ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΕ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ ΜΕ ΧΡΗΣΗ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Στασινού Χαράλαμπου – Νεκτάριου

Αριθμός Μητρώου: 711141036

Επιβλέποντες:

Καθηγήτρια Κλειώ Σγουροπούλου

Δρ. Χρήστος Τρούσσας

Αθήνα, Μάιος 2021

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΕ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ ΜΕ ΧΡΗΣΗ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Στασινός Χαράλαμπος – Νεκτάριος

Επιβλέποντες Καθηγητές:

Καθηγήτρια Κλειώ Σγουροπούλου

Δρ. Χρήστος Τρούσσας

Εγκρίθηκε από την Τριμελή Επιτροπή τον Ιούλιο του 2021:

KLEIO SGOUROPOULOU Digitally signed by KLEIO SGOUROPOULOU Date: 2021.07.22 10:23:05 +03'00'	Athanasios Voulodimos Digitally signed by Athanasios Voulodimos Date: 2021.07.23 00:44:59 +03'00'	Christos Troussas Digitally signed by Christos Troussas Date: 2021.07.20 17:04:12 +03'00'
(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Καθηγήτρια	Επ. Καθηγητής	Δρ. Χρήστος Τρούσσας
Κλειώ Σγουροπούλου	Αθανάσιος Βουλόδημος	

Δήλωση Συγγραφέα Διπλωματικής Εργασίας

Ο κάτωθι υπογεγραμμένος Στασινός Χαράλαμπος - Νεκτάριος του Γεωργίου, με αριθμό μητρώου 711141036 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Περιεχόμενα

Περίληψη

Με την ανάπτυξη της τεχνολογίας και την μεγάλη εξάπλωση των κινητών συσκευών ένα μεγάλο μέρος του πληθυσμού έχει έρθει σε επαφή με πλήθος νέων τεχνολογιών, οι οποίες παλαιότερα ήταν διαθέσιμες μόνο μέσω τη χρήσης εξειδικευμένων συσκευών ή δεν ήταν καθόλου προσβάσιμες. Μία τεχνολογία που έγινε εύκολα διαθέσιμη με την ανάπτυξη των κινητών συσκευών είναι η όραση υπολογιστών.

Στην παρούσα εργασία θα παρουσιαστεί η ανάπτυξη ενός εκπαιδευτικού λογισμικού για κινητές συσκευές που χρησιμοποιούν το λειτουργικό σύστημα Android, το οποίο θα βοηθάει τους χρήστες στην αναγνώριση αντικειμένων μέσα από μια φωτογραφία, θα τα σημειώνει πάνω σε αυτήν και θα αναφέρει από κάτω τις τρεις καλύτερες απαντήσεις με αναφορά του ποσοστού σιγουριάς ανά αντικείμενο. Το συγκεκριμένο λογισμικό θεωρείται εκπαιδευτικό, διότι μπορεί να χρησιμοποιηθεί ως βοήθημα αλλά και ως εργαλείο από τους χρήστες ,ώστε να αναγνωρίσουν τα αντικείμενα στο χώρο τους. Επιπλέον, μπορεί να βοηθήσει στην αναγνώριση ηλεκτρονικών ειδών από άτομα που δεν διαθέτουν την ανάλογη γνώση.

Για την επίτευξη του στόχου της εργασίας η εφαρμογή γράφτηκε με χρήση της γλώσσας προγραμματισμού Kotlin. Ακόμα αξιοποιήθηκαν οι τεχνολογίες AutoML της Google, για την εκπαίδευση των μοντέλων μηχανικής μάθησης, αλλά και τα εργαλεία για αναγνώριση αντικειμένων και για επισήμανση εικόνων του Google Machine Learning Kit.

Για την εκπαίδευση των μοντέλων μηχανικής μάθησης χρησιμοποιήθηκαν το σετ δεδομένων CIFAR – 100, ένα σύνολο δεδομένων με ηλεκτρονικά είδη και ένα σύνολο δεδομένων με πέντε διαφορετικούς τύπους λουλουδιών.

Κατά τη χρήση της εφαρμογής διαπιστώθηκε ότι αναγνωρίζονται κατά πλειοψηφία τα αντικείμενα στις εικόνες που δοκιμάστηκαν. Στο πρώτο μοντέλο μηχανικής μάθησης, με τα συνηθισμένα αντικείμενα, παρατηρήθηκε ότι αν το αντικείμενο ανήκει σε κάποια κατηγορία από αυτές που περιέχει το μοντέλο, τότε η σωστή απάντηση εμφανίζεται στις τρεις πρώτες προβλέψεις. Στο δεύτερο μοντέλο μηχανικής μάθησης, με τα ηλεκτρονικά είδη, παρατηρήθηκαν οι περισσότερες λανθασμένες προβλέψεις κυρίως λόγω του περιορισμού ανίχνευσης των αντικειμένων. Στο τελευταίο μοντέλο μηχανικής μάθησης, με τα λουλούδια, παρατηρήθηκε 100% επιτυχία αναγνώρισης των αντικειμένων εξαιτίας του μικρού αριθμού κατηγοριών.

Abstract

With the development of technology and the great spread of mobile devices, a large part of the population has come into contact with a number of new technologies, which in the past were only available through the use of specialized devices, or were not available at all. One technology that has become readily available with the development of mobile devices is computer vision.

In this thesis we will present the development of an educational software for mobile devices using the Android operating system, which will help users to identify objects through a photo, mark them on it and list below the three best answers with an indication of the percentage of certainty per item. This software is considered educational because it can be used as an aid but also as a tool by users to identify objects in their space. In addition, it can help identify electronic items from people who do not have the appropriate knowledge.

To achieve the goal of this thesis, the application was written using Kotlin as the programming language. Also, for the training of the custom models of machine learning, Google's AutoML technologies were used, and for object detection and for image labeling, Google's ML Kit tools were utilized.

For the training of machine learning models, the CIFAR – 100 was used, and also a dataset featuring electronics, and a dataset containing five different types of flowers.

While using the application, it was found that the objects in the tested images were identified by a majority. In the first machine learning model, with the usual objects, it was observed that if the object belongs to some of the categories contained in the model, then the correct answer appears in the first three predictions. In the second model of machine learning, with electronic items, the most erroneous predictions were observed mainly due to the limited detection of objects. In the latest model of machine learning, with flowers, 100% success of object recognition was observed due to the small number of categories.

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Ο ανθρώπινος νους επιχειρεί να κατανοήσει το περιβάλλον γύρω του μέσω της παρατήρησης. Οι πληροφορίες σε αυτό το περιβάλλον εμφανίζονται με διάφορες μορφές, κατανοητές και μη. Με την ανάπτυξη της τεχνολογίας, οι δυνατότητες που έρχονται με αυτή αυξάνονται. Μία από αυτές τις δυνατότητες είναι η μηχανική μάθηση. Το 1959, ο Άρθουρ Σάμιουελ, που θεωρείται ένας από τους πατέρες της τεχνητής νοημοσύνης και της ανάπτυξης βιντεοπαιχνιδιών για υπολογιστές, όρισε τη μηχανική μάθηση ως «Πεδίο μελέτης που δίνει τη δυνατότητα στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί» (Samuel, 1959). Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων, οι οποίοι έχουν τη δυνατότητα να μαθαίνουν από τα δεδομένα που δέχονται, και να κάνουν προβλέψεις σχετικά με αυτά. Πλέον η μηχανική μάθηση είναι εύκολα προσβάσιμη στη δημιουργία και την χρήση των μοντέλων της με εργαλεία όπως το Google Machine Learning Kit SDK που χρησιμοποιήθηκε στην ανάπτυξη αυτής της εφαρμογής.

1.2 Κίνητρο για τη διεξαγωγή της εργασίας

Με την ραγδαία εξέλιξη της τεχνολογίας των κινητών συσκευών, οι δυνατότητες που παρέχουν στους χρήστες αυξάνονται συνεχώς. Με τη συνεχή αναβάθμιση της επεξεργαστικής ισχύος και την προσθήκη νέων τεχνολογιών, έχει δημιουργηθεί η επιθυμία για εφαρμογές οι οποίες αξιοποιούν τις νέες αυτές δυνατότητες. Οι εφαρμογές αυτές μπορούν να χρησιμοποιηθούν και για εκπαιδευτικούς λόγους κι όχι μόνο για ψυχαγωγία. Σχεδόν όλα τα έξυπνα τηλέφωνα σήμερα διαθέτουν κάμερες, οι οποίες με το κατάλληλο λογισμικό μπορούν να χρησιμοποιηθούν και, μέσω των εφαρμογών, να διευκολύνουν αρκετά την καθημερινότητα των χρηστών ή να τους βοηθήσουν στην εκπαίδευση τους.

1.3 Στόχος της εργασίας

Ο στόχος της εργασίας είναι η ανάπτυξη της Android εφαρμογής Object Recognition, η οποία θα βοηθάει τους χρήστες να αναγνωρίσουν αντικείμενα σε μία εικόνα, είτε με χρήση της κάμερας, είτε επιλέγοντας φωτογραφίες από τον αποθηκευτικό χώρο της συσκευής. Κατά την επιτυχημένη αναγνώριση τουλάχιστον ενός αντικειμένου, η εφαρμογή θα δείχνει πάνω στην εικόνα σε ποιο σημείο βρήκε το αντικείμενο και σε μία λίστα κάτω ή δίπλα από την εικόνα, ανάλογα με τον προσανατολισμό της συσκευής, θα εμφανίζονται οι τρεις πιο πιθανές προβλέψεις του μοντέλου μηχανικής μάθησης για το αντικείμενο και το ποσοστό ακρίβειας της εκάστοτε πρόβλεψης. Επίσης ο χρήστης έχει την επιλογή να επιλέξει ανάμεσα σε τρία μοντέλα μηχανικής μάθησης για να αναγνωρίσει συγκεκριμένα πράγματα. Στο πρώτο μοντέλο υπάρχουν ετικέτες για 100 διαφορετικά συνηθισμένα αντικείμενα, στο δεύτερο μοντέλο υπάρχουν τίτλοι για 36 διαφορετικά ηλεκτρονικά είδη, ενώ στο τρίτο μοντέλο υπάρχουν 5 κατηγορίες για είδη λουλουδιών.

1.4 Κύριες Τεχνολογίες που χρησιμοποιήθηκαν

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού Kotlin, η οποία δημιουργήθηκε το 2011 από την εταιρεία JetBrains γνωστή για το IDE IntelliJ IDEA, όπου έχει βασιστεί πάνω του το Android Studio. Η Kotlin τρέχει πάνω στην εικονική μηχανή της Java (JVM) και έχει επηρεαστεί από τις γλώσσες Java και Scala. Η πρώτη επίσημη έκδοση της γλώσσας αυτής ή αλλιώς Kotlin v1.0 κυκλοφόρησε στις 15 Φεβρουαρίου 2016, ενώ τον Μάιο του 2019 η Google ανακοίνωσε πως η επίσημη γλώσσα για τον προγραμματισμό των Android εφαρμογών είναι πλέον η Kotlin και προέτρεψε τους προγραμματιστές, είτε να αλλάξουν την γλώσσα που χρησιμοποιούν, είτε να την δοκιμάσουν έτσι ώστε να έχουν μία πρώτη επαφή μαζί της.

Εικόνα 1.1: Λογότυπο προγραμματιστικής γλώσσας Kotlin.

Επίσης χρησιμοποιήθηκε το Machine Learning Kit SDK, ένα προϊόν φτιαγμένο από τη Google, το οποίο παρουσιάστηκε το 2018 και αποτελείται από ένα σύνολο εργαλείων ανάπτυξης λογισμικού, που επιτρέπουν στους προγραμματιστές να απλοποιήσουν την ενσωμάτωση μοντέλων μηχανικής μάθησης στις εφαρμογές τους για κινητές συσκευές. Κατά την χρήση αυτών των εργαλείων δίνεται η δυνατότητα χρήσης κάποιων έτοιμων μοντέλων της Google, καθώς και για ορισμένες περιπτώσεις, όπως η επισήμανση εικόνων (Image Recognition), υπάρχει η δυνατότητα της δημιουργίας νέων μοντέλων, ανάλογα πάντα με τις απαιτήσεις της εφαρμογής.

Εικόνα 1.2: Λογότυπο Google Machine Learning Kit SDK.

Τέλος χρησιμοποιήθηκε το εργαλείο Cloud AutoML Vision της Google, το οποίο ανήκει στην πλατφόρμα Google Cloud Platform. Το Google Cloud Platform ξεκίνησε τη λειτουργία του τον Απρίλιο του 2008 με την κυκλοφορία του Google App Engine και πλέον παρέχει πάνω από 100 προϊόντα για προγραμματιστές. Το Cloud AutoML Vision βρίσκεται σε beta έκδοση από τον Σεπτέμβριο του 2018, μέχρι και τη στιγμή που γράφτηκε αυτή η εργασία.

Εικόνα 1.3: Λογότυπο Google Cloud AutoML.

1.5 Ιστορική Αναδρομή Λειτουργικού Συστήματος Android

Το λειτουργικό σύστημα Android δημιουργήθηκε το 2003 στο Palo Alto της Καλιφόρνια από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Αρχικά ήταν ένα λειτουργικό σύστημα για ψηφιακές φωτογραφικές μηχανές, ενώ το 2004 πήραν την απόφαση να το μετατρέψουν σε λειτουργικό σύστημα για κινητά με σκοπό να ανταγωνιστεί το λογισμικό Symbian που υπήρχε κυρίως σε συσκευές Nokia, και το λογισμικό Microsoft Windows Mobile.

Το 2005 η Google απέκτησε το λειτουργικό σύστημα Android για 50 εκατομμύρια δολάρια. Το Android τροφοδοτείται από τον πυρήνα του Linux. Η πρώτη συσκευή που χρησιμοποίησε το λειτουργικό σύστημα ήταν το HTC Dream και κυκλοφόρησε στα τέλη του 2008 με την έκδοση Android 1.0. Από το 2008 το σύστημα Android έχει δει πολλές αναβαθμίσεις και εκδόσεις με την πιο πρόσφατη έκδοση να είναι το Android 11, την περίοδο που γράφεται αυτή η εργασία.

Εικόνα 1.4: Λογότυπο Λειτουργικού Συστήματος Android (2019 – σήμερα).

1.6 Δομή Εργασίας

Κεφάλαιο 1ο – Εισαγωγή: Πρόκειται για αυτό το κεφάλαιο το οποίο περιέχει γενικές πληροφορίες για την διπλωματική εργασία που εκπονείται στα πλαίσια του Προγράμματος Σπουδών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών.

Κεφάλαιο 2ο – Ανασκόπηση Πεδίου: Σε αυτό το κεφάλαιο παρουσιάζουμε κάποιες γνωστές εφαρμογές παρόμοιες με τη δική μας και κατόπιν τις συγκρίνουμε με αυτή.

Κεφάλαιο 3ο – Παρουσίαση Εφαρμογής: Σε αυτό το κεφάλαιο παρουσιάζουμε την εφαρμογή που δημιουργήσαμε, εξηγούμε τον τρόπο χρήσης της και αναλύουμε τον κώδικα της εφαρμογής.

Κεφάλαιο 4ο – Training Datasets: Αυτό το κεφάλαιο περιέχει αναφορές στη εκπαίδευση των μοντέλων μηχανικής μάθησης και στα εργαλεία τα οποία χρησιμοποιήθηκαν.

Κεφάλαιο 5ο – Αποτελέσματα: Αυτό το κεφάλαιο περιέχει τα αποτελέσματα της εφαρμογής μας, μαζί με στατιστικά στοιχεία για τα μοντέλα μηχανικής μάθησης.

Κεφάλαιο 6ο – Συμπεράσματα: Σε αυτό το κεφάλαιο αναφερόμαστε στα προβλήματα που αντιμετωπίσαμε κατά την διάρκεια της ανάπτυξης της εφαρμογής, τους περιορισμούς που είχαμε και στις πιθανές μελλοντικές επεκτάσεις που μπορούμε να κάνουμε.

Κεφάλαιο 2ο : Ανασκόπηση Πεδίου

2.1 Google Lens

Η εφαρμογή Google Lens είναι η πιο γνωστή εφαρμογή αναγνώρισης αντικειμένων και κυκλοφόρησε το 2017 για τις συσκευές Google Pixel και Google Pixel 2, ενώ από το 2018 κυκλοφορεί και για τις υπόλοιπες συσκευές Android, με προϋπόθεση να έχουν εγκατεστημένη την έκδοση Android 6.0, γνωστή και ως Marshmallow, ή και νεότερη έκδοση.

Εικόνα 2.1: Λογότυπο Εφαρμογής Google Lens.

Κάποιες δυνατότητες που έχει η εφαρμογή Google Lens:

- Μετάφραση κειμένου σε πραγματικό χρόνο
- Αντιγραφή κειμένου από φωτογραφία
- Αναζήτηση αντικειμένων για αγορά με χρήση της κάμερας
- Αναγνώριση τοποθεσιών, για παράδειγμα μνημεία
- Επίλυση μαθηματικών εξισώσεων
- Αναγνώριση Φαγητού
- Εύρεση οπτικά πανομοιότυπων εικόνων

Πιο αναλυτικά στις περισσότερες δυνατότητες χρησιμοποιούμε την κάμερα μέσω της εφαρμογής, τραβάμε μία φωτογραφία και η εφαρμογή συλλέγει όσες πληροφορίες μπορεί από αυτή. Στην περίπτωση της αναζήτησης αντικειμένων μας επιστρέφει το προϊόν, εκτός και αν δεν βρεθεί οπότε μας επιστρέφει ένα οπτικά πανομοιότυπο αντικείμενο, και εμφανίζει το όνομά του και ένα κουμπί που μας παραπέμπει σε μια αναζήτηση στο Google για το προϊόν αυτό, ενώ στην μετάφραση κειμένου χρησιμοποιώντας την κάμερα, μπορούμε να βρούμε κάποιο κείμενο σε άλλη γλώσσα και η εφαρμογή το αντικαθιστά στην ίδια θέση με το μεταφρασμένο στη γλώσσα που έχουμε επιλέξει. Ακόμα στην επίλυση μαθηματικών εξισώσεων έχουμε τη δυνατότητα να δούμε κατευθείαν τη λύση κάποιας οποιασδήποτε εξίσωσης, αλλά και αναλυτικά τα βήματα για να φτάσουμε σε αυτή τη λύση.

2.2 Aipoly Vision

Το Aipoly Vision είναι μία εφαρμογή της εταιρίας V7 και δίνει τη δυνατότητα στους χρήστες με χρήση της κάμερας της κινητής συσκευής να αναγνωρίσουν τα αντικείμενα που βρίσκονται στο χώρο. Επίσης αυτή η εφαρμογή δείχνει στο χρήστη το όνομα του αντικειμένου που αναγνώρισε αλλά και με φωνητική χρήση λέει δυνατά και καθαρά το όνομα του αντικειμένου στο χρήστη. Η εφαρμογή αυτή, έχει ως στόχο να βοηθήσει τους ανθρώπους που έχουν προβλήματα όρασης, όπως τύφλωση και αχρωματοψία.

Εικόνα 2.2: Λογότυπο εφαρμογής Aipoly Vision.

Η εφαρμογή αυτή χωρίζεται σε δύο εκδόσεις η πρώτη που είναι δωρεάν με συγκεκριμένες δυνατότητες και η δεύτερη που είναι διαθέσιμη μέσω μηνιαίας ή ετήσιας συνδρομής, όπου έχει όλες τις δυνατότητες διαθέσιμες.

Οι δυνατότητες που έχει η δωρεάν έκδοση της εφαρμογής Aipoly Vision είναι οι εξής:

- Αναγνώριση χιλίων συνηθισμένων αντικειμένων.
- Αναγνώριση διακοσίων διαφορετικών χρωμάτων.
- Αυτόματη έναρξη φλας συσκευής, όταν οι συνθήκες φωτισμού είναι κακές.
- Χρήση εφαρμογής χωρίς να χρειάζεται σύνδεση στο διαδίκτυο.

Οι έξτρα δυνατότητες που προσφέρει η συνδρομητική εφαρμογή είναι:

- Ανάγνωση κειμένου σε επτά γλώσσες.
- Αναγνώριση οκτακοσίων διαφορετικών πιάτων φαγητού.
- Αναγνώριση χιλίων ειδών φυτών και λουλουδιών.
- Αναγνώριση χιλίων ειδών ζώων.
- Αναγνώριση αξίας αμερικάνικων δολαρίων.

2.3 TapTapSee

Η εφαρμογή TapTapSee είναι μια εφαρμογή από την εταιρία Cloudsight Inc. και χρησιμοποιεί το Image Recognition API της ίδιας εταιρίας για να αναγνωρίζει τα αντικείμενα στο χώρο του χρήστη με την χρήση της κάμερας. Η εφαρμογή έχει στόχο να βοηθήσει όλους εκείνους τους ανθρώπους οι οποίοι έχουν προβλήματα όρασης στην καθημερινότητά τους. Ακόμα η εφαρμογή με την αναγνώριση του αντικειμένου δείχνει στο χρήστη αναλυτική περιγραφή του αντικειμένου, για παράδειγμα μαύρο πληκτρολόγιο πάνω σε καφέ τραπέζι, και με τη χρήση του TalkBack, σε συσκευές που το υποστηρίζουν, αναφέρει στο χρήστη με τη χρήση της φωνής την λεπτομερή περιγραφή του αντικειμένου.

Εικόνα 2.3: Λογότυπο εφαρμογής TapTapSee.

Οι επιλογές που προσφέρει η εφαρμογή TapTapSee είναι οι εξής:

- Αναγνώριση ενός αντικειμένου σε μία φωτογραφία μέσω της κάμερας.
- Αναγνώριση ενός αντικειμένου μέσω ενός βίντεο διάρκειας δέκα δευτερολέπτων.
- Αναγνώριση αντικειμένου από φωτογραφίες που υπάρχουν στον αποθηκευτικό χώρο της συσκευής.
- Αποθήκευση φωτογραφιών μαζί με την αναγνώριση του αντικειμένου σε αυτές.
- Κοινοποίηση αποτελεσμάτων μέσω κοινωνικών δικτύων, μηνυμάτων και email.

2.4 Calorie Mama AI

Η εφαρμογή Calorie Mama AI κυκλοφόρησε από την εταιρία Azumio Inc. και βοηθάει τους χρήστες να μετράνε τις θερμίδες κάθε φαγητού που τρώνε. Η εφαρμογή χρησιμοποιεί το Food AI API της ίδιας εταιρίας και με τη χρήση της κάμερας ο χρήστης τραβάει φωτογραφία το φαγητό του, έτσι ώστε η εφαρμογή να ψάξει στη βάση δεδομένων του Food AI και να βρει το όνομα του φαγητού αλλά και τις θερμίδες που αυτό έχει. Πριν την ολοκλήρωση της διαδικασίας αυτής, η εφαρμογή δίνει τη δυνατότητα στον χρήστη να βάλει χειροκίνητα το όνομα του πιάτου, σε περίπτωση που έχει γίνει λάθος στην αναγνώριση αυτού.

Εικόνα 2.4: Λογότυπο εφαρμογής Calorie Mama AI.

Η εφαρμογή χωρίζεται σε δύο εκδόσεις, όπου η μία είναι δωρεάν και προσφέρει μειωμένες δυνατότητες, ενώ η δεύτερη είναι διαθέσιμη μέσω μηνιαίας συνδρομής και προσφέρει περισσότερες επιλογές.

Μερικές δυνατότητες που δίνει η δωρεάν έκδοση της εφαρμογής Calorie Mama AI είναι οι εξής:

- Αναγνώριση φαγητών και εμφάνιση θερμίδων.
- Αποθήκευση αποτελεσμάτων ανά μερίδα.
- Δημιουργία πλάνου με προτεινόμενη ημερήσια κατανάλωση θερμίδων.

Ενώ με τη μηνιαία συνδρομή προσθέτονται οι εξής επιλογές:

- Δυνατότητα καταγραφής υδατανθράκων, λιπών και αλάτων, μαζί με τις θερμίδες.
- Δημιουργία εξειδικευμένων πλάνων, για παράδειγμα για αθλητές bodybuilding.
- Διάθεση ασκήσεων για γυμναστική στο σπίτι.

2.5 Σύγκριση Εφαρμογών

Σε αυτή την ενότητα θα συγκρίνουμε τις εφαρμογές και τις λειτουργίες τους με αυτή που δημιουργήσαμε στο πλαίσιο της διπλωματικής εργασίας.

Η εφαρμογή μας μάς δίνει τη δυνατότητα να αναγνωριστούν διαφορετικές ομάδες αντικειμένων, όπως ηλεκτρονικά είδη και λουλούδια. Αυτή τη δυνατότητα μας τη δίνουν και οι εφαρμογές Google Keep, η συνδρομητική έκδοση της εφαρμογής Aipoly Vision και η TapTapSee. Επίσης η εφαρμογή μας δεν χρειάζεται να έχει σύνδεση στο Ίντερνετ, όπως και η εφαρμογή Aipoly Vision. Η μεγαλύτερη διαφορά που έχουν όλες αυτές οι εφαρμογές με τη δικιά μας, είναι πως αυτές έχουν τη δυνατότητα να αναγνωρίσουν ένα αντικείμενο μόνο στην εικόνα, ενώ η δικιά μας εφαρμογή μπορεί να αναγνωρίσει περισσότερα αντικείμενα, εφόσον αυτά υπάρχουν στην εικόνα ή ανιχνευθούν από τον Object Detector του Google Machine Learning Kit.

Όμως ενώ οι εφαρμογές Google Keep και Aipoly Vision μπορούν σε πραγματικό χρόνο να αναγνωρίσουν αντικείμενα, η εφαρμογή μας έχει μόνο τη δυνατότητα να κάνει αναγνώριση αντικειμένων σε στατική φωτογραφία. Επίσης η εφαρμογή μας δεν έχει τη δυνατότητα να αναγνωρίσει αντικείμενα σε βίντεο, όπως κάνει η εφαρμογή TapTapSee. Ακόμα η εφαρμογή μας δεν προσφέρει τη δυνατότητα αναγνώρισης χρωμάτων, όπως κάνει η εφαρμογή Aipoly Vision, αλλά ούτε μπορεί να χρησιμοποιηθεί για μετάφραση ή σκανάρισμα κειμένου, όπως βλέπουμε στις εφαρμογές Google Keep και Aipoly Vision. Τέλος η εφαρμογή μας δεν έχει την επιλογή να αποθηκεύσει την φωτογραφία που τραβάμε με χρήση της κάμερας, εφόσον αναγνωριστούν αντικείμενα σε αυτή, όπως έχει η εφαρμογή TapTapSee.

Κεφάλαιο 3ο: Παρουσίαση Εφαρμογής

3.1 Εισαγωγή στην εφαρμογή Object Recognition

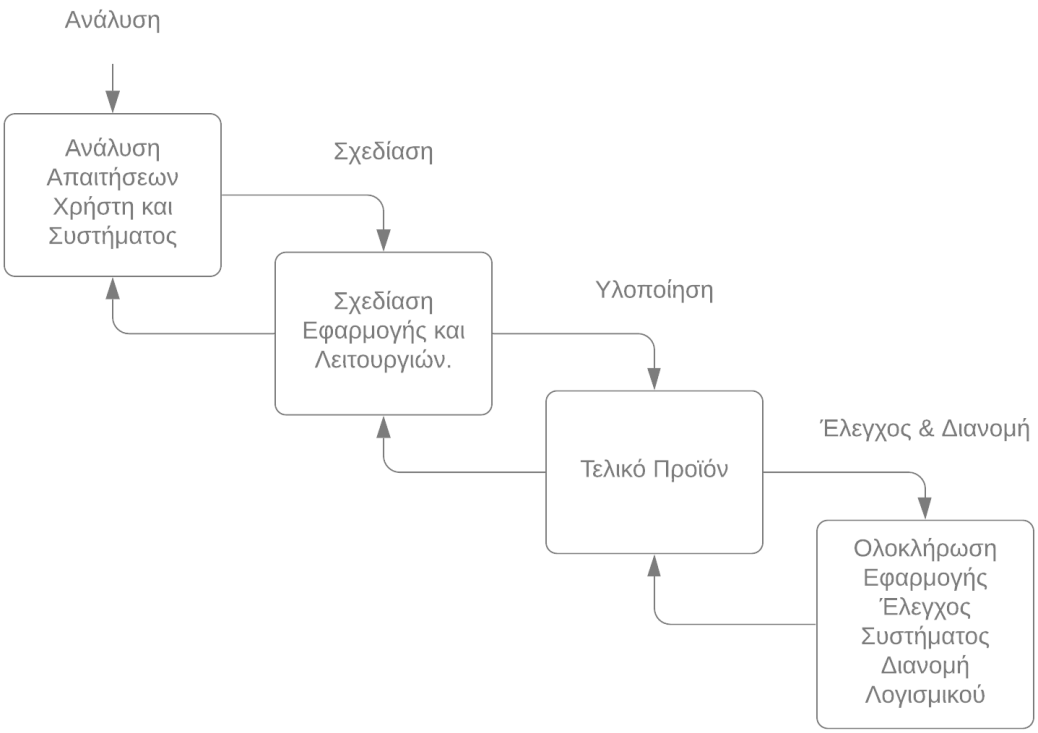
Σε αυτό το κεφάλαιο, θα παρουσιαστεί η εφαρμογή που δημιουργήθηκε στα πλαίσια της διπλωματικής εργασίας και θα δούμε τις φάσεις ανάπτυξης που ακολουθήσαμε για να φτιάξουμε αυτήν την εφαρμογή ενώ επίσης, θα δούμε κάποια χαρακτηριστικά κομμάτια του κώδικα της εφαρμογής, αλλά και στιγμιότυπα οθόνης, δηλαδή screenshots, της εφαρμογής όταν τρέχει σε μία κινητή συσκευή. Οι τρόποι χρήσης της εφαρμογής είναι δύο και αρκετά απλοί. Ο πρώτος τρόπος είναι η επιλογή από τον χρήστη μιας φωτογραφίας από τον αποθηκευτικό χώρο της συσκευής για να αναγνωρίσει τα αντικείμενα σε αυτή και ο δεύτερος τρόπος είναι να χρησιμοποιηθεί η κάμερα της κινητής συσκευής για να ψάξει ο χρήστης τα ονόματα των αντικειμένων στο χώρο του.

Εικόνα 3.1: Εικονίδιο εφαρμογής Object Recognition.

3.2 Φάσεις Ανάπτυξης Εφαρμογής

Σε αυτή την ενότητα θα αναφερθούμε και θα αναλύσουμε τις φάσεις ανάπτυξης της εφαρμογής μας. Η εφαρμογή αυτή αναπτύχθηκε με βάση το μοντέλο του καταρράκτη. Η πρώτη εμφάνιση του μοντέλου καταρράκτη είναι το μακρινό 1956 από τον Herbert D. Bennigton. Η ανάπτυξη του λογισμικού χωρίστηκε σε τέσσερις φάσεις. Η πρώτη φάση οποιασδήποτε εφαρμογής είναι η ανάλυση των απαιτήσεων του χρήστη και του συστήματος. Μόνο μετά την ανάλυση μπορεί να συνεχιστεί η ανάπτυξη της εφαρμογής, στη δεύτερη φάση, που θα είναι η σχεδίαση του συστήματος. Έπειτα η τρίτη φάση είναι η υλοποίηση της εφαρμογής και τέλος έχουμε την τέταρτη φάση κατά την οποία γίνεται ο έλεγχος του συστήματος και η διανομή της εφαρμογής. Οι φάσεις ανάπτυξης μιας εφαρμογής με χρήση του μοντέλου καταρράκτη φαίνονται παρακάτω στην εικόνα 3.2.[1]

[1] McConnell, S. (2006). *Software Estimation: Demystifying the Black Art.* Microsoft Press. "Waterfall vs. Agile Methodology." 2008. Agile Introduction for Dummies. Retrieved August 13, 2010, from http://agileintro.wordpress.com/2008/01/04/waterfall-vs-agile-methodology/



Εικόνα 3.2: Φάσεις Ανάπτυξης Εφαρμογής με μοντέλο καταρράκτη

3.2.1 Ανάλυση

Στην υποενότητα αυτή θα αναλύσουμε την φάση της ανάλυσης του συστήματος, καθώς και μια αρχική προσέγγιση της εφαρμογής που πρόκειται να αναπτυχθεί. Αυτό, δε, αποτελεί την πρώτη και πιο σημαντική φάση της ανάπτυξης ενός λογισμικού. Η ανάλυση θεωρείται η πιο σημαντική φάση, διότι όσο πιο σωστά γίνει, τόσο πιο ορθά θα προχωρήσουμε στις υπόλοιπες φάσεις της ανάπτυξης του λογισμικού, και με λιγότερα προβλήματα.

