ContextCompat.checkSelfPermission(this,permissions)
!=
        PackageManager.PERMISSION_GRANTED){
            return false;
        }
    }
    return true;
}
```

Αποδοχή αποτελεσμάτων παροχής αδειών. Σε περίπτωση άρνησης μεταφορά στις ρυθμίσεις εφαρμογής.

```

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull
    String[] permissions, @NonNull int[] grantResults) {
    if (requestCode == CODE) {
        if (allPermissionsGranted()) {
            Toast.makeText(this, "Access granted.",
                Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(this, "Permissions Denied.",
                Toast.LENGTH_LONG).show();

            Intent permIntent = new Intent();

            permIntent.setAction(Settings.ACTION_APPLICATION_DETAILS
            —
                SETTINGS);
            Uri uri = Uri.fromParts("package",
                getPackageName(), null);
            permIntent.setData(uri);
            startActivity(permIntent);
        }
    }
}

```

4.3 Επιλογή και Ενεργοποίηση Κάμερας

Επιλογή κάμερας συσκευής με βάση τον προσανατολισμό του φακού.

```
private static int getCamera() {
    Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
    for (int i = 0; i < Camera.getNumberOfCameras(); i++) {
        Camera.getCameraInfo(i, cameraInfo);
        if (cameraInfo.facing ==
            Camera.CameraInfo.CAMERA_FACING_BACK)
            return i;
    }
    return -1;
}
```

Ενεργοποίηση της κάμερας για την έναρξη προβολής διαστάσεων 640*480.

```
private void activateCamera() {
    mCamera = Camera.open(getCamera());
    liteDetector = new ObjectDetection(this);
    try {
        mCamera.setPreviewTexture(textureView.getSurfaceTexture());
        Camera.Parameters params = mCamera.getParameters();
        params.setPreviewSize(640, 480);
        int rotation =
getWindowManager().getDefaultDisplay().getRotation();
        mCamera.setDisplayOrientation(ORIENTATIONS.get(rotation));
        int orientation =
getResources().getConfiguration().orientation;
        int width = 640;
        int height = 480;
        if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
            textureView.setAspectRatio(width, height);
        } else {
            textureView.setAspectRatio(height, width);
        }
        mCamera.setParameters(params);
    } catch (IOException ioe) {
        mCamera.release();
    }
    mCamera.setPreviewCallbackWithBuffer(this);
    mCamera.addCallbackBuffer(new byte[461361]);
    mCamera.startPreview();
}
```

Για την δημιουργία επιφάνειας προβολής χρησιμοποιήθηκε το Interface `TextureView.SurfaceTextureListener`.

```
public class CameraActivity extends AppCompatActivity implements
TextureView.SurfaceTextureListener, Camera.PreviewCallback {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.camera_activity);
        linearLayout = findViewById(R.id.lr);
        textureView = new AutoFitTextureView(this);
        textureView.setSurfaceTextureListener(this);
        linearLayout.addView(textureView);

        imageView = findViewById(R.id.iv);
        imageView.setVisibility(View.INVISIBLE);

    }

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture surface, int
        width, int height) {
        activateCamera();
    }

    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int
        width, int height) {}

    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
        return true;
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surface) {}
}
```

Σε περίπτωση που για κάποιο λόγο διακοπή η λειτουργία της CameraActivity.Java πραγματοποιείται έλεγχος αν είναι διαθέσιμη ακόμα η επιφάνεια προβολής της ζωντανής ροής τότε συνεχίζεται η μετάδοση. Εναλλακτικά απελευθερώνεται η χρήση της κάμερας και της επιφάνειας προβολής.

```
@Override
public void onResume() {
    super.onResume();
    if (textureView.isAvailable()) {
        activateCamera();
    } else {
        textureView.setSurfaceTextureListener(this);
    }
}
```

```
@Override
public void onPause() {
    super.onPause();
    if (mCamera != null) {
        mCamera.setPreviewCallbackWithBuffer(null);
        try {
            new Thread(runnable);
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        mCamera.stopPreview();
        mCamera.release();
        mCamera = null;
    }
}
```

4.4 Συλλογή και Μετάδοση των Καρέ Κάμερας

Το Interface Camera.PreviewCallback επιτρέπει την χρήση της συνάρτησης onPreviewFrame() η οποία δέχεται αντίγραφο σε μορφή bytes των καρέ που μεταδίδει η κάμερα.

Για την επεξεργασία των καρε δημιουργήθηκε ένα background thread. Σκοπός του είναι να ελαφρύνει το φόρτο εργασίας του κυρίου thread εκτελώντας εργασίες παράλληλα για μια ομαλή λειτουργία της εφαρμογής.

