ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Herb Tracker:
Εφαρμογή android για την αναγνώριση βοτάνων και λουλουδιών με τη χρήση τεχνικών μηχανικής μάθησης

Διπλωματική Εργασία

Ζήσιος Κυριάκος

Επιβλέπων: Δρ. Φεύγας Αθανάσιος

Φεβρουάριος 2024

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Herb Tracker:
An android application for herb and flower recognition using Machine Learning techniques

Diploma Thesis

Zisios Kyriakos

Supervisor: Dr. Fevgas Athanasios

February 2024

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Δρ. Φεύγας Αθανάσιος** Μέλος Ε.ΔΙ.Π, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Δρ. Βασιλακόπουλος Μιχαήλ** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Δρ. Τουσίδου Ελένη** Μέλος Ε.ΔΙ.Π, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Στα πλαίσια της εκπόνησης της διπλωματικής μου εργασίας, θα ήθελα αρχικά να ευχαριστήσω τους επιβλέποντες καθηγητές μου κ. Φεύγα Αθανάσιο, κ. Βασιλακόπουλο Μιχαήλ και κ. Τουσίδου Ελένη, που μου παρείχαν την δυνατότητα να ενασχοληθώ με το θέμα της ανάπτυξης της εφαρμογής Android, Herb Tracker, με χρήση τεχνικών μηχανικής μάθησης. Θερμές ευχαριστίες οφείλω να δώσω επιπρόσθετα στον κ. Φεύγα, ο οποίος με τις πλούσιες γνώσεις του με καθοδήγησε με τον καλύτερο τρόπο, παρέχοντας μου στήριξη, πολύτιμες γνώσεις και ουσιαστική καθοδήγηση πάνω στο συγκεκριμένο θέμα, ενώ παράλληλα ήταν πρόθυμος να μου λύσει απορίες και προβληματισμούς που προέκυψαν κατά τη διάρκεια της ανάπτυξης της εφαρμογής.

Σημαντική βοήθεια, επίσης, έλαβα από φίλους και συναδέλφους οι οποίοι με παρότρυναν ψυχολογικά, ώστε να εκπληρώσω τον στόχο μου, αλλά και πρακτικά με την ανταλλαγή ιδεών και απόψεων, καθ΄ όλη της διάρκεια των φοιτητικών μου χρόνων στο Πανεπιστήμιο και θα ήθελα να τους ευχαριστήσω από τα βάθη της καρδιάς μου. Τέλος, θα ήθελα να αφιερώσω την παρούσα εργασία στην οικογένεια μου, οι οποίοι στήριξαν και στηρίζουν τις επιλογές μου με κάθε δυνατό τρόπο, τόσο ηθικά και πνευματικά, όσο και υλικά. Κυρίως, όμως, η παρούσα εργασία είναι αφιερωμένη στην μνήμη του αδερφού μου Άγγελου Ζήσιου, ο οποίος αποτέλεσε και αποτελεί έμπνευση τόσο στον επαγγελματικό, όσο και στον πνευματικό τομέα.

Περίληψη

Η Herb Tracker είναι μία εφαρμογή Android που χρησιμοποιεί τεχνικές μηχανικής μάθησης για την αναγνώριση βοτάνων, με αλγόριθμους που είναι κατάλληλοι για χρήση σε κινητά τηλέφωνα. Επίσης, παρέχει πληροφορίες για βότανα που δύναται να αναγνωρίσει. Η ιδέα της εφαρμογής αυτής, αποσκοπεί τόσο στη διευκόλυνση αναγνώρισης βοτάνων, όσο και στη στροφή προς μια πιο «πράσινη» και βιώσιμη ζωή του ευρύτερου κοινωνικού συνόλου. Μάλιστα, σαν έμπνευση και κίνητρο υπήρξε η ιδέα ότι οι νεότερες γενιές θα μπορούν, με έναν πιο ενδιαφέρον τρόπο, να κινηθούν προς αυτή τη βιώσιμη κατεύθυνση. Οι χρήστες Android συσκευών, με την εγκατάσταση της εφαρμογής, έχουν τη δυνατότητα αναγνώρισης βοτάνων μέσω 2 λειτουργιών: τη λήψη φωτογραφίας από τη κάμερα της συσκευής ή τη χρήση εικόνας από τη συλλογή της συσκευής. Στην περίπτωση που γίνεται χρήση της κάμερας, παρέχεται μια επιπλέον δυνατότητα στους χρήστες, αυτή της αποθήκευσης της τοποθεσίας. Με τη χρήση χαρτών, οι χρήστες μπορούν να «σημαδέψουν» το γεωγραφικό σημείο που λήφθηκε η φωτογραφία, ώστε να μπορέσουν να το επισκεφθούν ξανά σε μελλοντικό χρόνο.

Λέξεις κλειδιά: Κατηγοριοποίηση εικόνας, Νευρωνικά Δίκτυα, Εφαρμογή Android, Jetpack Compose

Abstract

Herb Tracker is an android application which can recognize various herbs and present information about each recognized herb, it utilizes machine learning techniques and specific algorithms suitable for use on mobile phones. The idea behind Herb Tracker application is a combination of facilitating the public in identifying herbs, but also the shift towards a greener and more sustainable life. Informing all android phone users, but mainly the young generation, in this sustainable direction, was an inspirational point of this application. By installing the application, users have the option, either by taking a photo through their own camera or by using an already saved photo in the images of their device, to identify the herb and read some information about it. Also, in case the user takes a photo with the camera, he will be able to save the location as a marker on a map, for a future visit or further exploration.

Keywords: Image Classification, Deep Neural Networks, Android application, Jetpack Compose

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Πινάκων

Κεφάλαιο 1 - Εισαγωγή

Σε μια εποχή ραγδαίας ψηφιακής εξέλιξης, η τεχνολογία εξελίσσεται σε καθημερινή βάση με εξίσου γοργούς ρυθμούς, προσφέροντας λύσεις που αναδιαμορφώνουν τον τρόπο ζωής μας. Σε αυτό το πλαίσιο, στην παρούσα διατριβή παρουσιάζεται η εφαρμογή Herb Tracker. Ένα εργαλείο, που συνδυάζει τις δυνατότητες της τεχνητής νοημοσύνης με την τεχνολογία και λειτουργικότητα των συσκευών Android, για την αναγνώριση φωτογραφίας βοτάνων και φυτών.

Η Herb Tracker αποτελεί μια εφαρμογή για τους λάτρεις της φύσης, της κηπουρικής, αλλά και όσων απλά επιθυμούν να γνωρίσουν τον κόσμο των ευεργετικών φυτών. Με τη χρήση τεχνικών μηχανικής μάθησης, η εφαρμογή δίνει τη δυνατότητα στο χρήστη να αναγνωρίζει βότανα και φυτά με μερικά κλικ στην οθόνη του κινητού του. Βασισμένη σε αλγόριθμους αναγνώρισης εικόνας, η Herb Tracker μπορεί να αναγνωρίσει 126 είδη φυτών, παρέχοντας τις απαραίτητες πληροφορίες για κάθε είδος.

Μία από τις διαδραστικές κι ενδιαφέρουσες λειτουργίες της εφαρμογής είναι η δυνατότητα λήψης φωτογραφιών μέσω της κάμερας της συσκευής. Η λειτουργία αυτή επιτρέπει στον χρήστη να αναγνωρίσει και να λάβει πληροφορίες για φυτά που τον περιβάλλουν, ακόμη και χωρίς σύνδεση στο διαδίκτυο. Επιπλέον, η αναγνώριση μπορεί να γίνει μέσω επιλογής εικόνας από τη συλλογή φωτογραφιών, που διαθέτει ο χρήστης στη συσκευή του. Η εφαρμογή προσφέρει, επίσης, τη δυνατότητα αποθήκευσης της τοποθεσίας λήψης μιας φωτογραφίας ως σημείο στο χάρτη, παρέχοντας μια ολοκληρωμένη εμπειρία ανακάλυψης της φύσης.

Οι λόγοι που καθιστούν την Herb Tracker ενδιαφέρουσα και διαφορετική είναι αρκετοί. Εν πρώτης, η διερεύνηση του πεδίου των μηχανικών μάθησης στην αναγνώριση εικόνας. Ιδιαιτέρως, η μελέτη αλγορίθμων που είναι κατάλληλοι προς χρήση σε Android συσκευές, καθώς υπάρχουν παράμετροι που δεν είναι αντιληπτοί στα αρχικά στάδια της ανάπτυξης. Ακόμη, κατά τη διάρκεια χρήσης της, ανοίγεται ένα παράθυρο γνώσης και εξερεύνησης για τον κόσμο των βοτάνων και των ευεργετικών φυτών. Γνώση η οποία ωθεί το χρήστη σε μια πιο «πράσινη» ζωή, με βιώσιμες επιλογές προς το περιβάλλον αλλά και προς τον ίδιο. Γενικότερα, η Herb Tracker, όχι μόνο δίνει την επιλογή της άμεσης αναγνώρισης βοτάνων σε ένα περίπατο στο βουνό ή σε κάποιο εξωτερικό περιβάλλον, αλλά επίσης την αξιόπιστη πληροφόρηση και αποθήκευση της τοποθεσίας τους. Δυνατότητες οι οποίες βοηθούν οποιονδήποτε ενδιαφερόμενο ή ακόμη και βοτανολόγους, προς αυτή τη βιώσιμη κατεύθυνση.

Στα επόμενα κεφάλαια αποτυπώνεται η διαδικασία που έχει ακολουθηθεί ώστε να επιτευχθεί η δημιουργία της Herb Tracker. Πιο συγκεκριμένα, το κεφάλαιο 2 περιγράφει τις τεχνολογίες που χρησιμοποιήθηκαν. Στο κεφάλαιο 3, αναλύονται οι τεχνικές μηχανικής μάθησης για την αναγνώριση εικόνων, ποιες, αλλά και με ποιο τρόπο χρησιμοποιήθηκαν καθώς και το σύνολο δεδομένων που χρησιμοποιήθηκε. Συνεχίζει το κεφάλαιο 4, με τις τεχνολογίες του Android app development, αναλύοντας την επιλογή της σύγχρονης τεχνολογίας, την αρχιτεκτονική και τις

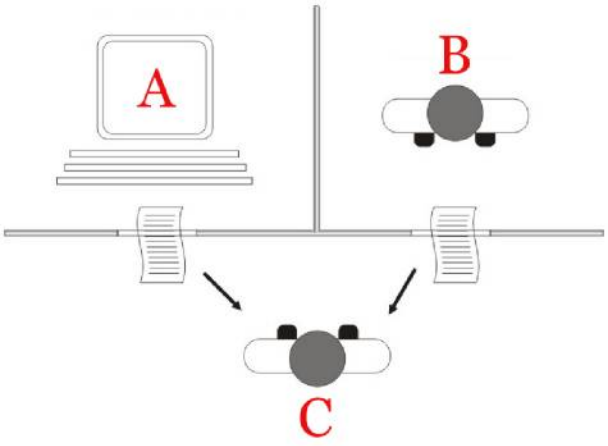
πρακτικές που η ίδια η Google προτείνει. Τέλος, το κεφάλαιο 5 πρόκειται για το μέλλον της εφαρμογής, πώς μπορεί να εξελιχθεί και να βελτιωθεί, αλλά και τί παρατηρήθηκε κατά τη διάρκεια εξέλιξης της εργασίας.

Κεφάλαιο 2 - Ιστορία και εξέλιξη Μηχανικής Μάθησης και Android development

2.1 - Μηχανική μάθηση

Η ιστορία της μηχανικής μάθησης, όπως περιγράφεται αναλυτικά στο [6], αντικατοπτρίζει την αναζήτηση της ανθρωπότητας να δημιουργήσει υπολογιστές σύμφωνα με τη ίδια της την εικόνα. Να εμποτίσει τις μηχανές με την ικανότητα να μαθαίνουν, να προσαρμόζονται και να λαμβάνουν έξυπνες αποφάσεις. Πρόκειται για ένα ταξίδι που αναδιαμόρφωσε τις βιομηχανίες, επαναπροσδιόρισε την αλληλεπίδραση ανθρώπου-υπολογιστή και εγκαινίασε μια εποχή πρωτοφανών δυνατοτήτων. Η τεχνητή νοημοσύνη και η μηχανική μάθηση μπορεί να θεωρούνται πολύ πρόσφατες ανακαλύψεις, αλλά βρίσκονται σε εξέλιξη εδώ και πολλά χρόνια.

Πρώτο μεγάλο βήμα προς τη κατεύθυνση αυτή, θεωρείται το τεστ Turing. Ο μαθηματικός και πρωτοπόρος επιστήμονας υπολογιστών, Alan Turing, πρότεινε το τεστ Turing το 1950 στην εργασία του "Computing Machinery and Intelligence". Είναι μια δοκιμή της ικανότητας μιας μηχανής να επιδεικνύει ανθρώπινη νοημοσύνη και συμπεριφορά. Το τεστ έχει σχεδιαστεί για να αξιολογήσει, εάν μια μηχανή μπορεί να μιμηθεί την ανθρώπινη συνομιλία αρκετά καλά, ώστε κάποιος παρατηρητής/αξιολογητής να μην μπορεί να διακρίνει τη διαφορά μεταξύ της μηχανής και ενός ανθρώπου, βασιζόμενος αποκλειστικά στις απαντήσεις που λαμβάνει. Το test Turing που βρίσκεται στο [6], απεικονίζεται παρακάτω:



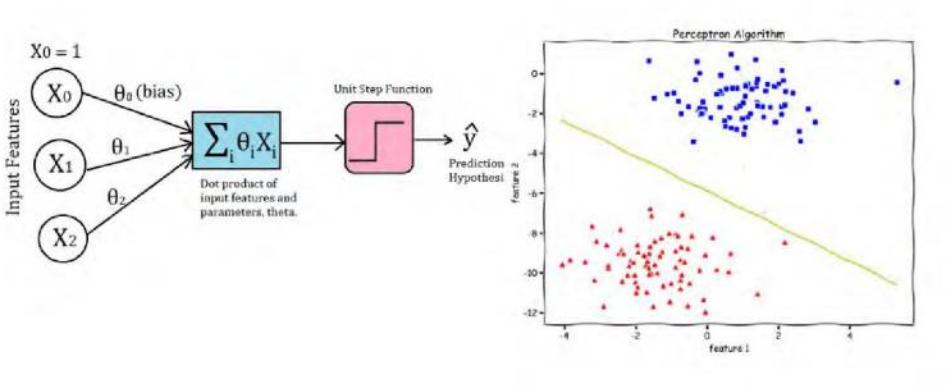
Εικόνα 1 : Turing Test, ο παρατηρητής (C) δεν θα ήταν δυνατό να αντιληφθεί τη διαφορά μεταξύ των δύο εξόδων από τη μηχανή (A) ή τον άνθρωπο (B), για μια μηχανή που το ολοκλήρωσε σωστά [6]

Ως επόμενο βήμα, θεωρείται το Checkers. Ο Arthur Samuel, πρωτοπόρος στην τεχνητή νοημοσύνη και την επιστήμη των υπολογιστών, εργάστηκε στην IBM και θεωρείται ο δημιουργός ενός από τα πρώτα παραδείγματα μιας μηχανής με ικανότητες αυτοδιδασκαλίας. Το πρόγραμμα

του Samuel, που αναπτύχθηκε μεταξύ 1952 και 1955, υλοποιήθηκε σε έναν υπολογιστή IBM 701 και χρησιμοποίησε μια μορφή μηχανικής μάθησης γνωστή ως «self-play reinforcement learning». Η μηχανή έπαιξε αμέτρητα παιχνίδια Checkers ενάντια στον εαυτό της, βελτιώνοντας σταδιακά τις στρατηγικές της μέσω δοκιμών και σφαλμάτων. Με την πάροδο του χρόνου, το πρόγραμμα βελτίωσε το παιχνίδι του, αποδεικνύοντας την ικανότητα του με το να ανταγωνίζεται ανθρώπινους παίκτες και ορισμένες φορές με επιτυχία να τους κερδίζει. Στην παρακάτω εικόνα που βρίσκεται στο [6], φαίνεται ο Arthur Samuel και η σκεπτόμενη μηχανή του:

Εικόνα 2 : Ο Arthur Samuel και η σκεπτόμενη μηχανή του [6]

Στη συνέχεια, εμφανίστηκε το Perceptron που ήταν ένα από τα πρώτα μοντέλα νευρωνικών δικτύων και μια σημαντική εξέλιξη στην ιστορία της τεχνητής νοημοσύνης και της μηχανικής μάθησης. Ο Frank Rosenblatt, στα τέλη της δεκαετίας του 1950, σε μια προσπάθεια να δημιουργήσει ένα υπολογιστικό μοντέλο που θα μπορούσε να μιμηθεί τη βασική λειτουργία ενός βιολογικού νευρώνα, παρουσίασε το perceptron. Ο προορισμός του ήταν να γίνει μια μηχανή, όμως τελικά έγινε ένας αλγόριθμος δυαδικών ταξινομητών, για την εκμάθηση με εποπτεία. Η βασική καινοτομία του perceptron ήταν η ικανότητά του να προσαρμόζει τα βάρη του, βάσει των δεδομένων εισόδου και την επιθυμητή έξοδο. Αυτή η διαδικασία προσαρμογής των βαρών, συχνά αναφέρεται ως *κανόνας μάθησης perceptron*. Ο αλγόριθμος αυτός, έθεσε τα θεμέλια για την ανάπτυξη των νευρωνικών δικτύων, τα οποία αποτελούν πλέον θεμελιώδη έννοια στη μηχανική μάθηση. Ανέδειξε, επίσης, την έννοια των μηχανών να μαθαίνουν από δεδομένα και να προσαρμόζουν τις παραμέτρους τους, με επαναλαμβανόμενο τρόπο, για την βελτίωση απόδοσής τους. Μια οπτική αναπαράσταση του perceptron που υπάρχει στο [6], απεικονίζεται παρακάτω:



Εικόνα 3 : Ο αλγόριθμος perceptron είναι ένας δυαδικός ταξινομητής που λειτουργεί με γραμμικώς διαχωρισμένα όρια αποφάσεων [6]

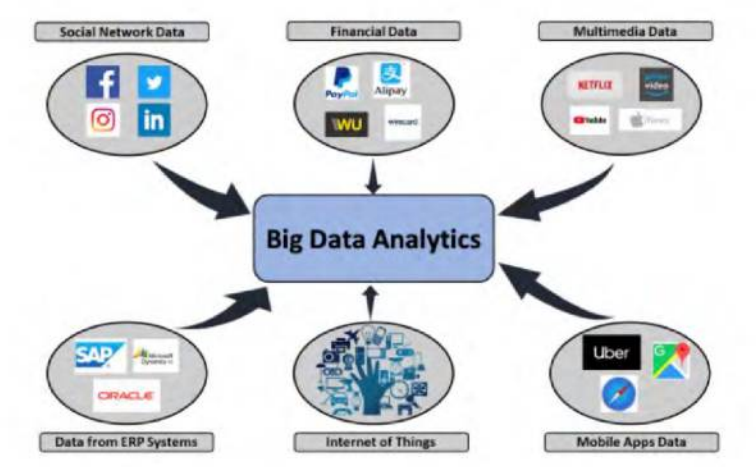
Στα τέλη της δεκαετίας του 1960 και στις αρχές της δεκαετίας του 1970, οι ερευνητές έστρεψαν την προσοχή τους σε εξειδικευμένες εφαρμογές, δημιουργώντας εξειδικευμένα συστήματα. Αυτά τα συστήματα στόχευαν στην αναπαραγωγή των ικανοτήτων λήψης αποφάσεων, χρησιμοποιώντας συμβολική λογική και κανόνες για την αναπαράσταση της γνώσης. Η έννοια των Decision trees εμφανίστηκε ως μια πολλά υποσχόμενη εναλλακτική λύση στις αρχές της δεκαετίας του 1970. Τα Decision trees προσέφεραν μια αυτοματοποιημένη προσέγγιση στη λήψη αποφάσεων που βασίζεται στα δεδομένα. Αναδεικνύοντας τις δυνατότητες της τεχνητής νοημοσύνης και της μηχανικής μάθησης στην προσομοίωση της ανθρώπινης ιδιότητας στη λήψη αποφάσεων. Ωστόσο, παρέμειναν στάσιμα λόγω χαμηλής ερμηνευτικότητας και χρησιμότητάς στο χειρισμό διαφορετικών τύπων δεδομένων.

Ως εναλλακτική λύση προέκυψε η ενισχυτική μάθηση (reinforcement learning). Αντικατέστησε τα παραδοσιακά συστήματα που βασίζονται σε συγκεκριμένους κανόνες, δίνοντας έμφαση στην ικανότητα ενός πράκτορα να αλληλοεπιδρά με το περιβάλλον του και να μαθαίνει μέσω της διαδικασίας trial and fail. Εμπνευσμένο από την ψυχολογία και τη τεχνική εκμάθησης μέσω του αποτελέσματος πράξεων (operant conditioning), οι πράκτορες προσαρμόζουν τη συμπεριφορά τους με βάση την ανατροφοδότηση, η οποία έχει τη μορφή ανταμοιβών ή ποινών. Έπειτα, εισήχθη η έννοια των μοντέλων Markov και κυρίως των διαδικασιών λήψης αποφάσεων Markov (Markov decision processes - MDPs). Τα MDPs παρέχουν ένα χρήσιμο μαθηματικό πλαίσιο για τη μοντελοποίηση της διαδοχικής λήψης αποφάσεων, όπου η μελλοντική κατάσταση ενός συστήματος βασίζεται αποκλειστικά στην παρούσα κατάστασή του, ανεξαρτήτως των προηγούμενων καταστάσεων του. Ακόμη, διευκολύνουν τον τρόπο με τον οποίο οι πράκτορες αλληλοεπιδρούν με το περιβάλλον, αναλαμβάνουν δράση και λαμβάνουν ανταμοιβές. Έτσι, σχηματίστηκαν τα θεμέλια για την επίλυση ακόμη δυσκολότερων προβλημάτων ενισχυτικής μάθησης. Με αυτό το τρόπο άνοιξε ο δρόμος για την ανάπτυξη αλγορίθμων όπως οι: dynamic programming, Q-learning και policy iteration.

Κατά τα τέλη του 20ου αιώνα, η αύξηση της υπολογιστικής ισχύος, που ξεκίνησε να είναι διαθέσιμη στους προγραμματιστές, επηρέασε σημαντικά την ανάπτυξη της μηχανικής μάθησης.

