ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΦΑΡΜΟΓΗ ANDROID ΓΙΑ ΤΗΝ ΗΧΟΓΡΑΦΗΣΗ ΩΔΙΚΩΝ ΠΤΗΝΩΝ ΚΑΙ ΤΗΝ ΤΑΞΙΝΟΜΗΣΗ ΤΟΥΣ ΜΕΣΩ ΕΝΣΩΜΑΤΩΜΕΝΟΥ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ

(BirdUp)

Διπλωματική Εργασία

Μεσσής Αναστάσιος

Επιβλέπων: Νέστωρας Ευμορφόπουλος

Βόλος, Σεπτέμβριος 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ANDROID APP FOR THE RECORDING OF BIRDSONGS AND THEIR CLASSIFICATION THROUGH EMBEDDED NEURAL NETWORK

(BirdUp)

Diploma Thesis

Messis Anastasios

Supervisor: Nestor Evmorfopoulos

Volos, September 2022

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Νέστωρας Ευμορφόπουλος** Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Γεώργιος Θάνος** Μέλος Ε.ΔΙ.Π, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Αθανάσιος Φεύγας** Μέλος Ε.ΔΙ.Π, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 21-09-2022

ΣΧΟΛΙΑ

Στα πλαίσια των σπουδών μου στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, ασχολήθηκα με διαφορετικά αντικείμενα μέχρι να βρω αυτά που μου προξενούσαν το περισσότερο ενδιαφέρον, το Hardware και το Data Science. Συγκεκριμένα, η ενασχόληση με ενσωματωμένα συστήματα και FPGAs και ο συνδυασμός τους με τη Μηχανική Μάθηση και τα Νευρωνικά Δίκτυα μου προκαλούσαν μεγάλο ενδιαφέρον. Όσον αφορά τη διπλωματική εργασία, μετά από σκέψη κατέληξα πως η αρχική μου ιδέα (η ενσωμάτωση νευρωνικού δικτύου σε FPGA για χρήση σε έρευνες πεδίου) θα συνδυαζόταν πιο αρμονικά με μία εφαρμογή για κινητά, και έτσι ακολούθησα αυτό το μονοπάτι. Ήταν μια απόφαση που σαφώς περιλάμβανε μεγάλο ρίσκο, άλλα και τη δυνατότητα εκμάθησης κάτι ολοκληρωτικά αγνώστου. Βλέποντας πίσω τώρα, ήταν μια σωστή επιλογή που μου προσέφερε γνώσεις νέων για εμένα γλωσσών προγραμματισμού και δυνατοτήτων που δεν ήξερα πρωτύτερα. Δημιουργώντας λοιπόν την «BirdUp», μία εφαρμογή καταγραφής και ταξινόμησης ωδικών πτηνών η οποία παρουσιάζεται στην παρούσα διπλωματική, είμαι βέβαιος ότι έχω κάνει μία φιλόδοξη αρχή στον τομέα του Software Development, κάτι που ενισχύει τις δεξιότητες μου και θα μου είναι αδιαμφισβήτητα πολύ χρήσιμο για την επαγγελματική μου ζωή.

Θέλω να ευχαριστήσω θερμά τον Δρ. Ευμορφόπουλο Νέστωρα για την άψογη συνεργασία και την προθυμία να βοηθήσει σε κάθε πρόβλημα μου στην παρούσα διπλωματική.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

Μεσσής Αναστάσιος

21-09-2022

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία πραγματεύεται την κατασκευή μιας Android εφαρμογής καταγραφής και ταξινόμησης ωδικών πτηνών για κινητά. Ο χρήστης έχει τη δυνατότητα να ηχογραφήσει ένα ωδικό πτηνό και να το ταξινομήσει ως ένα από τα 50 είδη στα οποία έχει εκπαιδευτεί το νευρωνικό δίκτυο. Επίσης, έχει τη δυνατότητα να δει το ποσοστό βεβαιότητας για τα πιθανότερα είδη και να αποθηκεύσει τις ηχογραφήσεις που αυτός επιθυμεί. Δημιουργήθηκε με την βοήθεια του περιβάλλοντος ανάπτυξης «Android Studio», και του «TensorFlow API». Ο κώδικας για το νευρωνικό δίκτυο είναι γραμμένος σε Python και για την εφαρμογή στην αντικειμενοστραφή γλώσσα Kotlin, την επίσημη γλώσσα δημιουργίας εφαρμογών. Γενικότερα στόχος αυτής της εργασίας είναι η απόκτηση εμπειρίας και ικανοτήτων στο πλαίσιο κατασκευής εφαρμογών.