```
@Override
public void onPreviewFrame(byte[] data, Camera camera) {
    runnable = new BackgroundRunnable(data);
    new Thread(runnable).start();
}
```

Η συνάρτηση getBitmap() με τις συνάρτησης cnvrtYUVToARGB8888() και YUVtoRGB() μετατρέπουν τα bytes κάθε καρέ σε bitmap όπου η τιμές των εικονοστοιχείων είναι σε μορφή RGB.

```
private Bitmap getBitmap(byte[] data) {
    Bitmap nb = Bitmap.createBitmap(640, 480,
                                   Bitmap.Config.ARGB_8888);
    cnvrtYUVToARGB8888(data, 640, 480, rgbValues);
    nb.setPixels(rgbValues, 0, 640, 0, 0, 640, 480);
    return nb;
}
```

Αρχικά τα καρε που μεταδίδει η καμερα ειναι της μορφης YUV και πρεπει να μετατραπούν σε RGB για την είσοδο τους στον αλγόριθμο εντοπισμού αντικειμένων.

```
public static void cnvrtYUVToARGB8888(byte[] input, int width, int
height, int[] output) {
    final int frameSize = width * height;
    for (int j = 0, yp = 0; j < height; j++) {
        int uvp = frameSize + (j >> 1) * width;
        int u = 0;
        int v = 0;

        for (int i = 0; i < width; i++, yp++) {
            int y = 0xff & input[yp];
            if ((i & 1) == 0) {
                v = 0xff & input[uvp++];
                u = 0xff & input[uvp++];
            }
            output[yp] = YUVtoRGB(y, u, v);
        }
    }
}
```

```
}  
}
```

61

Οι συναρτήσεις `cnvrtYUVToARGB8888()` και `YUVtoRGB()` ανήκουν στην ομάδα TensorFlow

```
/*  
 * Copyright 2019 The TensorFlow Authors. All Rights Reserved.  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * you may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 * http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
private static int YUVtoRGB(int y, int u, int v) {  
    y = Math.max((y - 16), 0);  
    u -= 128;  
    v -= 128;  
  
    int y1192 = 1192 * y;  
    int r = (y1192 + 1634 * v);  
    int g = (y1192 - 833 * v - 400 * u);  
    int b = (y1192 + 2066 * u);  
  
    r = r > 262143 ? 262143 : (Math.max(r, 0));  
    g = g > 262143 ? 262143 : (Math.max(g, 0));  
    b = b > 262143 ? 262143 : (Math.max(b, 0));  
  
    return 0xff000000 | ((r << 6) & 0xff0000) | ((g >> 2) & 0xff00)  
    |  
    ((b >> 10) & 0xff);  
}
```

4.5 Δημιουργία Φακέλου Αποθήκευσης

Για την αποθήκευση των ληφθέντων φωτογραφιών έπρεπε να δημιουργηθεί φάκελος στο σύστημα απο την εφαρμογή.

```
private String createDirectoryPath() {
    String absPath;
    File sdCardPath = Environment.getExternalStorageDirectory();
    File myDir = new File(sdCardPath.getAbsolutePath() + "/3DDD");
    absPath = myDir.toString();
    if(!myDir.exists() && !myDir.mkdir()) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(getApplicationContext(), "Oops
                    something went wrong.",
                    Toast.LENGTH_SHORT).show();
                finishAndRemoveTask();
                System.exit(0);
            }
        });
    }

    return absPath;
}
```

Η διαδικασία δημιουργίας πραγματοποιείται κατά την πρώτη εκκίνηση της εφαρμογής και στη συνέχεια απλά γίνεται έλεγχος αν εξακολουθεί να υπάρχει ο φάκελος. Αν υπάρχει ο φάκελος η εφαρμογή συνεχίζει την λειτουργία της ομαλά, αν δεν υπάρχει θα δημιουργηθεί, αν κάτι πήγε λάθος κατά την δημιουργία του η εφαρμογή κλείνει.

4.6 Αποθήκευση Φωτογραφίας

Με τον εντοπισμό της χειρονομίας νίκης πραγματοποιείται η λήψη φωτογραφίας. Η ονομασία της κάθε φωτογραφίας εξαρτάται από την ημερομηνία και την ώρα λήψης.