Αυτή η εποχή ήταν ένα αξιοσημείωτο σημείο ανοδικής πορείας. Κατά τη διάρκεια αυτής της περιόδου, οι ερευνητές και οι επαγγελματίες άρχισαν να πειραματίζονται με μεγαλύτερα σύνολα δεδομένων και πιο σύνθετα μοντέλα. Οι τεχνικές παράλληλης επεξεργασίας, όπως η χρήση επεξεργαστών πολλαπλών πυρήνων, άνοιξαν το δρόμο για ταχύτερους υπολογισμούς. Ωστόσο, στις αρχές του 21ου αιώνα εμφανίστηκε η μεγαλύτερη εξέλιξη με τις μονάδες επεξεργασίας γραφικών (GPUs). Αρχικά σχεδιάστηκαν για την απόδοση γραφικών, αργότερα όμως προσαρμόστηκαν για παράλληλους υπολογιστές για εργασίες μηχανικής μάθησης. Η ανάπτυξη μοντέλων βαθιάς μάθησης απέκτησε δυναμική γύρω στη δεκαετία του 2010. Με τη διαθεσιμότητα των GPUs και των κατανεμημένων υπολογιστικών frameworks, οι ερευνητές μπορούσαν να εκπαιδεύσουν βαθιά νευρωνικά δίκτυα σε αρκετά εύλογο χρονικό διάστημα.

Η ανάπτυξη του διαδικτύου και της τεχνολογίας έφερε την εποχή των Μεγάλων Δεδομένων (Big Data) - δεδομένα που είναι πολύ μεγάλα και περίπλοκα για να χειριστούν οι παραδοσιακές μέθοδοι. Αυτό οδήγησε στην εξόρυξη δεδομένων. Οι τεχνικές εξόρυξης δεδομένων, όπως η ομαδοποίηση (clustering) και η εξόρυξη κανόνων συσχέτισης (association rule mining), αναπτύχθηκαν για να αποκαλύψουν κρυμμένες γνώσεις μέσα σε αυτόν τον όγκο δεδομένων. Παράλληλα η υπολογιστική ισχύς αύξανε, οι αλγόριθμοι μηχανικής μάθησης εξελίχθηκαν και ενίσχυσαν την ικανότητα πρόγνωσής τους, καθώς έμαθαν να επεξεργάζονται και να αναλύουν αποτελεσματικά τα μεγάλα δεδομένα. Στην παρακάτω εικόνα, που βρίσκεται στο [6], εμφανίζονται οι πηγές των Μεγάλων Δεδομένων:



Εικόνα 4 : Πηγές δεδομένων που βοήθησαν στην ενίσχυση των αλγορίθμων Μηχανικής Μάθησης [6]

Προχωρώντας στα νευρωνικά δίκτυα και την επανάσταση της βαθιάς μάθησης, υπάρχει μια πρώιμη έρευνα νευρωνικών δικτύων, που εκτείνεται από τη δεκαετία του 1940 έως τη δεκαετία του 1960. Έρευνα η οποία χαρακτηρίστηκε από τις πρωτοποριακές προσπάθειες μίμησης της δομής και των λειτουργιών του ανθρώπινου εγκεφάλου μέσω υπολογιστικών μοντέλων. Τα πρώτα νευρωνικά δίκτυα περιορίζονταν κυρίως σε δομές ενός επιπέδου, γεγονός που περιόριζε τις δυνατότητές τους. Ο περιορισμός σε γραμμικώς διαχωρίσιμα προβλήματα, όπως αυτό που αντιμετώπιζε το perceptron, εμπόδισε τη δυνατότητά τους να αντιμετωπίσουν τις

πολυπλοκότητες του πραγματικού κόσμου που περιλαμβάνουν μη γραμμικές σχέσεις. Η έλλειψη θεωρητικής βάσης κατά τη διάρκεια της αρχικής έρευνας των νευρωνικών δικτύων, εμπόδισε τις εξελίξεις και συνέβαλε στην μείωση του ενδιαφέροντος για τα νευρωνικά δίκτυα στα τέλη της δεκαετίας του 1960 και στις αρχές της δεκαετίας του 1970.

Ο αλγόριθμος Backpropagation προέκυψε ως λύση στα προβλήματα που κατέκλυζαν την αρχική έρευνα νευρωνικών δικτύων. Ο αλγόριθμος αυτός αναδείχθηκε τη δεκαετία του 1980, αλλά είχε τις ρίζες του στη δεκαετία του '70. Ο Backpropagation παρείχε έναν συστηματικό μηχανισμό για τον υπολογισμό των gradients με αντίστροφη διαδικασία, καθώς ενημέρωνε με αποτελεσματικότητα τα weights από το επίπεδο εξόδου προς το επίπεδο εισόδου. Με αυτόν τον τρόπο, ρύθμιζε το πρόβλημα της εξάλειψης των gradients, επιτρέποντάς τα να κινούνται πιο αποτελεσματικά μέσα από τα επίπεδα του δικτύου, και το δίκτυο να συνεχίσει να ενημερώνεται καθώς εκπαιδεύεται. Επιπλέον, έφερε τη δυνατότητα εκπαίδευσης βαθύτερων δικτύων, ένα κατόρθωμα που δεν ήταν εφικτό στις πρώτες προσπάθειες νευρωνικών δικτύων. Πέρα από την αντιμετώπιση συγκεκριμένων δυσκολιών, ο αλγόριθμος Backpropagation επανάφερε το ενδιαφέρον στα νευρωνικά δίκτυα. Οι ερευνητές συνειδητοποίησαν ότι τα μοντέλα, ενισχυμένα από τις δυνατότητες του Backpropagation, θα μπορούσαν να συλλάβουν περίπλοκες σχέσεις και μοτίβα μέσα στα δεδομένα. Αυτό αποτέλεσε το κομβικό σημείο της «επανάστασης» της βαθιάς μάθησης, μιας περιόδου εκθετικής προόδου στην τεχνητή νοημοσύνη που οφείλεται στα βαθιά νευρωνικά δίκτυα.

Αυτά οδηγούν στη δεκαετία του 2010, μια περίοδο με αλματώδη πρόοδο στις δυνατότητες των νευρωνικών δικτύων. Ανάμεσα σε αυτές τις ανακαλύψεις ξεχωρίζει το Convolutional Neural Network (CNN), ένα νευρωνικό δίκτυο που έφερε την επανάσταση στην αναγνώριση εικόνας και στον τομέα της υπολογιστικής όρασης. Τα CNN είναι ειδικά προσαρμοσμένα δίκτυα για να κατανοούν και να ερμηνεύουν περίπλοκα χαρακτηριστικά στις εικόνες. Τα convolutional επίπεδα τεμαχίζουν περίπλοκα μοτίβα μέσω φίλτρων, συλλαμβάνοντας χαρακτηριστικά στοιχείων, όπως ακμές και σχήματα, που προηγουμένως ήταν αδύνατο από οποιουδήποτε είδους υπολογιστικό μοντέλο. Τα CNN ακολουθούνται από επίπεδα ομαδοποίησης, μειώνοντας το μέγεθος των δεδομένων, ενώ ταυτόχρονα διατηρούν όλα τα απαραίτητα στοιχεία των φωτογραφιών. Με τελικό στάδιο τα πλήρως διασυνδεμένα στρώματα, συνθέτουν τα χαρακτηριστικά που έχουν αφομοιωθεί και αποδίδουν εντυπωσιακά αποτελέσματα. Τα CNN είναι αποδεδειγμένα χρήσιμα στη ταξινόμηση εικόνων, την ανίχνευση αντικειμένων, τη δημιουργία εικόνων, τη μεταφορά μάθησης και πολλά άλλα.

Κλείνοντας, επιπρόσθετα σημαντικά ιστορικά σημεία στην εξέλιξη της μηχανικής μάθησης, που οφείλουν να αναφερθούν είναι:

- 1913: Η ανακάλυψη της αλυσίδας Markov.
- 1951: Η δημιουργία της πρώτης μηχανής νευρωνικού δικτύου, SNARC.
- 1967: Η δημιουργία του αλγόριθμου nearest neighbor και η αρχή βασικής αναγνώρισης προτύπων.
- 1970: Η ανακάλυψη του neocognitron, ένας τύπος νευρωνικού δικτύου που αποτελεί έμπνευση για τα CNNs.
- 1985: Η δημιουργία του NETtalk, ενός νευρωνικού δικτύου που εκπαιδεύεται στο πώς να προφέρει λέξεις όπως τα μωρά.
- 1997: Η μηχανή της IBM, Deep Blue, νικάει σε παρτίδα σκάκι τον παγκόσμιο πρωταθλητή στο σκάκι, Gary Kasparov.
- 2002: Η έναρξη λειτουργίας της βιβλιοθήκης Μηχανικής μάθησης Torch.
- 2009: Η δημιουργία μιας μεγάλης βάσης δεδομένων, ImageNet [20]. Η ImageNet θεωρείται από πολλούς ως η αφορμή της «έκρηξης» στην Τεχνητή νοημοσύνη τον 21^{ο} αιώνα.
- 2016: Η AlphaGo της Google είναι η πρώτη μηχανή που νικάει επαγγελματία παίχτη στο επιτραπέζιο Go.
- 2022: Η έναρξη λειτουργίας του μοντέλου ChatGPT, δωρεάν προς όλους.

2.2 - Android development

Το Android είναι ένα λειτουργικό σύστημα ανοικτού κώδικα, κυρίως για κινητές συσκευές, βασισμένο στον πυρήνα του κώδικα του Linux. Ξεκίνησε το 2003, ως Android Inc., από τους Andy Rubin, Chris White, Nick Sears και Rich Miner στο Palo Alto της California, σαν ένα λειτουργικό για κάμερες. Σύντομα, όμως, το 2004 έγινε στροφή προς τα κινητά τηλέφωνα, ακολουθώντας τις ανάγκες της αγοράς. Το Android είχε να αντιμετωπίσει αρκετά δυνατούς «αντιπάλους» τότε, καθώς υπήρχαν αρκετά λειτουργικά που είχαν συνάψει συνεργασίες με μεγάλες εταιρείες. Κάποια από αυτά τα λειτουργικά ήταν: Blackberry, Palm OS, webOS, Windows Mobile και, ίσως ο μεγαλύτερος αντίπαλος όλων, το Symbian. Η μεγάλη διαφορά του Android, σε σχέση με τα υπόλοιπα, ήταν ότι παρέχονταν δωρεάν. Αυτό είχε μεγάλο όφελος στις εταιρείες κινητών τηλεφώνων και σε συνδυασμό με τον ανοιχτό κώδικα που έδινε την ευελιξία στους προγραμματιστές να δημιουργήσουν κάθε είδους εφαρμογή, το Android έγινε ένα πολλά υποσχόμενο λειτουργικό αρκετά νωρίς. Κάτι, το οποίο παρατηρήθηκε άμεσα από την Google η οποία εξαγόρασε την Android Inc. το 2005, μαζί με ολόκληρο το προσωπικό της. Έκτοτε ξεκίνησε η ραγδαία ανάπτυξη και εξέλιξη του Android με πρώτη έκδοση το Android 1.0 στη συσκευή T-Mobile G1 τον Οκτώβριο του 2008. Ένα σημείο το οποίο σηματοδότησε και την έναρξη της εξέλιξης του κώδικα προγραμματισμού για εφαρμογές που λειτουργούν στο Android [7]. Στην

παρακάτω εικόνα, που βρίσκεται στο [13], εμφανίζεται η πρώτη κινητή συσκευή με λειτουργικό σύστημα Android:

Εικόνα 5 : Η T-Mobile G1, πρώτη κινητή συσκευή με το λειτουργικό Android 1.0 [13]

Οι τεχνολογίες ανάπτυξης εφαρμογών του Android ή αλλιώς το Android app development, αναπτύχθηκε παράλληλα με την εξέλιξη του λειτουργικού Android. Ξεκινώντας από το 2008, με γλώσσα προγραμματισμού τη Java και πλατφόρμα ανάπτυξης κώδικα (IDE) το Eclipse, δεν υπήρχε κανένας οδηγός, κατεύθυνση ή κοινή αντίληψη στο πως να αναπτυχθεί μια εφαρμογή. Κάθε προγραμματιστής ή ομάδα προγραμματισμού, έφτιαχνε το δικό του πλάνο ή ακόμη και βιβλιοθήκες, για την υλοποίηση κάποιας εφαρμογής. Αυτή, όμως, η εποχή της πλήρους ανεξαρτησίας, ενώ συνείσφερε στις διαδικασίες ανάπτυξης κώδικα, οδήγησε σε πολύ μεγάλα προβλήματα. Βιβλιοθήκες που έμεναν ανενημέρωτες οδηγούσαν σε εφαρμογές που δε δούλευαν, ο διαμοιρασμός κώδικα μεταξύ των προγραμματιστών ήταν χρονοβόρος και πολλές φορές αδύνατο να συνεχιστεί κάποιο project και πολλά άλλα. Την περίοδο αυτή, η Google, δεν είχε κάποια λύση να προσφέρει πέραν του καινούριου IDE που ανακοίνωσε το 2013, το Android Studio. Συνεπώς, η κοινότητα των προγραμματιστών του Android app development δημιούργησε τη δική της δομή και αρχιτεκτονική, περίπου το 2014. Η αρχιτεκτονική αυτή υπάρχει ακόμη και σήμερα, με αρκετές βελτιστοποιήσεις από τότε. Ωστόσο, το 2017, η Google ανακοίνωσε την Kotlin ως επίσημη γλώσσα προγραμματισμού για τις εφαρμογές και το 2018 ξεκίνησε η λειτουργία της βιβλιοθήκης Jetpack. Πλέον, με οδηγούς προγραμματισμού, αρχιτεκτονική και βιβλιοθήκη που υποστηρίζεται από τη Google, η εκμάθηση, η διατήρηση δομής και ο διαμοιρασμός κώδικα πέρασαν σε ένα ιδιαίτερα οργανωμένο επίπεδο. Λύνοντας, ένα-ένα, όλα τα προβλήματα που προέκυψαν κατά καιρούς τα προηγούμενα χρόνια, το 2019, έρχεται και η επιπρόσθετη βιβλιοθήκη Jetpack Compose που αφορά το User Interface (UI). Βιβλιοθήκη, η

οποία προφέρει έναν εντελώς καινούριο τρόπο σχεδιασμού και προγραμματισμού της εμφάνισης μιας εφαρμογής, μέσω της γλώσσας προγραμματισμού Kotlin και όλα τα οφέλη της [8].

Το Android και Android app development αποτελούν τομείς για αρκετά μεγάλο αριθμό προγραμματιστών και εταιρειών. Πρόκειται για ένα λειτουργικό και εφαρμογές που χρησιμοποιούνται, πλέον, από συσκευές που αφορούν πάνω του 70% της παγκόσμιας αγοράς [13]. Εκτός, όμως, του μεγάλου πλήθους χρήσης, η εξέλιξη που έχει προκύψει ανά τα χρόνια και ιδιαιτέρως τα τελευταία 5 έτη, καθιστά το ίδιο το Android app development αρκετά ενδιαφέρον. Ο κώδικας της Kotlin καθιστά την ανάπτυξη πιο δομημένη και εύκολη, με μεγάλες δυνατότητες εξέλιξης της ίδιας της εφαρμογής ακόμη κι από μικρές ομάδες προγραμματιστών. Κώδικας που μπορεί να δομηθεί, ώστε να είναι επαναχρησιμοποιήσιμος, χωρίς τον φόβο πως κάποια βιβλιοθήκη μπορεί να καταστρέψει ολόκληρη την εφαρμογή όπως και πολλά άλλα προτερήματα.

2.3 - Συνδυασμός τεχνολογιών

Η παρούσα εργασία συνδυάζει τις μεθόδους μηχανικής μάθησης με τις τεχνολογίες ανάπτυξης εφαρμογών για κινητές συσκευές. Η ανάπτυξη του κώδικα των μεθόδων μηχανικής μάθησης και η δημιουργία μοντέλων είναι δυνατή σε διάφορες γλώσσες προγραμματισμού, όπως R, Java, Python κ.α., ενώ η χρήση των μοντέλων αυτών μέσα από τις προγραμματιστικές διεπαφές του Android, δεν είναι τόσο ευέλικτη. Ο τρόπος χρήσης των μοντέλων εξαρτάται από την επιλογή της τεχνολογίας και της γλώσσας ανάπτυξης της εφαρμογής.

Για τον προγραμματισμό της επιλέχθηκε η γλώσσα προγραμματισμού Kotlin και η χρήση της βιβλιοθήκης Jetpack Compose. Η επιλογή αυτή έγινε επειδή είναι η πιο μοντέρνα μέθοδος ανάπτυξης. Ως συνέπεια της προηγούμενης επιλογής προέκυψε και η επιλογή της Python, ως γλώσσας προγραμματισμού για την δημιουργία μοντέλων Μηχανικής Μάθησης. Η βιβλιοθήκη TensorFlow της Python, είναι αυτή που δίνει τη δυνατότητα μετατροπής ενός μοντέλου Keras σε μοντέλο TensorFlow lite. Είδος μοντέλου που μπορεί να ενσωματωθεί στον κώδικα εφαρμογής Android, μέσω της πλατφόρμας του Android studio.

Η βιβλιοθήκη TensorFlow επιτρέπει στους προγραμματιστές να δημιουργούν γράφους ροής δεδομένων. Γράφος ροής δεδομένων είναι η δομή που περιγράφει τον τρόπο με τον οποίο τα δεδομένα κινούνται μέσω μιας σειράς κόμβων επεξεργασίας. Κάθε κόμβος στο γράφο αντιπροσωπεύει μια μαθηματική πράξη και κάθε σύνδεση ή πλευρά ένωσης στους κόμβους είναι ένας πολυδιάστατος πίνακας δεδομένων ή αλλιώς Tensor. Οι εφαρμογές TensorFlow μπορούν να εκτελεστούν σχεδόν σε οποιονδήποτε μηχάνημα ή τεχνολογία: σε ιδιωτικό ηλεκτρονικό υπολογιστή, σε κάποιο server στο cloud, σε συσκευές iOS και Android κ.α., χρησιμοποιώντας CPU ή GPU. Τον Οκτώβριο του 2019 κυκλοφόρησε η TensorFlow 2.0, που

ανανεώθηκε σημαντικά, βάσει των σχολίων της κοινότητας των προγραμματιστών της. Σαν αποτέλεσμα, δημιουργήθηκε ένα πλαίσιο μηχανικής μάθησης, όπου είναι πιο εύκολο να εργαστούν οι προγραμματιστές και είναι ακόμη πιο αποδοτικό. Η κατανεμημένη εκπαίδευση των μοντέλων είναι ευκολότερη εξαιτίας ενός νέου API, ενώ η υποστήριξη για το TensorFlow Lite είναι αυτή που δίνει την δυνατότητα ανάπτυξης μοντέλων σε ακόμη περισσότερες πλατφόρμες [17].

Η TensorFlow υποστηρίζεται στις εκδόσεις 3.7 έως 3.11 της Python. Η λειτουργία της TensorFlow σε παλαιότερες εκδόσεις της Python είναι εφικτή, αλλά δεν συνίσταται καθώς δεν είναι εγγυημένο ότι θα εξάγει σωστά αποτελέσματα. Οι κόμβοι και οι Tensors της TensorFlow είναι Python objects, όπως επίσης και οι εφαρμογές της γίνονται σε Python. Όμως, οι πραγματικές μαθηματικές πράξεις, δεν εκτελούνται στην Python. Οι βιβλιοθήκες μετασχηματισμών που είναι διαθέσιμες μέσω της TensorFlow γράφονται σε C++. Στη Python υλοποιείται η μεταφορά δεδομένων μεταξύ τους. Η εργασία υψηλού επιπέδου στη TensorFlow, όπως: η δημιουργία κόμβων, επιπέδων και η σύνδεσή μεταξύ τους, βασίζεται στη βιβλιοθήκη Keras. Το API της Keras είναι απλό στη χρήση του, καθώς για τον κώδικα που ορίζει ένα βασικό μοντέλο με τρία επίπεδα και για τον κώδικα της εκπαίδευσής του, αρκούν λιγότερες από 20 γραμμές για να διατυπωθεί [17].