Στην παρούσα φάση της εφαρμογής καθορίζουμε τους κύριους στόχους της αλλά, και θα γίνει συγκέντρωση των απαιτήσεων των χρηστών. Παρακάτω βλέπουμε δύο βοηθητικούς πίνακες με ερωτήσεις και απαντήσεις που καθιστούν εύκολη την περαιτέρω οργάνωση της δομής του λογισμικού. Ο πρώτος πίνακας αναφέρεται στις απαιτήσεις που θα έχει το σύστημα από τους χρήστες, όπως για παράδειγμα η γνώση της Αγγλικής γλώσσας, ενώ ο δεύτερος πίνακας αναφέρει τις τεχνολογίες και το χρόνο υλοποίησης που απαιτείται για την ανάπτυξη αυτού του λογισμικού.

Πίνακας 3.1: Ανάλυση και Απαιτήσεις Χρηστών

Ερώτηση	**Απάντηση**
Υπάρχει ελάχιστο όριο στην ηλικία του χρήστη;	Ναι, άνω των 6 ετών.
Χρειάζεται ο χρήστης να ξέρει ξένες γλώσσες;	Ναι, ο χρήστης θα πρέπει να γνωρίζει την αγγλική γλώσσα.
Χρειάζεται ο χρήστης να έχει χρησιμοποιήσει παλαιότερα έξυπνο τηλέφωνο;	Ναι
Χρειάζεται ο χρήστης να έχει χρησιμοποιήσει παλαιότερα κάποια παρόμοια εφαρμογή;	Όχι
Σε ποιά βαθμίδα εκπαίδευσης ανήκουν οι πιθανοί χρήστες αυτής της εφαρμογής;	Από πρωτοβάθμια και πάνω.

Πίνακας 3.2: Ανάλυση Τεχνολογιών και Υπολογισμός Χρόνου Υλοποίησης Εφαρμογής

Ερώτηση	***Απάντηση***
Χρειάζεται η εφαρμογή πρόσβαση στο Διαδίκτυο;	Όχι
Χρειάζεται η εφαρμογή άδεια για πρόσβαση σε υπηρεσίες της συσκευής;	Ναι, χρειάζεται άδεια για χρήση της κάμερας και για τον αποθηκευτικό χώρο.
Ποιες είναι οι ελάχιστες απαιτήσεις του συστήματος;	Χρειάζεται κινητή συσκευή ή tablet με λειτουργικό σύστημα Android με έκδοση 8 και πάνω
Πώς θα γίνει η τελική διανομή του προϊόντος;	Προσωπική χρήση στο πλαίσιο διπλωματικής εργασίας.
Πόσο χρόνο έχουμε στη διάθεση μας για την ανάπτυξη του έργου;	Τέσσερις μήνες

Με τη χρήση αυτών των πινάκων διαμορφώθηκαν τα τμήματα υλοποίησης του έργου και αποφασίστηκε ο προγραμματισμός του χρόνου για αυτό. Παράλληλα τα παραπάνω τμήματα τοποθετήθηκαν σε χρονική σειρά και καθορίστηκε η χρονική διάρκειά τους. Στον παρακάτω πίνακα ακολουθεί το χρονοδιάγραμμα εκπόνησης του έργου.

Πίνακας 3.3: Χρονοδιάγραμμα Εκπόνησης Έργου

Πλάνο Εργασιών	**Μάρτιος 2021**	**Απρίλιος 2021**	**Μάιος 2021**	**Ιούνιος 2021**
Φάση 1η: Ανάλυση				
Καταγραφή Ιδεών				
Χρονικός Προγραμματισμός και Δημιουργία Πλάνου Έργου				
Φάση 2η: Σχεδίαση				
Σχεδίαση Γραφικής Διεπαφής της Εφαρμογής				
Σχεδίαση και Προγραμματισμός Λειτουργιών Εφαρμογής				
Φάση 3η: Υλοποίηση				
Γράψιμο και Επεξεργασία Κώδικα Εφαρμογής				
Φάση 4η: Έλεγχος και Διανομή				
Έλεγχος Λειτουργιών Εφαρμογής				
Διορθώσεις κώδικα εφαρμογής, για την παραγωγή του τελικού προϊόντος				

3.2.2 Σχεδίαση

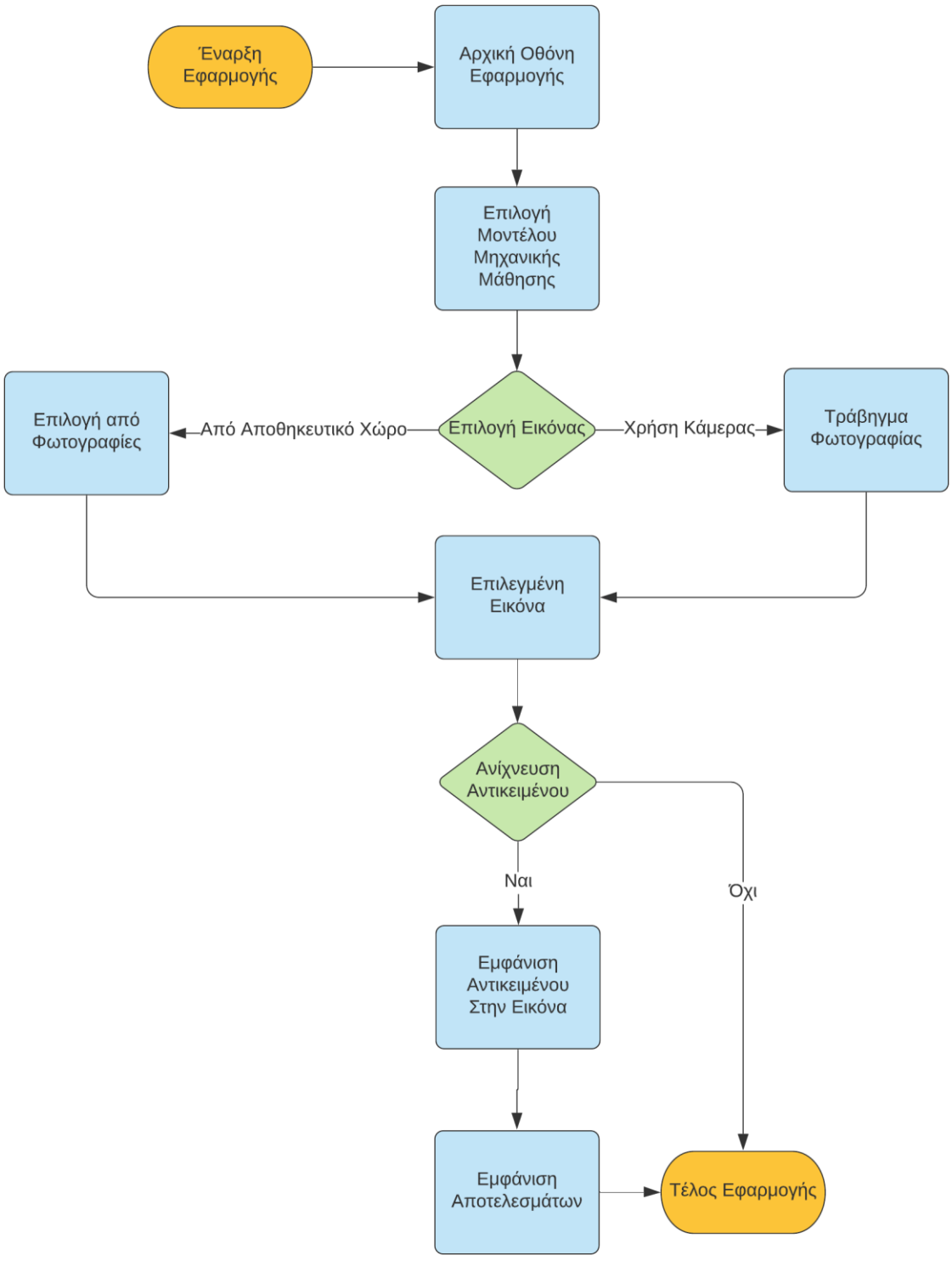
Η φάση της σχεδίασης είναι η δεύτερη κατά σειρά φάση για την ανάπτυξη του λογισμικού μας. Ο σκοπός αυτής της φάσης είναι να μετατραπούν οι γενικοί στόχοι που καθορίστηκαν στη φάση της ανάλυσης σε μια ολοκληρωμένη και αναλυτική περιγραφή της εφαρμογής. Επίσης, σε αυτή τη φάση καθορίζονται οι απαιτήσεις του συστήματος, η δομή αλλά και η αρχιτεκτονική του. Πιο αναλυτικά:

- *Απαιτήσεις Σχεδιασμού*:

Κατά τη διάρκεια του σχεδιασμού μιας εφαρμογής υπάρχουν κάποιες απαιτήσεις που θέλουμε να έχει. Τα εργαλεία που χρησιμοποιήθηκαν είναι open source ή ελεύθερα για χρήση, με μόνη εξαίρεση το Google Cloud AutoML, που το χρησιμοποιήσαμε για την εκπαίδευση των μοντέλων μηχανικής μάθησης, όπου κάναμε χρήση της δοκιμαστικής περιόδου που μας προσέφερε η Google. Για να αναλύσουμε αρχικά την εφαρμογή ελέγξαμε πρώτα διάφορες εφαρμογές με παρόμοιες λειτουργίες, όπως για παράδειγμα το Google Keep και το Aipoly Vision. Η μόνη απαίτηση που υπάρχει, όσον αφορά τη συσκευή, είναι να έχει τουλάχιστον την έκδοση 8 του λειτουργικού συστήματος Android, ενώ κατά τη διάρκεια του σχεδιασμού προστέθηκε η απαίτηση του απλού περιβάλλοντος διεπαφής για να είναι απλό, εύκολα προσβάσιμο για όλες τις ηλικιακές ομάδες και να μην απαιτεί ιδιαίτερες γνώσεις για το σωστό χειρισμό της εφαρμογής, πέραν μόνο της γνώσης της αγγλικής γλώσσας.

- *Δομή εφαρμογής*:

Η ανάπτυξη της εφαρμογής έγινε με τη χρήση της γλώσσας προγραμματισμού Kotlin, διότι είναι η γλώσσα που έχει επιλεχθεί από τη Google ως επίσημη γλώσσα για την ανάπτυξη εφαρμογών στο λειτουργικό σύστημα Android. Παρακάτω στην εικόνα 3.3 θα δούμε την διαγραμματική απεικόνιση της δομής του λογισμικού μας.



Εικόνα 3.3: Διάγραμμα Ροής Εφαρμογής

- *Αρχιτεκτονική Εφαρμογής:*

Για την αρχιτεκτονική της εφαρμογής επιλέχθηκε ο τρόπος Model – View – ViewModel (MVVM), ο οποίος είναι ο πιο περιζήτητος τρόπος γραφής λογισμικού Android την τρέχουσα χρονική στιγμή. Όπως φαίνεται και από το όνομα της αρχιτεκτονικής, υπάρχουν τρία μέρη σε αυτήν.

Το πρώτο μέρος είναι το Model, που περιέχει τις συναρτήσεις και τα δεδομένα της οθόνης που κοιτάει ο χρήστης. Για παράδειγμα στην εφαρμογή μας, η συνάρτηση για την αναγνώριση αντικειμένων βρίσκεται και εκτελείται στο Model κομμάτι του κώδικά μας.

Το επόμενο κομμάτι της αρχιτεκτονικής είναι το View, όπου σε αυτό υπάρχουν τα κομμάτια της γραφικής διεπαφής της εφαρμογής μας, όπως για παράδειγμα τα κουμπιά και το κείμενο που εμφανίζονται στο χρήστη.

Τέλος, το μέρος που απομένει είναι το ViewModel, όπου πήρε το όνομά του από τη χρήση του. Χρησιμοποιείται σαν σύνδεση μεταξύ των δύο άλλων μερών της αρχιτεκτονικής και η δουλειά του είναι να στέλνει τα δεδομένα στο κομμάτι View, αλλά και να λαμβάνει πληροφορίες από το View και να ζητάει να εκτελεστούν οι αντίστοιχες συναρτήσεις στο Model, για παράδειγμα να δίνει εντολή πως πατήθηκε ένα συγκεκριμένο κουμπί και να περιμένει το αποτέλεσμα.

Η πρώτη αναφορά στο μοντέλο MVVM έγινε το 2005 από τον John Gossman της Microsoft και είναι πλέον μία από τις κύριες αρχιτεκτονικές που χρησιμοποιούνται για την ανάπτυξη λογισμικού.

Ένα από τα πλεονεκτήματα της αρχιτεκτονικής MVVM είναι πως με τη χρήση της γίνεται πιο εύκολος ο έλεγχος ενός προγράμματος και κάνει το λογισμικό πιο ασφαλές και με καλύτερη απόδοση. Όμως η αρχιτεκτονική αυτή δεν συνιστάται στη δημιουργία απλών εφαρμογών, όπως για παράδειγμα μία εφαρμογή αριθμομηχανής διότι περιπλέκει τον κώδικα και θεωρείται υπερβολή.

3.2.3 Υλοποίηση

Σε αυτή την υποενότητα, θα αναφερθούμε στην φάση της υλοποίησης, η οποία αποτελεί και την τρίτη φάση στην ανάπτυξη του λογισμικού μας. Στη φάση αυτή θα βασιστούμε στα δεδομένα που προέκυψαν από τις δύο προηγούμενες φάσεις, την ανάλυση και τον σχεδιασμό, με σκοπό να προχωρήσουμε στην υλοποίηση της εφαρμογής μας. Παρακάτω θα αναφερθούμε στην πλατφόρμα που χρησιμοποίησαμε για να αναπτυχθεί η εφαρμογή μας, ενώ ο κώδικας της εφαρμογής θα παρουσιαστεί στις τελευταίες σελίδες της διπλωματικής εργασίας, στο σχετικό παράρτημα.

Κατά τη διάρκεια της ανάπτυξης της εφαρμογής χρησιμοποιήσαμε έναν υπολογιστή με λογισμικό Windows 10, με την τελευταία διαθέσιμη ενημέρωσή του. Στον συγκεκριμένο υπολογιστή εγκαταστήσαμε το περιβάλλον ανάπτυξης λογισμικού Android Studio, το οποίο είναι το επίσημο εργαλείο για δημιουργία και ανάπτυξη εφαρμογών σε συσκευές με Android λογισμικό, ενώ είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού βασισμένο πάνω στο IntelliJ IDEA της εταιρίας JetBrains. Η πρώτη εμφάνιση του Android Studio έγινε στις 16 Μαΐου 2013 στο συνέδριο Google I\O, ενώ η πρώτη σταθερή έκδοσή του έγινε τον Δεκέμβριο του 2014 με την έκδοση 1.0. Εμείς χρησιμοποιήσαμε τις εκδόσεις 4.1 και 4.2 για την ανάπτυξη της εφαρμογής μας, καθώς κατά τη διάρκεια της κυκλοφόρησε η νέα έκδοση του εργαλείου.

Εικόνα 3.4: Λογότυπο εφαρμογής Android Studio (Έκδοση 4.2)

Η τρέχουσα σταθερή έκδοση του Android Studio προσφέρει τις εξής δυνατότητες στον προγραμματιστή:

- Ένα ευέλικτο σύστημα κατασκευής βασισμένο στο Gradle
- Μία εικονική συσκευή Android, ή αλλιώς έναν Emulator, για γρήγορη εκτέλεση και εντοπισμό σφαλμάτων των εφαρμογών μέσα στο περιβάλλον του
- Ενσωματωμένη υποστήριξη για την πλατφόρμα Google Cloud, κάνοντας εύκολη την ενσωμάτωση τεχνολογιών όπως το Firebase Cloud Messaging, το Google App Engine και το Google Cloud Auto ML που χρησιμοποιήσαμε.
- Υποστήριξη για την ανάπτυξη εφαρμογών σε συσκευές Android πέρα από έξυπνα τηλέφωνα και tablet, όπως έξυπνα ρολόγια με λειτουργικό σύστημα Android Wear, έξυπνες τηλεοράσεις που χρησιμοποιούν το λειτουργικό σύστημα Android TV, αλλά και εφαρμογές για αυτοκίνητα συμβατά με το Android Auto.
- Έναν εμπλουτισμένο επεξεργαστή διάταξης (Layout Editor), ο οποίος επιτρέπει την εύκολη δημιουργία και σχεδίαση των οθονών με την τεχνική drag and drop, ενώ μπορεί να εμφανίζει την εφαρμογή σε διάφορα μεγέθη και διατάξεις της οθόνης.
- Υποστήριξη των γλωσσών προγραμματισμού Kotlin, Java και C++.
- Εύκολη σύνδεση και διαχείριση κώδικα στις πλατφόρμες Git και GitHub.

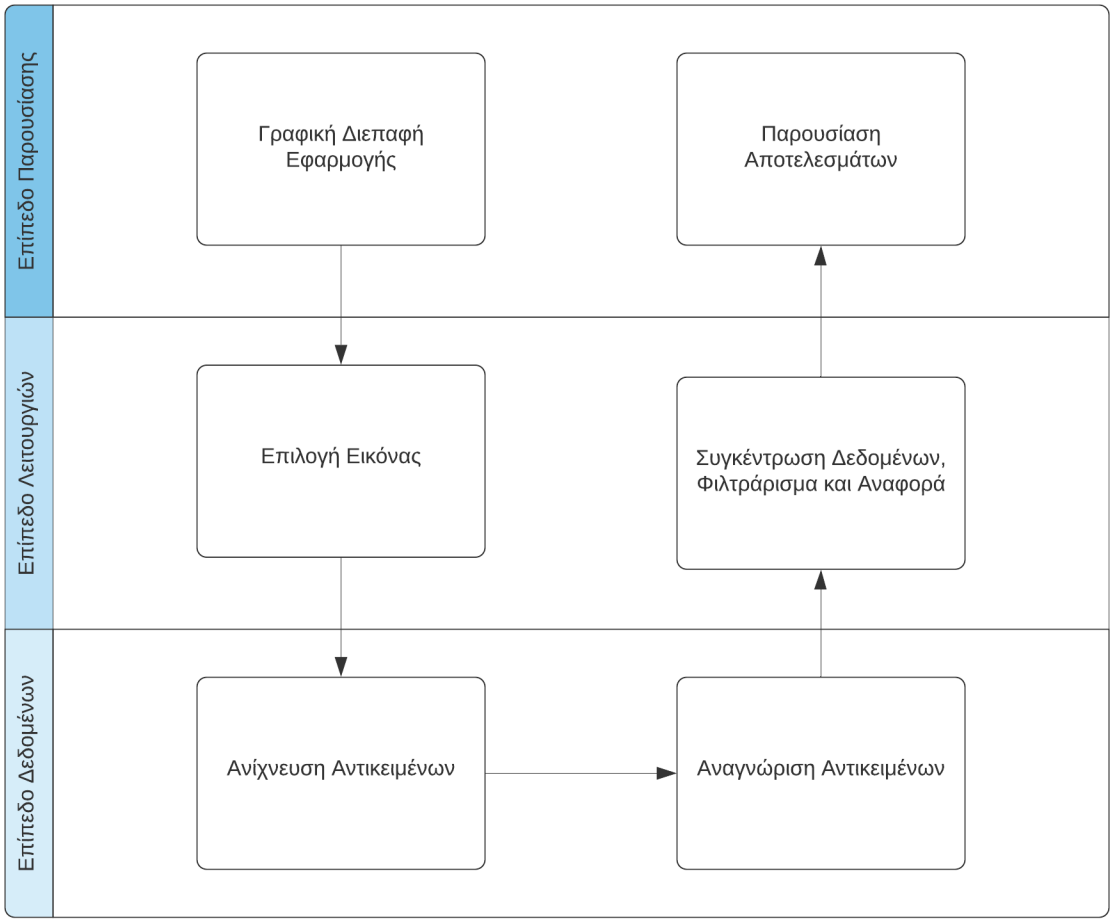
3.2.4 Έλεγχος & Διανομή

Σε αυτή την υποενότητα θα αναφερθούμε στην τέταρτη και τελευταία φάση της ανάπτυξης της εφαρμογής μας, που είναι ο έλεγχος και η διανομή της. Η φάση αυτή ακολουθεί πάντα τη φάση της υλοποίησης όταν χρησιμοποιούμε το μοντέλο του καταρράκτη, ενώ σε κάθε περίπτωση που εντοπιστούν πιθανές δυσλειτουργίες ή παραλείψεις, σημαντικές ή μη, επιστρέφουμε στη φάση της υλοποίησης, ή άμα κριθεί απαραίτητο και σε κάποια προγενέστερη φάση. Στη συγκεκριμένη εφαρμογή ο έλεγχος έγινε μέσω διαφόρων εικονικών συσκευών Android, μέσω του εργαλείου Android Studio, αλλά και φυσικών συσκευών που είχαμε στη διάθεσή μας. Ακόμα, σχετικά με τη διάθεση της εφαρμογής, δεν θα είναι

διαθέσιμη προς πώληση ή εγκατάσταση σε κάποια πλατφόρμα, για παράδειγμα Google Play Store, διότι ο αρχικός στόχος ήταν να σχεδιαστεί και να υλοποιηθεί στα πλαίσια της διπλωματικής εργασίας.

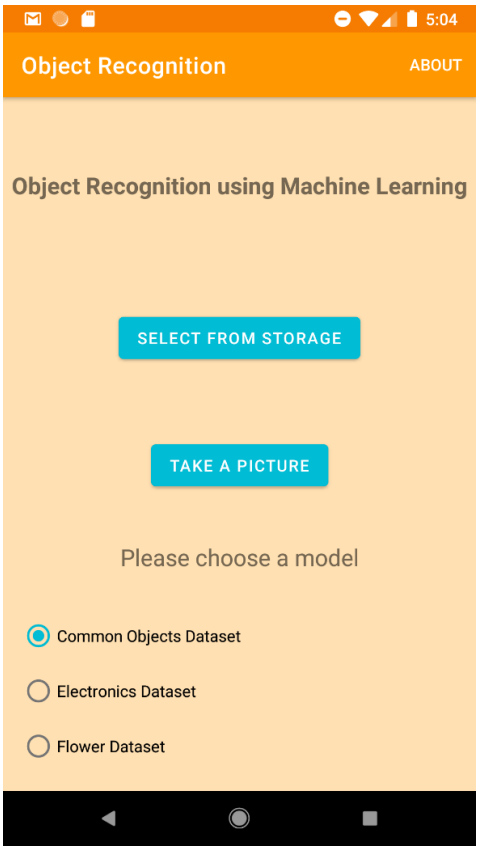
3.3 Λειτουργία Εφαρμογής

Στην ενότητα αυτή θα δώσουμε μια αναλυτική περιγραφή της εφαρμογής που δημιουργήσαμε στο πλαίσιο της διπλωματικής εργασίας, καθώς και θα δείξουμε τον τρόπο χρήσης της από τους πιθανούς χρήστες. Παρακάτω θα παρουσιαστούν οι δραστηριότητες της εφαρμογής μαζί με στιγμιότυπα οθόνης από την εφαρμογή, αλλά και οι συναρτήσεις που περιέχονται σε αυτές, όπως επίσης θα παρουσιαστούν και οι κλάσεις που βοηθούν στην υλοποίηση της εφαρμογής, ενώ η εικόνα 3.5 παρουσιάζει τη λογική αρχιτεκτονική της εφαρμογής μας, δηλαδή την διαδικασία που ακολουθείται για την ομαλή χρήση της εφαρμογής μας και στα επίπεδα που χωρίζονται.



Εικόνα 3.5: Λογική Αρχιτεκτονική Εφαρμογής

3.3.1 Δραστηριότητα MainActivity



Εικόνα 3.6: Οθόνη Δραστηριότητας MainActivity.

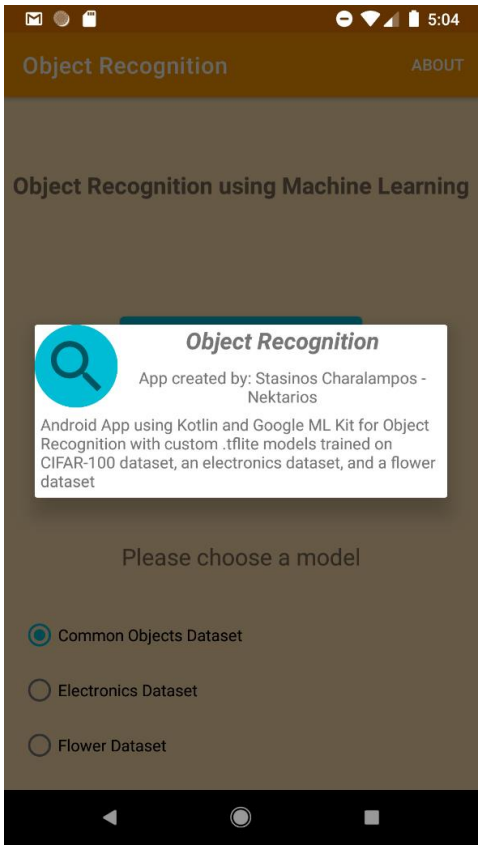
Η δραστηριότητα αυτή είναι η αρχική οθόνη της εφαρμογής μας. Αποτελείται από δύο μέρη όπως φαίνονται και στην εικόνα 3.6. Το πρώτο μέρος είναι η γραμμή εργαλείων που βρίσκεται στο πάνω μέρος της οθόνης, και το δεύτερο μέρος βρίσκεται στο κέντρο της οθόνης που χωρίζεται σε δύο τμήματα.

Στη γραμμή εργαλείων εμφανίζεται ο τίτλος της εφαρμογής και δίπλα του βρίσκεται ένα κουμπί με ονομασία «ABOUT», όπως φαίνεται στην εικόνα 3.7, το οποίο όταν πατηθεί μας εμφανίζει ένα παράθυρο με πληροφορίες για την εφαρμογή, όπως το εικονίδιο της εφαρμογής μας, το όνομα του δημιουργού, τον τίτλο της εφαρμογής , αλλά και μια σύντομη περιγραφή της εφαρμογής μας, ενώ εμφανίζεται και στην δραστηριότητα ChosenImage στην οποία θα αναφερθούμε παρακάτω.

Το δεύτερο μέρος, όπως αναφέραμε και πριν, χωρίζεται σε δύο τμήματα.

Το πρώτο και κύριο τμήμα αποτελείται από τα δύο κουμπιά με ονομασίες «SELECT FROM STORAGE» και «TAKE A PICTURE». Πιο αναλυτικά, όταν πατηθεί το κουμπί «SELECT FROM STORAGE» μεταφερόμαστε στην εφαρμογή της συσκευής που αποθηκεύονται και διαχειρίζονται οι φωτογραφίες, και έχουμε τη δυνατότητα να επιλέξουμε μία φωτογραφία από τον αποθηκευτικό χώρο της συσκευής για να γίνει αναγνώριση των αντικειμένων σε αυτή. Όταν επιλεχθεί το κουμπί «TAKE A PICTURE» μεταφερόμαστε στην εφαρμογή της συσκευής για την κάμερα, όπου στοχεύουμε τη κινητή συσκευή μας σε κάποιο αντικείμενο, που θέλουμε να αναγνωρίσουμε, και το τραβάμε μια φωτογραφία. Και οι δύο αυτές επιλογές έπειτα μας προωθούν στη δραστηριότητα ChosenImage.

Το δεύτερο κομμάτι αποτελείται από την ομάδα των τριών Radio Button με ονομασίες «Common Objects Dataset», «Electronics Dataset» και «Flower Dataset». Ανάλογα με την επιλογή του χρήστη θα χρησιμοποιηθεί το αντίστοιχο μοντέλο μηχανικής μάθησης για την αναγνώρηση των αντικειμένων. Η πρώτη επιλογή χρησιμοποιεί το μοντέλο που έχουμε εκπαιδεύσει με το CIFAR – 100 σύνολο δεδομένων, η δεύτερη είναι για την αναγνώριση ηλεκτρονικών ειδών, ενώ η τρίτη επιλογή χρησιμοποιεί το μοντέλο που έχει εκπαιδευτεί με τους πέντε διαφορετικούς τύπους λουλουδιών.



Εικόνα 3.7:Πληροφορίες Εφαρμογής.

Όταν ανοίγει η εφαρμογή επιλεγμένη είναι πάντα η πρώτη επιλογή, ενώ όταν αλλάζει ο προσανατολισμός της οθόνης η επιλογή του χρήστη μένει αποθηκευμένη και δεν αλλάζει, με χρήση της κλάσης SharedPreferences.

Ανεξαρτήτως της επιλογής του χρήστη για την εικόνα η επιλογή μοντέλου μηχανικής μάθησης μεταφέρεται στη δραστηριότητα ChosenImage με χρήση της κλάσης Intent.

Ο κώδικας της δραστηριότητας MainActivity αποτελείται από 13 συναρτήσεις, οι οποίες είναι είτε συναρτήσεις άλλων βιβλιοθηκών που έχουμε παρακάμψει, όπως οι συναρτήσεις onOptionsItemSelected() και η onActivityResult(), είτε συναρτήσεις που έχουμε δημιουργήσει εμείς για την ομαλή υλοποίηση της εφαρμογής, για παράδειγμα τις συναρτήσεις selectImage() και checkForBlankFile().

Οι συναρτήσεις που χρειάστηκε να παρακάμψουμε είναι οι παρακάτω:

- *onCreate() της κλάσης AppCompatActivity:* Είναι η βασική συνάρτηση που έχει κάθε Android εφαρμογή, και σε αυτή ορίζουμε τη διάταξη και τη γραμμή εργαλείων που εμφανίζονται στο χρήστη, και ρυθμίζουμε την κλάση SharedPreferences και τα Radio Button της δραστηριότητάς μας.
- *onCreateOptionsMenu() της κλάσης Activity:* Στη συνάρτηση αυτή ρυθμίζουμε το μενού που θέλουμε να εμφανιστεί στο χρήστη.
- *onOptionsItemSelected() της κλάσης Activity:* Στη συνάρτηση αυτή ρυθμίζουμε τις επιλογές που έχει το μενού που εμφανίζεται στο χρήστη.
- *onActivityResult() της κλάσης FragmentActivity:* Η συνάρτηση αυτή με την επιλογή φωτογραφίας από το χρήστη ή το τράβηγμα μιας φωτογραφίας, ελέγχει αν έγινε σωστά η διαδικασία επιλογής ή τραβήγματος της φωτογραφίας, και μετατρέπει τη φωτογραφία σε τύπο Bitmap. Εφόσον γίνει σωστά η μετατροπή, καλεί τη συνάρτηση startChosenImage().

Οι συναρτήσεις που δημιουργήσαμε είναι οι εξής:

- *radioButtonChecked():* Η συνάρτηση αυτή καλείται κάθε φορά που ο χρήστης επιλέγει το μοντέλο μηχανικής μάθησης που θέλει να χρησιμοποιήσει για την ανίχνευση και αναγνώριση των αντικειμένων μιας εικόνας. Η συνάρτηση ελέγχει ποιο radioButton επιλέχθηκε από το χρήστη και ανάλογα την επιλογή, αλλάζει την πληροφορία αυτή στη κλάση SharedPreferences.
- *setRadioButton():* Καλούμε τη συνάρτηση αυτή μέσω της onCreate() και ρυθμίζει ποιο radioButton θα είναι επιλεγμένο κατά την επιστροφή μας από άλλες δραστηριότητες στην δραστηριότητα MainActivity.
- *selectImage():* Η συνάρτηση αυτή καλείται όταν ο χρήστης πατήσει το κουμπί «SELECT FROM STORAGE». Αρχικά ελέγχουμε αν ο χρήστης έχει δώσει την άδεια του για να μπορεί η εφαρμογή να διαβάσει από τον αποθηκευτικό χώρο τις φωτογραφίες που υπάρχουν σε αυτόν. Άμα δεν έχει

δώσει την άδεια του την ζητάμε, ενώ στην περίπτωση που έχει εγκρίνει την άδεια, καλούμε τη συνάρτηση getPhoto().