ABSTRACT

This thesis presents the implementation of an Android mobile application for the recording and classifying of bird species. The user has the ability to record a birdsong and classify it as one of the 50 species that the neural network has been trained on. Furthermore, they can view the percentage of certainty for the most probable species and save the recordings they desire. The app was created in the context of «Android Studio» development environment with the help of «TensorFlow API». The code for the neural network is written in Python and for the app in the object-oriented language Kotlin, the official programming language for Android app development. The general purpose of developing this application is to gain experience and skills in the field of mobile app development.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Η τεχνολογία τα τελευταία χρόνια έχει γίνει αναπόσπαστο κομμάτι της καθημερινότητας των ανθρώπων και αυτό είναι ξεκάθαρο στη χρήση του κινητού τηλεφώνου, πιο γνωστού τώρα ως “smartphone”. Η ύπαρξη πολυπληθών εφαρμογών στοχευμένων στη διευκόλυνση της καθημερινής επικοινωνίας, ψυχαγωγίας, πλοήγησης και σε πολλές άλλες υπηρεσίες είναι τρανταχτά επιχειρήματα που επιδεικνύουν την όλο και στενότερη σχέση που έχουμε με τα κινητά μας τηλέφωνα. Συγκεκριμένα μάλιστα, η ψυχαγωγία σαν τομέας έχει γνωρίσει τεράστια άνθηση στο περιβάλλον του smartphone, προσφέροντας πληθώρα ευκαιριών σε χρήστες και προγραμματιστές παράλληλα.

Στον τομέα της ψυχαγωγίας συμπεριλαμβάνονται φυσικά και εφαρμογές που επικεντρώνονται σε διάφορα χόμπι και γνωστικά πεδία. Εφαρμογές που στοχεύουν στην εκμάθηση γεωγραφίας μέσω διαδραστικών παιχνιδιών, την αναπαραγωγή μουσικής με τη χρήση εικονικών οργάνων, την εκμάθηση ξένων γλωσσών και πολλών άλλων δεξιοτήτων έχουν πολύ υψηλές θέσεις στις προτιμήσεις των καταναλωτών, ακριβώς επειδή η ευκολία χρήσης και το διαδραστικό περιβάλλον του smartphone μετατρέπουν την εκμάθηση σε μια ευχάριστη και άνετη εμπειρία.

Αξίζει να σημειωθεί, πως, όσο εξελίσσεται η τεχνολογία και οι δυνατότητες των συσκευών smartphone, την ίδια εξέλιξη θα γνωρίζουν και αυτές οι εφαρμογές. Η ευκολία χρήσης, η διάθεση εξειδικευμένων εργαλείων και η υπέρογκη πληροφορία που μπορεί να χωρέσει σε τέτοιες εφαρμογές τις καθιστά αξιόπιστα επιστημονικά εργαλεία. Σχετικά με το κομμάτι καταγραφής και ταξινόμησης ωδικών πτηνών, υπάρχουν εφαρμογές που διαχειρίζονται αυτό το αντικείμενο με τρομερή ακρίβεια και παροχή πληθώρας ειδών. Ως αποτέλεσμα, η καταγραφή και ταξινόμηση πληθυσμών - πόσο μάλλον νυκτόβιων - από επιστήμονες και ερευνητές πεδίου γίνεται ακόμα πιο εύκολη, καθώς η χρήση ηχητικών εργαλείων στο φυσικό περιβάλλον είναι πιο αποδοτική και στοχευμένη από τον εντοπισμό με γυμνό μάτι και γενικότερα δυνατών οπτικών μέσων. Αυτό

ήταν και το αρχικό κίνητρο για τη δημιουργία της Android εφαρμογής «BirdUp» στην οποία επικεντρώνεται η παρούσα διπλωματική εργασία.

1.1 Δημοφιλείς εφαρμογές ταξινόμησης πτηνών

Ο αριθμός των διαθέσιμων εφαρμογών / Web APIs για ταξινόμηση ωδικών πτηνών ομολογουμένως δεν είναι μεγάλος, καθώς αποτελούν εργαλεία με ιδιαίτερο και εξειδικευμένο σκοπό. Παρόλα αυτά, οι υπάρχουσες εφαρμογές χαρακτηρίζονται από μεγάλο εύρος δυνατοτήτων, ευχρηστία και αξιοπιστία στις ταξινομήσεις τους, καθώς πολλές από αυτές είναι προϊόν συνεργασίας με ακαδημαϊκούς φορείς και πανεπιστήμια. Πρέπει να αναφερθεί ότι κάποιες από τις ακόλουθες εφαρμογές τελούν ταξινόμηση αποκλειστικά βάσει φωτογραφιών ή ήχου, ενώ άλλες έχουν τη δυνατότητα για οπτική αλλά και ηχητική ταξινόμηση. Παράλληλα υπάρχουν και εφαρμογές “Οδηγοί”, οι οποίες λειτουργούν κυριότερα ως εγκυκλοπαίδειες και αφήνουν την ταξινόμηση στο χρήστη, παρέχοντας βασικά γνωρίσματα για κάθε είδος. Συμπεριλαμβάνονται όλες για να δοθεί μια συνολική εικόνα στο υπάρχον υλικό.