Κεφάλαιο 3 - Μοντέλα Μηχανικής Μάθησης

3.1 - Δεδομένα για την εφαρμογή

Είναι ευρύτερα γνωστό στην επιστημονική κοινότητα, πως η εύρεση και συλλογή δεδομένων είναι, αν όχι το δυσκολότερο, ένα από τα δυσκολότερα μέρη μιας εργασίας. Στον ευρύτερο τομέα της βλάστησης, έχουν γίνει κάποιες ενέργειες ώστε να υπάρξουν δεδομένα, ανά τα χρόνια. Στον παρακάτω πίνακα, που βρίσκεται στο [9], αναφέρονται τα μεγαλύτερα σύνολα δεδομένων που έχουν δημιουργηθεί σε προηγούμενα έτη:

Dataset	Num. of Taxa	Total Num. of Images	Organ	Life Form
Swedish leaf	15	1125	leaf	tree
Flavia	32	1907	leaf	tree
Leafsnap	185	30,866	leaf	tree
ICL	220	17,032	leaf	tree, herb
Oxford Flower 17	17	1360	flower	herb
Oxford Flower 102	102	8189	flower	herb
Jena Flower 30	30	1479	flower	herb
German flowering plants	101	50,500	flower, leaf	herb
PlantCLEF2015/2016	1000	113,205	leaf, flower, fruit, stem	tree, herb, fern
GRASP-125	125	16,367	leaf, flower, fruit, stem	tree, herb, fern

Πίνακας 1 : Διάσημα σύνολα δεδομένων φωτογραφιών βλάστησης [9]

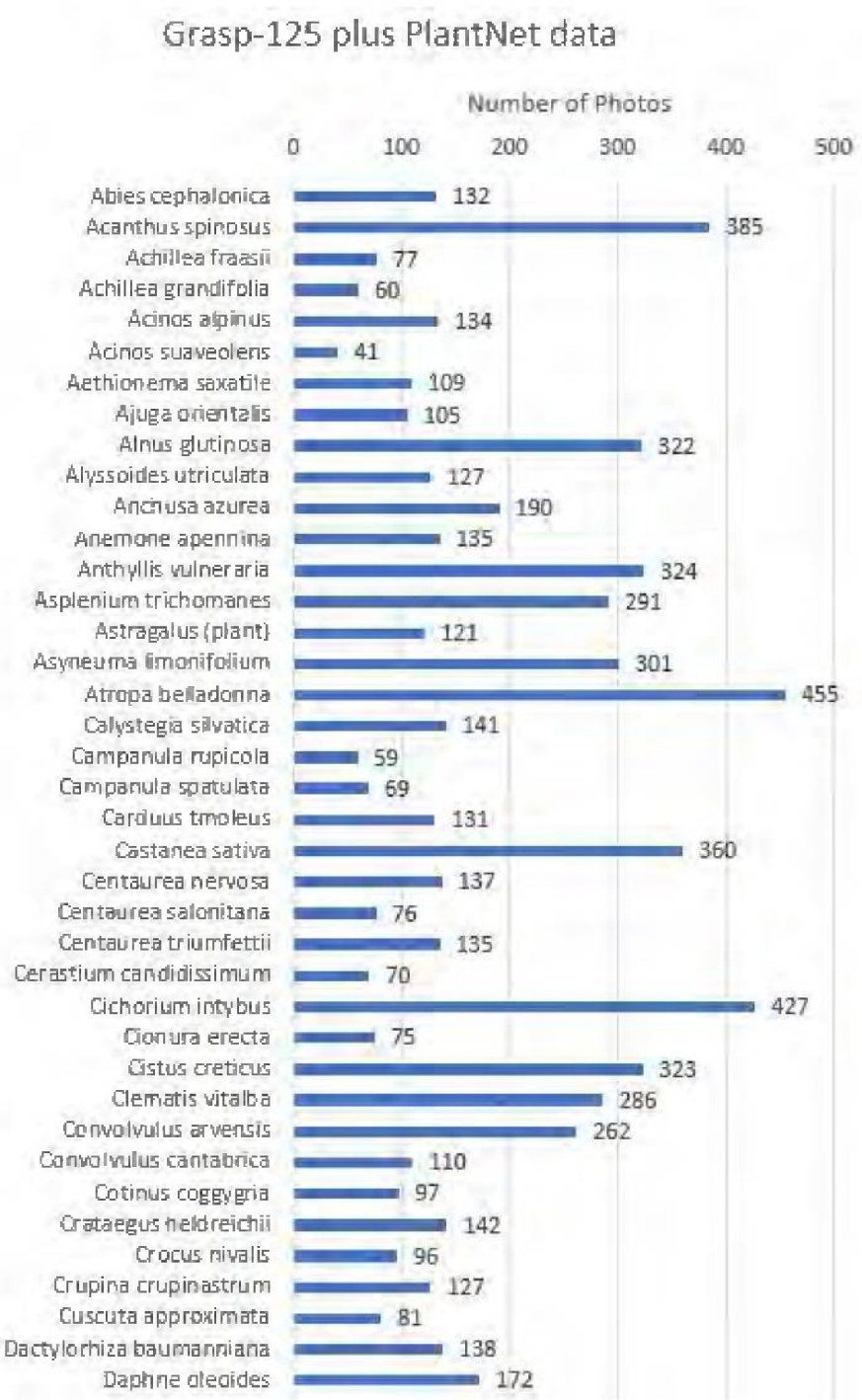
Πραγματοποιώντας μια πρώτη ανάλυση των δεδομένων αυτών, πρόκειται για δεδομένα εικόνων από το χώρο της Ευρώπης. Σύμφωνα με τις ενδείξεις των τελευταίων 2 στηλών του πίνακα κατηγοριοποιούνται ως προς τη διαφορετικότητα του περιεχομένου των εικόνων. Τα σύνολα δεδομένων: Swedish leaf, Flavia, Leafsnap και ICL αφορούν το σχήμα και τη μορφοποίηση των φύλλων διαφόρων δέντρων και μερικών βοτάνων. Τα επόμενα 4 σύνολα: Oxford Flower 17, Oxford Flower 102, Jena Flower 30 και German flowering plants περιέχουν εικόνες κυρίως των ανθών, αλλά και μερικές φωτογραφίες των φύλλων από βότανα και ευεργετικά λουλούδια κατά τη περίοδο άνθισής τους. Τέλος, τα τελευταία 2 σύνολα δεδομένων PlantCLEF2015/2016 και GRASP-125, πρόκειται για σύνολα φωτογραφιών με ποικιλόμορφες φωτογραφίες. Περιέχουν φύλλα, σχήματα κορμού, άνθη και καρπούς από βότανα, ευεργετικά λουλούδια και φυτά που εντοπίζονται σε εξωτερικό χώρο, όπου και φωτογραφήθηκαν κατά τη διάρκεια της άνθισής τους [9].

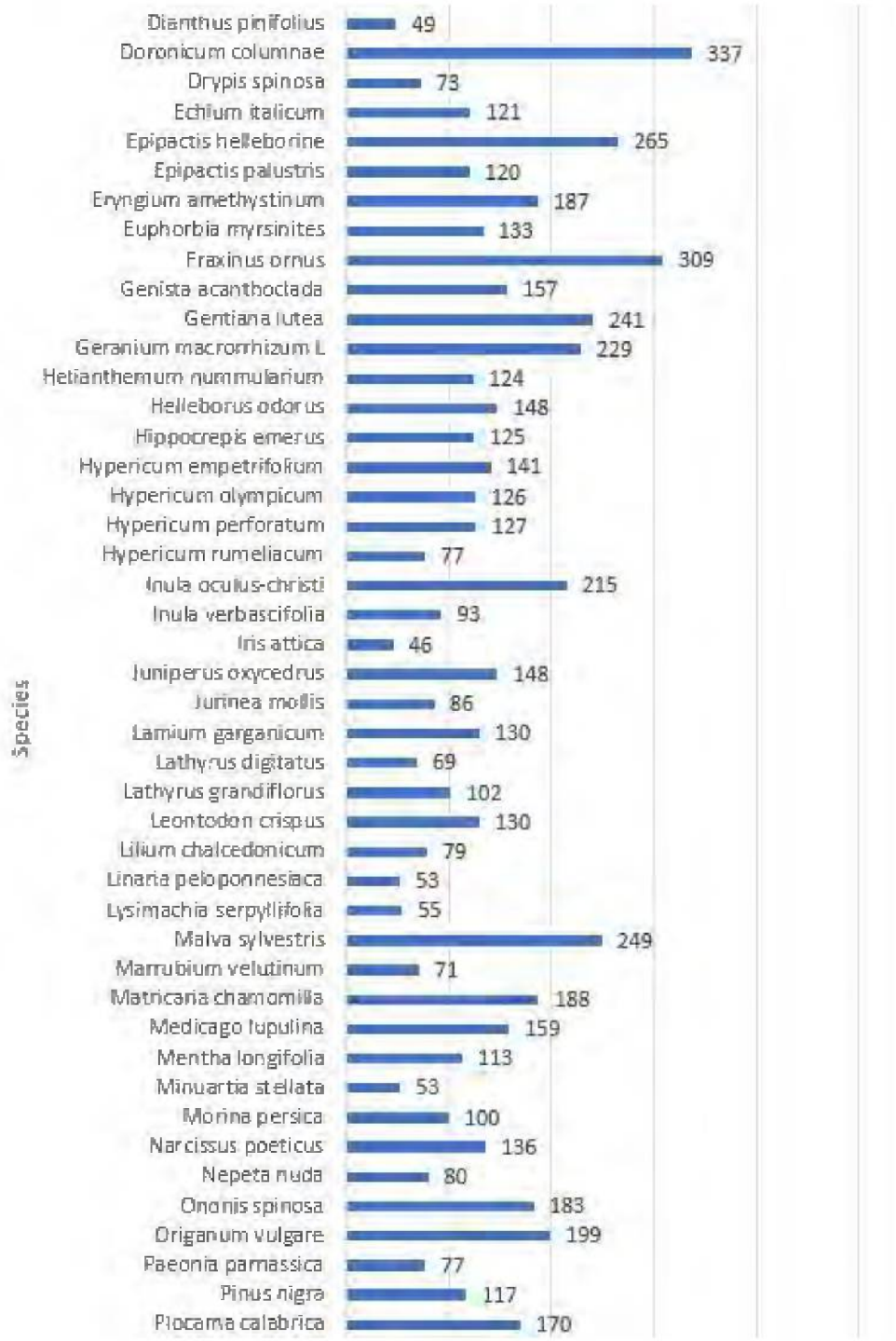
Συνεπώς, σύμφωνα με το σκοπό της εφαρμογής, τα τελευταία 2 σύνολα δεδομένων είναι αυτά που παρουσιάζουν το μεγαλύτερο ενδιαφέρον. Το PlantCLEF2015/2016, παρόλο που είναι

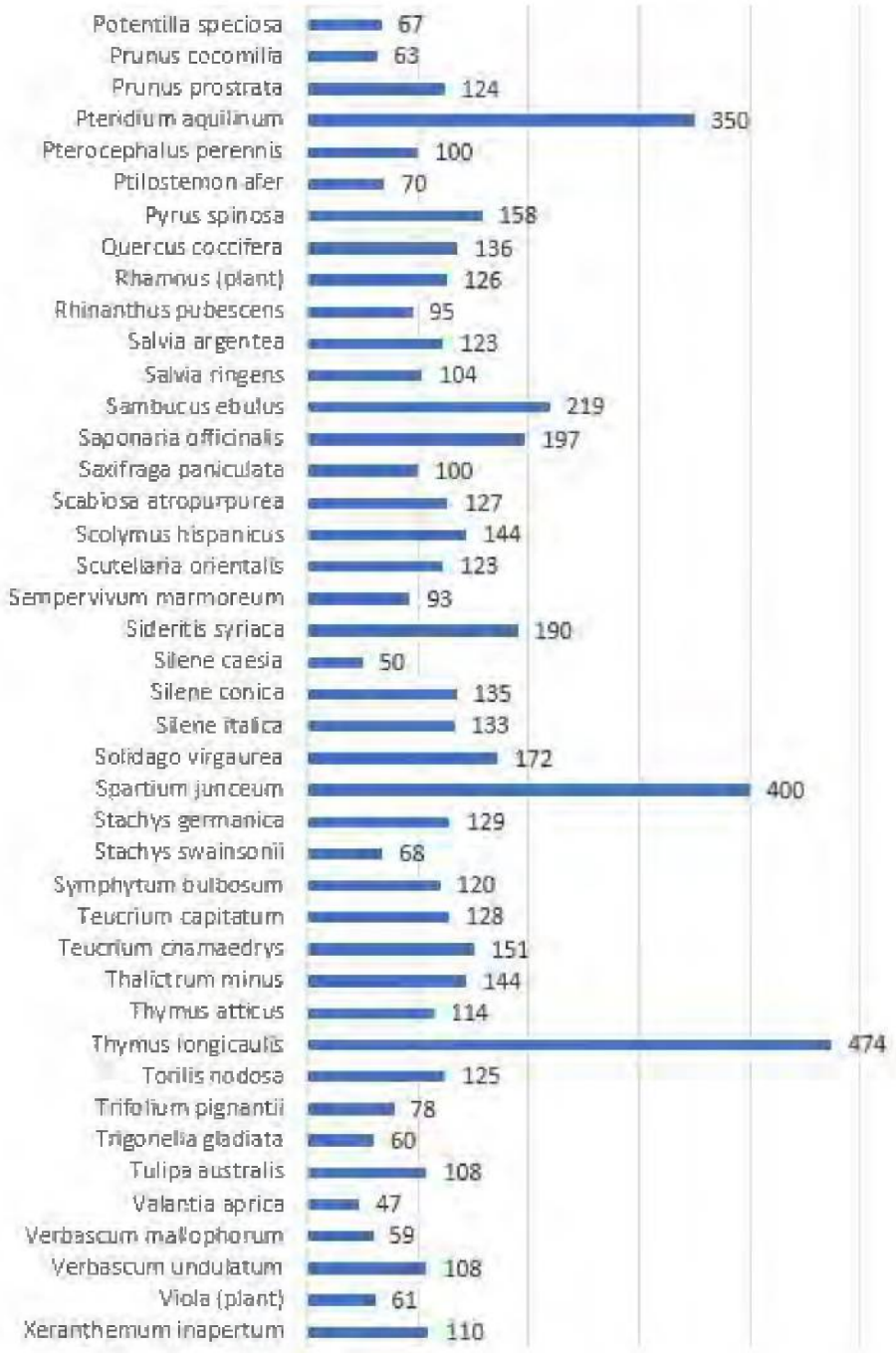
εκτενές, περιέχει κυρίως βότανα που εκτείνονται στο έδαφος της Γαλλίας. Από την άλλη πλευρά, το GRASP-125 περιέχει δεδομένα φυτών που συναντώνται στον ελληνικό χώρο, γεγονός που το καθίστησε πιο ενδιαφέρον.

Σχετικά με το σύνολο δεδομένων GRASP-125, όπως υποδεικνύει και το όνομα του, πρόκειται για ένα σύνολο δεδομένων από 125 διαφορετικά είδη βοτάνων, λουλουδιών, θάμνων και δένδρων. Τα δεδομένα έχουν συλλεχθεί από ειδήμονες βοτανολόγους καθώς κι από Web-crawling με διαλογή φωτογραφιών που αρμόζουν για τους σκοπούς του [9]. Πρόκειται για ένα μη ισορροπημένο σύνολο δεδομένων, με μικρότερο αριθμό φωτογραφιών τις 40 στο είδος Acinos suaveolens και μεγαλύτερο αριθμό τις 474 στο Thymus longicaulis. Συνολικά περιέχει 16.327 φωτογραφίες [9].

Στην παρούσα εργασία, έγινε μια προσπάθεια εξισορρόπησης του GRASP-125 με προσεκτική επιλογή ανά είδος, από δύο ιστοσελίδες περιεχομένου βλάστησης, τις Pl@ntNet.org [19] και greekflora.gr. Πρόκειται για ιστοσελίδες που ειδικεύονται στον τομέα της βλάστησης και ελέγχονται από βοτανολόγους. Στη βάση δεδομένων της Pl@ntNet συνεισφέρει ένας μεγάλος αριθμός ανθρώπων που δεν είναι απαραίτητα βοτανολόγοι. Ωστόσο, γίνεται έλεγχος και ορθή κατηγοριοποίηση των φωτογραφιών από βοτανολόγους [19]. Η Pl@ntNet είναι, επίσης, η ιστοσελίδα απ’ όπου λήφθηκε το μεγαλύτερο ποσοστό των νέων φωτογραφιών. Η ιστοσελίδα greekflora.gr, πρόκειται για την ελληνική ιστοσελίδα βλάστησης που οι ίδιοι βοτανολόγοι ανανεώνουν. Το σύνολο των φωτογραφιών αυξήθηκε από 16.327 σε 18.696 φωτογραφίες, με πρόσθεση ενός ακόμη είδους, συνεπώς πλέον υπάρχουν 126 είδη. Ωστόσο, δεν επετεύχθη η εξισορρόπηση του συνόλου των δεδομένων στο επιθυμητό επίπεδο καθώς συνεχίζει να περιλαμβάνει είδη με μικρό αριθμό φωτογραφιών. Συνεπώς, ο ελάχιστος αριθμός των δεδομένων παρέμεινε μικρός, στις 41 φωτογραφίες και ο μεγαλύτερος στις 474 στα ίδια είδη με πριν. Ο μέσος όρος φωτογραφιών ανά είδος είναι 148 με τυπική απόκλιση 91 φωτογραφίες. Αυτή η αύξηση του αριθμού των φωτογραφιών, παρόλο που δεν εξισορρόπησε το σύνολο, στη συνέχεια βοήθησε τους αλγόριθμους να έχουν καλύτερα αποτελέσματα αναγνώρισης. Στις επόμενες σελίδες δίνεται το αναλυτικό ραβδόγραμμα του τελικού συνόλου φωτογραφιών.







Εικόνα 6 : Ραβδόγραμμα συνόλου δεδομένων της Herb Tracker

3.2 - Γενικά Μοντέλα μηχανικής μάθησης αναγνώρισης εικόνων για κινητά

Η μηχανική μάθηση εκτείνεται σε διάφορους τομείς και οι αλγόριθμοί που υπάρχουν μπορούν να χρησιμοποιηθούν για διάφορες λειτουργίες. Όμως, δεν ενδείκνυνται όλοι για την αναγνώριση εικόνων. Όπως προαναφέρθηκε, η βάση για την αναγνώριση εικόνων είναι τα Convolutional Neural Networks (CNN) [16], όμως, υπάρχουν διάφορες αρχιτεκτονικές και μέθοδοι αναγνώρισης εικόνων που μπορούν να αποδώσουν διαφορετικά αποτελέσματα [18]. Αναλόγως το σύνολο το δεδομένων, των παραμετροποιήσεων και του σκοπού που είναι επιθυμητός να επιτευχθεί, πρέπει να γίνεται και η ανάλογη επιλογή αρχιτεκτονικής [9, 16, 18]. Κατά την αναζήτηση στο διαδίκτυο, μερικές απ' τις πιο διαδεδομένες αρχιτεκτονικές που βρέθηκαν και χρησιμοποιούνται, με αρκετά καλά αποτελέσματα αναγνώρισης, δίνονται στον παρακάτω πίνακα με βάση τη χρονολογία που προτάθηκαν:

Model Architecture	Year Inv.	Developed by	Parameters (Millions)	Top-5 Error (%)
EfficientNetV2	2020	Google	-	-
EfficientNet	2019	Mingxing Tan and Quoc V. Le	66	4.3
MobileNetV2	2018	Google	3.5	9.46
NASNet	2017	Google	-	-
ShuffleNet	2017	Megvii	5.4	7.6
DenseNet	2016	Gao Huang et al.	-	-
ResNeXt	2016	Facebook AI Research	-	-
SqueezeNet	2016	DeepScale	1.25	4.7
Xception	2016	Google	22.9	5.5
ResNet50	2015	Microsoft Research	25.6	5.7
Inception v3	2015	Google	23.8	5.5
VGG16	2014	Visual Geometry Group	138	7.3
GoogLeNet (Inception v1)	2014	Google	6.7	6.7
AlexNet	2012	Alex Krizhevsky et al.	60	16.4
LeNet	1998	Yann LeCun	-	-

Πίνακας 2 : Αρχιτεκτονικές αλγορίθμων αναγνώρισης εικόνων

Παρακάτω εξηγούνται οι τελευταίες 2 στήλες του πίνακα:

Parameters (Millions): Είναι οι εσωτερικές μεταβλητές εκμάθησης του αλγορίθμου (σε εκατομμύρια) που χρησιμοποιούνται κατά τη διάρκεια της εκπαίδευσής του. Περιέχουν τα βάρη (weights) και τα λάθη προβλέψεως (bias) που μεταφέρονται ή/και βελτιώνονται κατά τη μετάβασή τους στα layers. Όσο μεγαλύτερος ο αριθμός, τόσο αυξάνεται η πολυπλοκότητα των πράξεων και η υπολογιστική ισχύς που απαιτούν, αλλά και η δυνατότητα αναγνώρισης πιο περίπλοκων μοτίβων.

Top-5 Error (%): Το ποσοστό σφάλματος Top-5 αντιπροσωπεύει το ποσοστό των εικόνων που δοκιμάστηκαν στο μοντέλο, για τις οποίες η σωστή αναγνώριση δεν περιλαμβάνεται στις 5 κορυφαίες προβλέψεις του μοντέλου. Πρόκειται για μία μετρική αξιολόγησης των μοντέλων αναγνώρισης εικόνων, όπου όσο μικρότερο είναι το ποσοστό, τόσο καλύτερο είναι το μοντέλο σχετικά με τις προβλέψεις ή αναγνωρίσεις του.

Παρόλη την ύπαρξη ποικίλων αλγορίθμων που επιλύουν το πρόβλημα της αναγνώρισης εικόνας, δεν είναι εφικτό να χρησιμοποιηθεί οποιοσδήποτε αλγόριθμος σε οποιαδήποτε συσκευή στον πραγματικό κόσμο. Τα δύο κριτήρια που πρέπει να λαμβάνονται υπόψη για εφαρμογές Android, είναι: το μέγεθος των εσωτερικών μεταβλητών, που, όπως προαναφέραμε αυξάνουν το μέγεθος του μοντέλου και την υπολογιστική ισχύ, αλλά και το επιθυμητό αποτέλεσμα βάσει χρόνου εκτέλεσης και ακρίβειας αναγνώρισης. Οι αρχιτεκτονικές που αρμόζουν και πληρούν τα κριτήρια για την ανάπτυξη ενός τέτοιου μοντέλου είναι οι: MobileNetV2, EfficientNetB0, NasNetMobile και DenseNet121 [9].