```
private void savePic(Bitmap bitmap, byte[] data){

    int rotation = getWindowManager().getDefaultDisplay()
        .getRotation();
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd
        HH:mm:ss", Locale.getDefault());
    File imageTaken = new File(createDirectoryPath(),
        formatter.format(new Date()) + ".jpg");

    try {
        FileOutputStream fos = new FileOutputStream(imageTaken);
        Matrix matrix = new Matrix();
        if (rotation == Configuration.ORIENTATION_PORTRAIT) {
            matrix.postRotate(0);
            Bitmap b1 = Bitmap.createBitmap(bitmap, 0, 0,
                bitmap.getWidth(),
                bitmap.getHeight(),
                matrix, false);
            b1.compress(Bitmap.CompressFormat.PNG, 99, fos);
        } else {
            matrix.postRotate(90);
            Bitmap b1 = Bitmap.createBitmap(bitmap, 0, 0,
                bitmap.getWidth(),
                bitmap.getHeight(),
                matrix, false);
            b1.compress(Bitmap.CompressFormat.PNG, 99, fos);
        }
        fos.write(data);
        fos.flush();
        fos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

4.7 Διεργασίες του Background Thread

Στο εσωτερικό του thread πραγματοποιείται η διαδικασία αναγνώρισης αντικειμένων και επιστρέφονται οι τιμές όπως η κατηγορία, το ποσοστό ακρίβειας και μια από τις τέσσερις συντεταγμένες θέσεις του αντικειμένου εντοπισμένου.

```
liteDetector.detectObj(bitm);
int s = Math.round(liteDetector.getObjConf());
String lbl = liteDetector.getObjLabel();
```

4.7.1 Λήψη Φωτογραφίας

Αν η ετικέτα της χειρονομίας αντιστοιχεί στην κατηγορία “peace” τότε πραγματοποιείται η λήψη της φωτογραφίας. Αρχικά εμφανίζεται ένα μήνυμα το οποίο προειδοποιεί για την έναρξη λήψης, αποθηκεύεται η φωτογραφία και στην εμφανίζεται το δεύτερο μήνυμα το οποίο ενημερώνει για την ολοκλήρωση διαδικασίας.

```
if(lbl.equals("peace")) {
    if(p) {
        try {
            Thread.sleep(1000);
            CameraActivity.this.runOnUiThread(new Runnable() {
                public void run() {
                    Toast.makeText(getApplicationContext(),
                        "Taking Picture...",
                        Toast.LENGTH_SHORT).show(); }
            });
            savePic(bitm, data);
            CameraActivity.this.runOnUiThread(new Runnable() {
                public void run() {
                    Toast.makeText(getApplicationContext(),
                        "Image Saved",
                        Toast.LENGTH_SHORT).show(); }
            });
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

4.7.2 Προβολή Φωτογραφιών

Αν η ετικέτα της χειρονομίας αντιστοιχεί στην κατηγορία “palm” τότε πραγματοποιείται η προβολή της πρώτης φωτογραφίας.

```
if (lbl.equals("palm")) {
    accessImg();
    counter = 0;
}
```

Η μεταβλητή counter διατηρεί την θέση της εικόνας, μέσω της accessImg() ελέγχεται η κατάσταση ορατότητας της επιφάνειας προβολής. Αν ο χρήστης είναι στην προβολή φωτογραφιών και επιθυμεί να την κλείσει ώστε να μεταβεί πίσω στην ζωντανή ροή αρκεί να δείξει ξανά την χειρονομία της παλάμης. Τότε η προβολή τίθεται σε κατάσταση αόρατη (*INVISIBLE*) και η προβολή σε κατάσταση ορατότητας (*VISIBLE*).