Οι εφαρμογές είναι οι εξής:

Οδηγοί(Guides)

		Λειτουργικό Σύστημα(OS)	Είδος Ταξινόμησης
1.	**Audubon Bird Guide**	**Android, iOS**	**Εικόνα, metadata***
2.	**BirdsEye Bird Finding Guide**	**Android, iOS**	**Εικόνα, metadata**
3.	**eBird**	**Android, iOS, Web**	**Εικόνα, metadata**

4.	**iBird Yard+ Guide to Birds**	**Android, iOS**	**Εικόνα, metadata**
5.	**iBird Pro**	**Android, iOS**	**Εικόνα, metadata**
6.	**National Geographic Birds**	**iOS**	**Εικόνα, metadata**
7.	**Sibley eGuide to Birds**	**iOS**	**Εικόνα, metadata**

*****Βάσεις δεδομένων που περιλαμβάνουν στοιχεία όπως χρώμα, μέγεθος, συμπεριφορά, μεταναστευτικές συνήθειες, σχήμα και χρώμα αυγών, etc..

Ταξινόμηση (Recognition)

		Λειτουργικό Σύστημα(OS)	Είδος Ταξινόμησης
1.	**BirdNet**	**Android, iOS**	**Ήχος**
2.	**BirdSong ID**	**Android, iOS**	**Ήχος**
3.	**ChirpOMatic**	**iOS**	**Ήχος**
4.	**Picture Bird – Bird Identifier**	**Android, iOS**	**Εικόνα, Ήχος**
5.	**SongSleuth**	**Android, iOS**	**Ήχος**
6.	**Merlin Bird Id**	**Android, iOS**	**Εικόνα, Ήχος**

Ως διευκρίνιση, πρέπει να σημειωθεί πως οι εφαρμογές eBird [1], Merlin Bird Id [2] είναι ιδιοκτησία του πανεπιστημίου Cornell - συγκεκριμένα του Cornell Ornithology Lab [3] - ενώ οι BirdNet [4] και Audubon [5] αποτελούν προϊόν συνεργασίας φορέων με το Cornell Ornithology Lab. Σχετικά με τις εφαρμογές ταξινόμησης βάσει τραγουδιού, το BirdNet αποδεικνύεται να είναι η πιο εύστοχη από τις υπόλοιπες, με ChirpOMatic [6], Songsleuth [7] και BirdSong ID να λαμβάνουν χαμηλότερες θέσεις [8].

Πέρα από την οπτική και ηχητική ταξινόμηση, οι παραπάνω εφαρμογές κάνουν χρήση της τοποθεσίας του χρήστη για την παροχή ακριβέστερων αποτελεσμάτων, καθώς ανάλογα με την εποχή η τοποθεσία πολλών ειδών αλλάζει. Βέβαια, αυτό ενδέχεται να δρα μειονεκτικά, καθώς κάποιες εφαρμογές επικεντρώνονται σε συγκεκριμένες ηπείρους και όχι σε παγκόσμιο επίπεδο.

Στη δημιουργία του «BirdUp» πάρθηκε έμπνευση από την εφαρμογή BirdNet, ενώ το νευρωνικό δίκτυο εκπαιδεύτηκε με δεδομένα από το Xeno-Canto Web API [9]. Εκτενέστερη περιγραφή των δημοφιλέστερων εφαρμογών από τις παραπάνω ακολουθεί στις επόμενες υποενότητες.

1.1.1 – eBird

Διαθέσιμο σε iOS, Android και PC.

Το eBird ξεκίνησε ως μία διαδικτυακή βάση δεδομένων με κύριο στόχο την καταγραφή πληθυσμών πτηνών για λόγους προστασίας τους, με τους χρήστες του να είναι από αρχάριους παρατηρητές και φυσιοδίφες μέχρι εξειδικευμένους ερευνητές και επιστήμονες. Δημιουργήθηκε από το Εργαστήριο Ορνιθολογίας του Πανεπιστημίου Cornell το 2002, και σήμερα διατίθεται σε εφαρμογή για κινητά, ενώ η εφαρμογή Merlin Bird Id έχει ενσωματωθεί σε αυτό. Συνεργάζεται με πολλαπλούς φορείς συντήρησης της βιοποικιλότητας και αποτελεί βασική εργαλείο και πηγή πληροφορίας για αυτούς αλλά και για οποιονδήποτε επιθυμεί να ασχοληθεί με τον τομέα. Ο κάθε χρήστης έχει τη δυνατότητα να καταγράψει τις προσωπικές του παρατηρήσεις και να τις οπτικοποιήσει μέσα από χάρτες και γραφήματα, λαμβάνοντας έτσι μια συνολική εικόνα της προσπάθειας του ίδιου και ολόκληρης της κοινότητας.

Εικόνα 1.1 - Το λογότυπο της πλατφόρμας eBird

1.1.2 – Merlin Bird Id

Διαθέσιμο σε iOS και Android.

Η εφαρμογή Merlin Bird Id είναι κατασκευασμένη από το Cornell Lab of Ornithology και λειτουργεί ως οδηγός πεδίου με δυνατότητες ταυτοποίησης 3000 ειδών ενδημικών στην Αμερική, τη δυτική Ευρώπη και την Ινδία. Οι χρήστες μπορούν να απαντήσουν απλές ερωτήσεις σχετικά με το πτηνό που έχουν παρατηρήσει και να λάβουν μια ολοκληρωμένη λίστα με τα πιθανότερα είδη, συνοδευόμενη από φωτογραφίες και ήχους για το κάθε είδος. Σε νεότερες εκδόσεις έχουν προστεθεί οι λειτουργίες Photo ID και Sound ID, οι οποίες μπορούν να κάνουν αυτόματη ταυτοποίηση ειδών βάσει φωτογραφιών και ήχου αντίστοιχα, με τη χρήση Τεχνητής Νοημοσύνης. Από το 2018 έχει ενσωματωθεί στην πλατφόρμα eBird, προσφέροντας μια πιο ολοκληρωμένη και δυναμική εμπειρία στο χρήστη.