3.3 - Επιλογή Μοντέλων μηχανικής μάθησης στα δεδομένα της εφαρμογής και οι αρχιτεκτονικές τους

Μετά την ανάλυση των αλγορίθμων αναγνώρισης εικόνων και την επιλογή του κατάλληλου συνόλου δεδομένων, το επόμενο σημαντικό βήμα ήταν η υλοποίηση αλγορίθμων. Αυτή η επιλογή βασίστηκε στην ανάδειξη διαφορετικών αρχιτεκτονικών και δόμησης μοντέλων, αλλά και στην αποτελεσματικότητα τους σχετικά με τις αναγνωρίσεις τους. Για το σκοπό αυτό, επιλέχθηκαν δύο μοντέλα που θα χρησιμοποιηθούν: το MobileNetV2 και το DenseNet121. Οι αρχιτεκτονικές των μοντέλων αυτών, θα αναλυθούν λεπτομερώς στα επόμενα μέρη, με σκοπό τη διευκρίνιση της διαφορετικότητας τους. Η διαδικασία αυτή θα παράγει ένα λεπτομερώς κατανοητό υπόβαθρο, για την εφαρμογή των συγκεκριμένων αλγορίθμων αναγνώρισης εικόνων στα επιλεγμένα δεδομένα. Παρακάτω δίνεται ο πίνακας πληροφοριών των 2 μοντέλων:

Model Architecture	Year Inv.	Developed by	Parameters (Mils)	Top-5 Error (%)	Layers
DenseNet121	2016	Gao Huang et al.	7.6	7.83	121
MobileNetV2	2018	Google	3.5	9.46	53

Πίνακας 3 : Πληροφορίες μοντέλων μηχανικής μάθησης της Herb Tracker

3.3.1 - MobileNetV2

Η αρχιτεκτονική MobileNetV2 περιέχει ένα αρχικό convolutional επίπεδο με 32 φίλτρα και έπονται 19 επίπεδα residual bottleneck με χρήση της Rectified Linear Unit (ReLU) ως μη-γραμμική συνδεσιμότητα. Η ReLU είναι μια activation συνάρτηση, η οποία αποδίδει εξαιρετικά αποτελέσματα όταν πρόκειται για υπολογισμούς χαμηλής ακρίβειας [10]. Το μαθηματικό μοντέλο της συνάρτησης αυτής είναι:

$$f(x) = x^+ = \max(0, x) = \frac{x+|x|}{2} = \begin{cases} x & if\ x > 0, \\ 0 & otherwise \end{cases}$$ *, όπου x η είσοδος ενός layer*

Στον παρακάτω πίνακα, που αντλήθηκε από το [10], αναγράφεται αναλυτικά η αρχιτεκτονική MobileNetV2:

Input	Operator	t	c	n	s
224 x 224 x 3	conv2d	-	32	1	2
112 x 112 x 32	bottleneck	1	16	1	1
112 x 112 x 16	bottleneck	6	24	2	2
56 x 56 x 24	bottleneck	6	32	3	2
28 x 28 x 32	bottleneck	6	64	4	2
14 x 14 x 64	bottleneck	6	96	3	1
14 x 14 x 96	bottleneck	6	160	3	2
7 x 7 x 160	bottleneck	6	320	1	1
7 x 7 x 320	conv2d 1x1	-	1280	1	1
7 x 7 x 1280	avgpool 7x7	-	-	1	-
1 x 1 x 1280	conv2d 1x1	-	k	-	

Πίνακας 4 : Βασική Αρχιτεκτονική MobileNetV2 [10]

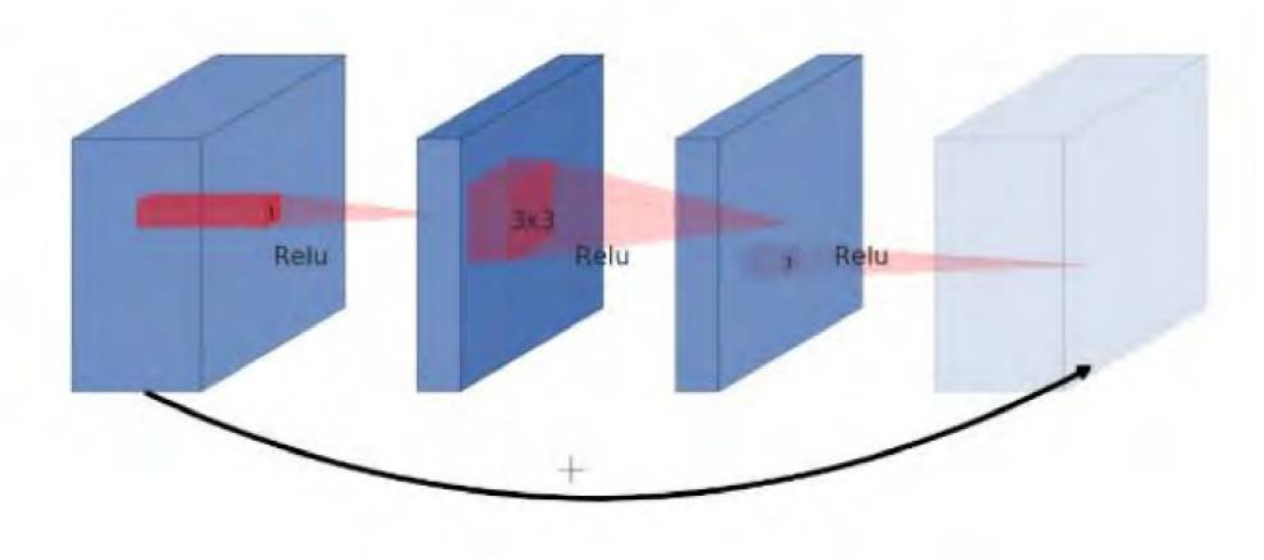
Κάθε γραμμή του πίνακα 4, αναπαριστά μια ακολουθία από ένα ή παραπάνω πανομοιότυπα επίπεδα τα οποία επαναλαμβάνονται n φορές. Όλα τα επίπεδα που βρίσκονται στην ίδια ακολουθία έχουν τον ίδιο αριθμό εξόδου καναλιών c. Το πρώτο επίπεδο κάθε ακολουθίας έχει διασκελισμό (stride) s, ενώ για όλα τα υπόλοιπα ισχύει s = 1. Ο συντελεστής επέκτασης t, ο οποίος εφαρμόζεται στο μέγεθος εισόδου, καθώς και όλα τα spatial convolutions που εχουν πυρήνες 3x3 αναλύονται στον παρακάτω πίνακα που υπάρχει στο [10]:

Input	Operator	Output
h x w x k	1x1 conv2d, ReLU6	h x w x (tk)
h x w x tk	3x3 dwises= s, ReLU6	(h/s) x (w/s) x (tk)
(h/s) x (w/s) x lk	linear 1x1 conv2d	(h/s) x (w/s) x k'

Πίνακας 5 : Ανάλυση επιπέδων residual bottleneck [10]

Ο πίνακας 5 είναι η ανάλυση των επιπέδων bottleneck σε μια συσχέτιση μετατροπής των k καναλιών σε k', με stride s και συντελεστή επέκτασης l [10]. Η ReLU6 που αναγράφεται στον πίνακα είναι μια παραλλαγή της ReLU, όπου αφαιρείται το κάτω μέρος του εύρους τιμών και διατηρούνται μόνο οι θετικές τιμές. Το μέγεθος εξόδου της ReLU περιορίζεται στον αριθμό 6, από όπου παίρνει την ονομασία του, και αυτό μαθηματικά μετατρέπεται σε $f(x) = \min(\max(0, x), 6)$ [14].

Στη συνέχεια, με σκοπό την ουσιαστική κατανόηση του αλγορίθμου, αναπαρίσταται οπτικά και επεξηγείται μαθηματικά η διαδικασία λειτουργίας των residual bottlenecks. Αυτά είναι και τα κύρια επίπεδα υπολογισμών για την αναγνώριση των στοιχείων των φωτογραφιών. Η παρακάτω εικόνα βρίσκεται στο [10]:



Εικόνα 7 : Οπτική προσομοίωση του bottleneck residual block [10]

Bottleneck Residual Block:

Ένας τελεστής F(x) του bottleneck block, όπως στην παραπάνω εικόνα, μπορεί να διατυπωθεί ως σύνθεση τριών τελεστών $F(x) = [A \circ N \circ B]\, x$. Ο A είναι ένας γραμμικός μετασχηματισμός $A : R^{s*s*k} \rightarrow R^{s*s*n}$, ο N είναι ένας μη γραμμικός ανασχηματισμός ανά κανάλι: $N : R^{s*s*n} \rightarrow R^{s'*s'*n}$ και ο B είναι ακόμη ένας γραμμικός μετασχηματισμός της εξόδου: $B : R^{s'*s'*n} \rightarrow R^{s'*s'*k'}$ [10].

Πιο συγκεκριμένα, στο δικό μας δίκτυο $N = \mathrm{ReLU6} \circ \mathrm{dwise} \circ \mathrm{ReLU6}$. Θεωρώντας ότι το μέγεθος εισόδου είναι |x| και το μέγεθος εξόδου |y|, τότε η ελάχιστη μνήμη που απαιτείται για τον υπολογισμό του F (X) είναι $|s^2k| + |s'^2k'| + O(\max(s^2, s'^2))$ [10].

Ο αλγόριθμος βασίζεται στην αναπαράσταση του εσωτερικού πολυδιάστατου πίνακα δεδομένων (tensor), ως μια αλληλουχία *t* tensors, μεγέθους n/t ο καθένας και η συνάρτηση είναι $F(x) = \sum_{i=1}^{t}(A_i \circ N \circ B_i)(x)$. Εφ' όσον υπολογίζεται το σύνολο, τότε, στη μνήμη μπορεί να διατηρείται συνεχόμενα μόνο ένας ενδιάμεσος tensor μεγέθους n/t, καταλαμβάνοντας έτσι μικρότερη μνήμη [10].

Η διαφορά αυτού του αλγορίθμου απ' τους υπόλοιπους, έγκειται στην δυνατότητα να τεθεί n = t. Γεγονός που διευκολύνει τη διαδικασία διατηρώντας ένα μόνο κανάλι για την ενδιάμεση απεικόνιση των δεδομένων καθ' όλη τη διάρκεια και την εναλλαγή επιπέδων. Αυτό προκύπτει εν πρώτης από τον εσωτερικό μετασχηματισμό που γίνεται ανά κανάλι, με μη-γραμμικό τρόπο και σε βάθος (ReLU6 και 3x3 dwise). Όπως επίσης και από τη σημαντική αναλογία των διαδοχικών τελεστών της εισόδου ως προς την έξοδο, που δεν υλοποιούνται ανά κανάλι [10].

Μετά τις απαραίτητες παραμετροποιήσεις για το σύνολο των δεδομένων, η τελική αρχιτεκτονική που χρησιμοποιήθηκε, περιγράφεται στην παρακάτω εικόνα:

Εικόνα 8 : Τελική αρχιτεκτονική MobileNetV2

Μετά το τελευταίο γενικό pooling, προστέθηκε η τεχνική dropout. Πρόκειται για μια τεχνική μείωσης του overfitting των δεδομένων. Έπειτα του dropout, προστέθηκε ένα επίπεδο Dense με αριθμό 126. Ο αριθμός αυτός αντιπροσωπεύει τα διαφορετικά είδη του συνόλου δεδομένων για

τα οποία αποδίδεται πιθανότητα αναγνώρισης. Ως activation συνάρτηση για την απόδοση πιθανότητας αναγνώρισης, τέθηκε ο Softmax, ο οποίος αποδίδει την συνολική πιθανότητα ίση με 1. Η πιθανότητα αυτή διαμοιράζεται στις διαφορετικές ετικέτες του συνόλου. Όσο μεγαλύτερη η πιθανότητα, τόσο πιο πιθανό η αναγνώριση της φωτογραφίας που του δόθηκε να είναι σωστή. Τέλος, ως αλγόριθμος βελτιστοποίησης χρησιμοποιήθηκε ο Adam με αρχική τιμή learning rate το 0.016 και μειώνεται κατά τη πρόοδο των epochs της εκπαίδευσης.

3.3.2 - DenseNet-121

Η αρχιτεκτονική του DenseNet-121 χρησιμοποιεί αρχικά ένα Convolution επίπεδο με 64 φίλτρα μεγέθους 7x7 και διασκελισμό (stride) 2. Ακολουθεί ένα επίπεδο με 3x3 max pool με stride 2 και έπονται 4 Dense blocks, με 3 ενδιάμεσα Transition επίπεδα που μειώνουν το μέγεθος εξόδου [11]. Στον παρακάτω πίνακα, που αντλήθηκε από το [11], δίνεται αναλυτικά η αρχιτεκτονική του:

Layers	Output Size	DenseNet-121
Convolution	112x112	7x7 conv, stride 2
Pooling	56x56	3x3 max pool, stride 2
Dense Block (1)	56x56	$\begin{matrix}1x1\ conv\\3x3\ conv\end{matrix}$ x 6
Transition Layer (1)	56x56 28x28	1x1 conv 2x2 average pool, stride 2
Dense Block (2)	28x28	$\begin{matrix}1x1\ conv\\3x3\ conv\end{matrix} x\ 12$
Transition Layer (2)	28x28 14x14	1x1 conv 2x2 average pool, stride 2
Dense Block (3)	14x14	$\begin{matrix}1x1\ conv\\3x3\ conv\end{matrix} x\ 24$
Transition Layer (3)	14x14 7x7	1x1 conv 2x2 average pool, stride 2
Dense Block (4)	7x7	$\begin{matrix}1x1\ conv\\3x3\ conv\end{matrix} x\ 16$
Classification Layer	1x1	7x7 global average pool 1000D fully-connected, softmax

Πίνακας 6 : Βασική αρχιτεκτονική DenseNet-121 [11]

Για να γίνει πιο κατανοητή η παραπάνω αρχιτεκτονική, αρκεί να αναλύσουμε το κύριο κομμάτι όπου γίνονται οι υπολογισμοί. Αυτό περιλαμβάνει τα Dense blocks και τη μεταφορά πληροφορίας με τα Transition επίπεδα.

Dense block: Σε κάθε block, υπάρχουν 2 convolutions με πυρήνες μεγέθους 1x1 και 3x3. Στα convolutions 1x1, έχουμε 4 φορές τα φίλτρα που υπάρχουν στα 3x3 και επίσης στο τέλος κάθε

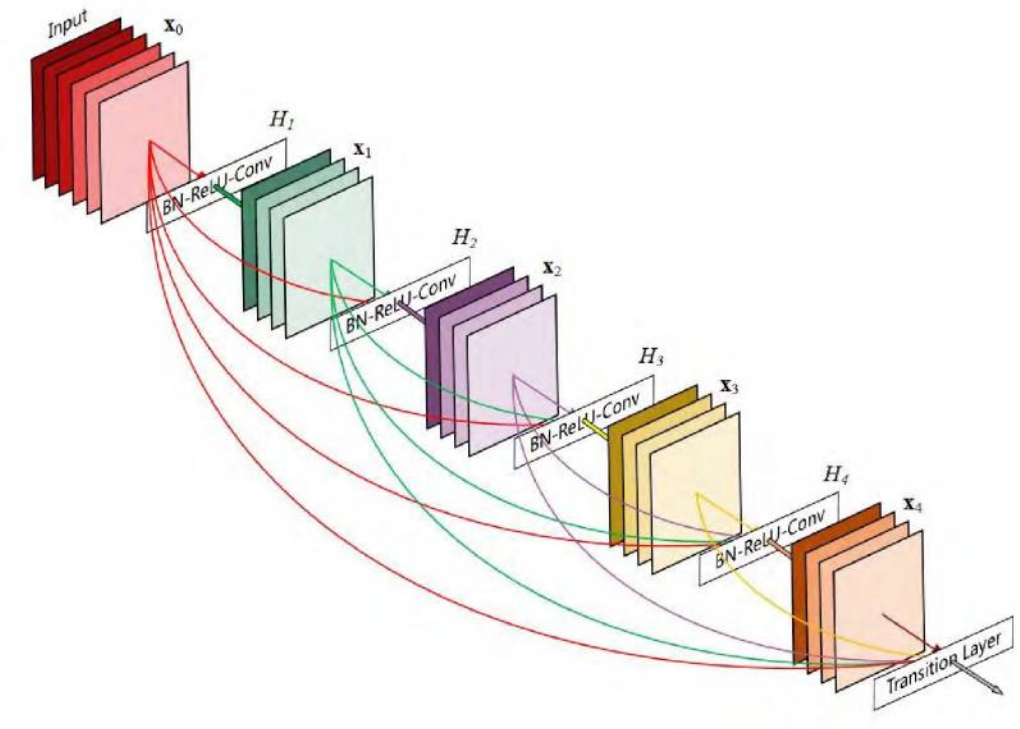
block γίνεται η ένωση των πολυδιάστατων πινάκων (tensors) της εισόδου και εξόδου. Κάθε block επαναλαμβάνει τη διαδικασία για διαφορετικό αριθμό επαναλήψεων: 6, 12, 24 και 16, κατά σειρά και αντίστοιχα [11].

Αν θεωρηθεί ότι εισάγεται μια φωτογραφία x_0 σε ένα convolution δίκτυο, τότε το δίκτυο σχηματίζει L επίπεδα. Σε κάθε επίπεδο υλοποιείται ένας μη γραμμικός μετασχηματισμός $H_\lambda(\cdot)$, με λ τον εκάστοτε αριθμό επιπέδου. Συνεπώς, η έξοδος του λ-οστού επιπέδου θα είναι x_λ. Ο μετασχηματισμός $H_\lambda(\cdot)$ μπορεί είναι μια σύνθετη συνάρτηση 3 διαφορετικών πράξεων:

1. Batch Normalization (BN)
2. Rectified Linear Units (ReLU)
3. Pooling ή Convolution (Conv)

Υπάρχει άμεση σύνδεση των επιπέδων με όλα τα επόμενα επίπεδα, όπως φαίνεται στην εικόνα 9 σε μια αναπαράσταση 5 επιπέδων, η οποία εικόνα βρίσκεται στο [11]. Συνεπώς, στην είσοδο του λ-οστού επιπέδου καταλήγει όλη η απαραίτητη πληροφορία των στοιχείων της φωτογραφίας των προηγούμενων επιπέδων. Αυτό μαθηματικά αποδίδεται ως $x_\lambda = H_\lambda([x_0, x_1, \ldots, x_{\lambda-1}])$, όπου [x0, x1, . . . , xλ−1] πρόκειται για την ένωση της απαραίτητης πληροφορίας της φωτογραφίας που παράχθηκε στα επίπεδα 0, . . . , λ−1 [11].

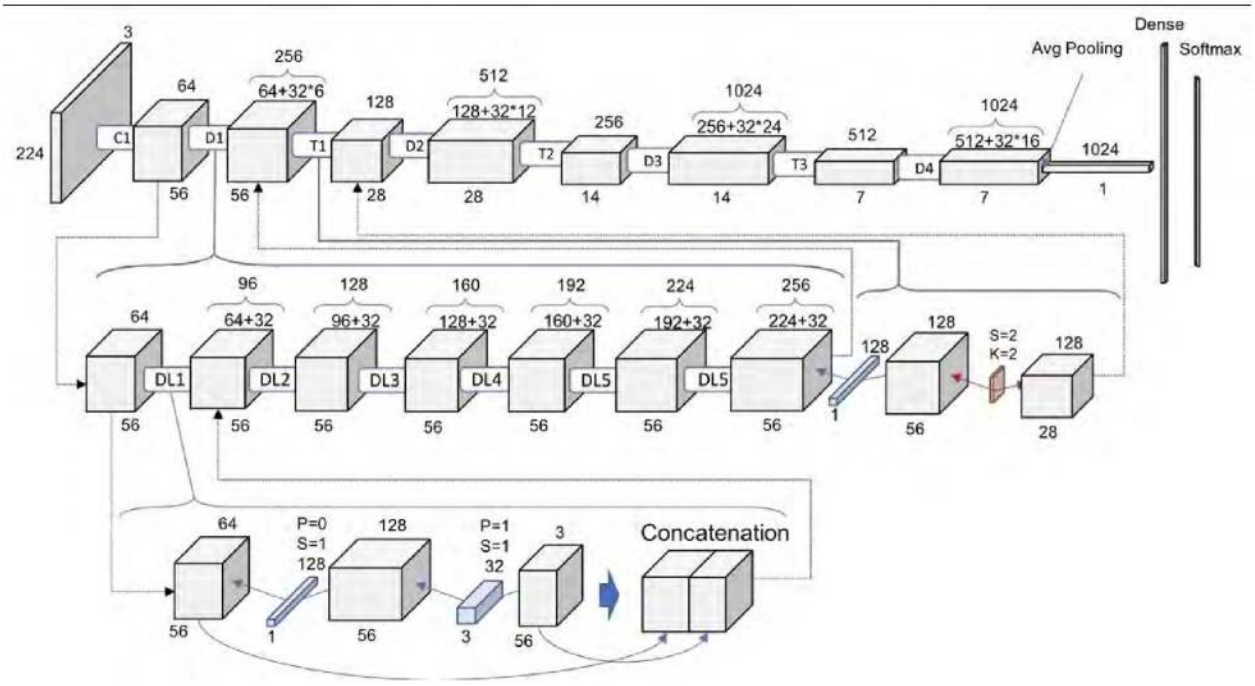
Για την διευκόλυνση της υλοποίησης του μετασχηματισμού προγραμματιστικά, οι πολλαπλές είσοδοι του $H_\lambda(\cdot)$ δίδονται ως πολυδιάστατοι πίνακες (tensors).



Εικόνα 9 : Οπτική αναπαράσταση του DenseNet για 5 επίπεδα [11]

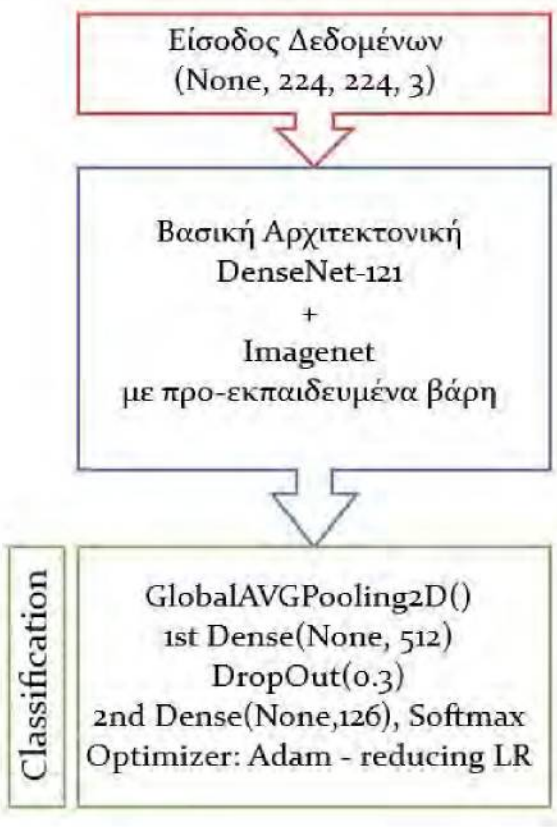
Transition επίπεδα: Σε αυτά τα επίπεδα υπάρχει ένα convolutional επίπεδο με πυρήνα μεγέθους 1x1, το οποίο υλοποιείται με BN + ReLU + 3x3 Conv. Ακόμη, υπάρχει ένα 2x2 average pooling επίπεδο με τιμή stride ίση με 2. Τα convolutional επίπεδα, με την ένωση της απαραίτητης πληροφορίας, βοηθούν στην μείωση της δειγματοληψίας, χωρίς όμως να μειώνεται το μέγεθος. Τα pooling επίπεδα είναι αυτά που βοηθούν στην ελάττωση του μεγέθους, συγκεντρώνοντας την απαραίτητη πληροφορία των convolution επιπέδων στο μισό μέγεθος του πίνακα που δόθηκε. Αυτός είναι και ο λόγος που γίνεται ο διαχωρισμός σε 4 Dense blocks με 3 Transition επίπεδα, ώστε η ελάττωση να γίνεται σταδιακά λαμβάνοντας τη σωστή πληροφορία. Συγκεκριμένα, σε μια είσοδο tensor x, για την μείωση των καναλιών στα μισά, εφαρμόζεται η συνάρτηση Keras backend (K), η οποία επιστρέφει τις διαστάσεις του x. Από τις διαστάσεις που επιστρέφονται, η τελευταία διάσταση είναι αυτή που μας ενδιαφέρει και διαιρείται στο μισό. Το μέγεθος της διάστασης αυτής αξιοποιείται και με ένα δεύτερο τρόπο καθώς είναι ένας αριθμός ίδιος με τον αριθμό των φίλτρων που είναι απαραίτητα για το επόμενο επίπεδο [11].

Στην παρακάτω εικόνα, που αντλήθηκε από το [12], δίνεται η λειτουργικότητα του DenseNet-121 σε βάθος, για την περαιτέρω κατανόηση των προηγουμένων.