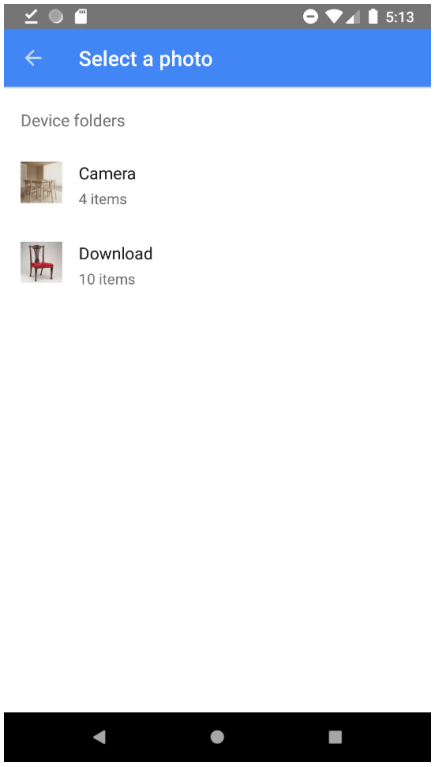
- *getPhoto():* Όταν καλούμε τη συνάρτηση αυτή, δημιουργούμε ένα Intent για να ξεκινήσουμε τη διαδικασία επιλογής φωτογραφίας από τον αποθηκευτικό χώρο και καλούμε τη συνάρτηση startActivityForResult().
- *openCamera():* Η συνάρτηση αυτή καλείται έπειτα από το πάτημα του κουμπιού «TAKE A PICTURE». Αρχικά αυτή η συνάρτηση ελέγχει αν ο χρήστης έχει δώσει την άδεια του για χρήση της κάμερας από την εφαρμογή μας. Στην περίπτωση που ο χρήστης έχει παραχωρήσει την άδεια αυτή, η εφαρμογή ελέγχει αν έχει δώσει ο χρήστης την άδειά του για την δημιουργία δεδομένων στον αποθηκευτικό χώρο. Εφόσον έχει παραχωρηθεί και αυτή η άδεια, καλείται η συνάρτηση dispatchPictureTakenIntent(). Στην περίπτωση που δεν δεχτεί ο χρήστης κάποια άδεια δεν γίνεται καμία νέα ενέργεια.
- *dispatchPictureTakenIntent():* Στη συνάρτηση αυτή δημιουργούμε ένα Intent για να ξεκινήσουμε τη δραστηριότητα της κάμερας, δημιουργούμε το αρχείο της εικόνας που θα τραβήξουμε με τη χρήση της κάμερας της συσκευής, προσθέτουμε τις λεπτομέρειες του αρχείου, αποθηκεύουμε τη θέση του αρχείου σε μία μεταβλητή και καλούμε τη συνάρτηση startActivityForResult().
- *checkForBlankFile():* Η συνάρτηση αυτή καλείται μέσα από τη συνάρτηση onActivityResult() που έχουμε παρακάμψει και ελέγχει, στη περίπτωση της επιλογής φωτογραφίας μέσω κάμερας, αν το αρχείο που δημιουργήσαμε έχει μηδενικό μέγεθος. Δημιουργήσαμε αυτή τη συνάρτηση διότι παρατηρήθηκε πως στην περίπτωση που δεν τραβάμε φωτογραφία και επιστρέφουμε στην αρχική οθόνη, δημιουργείται ένα κενό αρχείο.
- *getPath():* Κάλουμε τη συνάρτηση αυτή κατά τη διάρκεια της checkForBlankFile(). Δίνουμε ως είσοδο το Uri του αρχείου φωτογραφίας και μας επιστρέφει την ακριβή τοποθεσία του στον αποθηκευτικό χώρο της συσκευής.
- *startChosenImage():* Η συνάρτηση αυτή καλείται από τη συνάρτηση onActivityResult(), στην περίπτωση που η μεταβλητή τύπου Bitmap δεν είναι null. Μετατρέπουμε τη μεταβλητή σε τύπο ByteArray και την προσθέτουμε σαν πληροφορία σε ένα Intent που δημιουργούμε για να μεταφερθούμε στη δραστηριότητα ChosenImage. Επίσης προσθέτουμε σαν πληροφορία την επιλογή του χρήστη ως προς το μοντέλο μηχανικής μάθησης που θα χρησιμοποιήσει.

3.3.2 Επιλογή Φωτογραφίας Από Αποθηκευτικό Χώρο

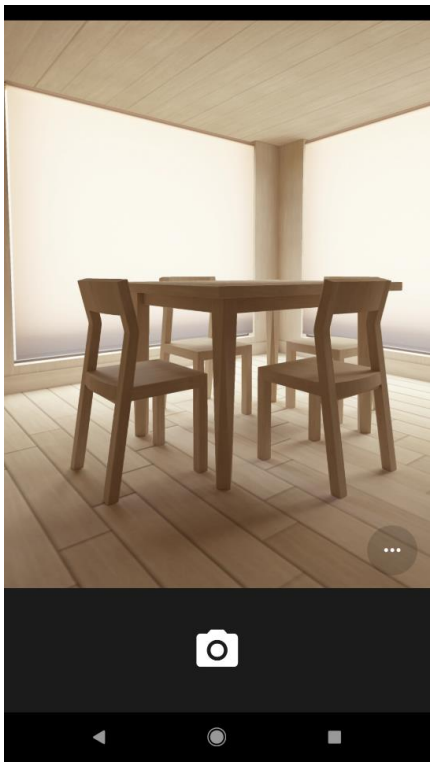
Στην περίπτωση που ο χρήστης επιλέξει το πρώτο κουμπί, όπως είδαμε στην εικόνα 3.6, με το όνομα «SELECT FROM STORAGE», η εφαρμογή θα μας μεταφέρει στην εφαρμογή της συσκευής που αποθηκεύονται οι φωτογραφίες. Συνήθως είναι η εφαρμογή για διαχείριση αρχείων του κινητού, ή η εφαρμογή Google Photos, ανάλογα με την επιλογή του χρήστη. Στην εικόνα 3.8 βλέπουμε την οθόνη της κινητής συσκευής κατά τη διαδικασία επιλογής φωτογραφίας από τον αποθηκευτικό χώρο. Αρχικά βλέπουμε τους φακέλους που έχουν δημιουργηθεί από εφαρμογές και από τον χρήστη, οι οποίοι περιέχουν φωτογραφίες. Έπειτα όταν ο χρήστης διαλέξει φάκελο, εμφανίζονται όλες οι φωτογραφίες που υπάρχουν μέσα σε αυτόν. Τέλος ο χρήστης επιλέγει την φωτογραφία στην οποία επιθυμεί να αναγνωρίσει κάποιο αντικείμενο μέσα σε αυτή. Εφόσον γίνει η επιλογή, η εφαρμογή για τις φωτογραφίες μεταφέρει την επιλεγμένη φωτογραφία στην εφαρμογή μας και στην δραστηριότητα ChosenImage.

3.3.3 Επιλογή Φωτογραφίας Μέσω Κάμερας

Όταν ο χρήστης στη δραστηριότητα MainActivity επιλέξει το κουμπί με όνομα «TAKE A PICTURE» η εφαρμογή μας μεταφέρει στην εφαρμογή της κινητής συσκευής για την κάμερα, όπου ο χρήστης καλείται να τραβήξει μια φωτογραφία το αντικείμενο που τον ενδιαφέρει. Στην εικόνα 3.9 βλέπουμε την οθόνη του κινητού όταν η εφαρμογή μας μάς μεταφέρει στην κάμερα. Επίσης βλέπουμε το ψηφιακό περιβάλλον που χρησιμοποιεί το Android Studio για χρήση και έλεγχο εφαρμογών που χρησιμοποιούν την εφαρμογή της κάμερας. Εφόσον ο χρήστης τραβήξει τη φωτογραφία που θέλει, εμφανίζονται δύο κουμπιά, τα οποία ανάλογα τη συσκευή έχουν και διαφορετικό εικαστικό. Το πρώτο κουμπί είναι ένα check mark (✔) που όταν πατηθεί δίνει την εντολή στην εφαρμογή της κάμερας να μεταφέρει τη συγκεκριμένη φωτογραφία στη δραστηριότητα ChosenImage της εφαρμογής μας. Το δεύτερο κουμπί είναι ένα X (✖), που όταν επιλεχθεί δίνει την εντολή στην εφαρμογή της κάμερας, να σβήσει τη φωτογραφία που μόλις τραβήχτηκε, καθώς δεν είναι επιθυμητή, και δίνει τη δυνατότητα στον χρήστη να τραβήξει μία πιο επιθυμητή φωτογραφία.



Εικόνα 3.8: Επιλογή Φωτογραφίας από Αποθ. Χώρο.



Εικόνα 3.9: Επιλογή Φωτογραφίας με χρήση Κάμερας.

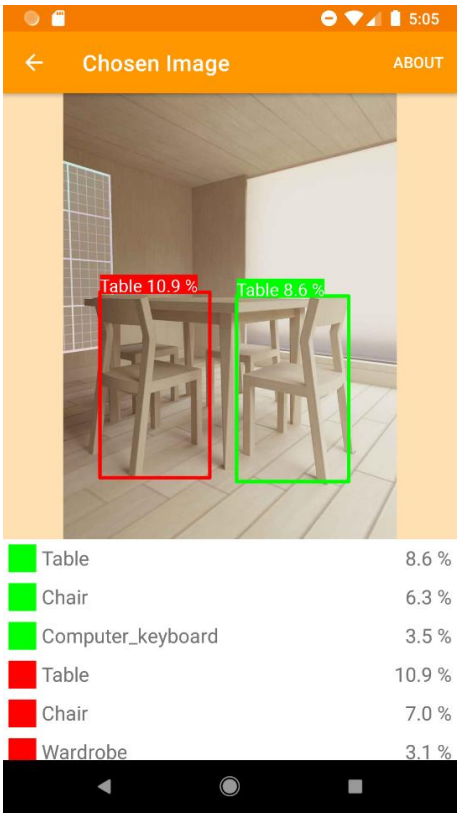
3.3.4 Δραστηριότητα ChosenImage

Σε αυτή την ενότητα θα αναφερθούμε στη δραστηριότητα ChosenImage, η οποία είναι και η πιο σημαντική δραστηριότητα της εφαρμογής μας. Όπως και η δραστηριότητα MainActivity, η ChosenImage αποτελείται από δύο μέρη, τη γραμμή εντολών και το κύριο μέρος που χωρίζεται σε δύο τμήματα.

Πιο αναλυτικά η γραμμή εντολών είναι παρόμοια με αυτή της δραστηριότητας MainActivity με δύο διαφορές. Η πρώτη διαφορά είναι πως έχουμε διαφορετικό όνομα στον τίτλο, που είναι το όνομα της δραστηριότητας, και η δεύτερη διαφορά είναι πως αριστερά του τίτλου βρίσκεται ένα κουμπί με σήμα ένα βελάκι. Αυτό το κουμπί όταν πατηθεί από το χρήστη τον επιστρέφει στην προηγούμενη δραστηριότητα, η οποία είναι η MainActivity. Αυτό γίνεται διότι έχουμε αναφέρει στο αρχείο μανιφέστο της εφαρμογής πως η δραστηριότητα ChosenImage έχει ως parent ή γονέα την δραστηριότητα MainActivity.

Το δεύτερο και κύριο μέρος, όπως αναφέραμε παραπάνω, αποτελείται από δύο τμήματα. Το πρώτο τμήμα περιέχει ένα ImageView όπου σε αυτό εμφανίζεται η εικόνα, όταν επιλεχθεί από το χρήστη και έχουν αναγνωριστεί αντικείμενα πάνω σε αυτή. Το δεύτερο τμήμα είναι ένα RecyclerView ή αλλιώς μια λίστα, όπου σε αυτή εμφανίζονται τα τρία καλύτερα αποτελέσματα ανά αντικείμενο μαζί με το ποσοστό σιγουριάς. Ακόμα εμφανίζουμε και το χρώμα του πλαισίου του αντικειμένου, έτσι ώστε σε περίπτωση αναγνώρισης δύο ή παραπάνω αντικειμένων, ο χρήστης να γνωρίζει ποιο αποτέλεσμα αναλογεί σε ποιο αντικείμενο. Στις εικόνες 3.10 και 3.11 βλέπουμε τα αποτελέσματα από την επιλογή φωτογραφιών και τη συνέχεια των εικόνων 3.8 και 3.9 αντίστοιχα.

Εικόνα 3.10: Αναγνώριση Φωτογραφίας από Αποθ. Χώρο



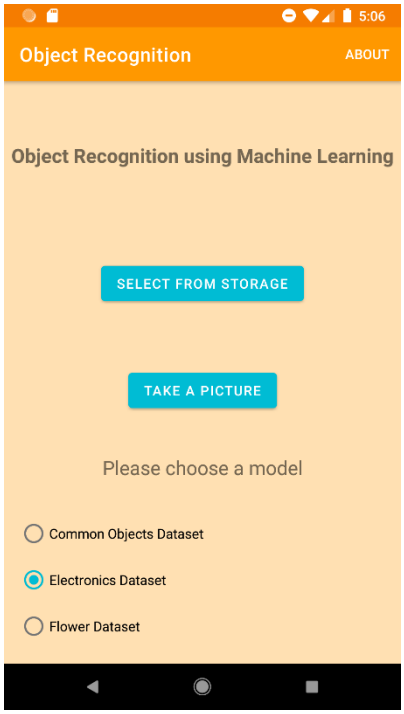
Εικόνα 3.11: Αναγνώριση Φωτογραφίας μέσω Κάμερας

Στην πρώτη εικόνα βλέπουμε τη φωτογραφία μιας καρέκλας, η οποία έχει αναγνωριστεί από το μοντέλο μηχανικής μάθησης που επιλέξαμε, ως καρέκλα με ποσοστό σιγουριάς 43.8% ενώ η δεύτερη πιο πιθανή επιλογή είναι το κρεβάτι με 3.5%. Στην δεύτερη εικόνα βλέπουμε και στις δύο περιπτώσεις να είναι λάθος η πρώτη εκτίμηση και σωστή η δεύτερη. Πιο αναλυτικά το πρώτο αντικείμενο έχει αναγνωριστεί πρώτα ως τραπέζι με ποσοστό 8.6% και έπειτα ως καρέκλα με ποσοστό 6.3%, ενώ το δεύτερο αντικείμενο είναι αναγνωρισμένο πρώτα ως τραπέζι με ποσοστό 10.9% και η δεύτερη αναγνώριση του είναι ως καρέκλα είναι με ποσοστό 7.0%. Αυτό γίνεται διότι το μοντέλο μηχανικής μάθησης δεν καταλαβαίνει πλήρως τη διαφορά ενός τραπεζιού με μιας καρέκλας, για αυτό και δεν έχουμε παρόμοια αποτελέσματα και στην εικόνα 3.9. Ένας τρόπος για επίλυση του προβλήματος είναι η περαιτέρω εκπαίδευση του μοντέλου μηχανικής μάθησης, ενώ ένας άλλος τρόπος είναι η αύξηση των δεδομένων εκπαίδευσης για τις κατηγορίες τραπέζι και καρέκλα, με σκοπό να εξοικιωθεί το μοντέλο μηχανικής μάθησης με τις διαφορές τους. Τα ποσοστά που εμφανίζονται είναι το αποτέλεσμα της σιγουριάς του μοντέλου μηχανικής μάθησης και κάθε κατηγορία που υπάρχει στο μοντέλο έχει από μία τιμή. Το συγκεκριμένο μοντέλο αποτελείται από εκατό κατηγορίες και η μικρότερη τιμή σιγουριάς που παρατηρήσαμε κατά τη διάρκεια του ελέγχου της εφαρμογής ήταν 32×10^{-8} %. Επίσης, όπως βλέπουμε και στις παραπάνω εικόνες, όταν γίνει η αναγνώριση αντικειμένων εμφανίζεται ένα πλαίσιο γύρω από το αντικείμενο και αναφέρεται το όνομα και το ποσοστό της καλύτερης επιλογής.

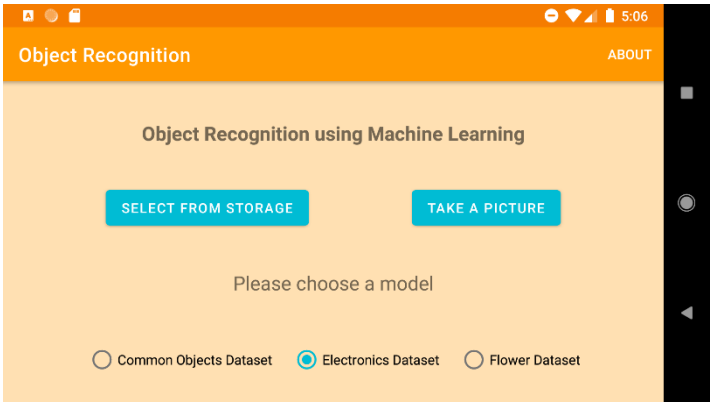
Στην περίπτωση που το πρόγραμμα δεν αναγνωρίσει κάποιο αντικείμενο στην επιλεγμένη εικόνα στον χρήστη φαίνεται η οθόνη της εικόνας 3.12, χωρίς κάποια επισήμανση.

Εικόνα 3.12: Οθόνη ChosenImage έπειτα από μη αναγνώριση αντικειμένου

Τέλος, η εφαρμογή μας υποστηρίζει και τα δύο είδη προσανατολισμών των κινητών συσκευών, δηλαδή τον οριζόντιο ή landscape προσανατολισμό και τον κάθετο ή portrait. Στις παρακάτω εικόνες εμφανίζονται και τα δύο είδη προσανατολισμού, στις δραστηριότητες MainActivity και ChosenImage.

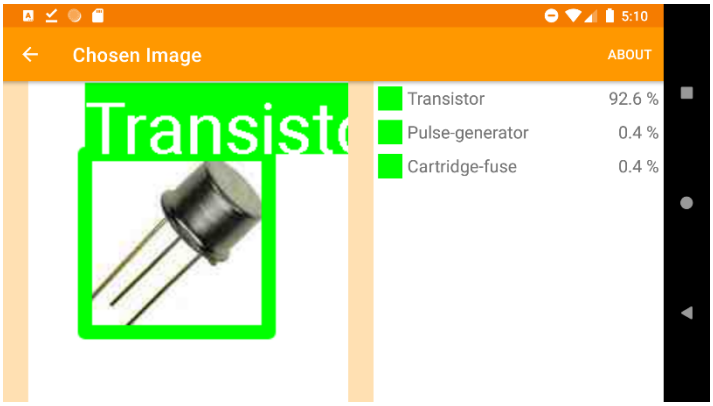


Εικόνα 3.13: Αρχική Οθόνη σε Κάθετο Προσανατολισμό.



Εικόνα 3.14: Αρχική Οθόνη σε Οριζόντιο Προσανατολισμό.

Εικόνα 3.15: Οθόνη ChosenImage σε Κάθετο Προσανατολισμό.



Εικόνα 3.16: Οθόνη ChosenImage σε Οριζόντιο Προσανατολισμό.

Η δραστηριότητα ChosenImage αποτελείται από πέντε συναρτήσεις, όπου τις τέσσερις από αυτές τις έχουμε παρακάμψει και μία την έχουμε δημιουργήσει. Οι συναρτήσεις αυτές είναι οι εξής:

- *onCreate() της κλάσης AppCompatActivity:* Η συνάρτηση καλείται κατά τη δημιουργίας της δραστηριότητας ChosenImage και ορίζει τη διάταξη και τη γραμμή εργαλείων που εμφανίζονται. Έπειτα συνδέεται με το ViewModel τη δραστηριότητας και δέχεται τις πληροφορίες που στείλαμε από τη δραστηριότητα MainActivity. Έπειτα μετατρέπει το ByteArray σε Bitmap και θέτει τον αντάπτορα του RecyclerView της δραστηριότητας. Τέλος ορίζει παρατηρητές στις λίστες του ViewModel και σε περίπτωση που το savedInstanceState είναι null καλεί τη συνάρτηση του ViewModel recognizeObjects(), για να ξεκινήσει η ανίχνευση και η αναγνώριση των αντικειμένων της φωτογραφίας.
- *onCreateOptionsMenu() της κλάσης Activity:* Στη συνάρτηση αυτή ρυθμίζουμε το μενού που θέλουμε να εμφανιστεί στον χρήστη.
- *onOptionsItemSelected() της κλάσης Activity:* Στη συνάρτηση αυτή ρυθμίζουμε τις δυνατότητες του μενού της εφαρμογής που εμφανίζεται στο χρήστη.
- *convert():* Η συνάρτηση αυτή είναι η μόνη συνάρτηση που δημιουργήσαμε από την αρχή και τη χρησιμοποιούμε για να μετατρέψουμε την εικόνα Bitmap που δέχεται ως είσοδο σε μορφή ARGB_8888.
- *onSaveInstanceState() της κλάσης AppCompatActivity:* Η συνάρτηση αυτή καλείται κάθε φορά που η συσκευή αλλάζει προσανατολισμό και προσθέτουμε στο Bundle το ByteArray που μεταφέραμε από τη δραστηριότητα MainActivity.

3.3.5 Κλάσεις – Βοηθοί MVVM

Στην ενότητα αυτή θα αναφερθούμε στις κλάσεις της εφαρμογής μας που βοηθούν στην υλοποίηση της εφαρμογής με αρχιτεκτονική MVVM. Στην εφαρμογή μας η κλάση που δρα ως ViewModel έχει σαν όνομα ItemRecognitionViewModel, ενώ η κλάση που είναι το Model είναι η κλάση με όνομα ObjectDetection. Η δραστηριότητα ChosenImage, που αναφέραμε παραπάνω, είναι το View της αρχιτεκτονικής μας.

Η κλάση του Model, αποτελείται από τις τρεις παρακάτω συναρτήσεις:

- *locateObjects:* Η συνάρτηση αυτή είναι η πιο σημαντική συνάρτηση ολόκληρης της εφαρμογής μας καθώς σε αυτή γίνεται η χρήση των μοντέλων μηχανικής μάθησης για να αναγνωριστούν τα αντικείμενα στις εικόνες. Η συνάρτηση δέχεται ως είσοδο την εικόνα σε μορφή Bitmap και έναν ακέραιο αριθμό που καθορίζει πιο μοντέλο μηχανικής μάθησης θα χρησιμοποιηθεί. Με τη χρήση του Google Machine Learning Kit έχουμε στη διάθεσή μας έναν Object Detector, ή αλλιώς Ανιχνευτή Αντικειμένων, στον οποίο έχουμε προσθέσει Listeners, δηλαδή Ακροατές, στις περιπτώσεις των επιτυχημένων και αποτυχημένων προσπαθειών. Στην περίπτωση της επιτυχημένης προσπάθειας ο Object Detector επιστρέφει δύο μεταβλητές, την ταυτότητα και το πλαίσιο που βρίσκεται το αντικείμενο. Με την χρήση του πλαισίου καλούμε τη συνάρτηση cropObject, που θα αναφέρουμε παρακάτω, και δημιουργούμε τον Image Labeler, ή αλλιώς Τρόπος Επισήμανσης Εικόνας. Στον Image Labeler δίνουμε την εικόνα που πήραμε από την έξοδο της cropObject και έχουμε προσθέσει Listeners, όπως και στον Object Detector. Στην περίπτωση της επιτυχημένης προσπάθειας ο Image Labeler επιστρέφει τρία αποτελέσματα ανά αντικείμενο, τα

οποία εμείς βάζουμε σε ένα πίνακα, έτσι ώστε μετά να εμφανιστούν τα αποτελέσματα όπως είπαμε στην ενότητα ChosenImage. Τέλος καλείται η συνάρτηση paintAround την οποία και θα αναλύσουμε παρακάτω.

- *cropObject:* Σε αυτή τη συνάρτηση δίνουμε ως είσοδο την εικόνα που έχει επιλέξει ο χρήστης σε μορφή Bitmap, αλλά και τις συντεταγμένες πλαισίου που μας επέστρεψε ο Object Detector. Έπειτα η συνάρτηση κόβει την εικόνα στις συντεταγμένες που έλαβε ως είσοδο και επιστρέφει την κομμένη εικόνα στη συνάρτηση locateObjects. Η συνάρτηση αυτή βοηθά στην πολλαπλή αναγνώριση αντικειμένων ανά φωτογραφία καθώς δημιουργεί μία καινούργια φωτογραφία για κάθε αναγνωρισμένο αντικείμενο

- *paintAround:* Η συνάρτηση αυτή χρησιμοποιείται για την πλαισίωση και εμφάνιση των αντικειμένων στην φωτογραφία που έχει επιλέξει ο χρήστης. Δέχεται ως είσοδο την εικόνα που πήραμε από τη συνάρτηση cropObject, το πλαίσιο συντεταγμένων του αντικειμένου, τα στοιχεία της πρώτης πρόβλεψης, όπως το όνομα και το ποσοστό σιγουριάς, και έναν ακέραιο αριθμό που αντιστοιχεί στο χρώμα που θα έχει το πλαίσιο στην εικόνα. Αρχικά ρυθμίζουμε το πάχος του πλαισίου σε σχέση με την ανάλυση της φωτογραφίας, ενώ έπειτα ρυθμίζουμε και τις συντεταγμένες για την εμφάνιση του κειμένου. Τέλος με χρήση της κλάσης Canvas δημιουργούμε το πλαίσιο πάνω στην εικόνα και την επιστρέφουμε στη συνάρτηση locateObjects.

Η κλάση του ViewModel, αποτελείται από τη συνάρτηση recognizeObjects, η οποία δρα ως μεσάζοντας και όποτε καλείται από τη δραστηριότητα ChosenImage αυτή καλεί τη συνάρτηση locateObjects του Model. Ακόμα η κλάση ItemRecognitionViewModel περιέχει δύο λίστες τύπου MutableLiveData, μία λίστα για τη φωτογραφία και μία για τα αναγνωρισμένα αντικείμενα. Τέλος το αρχείο του ViewModel περιέχει και μία ακόμα κλάση, η οποία είναι μία κλάση δεδομένων, και ορίζει τις μεταβλητές του ItemRecognition. Την κλάση αυτή τη χρησιμοποιούμε για να αποθηκεύσουμε τα αποτελέσματα της αναγνώρισης των αντικειμένων, αλλά και καθιστά εύκολη τη μεταφορά των δεδομένων από το Model της αρχιτεκτονικής στο View. Η κλάση αποτελείται από πέντε μεταβλητές, οι οποίες είναι οι εξής:

- id: Περιέχει έναν ακέραιο αριθμό και χρησιμοποιούμε αυτή τη μεταβλητή ως τρόπο ανίχνευσης του αντικειμένου για το οποίο αναφερόμαστε. Για παράδειγμα στην εικόνα 3.10 που είχαμε δύο αντικείμενα σε αυτή είχαμε δύο διαφορετικά id, ένα για κάθε αντικείμενο.
- label: Η μεταβλητή αυτή είναι της μορφής String, και δέχεται ως τιμή το όνομα της κατηγορίας που προέβλεψε το μοντέλο μηχανικής μάθησης για το συγκεκριμένο αντικείμενο
- probability: Η μεταβλητή αυτή περιέχει το ποσοστό σιγουριάς που προέβλεψε το μοντέλο μηχανικής μάθησης που χρησιμοποιήσαμε για το συγκεκριμένο αντικείμενο με το αντίστοιχο label.
- location: Η μεταβλητή location είναι του τύπου Rect και περιέχει τις συντεταγμένες πλαισίου του αντικειμένου, όπως αυτές επιστράφηκαν από τον Object Detector της συνάρτησης locateObjects.
- color: Περιέχει έναν ακέραιο αριθμό που αντιστοιχεί στο χρώμα του πλαισίου του αντικειμένου στην εικόνα που εμφανίζεται στη δραστηριότητα ChosenImage, εφόσον γίνει ανίχνευση και αναγνώριση των αντικειμένων της.

3.3.6 Λοιπές κλάσεις

Στην υποενότητα αυτή θα αναφερθούμε στις υπόλοιπες κλάσεις που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής τις BaseActivity, ItemRecognitionViewHolder και ItemRecognitionAdapter. Πιο αναλυτικά:

- *BaseActivity:*

  Η κλάση αυτή περιέχει δύο εσωτερικές συναρτήσεις τις activateToolbar() και showAboutDialog(). Η πρώτη συνάρτηση ρυθμίζει τη γραμμή εργαλείων για τις δραστηριότητές μας, ενώ η δεύτερη περιέχει τον κώδικα για την εμφάνιση των πληροφοριών της εφαρμογής με χρήση του κουμπιού με όνομα «ABOUT» που αναφέραμε και στην ενότητα 3.3.1.

- *ItemRecognitionViewHolder:*

  Σε αυτή την κλάση ορίζονται οι μεταβλητές που θα εμφανιστούν στο RecyclerView που αναφέραμε στην ενότητα 3.3.4, και αποτελούνται από το χρώμα του πλαισίου του αντικειμένου, το όνομα της κατηγορίας που ανήκει και το ποσοστό σιγουριάς που υπολόγισε το μοντέλο μηχανικής μάθησης που χρησιμοποιήσαμε.

- *ItemRecognitionAdapter:*

  Η κλάση αυτή είναι βοηθητική για το RecyclerView και ο σκοπός της είναι να παίρνει τις πληροφορίες των αντικειμένων και να τις εμφανίζει στην οθόνη μέσω του RecyclerView. Αποτελείται από τέσσερις συναρτήσεις, όπου οι τρεις από αυτές έχουν παρακαμφθεί και η τέταρτη έχει δημιουργηθεί από εμάς. Οι συναρτήσεις αυτές είναι οι εξής:
  - onCreateViewHolder() της κλάσης Adapter: Η συνάρτηση αυτή ορίζει τη διάταξη που θα έχουν τα δεδομένα μέσα στο RecyclerView της δραστηριότητας ChosenImage.
  - getItemCount() της κλάσης Adapter: Όταν καλείται η συνάρτηση επιστρέφει το μέγεθος της λίστας.
  - onBindViewHolder() της κλάσης Adapter: Η συνάρτηση αυτή εμφανίζει τα δεδομένα στο RecyclerView της δραστηριότητας ChosenImage. Αν το itemList του Adapter είναι άδειο τότε η συνάρτηση κρύβει το RecyclerView από τη δραστηριότητα, ενώ όταν περιέχει τιμές, εμφανίζει τα δεδομένα ανά αντικείμενο και ρυθμίζει το χρώμα του πλαισίου που θα εμφανιστεί, το όνομα της κατηγορίας και το ποσοστό σιγουριάς που προέβλεψε το μοντέλο μηχανικής μάθησης που επέλεξε ο χρήστης.
  - setRecognitionList(): Η συνάρτηση δέχεται ως είσοδο μία λίστα τύπου ItemRecognition την οποία ορίζει ως τη λίστα που θα εμφανίζεται στο RecyclerView και ενημερώνει τον Adapter για να εμφανίσει τα νέα αποτελέσματα.

Κεφάλαιο 4ο: Training Datasets

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούμε στα training datasets, ή αλλιώς στα δεδομένα – φωτογραφίες που χρησιμοποιήσαμε για την εκπαίδευση των μοντέλων μηχανικής μάθησης, θα αναφερθούμε στις τεχνικές και θα αναλύσουμε τα εργαλεία που χρησιμοποιήσαμε.

4.2 Τα σύνολα δεδομένων που χρησιμοποιήθηκαν

Για την εφαρμογή που δημιουργήθηκε στα πλαίσια της διπλωματικής εργασίας χρησιμοποιήθηκαν τρία διαφορετικά μοντέλα μηχανικής μάθησης. Το κάθε ένα μοντέλο εκπαιδεύτηκε με ένα ξεχωριστό σύνολο δεδομένων.

Στην περίπτωση του μοντέλου μηχανικής μάθησης που χρησιμοποιεί η εφαρμογή μας για την ανίχνευση και αναγνώριση συνηθισμένων αντικειμένων χρησιμοποιήσαμε το CIFAR – 100 σύνολο δεδομένων για την εκπαίδευσή του. Το CIFAR – 100, δημιουργήθηκε από τους Alex Krizhevsky, Vinod Nair και Geoffrey Hinton, και είναι ένα σύνολο δεδομένων που αποτελείται από 60.000 φωτογραφίες, οι οποίες είναι χωρισμένες σε 100 κλάσεις. Κάθε εικόνα είναι έγχρωμη και έχει ανάλυση 32 x 32, ενώ κάθε κλάση περιέχει 500 φωτογραφίες για την προπόνηση ενός μοντέλου μηχανικής μάθησης και 100 φωτογραφίες για τον έλεγχο του μοντέλου. Το CIFAR – 100, επίσης έχει 20 υπερκλάσεις, που έχουν χωριστεί οι 100 κλάσεις που αναφέραμε παραπάνω, όπως φαίνεται στον πίνακα 4.1. Με έντονα γράμματα είναι οι υπερκλάσεις, ενώ με κανονικά φαίνονται οι κλάσεις που περιέχονται στην εκάστοτε υπερκλάση.