Εικόνα 1.2 - Το λογότυπο της εφαρμογής Merlin Bird Id

1.1.3 – BirdNet

Διαθέσιμο σε iOS και Android.

Το Cornell Ornithology Lab σε συνεργασία με το Chemnitz University of Technology κατασκεύασαν το BirdNet, μία εφαρμογή που εκτελεί ηχητική ταυτοποίηση ειδών με τη χρήση Τεχνητής Νοημοσύνης και τη συνδυάζει με την τοποθεσία του χρήστη για να προσφέρει όσο το δυνατόν πιο εύστοχες προβλέψεις. Αυτήν τη στιγμή μπορεί να αναγνωρίσει μέχρι και 3000 είδη παγκοσμίως. Αναλυτικότερα, το BirdNet μεταχειρίζεται ακατέργαστα αρχεία ήχου μαζί με metadata και μετατρέπει τον κάθε ήχο σε εικόνα(φασματογράφημα [10]), πάνω στο οποίο κάνει ταυτοποίηση με το ισχυρό τεχνητό νευρωνικό δίκτυο του και επαληθεύει τις προβλέψεις του με τη βοήθεια της πλατφόρμας eBird.

Η συγκεκριμένη εφαρμογή έχει χρησιμοποιηθεί εν μέρει ως έμπνευση για τη δημιουργία της εφαρμογής που παρουσιάζεται σε αυτήν την πτυχιακή.

Εικόνα 1.3 - Το λογότυπο της εφαρμογής BirdNet

1.2 Εισαγωγή στην εφαρμογή BirdUp

Η εφαρμογή της παρούσας διπλωματικής χωρίστηκε σε δουλειά 2 εξαμήνων. Αρχικά, έγινε έρευνα για τις μεθόδους ηχητικής ταξινόμησης ωδικών πτηνών και στη συνέχεια κατασκευάστηκε το dataset και το νευρωνικό δίκτυο μαζί με τα απαραίτητα στάδια προ επεξεργασίας των αρχείων ήχου. Μόλις το νευρωνικό δίκτυο έφτασε σε ένα αποδεκτό επίπεδο, ξεκίνησε ο προγραμματισμός της εφαρμογής, με σημείο αναφοράς την εφαρμογή BirdNet, όπως προαναφέρθηκε. Η εφαρμογή ονομάζεται «BirdUp» και προορίζεται για smartphones ή tablets με λειτουργικό Android. Στόχος της είναι η ταυτοποίηση και καταγραφή ωδικών πτηνών μέσω του τραγουδιού τους.

Εικόνα 1.4 - Το λογότυπο της εφαρμογής BirdUp

Οι δυνατότητες της εφαρμογής είναι οι βασικές και το User Interface περιέχει μόνο τις απαραίτητες λειτουργίες για τη διευκόλυνση του χρήστη. Επιγραμματικά, ο χρήστης μπορεί να ξεκινήσει μία ηχογράφηση, να κάνει παύση και συνέχιση της, να ακούσει την ηχογράφηση πριν αποφασίσει να την περάσει από το νευρωνικό δίκτυο και να τη διαγράψει αν και όταν επιθυμεί.

Μόλις η εφαρμογή έχει παράξει τα πιθανά αποτελέσματα για την κάθε ηχογράφηση, ο χρήστης μπορεί να επιλέξει να την αποθηκεύσει, στην οποία περίπτωση η ηχογράφηση

προστίθεται σε μία λίστα με τις βασικές πληροφορίες της και αποθηκεύεται στον εσωτερικό χώρο αποθήκευσης της εφαρμογής, διαθέσιμη ανά πάσα στιγμή.

Τέλος, υπάρχει η δυνατότητα αποστολής παραπόνων ή προτάσεων για τον κάθε χρήστη μέσω mail από το περιβάλλον της εφαρμογής.

1.3 Διαμόρφωση της εργασίας

Στο επόμενο κεφάλαιο γίνεται ανάλυση της έρευνας γύρω από την κατασκευή του νευρωνικού δικτύου και συμπεριλαμβάνονται τα βήματα που ακολουθήθηκαν για τη συναρμολόγηση του κατάλληλου dataset, τη σωστή προ επεξεργασία αυτού του dataset, καθώς επίσης και οι δοκιμές των διαφορετικών μοντέλων μέχρι την εκπαίδευση του τελικού μοντέλου που είναι ενσωματωμένο στην εφαρμογή.

Στο κεφάλαιο 3 γίνεται λεπτομερής ανάλυση των δυνατοτήτων της εφαρμογής και περιγράφεται η λειτουργία κάθε ξεχωριστού σταδίου.