Εικόνα 10 : Οπτική αναπαράσταση λειτουργίας του DenseNet-121 [12]

Κάνοντας χρήση της βασικής αρχιτεκτονικής που μόλις αναλύθηκε και με βάση το σύνολο δεδομένων, προστέθηκαν τρία τελευταία επίπεδα που αφορούν την έξοδο της αναγνώρισης. Έπειτα από το τελευταίο γενικό pooling προστέθηκε ένα επίπεδο Dense block διάστασης 512, συνεχίζει η τεχνική dropout με συντελεστή 0.3 και τέλος ένα δεύτερο Dense block με τον αριθμό των 126 ειδών και activation συνάρτηση τη Softmax, που αποδίδει την επιθυμητή πιθανότητα. Και σε αυτή τη περίπτωση, ως αλγόριθμος βελτιστοποίησης της ακρίβειας των αποτελεσμάτων, χρησιμοποιήθηκε ο Adam με μείωση του learning rate και αρχική τιμή το 0.004. Στην εικόνα 11 απεικονίζεται η τελική αρχιτεκτονική που χρησιμοποιήθηκε.



Εικόνα 11 : Τελική αρχιτεκτονική DenseNet-121

3.4 - Αποτελέσματα και Παρατηρήσεις

Στην εφαρμογή και των δύο αρχιτεκτονικών τα 3 τελευταία επίπεδα διατηρήθηκαν παρόμοια. Σκοπός είναι να αναδειχθεί η διαφορετικότητα των μοντέλων με όσο το δυνατόν παραπλήσιες παραμετροποιήσεις. Το dropout είναι μια τεχνική κανονικοποίησης, όπου από το τελικό αποτέλεσμα αφαιρούνται οι πιθανότητες ίσες με την τιμή που δίνεται μαζί με τις συνδέσεις τους. Στόχος είναι η δημιουργία ενός νευρωνικού δικτύου που εξαλείφει το overfitting των δεδομένων. Ο Adam (Adaptive Moment Estimation) είναι ένας αποδοτικός αλγόριθμος βελτιστοποίησης, που προτιμάται σε ανάλογες περιπτώσεις αναγνώρισης εικόνων. Χρησιμοποιεί το learning rate για να ανακαλύπτει και να αποθηκεύει διαφορετικά gradients, με σκοπό την αναγνώριση περίπλοκων μοτίβων και μορφολογιών στις εικόνες. Αναλύεται πλήρως στο [15] με ολοκληρωμένο τον ορισμό και το μαθηματικό του μοντέλο. Η τεχνική της μείωσης του learning rate είναι μια τεχνική που αρχικά αντλεί πληροφορία από ευρύτερη περιοχή, με πιο γενικά μοτίβα εικόνας και έπειτα μεταβαίνει σε πιο περίπλοκα και λεπτομερή στοιχεία. Πιο συγκεκριμένα, όσο το learning rate έχει μεγαλύτερη τιμή, ο αλγόριθμος εκπαιδεύεται με γρήγορο ρυθμό και λαμβάνει πληροφορίες από διάφορα σημεία της εικόνας, αγνοώντας όμως τις λεπτομέρειες. Στη συνέχεια, έχοντας μικρότερη τιμή, η διαδικασία εκπαίδευσης γίνεται με αργό ρυθμό, αντλώντας λεπτομερή πληροφορία και μοτίβα εικόνας ανά επανάληψη εκπαίδευσης (epoch). Οι τιμές των learning rate πρέπει να προσεχθούν ιδιαιτέρως και στις 2 περιπτώσεις. Εάν το learning rate ξεκινήσει από πολύ μεγάλη τιμή, μπορεί να επιτευχθεί μεγάλο ποσοστό επιτυχίας στην εκπαίδευση, όμως στην πράξη οι αναγνωρίσεις θα είναι κυρίως λανθασμένες. Αυτό συμβαίνει ιδιαιτέρως, αν υπάρχουν παραπλήσια μοτίβα σε εικόνες, που όμως διαφέρουν σε λεπτομέρειες και αποτελούν διαφορετικό είδος. Επίσης, το κάτω όριο του learning rate, είναι εξίσου σημαντικό να οριοθετηθεί. Το μικρό learning rate απαιτεί αρκετή υπολογιστική ισχύ, μεγαλύτερο αριθμό epoch και με πιθανότητα η εκμάθηση να παραμείνει στο local minima. Το local minima είναι το μικρότερο σημείο της εσωτερικής συνάρτησης, όπου αν παραμείνει η εκμάθηση είτε απαιτεί αρκετό χρόνο και epochs ή δεν θα επιτευχθεί μεγάλο ποσοστό ευστοχίας. Συνεπώς, στους αλγόριθμους της παρούσας διατριβής χρησιμοποιήθηκε ένας συνδυασμός, ξεκινώντας από ένα μεγαλύτερο learning rate και μειώνεται στο μισό μετά από ένα συγκεκριμένο αριθμό epochs, που δεν υπάρχει αλλαγή στη μετρική validate accuracy.

3.4.1 - Εκπαίδευση και αποτελέσματα

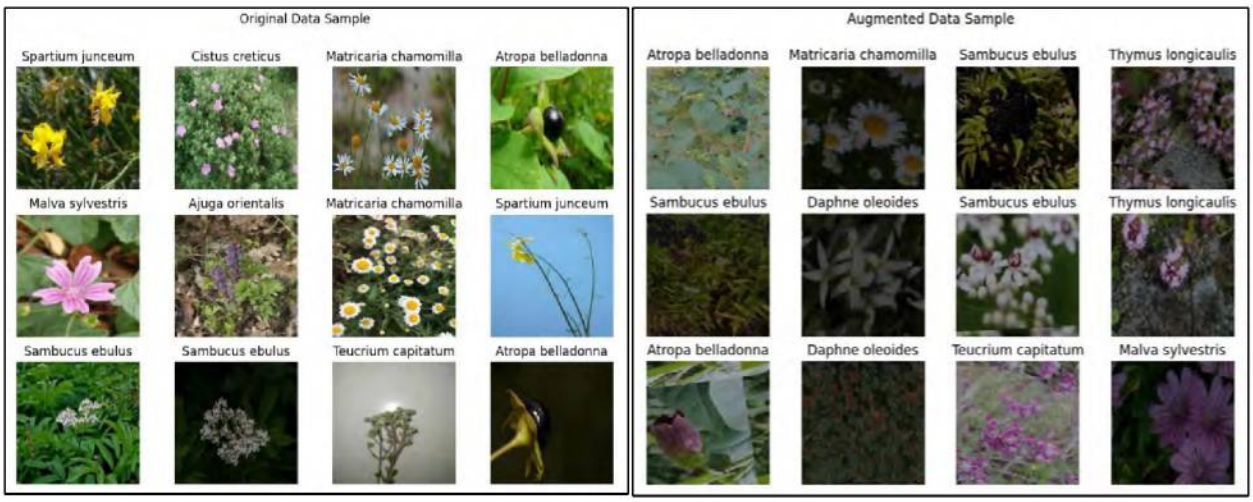
Για την εκπαίδευση των μοντέλων, το σύνολο των δεδομένων μετατράπηκε σε εικόνες μεγέθους 224x224 pixels. Ακόμη, χωρίστηκε σε 2 υποσύνολα, το σύνολο εκπαίδευσης (training set) και το σύνολο ελέγχου και πιστοποίησης της εκπαίδευσης (validation set), κατά ένα ποσοστό 85% και 15%, αντίστοιχα. Ο τελικός αριθμός των set ήταν 15.831 φωτογραφίες για το training set και 2.749 για το validation set. Στη συνέχεια, στο training set χρειάστηκε να γίνει μια

προεπεξεργασία των δεδομένων, ώστε σε κάθε επανάληψη (epoch) ο αλγόριθμος να λαμβάνει όσο το δυνατό περισσότερη και διαφορετική πληροφορία. Η τεχνική του data augmentation είναι αυτή που αποδίδει ακριβώς αυτή τη μετατροπή, και πιο συγκεκριμένα, σε τυχαίο αλλά συγκεκριμένο ποσοστό του συνόλου, εφαρμόστηκε:

- Περιστροφή κατά ένα εύρος γωνιών
- Μεγέθυνση κατά ένα εύρος αριθμού
- Οριζόντια αλλαγή κατεύθυνσης
- Κάθετη αλλαγή κατεύθυνσης
- Αυξομείωση φωτεινότητας κατά ένα εύρος
- Αυξομείωση αντίθεσης κατά ένα εύρος
- Κόψιμο τυχαίου μέρους φωτογραφίας
- Ανακάτεμα φωτογραφιών

Ένα δείγμα φαίνεται στις παρακάτω φωτογραφίες για τους 2 διαφορετικούς αλγορίθμους:

Εικόνα 12 : Δείγμα δεδομένων πριν (αριστερά) και μετά (δεξιά) του data augmentation του DenseNet-121



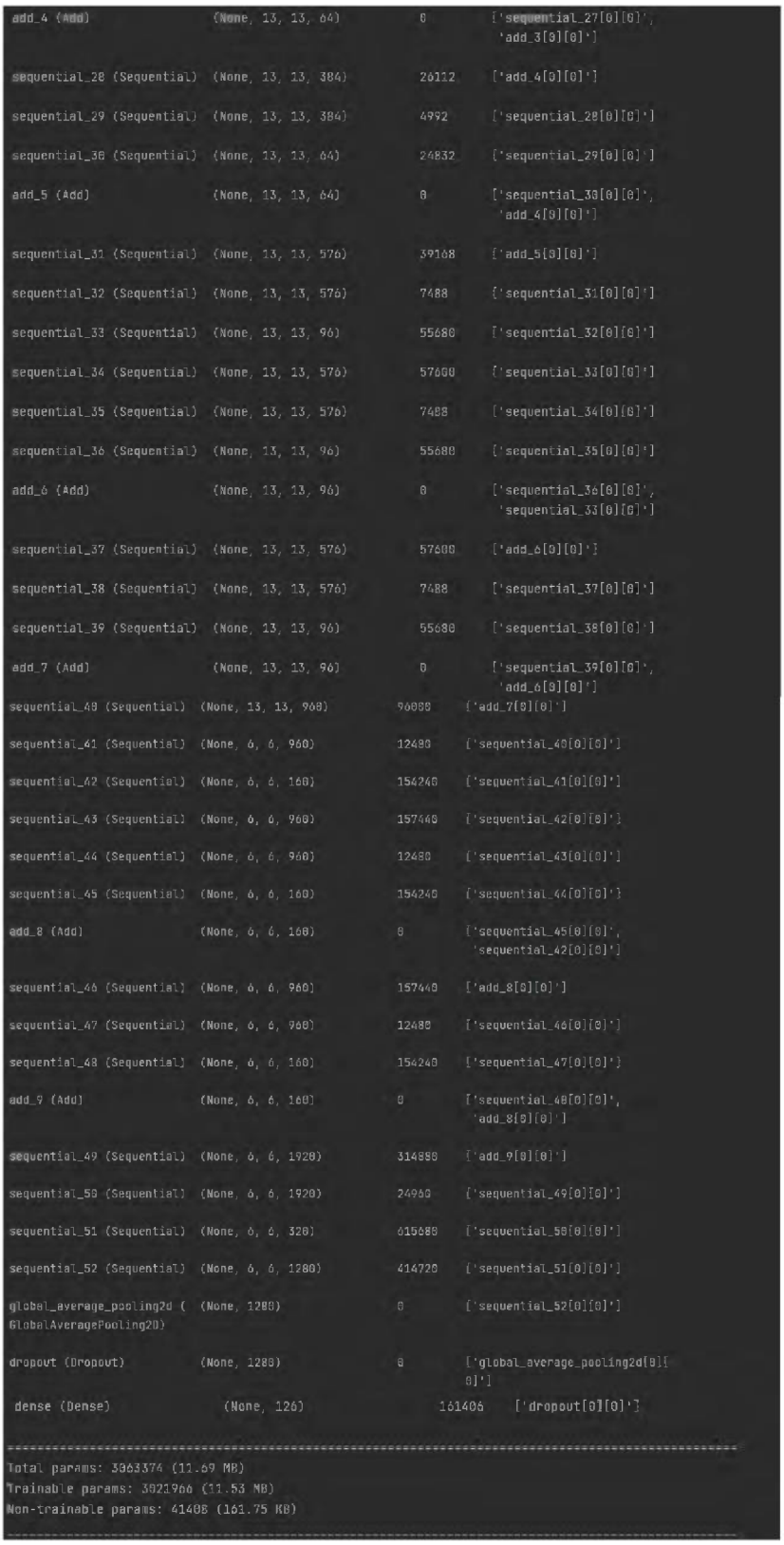
Εικόνα 13 : Δείγμα δεδομένων πριν (αριστερά) και μετά (δεξιά) του data augmentation του MobileNetV2

Στη συνέχεια παρατίθενται οι συνόψεις των μοντέλων όπως παρήχθησαν από τον κώδικα:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 7, 7, 1024)        7037504

 global_average_pooling2d (  (None, 1024)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 512)               524800

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 126)               64638

=================================================================
Total params: 7626942 (29.09 MB)
Trainable params: 7543294 (28.78 MB)
Non-trainable params: 83648 (326.75 KB)
_________________________________________________________________
```

Εικόνα 14 : Πληροφορίες μοντέλου DenseNet-121

```
Model: "mobilenetv2"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0         []

sequential (Sequential)      (None, 111, 111, 32)      992       ['input_1[0][0]']

sequential_1 (Sequential)    (None, 111, 111, 16)      576       ['sequential[0][0]']

sequential_2 (Sequential)    (None, 111, 111, 16)      288       ['sequential_1[0][0]']

sequential_3 (Sequential)    (None, 111, 111, 16)      320       ['sequential_2[0][0]']

sequential_4 (Sequential)    (None, 111, 111, 144)     2880      ['sequential_3[0][0]']

sequential_5 (Sequential)    (None, 55, 55, 144)       1872      ['sequential_4[0][0]']

sequential_6 (Sequential)    (None, 55, 55, 24)        3552      ['sequential_5[0][0]']

sequential_7 (Sequential)    (None, 55, 55, 144)       4032      ['sequential_6[0][0]']

sequential_8 (Sequential)    (None, 55, 55, 144)       1872      ['sequential_7[0][0]']

sequential_9 (Sequential)    (None, 55, 55, 24)        3552      ['sequential_8[0][0]']

add (Add)                    (None, 55, 55, 24)        0         ['sequential_9[0][0]',
                                                                  'sequential_6[0][0]']

sequential_10 (Sequential)   (None, 55, 55, 192)       5376      ['add[0][0]']

sequential_11 (Sequential)   (None, 27, 27, 192)       2496      ['sequential_10[0][0]']

sequential_12 (Sequential)   (None, 27, 27, 32)        6272      ['sequential_11[0][0]']

sequential_13 (Sequential)   (None, 27, 27, 192)       6912      ['sequential_12[0][0]']

sequential_14 (Sequential)   (None, 27, 27, 192)       2496      ['sequential_13[0][0]']

sequential_15 (Sequential)   (None, 27, 27, 32)        6272      ['sequential_14[0][0]']

add_1 (Add)                  (None, 27, 27, 32)        0         ['sequential_15[0][0]',
                                                                  'sequential_12[0][0]']

sequential_16 (Sequential)   (None, 27, 27, 192)       6912      ['add_1[0][0]']

sequential_17 (Sequential)   (None, 27, 27, 192)       2496      ['sequential_16[0][0]']

sequential_18 (Sequential)   (None, 27, 27, 32)        6272      ['sequential_17[0][0]']

add_2 (Add)                  (None, 27, 27, 32)        0         ['sequential_18[0][0]',
                                                                  'add_1[0][0]']

sequential_19 (Sequential)   (None, 27, 27, 384)       13824     ['add_2[0][0]']

sequential_20 (Sequential)   (None, 13, 13, 384)       4992      ['sequential_19[0][0]']

sequential_21 (Sequential)   (None, 13, 13, 64)        24832     ['sequential_20[0][0]']

sequential_22 (Sequential)   (None, 13, 13, 384)       26112     ['sequential_21[0][0]']

sequential_23 (Sequential)   (None, 13, 13, 384)       4992      ['sequential_22[0][0]']

sequential_24 (Sequential)   (None, 13, 13, 64)        24832     ['sequential_23[0][0]']

add_3 (Add)                  (None, 13, 13, 64)        0         ['sequential_24[0][0]',
                                                                  'sequential_21[0][0]']

sequential_25 (Sequential)   (None, 13, 13, 384)       26112     ['add_3[0][0]']

sequential_26 (Sequential)   (None, 13, 13, 384)       4992      ['sequential_25[0][0]']

sequential_27 (Sequential)   (None, 13, 13, 64)        24832     ['sequential_26[0][0]']
```



Εικόνα 15 : Πληροφορίες μοντέλου MobileNetV2

Κατά την εκπαίδευση των δύο αλγορίθμων, υπήρξαν κάποιες διαφοροποιήσεις. Κατ' αρχάς, ο αριθμός των epochs διαφέρει σημαντικά. Στο DenseNet-121 χρειάστηκαν 25 epochs για να επιτευχθεί το ποσοστό ακρίβειας validate accuracy 0.811 ή 81.1%, ενώ στο MobileNetV2 χρειάστηκαν 100 epochs, ώστε να επιτευχθεί το ποσοστό ακρίβειας 0.742 ή 74.2%. Εφ' όσον, ο αριθμός των epochs διέφερε, έτσι και η αρχικοποίηση του learning rate χρειάστηκε να διαφοροποιηθεί. Στο DenseNet-121 ξεκίνησε με την τιμή 0.004, ενώ στο MobileNetV2 ξεκίνησε με τη τιμή 0.016. Κάθε 3 epoch που το validate accuracy δεν βελτιωνόταν, μειωνόταν η τιμή του learning rate στο μισό και για τους 2 αλγόριθμους.

Αυτές οι διαφοροποιήσεις οφείλονται σε δύο παράγοντες. Εν πρώτης, ο DenseNet-121, όπως αναφέρθηκε προηγουμένως, περιέχει βάρη του συνόλου ImageNet [20]. Πρόκειται για βάρη από διάφορα μοντέλα αναγνώρισης εικόνων, όπου μεταφέρεται σημαντική γνώση κατά ποιο τρόπο θα γίνει η εκπαίδευση του μοντέλου. Μια παραμετροποίηση η οποία επιταχύνει αρκετά την ορθή εκπαίδευσή του. Επίσης, όπως μπορεί κανείς να διακρίνει από τον πίνακα 3, τα layers και οι Parameters του DenseNet-121 είναι παραπάνω από διπλάσιοι σε σχέση με του MobileNetV2. Γεγονός που αντιπροσωπεύει ότι σε κάθε epoch γίνεται βαθύτερη εκπαίδευση στο DenseNet-121.

Παρακάτω, παρατίθενται τα στιγμιότυπα των τελευταίων epoch, παρουσιάζοντας το τελικό ποσοστό validate accuracy, όπως επίσης και τις τιμές των training accuracy/loss, validate loss και learning rate.

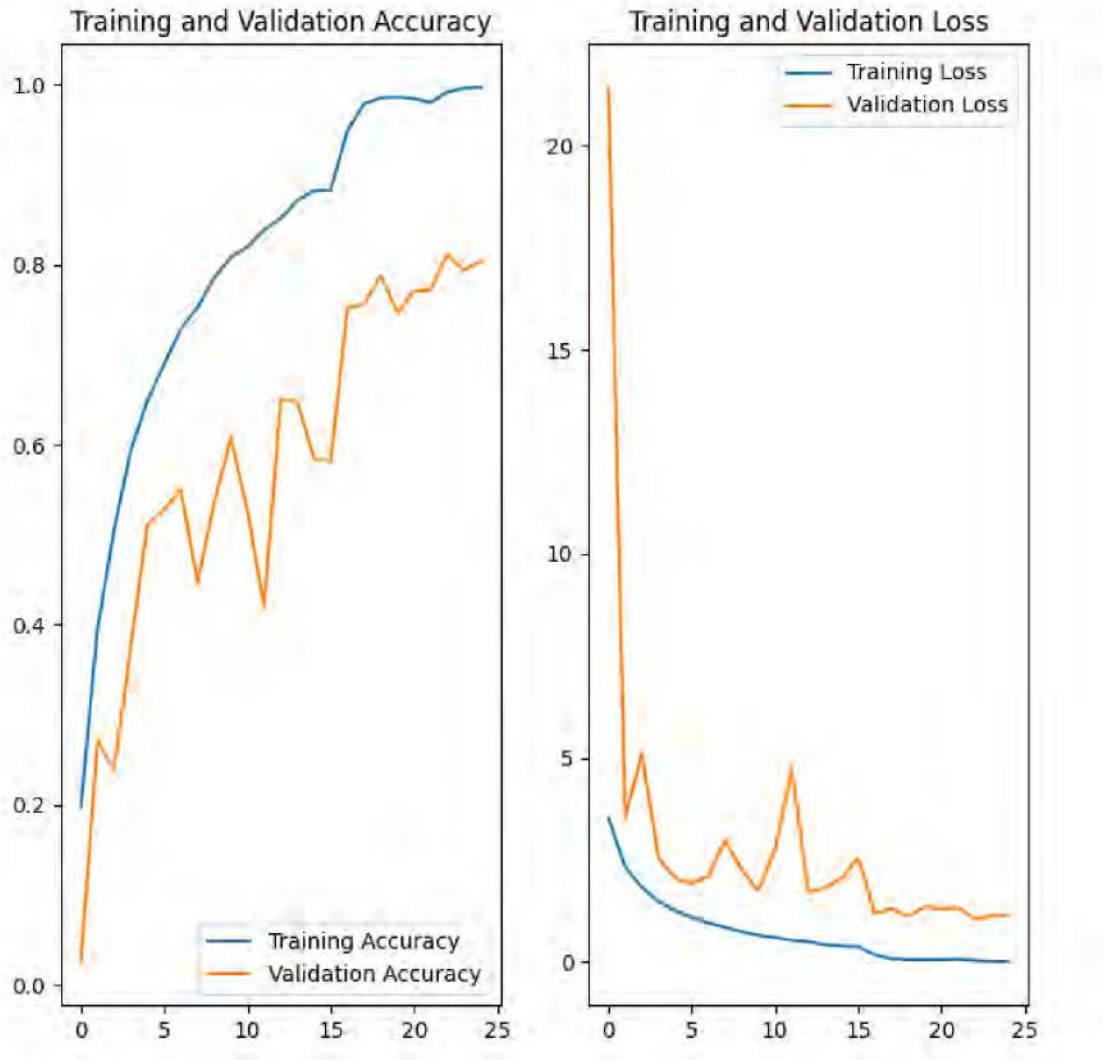
```
Epoch 22/25
165/165 [==============================] - ETA: 0s - loss: 0.0602 - accuracy: 0.9880
Epoch 22: ReduceLROnPlateau reducing learning rate to 0.0009500000237487257.
165/165 [==============================] - 4330s 26s/step - loss: 0.0602 - accuracy: 0.9880 - val_loss: 1.3325 - val_accuracy: 0.7723 - lr: 0.0010
Epoch 23/25
165/165 [==============================] - ETA: 0s - loss: 0.0301 - accuracy: 0.9917