Υπερκλάση	Κλάσεις
Θαλάσσια Θηλαστικά	Κάστορας, Δελφίνι, Βίδρα, Φώκια, Φάλαινα
Ψάρια	Ψάρια Ενυδρείου, Πλατύψαρο, Σαλάχι, Καρχαρίας, Πέστροφα
Λουλούδια	Ορχιδέες, Παπαρούνες, Τριαντάφυλλα, Ήλιοι, Τουλίπες
Δοχεία Τροφίμων	Μπουκάλια, Μπωλ, Κουτάκια, Κούπες, Πιάτα
Φρούτα και Λαχανικά	Μήλα, Μανιτάρια, Πορτοκάλια, Αχλάδια, Πιπεριές
Ηλεκτρικές Συσκευές	Ρολόι, Πληκτρολόγιο, Λάμπα, Τηλέφωνο, Τηλεόραση
Έπιπλα	Κρεβάτι, Καρέκλα, Καναπές, Τραπέζι, Ντουλάπα
Έντομα	Μέλισσα, Σκαθάρι, Πεταλούδα, Κάμπια, Κατσαρίδα
Σαρκοφάγα Ζώα	Αρκούδα, Λεοπάρδαλη, Λιοντάρι, Τίγρης, Λύκος
Τεχνητά Υπαίθρια Πράγματα	Γέφυρα, Κάστρο, Σπίτι, Δρόμος, Ουρανοξύστης
Φυσικές Υπαίθριες Σκηνές	Σύννεφο, Δάσος, Βουνό, Πεδιάδα, Θάλασσα
Παμφάγα και Φυτοφάγα Ζώα	Καμήλα, Βοοειδή, Χιμπατζής, Ελέφαντας, Καγκουρό
Θηλαστικά Μεσαίου Μεγέθους	Αλεπού, Ακανθόχοιρος, Πόσουμ, Ρακούν, Κουνάβι
Ασπόνδυλα Εκτός Εντόμων	Κάβουρας, Αστακός, Σαλιγκάρι, Αράχνη, Σκουλήκι
Άνθρωποι	Μωρό, Αγόρι, Κορίτσι, Άντρας, Γυναίκα
Ερπετά	Κροκόδειλος, Δεινόσαυρος, Σαύρα, Φίδι, Χελώνα

Θηλαστικά Μικρού Μεγέθους	Χάμστερ, Ποντίκι, Κουνέλι, Μέγαιρα, Σκίουρος
Δέντρα	Σφένδαμος, Δρυς, Φοίνικας, Πεύκο, Ιτιά
Οχήματα 1	Ποδήλατο, Λεωφορείο, Μηχανή, Φορτηγό, Τρένο
Οχήματα 2	Μηχανή Γκαζόν, Πύραυλος, Τραμ, Τανκς, Τρακτέρ

Πίνακας 4.1: Υπερκλάσεις και Κλάσεις CIFAR – 100 Συνόλου Δεδομένων.

Για την εύρεση συνόλων δεδομένων για τα μοντέλα μηχανικής μάθησης για ηλεκτρονικά είδη, αλλά και για το μοντέλο με τύπους λουλουδιών, χρησιμοποιήθηκε η πλατφόρμα Kaggle, η οποία είναι μία διαδικτυακή κοινότητα που αποτελείται από επιστήμονες δεδομένων και χρήστες τεχνολογιών μηχανικής μάθησης. Πιο συγκεκριμένα για το μοντέλο για αναγνώριση ηλεκτρονικών ειδών χρησιμοποιήθηκε ένα σύνολο δεδομένων αναρτημένο από τον Sunim Acharya, με συνολικά 11.000 φωτογραφίες διαφόρων αναλύσεων χωρισμένες σε 36 κατηγορίες. Οι κατηγορίες που είναι χωρισμένες οι φωτογραφίες και συνεπώς χρησιμοποιεί το μοντέλο μηχανικής μάθησης που εκπαιδεύσαμε φαίνονται στον πίνακα 4.2.

Επαγώγιμο	Εξασθενητής	Πυκνωτής Παράκαμψης	Φυσίγγια Ασφαλειών	Καλώδιο με Κροκοδειλάκια	Ηλεκτρικό Ρελέ
Ηλεκτρολυτικός Πυκνωτής	Λεπτό Σύρμα	Ψύκτρα	Επαγωγικό Πηνίο	Ολοκληρωμένο Μικροκύκλωμα	Καλώδιο Βραχυκυκλωτήρα
Διπολικό Τρανζίστορ	Φως τύπου LED	Ολοκληρωμένο Κύκλωμα με Φως	Limiter & Clipper	Τοπικός Ταλαντωτής	Τσιπ Μνήμης
Μικροτσίπ	Μικροεπεξεργαστής	Πολυπλέκτης	Παντοκατευθυντική Κεραία	Τρανζίστορ PnP	Διαιρέτης Τάσης
Ποτενσιόμετρο	Γεννήτρια Σφυγμού	Ηλεκτρονόμος	Ρυθμιστής Ηλεκτρικού Ρεύματος	Ημιαγωγός	Δίοδος Ημιαγωγού
Παραδιακλάδωση	Σωληνοειδές	Σταθεροποιητής	Μετασχηματιστής προς τα κάτω	Μετασχηματιστής προς τα πάνω	Τρανζίστορ

Πίνακας 4.2: Κλάσεις Συνόλου Δεδομένων για Ηλεκτρονικά Είδη.

Τέλος, το σύνολο δεδομένων για λουλούδια, το προμηθευτήκαμε από το Kaggle όπως το προηγούμενο σύνολο δεδομένων που αναφέραμε, αναρτήθηκε από τον Alexander Mamaev και αποτελείται από 4242 φωτογραφίες λουλουδιών. Οι φωτογραφίες αυτές μαζεύτηκαν μέσω των ιστοσελίδων Flickr, Google Images και Yandex Images. Τα δεδομένα χωρίζονται σε πέντε φακέλους – κλάσεις, ενώ κάθε φάκελος περιέχει τουλάχιστον 700 φωτογραφίες. Επίσης οι φωτογραφίες είναι διαφόρων μεγεθών και έχουν ανάλυση 320 x 240 pixels. Οι κατηγορίες που έχουν χωριστεί οι φωτογραφίες εμφανίζονται στον πίνακα 4.3

Μαργαρίτα	Πικραλίδα	Τριαντάφυλλο	Ήλιος	Τουλίπα

Πίνακας 4.3: Κλάσεις Συνόλου Δεδομένων για Λουλούδια

4.3 Εργαλεία που Χρησιμοποιήθηκαν

Για την εκπαίδευση των μοντέλων μηχανικής μάθησης χρησιμοποιήσαμε κάποια εργαλεία. Όπως αναφέραμε στην προηγούμενη ενότητα για την εύρεση συνόλων δεδομένων για τα δύο μοντέλα που φτιάξαμε χρησιμοποιήσαμε την πλατφόρμα Kaggle. Η πλατφόρμα αυτή δημιουργήθηκε το 2010 από τους Anthony Goldbloom και Ben Hammer, ενώ η αρχική της λειτουργία ήταν να διοργανώνει διαγωνισμούς μηχανικής μάθησης. Το 2017 η Google αγόρασε την πλατφόρμα και πλέον προσφέρει περισσότερες λειτουργίες όπως μια δημόσια πλατφόρμα δεδομένων, ένα ψηφιακό πάγκο εργασίας βασισμένο σε Cloud τεχνολογίες για χρήση με την επιστήμη δεδομένων και εκπαίδευση στον τομέα της Τεχνητής Νοημοσύνης.

Εικόνα 4.1: Λογότυπο Πλατφόρμας Kaggle.

Όπως αναφέραμε στην ενότητα 1.4 για την εκπαίδευση των μοντέλων μηχανικής μάθησης χρησιμοποιήσαμε την πλατφόρμα Google Cloud Platform, και πιο συγκεκριμένα το Google Cloud Auto ML Vision που προσφέρει η πλατφόρμα αυτή.

Τέλος χρησιμοποιήθηκε η open – source βιβλιοθήκη για μηχανική μάθηση TensorFlow και πιο συγκεκριμένα στη μορφή TensorFlow Lite. Η βιβλιοθήκη αυτή χρησιμοποιείται για ένα ευρύ φάσμα εργασιών, αλλά η κύρια χρήση της είναι στην εκπαίδευση και την εξαγωγή συμπερασμάτων σε βαθιά νευρωνικά δίκτυα (deep neural networks). Το Tensorflow είναι μια βιβλιοθήκη που αποτελείται από συμβολικά μαθηματικά και βασίζεται στη ροή δεδομένων και στον διαφοροποιήσιμο προγραμματισμό. Χρησιμοποιείται τόσο για έρευνα όσο και για παραγωγή στην εταιρεία Google. Αναπτύχθηκε το 2015 από την ομάδα Google Brain για εσωτερική χρήση στο Google, ενώ κυκλοφόρησε την ίδια χρονιά με την άδεια Apache License 2.0.

Η πρώτη έκδοση του TensorFlow λεγόταν DistBelief και είχε δημιουργηθεί το 2011 από την ίδια ομάδα, ενώ η πρώτη σταθερή έκδοση, η έκδοση 1.0.0 του Tensorflow, έγινε διαθέσιμη τον Φεβρουάριο του 2017. Είναι διαθέσιμο στις πλατφόρμες Linux, macOS και Windows για σταθερούς και φορητούς υπολογιστές, ενώ επίσης είναι διαθέσιμο για κινητές συσκευές Android και iOS.

Το TensorFlow Lite που χρησιμοποιήσαμε για τα μοντέλα μηχανικής μάθησης που εκπαιδεύσαμε ανακοινώθηκε τον Μάιο του 2017 από την Google και κυκλοφόρησε σε δοκιμαστική έκδοση μετά από σχεδόν δύο χρόνια, τον Ιανουάριο του 2019. Η διαφορά των TensorFlow και TensorFlow Lite είναι πως το πρώτο χρησιμοποιεί για μορφή σειριοποίησης δεδομένων για μοντέλα δικτύου την open – source βιβλιοθήκη Protobuf (Protocol Buffers), ενώ το δεύτερο χρησιμοποιεί την δωρεάν βιβλιοθήκη FlatBuffers.

Εικόνα 4.2: Λογότυπο Βιβλιοθήκης TensorFlow Lite.

4.4 Διαδικασία Εκπαίδευσης Μοντέλων Μηχανικής Μάθησης

Σε αυτή την ενότητα θα αναφερθούμε και θα αναλύσουμε τα βήματα που ακολουθήσαμε για την εκπαίδευση των μοντέλων μηχανικής μάθησης. Για τη σωστή εκπαίδευση των μοντέλων ακολουθήσαμε πιστά τις οδηγίες και την τεκμηρίωση, ή πιο σωστά το documentation του Google Cloud Auto ML Vision. Η πλατφόρμα αυτή αποτελεί ένα σύστημα επιβλεπόμενης μάθησης καθώς κάθε παράδειγμα ή φωτογραφία, όπως στη δική μας περίπτωση, αποτελείται από ένα σύνολο δεδομένων και μία επιθυμητή τιμή εξόδου.

Αρχικά μεταφέραμε τις φωτογραφίες σε ένα storage bucket (κουβά αποθήκευσης), όπως μας πρότεινε η πλατφόρμα, και με χρήση κώδικα δημιουργήσαμε ένα αρχείο .csv όπου περιέχει τις διευθύνσεις των φωτογραφιών στη πλατφόρμα Google Cloud Platform αλλά και το όνομα της κατηγορίας που ανήκει η κάθε φωτογραφία. Συνιστάται κάθε κατηγορία να έχει πάνω από 100 διαφορετικές φωτογραφίες. Έπειτα μεταφερθήκαμε στην ιστοσελίδα του Google Cloud Auto ML Vision όπου καλούμαστε να επιλέξουμε το storage bucket που αποθηκεύσαμε τα δεδομένα για να εισαχθούν στο Cloud Auto ML Vision. Όταν ολοκληρωθεί η εισαγωγή, με χρήση του αρχείου .csv που είχαμε φτιάξει, η εφαρμογή αυτή χωρίζει τις φωτογραφίες ανά κατηγορία, ενώ όσες φωτογραφίες δεν ήταν δυνατόν να καταναμεθούν σε κάποια κατηγορία, ο χρήστης έχει την επιλογή να κάνει χειροκίνητη εισαγωγή. Η πλατφόρμα υποστηρίζει φωτογραφίες τύπων JPEG, PNG, GIF, BMP και ICO με μέγιστο μέγεθος ανά φωτογραφία τα 30 Megabyte. Κατά την εκπαίδευση του μοντέλου το Google Cloud Auto ML Vision χωρίζει τυχαία το σύνολο δεδομένων στα τρία παρακάτω μέρη:

- *Training Dataset*

  Το training dataset, ή αλλιώς σύνολο δεδομένων εκπαίδευσης, περιέχει το 80% των φωτογραφιών που εισάγαμε στην πλατφόρμα Google Cloud Auto ML Vision, και χρησιμοποιείται για την κατασκευή του μοντέλου μηχανικής μάθησης. Το μοντέλο αυτό δοκιμάζει πολλαπλούς αλγόριθμους και παραμέτρους, ενώ ταυτόχρονα ψάχνει για μοτίβα στα δεδομένα εκπαίδευσης.

- *Validation Dataset*

  Το validation dataset, ή όπως μεταφράζεται στα ελληνικά σύνολο δεδομένων επικύρωσης, περιέχει τις μισές φωτογραφίες που απέμειναν μετά την αφαίρεση των φωτογραφιών του training dataset,

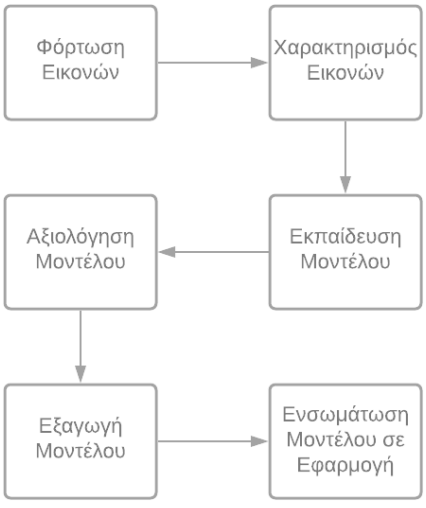
δηλαδή περιέχει 10% των φωτογραφιών. Η χρήση του είναι στη διαδικασία της εκπαίδευσης του μοντέλου, καθώς βοηθά κατά τη διάρκεια της αναγνώρισης των μοτίβων στο συντονισμό των υπερ – παραμέτρων του μοντέλου, αλλά και συμβάλλει σε έναν πρόωρο έλεγχο του μοντέλου. Στο χρονικό διάστημα που γίνεται ο συγκεκριμένος έλεγχος και εφόσον υπάρχουν επαρκή αποτελέσματα και μετρήσεις, τα οποία δεν είναι απόλυτα ακριβή, η πλατφόρμα αποφασίζει να σταματήσει την εκπαίδευση του μοντέλου. Οι αλγόριθμοι και τα μοτίβα με την καλύτερη απόδοση, επιλέγονται από όλους όσους δοκιμάστηκαν με τη χρήση του training dataset.

- *Test Dataset*

  Το test dataset, ή σύνολο δεδομένων ελέγχου, αποτελείται από το υπόλοιπο 10% των φωτογραφιών που απομένουν και χρησιμοποιείται για να μετρήσει και να ελέγξει τους αλγόριθμους και τα μοτίβα, που επιλέχθηκαν στην προηγούμενη φάση, τα ποσοστά σφάλματος, την ποιότητά τους και την ακρίβειά τους. Η χρήση ενός test dataset για τον έλεγχο του μοντέλου παρέχει μια αμερόληπτη αξιολόγηση της ποιότητάς του.

Εφόσον χωριστούν τα δεδομένα στα παραπάνω σύνολα δεδομένων, η εφαρμογή μας δίνει τη δυνατότητα να επιλέξουμε τις ώρες που θα εκπαιδεύσουμε το μοντέλο, ενώ από κάτω υπάρχει μια πρόταση στην οποία μας λέει τον προτεινόμενο χρόνο για την εκπαίδευση αυτού του μοντέλου, η οποία τιμή του χρόνου προκύπτει από το πλήθος των φωτογραφιών και τον αριθμό των κατηγοριών που υπάρχουν. Μετά την ολοκλήρωση της εκπαίδευσης γίνονται διαθέσιμες οι μετρήσεις και τα αποτελέσματα του μοντέλου, ενώ μας δίνεται η δυνατότητα να ελέγξουμε χειροκίνητα το μοντέλο με δικές μας φωτογραφίες, αλλά και να το εξάγουμε σε μορφές TFLite, που χρησιμοποιήσαμε για να κάνουμε τη σύνδεση με την Android εφαρμογή, ενώ ακόμα το μοντέλο που εκπαιδεύθηκε μπορεί να εξαχθεί σε μορφές Tensorflow.js, Core ML, Container, Coral, REST API και Python.

Στην εικόνα 4.3 εμφανίζεται διαγραμματικά η διαδικασία που ακολουθήσαμε για την εκπαίδευση των μοντέλων μας.



Εικόνα 4.3: Διαδικασία Εκπαίδευσης Μοντέλων.

4.5 Εξαγωγή Μοντέλων

Στην ενότητα αυτή θα αναφερθούμε στη διαδικασία που ακολουθήσαμε για την εξαγωγή των μοντέλων μηχανικής μάθησης από την πλατφόρμα Google Cloud Auto ML Vision και την ενσωμάτωσή τους στην εφαρμογή μας.

Όπως αναφέραμε στην προηγούμενη ενότητα, η πλατφόρμα που χρησιμοποιήσαμε μας παρέχει πολλές επιλογές για την εξαγωγή των μοντέλων μηχανικής μάθησης. Στην δική μας περίπτωση επιλέξαμε να εξάγουμε τα μοντέλα σε μορφή TensorFlow Lite, καθώς αυτή χρησιμοποιείται στις εφαρμογές Android μαζί με το Google ML Kit.

Η πλατφόρμα αποθηκεύει το μοντέλο σε ένα σημείο του storage bucket με τις φωτογραφίες, που έχουμε επιλέξει εμείς, ενώ το μοντέλο αποτελείται από τρία διαφορετικά αρχεία. Το πρώτο αρχείο έχει ως προέκταση το .tflite και είναι το μοντέλο μηχανικής μάθησης που εκπαιδεύσαμε με το Google Cloud Auto ML Vision. Το δεύτερο αρχείο είναι το dict.txt και αποτελείται από τις κατηγορίες – κλάσεις του μοντέλου μας. Η σειρά που είναι γραμμένες αυτές οι κατηγορίες δεν ακολουθούν κάποια κανονική ταξινόμηση, για παράδειγμα αλφαβητική, αλλά ταξινομήθηκαν στην εκπαίδευση του μοντέλου μας. Για παράδειγμα στην περίπτωση του μοντέλου μηχανικής μάθησης για ηλεκτρονικά είδη, όταν το μοντέλο ζητάει το όνομα της πρώτης κατηγορίας για να εμφανιστεί στην εφαρμογή μας, λαμβάνει ως απάντηση την κατηγορία «relay». Τέλος το τρίτο αρχείο που προκύπτει από την εξαγωγή του μοντέλου μας είναι το αρχείο tflite_metadata.json που περιέχει τα metadata, ή αλλιώς μεταδεδομένα, του μοντέλου μας. Η χρήση τους είναι να προσδιορίσουν τους τύπους που δέχεται το μοντέλο μας ως είσοδο για σωστή και αποτελεσματική αναγνώριση αντικειμένων. Κάποιες από τις πληροφορίες που περιέχει είναι τα κανάλια χρωμάτων της εικόνας, το πλάτος και το ύψος της εικόνας που θα δεχθεί για είσοδο.

Τέλος με τη χρήση του Android Studio κάνουμε εισαγωγή των αρχείων των μοντέλων μηχανικής μάθησης που εκπαιδεύσαμε και τα καταχωρούμε σε φακέλους, κάθε μοντέλο ξεχωριστά από τα άλλα για να μην μπερδευτούν τα αρχεία των κατηγοριών και των μεταδεδομένων, καθώς πρέπει να έχουν συγκεκριμένα ονόματα.

4.6 Προεπεξεργασία Δεδομένων

Στην ενότητα αυτή θα αναφερθούμε στην προεπεξεργασία δεδομένων και θα αναλύσουμε γιατί θεωρείται σημαντική για την εκπαίδευση μοντέλων μηχανικής μάθησης. Η προεπεξεργασία δεδομένων, είναι ένα από τα πιο σημαντικά βήματα της μηχανικής μάθησης, καθώς χωρίς σωστά δεδομένα δεν θα υπάρξει ποτέ σωστό μοντέλο. Κατά τη διάρκεια αυτής της διαδικασίας τα δεδομένα που δίνουμε στον αλγόριθμο μηχανικής μάθησης μετατρέπονται σε τέτοιο τύπο έτσι ώστε να μπορεί ο υπολογιστής να τα καταλάβει. Για παράδειγμα στο μοντέλο που εκπαιδεύσαμε με τους τύπους λουλουδιών οι πέντε κατηγορίες μετατράπηκαν σε ακέραιους αριθμούς με σκοπό την εύκολη κατανόησή τους από τον υπολογιστή. Επίσης οι φωτογραφίες που δίνουμε ως είσοδο μετατρέπονται σε πίνακες που περιέχουν αριθμητικά δεδομένα.

Η προεπεξεργασία δεδομένων χωρίζεται στα εξής τρία μέρη:

- *Καθαρισμός Δεδομένων*

  Στον καθαρισμό δεδομένων αφαιρούνται δεδομένα με ανεπαρκείς ή εσφαλμένες πληροφορίες. Τέτοιου είδους δεδομένα προκαλούνται συνήθως από σφάλματα κατά την εισαγωγή των δεδομένων από τη χρήστη ή κάποια καταστροφή αυτών. Η αφαίρεση αυτών των δεδομένων προτείνεται καθώς μπορούν να επηρεάσουν την διαδικασία της εκπαίδευσης του μοντέλου μηχανικής μάθησης και να αλλοιωθούν τα αποτελέσματά του. Μία άλλη περίπτωση καθαρισμού δεδομένων είναι η αφαίρεση εικόνων με θόρυβο ή διπλότυπων εικόνων.

- *Επεξεργασία Δεδομένων*

  Με την χρήση της επεξεργασίας δεδομένων έχουμε τη δυνατότητα να διορθώσουμε τις τιμές κάποιων από αυτά για να τα χρησιμοποιήσουμε στην εκπαίδευση των μοντέλων μας. Για παράδειγμα στη δική μας περίπτωση με τη χρήση του Google Cloud Auto ML Vision, κάποια δεδομένα δεν είχαν τιμή για την κατηγορία που ανήκουν ως φωτογραφίες και χρειάστηκε να την ορίσουμε χειροκίνητα.

- *Μείωση Δεδομένων*

  Τη μείωση δεδομένων τη χρησιμοποιούμε για να απλοποιήσουμε το σύνολο δεδομένων μας. Μία συχνή χρήση είναι η αφαίρεση δεδομένων μιας κατηγορίας καθώς περιέχει πολλά δεδομένα παραπάνω από άλλες κατηγορίες. Επίσης ένας ακόμα λόγος μείωσης δεδομένων είναι η μείωση του χρόνου εκπαίδευσης που θα χρειαστεί για το μοντέλο μας, αλλά και η μείωση του μεγέθους του για να καταλαμβάνει το μικρότερο δυνατό μέγεθος στην εφαρμογή και συνεπώς στις συσκευές που θα εγκατασταθεί αυτή.

  Στην περίπτωση του Google Cloud Auto ML Vision, που χρησιμοποιήσαμε για την εκπαίδευση των μοντέλων μηχανικής μάθησης, για την σωστή ροή της διαδικασίας της εκπαίδευσης μας ζητείται η παροχή ενός αρχείου με προέκταση .csv που περιέχει την κατηγορία που ανήκει το κάθε δεδομένο. Με αυτήν την κατηγορία ως κριτήριο γίνεται η κατανομή των δεδομένων μας.

Κεφάλαιο 5ο: Αποτελέσματα

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφέρουμε και θα αναλύσουμε τα αποτελέσματα που είχαμε με τη χρήση της εφαρμογής μας σε πραγματικές συνθήκες. Επίσης στις πρώτες ενότητες του συγκεκριμένου κεφαλαίου θα ασχοληθούμε με τις μετρήσεις Precision, Recall και F1 για τα τρία μοντέλα μηχανικής μάθησης, αλλά και τα δεδομένα ανά κατηγορία που έχουν, για παράδειγμα τους αριθμούς των ψευδών αρνητικών και ψευδών θετικών αποτελεσμάτων.

5.2 Στατιστικές Μετρήσεις Μοντέλων Μηχανικής Μάθησης

Οι μετρήσεις Precision, Recall και F1 χρησιμοποιούνται για να δείξουν την αποτελεσματικότητα ενός μοντέλου μηχανικής μάθησης.

Στις παρακάτω εξισώσεις με τη μεταβλητή tp συμβολίζουμε τον αριθμό των αληθινών θετικών αποτελεσμάτων και προκύπτει από το όνομα true positives. Η μεταβλητή fp αντίστοιχα, είναι ο αριθμός των ψευδών θετικών αποτελεσμάτων και προκύπτει από το όνομα false positives και η μεταβλητή fn είναι ο αριθμός των ψευδών αρνητικών αποτελεσμάτων, ενώ το όνομά της προκύπτει από το όνομα false negatives

Precision σημαίνει ακρίβεια και το αποτέλεσμά της μέτρησης αυτής προκύπτει από τη διαίρεση του αριθμού των αληθινών θετικών αποτελεσμάτων προς το άθροισμα των αληθινών θετικών αποτελεσμάτων και των ψευδών θετικών αποτελεσμάτων.

$$Precision = \frac{tp}{tp + fp}$$

Recall σημαίνει ανάκληση και προκύπτει από τη διαίρεση του αριθμού των αληθινών θετικών αποτελεσμάτων προς το άθροισμα των αληθινών θετικών αποτελεσμάτων και των ψευδών αρνητικών αποτελεσμάτων.

$$Recall = \frac{tp}{tp + fn}$$

Ακόμα χρησιμοποιήσαμε τη μέτρηση F1 Score, η οποία προκύπτει από την παρακάτω εξίσωση.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Οι μετρήσεις Precision, Recall και F1 έχουν ως μέγιστη τιμή το 100.0% και όσο πιο κοντά είναι η μέτρηση σε αυτή τη τιμή, τόσο καλύτερο θεωρείται το μοντέλο μηχανικής μάθησης.

Στους παρακάτω πίνακες παρουσιάζονται οι μετρήσεις αυτές και για τα τρία μοντέλα, καθώς και σε διαφορετικά ποσοστά σιγουριάς. Στον πρώτο πίνακα βλέπουμε τα αποτελέσματα της εκπαίδευσης του μοντέλου μηχανικής μάθησης που εκπαιδεύτηκε με το CIFAR – 100 σύνολο δεδομένων, στον δεύτερο πίνακα υπάρχουν τα αποτελέσματα του μοντέλου μηχανικής μάθησης για αναγνώριση ηλεκτρονικών ειδών και στον τρίτο και τελευταίο πίνακα έχουμε συγκεντρώσει τα αποτελέσματα της εκπαίδευσης του μοντέλου μηχανικής μάθησης για αναγνώριση τύπων λουλουδιών. Σε κάθε πίνακα το πρώτο ποσοστό σιγουριάς είναι το 25%, το δεύτερο είναι το 50% και το τελευταίο το 75% και τα αναφέρουμε με την ονομασία Confidence Threshold, ή στα ελληνικά Όριο Σιγουριάς. Επίσης όσο μεγαλύτερο είναι το ποσοστό σιγουριάς τόσο μεγαλύτερο είναι το Precision και μικρότερη η μέτρηση του Recall.

	Precision	Recall	F1
Confidence Threshold 25%	72,61%	70,13%	71,35%
Confidence Threshold 50%	87,21%	54,99%	67,45%
Confidence Threshold 75%	93,80%	38,94%	55,03%

Πίνακας 5.1: Μετρήσεις Μοντέλου Μηχανικής Μάθησης εκπαιδευμένο με το CIFAR – 100 σύνολο δεδομένων.

	Precision	Recall	F1
Confidence Threshold 25%	60,68%	40,71%	48,72%
Confidence Threshold 50%	74,50%	25,99%	38,53%
Confidence Threshold 75%	82,05%	12,65%	21,92%

Πίνακας 5.2: Μετρήσεις Μοντέλου Μηχανικής Μάθησης εκπαιδευμένο με το σύνολο δεδομένων για ηλεκτρονικά είδη.

	Precision	Recall	F1
Confidence Threshold 25%	93,01%	97,93%	95,40%
Confidence Threshold 50%	98,56%	94,48%	96,47%
Confidence Threshold 75%	99,47%	85,75%	92,10%

Πίνακας 5.3: Μετρήσεις Μοντέλου Μηχανικής Μάθησης εκπαιδευμένο για αναγνώριση τύπων λουλουδιών.

Στους παραπάνω πίνακες παρατηρούμε αύξηση του Precision με κάθε αύξηση του Confidence Threshold, καθώς μειώνεται ο αριθμός των ψευδών θετικών αποτελεσμάτων, ενώ παρατηρείται μείωση του αριθμού της μέτρησης Recall καθώς η αύξηση του Ποσοστού Σιγουριάς αυξάνει τον αριθμό ψευδών αρνητικών αποτελεσμάτων. Το μοντέλο αναγνώρισης λουλουδιών σύμφωνα με τις μετρήσεις θεωρείται το πιο ακριβές και το πιο αποτελεσματικό από τα τρία. Αυτό συμβαίνει διότι το τρίτο μοντέλο έχει τις

λιγότερες κατηγορίες αναγνώρισης ενώ υπήρχε επαρκής αριθμός δεδομένων ανά κατηγορία. Το δεύτερο σε ποιότητα μοντέλο είναι το μοντέλο μηχανικής μάθησης που είχε εκπαιδευτεί με το σύνολο δεδομένων CIFAR – 100. Όπως και το προηγούμενο μοντέλο που αναφέραμε, το μοντέλο αυτό έχει αυτές τις τιμές στις μετρήσεις Precision, Recall και F1 διότι σε κάθε κατηγορία περιείχε 500 φωτογραφίες. Το μοντέλο αναγνώρισης ηλεκτρονικών ειδών είχε τις χειρότερες μετρήσεις σε σύγκριση με τα υπόλοιπα μοντέλα, διότι υπήρχε μεγάλη διαφορά μεταξύ των κατηγοριών σχετικά με το εύρος των δεδομένων τους. Για παράδειγμα στην κατηγορία «Ολοκληρωμένο Κύκλωμα με Φως», που ήταν η κατηγορία με τα λιγότερα δεδομένα, υπήρχαν 57 φωτογραφίες, ενώ στην κατηγορία «Φως τύπου LED» υπήρχαν 472 φωτογραφίες και ήταν η κατηγορία με τις περισσότερες φωτογραφίες στο σύνολο δεδομένων. Επίσης ο αριθμός 57 θεωρείται πολύ μικρό δείγμα στην εκπαίδευση ενός μοντέλου μηχανικής μάθησης για αναγνώριση εικόνων ανά κατηγορία.

Στους πίνακες που ακολουθούν θα παρουσιάσουμε τα αποτελέσματα ανά κατηγορία των μοντέλων μας σε ποσοστό σιγουριάς 50%. Πιο αναλυτικά θα δούμε τις μετρήσεις Precision και Recall, τον αριθμό δεδομένων που εξετάστηκαν ανά κατηγορία, αλλά και τους αριθμούς αληθινών θετικών αποτελεσμάτων, τους αριθμούς ψευδών αρνητικών αποτελεσμάτων και τους αριθμούς ψευδών θετικών αποτελεσμάτων. Οι παρακάτω πίνακες περιέχουν τα αποτελέσματα των μετρήσεων για το μοντέλο μηχανικής μάθησης που εκπαιδεύτηκε με το σύνολο δεδομένων CIFAR – 100. Ο πρώτος πίνακας περιέχει τις μετρήσεις ανά κατηγορία, ενώ ο δεύτερος πίνακας έχει τις μετρήσεις ανά υπερκλάση, με σκοπό την καλύτερη κατανόηση των αποτελεσμάτων.