Στο κεφάλαιο 4 περιγράφεται το λειτουργικό σύστημα Android, το TensorFlow API, το περιβάλλον «Android Studio» και η γλώσσα προγραμματισμού Kotlin, όπως επίσης και τα διάφορα πακέτα και βιβλιοθήκες που χρησιμοποιήθηκαν στην κατασκευή της εφαρμογής, τόσο στο κομμάτι του κώδικα Android όσο και στη σύνθεση του νευρωνικού δικτύου.

Στο κεφάλαιο 5 περιλαμβάνεται αναλυτική περιγραφή της εφαρμογής σχετικά με την αρχιτεκτονική κώδικα και τους πόρους που απασχολήθηκαν.

Στο κεφάλαιο 6 παρουσιάζονται μερικά συμπεράσματα, σημαντικά εμπόδια που εμφανίστηκαν κατά την κατασκευή αλλά και μελλοντικές επεκτάσεις και αναβαθμίσεις που μπορούν να ενσωματωθούν στην εφαρμογή.

Συνέχεια έχει η Βιβλιογραφία, στην οποία περιέχονται όλες οι σχετικές πηγές οι οποίες στο κείμενο παρουσιάζονται ως αριθμοί έγκλειστοι σε αγκύλες [].

Τελευταία είναι τα παραρτήματα. Το παράρτημα Α καταγράφει αλφαβητικά τα ακρωνύμια και τις συντομογραφίες που εμφανίζονται στο κείμενο.

Το παράρτημα Β περιέχει οδηγίες για τον αναγνώστη σχετικά με το κατέβασμα και την εγκατάσταση της εφαρμογής στην προσωπική Android συσκευή του.

ΚΕΦΑΛΑΙΟ 2

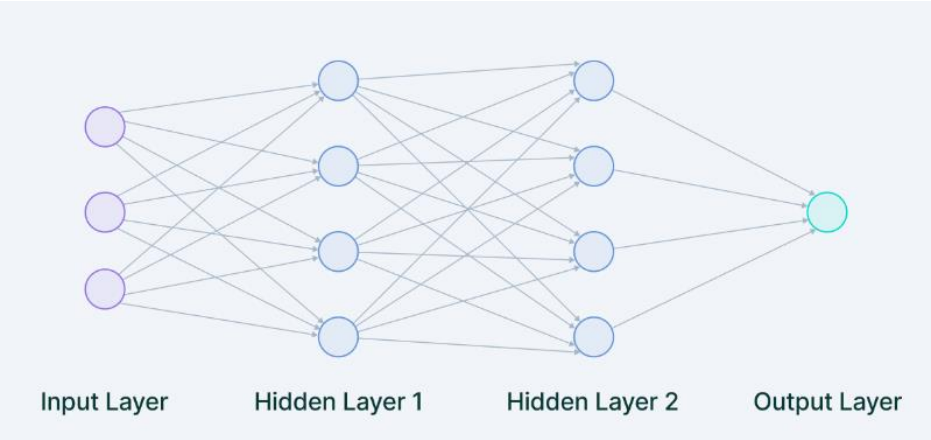
ΤΟ ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ ΚΑΙ Η ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ

2.1 Ορισμός και Αρχιτεκτονικές

Ξεκινώντας από τα βασικά, ένα τεχνητό νευρωνικό δίκτυο (ANN) βάσει ορισμού είναι μία σειρά από υψηλά διασυνδεδεμένους νευρώνες ή αλλιώς κόμβους που χρησιμοποιούν αλγορίθμους για να αναγνωρίσουν υποκείμενες σχέσεις σε ένα σύνολο από δεδομένα [11]. Η μέθοδος με την οποία το κάνουν αυτό μιμείται τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου, εξ ου και το όνομα νευρωνικό δίκτυο. Ο κύριος στόχος αυτών των δικτύων είναι η επίλυση πολύπλοκων προβλημάτων που συνήθως εντάσσονται στον τομέα της Τεχνητής Νοημοσύνης.

Το πεδίο της Τεχνητής Νοημοσύνης είναι μία τεχνολογία που έχει γνωρίσει τρομερή άνθηση τα τελευταία χρόνια και χρησιμοποιείται σε πολλούς τομείς της τεχνολογίας και της ανθρώπινης ζωής. Μπορεί να εφαρμοστεί από την ανασκόπηση [12] και βελτιστοποίηση [13] προγραμματιστικού κώδικα σε εταιρικά περιβάλλοντα, μέχρι και στη γεωργία, προσφέροντας πιο αποδοτικές μεθόδους καλλιέργειας [14] και ποτίσματος, αλλά και υποστήριξη υγειονομικού ελέγχου και προστασίας των καλλιεργειών [15], όπως έχουν πετύχει startups του Πανεπιστημίου Θεσσαλίας . Είναι προφανές από τα παραπάνω πως η Τεχνητή Νοημοσύνη είναι ένα πολύ ευρύ πεδίο και προκειμένου να μπορεί να εφαρμοστεί σε τόσες πτυχές τις καθημερινότητάς μας απαιτείται μία εξειδίκευση στα εργαλεία που τη χρησιμοποιούν. Το ίδιο ισχύει και για τα νευρωνικά δίκτυα.