Saving model with the updated val_accuracy = 0.8111940758132935
165/165 [==============================] - 4325s 26s/step - loss: 0.0301 - accuracy: 0.9917 - val_loss: 1.0453 - val_accuracy: 0.8112 - lr: 5.0000e-04
Epoch 24/25
165/165 [==============================] - 4435s 27s/step - loss: 0.0149 - accuracy: 0.9961 - val_loss: 1.1258 - val_accuracy: 0.7937 - lr: 5.0000e-04
Epoch 25/25
165/165 [==============================] - 4613s 28s/step - loss: 0.0115 - accuracy: 0.9971 - val_loss: 1.1392 - val_accuracy: 0.8043 - lr: 5.0000e-04
```

Εικόνα 16 : DenseNet-121 τιμές τελευταίων epoch

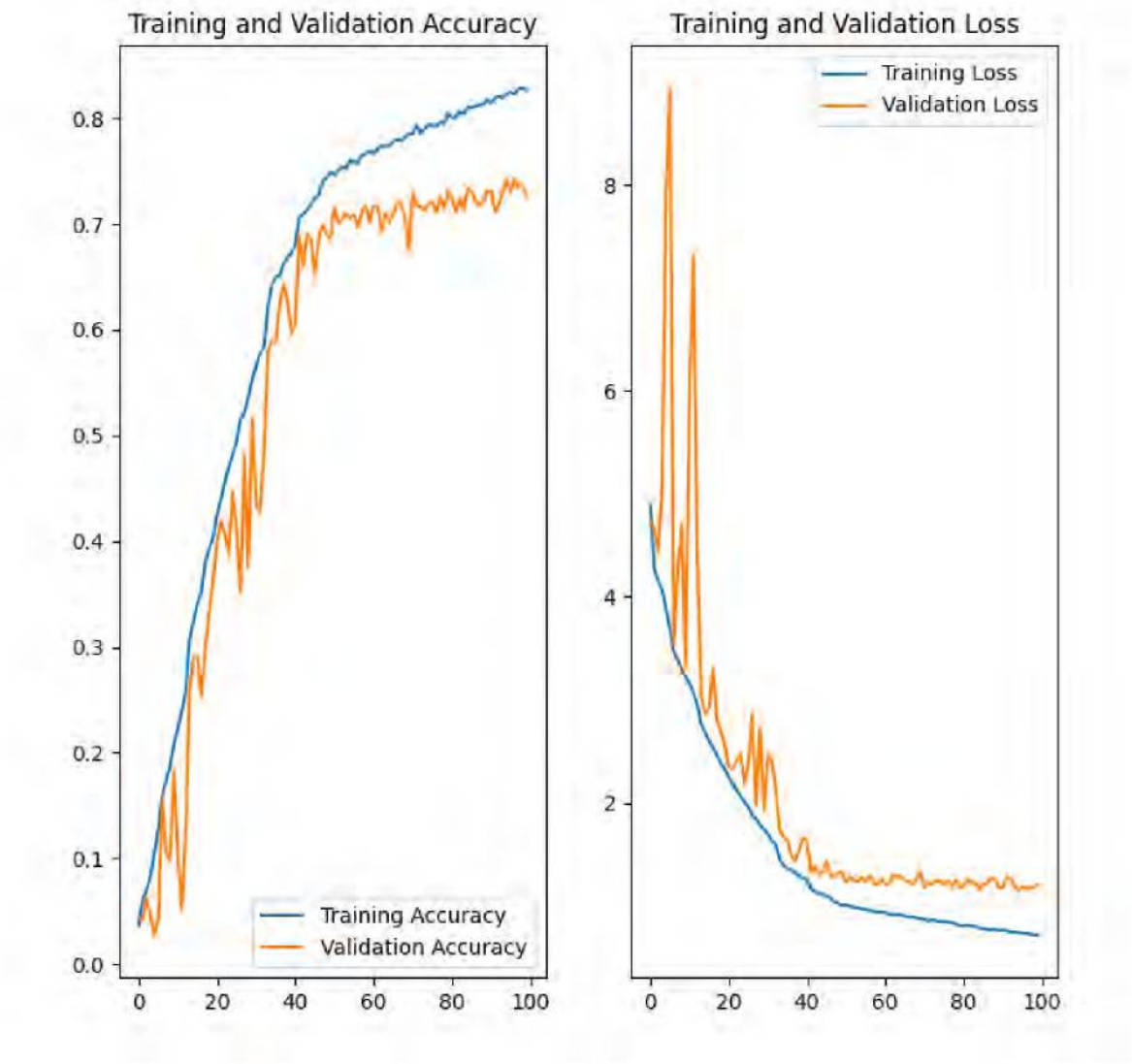
```
Epoch 97: val_acc improved from 0.74209 to 0.74282, saving model to MobilenetV2.h5
248/248 [==============================] - 1271s 5s/step - loss: 0.7311 - acc: 0.8228 - val_loss: 1.1552 - val_acc: 0.7428 - lr: 5.0000e-04
Epoch 98/100
248/248 [==============================] - ETA: 0s - loss: 0.7217 - acc: 0.8282
Epoch 98: val_acc did not improve from 0.74282
248/248 [==============================] - 1272s 5s/step - loss: 0.7217 - acc: 0.8282 - val_loss: 1.1848 - val_acc: 0.7348 - lr: 5.0000e-04
Epoch 99/100
248/248 [==============================] - ETA: 0s - loss: 0.7150 - acc: 0.8291
Epoch 99: val_acc did not improve from 0.74282
248/248 [==============================] - 1256s 5s/step - loss: 0.7150 - acc: 0.8291 - val_loss: 1.1779 - val_acc: 0.7355 - lr: 5.0000e-04
Epoch 100/100
248/248 [==============================] - ETA: 0s - loss: 0.7169 - acc: 0.8267
Epoch 100: val_acc did not improve from 0.74282
248/248 [==============================] - 1261s 5s/step - loss: 0.7169 - acc: 0.8267 - val_loss: 1.2011 - val_acc: 0.7264 - lr: 5.0000e-04
```

Εικόνα 17 : MobileNetV2 τιμές τελευταίων epoch

Ακόμη, από τις γραφικές παραστάσεις συσχέτισης training/validation accuracy και training/validation loss, επιβεβαιώνεται πως υπάρχει ένα μικρό ποσοστό overfitting. Υπάρχει μια απόκλιση μεταξύ των γραφικών training και validation. Αυτό είναι αναμενόμενο, αφού το σύνολο δεδομένων μας δεν είναι ισορροπημένο και υπάρχουν είδη φυτών με χαμηλό αριθμό εικόνων, όπως προαναφέραμε. Δίνονται στη συνέχεια και οι γραφικές παραστάσεις αυτές.

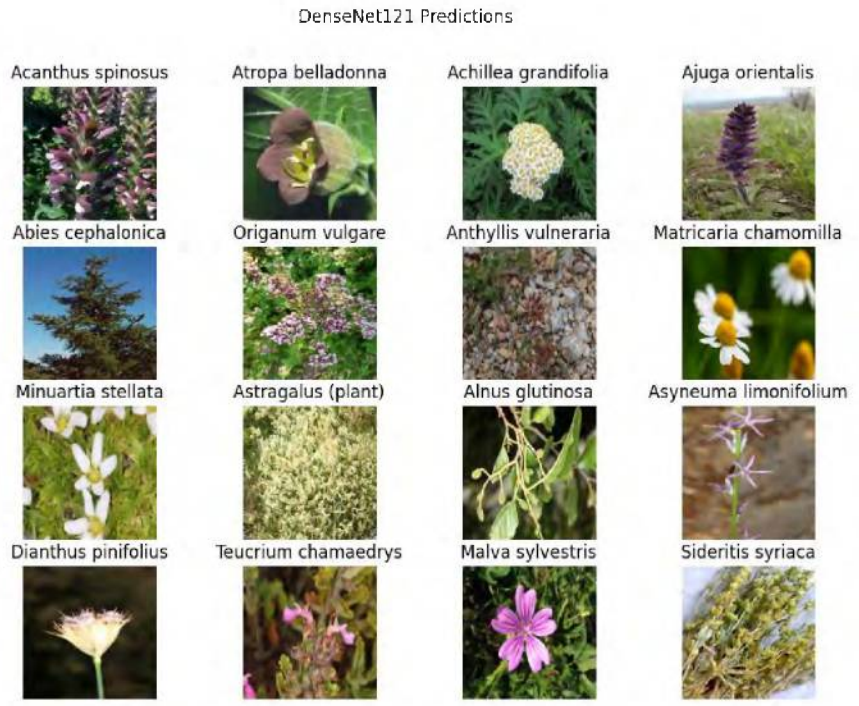


Εικόνα 18 : Γραφικές παραστάσεις DenseNet-121 σύγκρισης training/validation accuracy και loss

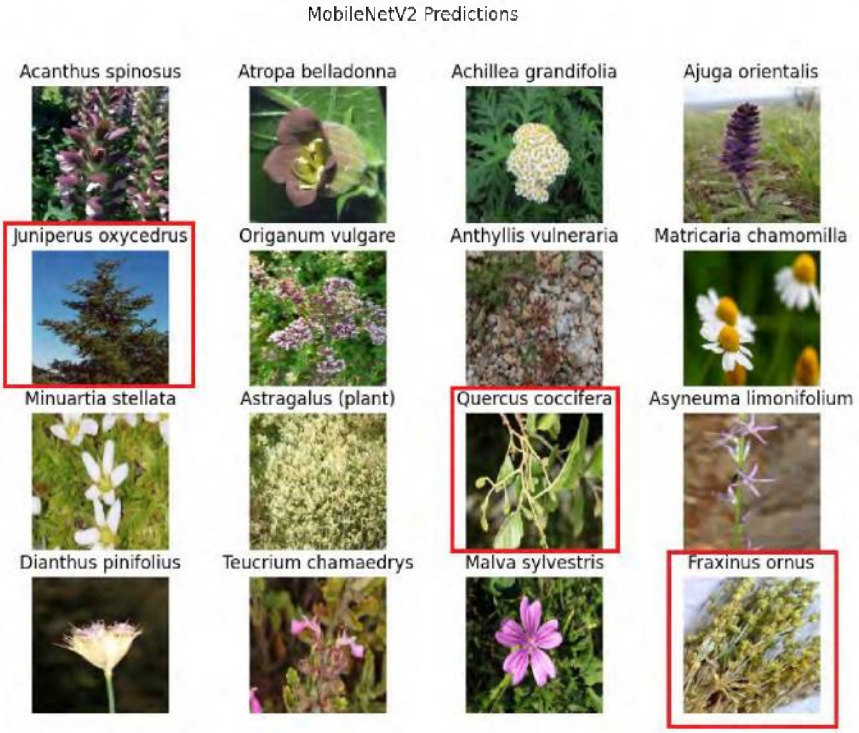


Εικόνα 19 : Γραφικές παραστάσεις MobileNetV2 σύγκρισης training/validation accuracy και loss

Τέλος, για ένα σύνολο εικόνων, διαφορετικό απ' αυτές των 2 υποσυνόλων training και validation, εφαρμόστηκαν οι 2 αλγόριθμοι, με σκοπό να ελεγχθεί η αναγνώριση και τα αποτελέσματα που αποδίδουν. Ο έλεγχος αυτός έγινε στην γλώσσα προγραμματισμού Python, πριν ακόμη γίνει μετατροπή του αρχείου σε tflite για το Android app development. Τα αποτελέσματα διαφέρουν, καθώς το DenseNet-121 παρουσιάζει μεγάλο ποσοστό επιτυχίας, ενώ το MobileNetV2 όχι όσο ήταν αναμενόμενο. Δείγματα των δοκιμών αυτών δίνονται στις επόμενες εικόνες.



Εικόνα 20 : Αναγνωρίσεις DenseNet-121, ποσοστό επιτυχίας 100% στο συγκεκριμένο σύνολο.



Εικόνα 21 : Αναγνωρίσεις MobileNetV2, ποσοστό επιτυχίας 81.25% στο συγκεκριμένο σύνολο

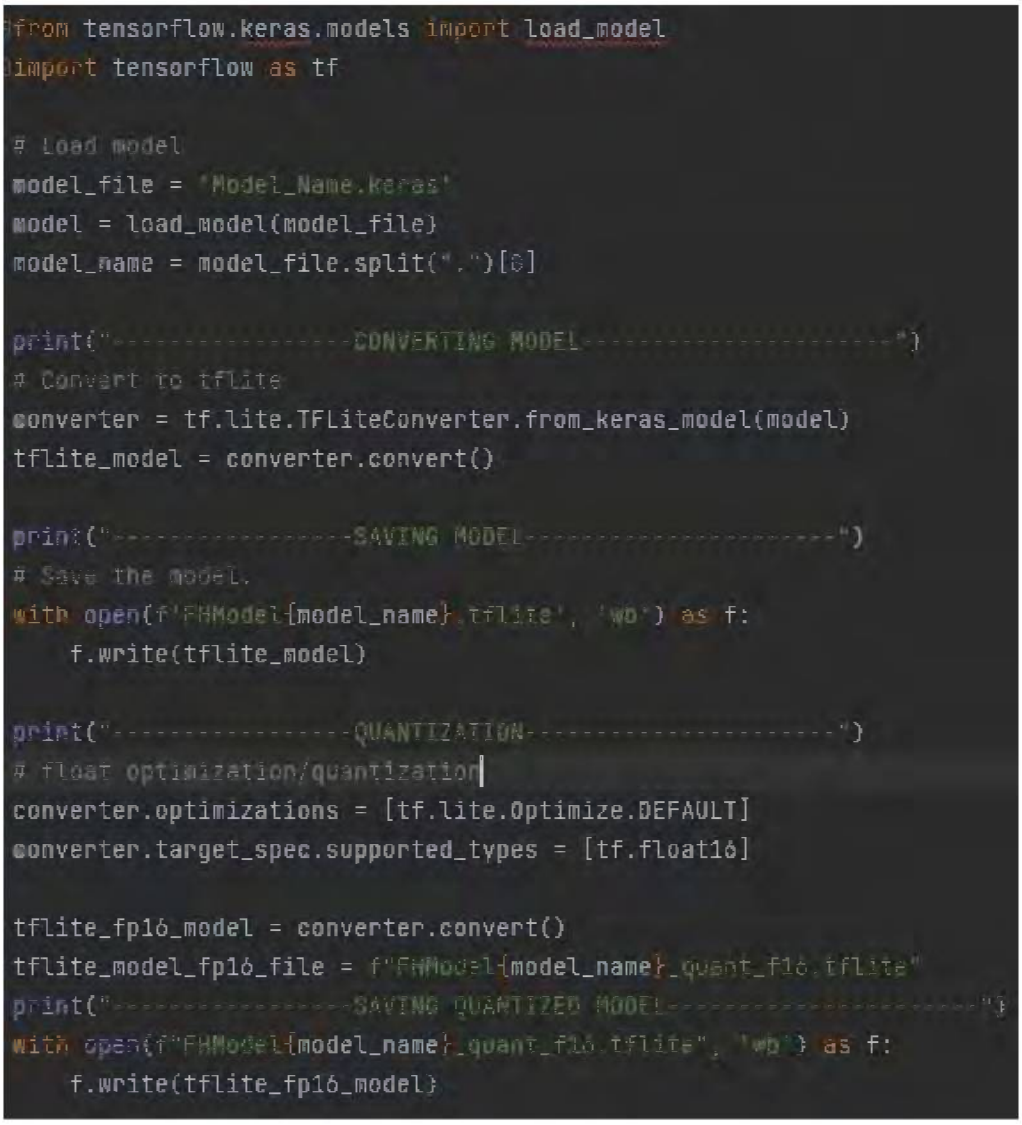
3.4.2 - Μετατροπή σε συμβατό αρχείο για Android app development

Μετά τον έλεγχο αποτελεσμάτων των μοντέλων, απαραίτητη ήταν η μετατροπή τους σε αρχείο συμβατό και αναγνώσιμο από τον κώδικα της Kotlin. Η προσέγγιση που ακολουθήθηκε περιγράφεται στους οδηγούς εκμάθησης της TensorFlow [3], [4] και [5]. Στη συνέχεια θα περιγραφεί η διαδικασία και ο κώδικας που αναπτύχθηκε.

Η μετατροπή ξεκινάει με την ανάγνωση του Keras μοντέλου, που έχει αποθηκευτεί προηγουμένως κατά την εκπαίδευσή του. Μέσω της συνάρτησης convert(), που παρέχεται από τη βιβλιοθήκη TFLiteConverter, γίνεται αυτόματα η μετατροπή.

Πριν συνεχιστεί η διαδικασία, πρέπει να αναφερθεί, πως τα επόμενα βήματα δεν είναι αναγκαία. Το μοντέλο έχει μετατραπεί σε μορφή που αναγνωρίζεται και διαβάζεται από τον κώδικα και τις διαδικασίες του Android app development. Όμως, έρχεται σε αντίθεση με τις σωστές πρακτικές κώδικα που πλέον ακολουθούνται. Αν χρησιμοποιηθεί αυτή η έκδοση του μοντέλου, μεταφέρεται το πρόβλημα της κωδικοποίησης της φωτογραφίας εισόδου, αλλά και της αποκωδικοποίησης της εξόδου του μοντέλου στο κώδικα της Kotlin. Αυτό έχει σαν αποτέλεσμα την αύξηση πολυπλοκότητας στον κώδικα προγραμματισμού της εφαρμογής, αλλά και την κατανάλωση πόρων, αφού πρόκειται για κώδικα που θα τρέξει σε συσκευές android. Συνεπώς, παρόλο που δεν είναι αναγκαία, θεωρούνται πλέον απαραίτητα.

Στη συνέχεια, ένα βήμα που ενδείκνυται για τη βελτιστοποίηση της ταχύτητας της λειτουργίας του μοντέλου, αλλά και για τη διαφοροποίηση του αρχείου εισόδου που αναγνωρίζει, είναι το post-training quantization. Υπάρχουν διάφοροι τρόποι μετατροπής αναλόγως το σκοπό, που αναλύονται στο [4]. Όμως, το quantization που εφαρμόστηκε αφορά τη βελτιστοποίηση χρόνου και την υποστήριξη της μορφής float16, αναγνώρισης δεδομένων. Στη συνέχεια παρατίθεται ο κώδικας μετατροπής:



```python
from tensorflow.keras.models import load_model
import tensorflow as tf

# Load model
model_file = 'Model_Name.keras'
model = load_model(model_file)
model_name = model_file.split(".")[0]

print("-----------------CONVERTING MODEL----------------------")
# Convert to tflite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

print("-----------------SAVING MODEL----------------------")
# Save the model.
with open(f'FHModel{model_name}.tflite', 'wb') as f:
    f.write(tflite_model)

print("-----------------QUANTIZATION----------------------")
# float optimization/quantization
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]