Κατηγορία	Precision	Recall	Αριθμός Δεδομένων	TP	FN	FP
Apple	90%	72%	50	36	14	4
Aquarium Fish	97,14%	69,39%	49	34	15	1
Baby	63,33%	37,25%	51	19	32	11
Bear	48%	24%	50	12	38	13
Beaver	61,11%	22%	50	11	39	7
Bed	80%	64%	50	32	18	8
Bee	93,75%	60%	50	30	20	2
Beetle	66,67%	44%	50	22	28	11
Bicycle	87,10%	54%	50	27	23	4
Bottle	94,29%	66%	50	33	17	2
Bowl	87,10%	54%	50	27	23	4
Boy	84,21%	32%	50	16	34	3
Bridge	93,33%	56%	50	28	22	2
Bus	92,59%	50%	50	25	25	2
Butterfly	93,10%	54%	50	27	23	2
Camel	93,33%	56%	50	28	22	2
Can	88,57%	62%	50	31	19	4
Castle	97,06%	66%	50	33	17	1
Caterpillar	92,86%	52%	50	26	24	2
Cattle	92%	46%	50	23	27	2
Chair	90%	54%	50	27	23	3
Chimpanzee	100%	64%	50	32	18	0
Clock	92,50%	74%	50	37	13	3

Cloud	80,95%	68%	50	34	16	8
Cockroach	90,24%	74%	50	37	13	4
Computer Keyboard	90,24%	74%	50	37	13	4
Couch	91,67%	44%	50	22	28	2
Crab	75%	42%	50	21	29	7
Crocodile	75,68%	44%	50	22	28	7
Cup	94,29%	66%	50	33	17	2
Dinosaur	96,43%	54%	50	27	23	1
Dolphin	82,14%	46%	50	23	27	5
Elephant	96,30%	52%	50	26	24	1
Flatfish	89,66%	52%	50	26	24	3
Forest	88,89%	48%	50	24	26	3
Fox	90,91%	60%	50	30	20	3
Girl	68,42%	26%	50	13	37	6
Hamster	84,21%	64%	50	32	18	6
House	87,88%	58%	50	29	21	4
Kangaroo	83,33%	60%	50	30	20	6
Lamp	87,10%	54%	50	27	23	4
Lawn Mower	97,56%	80%	50	40	10	1
Leopard	83,78%	62%	50	31	19	6
Lion	88,10%	74%	50	37	13	5
Lizard	74,07%	40%	50	20	30	7
Lobster	91,67%	44%	50	22	28	2
Man	73,68%	28%	50	14	36	5
Maple Tree	75%	30%	50	15	35	5
Motorcycle	94,87%	74%	50	37	13	2
Mountain	89,74%	70%	50	35	15	4
Mouse	78,95%	30%	50	15	35	4
Mushroom	88,89%	64%	50	32	18	4
Oak Tree	69,23%	36%	50	18	32	8
Orange	88,89%	80%	50	40	10	5
Orchid	96,77%	60%	50	30	20	1
Otter	80%	15,09%	53	8	45	2
Palm Tree	90,24%	74%	50	37	13	4
Pear	96,43%	54%	50	27	23	1
Pickup Truck	91,18%	62%	50	31	19	3
Pine Tree	83,33%	40%	50	20	30	4
Plain	91,89%	68%	50	34	16	3
Plate	81,08%	60%	50	30	20	7
Poppy	79,41%	54%	50	27	23	7
Porcupine	90,63%	58%	45	24	21	3
Possum	84,62%	44%	50	22	28	4
Rabbit	96,15%	50%	50	25	25	1
Raccoon	94,59%	70%	50	35	15	2

Ray	78,95%	60%	50	30	20	8
Road	100%	62%	50	31	19	0
Rocket	94,74%	72%	50	36	14	2
Rose	90,24%	74%	50	37	13	4
Sea	70%	70%	50	35	15	15
Seal	66,67%	24%	50	12	38	6
Shark	84%	42%	50	21	29	4
Shrew	69,57%	31,37%	51	16	35	7
Skunk	97,73%	86%	50	43	7	1
Skyscraper	100%	76%	50	38	12	0
Snail	88,89%	48%	50	24	26	3
Snake	71,05%	52,94%	51	27	24	6
Spider	82,76%	47,06%	51	24	27	5
Squirrel	86,67%	26%	50	13	37	2
Streetcar	77,14%	54%	50	27	23	8
Sunflower	100%	84%	50	42	8	0
Sweet Pepper	81,48%	44%	50	22	28	5
Table	90,32%	56%	50	28	22	3
Tank	100%	80%	50	40	10	0
Telephone	92,86%	78%	50	39	11	3
Television	94,44%	68%	50	34	16	2
Tiger	88,89%	64%	50	32	18	4
Tractor	93,10%	54%	50	27	23	2
Train	86,21%	50%	50	25	25	4
Trout	82,86%	58%	50	29	21	6
Tulip	73,91%	34%	50	17	33	6
Turtle	81,48%	44%	50	22	28	5
Wardrobe	95,35%	82%	50	41	9	2
Whale	88,46%	45,10%	51	23	28	3
Willow Tree	81,82%	36%	50	18	32	4
Wolf	90,63%	58%	50	29	21	3
Woman	88,89%	32%	50	16	34	2
Worm	87,80%	70,59%	51	36	15	5

Πίνακας 5.4: Αναλυτικές Μετρήσεις Μοντέλου Μηχανικής Μάθησης εκπαιδευμένο με το CIFAR – 100 dataset.

Υπερκλάση	Precision	Recall	Αριθμός Δεδομένων	TP	FN	FP
Aquatic Mammals	77%	30,31%	254	77	177	23
Fish	86,42%	56,22%	249	140	109	22
Flowers	89,47%	61,20%	250	153	97	18
Food Containers	89,02%	61,60%	250	154	96	19
Fruits and Vegetables	89,20%	62,80%	250	157	93	19

Household Electrical Devices	91,58%	69,60%	250	174	76	16
Household Furniture	89,29%	60%	250	150	100	18
Insects	87,12%	56,80%	250	142	108	21
Large Carnivores	81,98%	56,40%	250	141	109	31
Large Man-Made Outdoor Things	95,78%	63,60%	250	159	91	7
Large Natural Outdoor Scenes	83,08%	64,80%	250	162	88	33
Large Omnivores and Herbivores	92,67%	55,60%	250	139	111	11
Medium – sized Mammals	92,22%	62,86%	245	154	91	13
Non – insects Invertebrates	85,23%	50,40%	252	127	125	22
People	74,29%	31,08%	251	78	173	27
Reptiles	81,94%	47,01%	251	118	133	26
Small Mammals	83,47%	40,23%	251	101	150	20
Trees	81,20%	43,20%	250	108	142	25
Vehicles 1	90,63%	58%	250	145	105	15
Vehicles 2	92,90%	68%	250	170	80	13

Πίνακας 5.5: Αναλυτικές Μετρήσεις Υπερκλάσεων Μοντέλου Μηχανικής Μάθησης εκπαιδευμένο με το CIFAR – 100 dataset.

Από τα παραπάνω αποτελέσματα διακρίνουμε πως η υπερκατηγορία των ανθρώπων (People) είναι η χειρότερη με βάση την ακρίβεια (Precision) και δεύτερη χειρότερη συγκριτικά με τις υπόλοιπες στη μέτρηση της ανάκλησης (Recall). Επίσης η υπερκατηγορία των θαλάσσιων θηλαστικών (Aquatic Mammals) είναι η δεύτερη χειρότερη υπερκατηγορία στη μέτρηση της ακρίβειας και η χειρότερη στη μέτρηση της ανάκλησης.

Από την άλλη πλευρά η καλύτερη υπερκατηγορία στη μέτρηση του Precision είναι η υπερκατηγορία των Τεχνητών Υπαίθριων Πραγμάτων (Large Man – Made Objects) και η καλύτερη υπερκατηγορία με βάση την ανάκληση είναι η υπερκατηγορία των Ηλεκτρικών Συσκευών (Homemade Electrical Objects).

Στην περίπτωση των μετρήσεων των κατηγοριών παρατηρούμε ότι πέντε κατηγορίες είχαν τη μέγιστη τιμή για τη μέτρηση του Precision και ήταν οι κατηγορίες Chimpanzee (Χιμπατζής), Road (Δρόμος), Skyscraper (Ουρανοξύστης), Sunflower (Ήλιος) και Tank (Τανκς). Στην μέτρηση της τιμής της ανάκλησης οι κατηγορίες με το καλύτερο ποσοστό ήταν οι κατηγορίες Skunk (Ασβός) με ποσοστό 86%, Sunflower (Ήλιος) έχοντας αποτέλεσμα 84% και Wardrobe (Ντουλάπα) με μέτρηση 82%.

Από την άλλη πλευρά οι χειρότερες κατηγορίες στη μέτρηση του Precision είναι οι κατηγορίες Bear (Αρκούδα) με αποτέλεσμα μέτρησης 48%, Beaver (Κάστορας) με ποσοστό 61,11% και Baby (Μωρό) με μέτρηση 63,33%. Ακόμα στη μέτρηση της ανάκλησης οι κατηγορίες με τη χειρότερη μέτρηση είναι οι κατηγορίες Otter (Βίδρα) με ποσοστό 15,09%, Beaver (Κάστορας) με αποτέλεσμα 22% και Bear (Αρκούδα) με αποτέλεσμα 24%.

Ο παρακάτω πίνακας αναφέρεται στις μετρήσεις του μοντέλου μηχανικής μάθησης που εκπαιδεύτηκε με το σύνολο δεδομένων ηλεκτρονικών ειδών.

Κατηγορία	Precision	Recall	Αριθμός Δεδομένων	TP	FN	FP
Armature	100%	40,63%	32	13	19	0
Attenuator	100%	11,11%	27	3	24	0
Bypass Capacitor	0%	0%	30	0	30	0
Cartridge Fuse	100%	63,64%	22	14	8	0
Clip Lead	76,92%	34,48%	29	10	19	3
Electric Relay	50%	4,55%	22	1	21	1
Electrolytic Capacitor	88,89%	20%	40	8	32	1
Filament	96%	60%	40	24	16	1
Heat Sink	68,18%	35,71%	42	15	27	7
Induction Coil	90%	60%	15	9	6	1
Integrated Micro Circuit	35,71%	11,90%	42	5	37	9
Jumper Cable	75%	36%	25	9	16	3
Junction Transistor	30%	6,82%	44	3	41	7
LED	81,82%	57,45%	47	27	20	6
Light Circuit	0%	0%	6	0	6	0
Limiter Clipper	88,89%	37,21%	43	16	27	2
Local Oscillator	0%	0%	11	0	11	1
Memory Chip	85,71%	20,69%	29	6	23	1
Microchip	25%	5%	40	2	38	6
Microprocessor	41,67%	11,90%	42	5	37	7
Multiplexer	100%	62,50%	8	5	3	0
Omni – Directional Antenna	100%	58,33%	12	7	5	0
PNP – Transistor	33,33%	7,14%	28	2	26	4

Potential Divider	0%	0%	23	0	23	1
Potentiometer	53,33%	17,39%	46	8	38	7
Pulse Generator	88,24%	46,88%	32	15	17	2
Relay	75%	12,50%	24	3	21	1
Rheostat	50%	6,25%	16	1	15	1
Semi – Conductor	88,89%	21,05%	38	8	30	1
Semi – Conductor Diode	63,64%	17,95%	39	7	32	4
Shunt	100%	40%	10	4	6	0
Solenoid	83,33%	16,13%	31	5	26	1
Stabilizer	100%	31,25%	16	5	11	0
Step – Down Transformer	68,75%	28,95%	38	11	27	5
Step – Up Transformer	75%	23,08%	13	3	10	1
Transistor	60%	90%	10	9	1	6

Πίνακας 5.6: Αποτελέσματα Μετρήσεων ανά Κατηγορία Μοντέλου Μηχανικής Μάθησης εκπαιδευμένο για Ηλεκτρονικά Είδη.

Στον παραπάνω πίνακα παρατηρούμε αναλυτικά τις μετρήσεις ανά κατηγορία – κλάση του μοντέλου μηχανικής μάθησης για αναγνώριση ηλεκτρονικών ειδών. Κατά τον έλεγχο παρατηρήσαμε πως τέσσερις κατηγορίες δεν είχαν αληθινά θετικά αποτελέσματα και συνεπώς στις μετρήσεις Precision και Recall είχαν αποτέλεσμα 0%. Οι κατηγορίες αυτές είναι οι Bypass Capacitor (Πυκνωτής Παράκαμψης), Light Circuit (Ολοκληρωμένο Κύκλωμα με Φως), Local Oscillator (Τοπικός Ταλαντωτής) και Potential Divider (Διαιρέτης Τάσης).

Οι χειρότερες κατηγορίες με αποτελέσματα στις μετρήσεις της ακρίβειας είναι οι κατηγορίες Microchip (Μικροτσίπ) με αποτέλεσμα 25%, Junction Transistor (Διπολικό Τρανζίστορ) με ποσοστό ακρίβειας 30% και PnP – Transistor (Τρανζίστορ PnP) με αποτέλεσμα 33,33%. Από την άλλη πλευρά επτά κατηγορίες είχαν το μέγιστο πιθανό αποτέλεσμα στη μέτρηση της ακρίβειας, δηλαδή είχαν ως αποτέλεσμα 100%, και είναι οι κατηγορίες Armature (Επαγώγιμο), Attenuator (Εξασθενητής), Cartridge Fuse (Φυσίγγια Ασφαλειών) , Multiplexer (Πολυπλέκτης) , Omni – Directional Antenna (Παντοκατευθυντική Κεραία), Shunt (Παραδιακλάδωση) και Stabilizer (Σταθεροποιητής).

Στην περίπτωση της μέτρησης Recall οι χειρότερες κατηγορίες, εκτός των προαναφερθεισών που είχαν μηδενική μέτρηση, είναι οι κατηγορίες Electric Relay (Ηλεκτρικό Ρελέ) με μέτρηση 4,55%, Microchip (Μικροτσίπ) με αποτέλεσμα 5% και Rheostat (Ρυθμιστής Ηλεκτρικού Ρεύματος) με ποσοστό ανάκλησης 6,25%. Αντίθετα οι κατηγορίες με τις καλύτερες μετρήσεις είναι η κατηγορία Transistor (Τρανζίστορ) με ποσοστό ανάκλησης 90%,η κλάση Cartridge Fuse (Φυσίγγια Ασφαλειών) με μέτρηση 63,64% και η κατηγορία Multiplexer (Πολυπλέκτης) με αποτέλεσμα 62,50%.

Ο παρακάτω πίνακας περιέχει τα αποτελέσματα των μετρήσεων για το μοντέλο μηχανικής μάθησης εκπαιδευμένο για αναγνώριση τύπων λουλουδιών.

Κατηγορία	Precision	Recall	Αριθμός Δεδομένων	TP	FN	FP
Daisy	100%	97,40%	77	75	2	0
Dandelion	100%	94,29%	105	99	6	0
Rose	98,63%	87,80%	82	72	10	1
Sunflower	100%	97,26%	73	71	2	0
Tulip	94,95%	95,92%	98	94	4	5

Πίνακας 5.7: Αποτελέσματα Μετρήσεων Μοντέλου Μηχανικής Μάθησης για αναγνώριση λουλουδιών.

Από τον παραπάνω πίνακα παρατηρούμε την μεγάλη διαφορά που υπάρχει στα αποτελέσματα σε σχέση με τους προηγούμενους πίνακες. Αυτό οφείλεται στην καλύτερη ανάλυση των φωτογραφιών, στις μειωμένες κατηγορίες και στον αυξημένο αριθμό δεδομένων ανά κατηγορία. Πιο αναλυτικά έχουμε τρεις κατηγορίες με τον μέγιστο πιθανό αριθμό στη μέτρηση της ακρίβειας οι οποίες είναι, η κατηγορία Daisy (Μαργαρίτα), η κατηγορία Dandelion (Πικραλίδα) και η κατηγορία Sunflower (Ήλιος), ενώ η κατηγορία Tulip (Τουλίπα) έχει τη μικρότερη τιμή της μέτρησης Precision με ποσοστό 94,95%.

Στην περίπτωση της μέτρησης Recall η κατηγορία Daisy (Μαργαρίτα) έχει τη καλύτερη τιμή με ποσοστό 97,40%, ενώ η κατηγορία Rose (Τριαντάφυλλο) έχει τη χειρότερη μέτρηση με αποτέλεσμα 87,80%.

5.3 Αποτελέσματα Πραγματικής Χρήσης

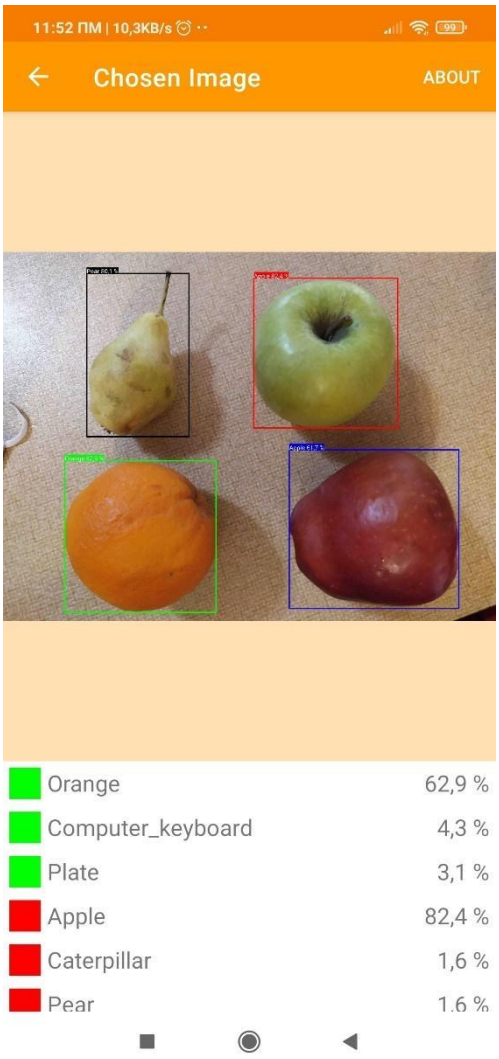
Στην ενότητα αυτή θα παρουσιάσουμε φωτογραφίες και στιγμιότυπα οθόνης από πραγματική συσκευή κατά τον έλεγχο της εφαρμογής που δημιουργήσαμε για τη διπλωματική εργασία. Οι συσκευές που χρησιμοποιήθηκαν σε αυτές τις φωτογραφίες είναι ένα tablet μάρκας Lenovo και πιο συγκεκριμένα το μοντέλο Tab 4 10, και μία κινητή συσκευή Xiaomi Redmi Note 7.

5.3.1 Αποτελέσματα χρήσης μοντέλου για αναγνώριση συνηθισμένων αντικειμένων

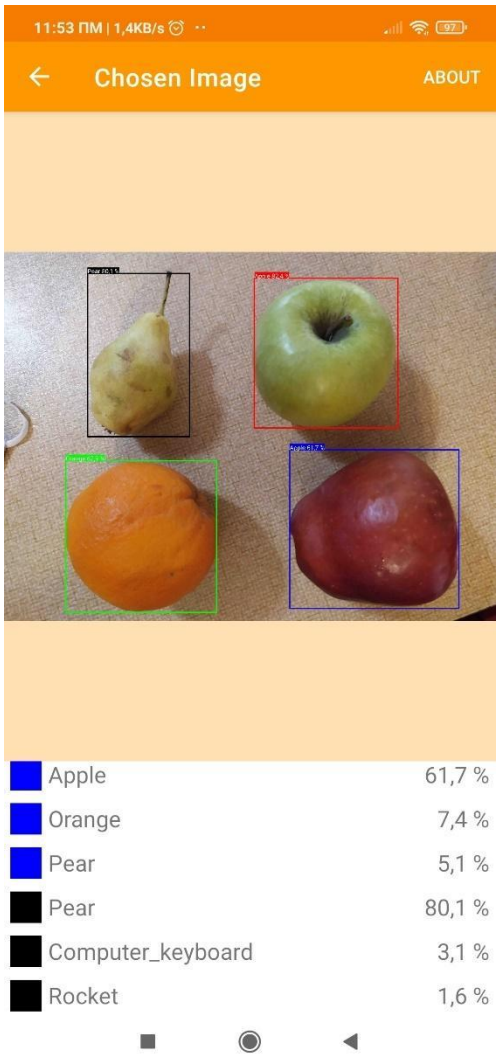
Εικόνα 5.1: Έλεγχος εφαρμογής με φρούτα.

Στην υποενότητα αυτή θα αναφέρουμε και θα αναλύσουμε τα αποτελέσματα χρήσης της εφαρμογής μας για ανίχνευση αντικειμένων με το μοντέλο μηχανικής μάθησης που εκπαιδεύσαμε με το CIFAR – 100 σύνολο δεδομένων.

Στην παραπάνω εικόνα εμφανίζονται τέσσερα διαφορετικά φρούτα. Η φωτογραφία τραβήχτηκε με χρήση της κάμερας μέσω της εφαρμογής μας και περιέχει ένα αχλάδι πάνω αριστερά, ένα πορτοκάλι κάτω αριστερά, ένα πράσινο μήλο πάνω δεξιά και ένα κόκκινο μήλο κάτω δεξιά στη φωτογραφία. Οι παρακάτω εικόνες είναι στιγμιότυπα εικόνας κατά το τρέξιμο της εφαρμογής, και δείχνουν τα αποτελέσματα που εμφανίστηκαν από την εφαρμογή μας όταν χρησιμοποιήθηκε η εικόνα 5.1.



Εικόνα 5.2: Αποτελέσματα Ανίχνευσης Φρούτων για Πράσινο και Κόκκινο Πλαίσιο

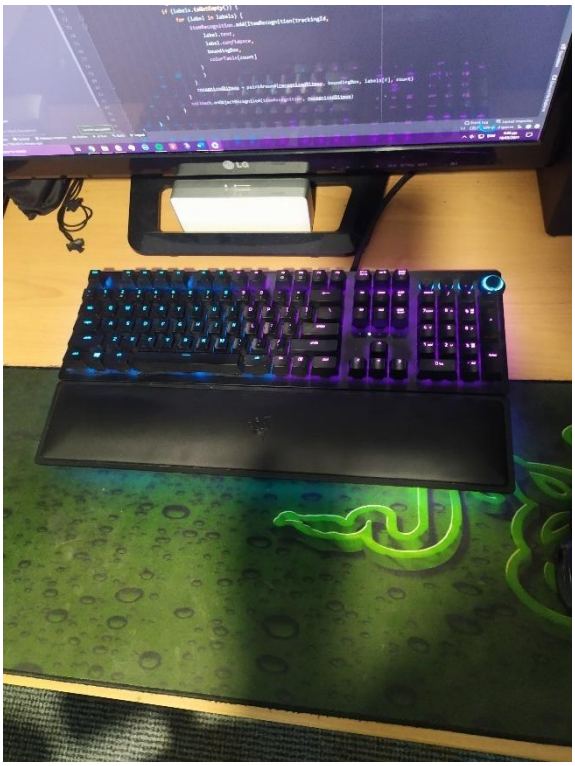


Εικόνα 5.3: Αποτελέσματα Ανίχνευσης Φρούτων για Μπλε και Μαύρο Πλαίσιο.

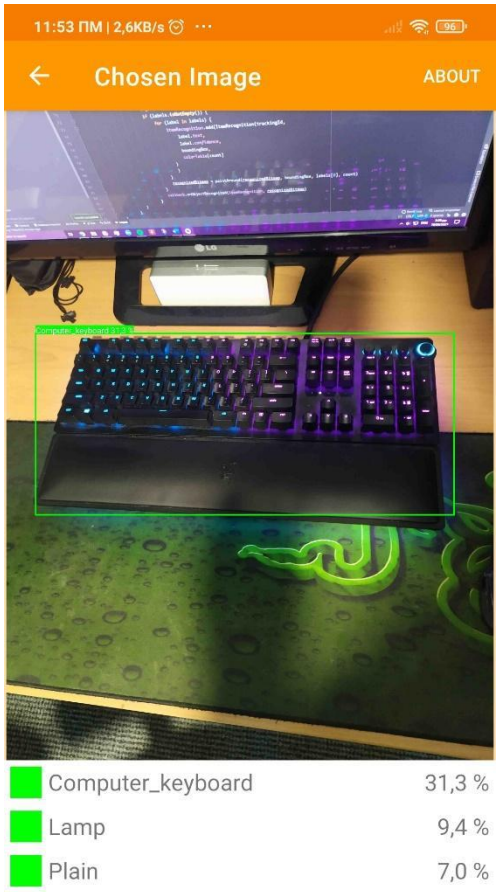
Στις εικόνες 5.2 και 5.3 βλέπουμε τα αποτελέσματα της αναγνώρισης των αντικειμένων και παρατηρούμε πως όλα τα αντικείμενα ανιχνεύθηκαν σωστά και με μεγάλα ποσοστά σιγουριάς. Πιο αναλυτικά, όπως παρατηρούμε από την εικόνα 5.2, το πράσινο πλαίσιο βρίσκεται γύρω από το πορτοκάλι

και αναγνωρίστηκε από το μοντέλο μας ως Πορτοκάλι με ποσοστό σιγουριάς 62,9% και το κόκκινο πλαίσιο βρίσκεται γύρω από το πράσινο μήλο της φωτογραφίας και η εφαρμογή μας αναγνώρισε το αντικείμενο ως Μήλο με ποσοστό σιγουριάς 82,4%. Με τη βοήθεια της εικόνας 5.3 παρατηρούμε πως το κόκκινο μήλο είναι μέσα σε ένα πλαίσιο χρώματος μπλε και το λογισμικό μας, με τη χρήση του μοντέλου μηχανικής μάθησης, αναγνώρισε το φρούτο ως Μήλο με αποτέλεσμα μέτρησης 61,7%. Τέλος, το αχλάδι της φωτογραφίας υπάρχει μέσα σε ένα μαύρο πλαίσιο και ανιχνεύτηκε από την εφαρμογή μας, ως Αχλάδι με ποσοστό 80,1%.

Επίσης δοκιμάσαμε την ανίχνευση ενός πληκτρολογίου, το οποίο εμφανίζεται στην εικόνα 5.4.



Εικόνα 5.4: Φωτογραφία Εισόδου για Ανίχνευση Πληκτρολογίου.



Εικόνα 5.5: Αποτελέσματα Ανίχνευσης Πληκτρολογίου.

Στην εικόνα 5.5 παρατηρούμε πως έγινε επιτυχής ανίχνευση του πληκτρολογίου από την εφαρμογή μας, με ποσοστό σιγουριάς 31,3%. Η δεύτερη κατηγορία αναγνώρισης είναι η κατηγορία Λάμπα με ποσοστό 9,4%, και η τρίτη κατηγορία είναι η Πεδιάδα με ποσοστό σιγουριάς 7.0%. Παρατηρούμε επίσης

πως η διαφορά των δύο πρώτων επιλογών είναι μεγάλη, πιο συγκεκριμένα 21,9%, και συνεπώς δεν μπορεί να αμφισβητηθεί το αποτέλεσμα.

Εικόνα 5.6: Φωτογραφία Εισόδου για Ανίχνευση Ντουλάπας.

Εικόνα 5.7: Αποτελέσματα Πρόβλεψης για Ανίχνευση Ντουλάπας.

Στην εικόνα 5.6 υπάρχει η φωτογραφία μιας ντουλάπας που επιλέξαμε να χρησιμοποιήσουμε στην εφαρμογή μας, ενώ στην εικόνα 5.7 παρατηρούμε τα αποτελέσματα της αναγνώρισης. Βλέπουμε πως η ντουλάπα αναγνωρίστηκε σωστά με μέτρηση 30,1%, ενώ οι άλλες δύο κατηγορίες που αναγνωρίστηκαν είναι οι κατηγορίες Τηλεόραση και Πληκτρολόγιο με μέτρηση 4,3%. Επίσης παρατηρούμε πως η περίπτωση της αναγνώρισης της εικόνας 5.7 είναι παρόμοια με την περίπτωση του πληκτρολογίου, όπως είδαμε στην εικόνα 5.5, δηλαδή πως η κατηγορία με την καλύτερη μέτρηση έχει διαφορά 25,8% από την επόμενη πιθανή αναγνώριση.

Τέλος, στην παρακάτω εικόνα βλέπουμε δύο παιδιά τα οποία παρακολουθούν τηλεόραση. Η φωτογραφία αυτή επιλέχθηκε διότι περιέχει πολλά αντικείμενα που το μοντέλο μηχανικής μάθησης έχει ως κατηγορίες αναγνώρισης, όπως για παράδειγμα την τηλεόραση και την λάμπα.

Εικόνα 5.8: Φωτογραφία Ελέγχου Εφαρμογής για Πολλαπλά Αντικείμενα.

Εικόνα 5.9:Αποτελέσματα Μετρήσεων για Πράσινο και Κόκκινο Πλαίσιο.

Εικόνα 5.10: Αποτελέσματα Μετρήσεων για Κόκκινο και Μπλε Πλαίσιο.

Στις εικόνες 5.9 και 5.10 βλέπουμε τα αποτελέσματα της αναγνώρισης αντικειμένων για την εικόνα 5.8. Παρατηρούμε πως σε κάθε αποτέλεσμα η πρώτη επιλογή είναι εσφαλμένη. Πιο αναλυτικά το πράσινο πλαίσιο βρίσκεται γύρω από το αγόρι που βρίσκεται στα δεξιά και τα αποτελέσματα που προέκυψαν είναι με τη σειρά, Άντρας με ποσοστό 48,0%, Αγόρι με ποσοστό 8,2% και Γυναίκα με ποσοστό 4,7%. Στην περίπτωση αυτή το μοντέλο μας δεν είναι τελείως λάθος, καθώς η δεύτερη επιλογή είναι η σωστή.

Το κόκκινο πλαίσιο βρίσκεται γύρω από την τηλεόραση στη φωτογραφία και με τη χρήση του μοντέλου μηχανικής μάθησης, πήραμε ως αναγνώριση τις κατηγορίες Πληκτρολόγιο με ποσοστό 45,7%, Τηλεόραση με ποσοστό 13,7% και Πεδιάδα με ποσοστό 12,9%. Όπως ακριβώς και στην περίπτωση του πράσινου πλαισίου, παρατηρούμε πως η πρώτη πρόβλεψη είναι λάθος και πως η δεύτερη πρόβλεψη είναι η σωστή.

Στην περίπτωση του μπλε πλαισίου, που βρίσκεται γύρω από το αγόρι στα αριστερά, παρατηρούμε πως η σωστή κατηγορία αναγνώρισης είναι η τρίτη. Οι κατηγορίες που εμφανίστηκαν από το μοντέλο μηχανικής μάθησης είναι η κατηγορία Κουτάκι με ποσοστό 30,1%, η κατηγορία Πιάτο με ποσοστό 11,3% και η κατηγορία Αγόρι με ποσοστό 7,4% που είναι και η σωστή.

Τέλος, παρατηρούμε πως η λάμπα που βρίσκεται στα αριστερά των παραπάνω εικόνων, δεν ανιχνεύθηκε και συνεπώς, δεν αναγνωρίστηκε. Αυτό οφείλεται στη φωτεινότητα της περιοχής της εικόνας που βρίσκεται το αντικείμενο αυτό, καθώς και το φόντο πίσω από το αντικείμενο διότι είναι παρόμοιων χρωμάτων και δεν υπάρχει εμφανής διαφορά. Εξαιτίας αυτών των λόγων, η λάμπα στη φωτογραφία δεν ανιχνεύθηκε από τον Object Detector της εφαρμογής μας.

5.3.2 Αποτελέσματα χρήσης μοντέλου για αναγνώριση ηλεκτρονικών ειδών

Στην υποενότητα αυτή θα αναλύσουμε τα αποτελέσματα που βρήκαμε με τη χρήση της εφαρμογής μας για ανίχνευση και αναγνώριση ηλεκτρονικών ειδών.

Στην παρακάτω εικόνα βλέπουμε ένα ολοκληρωμένο κύκλωμα ή χρονιστή 555 που χρησιμοποιήσαμε στην εφαρμογή μας για δοκιμή της αναγνώρισης ηλεκτρονικών ειδών.

Εικόνα 5.11: Ολοκληρωμένο Κύκλωμα 555.

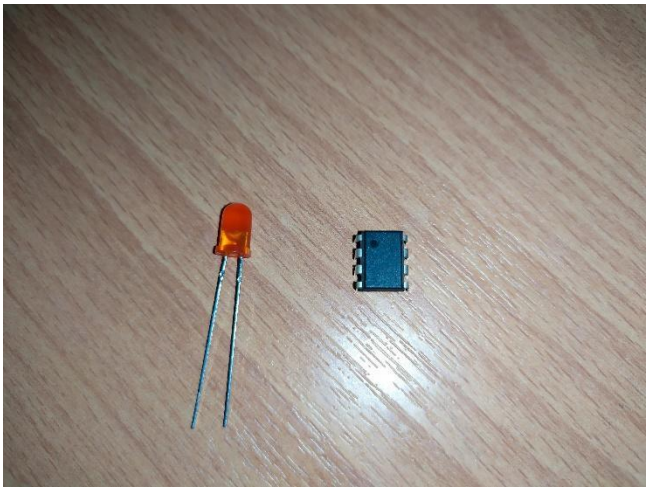
Εικόνα 5.12: Αποτέλεσμα Αναγνώρισης για το Ολοκληρωμένο Κύκλωμα 555.