Υπάρχουν διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων που εξειδικεύονται σε διαφορετικής φύσης προβλήματα. Γενικά όμως, όλα τα νευρωνικά δίκτυα βασίζονται πάνω στην ίδια αρχιτεκτονική, το ίδιο και οι νευρώνες τους. Αυτή η αρχιτεκτονική είναι η ακόλουθη:



Εικόνα 2.1 - Η βασική αρχιτεκτονική ενός νευρωνικού δικτύου [16]

Όπως φαίνεται και από την εικόνα, κάθε νευρωνικό δίκτυο αποτελείται από στρώματα (layers), τα οποία με τη σειρά τους αποτελούνται από συστάδες διασυνδεδεμένων νευρώνων με συγκεκριμένα χαρακτηριστικά.

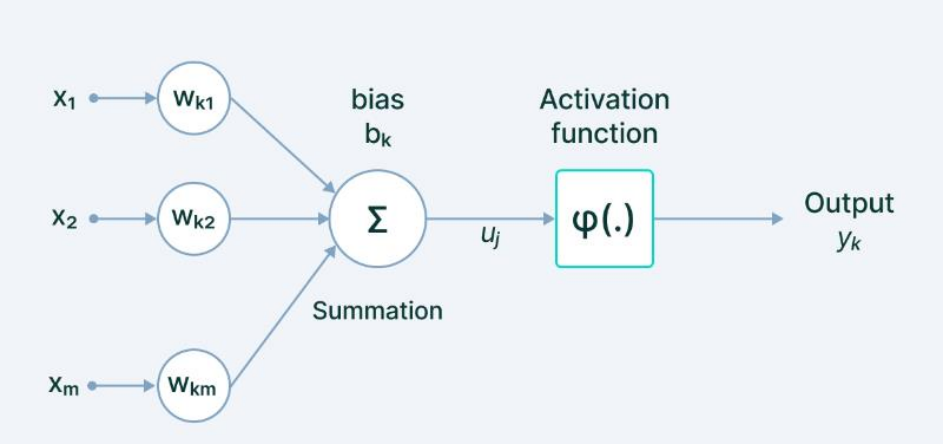
Τα δεδομένα που παρέχουμε στο μοντέλο φορτώνονται στο αρχικό στρώμα (Input Layer) μέσω εξωτερικών πηγών, λόγου χάρη ένα αρχείο CSV ή ενός Web API. Είναι το μοναδικό φανερό στρώμα του νευρωνικού δικτύου και παρέχει τα ζητούμενα δεδομένα από τον έξω κόσμο στο υπόλοιπο μοντέλο αδιάλλακτα, χωρίς μεταποιήσεις [16].

Ακολουθούν τα κρυφά στρώματα (Hidden Layers), τα οποία μπορεί να είναι ένα ή περισσότερα. Εδώ γίνονται όλοι οι απαραίτητοι υπολογισμοί και αποσπώνται τα χρήσιμα χαρακτηριστικά από τα δεδομένα. Συνήθως υπάρχουν πολλαπλά διασυνδεδεμένα κρυφά στρώματα και το καθένα από αυτά αναζητά διαφορετικά χαρακτηριστικά στα δεδομένα. Επί παραδείγματι, στην επεξεργασία εικόνας τα πρώτα κρυμμένα στρώματα είναι υπεύθυνα για την απόσπαση χαρακτηριστικών υψηλότερου επιπέδου, όπως γωνίες, σχήματα ή όρια. Από την άλλη, τα επόμενα στρώματα εκτελούν πιο πολύπλοκες διεργασίες, όπως την ταυτοποίηση ολόκληρων αντικειμένων [16].

Τέλος είναι το στρώμα εξόδου (Output Layer), που δέχεται ως είσοδο τα αποτελέσματα των κρυμμένων στρωμάτων και είναι επιφορτισμένο με την παραγωγή της τελικής πρόβλεψης. Είναι

το πιο σημαντικό στρώμα όπου λαμβάνουμε το τελικό αποτέλεσμα. Σε προβλήματα classification και regression το στρώμα αυτό έχει κυρίως ένα κόμβο ή νευρώνα. Ωστόσο, εξαρτάται αποκλειστικά από το πρόβλημα και τον τρόπο με τον οποίο παράχθηκε το μοντέλο.

Η αρχιτεκτονική για τους νευρώνες ενός δικτύου είναι η εξής:



Εικόνα 2.2 - Η αρχιτεκτονική ενός απλού νευρώνα [16]

Τα συστατικά μέρη ενός νευρώνα είναι η είσοδος, τα βάρη, η συνάρτηση μεταφοράς ή ενεργοποίησης και η κλίση. Η λειτουργία τους είναι με τη σειρά:

Είσοδος (Input): Το σύνολο των χαρακτηριστικών που εισέρχονται στο μοντέλο με στόχο την εκπαίδευση του. Για παράδειγμα, είσοδος σε ένα μοντέλο για ανίχνευση αντικειμένων μπορεί να είναι ένας πίνακας με τιμές pixels παρμένες από μία εικόνα.