```
private void accessImg() {
    state = ! state;
    try {
        if (state) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    imageView.setVisibility(View.INVISIBLE);
                    linearLayout.setVisibility(View.VISIBLE);
                    p = true;
                }
            });
        } else {
            openImages();
            p=false;
        }
        Thread.sleep(3000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

```
private void displayImages(String imgString) {
    Bitmap imgBitmap = BitmapFactory.decodeFile(imgString);
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
```

```

        imageView.setImageBitmap(imgBitmap);
    }
});
}

```

66

Οι φωτογραφίες απο τον φάκελο φορτώνονται σε μια λίστα με την συναρτηση loadImages().

```

private ArrayList<String> loadImages() {
    ArrayList<String> imgL = new ArrayList<>();
    File imgFolderPath = new File("/storage/emulated/0/3DDD");
    if (imgFolderPath.exists()) {
        File[] imageFiles = imgFolderPath.listFiles();
        if (imageFiles != null && imageFiles.length != 0) {
            for (File element : imageFiles) {
                imgL.add(element.toString());
            }
        }
    }
    return imgL;
}

```

Αν η επιφάνεια προβολής είναι ορατή τότε απενεργοποιείται και εμφανίζεται η πρώτη φωτογραφία απο την openImages() και displayImages().

```

private void openImages() {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            imageView.setVisibility(View.VISIBLE);
            linearLayout.setVisibility(View.INVISIBLE);
        }
    });
    imageList = loadImages();
    if (imageList.size() == 0) {
        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(),
                    "No Pictures",
                    Toast.LENGTH_SHORT).show();
            }
        });
    } else {
        displayImages(imageList.get(0));
    }
}

```

4.7.3 Περιήγηση Φωτογραφιών

Αν η ετικέτα της χειρονομίας αντιστοιχεί στην κατηγορία “fist” τότε ο χρήστης μπορεί να περιηγείται στο φάκελο με τις φωτογραφίες. Με βάση την διαφορά της αρχικής θέσης απο την τελική των συντεταγμενων της χειρονομίας πραγματοποιείται η μετάβαση είτε στην επόμενη είτε στην προηγούμενη εικόνα.

```
if(lbl.equals("fist")) {
    if (Pos < 6) {
        leftCoordinates[Pos] = (int)
            liteDetector.getObjPosition().left;
        Pos++;
    } else if (Pos == 6) {
        if (leftCoordinates[0] - leftCoordinates[5] > 0 ) {
            counter--;
            if(counter >= 0){
                displayImages(imageList.get(counter));
            } else {
                counter=0;
            }
            Pos = 0;
            Arrays.fill(leftCoordinates, 0);
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } else if (leftCoordinates[0] - leftCoordinates[5] < -1 ) {
            counter++;
            if(counter < imageList.size()){
                displayImages(imageList.get(counter));
            }
            Pos = 0;
            Arrays.fill(leftCoordinates, 0);
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
}
```

4.8 Αναγνώριση Χειρονομίας

Τα επεξεργασμένα καρέ πλέον σε μορφή bitmap στέλνονται στην συνάρτηση `detectObj()` Φορτώνεται στην μεταβλητή `liteDetector` το μοντέλο αναγνώρισης και στο `labelsList` οι ετικέτες των κατηγοριών. Αρχικοποιούνται οι πίνακες στους οποίους θα αποθηκευτούν τα αποτελέσματα αναγνώρισης. Μόνο τα αποτελέσματα με ποσοστό ακρίβειας μεγαλύτερου ίσου του 90% γίνονται δεκτά.

```
void detectObj(Bitmap bitmap) {

    arrayOfBytes = ByteBuffer.allocateDirect(imgX * imgY * 3);
    arrayOfBytes.order(ByteOrder.nativeOrder());
    pix = new int[imgX * imgY];

    try {
        liteDetector = new Interpreter(loadModelFile(), null);
        labelsList = loadLabelsList();
    } catch (Exception ex) {
        ex.printStackTrace();
        finishAndRemoveTask();
        System.exit(0);
    }

    Bitmap resizedBitmap = getResizedBitmap(bitmap);
    convertBitmapToByteBuffer(resizedBitmap);

    float[][] outputClasses = new float[1][NUM_DETECTIONS];
    float[][] outputScores = new float[1][NUM_DETECTIONS];
    float[][][] outputLocations = new float[1][NUM_DETECTIONS][4];
    float[] numDetections = new float[1];

    Object[] inputArray = {arrayOfBytes};
    Map<Integer, Object> outputMap = new HashMap<>();
    outputMap.put(0, outputLocations);
    outputMap.put(1, outputClasses);
    outputMap.put(2, outputScores);
    outputMap.put(3, numDetections);

    liteDetector.runForMultipleInputsOutputs(inputArray, outputMap);
```

```

objConf = outputScores[0][0]*100;
if (Math.round(objConf) > 89){
    objLabel = labelsList.get((int) outputClasses[0][0]);
    objPos = new RectF(outputLocations[0][0][1] * imgX,
        outputLocations[0][0][0] * imgY,
        outputLocations[0][0][3] * imgX,
        outputLocations[0][0][2] * imgY);
    numObjDetected = numDetections[0];
} else {
    objLabel = "None";
}
}