tflite_fp16_model = converter.convert()
tflite_model_fp16_file = f"FHModel{model_name}_quant_f16.tflite"
print("-----------------SAVING QUANTIZED MODEL----------------------")
with open(f"FHModel{model_name}_quant_f16.tflite", 'wb') as f:
    f.write(tflite_fp16_model)
```

Εικόνα 22 : Κώδικας Python μετατροπής μοντέλου Keras σε TensorFlowLite

Μετά το quantization, σειρά έχει η εισαγωγή των metadata στο μοντέλο. Τα metadata είναι δεδομένα που ενσωματώνουν πληροφορίες στο ίδιο το μοντέλο, σχετικές με το είδος: του μοντέλου, της εισόδου και της εξόδου. Εκτός, όμως, από τις πληροφορίες, εισάγουν και τον τρόπο κωδικοποίησης της εισόδου του μοντέλου, αλλά και της αποκωδικοποίησης του αποτελέσματος, ώστε να εμφανιστούν στην οθόνη της συσκευής. Ο κώδικας δίνεται από την ίδια την TensorFlow στο [5], σε σύνδεσμο του Github εδώ, όπου οι μόνες αλλαγές έγκειται στις πληροφορίες μοντέλου: Όνομα, έκδοση, ύψος/πλάτος φωτογραφιών, αριθμό ετικετών κτλ.

Κεφάλαιο 4 - Android Ανάπτυξη και Τεχνολογίες

4.1 - Jetpack compose και η επιλογή του

Η Jetpack Compose είναι μια σύγχρονη βιβλιοθήκη ανάπτυξης User Interface (UI) για το Android app development. Αναπτύχθηκε από την Google, με στόχο να προσφέρει στους προγραμματιστές έναν εντελώς νέο τρόπο δημιουργίας UI στις εφαρμογές τους. Αντιπροσωπεύει μια καινοτόμο προσέγγιση στον τομέα της ανάπτυξης εφαρμογών Android και χρησιμοποιείται με τη γλώσσα προγραμματισμού Kotlin. Επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές με διαδικασίες πιο σύγχρονες, πιο ευέλικτες και με εύκολο τρόπο ελέγχου και συντήρησης. Μερικά από τα πιο σημαντικά οφέλη της είναι:

1. *Δυναμική και Σαφής Δημιουργία UI:* επιτρέπει στους προγραμματιστές να κατασκευάζουν UI με πιο δυναμικό τρόπο, με χρήση κώδικα Kotlin αντί για XML, προσφέροντας περισσότερη ευελιξία και αμεσότητα στη διαχείριση των στοιχείων του UI.
2. *Στατική Ανάλυση και Πρόληψη Σφαλμάτων:* χρησιμοποιεί την τεχνολογία του Kotlin Compiler για στατική ανάλυση κώδικα του UI, εντοπίζοντας σφάλματα κατά τη συγγραφή κώδικα και βελτιώνοντας την ποιότητα της εφαρμογής.
3. *Υψηλή Απόδοση και Αποδοτικότητα:* παρέχει υψηλή απόδοση και αποδοτικότητα ακόμη και σε δημιουργία σύνθετου UI.
4. *Εύκολη Ανταλλαγή Κώδικα:* Η Compose βασίζεται στη γλώσσα Kotlin, προσφέροντας σαφή και σύγχρονη σύνταξη, γεγονός που δίνει τη δυνατότητα εύκολης ανταλλαγής κώδικα μεταξύ των προγραμματιστών.
5. *Ενσωμάτωση με Άλλα Τμήματα του Android:* Η compose ενσωματώνεται εύκολα με άλλα τμήματα της Jetpack και τις υπηρεσίες του Android.
6. *Υποστήριξη Localization και Accessibility:* προσφέρει ενσωματωμένη υποστήριξη για την υλοποίηση διεθνοποίησης (localization) και προσβασιμότητας (accessibility).
7. *Ενσωματωμένα Animation και Motion:* επιτρέπει τη δημιουργία εντυπωσιακών animation και κινήσεων (motion) στο UI.
8. *Συνεχής Βελτίωση και Υποστήριξη:* Η Google συνεχίζει να βελτιώνει και να αναπτύσσει την Jetpack Compose, προσφέροντας συνεχή υποστήριξη και αναβαθμίσεις [1, 2].

Η υποστήριξη της κοινότητας των προγραμματιστών και της Google, η αποδοτικότητα κώδικα και η συνολική καινοτομία, καθιστούν την τεχνολογία Compose κύριο μέσο ανάπτυξης εφαρμογών. Αυτοί είναι και οι λόγοι που οδήγησαν στην επιλογή υλοποίησης της Herb Tracker με αυτή τη τεχνολογία.

4.2 - Αρχιτεκτονική της Herb Tracker

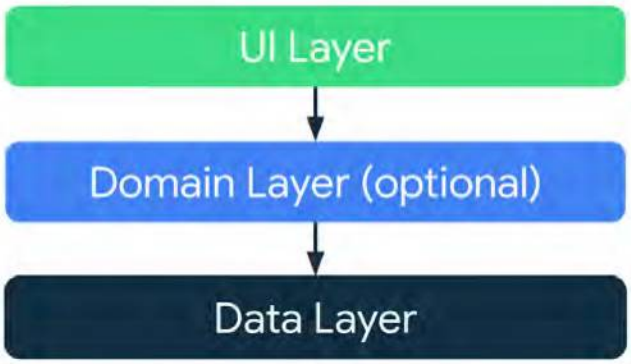
Σύμφωνα με τα [1] και [2], που πρόκειται για μαθήματα εκπαίδευσης προγραμματιστών της Google, μεταξύ άλλων, δίνεται αρκετή έμφαση στην αρχιτεκτονική εφαρμογών. Συνήθως, οι εφαρμογές Android αποτελούνται από activities, fragments, services, content providers και broadcast receivers. Όλα αυτά τα στοιχεία δηλώνονται στο αρχείο manifest της κάθε εφαρμογής, το οποίο χρησιμοποιείται αργότερα από το λειτουργικό Android για την μεταφορά κι εγκατάσταση όλων των απαραίτητων στοιχείων. Η ροή λειτουργίας των εφαρμογών, μιας συσκευής, εναλλάσσεται σύμφωνα με τις προτιμήσεις του χρήστη, συνεπώς η ενεργοποίηση/απενεργοποίηση τους είναι τυχαία. Όμως, οι πόροι της Android συσκευής είναι περιορισμένοι. Έτσι, το λειτουργικό θέτει τις εφαρμογές σε βαθμίδες προτεραιότητας και είναι πολύ πιθανόν κάποιες, που έχουν πολύ χαμηλή προτεραιότητα, να τις τερματίσει. Συνεπώς, σε κάθε εφαρμογή, καθίσταται απαραίτητη μια αρχιτεκτονική που διαχειρίζεται τέτοιες περιπτώσεις. Ακολουθώντας τις σωστές πρακτικές σύνταξης κώδικα, θα πρέπει να είναι εν δυνάμει επεκτάσιμη, να υπάρχει δυνατότητα ελέγχου του ίδιου του κώδικα και να αποτρέπει την αποθήκευση δεδομένων σε «λάθος» σημεία της εφαρμογής.

Οι αρχές μιας αρχιτεκτονικής που περιέχει όλα όσα αναφέρθηκαν έχουν ως εξής:

1. *Διαχωρισμός των προβλημάτων*: Ελαχιστοποίηση των εξαρτήσεων και της σύνταξης κώδικα στις κύριες κλάσεις λειτουργίας: Activity ή/και Fragment. Η χρήση των κλάσεων αυτών θα περιορίζεται στην ένωση του λειτουργικού και της εφαρμογής.
2. *Τροφοδότηση του UI από μοντέλα δεδομένων*: Δημιουργία σταθερών μοντέλων δεδομένων, ξεχωριστά και ανεξάρτητα από τον κώδικα των στοιχείων του UI ή άλλων δομών. Πρόκειται να τροφοδοτούν το UI, όπου είναι απαραίτητο κατά τη χρήση.
3. *Single source cf truth (SSOT)*: Διαδικασία κατά την οποία ένας τύπος δεδομένων δύναται να μετατραπεί και να ενημερωθεί μόνο από ένα συγκεκριμένο σύστημα ιδιωτικής ανάθεσης. Με το SSOT, ορίζεται συγκεκριμένος τύπος δεδομένων και συναρτήσεις ανάθεσης ή/και παραλλαγής δεδομένων. Έτσι, αποφεύγονται παραβιάσεις τύπων δεδομένων και γίνεται πιο εύκολη η διαδικασία εύρεσης λάθους στον κώδικα.
4. *Μοτίβο Unidirectional Data Flow (UDF)*: Η ροή των δεδομένων της εφαρμογής ξεκινάει από τους τύπους υψηλότερου σκοπού (SSOT, βάση δεδομένων κτλ.) και καταλήγει στους τύπους χαμηλότερου σκοπού (στοιχεία UI). Αντίθετα, το κάθε συμβάν εναλλαγής κατάστασης των δεδομένων, που κυρίως προέρχεται από το χρήστη, ξεκινάει από την εναλλαγή που έγινε στο UI και καταλήγει στην κατάλληλη ενημέρωση των SSOT.

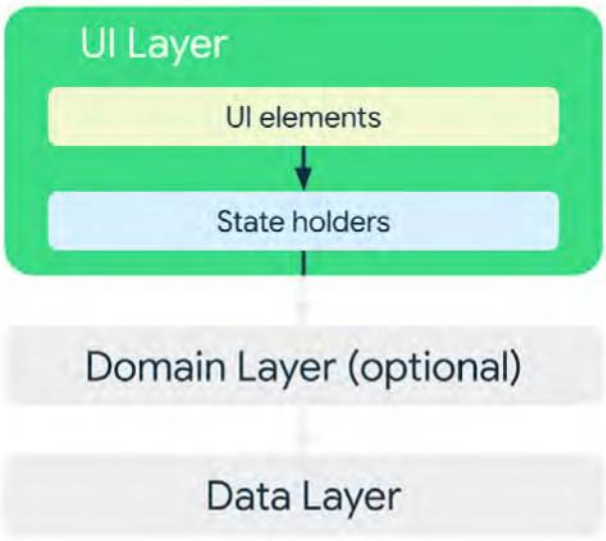
Η υλοποίηση της προτεινόμενης αρχιτεκτονικής αποτελείται κυρίως, από 2 επίπεδα. Το UI επίπεδο, που είναι υπεύθυνο για τον τρόπο εμφάνισης των δεδομένων και το επίπεδο δεδομένων (data), που περιέχει την επιχειρηματική λογική διαχείρισης, μετατροπής και μεταφοράς των δεδομένων. Υπάρχει κι ένα επιπρόσθετο επίπεδο, που συνήθως εφαρμόζεται σε

πολύπλοκες ή/και μεγάλου μεγέθους εφαρμογές, το οποίο ονομάζεται επίπεδο τομέα (domain). Πρόκειται για ένα ενδιάμεσο επίπεδο των προηγούμενων δυο, με σκοπό την απλοποίηση του κώδικα των άλλων δυο επιπέδων, αλλά και τη δημιουργία κώδικα που μπορεί να επαναχρησιμοποιηθεί.



Εικόνα 23 : Προτεινόμενο μοντέλο γενικής αρχιτεκτονικής [1]

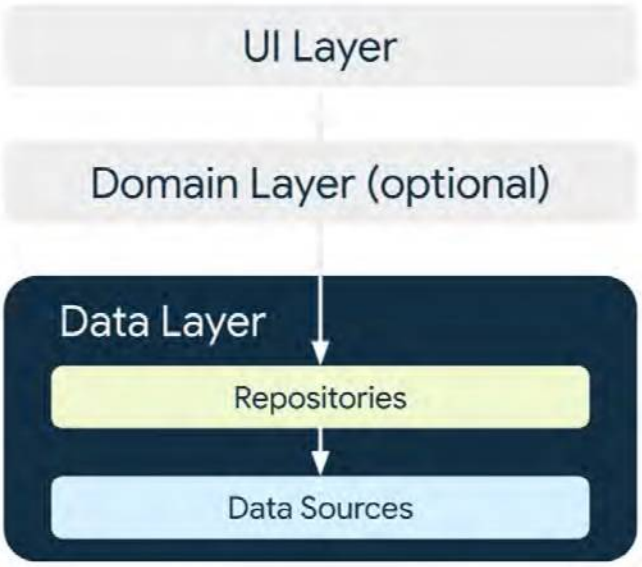
Εμβαθύνοντας λίγο, σε αυτή τη μοντέρνα αρχιτεκτονική εφαρμογών, το Unidirectional Data Flow (UDF) εφαρμόζεται σε όλα τα επίπεδα μιας εφαρμογής. Το UI επίπεδο είναι υπεύθυνο για την ανανέωση των στοιχείων της οθόνης σύμφωνα με τη διαδραστικότητα του χρήστη ή εξωτερικών παραγόντων. Αυτό επιτυγχάνεται με τη χρήση μηχανισμών καλής πρακτικής, όπως οι: state holders, coroutines, flows και dependency injection. Τα διάφορα UI στοιχεία είναι αυτά που αποδίδουν με γραφικό τρόπο τα δεδομένα, διατηρούν τις τιμές των δεδομένων και διαχειρίζονται όλη τη λογική των γραφικών.



Εικόνα 24 : Ο ρόλος του επιπέδου UI στην αρχιτεκτονική [1]

Το data επίπεδο είναι αυτό που διατηρεί την επιχειρηματική λογική, η οποία είναι ένα σύνολο κανόνων που ορίζουν πως η εφαρμογή δημιουργεί, αποθηκεύει και μετατρέπει τα δεδομένα.

Για τη διαχείριση αυτή είναι υπεύθυνα τα repositories, τα οποία περιέχουν από καμία έως αρκετές πηγές διαφορετικών τύπων δεδομένων. Η κύρια εργασία των repositories είναι η διανομή των δεδομένων στην εφαρμογή, συγκεντρώνοντας τις μετατροπές, λύνοντας τυχόν συγκρούσεις και αφαιρώντας άλλες πηγές δεδομένων που μπορεί να έχουν δημιουργηθεί στην εφαρμογή.



Εικόνα 25 : Ο ρόλος του επιπέδου Δεδομένων (Data) στην αρχιτεκτονική [1]

Όπως προαναφέρθηκε, το επίπεδο domain είναι προαιρετικό και πρόκειται για ένα ενδιάμεσο επίπεδο που διατηρεί πολύπλοκο ή/και επαναχρησιμοποιούμενο κώδικα. Δημιουργεί εξαρτήσεις στο UI επίπεδο, ενώ ταυτόχρονα στηρίζεται στο data επίπεδο με τις επονομαζόμενες κλάσεις cases ή interactors, οι οποίες η κάθε μία είναι υπεύθυνη για ξεχωριστή λειτουργικότητα.

Ολοκληρώνοντας με την επεξήγηση της αρχιτεκτονικής, απομένουν οι εξαρτήσεις (dependencies) του domain επιπέδου. Πρόκειται για τη διαχείριση αλληλεξαρτήσεων μεταξύ των κλάσεων, το οποίο επιτυγχάνεται με δύο διαφορετικά μοτίβα: το Dependency injection (DI) και το Service locator. Το πρώτο πρόκειται για την άμεση δήλωση των dependencies χωρίς αυτά να έχουν ήδη υπάρχουσα δομή, ενώ το δεύτερο περιέχει πλήρη δόμηση για το τρόπο απόκτησής τους. Τα μοτίβα αυτά αποδίδουν στο κώδικα επεκτασιμότητα, αποφυγή διπλοτυπίας και διευκολύνουν τους προγραμματιστές τόσο στη παραγωγή του κώδικα, όσο και στον έλεγχό του.

4.3 – Κωδικοποίηση εισόδου κι αποκωδικοποίηση εξόδου του μοντέλου

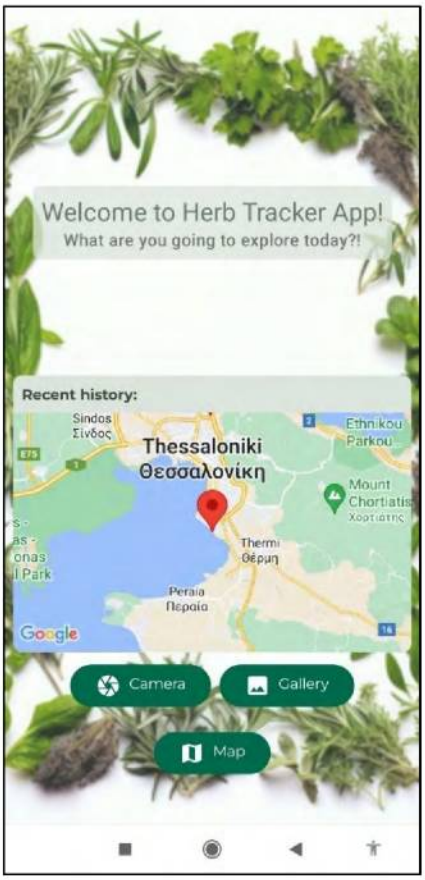
Στην παρακάτω εικόνα παρατίθεται ο κώδικας που υλοποιεί όλες τις απαραίτητες ενέργειες για την κωδικοποίηση της εισόδου και την αποκωδικοποίηση της εξόδου του μοντέλου. Ο κώδικας δίνεται ώστε να υπάρχει δυνατότητα αναπαραγωγής από τον αναγνώστη. Η φωτογραφία που γίνεται λήψη από τη κάμερα ή η εικόνα που εισάγεται από την συλλογή φωτογραφιών, μεταφράζεται ως ένα Bitmap στο κώδικα. Το Bitmap είναι ένα σύνολο pixels με στοιχεία πλάτους, μήκους και είδους κωδικοποίησης χρώματος. Έπειτα, για να αποδώσουν τα μοντέλα ορθά αποτελέσματα, είναι απαραίτητη η μετατροπή αυτού του Bitmap. Η μετατροπή αυτή γίνεται σύμφωνα με το τρόπο που επεξεργάστηκαν και δόθηκαν τα αρχεία, στο κάθε μοντέλο, κατά την εκπαίδευση του, αλλά και του τρόπου που έγινε quantized. Στην προκειμένη περίπτωση πρόκειται για αρχεία μεγέθους 224x224 pixels, κωδικοποίηση χρώματος ARGB_8888 και τύπος ανάγνωσης δεδομένων float. Ακόμη, η χρήση της TensorImage και του TensorBuffer είναι οι διαδικασίες που προσφέρει η TensorFlow για τη διευκόλυνση λειτουργίας του μοντέλου tflite. Τέλος, τα μοντέλα επιστρέφουν σαν αποτέλεσμα τις 5 καλύτερες πιθανότητες αναγνώρισης, με τις αντίστοιχες ετικέτες, αλλά στο συγκεκριμένο κώδικα λαμβάνεται μόνο η καλύτερη πιθανότητα και ετικέτα.

```
fun modelInferenceDenseNet121(uiState: PhotoItemUiState, model: FhmodeldensenetQuantF16, viewModel: HomeViewModel) {

    // Creates a processor for preprocessing the input image for the model
    val imageProcessor = ImageProcessor.Builder()
        .add(ResizeOp(IMAGE_SIZE, IMAGE_SIZE, ResizeOp.ResizeMethod.BILINEAR))
        .build()

    // convert the image to appropriate bitmap for model
    val properBitmap = uiState.bitmap?.copy(Bitmap.Config.ARGB_8888, true)

    // loads the float array of the image and preprocess it
    var tfImage = TensorImage(DataType.FLOAT32)
    tfImage.load(properBitmap)
    tfImage = imageProcessor.process(tfImage)

    // Runs model inference and gets result sorted descending by probability
    val outputs = model.process(tfImage.tensorBuffer).probabilityAsCategoryList.apply {
        sortByDescending { it.score }
    }

    // update the state depending the result
    var label = outputs[0].label
    if (outputs[0].score.toDouble() < ACCEPTABLE_PROBABILITY_LIMIT) {
        label = ""
    }
    viewModel.updateNameAndProbability(label, outputs[0].score.toDouble())