Στην εικόνα 5.12 βλέπουμε το αποτέλεσμα της αναγνώρισης του ολοκληρωμένου κυκλώματος 555. Η πρώτη πιθανή κατηγορία είναι η κατηγορία Μικροεπεξεργαστής με ποσοστό μέτρησης 13,3%, η δεύτερη είναι η κατηγορία Ψύκτρα με μέτρηση 11,3% και η τρίτη και τελευταία κατηγορία είναι αυτή του Ενσωματωμένου Μικροκυκλώματος, με ποσοστό σιγουριάς 9,4%. Η σωστή αναγνώριση είναι η κατηγορία του Ενσωματωμένου Μικροκυκλώματος, καθώς το ολοκληρωμένο κύκλωμα 555 είναι τύπος Ενσωματωμένου Μικροκυκλώματος. Επίσης η ποσοστιαία διαφορά μεταξύ της πρώτης πρόβλεψης και της σωστής είναι 3,9%.

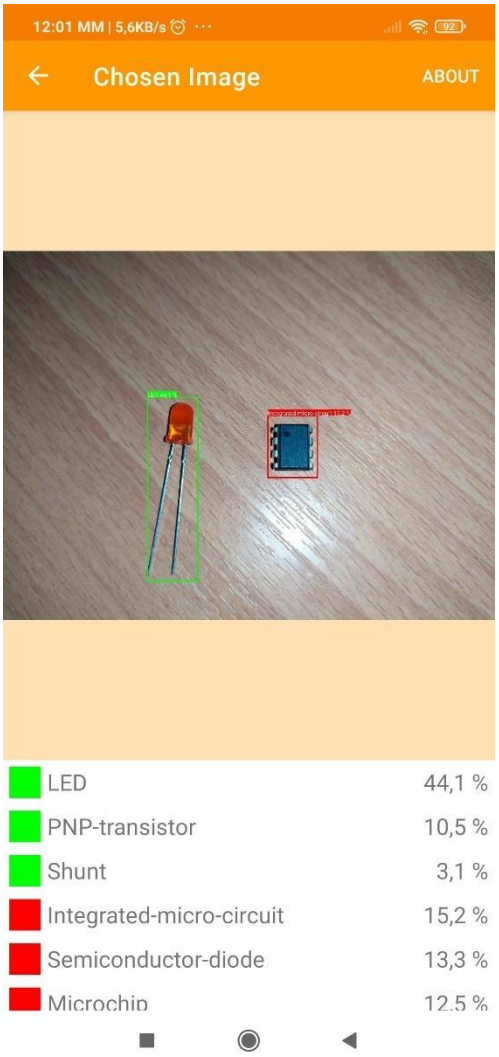
Στην εικόνα 5.13 βλέπουμε το ολοκληρωμένο κύκλωμα 555 μαζί με ένα πορτοκαλί φως τύπου LED, και στην 5.14 παρατηρούμε τα αποτελέσματα της πρόβλεψη του μοντέλου μας για τα δύο αντικείμενα.

Στη εικόνα 5.14 παρατηρούμε πως έγινε σωστή πρόβλεψη των αντικειμένων αλλά και πως η πρώτη πρόβλεψη για κάθε αντικείμενο είναι η σωστή. Πιο συγκεκριμένα το πορτοκαλί φως τύπου LED περιβάλλεται από ένα πράσινο πλαίσιο και το μοντέλο μηχανικής μάθησης προέβλεψε πως ανήκει στην

κατηγορία Φως Τύπου LED με ποσοστό σιγουριάς 44,1%. Η επόμενη πιθανή κατηγορία είναι η κατηγορία Τρανζίστορ PnP με μέτρηση 10,5%, ενώ η τελευταία είναι η κατηγορία Παραδιακλάδωση με ποσοστό 3,1%. Η πρώτη πρόβλεψη, που είναι σωστή, έχει ποσοστιαία διαφορά 33,6% από την επόμενη, οπότε το αποτέλεσμα δεν μπορεί να αμφισβητηθεί.



Εικόνα 5.13: Φωτογραφία Εισόδου για Ανίχνευση Πολλαπλών Αντικειμένων.



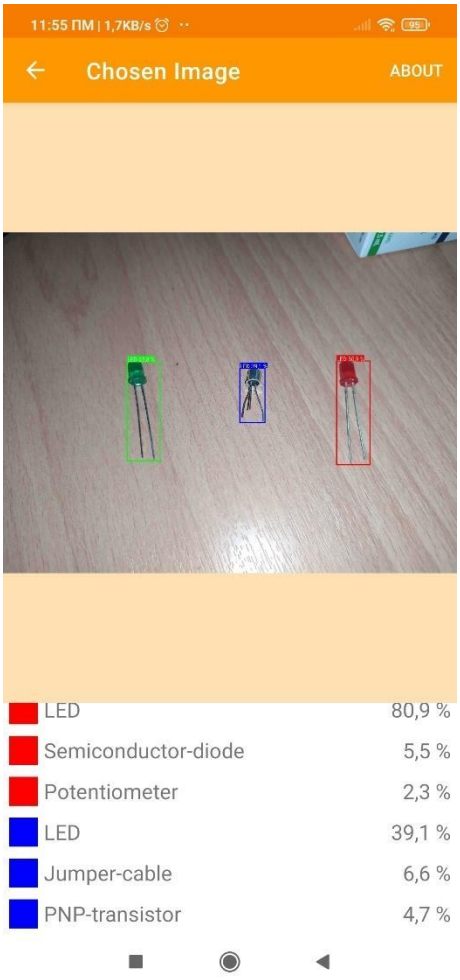
Εικόνα 5.14: Αποτελέσματα Πρόβλεψεων Πολλαπλών Αντικειμένων.

Στην περίπτωση του ολοκληρωμένου κυκλώματος, στην εικόνα 5.14 βρίσκεται μέσα σε ένα κόκκινο πλαίσιο, το μοντέλο μηχανικής μάθησης προέβλεψε πως ανήκει στην κατηγορία Ενσωματωμένο Μικροκύκλωμα με ποσοστό σιγουριάς 15,2%. Η δεύτερη πιθανή κατηγορία του αντικειμένου είναι η κατηγορία Δίοδος Ημιαγωγού με μέτρηση 13,3%, ενώ η τρίτη είναι η κατηγορία Μικροτσίπ με ποσοστό σιγουριάς 12,5%. Σε αντίθεση με το πορτοκαλί φως τύπου LED που αναγνωρίσαμε, παρατηρούμε πως οι προβλέψεις έχουν πολύ μικρή διαφορά μεταξύ τους και συνεπώς το αποτέλεσμα θεωρείται αμφιλεγόμενο.

Στην παρακάτω εικόνα βλέπουμε τη φωτογραφία που τραβήξαμε μέσω τη εφαρμογής μας και περιέχει δύο φώτα τύπου LED, ένα πράσινο και ένα κόκκινο, αλλά και ένα τρανζίστορ τύπου PnP.

Εικόνα 5.15:Φωτογραφία με φωτάκια τύπου LED και τρανζίστορ PnP.

Εικόνα 5.16: Αποτελέσματα Μετρήσεων για Πράσινο και Κόκκινο Πλαίσιο.



Εικόνα 5.17: Αποτελέσματα Μετρήσεων για Κόκκινο και Μπλε Πλαίσιο.

Στις εικόνες 5.16 και 5.17 βλέπουμε τα αποτελέσματα των προβλέψεων του μοντέλου μηχανικής μάθησης και παρατηρούμε πως κάθε αντικείμενο της φωτογραφίας έχει ανιχνευθεί σωστά, αλλά αυτά δεν έχουν αναγνωριστεί όλα σωστά, σύμφωνα με την πρόβλεψη του μοντέλου μας.

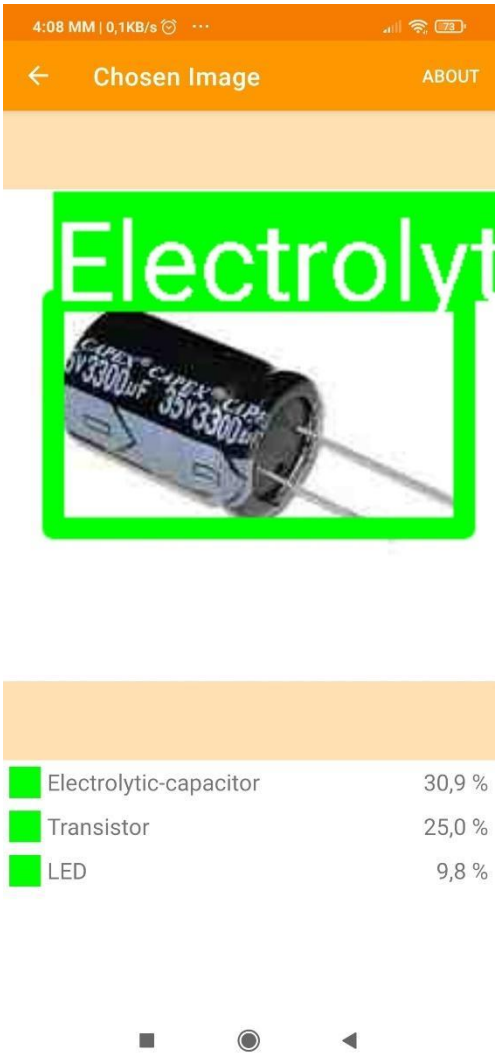
Αρχικά το πράσινο φως τύπου LED, που είναι στην αριστερή πλευρά της φωτογραφίας, βρίσκεται μέσα σε ένα πράσινο πλαίσιο και η πρώτη πρόβλεψη που έκανε το μοντέλο μας για αυτό είναι η κατηγορία Φως Τύπου LED με ποσοστό σιγουριάς 25,8%. Η δεύτερη πιθανή κατηγορία για το αντικείμενο αυτό είναι η κατηγορία Ψύκτρα με μέτρηση 8,2%, ενώ η τρίτη πρόβλεψη είναι πως το αντικείμενο ανήκει στην κατηγορία Δίοδος Ημιαγωγού με ποσοστό σιγουριάς 5,5%. Συνεπώς το μοντέλο μας έκανε σωστή πρόβλεψη σχετικά με το πρώτο αντικείμενο.

Το επόμενο αντικείμενο είναι το κόκκινο φως τύπου LED, το οποίο περικλείεται από το κόκκινο πλαίσιο και βρίσκεται στη δεξιά πλευρά της φωτογραφίας. Η πρώτη πρόβλεψη του μοντέλου μας είναι πως το αντικείμενο αυτό ανήκει στην κατηγορία Φως Τύπου LED και έχει μέτρηση 80,9%. Η επόμενη πιθανή κατηγορία για το αντικείμενο είναι η κατηγορία Δίοδος Ημιαγωγού με ποσοστό σιγουριάς 5,5%, ενώ η τελευταία μέτρηση που μας εμφανίζεται είναι η κατηγορία Ποτενσιόμετρο με ποσοστό σιγουριάς 2,3%. Με βάση τις παραπάνω μετρήσεις, παρατηρούμε πως το μοντέλο μηχανικής μάθησης που χρησιμοποιήσαμε έκανε σωστή πρόβλεψη για το συγκεκριμένο αντικείμενο.

Το τελευταίο αντικείμενο της φωτογραφίας που ανιχνεύθηκε είναι το τρανζίστορ τύπου PnP, που βρίσκεται στη μέση της φωτογραφίας και περιβάλλεται από το μπλε πλαίσιο. Το μοντέλο μας αρχικά προέβλεψε πως το αντικείμενο αυτό ανήκει στην κατηγορία Φως Τύπου LED με ποσοστό σιγουριάς 39,1%, ενώ η δεύτερη πιο πιθανή κατηγορία είναι η κατηγορία Καλώδιο Βραχυκυκλωτήρα με μέτρηση 6,6%. Η τελευταία κατηγορία, που προέβλεψε το μοντέλο, είναι η κατηγορία Τρανζίστορ PnP με ποσοστό σιγουριάς 4,7%. Παρατηρούμε πως το μοντέλο μας έκανε λάθος πρόβλεψη για το αντικείμενο αυτό, καθώς η διαφορά στο ποσοστό σιγουριάς που έχει η σωστή κατηγορία με την πρώτη προβλεπόμενη είναι αρκετά μεγάλη, η διαφορά είναι 34,4%, άρα το αποτέλεσμα της πρόβλεψης δεν θεωρείται αμφιλεγόμενο.

Στην παρακάτω εικόνα βλέπουμε τη φωτογραφία ενός πυκνωτή που χρησιμοποιήσαμε στην εφαρμογή μας για να ελέγξουμε την επιλογή φωτογραφιών ηλεκτρονικών ειδών από τον αποθηκευτικό χώρο.

Εικόνα 5.18: Φωτογραφία ενός Πυκνωτή.



Εικόνα 5.19: Αποτέλεσμα Πρόβλεψης της Εφαρμογής.

Στην παραπάνω εικόνα διακρίνουμε την πρόβλεψη που έκανε το μοντέλο μηχανικής μάθησης για την φωτογραφία ενός πυκνωτή που δώσαμε ως είσοδο. Η πρώτη πρόβλεψη του μοντέλου, ήταν πως το αντικείμενο της φωτογραφίας ανήκει στην κατηγορία Ηλεκτρολυτικός Πυκνωτής με ποσοστό σιγουριάς 30,9%. Η δεύτερη πιθανή κατηγορία που ανήκει το αντικείμενο είναι η κατηγορία Τρανζίστορ με μέτρηση 25,0% ενώ η τρίτη πρόβλεψη είναι πως το αντικείμενο ανήκει στην κατηγορία Φως Τύπου LED με ποσοστό 9,8%. Συνεπώς το μοντέλο μηχανικής μάθησης προέβλεψε σωστά την κατηγορία του αντικειμένου, αλλά λόγω της μικρής διαφοράς με την επόμενη πρόβλεψη, το αποτέλεσμα θεωρείται αμφισβητούμενο.

Τέλος, στην παρακάτω φωτογραφία υπάρχουν τρία ίδια αντικείμενα που ανήκουν στην κατηγορία Ψύκτρα. Την φωτογραφία αυτή τη χρησιμοποιήσαμε για να δοκιμάσουμε τη δυνατότητα αναγνώρισης πολλαπλών ηλεκτρονικών ειδών, όταν η φωτογραφία έχει επιλεχθεί από τον αποθηκευτικό χώρο.

Εικόνα 5.20: Φωτογραφία με πολλαπλές ψύκτρες.

Εικόνα 5.21: Αποτελέσματα Μετρήσεων για Πράσινο και Κόκκινο Πλαίσιο.

Εικόνα 5.22: Αποτελέσματα Μετρήσεων για Κόκκινο και Μπλε Πλαίσιο.

Στις εικόνες 5.21 και 5.22 βλέπουμε τα αποτελέσματα των προβλέψεων του μοντέλου μηχανικής μάθησης για την εικόνα 5.20. Επίσης παρατηρούμε πως και τα τρία αντικείμενα της εικόνας ανιχνεύθηκαν σωστά, αλλά δεν αναγνωρίστηκαν όλα τα αντικείμενα με τη σωστή κατηγορία.

Πιο αναλυτικά το πρώτο αντικείμενο που ανιχνεύθηκε είναι η ψύκτρα, που βρίσκεται στα αριστερά της φωτογραφίας και περιβάλλεται από το πράσινο πλαίσιο. Η πρώτη πρόβλεψη του μοντέλου μας, είναι πως το αντικείμενο ανήκει στην κατηγορία Ψύκτρα με ποσοστό σιγουριάς 44,9%. Η επόμενη πιθανή κατηγορία, που μπορεί να ανήκει το αντικείμενο αυτό είναι η κατηγορία Φως τύπου LED με ποσοστό μέτρησης 6,3%, ενώ η τελευταία πιθανή κατηγορία είναι η κατηγορία Διπολικό Τρανζίστορ με μέτρηση 3,1%. Από τα παραπάνω αποτελέσματα παρατηρούμε πως το μοντέλο μας έκανε σωστή πρόβλεψη για το πρώτο αντικείμενο.

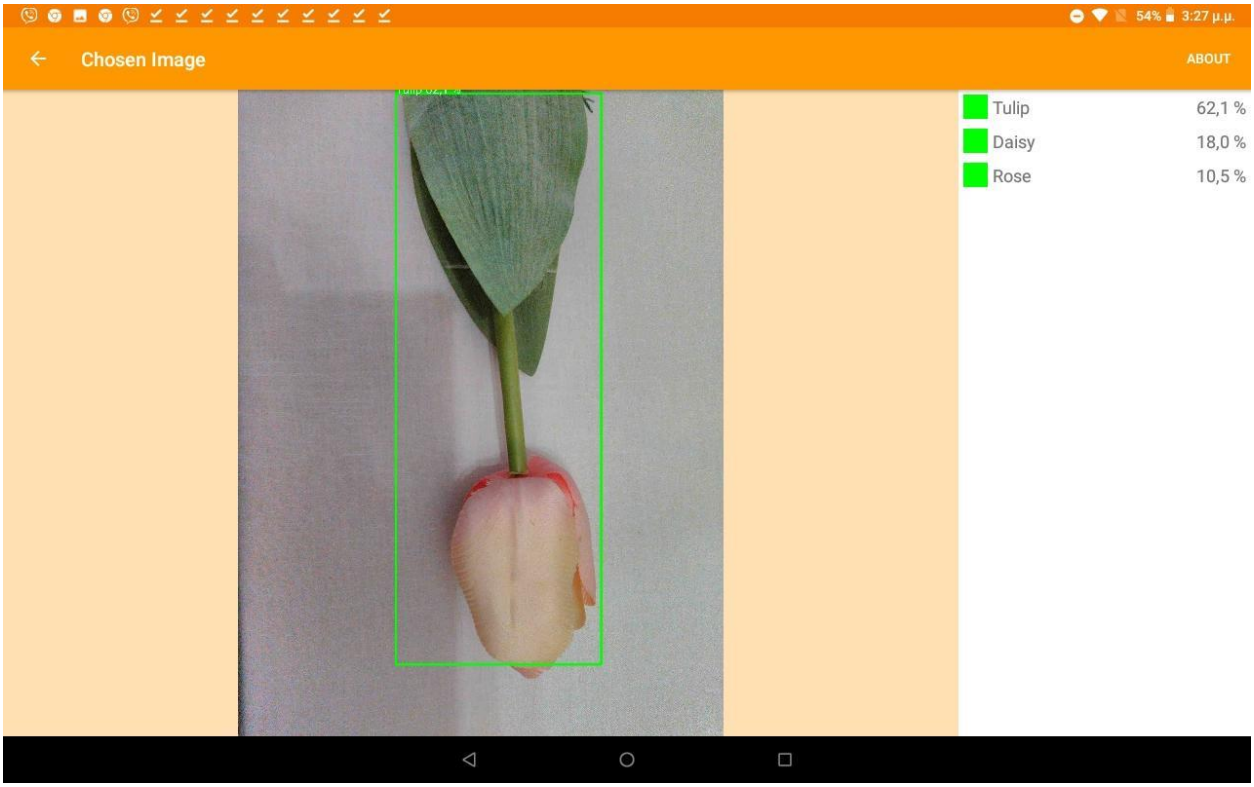
Στην περίπτωση του δεύτερου αντικειμένου, που βρίσκεται στα δεξιά της φωτογραφίας και περικλείεται από το κόκκινο πλαίσιο, το μοντέλο μας προέβλεψε πως ανήκει στην κατηγορία Ψύκτρα με ποσοστό σιγουριάς 60,5%, ενώ οι άλλες δύο πιθανές κατηγορίες είναι η κατηγορία Τρανζίστορ και η κατηγορία Φως τύπου LED και έχουν την ίδια μέτρηση από το μοντέλο μας με ποσοστό 7,0%. Με βάση τα αποτελέσματα αυτά, φτάνουμε στο συμπέρασμα πως το μοντέλο μας έκανε σωστή πρόβλεψη και για το δεύτερο αντικείμενο.

Τέλος, στη περίπτωση του τρίτου αντικειμένου που είναι στη μέση της φωτογραφίας και βρίσκεται μέσα στο μπλε πλαίσιο, η πρώτη πρόβλεψη του μοντέλου μας ήταν πως το αντικείμενο ανήκει στην κατηγορία Τρανζίστορ PnP με ποσοστό σιγουριάς 43,4%, ενώ οι επόμενες δύο πιθανές κατηγορίες έχουν ποσοστό σιγουριάς 8,6% και είναι η κατηγορία Ψύκτρα και η κατηγορία Διπολικό Τρανζίστορ. Συνεπώς το μοντέλο μας έκανε λάθος πρώτη πρόβλεψη για το τρίτο αντικείμενο, καθώς η σωστή πρόβλεψη είναι η δεύτερη.

5.3.3 Αποτελέσματα χρήσης μοντέλου για αναγνώριση λουλουδιών

Στην υποενότητα αυτή θα παρουσιάσουμε και θα αναλύσουμε τα αποτελέσματα που προέκυψαν από τη χρήση της εφαρμογής για ανίχνευση και αναγνώριση λουλουδιών.

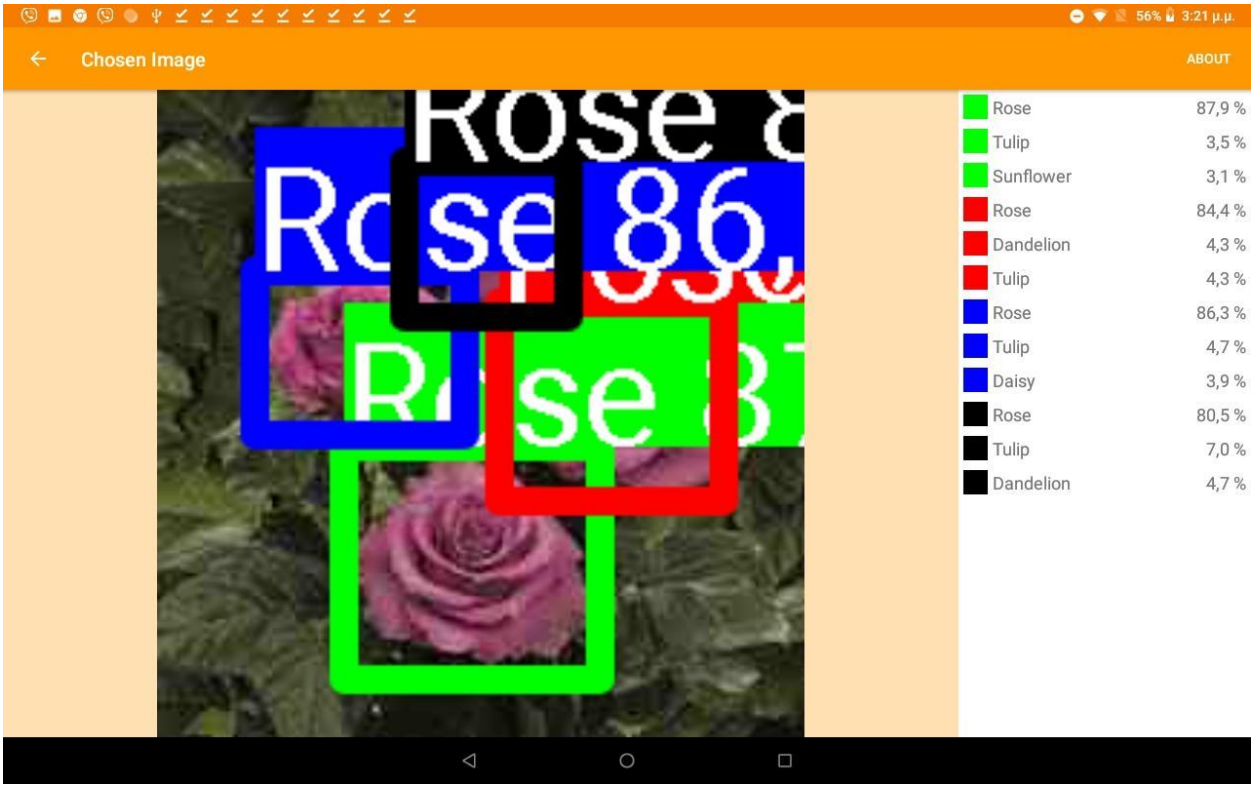
Εικόνα 5.23: Φωτογραφία μίας τουλίπας



Εικόνα 5.24: Αποτελέσματα Αναγνώρισης Αντικειμένου μέσω Κάμερας.

Στην εικόνα 5.23 βλέπουμε τη φωτογραφία μίας τουλίπας που τραβήξαμε με την κάμερα ενός τάμπλετ και στην εικόνα 5.24 βλέπουμε τα αποτελέσματα της ανίχνευσης και αναγνώρισης του μοντέλου μας. Παρατηρούμε πως η πρώτη πρόβλεψη που έκανε το μοντέλο μας για το αντικείμενο ήταν η κατηγορία Τουλίπα με ποσοστό σιγουριάς 62,1%. Η επόμενη κατηγορία που προβλέφθηκε ήταν η κατηγορία Μαργαρίτα με μέτρηση 18,0%, ενώ η τελευταία πιθανή κατηγορία είναι η κατηγορία Τριαντάφυλλο με ποσοστό 10,5%. Σύμφωνα με τα αποτελέσματα αυτά, το μοντέλο μας έκανε σωστή πρόβλεψη για το αντικείμενο που ανιχνεύθηκε στη φωτογραφία.

Εικόνα 5.25: Φωτογραφία με Τριαντάφυλλα.



Εικόνα 5.26: Αποτελέσματα Αναγνώρισης Πολλαπλών Αντικειμένων.

Στην εικόνα 5.25 βλέπουμε τη φωτογραφία που χρησιμοποιήσαμε για ανίχνευση πολλαπλών αντικειμένων και στην εικόνα 5.26 παρατηρούμε τα αποτελέσματα των προβλέψεων του μοντέλου μηχανικής μάθησης που χρησιμοποιεί η εφαρμογή μας. Επίσης η εφαρμογή μας ανίχνευσε και αναγνώρισε σωστά όλα τα λουλούδια στην εικόνα.

Πιο αναλυτικά, το πρώτο αντικείμενο που ανιχνεύθηκε είναι το αντικείμενο στο κάτω μέρος της εικόνας και περιβάλλεται από το πράσινο πλαίσιο. Η πρώτη πρόβλεψη που έγινε από το μοντέλο της εφαρμογής μας ήταν πως το αντικείμενο ανήκει στη κατηγορία Τριαντάφυλλο με ποσοστό σιγουριάς 87,9%. Η επόμενη πιθανή κατηγορία έχει ποσοστό 3,5% και είναι η κατηγορία Τουλίπα, ενώ η τελευταία είναι η κατηγορία Ήλιος με μέτρηση 3,1%.

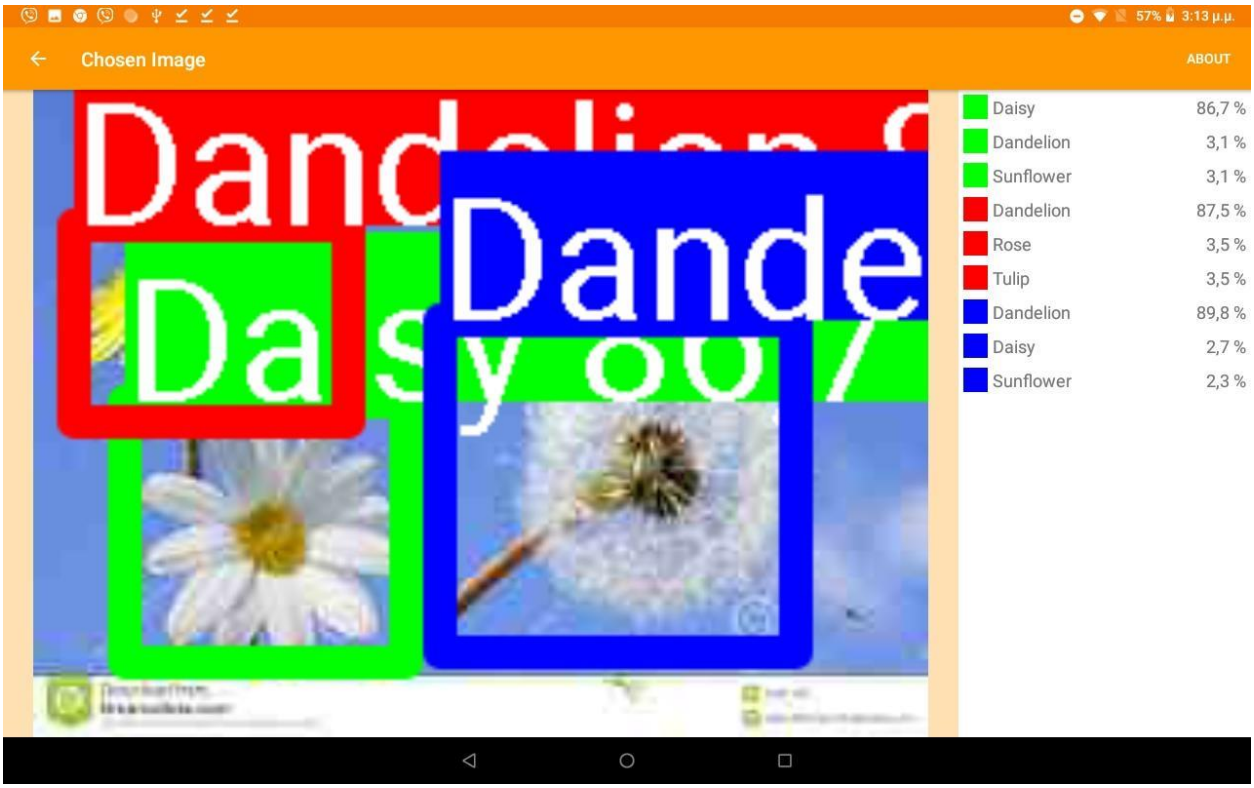
Το επόμενο αντικείμενο που αναγνωρίστηκε είναι αυτό που βρίσκεται στα δεξιά της φωτογραφίας και είναι μέσα στο κόκκινο πλαίσιο. Το μοντέλο προέβλεψε πως το αντικείμενο ανήκει κατά 84,4% στην κατηγορία Τριαντάφυλλο, ενώ οι επόμενες δύο προβλέψεις έχουν ποσοστό 4,3% και είναι οι κατηγορίες Πικραλίδα και Τουλίπα.

Το τρίτο αντικείμενο που ανίχνευσε η εφαρμογή μας, είναι το τριαντάφυλλο που περικλείεται από το μπλε πλαίσιο και είναι στην αριστερή πλευρά της φωτογραφίας που δώσαμε ως είσοδο. Η πρώτη κατηγορία που προέβλεψε το μοντέλο μας είναι η κατηγορία Τριαντάφυλλο με ποσοστό σιγουριάς 86,3%. Η δεύτερη πιθανή κατηγορία είναι η κατηγορία Τουλίπα με μέτρηση 4,7%, ενώ η τελευταία πιθανή κατηγορία που εμφανίζεται είναι η κατηγορία Μαργαρίτα με μέτρηση 3,9%.

Το τελευταίο αντικείμενο που ανιχνεύθηκε και αναγνωρίστηκε από το μοντέλο μηχανικής μάθησης που χρησιμοποιήσαμε είναι το αντικείμενο στο πάνω μέρος της φωτογραφίας, που βρίσκεται μέσα στο μαύρο πλαίσιο. Το μοντέλο προέβλεψε πως το αντικείμενο ανήκει στην κατηγορία Τριαντάφυλλο με ποσοστό σιγουριάς 80,5%, ενώ οι επόμενες δύο κατηγορίες είναι η κατηγορία Τουλίπα και η κατηγορία Πικραλίδα, με ποσοστά σιγουριάς 7,0% και 4,7% αντίστοιχα.

Στις παρακάτω εικόνες, βλέπουμε την φωτογραφία που χρησιμοποιήσαμε ως είσοδο για ανίχνευση και αναγνώριση λουλουδιών σε αυτή, αλλά και τα αποτελέσματα που προέκυψαν.

Εικόνα 5.27: Φωτογραφία Διαφορετικών Λουλουδιών.



Εικόνα 5.28: Αποτελέσματα Προβλέψεων για Διαφορετικά Λουλούδια.

Από τις δύο φωτογραφίες, καταλαβαίνουμε πως το μοντέλο μηχανικής μάθησης που χρησιμοποιήσαμε, ανίχνευσε και αναγνώρισε σωστά όλα τα λουλούδια της φωτογραφίας που διαλέξαμε.

Το πρώτο αντικείμενο που ανιχνεύθηκε περιβάλλεται από το πράσινο πλαίσιο στην εικόνα 5.28 και βρίσκεται στην κάτω – αριστερά πλευρά της φωτογραφίας. Η εφαρμογή μας προέβλεψε πως το αντικείμενο ανήκει στην κατηγορία Μαργαρίτα με ποσοστό σιγουριάς 86,7%, ενώ οι υπόλοιπες κατηγορίες που εμφανίζονται για το συγκεκριμένο αντικείμενο είναι η κατηγορία Πικραλίδα και η κατηγορία Ήλιος με ποσοστά σιγουριάς 3,1%.