```

Επειδή ο αλγόριθμος δέχεται ως είσοδο εικόνες διαστάσεων 300*300 πρέπει το αρχικό bitmap διαστάσεων 640*480 να αναπροσαρμοστεί. Για την διαδικασία αυτή είναι υπεύθυνη η συνάρτηση `getResizedBitmap()`.

```

private Bitmap getResizedBitmap(Bitmap bitmap){
    int width = bitmap.getWidth();
    int height = bitmap.getHeight();

    float scaleWidth = ((float) 300) / width;
    float scaleHeight = ((float) 300) / height;
    Matrix matrix = new Matrix();
    matrix.postScale(scaleWidth, scaleHeight);

    int orientation =
context.getResources().getConfiguration().orientation;
    if (orientation == Configuration.ORIENTATION_PORTRAIT) {
        matrix.postRotate(90);
    } else {
        matrix.postRotate(0);
    }

    return Bitmap.createBitmap(bitmap, 0, 0, width, height, matrix,
        false);
}

```

Τέλος το νέο bitmap διαστάσεων 300*300, μετατρέπεται σε πίνακα byte με την συνάντηση `convertBitmapToByteBuffer()` με την για την γρηγορότερη εκτέλεση πράξεων από τον αλγόριθμο εντοπισμού.

```
private void convertBitmapToByteBuffer(Bitmap resizedBitmap) {
    if (arrayOfBytes == null) {
        return;
    }
    arrayOfBytes.rewind();
    resizedBitmap.getPixels(pix, 0, resizedBitmap.getWidth(),
0,0,resizedBitmap.getWidth(), resizedBitmap.getHeight());
    int pixel = 0;
    for (int i = 0; i < imgX; i++){
        for (int j = 0; j < imgY; j++){
            final int val = pix[pixel++];
            arrayOfBytes.put((byte) ((val >> 16) & 0xFF));
            arrayOfBytes.put((byte) ((val >> 8) & 0xFF));
            arrayOfBytes.put((byte) (val & 0xFF));
        }
    }
}
```

Οι συναρτήσεις για την πρόσβαση στα αποτελέσματα εντοπισμού είναι η `getObjPosition()` για τις συντεταγμένες του αντικειμένου, η `getObjLabel()` για την ετικέτα, η `getObjConf()` για το ποσοστό ακρίβειας αναγνώρισης και η `getNumDetections()` για το πλήθος των εντοπισμών. Στη δική μου περίπτωση ο αριθμός αντικειμένων ανά εικόνα έχει περιοριστεί σε ένα την φορά

```
public RectF getObjPosition() {return objPos;}
public String getObjLabel() {return objLabel;}
public Float getObjConf() {return objConf;}
public Float getNumDetections() {return numOfObjDetected;}
```

4.9 Ανάγνωση Αλγορίθμου και Ετικετών

Για την ανάγνωση του αλγορίθμου εντοπισμού απο το αρχείο detect.tflite χρησιμοποιήθηκε η συνάρτηση loadModelFile().

```
private MappedByteBuffer loadModelFile() throws IOException {
    String modelFile = "detect.tflite";
    AssetFileDescriptor fileDescriptor = context.getAssets()
                                                .openFd(modelFile);
    FileInputStream inputStream = new FileInputStream(
        fileDescriptor.getFileDescriptor());
    FileChannel fileChannel = inputStream.getChannel();
    long startOffset = fileDescriptor.getStartOffset();
    long declaredLength = fileDescriptor.getDeclaredLength();
    MappedByteBuffer mappedByteBuffer = fileChannel.map(
        FileChannel.MapMode.READ_ONLY, startOffset,
declaredLength);
    fileDescriptor.close();
    inputStream.close();
    return mappedByteBuffer;
}
```