Βάρος (Weight): Η κύρια λειτουργία του είναι να δώσει αξία σε αυτά τα χαρακτηριστικά που συνεισφέρουν περισσότερο στην εκπαιδευτική διαδικασία. Το πετυχαίνει αυτό εφαρμόζοντας κλιμακωτούς πολλαπλασιασμούς μεταξύ της αρχικής τιμής της εισόδου και του πίνακα βαρών. Για παράδειγμα, μία αρνητική λέξη θα επηρέαζε περισσότερο την πρόβλεψη σε μία ανάλυση συναισθημάτων από ότι ένα σύνολο ουδέτερων λέξεων.

Συνάρτηση Ενεργοποίησης (Activation Function): Η δουλειά της συνάρτησης ενεργοποίησης είναι να συνδυάσει πολλαπλές εισόδους στην είσοδο της σε μία μοναδική τιμή στην έξοδο της, έτσι ώστε να μπορέσει να παραχθεί μία πρόβλεψη στο τέλος του στρώματος. Αυτό επιτυγχάνεται με μία απλή άθροιση όλων των εισόδων. Πολλές από αυτές τις συναρτήσεις είναι μη γραμμικές και διαφορετικά είδη τους μπορούν να χρησιμοποιηθούν σε κάθε στάδιο του μοντέλου για να βελτιώσουν την επίδοση του.

Κλίση (Bias): Ο ρόλος της κλίσης είναι να μετατοπίσει την τιμή που υπολογίζεται από τη συνάρτηση ενεργοποίησης, παρόμοια με το ρόλο μίας σταθεράς σε μία γραμμική συνάρτηση. Είναι ένα πολύ χρήσιμο στοιχείο που μπορεί να επισπεύσει την επιτυχή εκπαίδευση του μοντέλου.

Οι διαφορές στις αρχιτεκτονικές των ειδών νευρωνικών δικτύων εστιάζουν κυρίως στην κατεύθυνση της πληροφορίας μεταξύ των νευρώνων, τον αριθμό των απασχολημένων στρωμάτων και την τοπολογία τους. Ενδεικτικά, οι πιο γνωστές αρχιτεκτονικές είναι:

Perceptron: Το perceptron είναι η απλούστερη αρχιτεκτονική για νευρωνικά δίκτυα. Δέχεται έναν αριθμό εισόδων, εφαρμόζει ορισμένες μαθηματικές πράξεις και παράγει ένα αποτέλεσμα. Ειδικότερα, παίρνει ένα διάνυσμα εισόδων πραγματικών τιμών και εκτελεί ένα γραμμικό συνδυασμό κάθε χαρακτηριστικού με το αντίστοιχο βάρος που έχει εκχωρηθεί στο καθένα από αυτά. Με λίγα λόγια, αντιπροσωπεύει πως λειτουργεί ένας μεμονωμένος νευρώνας [16].

Feed – Forward Networks: Τα δίκτυα τροφοδότησης είναι ένα σύνολο από perceptrons και, όπως λέει και το όνομα, η πληροφορία σε αυτά περνάει μόνο από την είσοδο στην έξοδο και δεν ανατροφοδοτείται σε προηγούμενα στρώματα [16].

RNN – Recurrent Neural Networks: Σε ένα επαναλαμβανόμενο νευρωνικό δίκτυο οι συνδέσεις μεταξύ κόμβων σχηματίζουν ένα κατευθυνόμενο ή μη κατευθυνόμενο γράφημα κατά μήκος μιας χρονικής ακολουθίας. Αυτό του επιτρέπει να επιδεικνύει χρονικά δυναμική συμπεριφορά. Προερχόμενα από τα feed-forward networks, τα RNN μπορούν να χρησιμοποιήσουν την εσωτερική τους μνήμη για να επεξεργαστούν ακολουθίες εισόδου μεταβλητού μήκους. Αυτό τα καθιστά κατάλληλα για εργασίες όπως η αναγνώριση μη

τμηματοποιημένου χειρόγραφου κειμένου ή η αναγνώριση ομιλίας και η πρόβλεψη χρονοσειρών [17].

LSTM – Long Short-Term Memory: Εξέλιξη του RNN, το συγκεκριμένο είδος δικτύων διαθέτει συνδέσεις στα στρώματα του που επιτρέπουν την ανατροφοδότηση πληροφορίας σε μεγαλύτερο βαθμό από το RNN. Μπορεί να επεξεργαστεί ακολουθίες δεδομένων με μεγαλύτερη ευκολία και εφαρμόζεται, μεταξύ άλλων, στην αναγνώριση ομιλίας, τον έλεγχο ρομπότ και την υγειονομική περίθαλψη. Το LSTM είναι το πιο αναφερόμενο ("most cited") νευρωνικό δίκτυο του 20ου αιώνα [18].

GAN – Generative Adversarial Network: Το GAN είναι κομμάτι της παραγωγικής μοντελοποίησης, η οποία ανήκει στην κατηγορία του unsupervised learning, όπου νέα, συνθετικά δεδομένα δημιουργούνται με βάση τα μοτίβα που ανακαλύπτονται από το σύνολο των δεδομένων εισόδου. Το GAN χρησιμοποιείται για τη δημιουργία συνθετικών δεδομένων και, ως εκ τούτου, είναι ένας ενεργός τομέας έρευνας για την τεχνητή νοημοσύνη. Εξειδικεύεται στην ανίχνευση αντικειμένων και την αναγνώριση και τμηματοποίηση γραπτού κειμένου [16].