Το επόμενο αντικείμενο που αναγνωρίστηκε βρίσκεται στην πάνω – αριστερά πλευρά της φωτογραφίας και είναι μέσα στο κόκκινο πλαίσιο. Η πρώτη πιθανή κατηγορία που προέβλεψε το μοντέλο της εφαρμογής μας είναι η κατηγορία Πικραλίδα με ποσοστό σιγουριάς 87,5%. Οι επόμενες πιθανές κατηγορίες έχουν και οι δύο ποσοστό 3,5% και είναι οι κατηγορίες Τριαντάφυλλο και Τουλίπα.

Το τελευταίο αντικείμενο της εικόνας περικλείεται από το μπλε πλαίσιο και βρίσκεται στη δεξιά πλευρά της φωτογραφίας. Το μοντέλο προέβλεψε πως το αντικείμενο ανήκει στην κατηγορία Πικραλίδα με ποσοστό σιγουριάς 89,8%. Η δεύτερη πιθανή κατηγορία είναι η κατηγορία Μαργαρίτα με μέτρηση 2,7%, ενώ η τελευταία πιθανή είναι η κατηγορία Ήλιος με ποσοστό σιγουριάς 2,3%.

Κεφάλαιο 6ο: Συμπέρασμα

6.1 Εισαγωγή

Το κεφάλαιο αυτό αποτελεί το τελευταίο κεφάλαιο της διπλωματικής εργασίας και σε αυτό θα αναφερθούμε στα προβλήματα που αντιμετωπίσαμε κατά την ανάπτυξη της εφαρμογής, στους περιορισμούς που είχαμε στο κομμάτι του λογισμικού καθώς και του υλικού. Επίσης θα αναφέρουμε και κάποιες πιθανές μελλοντικές επεκτάσεις που έχουμε να προτείνουμε για την εφαρμογή.

6.2 Προβλήματα κατά την Ανάπτυξη

Κατά την ανάπτυξη του λογισμικού αντιμετωπίσαμε κάποια προβλήματα τα οποία καθυστέρησαν την ολοκλήρωση της εφαρμογής.

Το πιο σημαντικό πρόβλημα που αντιμετωπίσαμε είχε σχέση με την εκπαίδευση των μοντέλων μηχανικής μάθησης που χρησιμοποιούμε στην εφαρμογή μας. Η αρχική μας ιδέα ήταν η δημιουργία μοντέλων με χρήση της γλώσσας Python και των βιβλιοθηκών Keras και Tensorflow. Η εκπαίδευση θα ήταν της μορφής Supervised Learning (εποπτευόμενη μάθηση) και δοκιμάστηκε η τεχνική Transfer Learning (μεταφορά μάθησης). Έπειτα από έρευνα και δοκιμές που κάναμε χρησιμοποιήσαμε το μοντέλο EfficientNet B0 για βάση του μοντέλου μας και με τη χρήση της ιστοσελίδας GitHub βρήκαμε ένα κομμάτι κώδικα για τη δημιουργία των μεταδεδομένων του μοντέλου. Κατά τη χρήση του συγκεκριμένου μοντέλου με την εφαρμογή μας παρατηρήθηκαν λάθος αποτελέσματα με σταθερά ποσοστά. Για παράδειγμα το μοντέλο μηχανικής μάθησης που χρησιμοποιήσαμε είχε εκπαιδευτεί με το σύνολο δεδομένων CIFAR – 100 και κάθε αντικείμενο που αναγνώριζε η εφαρμογή, με χρήση του Object Detector που έχει ενσωματωμένο το Google ML Kit, αρχικά επέστρεφε ως την κατηγορία που ανήκε, την κλάση Worm με σταθερό το ποσοστό σιγουριάς και τιμή 10.9%. Μετά από αρκετές αλλαγές και μετατροπές στους κώδικες για την εκπαίδευση του μοντέλου και την δημιουργία των μεταδεδομένων του, καταφέραμε να αλλάξουμε την κλάση που επιστρεφόταν στην κατηγορία Willow Tree με το ίδιο ποσοστό. Έπειτα από αρκετές μέρες αποφασίσαμε να δοκιμάσουμε την πλατφόρμα Google Cloud Platform και την υπηρεσία Google Cloud Auto ML Vision που διαθέτει, για την εκπαίδευση του αρχικού μοντέλου μηχανικής μάθησης, και με τη χρήση τους μέσω της εφαρμογής μας ήμασταν ευχαριστημένοι με το αποτέλεσμα και αποφασίσαμε να γίνει η εκπαίδευση και των υπόλοιπων μοντέλων μηχανικής μάθησης με χρήση της υπηρεσίας που αναφέραμε.

Ένα άλλο πρόβλημα που αντιμετωπίσαμε αφορούσε τη χρήση του Google ML Kit και πιο συγκεκριμένα την επιλογή χρήσης δικού μας Object Detector. Επειδή το συγκεκριμένο API βρίσκεται σε Beta έκδοση δεν είναι δυνατή η χρήση ενός προσαρμοσμένου Ανιχνευτή Αντικειμένων, τη χρονική στιγμή που εκπονούμε τη συγκεκριμένη διπλωματική εργασία, το οποίο δυστυχώς δεν αναφέρεται στο επίσημο documentation του API. Αυτό μας έγινε γνωστό μέσω της ιστοσελίδας stackoverflow.com και απαντήθηκε από ένα μηχανικό της Google που εργάζεται πάνω στο συγκεκριμένο API. Για αυτό το λόγο αρκετά αντικείμενα, κυρίως ηλεκτρονικά είδη, η εφαρμογή μας δεν μπορεί να τα αντιληφθεί ως αντικείμενα και συνεπώς δεν μπορεί να γίνει η αναγνώρισή τους.

Ακόμα ένα πρόβλημα που αντιμετωπίσαμε κατά την ανάπτυξη του λογισμικού μας αφορούσε τον προσανατολισμό της συσκευής. Οι συσκευές με Android λογισμικό ακολουθούν μία συγκεκριμένη διαδικασία κατά το άνοιγμα της εφαρμογής ή μια καινουργίας δραστηριότητας. Κατά την περιστροφή μιας συσκευής η εφαρμογή εμφανίζεται κανονικά στον χρήστη, αλλά το σύστημα πίσω από την εφαρμογή διακόπτει τη δραστηριότητα που έβλεπε μέχρι πρότινος ο χρήστης και δημιουργεί καινούργια με τον προσανατολισμό που έχει εκείνη τη στιγμή η συσκευή. Για την καλύτερη εμπειρία του χρήστη θέλουμε να οι επιλογές που είχαν επιλεχθεί και τα δεδομένα που είχαν εμφανιστεί να μείνουν ως έχουν και να μη γίνουν νέες κινήσεις από το χρήστη ή να χρειαστεί να ξανατρέξει η εφαρμογή. Αυτό το πρόβλημα έχει τρεις λύσεις. Η πρώτη λύση είναι να προσθέσουμε στο μανιφέστο του κώδικά μας κάποιες εντολές για να κάνουμε την οθόνη σταθερή έτσι ώστε να μην μπορεί να αλλάξει ο προσανατολισμός της. Η δεύτερη λύση είναι να χρησιμοποιήσουμε τις συναρτήσεις onPause(), onStop(), onStart() και onResume() για τη μεταφορά των πληροφοριών. Εμείς επιλέξαμε την τρίτη λύση που ήταν να προσαρμόσουμε την εφαρμογή μας σε αρχιτεκτονική Model – View - ViewModel (MVVM). Το πρόβλημα όμως προέκυψε με το συνδυασμό του Google ML Kit που χρησιμοποιήσαμε για την ανίχνευση των αντικειμένων και της αρχιτεκτονικής MVVM. Ο τρόπος που αναγνωρίζει η εφαρμογή μας τα αντικείμενα είναι ως εξής. Αρχικά ο Object Detector ελέγχει την εικόνα για αντικείμενα και σε περίπτωση που βρεθεί έστω και ένα δημιουργείται ένα task. Έπειτα ανά task δημιουργούμε μία νέα εικόνα με τις συντεταγμένες που μας έδωσε ο Ανιχνευτής Αντικειμένων και με χρήση των εικόνων αυτών και των μοντέλων μηχανικής μάθησης που έχουμε εκπαιδεύσει εμφανίζουμε πιθανές κατηγορίες που ανήκουν τα αντικείμενα αυτά. Τέλος, σε περίπτωση που βρεθούν πιθανές κατηγορίες, δημιουργούμε νέα task και προσθέτουμε τις πληροφορίες που έχουμε συλλέξει σε μία λίστα για να τις μεταφέρουμε στο ViewModel κομμάτι της αρχιτεκτονικής μας. Το πρόβλημα που αντιμετωπίσαμε ήταν πως όταν, στη φωτογραφία που είχαμε επιλέξει για αναγνώριση αντικειμένων, ο Object Detector έβρισκε πάνω από ένα αντικείμενο, κατά την περιστροφή έμενε αποθηκευμένο μόνο το πρώτο αντικείμενο που αναγνωρίστηκε και για τα υπόλοιπα ξαναέτρεχε το πρόγραμμα. Η λύση που βρήκαμε ήταν να προσθέσουμε τα δεδομένα που είχαν βρεθεί στο bundle savedInstanceState και να κάνουμε έλεγχο σχετικά με την κατάστασή του, κατά την διάρκεια της συνάρτησης onCreate() που χρησιμοποιούμε για να καλέσουμε τη συνάρτηση που αναγνωρίζει τα αντικείμενα μιας εικόνας.

Επίσης ένα ακόμα πρόβλημα που αντιμετωπίσαμε, κατά την ανάπτυξη της εφαρμογής μας, αφορούσε τη χρήση κάμερας για την αναγνώριση των αντικειμένων. Αρχικά η φωτογραφία που επέστρεφε η δραστηριότητα της κάμερας, ήταν μια μικρογραφία της κανονικής και πιο συγκεκριμένα το thumbnail της. Η λύση που εφαρμόσαμε ήταν να αποθηκεύουμε τη φωτογραφία στον αποθηκευτικό χώρο της συσκευής και έπειτα να χρησιμοποιούμε την αποθηκευμένη φωτογραφία για την αναγνώριση. Το πρόβλημα που προέκυψε έπειτα είχε να κάνει με το μέγεθος των εικόνων. Ο τρόπος που είχαμε επιλέξει για την μεταφορά των εικόνων στη δραστηριότητα ChosenImage είχε έναν περιορισμό όσον αφορά το μέγεθος των δεδομένων, και η λύση που επιλέξαμε στο πρόβλημα αυτό ήταν να μειώσουμε στο 25% την ποιότητα της εικόνας πριν μεταφερθεί στην επόμενη δραστηριότητα. Οι κάμερες στις σύγχρονες κινητές συσκευές έχουν τη δυνατότητα να τραβάνε φωτογραφίες με ένα πολύ μεγάλο αριθμό Pixel. Για παράδειγμα η κινητή συσκευή που χρησιμοποιήσαμε στον έλεγχο της εφαρμογής μας, έχει τη δυνατότητα να τραβήξει φωτογραφίες 48 Megapixel. Για να αντιμετωπίσουμε αυτό το πρόβλημα αρχικά ελέγξαμε φωτογραφίες με μικρή ανάλυση και έπειτα ελέγξαμε τις ίδιες φωτογραφίες με μειωμένη την ποιότητα στα ποσοστά 75%, 50% και 25%. Κατά τον έλεγχο αυτόν δεν παρατηρήθηκαν αλλαγές στις προβλέψεις των μοντέλων μηχανικής μάθησης. Συνεπώς αποφασίσαμε να χρησιμοποιήσουμε τις εικόνες με ποιότητα 25% για να μην συναντήσουμε κάποιο περιορισμό με την χρήση της κάμερας για την αναγνώριση αντικειμένων στην εφαρμογή μας.

Το τελευταίο πρόβλημα που αντιμετωπίσαμε δημιουργήθηκε με τη λύση που είχαμε για τη χρήση της κάμερας μέσω της εφαρμογής. Παραπάνω αναφέραμε πως όταν χρησιμοποιούμε την κάμερα, αποθηκεύουμε τη φωτογραφία στον αποθηκευτικό χώρο. Στην περίπτωση όμως που ο χρήστης επέστρεφε στην αρχική οθόνη χωρίς να έχει χρησιμοποιήσει την κάμερα για να τραβήξει κάποια φωτογραφία, η εφαρμογή δημιουργούσε ένα αρχείο εικόνας με μέγεθος 0 Byte. Η λύση που βρήκαμε ήταν η προσθήκη ελέγχου για κενά αρχεία. Για αυτό χρησιμοποιούμε τη συνάρτηση checkForBlanks() και με τη χρήση του Uri της εικόνας, ελέγχουμε το μέγεθός της. Εάν το μέγεθός της είναι 0 Byte διαγράφουμε το αρχείο, ενώ σε αντίθετη περίπτωση δεν κάνουμε κάποια ενέργεια. Επίσης η συνάρτηση checkForBlanks() καλείται μόνο στην περίπτωση της χρήσης κάμερας, καθώς στην άλλη περίπτωση, δηλαδή της επιλογής εικόνας από τον αποθηκευτικό χώρο, δεν αντιμετωπίσαμε αυτό το πρόβλημα.

6.3 Περιορισμοί της Εφαρμογής

Στην ενότητα αυτή θα αναφέρουμε και θα αναλύσουμε τους περιορισμούς που είχαμε κατά τη διάρκεια της ανάπτυξης της εφαρμογής μας. Οι περιορισμοί χωρίζονται σε δύο κατηγορίες, στους υλικούς και τους τεχνολογικούς.

Οι υλικοί περιορισμοί αφορούν τους περιορισμούς που υπάρχουν όσον αφορά τα διαθέσιμα υλικά. Στην περίπτωση μας ο κύριος υλικός περιορισμός είναι η διάθεση των κινητών συσκευών που έχουν ως λειτουργικό σύστημα, το σύστημα Android. Στη διάθεσή μας υπήρχαν μόνο δύο διαφορετικές συσκευές, οι συσκευές που χρησιμοποιήθηκαν και στην ενότητα 5.3 που αφορούσε τα αποτελέσματα πραγματικής χρήσης. Δυστυχώς μία εφαρμογή δεν μπορεί να κριθεί ως έτοιμη μόνο με τη χρήση της ψηφιακής συσκευής που παρέχει το εργαλείο Android Studio ή με χρήση λίγων φυσικών συσκευών. Ο προτεινόμενος έλεγχος είναι με την χρήση φυσικών συσκευών με διαφορετικές αναλογίες, αναλύσεις και διαστάσεις στις οθόνες τους, αλλά και φυσικές συσκευές με διαφορετική έκδοση του λειτουργικού συστήματος Android και συσκευές διαφορετικών κατασκευαστών με σκοπό να έχει γίνει έλεγχος σε συσκευές ενός μεγάλου εύρους.

Από την άλλη πλευρά οι τεχνολογικοί περιορισμοί είναι οι περιορισμοί που υπάρχουν στην ανάπτυξη του λογισμικού. Ο κυριότερος τεχνολογικός περιορισμός που είχαμε αφορούσε το API Google ML Kit που χρησιμοποιήσαμε. Το API αυτό, κατά την περίοδο συγγραφής και εκπόνησης της διπλωματικής εργασίας, βρίσκεται σε beta έκδοση και αρκετές λειτουργίες δεν είναι διαθέσιμες. Η λειτουργία της χρήσης προσαρμοσμένων μοντέλων μηχανικής μάθησης για ανίχνευση αντικειμένων, δηλαδή Custom Object Detection Models, που θέλαμε να χρησιμοποιήσουμε δεν ήταν διαθέσιμη και αναγκαστήκαμε να χρησιμοποιούσαμε το ενσωματωμένο μοντέλο. Το ενσωματωμένο αυτό μοντέλο, ήταν χρήσιμο για την ανίχνευση συνηθισμένων αντικειμένων και λουλουδιών, δηλαδή για χρήση με τα μοντέλα μηχανικής μάθησης εκπαιδευμένα με το CIFAR – 100 σύνολο δεδομένων και το dataset με τους πέντε τύπους λουλουδιών αντίστοιχα, αλλά ήταν δύσχρηστο για την ανίχνευση ηλεκτρονικών ειδών, κυρίως λόγω των περίεργων διαστάσεων και μεγεθών τους.

6.4 Μελλοντικές Επεκτάσεις

Στην ενότητα αυτή θα αναφέρουμε κάποιες πιθανές μελλοντικές επεκτάσεις της εφαρμογής ως προς τη χρήση και τις επιλογές που παρέχει αυτή στον χρήστη.

Η κύρια δυνατότητα που θα θέλαμε να προστεθεί είναι η αναγνώριση αντικειμένων σε πραγματικό χρόνο. Πιο αναλυτικά ο χρήστης δεν θα χρειαζόταν να τραβήξει τη φωτογραφία ενός αντικειμένου, αλλά με το πάτημα ενός κουμπιού θα άνοιγε η δραστηριότητα της κάμερας στην οθόνη και τα πλαίσια αναγνώρισης θα εμφανίζονταν πάνω στο αντικείμενο. Αυτή τη δυνατότητα την παρέχουν οι εφαρμογές Google Keep και Aipoly Vision.

Επίσης, θα θέλαμε να βελτιώσουμε τη διαδικασία εμφάνισης του πλαισίου γύρω από τα αντικείμενα της εικόνας, καθώς εξαρτάται από την ανάλυση της φωτογραφίας. Η καλύτερη επιλογή θα είναι να υπάρχει συγκεκριμένο πάχος του πλαισίου με διαφορετικές τοποθεσίες για την εμφάνιση του τίτλου της κατηγορίας του αντικειμένου.

Ακόμα θα θέλαμε να δώσουμε τη δυνατότητα στον χρήστη εκχώρησης δικών του μοντέλων μηχανικής μάθησης για να προσαρμόζεται η εφαρμογή στις ανάγκες του. Για να γίνει αυτό θα πρέπει να αλλάξει ο τρόπος επιλογής μοντέλου μηχανικής μάθησης που έχουμε.

Τέλος θα θέλαμε να δημιουργήσουμε μια δραστηριότητα για επιλογές. Σε αυτή τη δραστηριότητα θα μπορέσουμε για παράδειγμα να προσθέσουμε την επιλογή για την αυτόματη ή χειροκίνητη αποθήκευση των φωτογραφιών με αναγνωρισμένα αντικείμενα στον αποθηκευτικό χώρο της συσκευής, καθώς επίσης κάποιες ακόμα επιλογές που θα μπορούσαμε να προσθέσουμε είναι η αλλαγή θέματος ή χρωμάτων της εφαρμογής, η αλλαγή της γραμματοσειράς και η δυνατότητα κλειδώματος της περιστροφής της εφαρμογής.

Βιβλιογραφία

1. Christos Troussas, Akrivi Krouska, Cleo Sgouropoulou: Collaboration and fuzzy-modeled personalization for mobile game-based learning in higher education. Computers & Education, 144 (2020).
2. Christos Troussas, Akrivi Krouska, Cleo Sgouropoulou, Ioannis Voyiatzis: Ensemble Learning Using Fuzzy Weights to Improve Learning Style Identification for Adapted Instructional Routines. Entropy 22(7): 735 (2020)
3. Akrivi Krouska, Christos Troussas, Cleo Sgouropoulou: A Personalized Brain-Based Quiz Game for Improving Students' Cognitive Functions. BFAL 2020: 102-106
4. Papakostas, C., Troussas, C., Krouska, A. & Sgouropoulou, C. User acceptance of augmented reality welding simulator in engineering training. Educ Inf Technol (2021). https://doi.org/10.1007/s10639-020-10418-7
5. Troussas, C., Krouska, A. & Sgouropoulou, C. Impact of social networking for advancing learners' knowledge in E-learning environments. Educ Inf Technol (2021). https://doi.org/10.1007/s10639-021-10483-6
6. Krouska A., Troussas C., Sgouropoulou C. (2020) Applying Genetic Algorithms for Recommending Adequate Competitors in Mobile Game-Based Learning Environments. In: Kumar V., Troussas C. (eds) Intelligent Tutoring Systems. ITS 2020. Lecture Notes in Computer Science, vol 12149. Springer, Cham. https://doi.org/10.1007/978-3-030-49663-0_23
7. Christos Troussas, Cleo Sgouropoulou: Innovative Trends in Personalized Software Engineering and Information Systems - The Case of Intelligent and Adaptive E-learning Systems. Frontiers in Artificial Intelligence and Applications 324, IOS Press 2020, ISBN 978-1-64368-096-5, pp. 1-96
8. Troussas C., Krouska A., Sgouropoulou C. (2020) Towards a Reference Model to Ensure the Quality of Massive Open Online Courses and E-Learning. In: Frasson C., Bamidis P., Vlamos P. (eds) Brain Function Assessment in Learning. BFAL 2020. Lecture Notes in Computer Science, vol 12462. Springer, Cham. https://doi.org/10.1007/978-3-030-60735-7_18
9. Troussas C., Krouska A., Sgouropoulou C. (2020) Dynamic Detection of Learning Modalities Using Fuzzy Logic in Students' Interaction Activities. In: Kumar V., Troussas C. (eds) Intelligent Tutoring Systems. ITS 2020. Lecture Notes in Computer Science, vol 12149. Springer, Cham. https://doi.org/10.1007/978-3-030-49663-0_24
10. Christos Troussas, Akrivi Krouska, Filippos Giannakas, Cleo Sgouropoulou, Ioannis Voyiatzis: Automated reasoning of learners' cognitive states using classification analysis. PCI 2020-24th Pan-Hellenic Conference on Informatics, November 2020, Pages 103–106. https://doi.org/10.1145/3437120.3437285
11. Christos Troussas, Akrivi Krouska, Filippos Giannakas, Cleo Sgouropoulou, Ioannis Voyiatzis: Redesigning teaching strategies through an information filtering system. PCI 2020-24th Pan-Hellenic Conference on Informatics, November 2020, Pages 111-114. https://doi.org/10.1145/3437120.3437287
12. Akrivi Krouska, Christos Troussas, Cleo Sgouropoulou: Usability and Educational Affordance of Web 2.0 tools from Teachers' Perspectives. PCI 2020-24th Pan-Hellenic Conference on Informatics, November 2020, Pages 107-110. https://doi.org/10.1145/3437120.3437286
13. Christos Troussas, Filippos Giannakas, Cleo Sgouropoulou & Ioannis Voyiatzis (2020) Collaborative activities recommendation based on students' collaborative learning styles using ANN and WSM, Interactive Learning Environments, DOI: 10.1080/10494820.2020.1761835

14. F. Giannakas, C. Troussas, I. Voyiatzis, C. Sgouropoulou, A deep learning classification framework for early prediction of team-based academic performance, Applied Soft Computing, Volume 106, 2021, 107355. https://doi.org/10.1016/j.asoc.2021.107355.
15. Christos PAPAKOSTAS, Christos TROUSSAS, Akrivi KROUSKA, Cleo SGOUROPOULOU, Exploration of Augmented Reality in Spatial Abilities Training: A Systematic Literature Review for the Last Decade, Informatics in Education 20 (2021), no. 1, 107-130, DOI 10.15388/infedu.2021.06
16. C. Troussas, A. Krouska and C. Sgouropoulou, "A Novel Teaching Strategy Through Adaptive Learning Activities for Computer Programming," in IEEE Transactions on Education, vol. 64, no. 2, pp. 103-109, May 2021, doi: 10.1109/TE.2020.3012744.
17. C. Troussas, A. Krouska, E. Alepis & M. Virvou (2021) Intelligent and adaptive tutoring through a social network for higher education, New Review of Hypermedia and Multimedia, DOI: 10.1080/13614568.2021.1908436
18. Troussas, C., Krouska, A. & Virvou, M. A multilayer inference engine for individualized tutoring model: adapting learning material and its granularity. Neural Comput & Applic (2021). https://doi.org/10.1007/s00521-021-05740-1
19. C. Troussas, M. Virvou, J. Caro, K. J. Espinosa, Language Learning Assisted by Group Profiling in Social Networks, International Journal of Emerging Technologies in Learning (iJET), Vol 8, No 3, pp. 35-38, 2013.
20. C. Troussas, M. Virvou, K. J. Espinosa, Using Visualization Algorithms for Discovering Patterns in Groups of Users for Tutoring Multiple Languages through Social Networking. Journal of Networks 10(12): 668-674 (2015).

Ηλεκτρονική Βιβλιογραφία:

1. Android Developers Website. Διαθέσιμο στον ιστότοπο: https://developer.android.com/
2. Σύνολο Δεδομένων CIFAR – 100.
   Διαθέσιμο στον ιστότοπο: https://www.cs.toronto.edu/~kriz/cifar.html
3. Σύνολο Δεδομένων για Ηλεκτρονικά Είδη.
   Διαθέσιμο στον ιστότοπο: https://www.kaggle.com/aryaminus/electronic-components
4. Σύνολο Δεδομένων για τύπους Λουλουδιών.
   Διαθέσιμο στον ιστότοπο: https://www.kaggle.com/alxmamaev/flowers-recognition
5. Google Cloud Platform Website. Διαθέσιμο στον ιστότοπο: https://cloud.google.com/
6. Google ML Kit API Website. Διαθέσιμο στον ιστότοπο: https://developers.google.com/ml-kit
7. Γλώσσα Προγραμματισμού Kotlin. Διαθέσιμη στον ιστότοπο: https://kotlinlang.org/
8. Stack Overflow Website. Διαθέσιμο στον ιστότοπο: https://stackoverflow.com/
9. Lucid Chart. Διαθέσιμο στον ιστότοπο: https://lucid.app/
10. Google Images. Διαθέσιμο στον ιστότοπο: https://www.google.com/imghp?hl=EN
11. Elgin, Ben. “Google Buys Android for Its Mobile Arsenal”, Bloomberg Businessweek, 17 Αυγούστου 2005, https://web.archive.org/web/20110205190729/http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm
12. Manjoo, Farhat. “A Murky Road Ahead for Android, Despite Market Dominance”, New York Times, 27 Μαΐου 2015, https://web.archive.org/web/20170706094446/https://www.nytimes.com/2015/05/28/technology/personaltech/a-murky-road-ahead-for-android-despite-market-dominance.html

13. Nabeel. “THE 12 BEST IMAGE RECOGNITION APPS THAT YOU SHOULD TRY IN 2021”, TekRevol, 11 Μαρτίου 2021, https://www.tekrevol.com/blogs/best-image-recognition-apps/
14. Google Lens. Διαθέσιμο στον ιστότοπο: https://lens.google.com/
15. Aipoly Vision. Διαθέσιμο στον ιστότοπο: https://www.aipoly.com/
16. TapTapSee. Διαθέσιμο στον ιστότοπο: https://taptapseeapp.com/
17. Calorie Mama AI. Διαθέσιμο στον ιστότοπο: https://www.caloriemama.ai/
18. Gavrilova Yulia, Bolgurtseva Olga. “What Is Data Preprocessing in ML?”, Serokell, 23 Σεπτεμβρίου 2020, https://serokell.io/blog/data-preprocessing
19. McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*. Microsoft Press. “Waterfall vs. Agile Methodology.” 2008. Agile Introduction for Dummies. Retrieved August 13, 2010, from http://agileintro.wordpress.com/2008/01/04/waterfall-vs-agile-methodology/

Παράρτημα

Αρχείο AndroidManifest.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.      package="com.xarhssta.objectrecognition">
4.
5.      <uses-feature
6.          android:name="android.hardware.camera2"
7.          android:required="true" />
8.
9.      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
10.     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
11.     <uses-permission android:name="android.permission.CAMERA" />
12.
13.     <queries>
14.         <intent>
15.             <action android:name="android.media.action.IMAGE_CAPTURE" />
16.         </intent>
17.     </queries>
18.
19.     <application
20.         android:allowBackup="true"
21.         android:icon="@mipmap/launcher"
22.         android:label="@string/app_name"
23.         android:roundIcon="@mipmap/launcher_round"
24.         android:supportsRtl="true"
25.         android:theme="@style/Theme.ObjectRecognition">
26.         <activity
27.             android:name=".MainActivity"
28.             android:label="@string/app_name"
29.             android:theme="@style/Theme.ObjectRecognition.NoActionBar">
30.             <intent-filter>
31.                 <action android:name="android.intent.action.MAIN" />
32.
33.                 <category android:name="android.intent.category.LAUNCHER" />
34.             </intent-filter>
35.         </activity>
36.         <activity
37.             android:name=".ChosenImage"
38.             android:label="@string/title_activity_chosen_image"
39.             android:parentActivityName=".MainActivity"
40.             android:theme="@style/Theme.ObjectRecognition.NoActionBar">
41.             <meta-data
42.                 android:name="android.support.PARENT_ACTIVITY"
43.                 android:value=".MainActivity" />
44.         </activity>
45.         <provider
46.             android:name="androidx.core.content.FileProvider"
47.             android:authorities="com.example.android.fileprovider"
48.             android:exported="false"
49.             android:grantUriPermissions="true">
50.             <meta-data
51.                 android:name="android.support.FILE_PROVIDER_PATHS"
52.                 android:resource="@xml/file_paths"></meta-data>
53.         </provider>
54.     </application>
55.
```

```
</manifest>
```

Αρχείο activity about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/appIconImageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@mipmap/launcher_round" />

    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:text="@string/app_name"
        android:textAlignment="center"
        android:textSize="20sp"
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toTopOf="@+id/authorTextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/appIconImageView"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/descriptionTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="@string/app_description"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/appIconImageView" />

    <TextView
        android:id="@+id/authorTextView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:text="@string/app_creator"
        android:textAlignment="center"
        app:layout_constraintBottom_toTopOf="@+id/descriptionTextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toEndOf="@+id/appIconImageView"
        app:layout_constraintTop_toBottomOf="@+id/titleTextView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Αρχείο activity chosen image.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:app="http://schemas.android.com/apk/res-auto"
4.      xmlns:tools="http://schemas.android.com/tools"
5.      android:id="@+id/coordinatorLayout"
6.      android:layout_width="match_parent"
7.      android:layout_height="match_parent"
8.      android:background="@color/primary_light"
9.      tools:context=".ChosenImage">
10.
11.     <com.google.android.material.appbar.AppBarLayout
12.         android:id="@+id/appBarLayout"
13.         android:layout_width="0dp"
14.         android:layout_height="wrap_content"
15.         android:theme="@style/Theme.ObjectRecognition.AppBarOverlay"
16.         app:layout_constraintEnd_toEndOf="parent"
17.         app:layout_constraintStart_toStartOf="parent"
18.         app:layout_constraintTop_toTopOf="parent">
19.
20.         <androidx.appcompat.widget.Toolbar
21.             android:id="@+id/toolbar"
22.             android:layout_width="match_parent"
23.             android:layout_height="wrap_content"
24.             android:background="?attr/colorPrimary"
25.             android:minHeight="?attr/actionBarSize"
26.             android:theme="?attr/actionBarTheme" />
27.     </com.google.android.material.appbar.AppBarLayout>
28.
29.     <ImageView
30.         android:id="@+id/chosenImageView"
31.         android:layout_width="0dp"
32.         android:layout_height="0dp"
33.         android:scaleType="fitCenter"
34.         app:layout_constraintBottom_toTopOf="@+id/itemList"
35.         app:layout_constraintEnd_toEndOf="parent"
36.         app:layout_constraintStart_toStartOf="parent"
37.         app:layout_constraintTop_toBottomOf="@+id/appBarLayout"
38.         app:srcCompat="@drawable/outline_insert_photo_24" />
39.
40.     <androidx.recyclerview.widget.RecyclerView
41.         android:id="@+id/itemList"
42.         android:layout_width="0dp"
43.         android:layout_height="200dp"
44.         android:background="@color/white"
45.         android:visibility="visible"
46.         app:layout_constraintBottom_toBottomOf="parent"
47.         app:layout_constraintEnd_toEndOf="parent"
48.         app:layout_constraintHorizontal_bias="0.0"
49.         app:layout_constraintStart_toStartOf="parent"
50.         app:layout_constraintTop_toBottomOf="@+id/chosenImageView" />
51.
52. </androidx.constraintlayout.widget.ConstraintLayout>
53.
```

Αρχείο activity chosen image.xml (landscape)