    if (uiState.canNavigateToMap) {
        viewModel.updateCanNavigateToMap(outputs[0].score.toDouble() > ACCEPTABLE_PROBABILITY_LIMIT)
    }
    model.close()
}
```

Εικόνα 26 : Κώδικας εύρυθμης λειτουργίας μοντέλου μηχανικής μάθησης της Herb Tracker

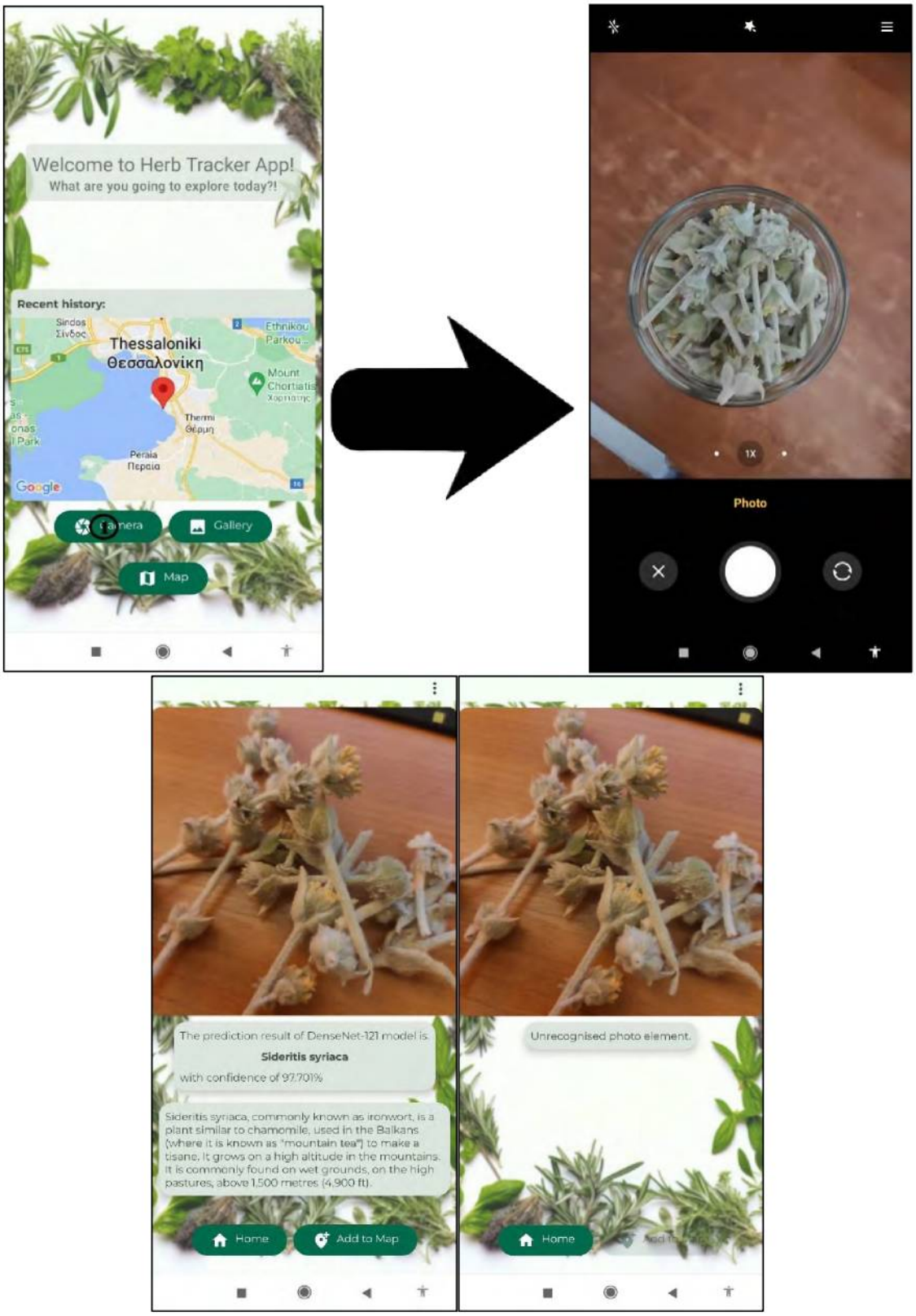
4.4 - Οθόνες εφαρμογής

Συνοψίζοντας, μένει να παρουσιαστεί η εφαρμογή Herb Tracker. Η εμπειρία του χρήστη ξεκινάει από την αρχική οθόνη. Σε αυτή περιέχεται ένα μήνυμα καλωσορίσματος, το ιστορικό των τελευταίων λήψεων φωτογραφίας και τα 3 κουμπιά περιήγησης με τις ανάλογες λειτουργικότητες: Camera, Gallery και Map. Αυτή, παρουσιάζεται στην ακόλουθη εικόνα:



Εικόνα 27 : Αρχική οθόνη Herb Tracker

Προχωρώντας, αναλύεται η κάθε λειτουργικότητα ξεχωριστά. Στην επιλογή Camera, εκκινείται η λειτουργικότητα λήψης φωτογραφίας της συσκευής, όπου ο χρήστης λαμβάνει την επιθυμητή φωτογραφία. Μετά την επιβεβαίωση λήψης, μεταβαίνει στην οθόνη που εμφανίζεται η αναγνώριση της εικόνας από το μοντέλο, μαζί με τις σχετικές πληροφορίες για το είδος βοτάνου ή φυτού:

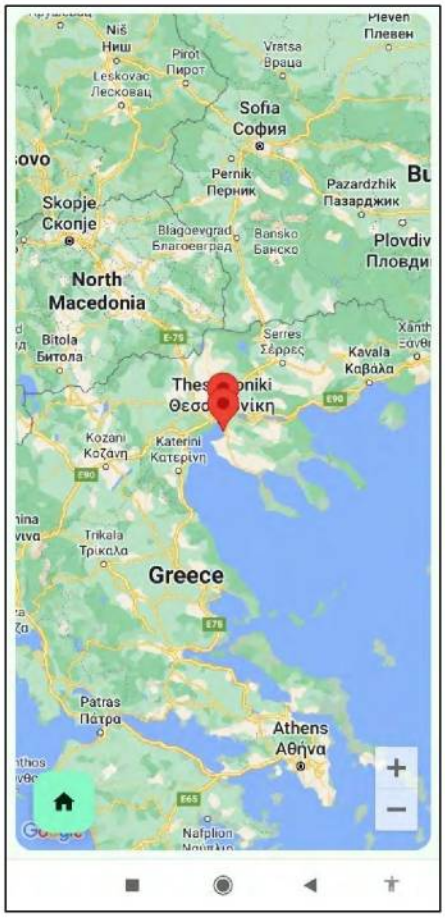


Εικόνα 28 : Λειτουργικότητα κάμερας και αναγνώρισης DenseNet-121 (αριστερά), MobileNetV2 (δεξιά)

Στις προηγούμενες φωτογραφίες, μπορεί κάποιος να παρατηρήσει ότι υπάρχουν 2 διαφορετικές εικόνες σαν αποτέλεσμα αναγνώρισης. Πρόκειται για το διαφορετικό αποτέλεσμα που δίνουν τα 2 μοντέλα. Αυτό μπορεί να εναλλαχθεί δυναμικά από το αναδυόμενο μενού, στο πάνω δεξιά μέρος της οθόνης, το οποίο απεικονίζεται με τρείς συνεχόμενες τελείες σε κάθετη διάταξη. Το αναδυόμενο μενού εμφανίζεται στην ακόλουθη φωτογραφία:

Εικόνα 29 : Μενού επιλογής μοντέλου αναγνώρισης εικόνων

Έπειτα, ο χρήστης έχει τη δυνατότητα είτε να επιστρέψει στην αρχική οθόνη με το κουμπί 'Home' ή να αποθηκεύσει την τοποθεσία του στο χάρτη και στη βάση δεδομένων της εφαρμογής, πιέζοντας το κουμπί 'Add to Map'. Η πρόσθεση τοποθεσίας είναι διαθέσιμη σαν επιλογή, μόνο εάν έχει γίνει επιτυχής αναγνώριση. Στην επιλογή 'Add to Map', η εφαρμογή αυτόματα ενημερώνεται για τη τοποθεσία της συσκευής μέσω του GPS, τοποθετεί την αντίστοιχη πινέζα με το όνομα του είδους που αναγνωρίσθηκε και μεταφέρει τον χρήστη στην οθόνη του χάρτη.



Εικόνα 30 : Οθόνη χάρτη με αποθηκευμένα σημεία λήψης φωτογραφίας

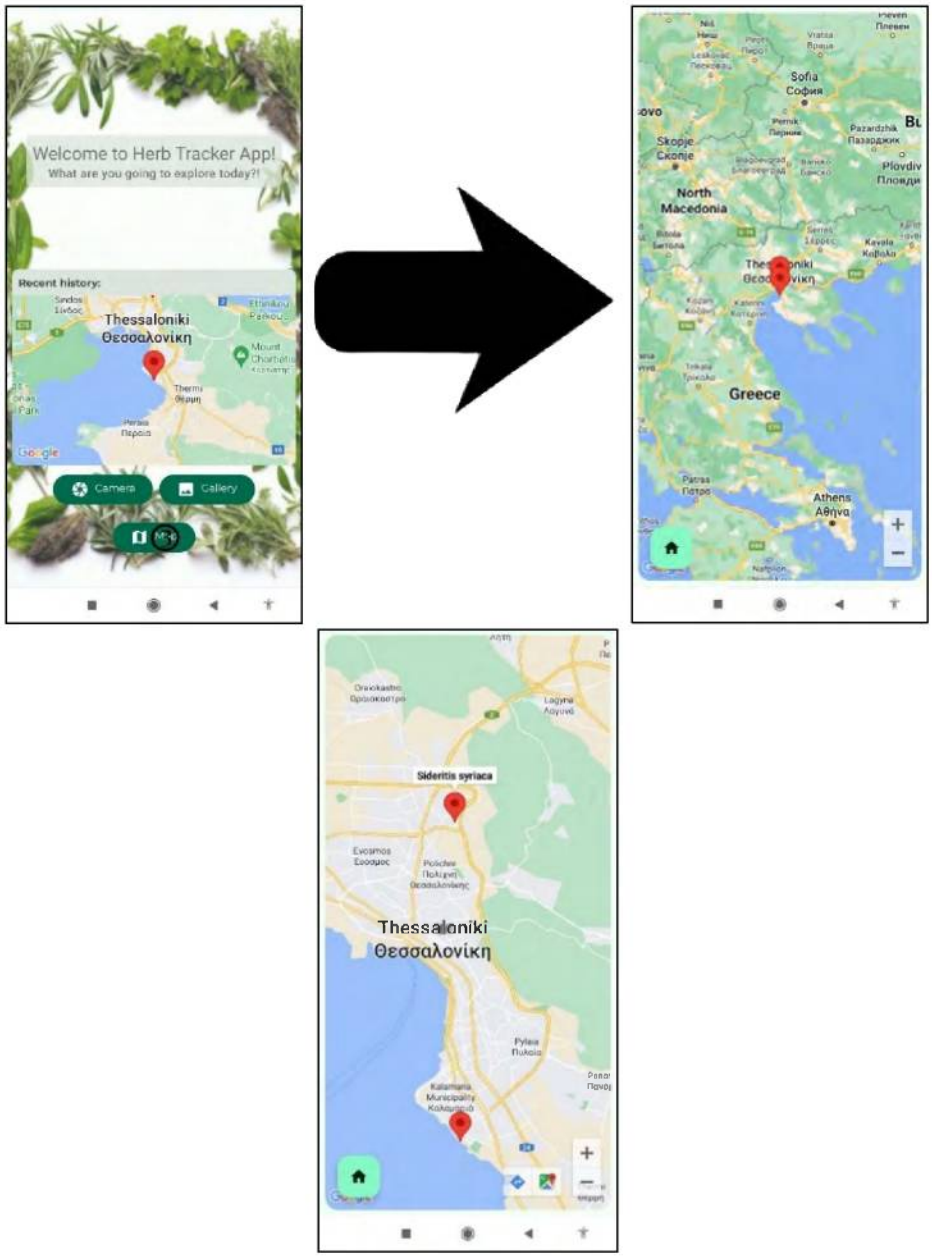
Στην οθόνη αυτή ο χρήστης έχει τη δυνατότητα να περιηγηθεί στο χάρτη κάνοντας μεγέθυνση/σμίκρυνση ή να κινηθεί προς οποιαδήποτε κατεύθυνση ώστε να δει τα σημεία που έχει αποθηκεύσει κατά καιρούς.

Εκτός, όμως, της λειτουργίας λήψης φωτογραφίας, υπάρχει και η δυνατότητα αναγνώρισης εικόνας που είναι αποθηκευμένη ήδη στη συλλογή της συσκευής του χρήστη. Όπως και προηγουμένως, τώρα από την αρχική οθόνη επιλέγεται το κουμπί ‘Gallery’, μεταβαίνει στην οθόνη επιλογής φωτογραφίας από τη συλλογή κι έπειτα γίνεται και πάλι η αναγνώριση εικόνας.

Εικόνα 31 : Λειτουργικότητα συλλογής και αναγνώρισης DenseNet-121 (αριστερά), MobileNetV2 (δεξιά)

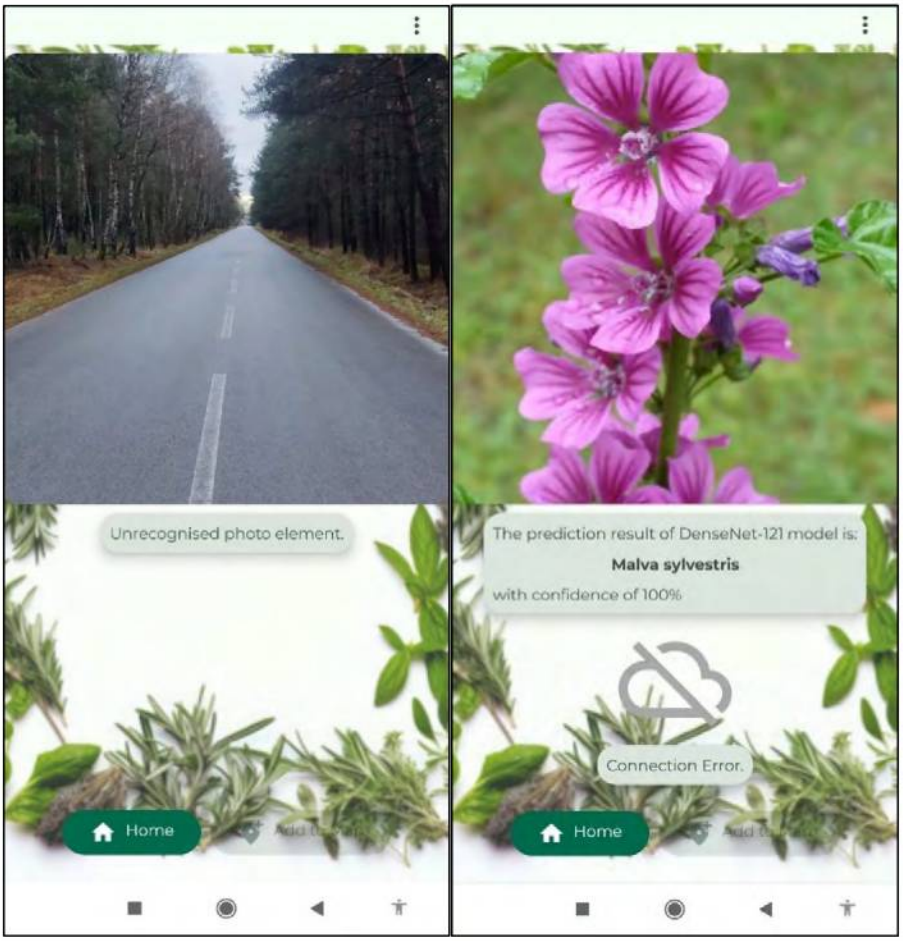
Η οθόνη όπου καταλήγει ο χρήστης, όπως και πριν, είναι παρόμοια, με τη διαφορά ότι μέσω αυτής της λειτουργικότητας δεν δίνεται η επιλογή αποθήκευσης της τοποθεσίας. Λειτουργικότητα η οποία είναι λογική και επιθυμητή, αφού δεν υπάρχει δυνατότητα να αναγνωρισθεί που λήφθηκε η εικόνα που βρίσκεται στη συλλογή του χρήστη.

Σαν τελευταία λειτουργικότητα από την αρχική οθόνη, είναι αυτή της επιλογής του 'Map', η οποία μεταφέρει το χρήστη στο χάρτη με όλες τις αποθηκευμένες τοποθεσίες.



Εικόνα 32 : Λειτουργικότητα Map

Εκτός, όμως, από την εύρυθμη λειτουργία της εφαρμογής, υπάρχει και η πιθανότητα, είτε να μην αναγνωρίσει κάποια φωτογραφία ή η συσκευή να βρίσκεται εκτός δικτύου. Στην περίπτωση που δεν γίνεται αναγνώριση εικόνας, η εφαρμογή επιστρέφει το αποτέλεσμα «Unrecognised photo element», χωρίς να αναγράφεται κάτι στις λεπτομέρειες. Στην έλλειψη σύνδεσης στο διαδίκτυο, η αναγνώριση της φωτογραφίας από τα μοντέλα λειτουργεί κανονικά επιστρέφοντας πιθανή αναγνώριση. Όμως, δε δίνεται η δυνατότητα λήψης πληροφοριών για το αναγνωρισθέν βότανο ή ευεργετικό φυτό, καθώς οι πληροφορίες λαμβάνονται δυναμικά με μια κλήση σε API endpoint του Wikipedia.



Περίπτωση μη αναγνώρισης φωτογραφίας| Περίπτωση αναγνώρισης χωρίς σύνδεση | δικτύου

Εικόνα 33 : Οθόνες μη κανονικής λειτουργίας της Herb Tracker

Κεφάλαιο 5 - Επίλογος

Σύνοψη - γνωστά προβλήματα

Συνοψίζοντας τα αποτελέσματα και την εμπειρία που αποκτήθηκε κατά την εκπόνηση της εργασίας αυτής, μπορεί να επιβεβαιωθεί ότι η ταυτοποίηση βοτάνων και ευεργετικών φυτών είναι ένα αρκετά δύσκολο αλλά και ενδιαφέρον έργο στο πλαίσιο της μηχανικής μάθησης. Η μορφολογική πολυπλοκότητα και ομοιότητα που παρουσιάζουν αρκετά είδη, καθιστούν την αναγνώριση τους δύσκολη. Κατ' επέκταση γίνεται αναγκαία η εκτενής εκπαίδευση των μοντέλων μηχανικής μάθησης, με όσο το δυνατόν μεγαλύτερα σύνολα δεδομένων. Παρουσιάστηκαν, δυο διαφορετικά μοντέλα μηχανικής μάθησης, το DenseNet-121 και MobileNetV2. Οι αποδόσεις και των δύο μοντέλων είναι αρκετά καλές τόσο σε αποτέλεσμα, όσο και σε ταχύτητα αναγνώρισης. Στη συνέχεια περιγράφηκε η διαδικασία μετατροπής των μοντέλων από Keras σε TensorFlow lite για την συμβατότητα τους στο Android app development. Μία διαδικασία που θα πρέπει οι προγραμματιστές να είναι προσεκτικοί, ώστε να γνωρίζουν τις εισόδους και εξόδους των μοντέλων, για να τη μεταφέρουν αντίστοιχα, στον κώδικα της Kotlin.

Στο σημείο αυτό, οφείλεται να αναφερθεί πως κατά τη διάρκεια της εκπόνησης της εργασίας, η TensorFlow προχώρησε σε σημαντικές αλλαγές στις βιβλιοθήκες της. Μεταξύ άλλων, οι αλλαγές αφορούσαν τον τρόπο που εκπαιδεύονται και μετατρέπονται τα μοντέλα σε συμβατά για το Android. Γεγονός που οδήγησε σε αλλαγές στον κώδικα και τις διαδικασίες της παρούσας εργασίας.

Εν συνεχεία, παρουσιάστηκε η βιβλιοθήκη Jetpack Compose με τα πλεονεκτήματα, τους λόγους που επιλέχθηκε, την αρχιτεκτονική που αναπτύχθηκε και όλη η λειτουργικότητα της εφαρμογής Herb Tracker.

Τέλος, όσον αφορά προβλήματα που εντοπίστηκαν, σχετικά με τα μοντέλα μηχανικής μάθησης το βασικότερο πρόβλημα συναντάται στον τρόπο λειτουργίας των μοντέλων. Τα μοντέλα λειτουργούν με θετικό τρόπο εκτέλεσης, το οποίο σημαίνει ότι οποιαδήποτε εικόνα τους δοθεί, την αναλύουν ώστε να την τοποθετήσουν σε κάποια κλάση/είδος που έχουν διαθέσιμο. Αυτό είναι επιθυμητό στην περίπτωση που ο χρήστης γνωρίζει ότι φωτογραφίζει ή δίνει μια εικόνα, από κάποιο από τα είδη που έχουν εκπαιδευτεί τα μοντέλα. Όταν, όμως, πρόκειται για μη σχετικά αντικείμενα ή παρόμοια είδη φυτών, τότε είναι πιθανόν αυτά να κατηγοριοποιηθούν λανθασμένα, με μεγάλο ποσοστό απόδοσης. Αυτό το πρόβλημα, δεν σχετίζεται με το σύνολο των δεδομένων στο οποίο εκπαιδεύτηκαν τα μοντέλα, αλλά με όλα τα υπόλοιπα δεδομένα που δεν έχουν συμπεριληφθεί στην εκπαίδευση των αλγορίθμων. Η λύση του προβλήματος είναι αρκετά χρονοβόρα, καθώς περιλαμβάνει την εύρεση εικόνων για όλα τα αντικείμενα και είδη που δεν είναι στο επιθυμητό σύνολο δεδομένων και την επανεκπαίδευση των μοντέλων.

Όσον αφορά την εφαρμογή, η βελτίωση της εμφάνισης και της λειτουργικότητας είναι η πρώτη προτεραιότητα.

Μελλοντικά βήματα

Ο τελικός στόχος είναι η δημοσίευση της εφαρμογής στο Google Play, ώστε να είναι διαθέσιμη στο ευρύτερο κοινό και σε συνέχεια αυτού η προσκόμιση κερδών είτε μέσω διαφημίσεων ή/και δωρεών. Για την επίτευξη του στόχου αυτού, τα επόμενα βήματα είναι η επίλυση των προηγούμενων προβλημάτων που αναφέρθηκαν.

Στα μοντέλα μηχανικής μάθησης πρέπει να γίνει επαύξηση του ήδη υπάρχοντος συνόλου δεδομένων. Η διαδικασία αυτή περιλαμβάνει περισσότερα είδη βοτάνων, περισσότερες εικόνες στα ήδη υπάρχοντα είδη για εξισορρόπηση του συνόλου και κυρίως, τουλάχιστον μια ετικέτα με όσο το δυνατόν μεγαλύτερο αριθμό και διαφορετικών εικόνων, ώστε να αποφεύγεται η λανθασμένη κατηγοριοποίηση. Στη συνέχεια, θα χρειαστούν περαιτέρω παραμετροποιήσεις και βελτιστοποίηση της διαδικασίας εκπαίδευσης των μοντέλων. Τέλος, σε ακόμη πιο μελλοντικό χρόνο, η ένθεση περισσότερων μοντέλων αναγνώρισης.

Στον κώδικα της εφαρμογής, κατ' αρχάς, θα γίνουν κάποιες πιο βασικές αλλαγές. Τέτοιες είναι: η εισαγωγή εικόνας στην κάθε πινέζα του χάρτη για μεγαλύτερη ευκρίνεια, η εισαγωγή λειτουργικότητας της αφαίρεσης σημείων από το χάρτη και η αλλαγή της τοπικής βάσης δεδομένων σε αποθήκευση των σημείων σε cloud. Βελτίωση επιδέχεται και το UI, ώστε να παρέχει καλύτερη εμπειρία χρήστη. Η εισαγωγή περισσότερων πληροφοριών για το αναγνωρισθέν βότανο ή ευεργετικό φυτό, σχετικά με τις χρήσεις του ή/και τί ακριβώς προσφέρει στον ανθρώπινο οργανισμό. Τέλος, σε ακόμη πιο μελλοντικό χρόνο, τη δημιουργία προφίλ χρήστη και διαδικτυακής κοινότητας, ώστε να υπάρξει διαδραστικότητα, διαμοιρασμός πληροφορίας και ακόμη μεγαλύτερο ενδιαφέρον προς την φύση.

Βιβλιογραφία

1. Google for developers (2022, April), *Android Basics with Compose*, Retrieved 2023 October. https://developer.android.com/courses/android-basics-compose/course
2. Google for developers (2022, August), *Jetpack Compose for Android developers*, Retrieved 2023 December. https://developer.android.com/courses/jetpack-compose/course
3. TensorFlow (2023, October, 27), *TensorFlow tutorials – Image Classification*, Retrieved 2023 December.https://www.tensorflow.org/tutorials/images/classification
4. TensorFlow (2023, August, 30), *TensorFlow tutorials – Post-training quantization*, Retrieved 2023 December. https://www.tensorflow.org/lite/performance/post_training_quantization
5. TensorFlow (2023, August, 30), *Adding metadata to TensorFlow Lite models*, Retrieved 2023 December. https://www.tensorflow.org/lite/models/convert/metadata
6. Sparkfun (2023, August, 22), *A (Brief) History of Machine Learning*, Retrieved 2024, January. https://www.sparkfun.com/news/7896
7. Lauren Shroll (2022, September, 26), *A Brief History of Android: Founding, Evolution & Industry Impact*, Retrieved 2024, January. https://codesubmit.io/blog/history-of-android-operating-system/
8. Elye (2022, October, 29), *How Android Development Evolves Over The Years*, Retrieved 2024, January. https://medium.com/mobile-app-development-publication/how-android-development-evolves-over-the-years-4abd583ef692
9. Kritsis, K., Kiourt, C., Stamouli, S., Sevetlidis, V., Solomou, A., Karetsos, G., Katsouros, V., Pavlidis, G., "GRASP-125: A Dataset for Greek Vascular Plant Recognition in Natural Environment." *Sustainability* **2021**, *13*, 11865. https://doi.org/10.3390/su132111865
10. Sandler, Mark, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov and Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018): 4510-4520. https://doi.org/10.48550/arXiv.1801.04381
11. G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261-2269, https://doi.org/10.48550/arXiv.1608.06993
12. Pablo Ruiz (2018, October, 10), *Understanding and visualizing DenseNets*, Retrieved 2024, January. https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a
13. John Callaham (2023, December, 1), *The history of Android: The evolution of the biggest mobile OS in the world*, Retrieved 2024, January. https://www.androidauthority.com/history-android-os-name-789433/
14. Devin Schumacher (2023, April, 23), *ReLU6: A Modified Version of Rectified Linear Unit*, Retrieved 2024, February. https://serp.ai/relu6/
15. Christian Leo (2024, January, 30), *The Math Behind the Adam Optimizer*, Retrieved 2024, February. https://towardsdatascience.com/the-math-behind-adam-optimizer-c41407efe59b#5ff9
16. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam: *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* (2017). https://doi.org/10.48550/arXiv.1704.04861
17. Serdar Yegulalp (2024, January, 5), *What is TensorFlow? The machine learning library explained*, Retrieved 2024, February. https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

18. Yu Sun, Yuan Liu, Guan Wang, Haiyan Zhang: *Deep Learning for Plant Identification in Natural Environment*, Computational Intelligence and Neuroscience, vol. 2017, Article ID 7361042, 6 pages, 2017. https://doi.org/10.1155/2017/7361042
19. Hervé Goëau, Pierre Bonnet, Alexis Joly, Vera Bakić, Julien Barbe, Itheri Yahiaoui, Souheil Selmi, Jennifer Carré, Daniel Barthélémy, Nozha Boujemaa, Jean-François Molino, Grégoire Duché, and Aurélien Péronnet. 2013. Pl@ntNet mobile app. In Proceedings of the 21st ACM international conference on Multimedia (MM '13). Association for Computing Machinery, New York, NY, USA, 423–424. https://doi.org/10.1145/2502081.2502251
20. J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: https://doi.org/10.1109/CVPR.2009.5206848