Ενώ για την ανάγνωση των ετικέτων των χειρονομιών η loadLabelsList().

```
private List<String> loadLabelsList() throws IOException {
    List<String> lList = new ArrayList<String>();
    String labelsFile = "label.txt";
    BufferedReader reader = new BufferedReader(new
    InputStreamReader(context.getAssets().open(labelsFile)));
    String line;
    while ((line = reader.readLine()) != null) {
        lList.add(line);
    }
    reader.close();
    return lList;
}
```

Παρατηρήσεις και Αποτελέσματα

Με την ολοκλήρωση της ανάπτυξης της εφαρμογής ακολούθησαν δοκιμές για τον έλεγχο των λειτουργιών στο Raspberry Pi 3 B+. Τα αποτελέσματα έδειξαν πως με την χρήση του λειτουργικού συστήματος Android στην συσκευή η απόδοση της εφαρμογής έφτασε τα 17-25 καρέ το δευτερόλεπτο σε ότι αφορά την μετάδοση εικόνας απο την κάμερα παρεχοντας ετσι ομαλή ανταπόκριση κατά την εκτέλεση των εντολών με βάση τις χειρονομίες του χρήστη. Η εκτέλεση αντίστοιχης εφαρμογής σε περιβάλλον Raspberry Pi OS με χρήση της γλώσσας Python είχε ως αποτέλεσμα τα 3-5 καρέ το δευτερόλεπτο με αποτέλεσμα να είναι πρακτικά αδύνατη η λειτουργία της εφαρμογής σε πραγματικό χρόνο καθώς η απώλεια οπτικών δεδομένων δεν επιτρέπει την έγκαιρη ανταπόκριση στις εντολές απο την αναγνώριση των χειρονομιών. Εξαιτίας της απόδοσης του Android λειτουργικού συστήματος δεν ήταν αναγκαία η χρήση κάποιου Usb Accelerator το οποίο είναι μια εξωτερική συσκευή η οποία αναλαμβάνει την πραγματοποίηση των υπολογισμών που εκτελεί ο αλγόριθμος μηχανικής όρασης κατά την διάρκεια της διαδικασίας εντοπισμού και αυξάνει την επίδοση της εφαρμογής.

Για την μετεκπαίδευση του αλγορίθμου SSD-MobileNet V2 δημιούργησα ένα σετ δεδομένων απο 180 εικόνες για τις 3 κατηγορίες χειρονομιών. Το σετ χωρίστηκε σε αναλογία 80% για την εκπαίδευση και το υπολοιπο 20% για επαλήθευση, ύστερα απο πολύωρη εκπαίδευση το ποσοστό σφάλματος έφτασε περίπου τα 2%. Ο αλγόριθμος πραγματοποιεί εξαιρετικά αποτελεσματικά την αναγνώριση των χειρονομιών, αλλά η απόδοση σταδιακά μειώνεται με βάση την απόσταση της χειρονομίας απο την κάμερα με αποτέλεσμα το ποσοστό επιτυχημένης αναγνώρισης να είναι πολύ μικρότερο του 98%. Πιθανοί παράγοντες που ευθύνονται για την ανάλογη συμπεριφορά μπορεί να είναι α) Η ποσότητα των δεδομένων εκπαίδευσης, δηλαδή να αυξηθεί ο αριθμός των εικόνων για κάθε κατηγορία τουλάχιστον στο διπλάσιο, β) Βελτίωση της ποιότητας των φωτογραφιών, η συσκευή με την οποία τράβηξα της φωτογραφίες των χειρονομιών ειχε κάμερα 12 megapixel, συνεπώς χάνονται οι λεπτομέρειες και μειώνεται η ευκρίνεια των φωτογραφιών και γ) Εκπαίδευση του αλγορίθμου για μεγαλύτερο διάστημα. Βέβαια ένα ακόμα πρόβλημα είναι ότι για την καλύτερη απόδοση της εφαρμογής στο Raspberry Pi η εικόνα που μεταδίδει η κάμερα έχει ρυθμιστεί σε διαστάσεις 640x480 και όχι στα 1920x1080.