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:app="http://schemas.android.com/apk/res-auto"
4.      xmlns:tools="http://schemas.android.com/tools"
5.      android:id="@+id/coordinatorLayout"
6.      android:layout_width="match_parent"
7.      android:layout_height="match_parent"
8.      android:background="@color/primary_light"
9.      tools:context=".ChosenImage">
10.
11.     <com.google.android.material.appbar.AppBarLayout
12.         android:id="@+id/appBarLayout"
13.         android:layout_width="0dp"
14.         android:layout_height="wrap_content"
15.         android:theme="@style/Theme.ObjectRecognition.AppBarOverlay"
16.         app:layout_constraintEnd_toEndOf="parent"
17.         app:layout_constraintStart_toStartOf="parent"
18.         app:layout_constraintTop_toTopOf="parent">
19.
20.         <androidx.appcompat.widget.Toolbar
21.             android:id="@+id/toolbar"
22.             android:layout_width="match_parent"
23.             android:layout_height="wrap_content"
24.             android:background="?attr/colorPrimary"
25.             android:minHeight="?attr/actionBarSize"
26.             android:theme="?attr/actionBarTheme" />
27.     </com.google.android.material.appbar.AppBarLayout>
28.
29.     <ImageView
30.         android:id="@+id/chosenImageView"
31.         android:layout_width="0dp"
32.         android:layout_height="0dp"
33.         android:scaleType="fitCenter"
34.         app:layout_constraintBottom_toBottomOf="parent"
35.         app:layout_constraintEnd_toStartOf="@+id/itemList"
36.         app:layout_constraintStart_toStartOf="parent"
37.         app:layout_constraintTop_toBottomOf="@+id/appBarLayout"
38.         app:srcCompat="@drawable/outline_insert_photo_24" />
39.
40.     <androidx.recyclerview.widget.RecyclerView
41.         android:id="@+id/itemList"
42.         android:layout_width="300dp"
43.         android:layout_height="0dp"
44.         android:background="@color/white"
45.         android:visibility="visible"
46.         app:layout_constraintBottom_toBottomOf="parent"
47.         app:layout_constraintEnd_toEndOf="parent"
48.         app:layout_constraintStart_toEndOf="@+id/chosenImageView"
49.         app:layout_constraintTop_toBottomOf="@+id/appBarLayout" />
50.
51. </androidx.constraintlayout.widget.ConstraintLayout>
```

Αρχείο activity_main.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:app="http://schemas.android.com/apk/res-auto"
4.      xmlns:tools="http://schemas.android.com/tools"
5.      android:layout_width="match_parent"
6.      android:layout_height="match_parent"
7.      tools:context=".MainActivity">
8.
9.      <com.google.android.material.appbar.AppBarLayout
10.         android:layout_width="match_parent"
11.         android:layout_height="wrap_content"
12.         android:theme="@style/Theme.ObjectRecognition.AppBarOverlay">
13.
14.         <androidx.appcompat.widget.Toolbar
15.             android:id="@+id/toolbar"
16.             android:layout_width="match_parent"
17.             android:layout_height="?attr/actionBarSize"
18.             android:background="?attr/colorPrimary"
19.             app:popupTheme="@style/Theme.ObjectRecognition.PopupOverlay" />
20.
21.     </com.google.android.material.appbar.AppBarLayout>
22.
23.     <include layout="@layout/content_main" />
24.
25. </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Αρχείο content_main.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:app="http://schemas.android.com/apk/res-auto"
4.      xmlns:tools="http://schemas.android.com/tools"
5.      android:layout_width="match_parent"
6.      android:layout_height="match_parent"
7.      android:background="@color/primary_light"
8.      android:backgroundTint="@color/primary_light"
9.      app:layout_behavior="@string/appbar_scrolling_view_behavior">
10.
11.     <Button
12.         android:id="@+id/selectButton"
13.         android:layout_width="wrap_content"
14.         android:layout_height="wrap_content"
15.         android:layout_marginStart="8dp"
16.         android:layout_marginTop="32dp"
17.         android:layout_marginEnd="8dp"
18.         android:backgroundTint="@color/accent"
19.         android:onClick="selectImage"
20.         android:text="@string/selectButton"
21.         app:layout_constraintBottom_toTopOf="@+id/cameraButton"
22.         app:layout_constraintEnd_toEndOf="parent"
23.         app:layout_constraintHorizontal_bias="0.5"
24.         app:layout_constraintStart_toStartOf="parent"
25.         app:layout_constraintTop_toBottomOf="@+id/titleMainTextView" />
26.
27.     <Button
```

```
28.         android:id="@+id/cameraButton"
29.         android:layout_width="wrap_content"
30.         android:layout_height="wrap_content"
31.         android:layout_marginStart="8dp"
32.         android:layout_marginEnd="8dp"
33.         android:layout_marginBottom="32dp"
34.         android:backgroundTint="@color/accent"
35.         android:onClick="openCamera"
36.         android:text="@string/cameraButton"
37.         app:layout_constraintBottom_toTopOf="@+id/modelRadioGroup"
38.         app:layout_constraintEnd_toEndOf="parent"
39.         app:layout_constraintHorizontal_bias="0.5"
40.         app:layout_constraintStart_toStartOf="parent"
41.         app:layout_constraintTop_toBottomOf="@+id/selectButton" />
42.
43.     <TextView
44.         android:id="@+id/titleMainTextView"
45.         android:layout_width="wrap_content"
46.         android:layout_height="wrap_content"
47.         android:text="@string/title"
48.         android:textAlignment="center"
49.         android:textSize="20sp"
50.         android:textStyle="bold"
51.         app:layout_constraintBottom_toTopOf="@+id/selectButton"
52.         app:layout_constraintEnd_toEndOf="parent"
53.         app:layout_constraintHorizontal_bias="0.5"
54.         app:layout_constraintStart_toStartOf="parent"
55.         app:layout_constraintTop_toTopOf="parent" />
56.
57.     <RadioGroup
58.         android:id="@+id/modelRadioGroup"
59.         android:layout_width="0dp"
60.         android:layout_height="wrap_content"
61.         android:layout_margin="10dp"
62.         android:layout_marginBottom="32dp"
63.         android:checkedButton="@id/commonObjectsRadioButton"
64.         android:padding="5dp"
65.         app:layout_constraintBottom_toBottomOf="parent"
66.         app:layout_constraintEnd_toEndOf="parent"
67.         app:layout_constraintStart_toStartOf="parent">
68.
69.         <RadioButton
70.             android:id="@+id/commonObjectsRadioButton"
71.             android:layout_width="match_parent"
72.             android:layout_height="wrap_content"
73.             android:checked="true"
74.             android:onClick="radioButtonChecked"
75.             android:tag="common objects"
76.             android:text="@string/radioButton1" />
77.
78.         <RadioButton
79.             android:id="@+id/electronicsRadioButton"
80.             android:layout_width="match_parent"
81.             android:layout_height="wrap_content"
82.             android:onClick="radioButtonChecked"
83.             android:tag="electronics"
84.             android:text="@string/radioButton2" />
85.
86.         <RadioButton
87.             android:id="@+id/flowerRadioButton"
88.             android:layout_width="match_parent"
89.             android:layout_height="wrap_content"
90.             android:onClick="radioButtonChecked"
```

```
            android:tag="flower"
            android:text="@string/radioButton3" />
    </RadioGroup>

    <TextView
        android:id="@+id/radioButtonTextName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
          android:layout_marginBottom="16dp"
          android:text="@string/radioButtonText"
          android:textSize="20sp"
          app:layout_constraintBottom_toTopOf="@+id/modelRadioGroup"
          app:layout_constraintEnd_toEndOf="parent"
          app:layout_constraintStart_toStartOf="parent"
          app:layout_constraintTop_toBottomOf="@+id/cameraButton" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Αρχείο content_main.xml (landscape)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/primary_light"
    android:backgroundTint="@color/primary_light"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <Button
        android:id="@+id/selectButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/accent"
        android:onClick="selectImage"
        android:text="@string/selectButton"
        app:layout_constraintBottom_toTopOf="@+id/radioButtonTextName"
        app:layout_constraintEnd_toStartOf="@+id/cameraButton"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/titleMainTextView" />

    <Button
        android:id="@+id/cameraButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/accent"
        android:onClick="openCamera"
        android:text="@string/cameraButton"
        app:layout_constraintBottom_toTopOf="@+id/radioButtonTextName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/selectButton"
        app:layout_constraintTop_toBottomOf="@+id/titleMainTextView" />

    <TextView
        android:id="@+id/titleMainTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/title"
```

```
40.         android:textSize="20sp"
41.         android:textStyle="bold"
42.         app:layout_constraintBottom_toTopOf="@+id/selectButton"
43.         app:layout_constraintEnd_toEndOf="parent"
44.         app:layout_constraintHorizontal_bias="0.5"
45.         app:layout_constraintStart_toStartOf="parent"
46.         app:layout_constraintTop_toTopOf="parent" />
47.
48.     <TextView
49.         android:id="@+id/radioButtonTextName"
50.         android:layout_width="wrap_content"
51.         android:layout_height="wrap_content"
52.         android:layout_marginBottom="32dp"
53.         android:text="@string/radioButtonText"
54.         android:textSize="20sp"
55.         app:layout_constraintBottom_toTopOf="@+id/modelRadioGroup"
56.         app:layout_constraintEnd_toEndOf="parent"
57.         app:layout_constraintHorizontal_bias="0.5"
58.         app:layout_constraintStart_toStartOf="parent" />
59.
60.     <RadioGroup
61.         android:id="@+id/modelRadioGroup"
62.         android:layout_width="wrap_content"
63.         android:layout_height="wrap_content"
64.         android:layout_margin="10dp"
65.         android:orientation="horizontal"
66.         app:layout_constraintBottom_toBottomOf="parent"
67.         app:layout_constraintEnd_toEndOf="parent"
68.         app:layout_constraintHorizontal_bias="0.5"
69.         app:layout_constraintStart_toStartOf="parent">
70.
71.         <RadioButton
72.             android:id="@+id/commonObjectsRadioButton"
73.             android:layout_width="match_parent"
74.             android:layout_height="wrap_content"
75.             android:layout_margin="10dp"
76.             android:checked="true"
77.             android:onClick="radioButtonChecked"
78.             android:tag="common objects"
79.             android:text="@string/radioButton1" />
80.
81.         <RadioButton
82.             android:id="@+id/electronicsRadioButton"
83.             android:layout_width="match_parent"
84.             android:layout_height="wrap_content"
85.             android:layout_margin="10dp"
86.             android:onClick="radioButtonChecked"
87.             android:tag="electronics"
88.             android:text="@string/radioButton2" />
89.
90.         <RadioButton
91.             android:id="@+id/flowerRadioButton"
92.             android:layout_width="match_parent"
93.             android:layout_height="wrap_content"
94.             android:layout_margin="10dp"
95.             android:onClick="radioButtonChecked"
96.             android:tag="flower"
97.             android:text="@string/radioButton3" />
98.
99.     </RadioGroup>
100.
101. </androidx.constraintlayout.widget.ConstraintLayout>
```

Αρχείο item recognition.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp">

    <ImageView
        android:id="@+id/itemColor"
        android:layout_width="25dp"
        android:layout_height="25dp"
        android:layout_weight="0"
        app:srcCompat="@color/black" />

    <TextView
        android:id="@+id/itemName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_weight="3"
        android:text="@string/itemName"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/itemProbability"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/itemProbability"
        android:textAlignment="viewEnd"
        android:textSize="18sp" />
</LinearLayout>
```

Αρχείο menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.xarhssta.objectrecognition.MainActivity">
    <item
        android:id="@+id/action_about"
        android:orderInCategory="100"
        android:title="@string/action_about"
        app:showAsAction="ifRoom" />
</menu>
```

Αρχείο BaseActivity.kt

```
package com.xarhssta.objectrecognition

import android.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

open class BaseActivity : AppCompatActivity() {

```

```
8.      internal fun activateToolbar(enableHome: Boolean) {
9.          val toolbar = findViewById<androidx.appcompat.widget.Toolbar>(R.id.toolbar)
10.         setSupportActionBar(toolbar)
11.         supportActionBar?.setDisplayHomeAsUpEnabled(enableHome)
12.     }
13.
14.     internal fun showAboutDialog() {
15.         val messageView = layoutInflater.inflate(R.layout.activity_about, null, false)
16.         val builder = AlertDialog.Builder(this)
17.         val aboutDialog: AlertDialog = builder.setView(messageView).create()
18.         aboutDialog.setCanceledOnTouchOutside(true)
19.         aboutDialog.show()
20.     }
21. }
```

Αρχείο MainActivity.kt

```
1.  package com.xarhssta.objectrecognition
2.
3.  import android.app.Activity
4.  import android.content.ContentValues
5.  import android.content.Intent
6.  import android.content.SharedPreferences
7.  import android.content.pm.PackageManager
8.  import android.graphics.Bitmap
9.  import android.graphics.ImageDecoder
10. import android.net.Uri
11. import android.os.Build
12. import android.os.Bundle
13. import android.provider.MediaStore
14. import android.util.Log
15. import android.view.Menu
16. import android.view.MenuItem
17. import android.view.View
18. import android.widget.RadioButton
19. import java.io.ByteArrayOutputStream
20. import java.io.File
21.
22. private const val TAG = "MainActivity"
23.
24. class MainActivity : BaseActivity() {
25.
26.     private val requestImageFromStorage = 1
27.     private val requestPictureCode = 2
28.     private var imageUri: Uri? = Uri.EMPTY
29.     private lateinit var mySharedPreferences: SharedPreferences
30.     private lateinit var mySharedPreferencesEditor: SharedPreferences.Editor
31.     private var radioButtonSelected: Int = 0
32.
33.     override fun onCreate(savedInstanceState: Bundle?) {
34.         super.onCreate(savedInstanceState)
35.         setContentView(R.layout.activity_main)
36.         setSupportActionBar(findViewById(R.id.toolbar))
37.         mySharedPreferences = getSharedPreferences("radioChecked", MODE_PRIVATE)
38.         mySharedPreferencesEditor = mySharedPreferences.edit()
39.         radioButtonSelected = mySharedPreferences.getInt("number", 1)
40.         setRadioButton(radioButtonSelected)
41.     }
42.
43.     override fun onCreateOptionsMenu(menu: Menu): Boolean {
44.         menuInflater.inflate(R.menu.menu_main, menu)
```

```
45.         return true
46.     }
47.
48.     override fun onOptionsItemSelected(item: MenuItem): Boolean {
49.         return when (item.itemId) {
50.             R.id.action_about -> {
51.                 showAboutDialog()
52.                 true
53.             }
54.             else -> super.onOptionsItemSelected(item)
55.         }
56.     }
57.
58.     fun radioButtonChecked(view: View) {
59.         radioButtonSelected = when (view.tag) {
60.             "common objects" -> 1
61.             "electronics" -> 2
62.             "flower" -> 3
63.             else -> 0
64.         }
65.         mySharedPreferencesEditor.putInt("number", radioButtonSelected)
66.             .apply()
67.     }
68.
69.     private fun setRadioButton(radioButtonChecked: Int) {
70.         val firstRadioButton = findViewById<RadioButton>(R.id.commonObjectsRadioButton)
71.         val secondRadioButton = findViewById<RadioButton>(R.id.electronicsRadioButton)
72.         val thirdRadioButton = findViewById<RadioButton>(R.id.flowerRadioButton)
73.         when (radioButtonChecked) {
74.             1 -> firstRadioButton.isChecked = true
75.             2 -> secondRadioButton.isChecked = true
76.             3 -> thirdRadioButton.isChecked = true
77.         }
78.     }
79.
80.     fun selectImage(view: View) {
81.         if (checkSelfPermission(android.Manifest.permission.READ_EXTERNAL_STORAGE) !=
    PackageManager.PERMISSION_GRANTED) {
82.             requestPermissions(
83.                 arrayOf(android.Manifest.permission.READ_EXTERNAL_STORAGE),
84.                 requestImageFromStorage
85.             )
86.         } else {
87.             getPhoto()
88.         }
89.     }
90.
91.     private fun getPhoto() {
92.         val intent = Intent(Intent.ACTION_PICK,
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
93.         startActivityForResult(intent, requestImageFromStorage)
94.     }
95.
96.     fun openCamera(view: View) {
97.         if (checkSelfPermission(android.Manifest.permission.CAMERA) !=
    PackageManager.PERMISSION_GRANTED) {
98.             requestPermissions(arrayOf(android.Manifest.permission.CAMERA),
    requestPictureCode)
99.         } else if (checkSelfPermission(android.Manifest.permission.WRITE_EXTERNAL_STORAGE)
    != PackageManager.PERMISSION_GRANTED) {
100.
    requestPermissions(arrayOf(android.Manifest.permission.WRITE_EXTERNAL_STORAGE),
    requestPictureCode)
```

```
101.          } else {
102.              dispatchTakePictureIntent()
103.          }
104.      }
105.
106.      private fun dispatchTakePictureIntent() {
107.          val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
108.          if (takePictureIntent.resolveActivity(packageManager) != null) {
109.              val values = ContentValues()
110.              values.put(MediaStore.Images.Media.TITLE, "myPicture")
111.              values.put(
112.                  MediaStore.Images.Media.DESCRIPTION,
113.                  "Taken On " + System.currentTimeMillis()
114.              )
115.              imageUri =
    contentResolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values)
116.              takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri)
117.              startActivityForResult(takePictureIntent, requestPictureCode)
118.          }
119.      }
120.
121.      private fun checkForBlankFile() {
122.          val fileDelete: File = File(getPath(imageUri))
123.          if (fileDelete.length().toInt() == 0) {
124.              contentResolver.delete(imageUri!!, null, null)
125.          }
126.      }
127.
128.      private fun getPath(uri: Uri?): String {
129.          val projection = arrayOf(MediaStore.Images.Media.DATA)
130.          val cursor = contentResolver.query(uri!!, projection, null, null, null)
131.          val columnIndex = cursor?.getColumnIndexOrThrow(MediaStore.Images.Media.DATA)
132.          cursor?.moveToFirst()
133.          val s = cursor?.getString(columnIndex!!)
134.          cursor?.close()
135.          return s!!
136.      }
137.
138.      override fun onActivityResult(requestCode: Int, resultCode: Int, dataIntent: Intent?)
    {
139.          super.onActivityResult(requestCode, resultCode, dataIntent)
140.          var image: Bitmap? = null
141.          if (requestCode == requestPictureCode) {
142.              checkForBlankFile()
143.          }
144.          if (dataIntent != null && requestCode == requestImageFromStorage && resultCode ==
    Activity.RESULT_OK) {
145.              val selectedImage = dataIntent.data
146.              try {
147.                  image = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
148.                      val source = ImageDecoder.createSource(this.contentResolver,
    selectedImage!!)
149.                      ImageDecoder.decodeBitmap(source)
150.                  } else {
151.                      MediaStore.Images.Media.getBitmap(this.contentResolver, selectedImage)
152.                  }
153.              } catch (e: Exception) {
154.                  Log.e(TAG, e.message!!)
155.              }
156.          } else if (requestCode == requestPictureCode && resultCode == Activity.RESULT_OK)
    {
157.              image = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
158.                  val source = ImageDecoder.createSource(this.contentResolver, imageUri!!)
```

```
159.                 ImageDecoder.decodeBitmap(source)
160.             } else {
161.                 MediaStore.Images.Media.getBitmap(contentResolver, imageUri)
162.             }
163.         }
164.         if (image != null) {
165.             startChosenImage(image)
166.         }
167.     }
168.
169.     private fun startChosenImage(image: Bitmap) {
170.         val byteArrayOutputStream = ByteArrayOutputStream()
171.         image.compress(Bitmap.CompressFormat.JPEG, 25, byteArrayOutputStream)
172.         val bitmapData = byteArrayOutputStream.toByteArray()
173.
174.         val intent = Intent(this, ChosenImage::class.java)
175.         intent.putExtra("photo", bitmapData)
176.         intent.putExtra("model", radioButtonSelected)
177.         mySharedPreferencesEditor.remove("number").apply()
178.         startActivity(intent)
179.     }
180. }
```

Αρχείο ChosenImage.kt

```
1.  package com.xarhssta.objectrecognition
2.
3.  import android.graphics.*
4.  import android.os.Bundle
5.  import android.view.Menu
6.  import android.view.MenuItem
7.  import android.widget.ImageView
8.  import androidx.lifecycle.ViewModelProvider
9.  import androidx.recyclerview.widget.LinearLayoutManager
10. import androidx.recyclerview.widget.RecyclerView
11.
12. private const val STATE_RECOGNIZE = "Image"
13.
14. class ChosenImage : BaseActivity() {
15.
16.     private var data: ByteArray? = null
17.     private var chosenImageView: ImageView? = null
18.     private val itemRecognitions: List<ItemRecognition> = EMPTY_ITEM_LIST
19.     private val itemRecognitionAdapter = ItemRecognitionAdapter(itemRecognitions)
20.
21.     override fun onCreate(savedInstanceState: Bundle?) {
22.         super.onCreate(savedInstanceState)
23.         setContentView(R.layout.activity_chosen_image)
24.         setSupportActionBar(findViewById(R.id.toolbar))
25.         activateToolbar(true)
26.
27.         val viewModel = ViewModelProvider(this).get(ItemRecognitionViewModel::class.java)
28.         chosenImageView = findViewById(R.id.chosenImageView)
29.
30.         data = if (savedInstanceState == null) {
31.             intent.getByteArrayExtra("photo")!!
32.         } else {
33.             savedInstanceState.getByteArray(STATE_RECOGNIZE)!!
34.         }
35.
36.         val modelChosen = intent.getIntExtra("model", 0)
```

```
37.
38.         val imageBitmap = BitmapFactory.decodeByteArray(data, 0, data!!.size)
39.         val convertedBitmap = convert(imageBitmap)
40.
41.         val itemList: RecyclerView = findViewById(R.id.itemList)
42.         itemList.layoutManager = LinearLayoutManager(this)
43.         itemList.adapter = itemRecognitionAdapter
44.
45.         viewModel.recognitionList.observe(this,
46.             {
47.                 itemRecognitionAdapter.setRecognitionList(it ?: EMPTY_ITEM_LIST)
48.             })
49.
50.         viewModel.recognizedBitmap.observe(this,
51.             {
52.                 chosenImageView?.setImageBitmap(it)
53.             })
54.         if (savedInstanceState == null) {
55.             viewModel.recognizeObjects(convertedBitmap, modelChosen)
56.         }
57.     }
58.
59.     override fun onCreateOptionsMenu(menu: Menu?): Boolean {
60.         menuInflater.inflate(R.menu.menu_main, menu)
61.         return true
62.     }
63.
64.     override fun onOptionsItemSelected(item: MenuItem): Boolean {
65.         return when (item.itemId) {
66.             R.id.action_about -> {
67.                 showAboutDialog()
68.                 true
69.             }
70.             else -> super.onOptionsItemSelected(item)
71.         }
72.     }
73.
74.     private fun convert(bitmap: Bitmap): Bitmap {
75.         val convertedBitmap =
76.             Bitmap.createBitmap(bitmap.width, bitmap.height, Bitmap.Config.ARGB_8888)
77.         val canvas = Canvas(convertedBitmap)
78.         val paint = Paint()
79.         paint.color = Color.BLACK
80.         val zero = .0f
81.         canvas.drawBitmap(bitmap, zero, zero, paint)
82.         return convertedBitmap
83.     }
84.
85.     override fun onSaveInstanceState(outState: Bundle) {
86.         super.onSaveInstanceState(outState)
87.         outState.putByteArray(STATE_RECOGNIZE, data!!)
88.     }
89. }
90.
```

Αρχείο ItemRecognitionAdapter.kt

```
1.  package com.xarhssta.objectrecognition
2.
3.  import android.graphics.Bitmap
4.  import android.graphics.Canvas
5.  import android.view.LayoutInflater
6.  import android.view.View
7.  import android.view.ViewGroup
8.  import android.widget.ImageView
9.  import android.widget.TextView
10. import androidx.recyclerview.widget.RecyclerView
11. import java.util.*
12.
13. class ItemRecognitionAdapter(private var itemList: List<ItemRecognition>) :
14.     RecyclerView.Adapter<ItemRecognitionViewHolder>() {
15.     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
    ItemRecognitionViewHolder {
16.         val view =
17.             LayoutInflater.from(parent.context).inflate(R.layout.item_recognition, parent,
    false)
18.         return ItemRecognitionViewHolder(view)
19.     }
20.
21.     override fun getItemCount(): Int {
22.         return if (itemList.isNotEmpty()) itemList.size else 1
23.     }
24.
25.     override fun onBindViewHolder(holder: ItemRecognitionViewHolder, position: Int) {
26.         if (itemList.isNotEmpty()) {
27.             holder.itemView.visibility = View.VISIBLE
28.             val data = itemList.get(position)
29.             val probabilityString = String.format("%.1f", data.probability * 100.0f)
30.             "$probabilityString %".also { holder.itemProbability.text = it }
31.             holder.itemName.text = data.label.capitalize(Locale.ROOT)
32.             val bitmap: Bitmap = Bitmap.createBitmap(25, 25, Bitmap.Config.ARGB_8888)
33.             val canvas = Canvas(bitmap)
34.             canvas.drawColor(data.color)
35.             holder.itemColor.setImageBitmap(bitmap)
36.         } else {
37.             holder.itemView.visibility = View.INVISIBLE
38.         }
39.     }
40.
41.     fun setRecognitionList(recognitionList: List<ItemRecognition>) {
42.         this.itemList = recognitionList
43.         notifyDataSetChanged()
44.     }
45. }
46.
47. class ItemRecognitionViewHolder(view: View) : RecyclerView.ViewHolder(view) {
48.
49.     var itemColor: ImageView = view.findViewById(R.id.itemColor)
50.     var itemName: TextView = view.findViewById(R.id.itemName)
51.     var itemProbability: TextView = view.findViewById(R.id.itemProbability)
52.
53. }
```

Αρχείο ItemRecognitionViewModel.kt

```
1.  package com.xarhssta.objectrecognition
2.
3.  import android.graphics.*
4.  import androidx.lifecycle.LiveData
5.  import androidx.lifecycle.MutableLiveData
6.  import androidx.lifecycle.ViewModel
7.  import java.util.*
8.
9.  val EMPTY_ITEM_LIST: List<ItemRecognition> = Collections.emptyList()
10.
11. class ItemRecognitionViewModel : ViewModel(), ObjectDetection.ObjectCallback {
12.
13.     private val recognitionListMutable = MutableLiveData<List<ItemRecognition>>()
14.     private val recognizedBitmapMutable = MutableLiveData<Bitmap>()
15.
16.     val recognitionList: LiveData<List<ItemRecognition>>
17.         get() = recognitionListMutable
18.     val recognizedBitmap: LiveData<Bitmap>
19.         get() = recognizedBitmapMutable
20.
21.     init {
22.         recognitionListMutable.postValue(EMPTY_ITEM_LIST)
23.     }
24.
25.     override fun onObjectRecognized(itemList: List<ItemRecognition>, bitmap: Bitmap) {
26.         recognitionListMutable.value = itemList
27.         recognizedBitmapMutable.value = bitmap
28.     }
29.
30.     fun recognizeObjects(bitmap: Bitmap, modelChosen: Int) {
31.         val objectDetection = ObjectDetection(this)
32.         objectDetection.locateObjects(bitmap, modelChosen)
33.     }
34.
35. }
36.
37. data class ItemRecognition(
38.     val id: Int?,
39.     val label: String,
40.     val probability: Float,
41.     val location: Rect,
42.     val color: Int
43. ) {
44.
45.     private val probabilityString = String.format("%.1f", probability * 100.0f)
46.
47.     override fun toString(): String {
48.         return "$id |$label | $probabilityString |$location |$color"
49.     }
50. }
```

Αρχείο ObjectDetection.kt

```
1.  package com.xarhssta.objectrecognition
2.
3.  import android.graphics.*
4.  import android.util.Log
```

```
5.  import com.google.mlkit.common.model.LocalModel
6.  import com.google.mlkit.vision.common.InputImage
7.  import com.google.mlkit.vision.label.ImageLabel
8.  import com.google.mlkit.vision.label.ImageLabeling
9.  import com.google.mlkit.vision.label.custom.CustomImageLabelerOptions
10. import com.google.mlkit.vision.objects.ObjectDetection
11. import com.google.mlkit.vision.objects.defaults.ObjectDetectorOptions
12. import java.util.*
13.
14. private const val TAG = "ObjectDetection"
15.
16. class ObjectDetection(private val callback: ObjectCallback) {
17.
18.     private val colorTable: List<Int> =
19.         listOf(Color.GREEN, Color.RED, Color.BLUE, Color.BLACK, Color.YELLOW)
20.     private val itemRecognition: MutableList<ItemRecognition> = mutableListOf()
21.
22.     interface ObjectCallback {
23.         fun onObjectRecognized(itemList: List<ItemRecognition>, bitmap: Bitmap)
24.     }
25.
26.     fun locateObjects(bitmap: Bitmap, modelChosen: Int) {
27.
28.         // Setting the model
29.         val filePath = when (modelChosen) {
30.             1 -> "object dataset"
31.             2 -> "electronics dataset"
32.             3 -> "flower dataset"
33.             else -> ""
34.         }
35.
36.         // Finding the object
37.         val objectDetectorOptions = ObjectDetectorOptions.Builder()
38.             .setDetectorMode(ObjectDetectorOptions.SINGLE_IMAGE_MODE)
39.             .enableMultipleObjects()
40.             .enableClassification()
41.             .build()
42.
43.         val objectDetector = ObjectDetection.getClient(objectDetectorOptions)
44.         val inputImage = InputImage.fromBitmap(bitmap, 0)
45.         var recognizedBitmap: Bitmap = bitmap
46.         var croppedBitmap: Bitmap
47.
48.         objectDetector.process(inputImage)
49.             .addOnFailureListener { e ->
50.                 Log.e(TAG, "Failed to find object ${e.message}")
51.             }
52.             .addOnSuccessListener { results ->
53.                 for ((count, detectedObject) in results.withIndex()) {
54.                     val boundingBox = detectedObject.boundingBox
55.                     val trackingId = detectedObject.trackingId
56.                     croppedBitmap = cropObject(bitmap, boundingBox)
57.
58.                     val localModel = LocalModel.Builder()
59.                         .setAssetFilePath("$filePath/model.tflite")
60.                         .build()
61.
62.                     val customImageLabelerOptions =
    CustomImageLabelerOptions.Builder(localModel)
63.                         .setMaxResultCount(3)
64.                         .build()
65.
66.                     val labeler = ImageLabeling.getClient(customImageLabelerOptions)
```

```
                    labeler.process(InputImage.fromBitmap(croppedBitmap, 0))
                        .addOnFailureListener { e -> Log.e(TAG, "FAILED! ${e.message}") }
                        .addOnSuccessListener { labels ->
                            if (labels.isNotEmpty()) {
                                for (label in labels) {
                                    itemRecognition.add(
                                        ItemRecognition(
                                            trackingId,
                                            label.text,
                                            label.confidence,
                                            boundingBox,
                                            colorTable[count]
                                        )
                                    )
                                }
                                recognizedBitmap =
                                    paintAround(recognizedBitmap, boundingBox, labels[0],
    count)
                            }
                            callback.onObjectRecognized(itemRecognition, recognizedBitmap)
                        }
                }
            }
    }


    private fun cropObject(bitmap: Bitmap, rect: Rect): Bitmap {
        return try {
            val width = rect.right - rect.left
            val height = rect.bottom - rect.top
            Bitmap.createBitmap(
                bitmap,
                rect.left,
                rect.top,
                  width,
                  height
              )
          } catch (e: IllegalArgumentException) {
              bitmap
          }
      }

      private fun paintAround(
          bitmap: Bitmap,
          boundingBox: Rect,
          label: ImageLabel,
          count: Int
      ): Bitmap {
          val canvas = Canvas(bitmap)
          val paint = Paint()
          paint.style = Paint.Style.STROKE
          paint.color = colorTable[count]
          paint.strokeCap = Paint.Cap.ROUND
          paint.strokeWidth = 10.0f
          paint.strokeJoin = Paint.Join.ROUND
          paint.strokeMiter = 100f
          paint.isAntiAlias = true
          canvas.drawRect(boundingBox, paint)

          // Drawing the rectangle for the object name
          val textPaint = Paint()
          textPaint.color = Color.WHITE
          textPaint.textSize = 50f
```

```
129.
130.            val boxPaint = Paint()
131.            boxPaint.color = colorTable[count]
132.            boxPaint.style = Paint.Style.FILL
133.
134.            val textToShow = label.text.capitalize(Locale.ROOT) + " " + String.format(
135.                "%.1f",
136.                label.confidence * 100.0f
137.            ) + " %"
138.            val textWidth = textPaint.measureText(textToShow)
139.
140.            val rectLeft = boundingBox.left.toFloat()
141.            val rectTop = boundingBox.top - 50.toFloat()
142.            val rectRight = boundingBox.left + textWidth
143.            val rectBottom = boundingBox.top.toFloat()
144.
145.            canvas.drawRect(rectLeft, rectTop, rectRight, rectBottom, boxPaint)
146.            canvas.drawText(textToShow, rectLeft, rectBottom, textPaint)
147.            return bitmap
148.        }
149.
150.
151.    }
```