Επίσης στο Κεφάλαιο 2 έγινε αναφορά στο Raspberry pi 3 B+ και στα τεχνικά χαρακτηριστικά του, πλέον στην αγορά κυκλοφορεί το επόμενο μοντέλο το Raspberry Pi 4 B με τετραπύρηνo επεξεργαστή Cortex-A72 (ARM v8) 64bit και μνήμη ram έως και 8GB. Σε μελλοντικές δοκιμές και περαιτέρω ανάπτυξη της εφαρμογής θα ήθελα να δοκιμάσω τις δυνατότητες που μπορεί να προσφέρει το νέο μοντέλο κυρίως για τον εμπλουτισμό των

λειτουργιών της εφαρμογής όπως η λήψη βίντεο, ταυτόχρονη προβολή εικόνας απο δύο κάμερες και προσθήκη λειτουργίας ζουμ με βάση το άνοιγμα και το κλείσιμο του χεριού.

Τέλος η σημαντικότερη αλλαγή που θα ήθελα να πραγματοποιήσω στην εφαρμογή που έφτιαξα είναι να αντικαταστήσω τον αλγόριθμο τεχνητής νοημοσύνης με έναν που θα αναπτύξω ο ίδιος απο την αρχή ως το τέλος.

73

Βιβλιογραφία

- [1] Machine Perception of Three-Dimensional Solids by Lawrence Gilman Roberts.
<https://dspace.mit.edu/bitstream/handle/1721.1/11589/33959125-MIT.pdf?sequence=2>
- [2] SHAPE AND ARRANGEMENT OF COLUMNS IN CAT'S STRIATE CORTEX BY D. H. HUBEL AND T. N. WIESEL From the Neurophysiology Laboratory, Department of Pharmacology, Harvard Medical School, Boston, Massachusetts, U.S.A.
<https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1963.sp007079>
- [3] Neocognitron: A hierarchical neural network capable of visual pattern recognition by Kuniyiko Fukushima.
<https://www.sciencedirect.com/science/article/abs/pii/0893608088900147>
- [4] Gradient-Base Learning Applied to Document Recognition by Yann LeCun, Leon Battou, Yoshua Bengio, and Patrick Haffner.
<http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>
- [5] ImageNet Classification with Deep Convolutional Neural Networks by Alex Krizhevsky University of Toronto, Ilya Sutskever University of Toronto, Geoffrey E. Hinton University of Toronto.
<https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- [6] Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5) Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik UC Berkeley.
<https://arxiv.org/abs/1311.2524>
- [7] Fast R-CNN Ross Girshick Microsoft Research.
<https://arxiv.org/abs/1504.08083>
- [8] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION Karen Simonyan * & Andrew Zisserman + Visual Geometry Group, Department of Engineering Science, University of Oxford.
<https://arxiv.org/abs/1409.1556>

[9] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun.
<https://arxiv.org/abs/1506.01497>

74

[10] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*† University of Washington*, Allen Institute for AI†, Facebook AI Research.
<https://arxiv.org/abs/1506.02640>

[11] Going deeper with convolutions. Christian Szegedy Google Inc. Wei Liu University of North Carolina, Chapel Hill Yangqing Jia Google Inc. Pierre Sermanet Google Inc. Scott Reed University of Michigan Dragomir Anguelov Google Inc. Dumitru Erhan Google Inc. Vincent Vanhoucke Google Inc. Andrew Rabinovich Google Inc.
<https://arxiv.org/abs/1409.4842>

[12] SSD: Single Shot MultiBox Detector Wei Liu¹, Dragomir Anguelov², Dumitru Erhan³, Christian Szegedy³, Scott Reed⁴, Cheng-Yang Fu¹, Alexander C. Berg¹ ¹UNC Chapel Hill ²Zoox Inc. ³Google Inc. ⁴University of Michigan, Ann-Arbor.
<https://arxiv.org/abs/1512.02325>

[13] MobileNetV2: Inverted Residuals and Linear Bottlenecks Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen Google Inc.
<https://arxiv.org/pdf/1801.04381.pdf>

[14] TensorFlow: A system for large-scale machine learning.
(November 2–4, 2016 • Savannah, GA, USA).

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zhen.
<https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>