CNN – Convolutional Neural Network: Τα συνελικτικά νευρωνικά δίκτυα είναι υποείδος των δικτύων τροφοδότησης και χρησιμοποιούνται σε εργασίες όπως η ανάλυση εικόνας, η επεξεργασία φυσικής γλώσσας και άλλα πολύπλοκα προβλήματα ταξινόμησης εικόνων. Κάνουν εξαγωγή των βασικών χαρακτηριστικών της εικόνας και με τη βοήθεια φίλτρων μπορούν να ανιχνεύουν μοτίβα στα δεδομένα της εικόνας. Όσο πιο βαθιά πηγαίνει το δίκτυο, όσο αυξάνεται δηλαδή ο αριθμός των κρυφών στρωμάτων, τόσο πιο περίπλοκη γίνεται η αναζήτηση μοτίβων [16]**.**

2.2 Επιλογή Αρχιτεκτονικής και Dataset

Όπως είναι απαραίτητο στην κατασκευή κάθε μοντέλου για προβλήματα deep learning, πρωταρχικός στόχος είναι η επιλογή της κατάλληλης μεθόδου εκπαίδευσης και η εύρεση ή δημιουργία ενός αντιπροσωπευτικού dataset.

2.2.1 – Διαθέσιμες Μέθοδοι Εκπαίδευσης

Η επιλογή της κατάλληλης μεθόδου εκμάθησης εξαρτάται από το είδος του προβλήματος αλλά και από τη μορφή των δεδομένων που χρησιμοποιούνται στην εκπαίδευση. Στη βιοακουστική, οι αλγόριθμοι μηχανικής μάθησης εκπαιδεύονται γενικά σε χαρακτηριστικά που εξάγονται από τον ήχο, καθώς η εκπαίδευση απευθείας στον ήχο δεν είναι πρακτική [19]. Πρακτικά, η ακατέργαστη κυματομορφή είναι χρήσιμη για αποθήκευση, αναπαραγωγή και περαιτέρω επεξεργασία. Αλλά η υψηλή διάσταση (high dimensionality) της περιορίζει την άμεση χρήση της στη μηχανική μάθηση. Παραδοσιακά, οι αλγόριθμοι μηχανικής μάθησης εκπαιδεύονται σε εξαγόμενα χαρακτηριστικά, τυπικά αναπαραστάσεις του ήχου που μοιάζουν με φασματογραφήματα [19].

Ορισμένες μετατροπές, συμπεριλαμβανομένου του Fast Fourier Transform (FFT), του Short-Time Fourier Transform (STFT), των συντελεστών γραμμικής πρόβλεψης (LPCs), των wavelets και των chirplets [19], μειώνουν τη διάσταση του ήχου για να διευκολύνουν τη διαδικασία μηχανικής εκμάθησης. Αυτά μπορούν να υποβληθούν σε περαιτέρω επεξεργασία σε χειροποίητα χαρακτηριστικά, όπως εγκεφαλικοί συντελεστές συχνότητας Mel (MFCCs), που μετασχηματίζουν τη γραμμική συχνότητα σύμφωνα με μια αντιληπτική κλίμακα (κλίμακα Mel) που αντικατοπτρίζει τη μη γραμμική ανθρώπινη αντίληψη του ήχου [20].

Τα χαρακτηριστικά που προαναφέρθηκαν χωρίζονται σε 2 κατηγορίες. Τα οπτικά χαρακτηριστικά, στα οποία ανήκουν τα φασματογραφήματα, και τα ηχητικά χαρακτηριστικά, στα οποία ανήκουν τα wavelets, MFCCs, zero-crossing rate [21] και άλλα. Η χρήση των ηχητικών χαρακτηριστικών μετατοπίζει την επίλυση περισσότερο στο speech recognition και το sound event detection (SED), ενώ η χρήση των φασματογραφημάτων επιζητά λύση μέσω image classification. Η εκπαίδευση του μοντέλου μπορεί να γίνει με τη χρήση και των 2 ή μόνο ενός εκ των χαρακτηριστικών.

Στην ήδη υπάρχουσα λογοτεχνία υπάρχουν αρκετές μέθοδοι εκπαίδευσης. Καθώς η ταξινόμηση ωδικών πτηνών μπορεί να συμπεριληφθεί στο φάσμα του speech recognition, η εφαρμογή LSTMs και γενικότερα επαναλαμβανόμενων αρχιτεκτονικών αποδεικνύεται επιτυχής [22]. Ο συνδυασμός CNNs και LSTM-based δικτύων σε προβλήματα speech recognition ενδέχεται να βελτιώσει την ακρίβεια του μοντέλου ακόμα παραπάνω [23]. Η εφαρμογή κατάτμησης βασισμένης σε ταξινόμηση Random Forest [24] στην [25] δείχνει να επιλέγει καλέσματα σε θορυβώδη περιβάλλοντα με 93.6% ακρίβεια. Η εργασία που παρουσιάζεται στο [26] χρησιμοποιεί δεσμευμένο φάσμα συχνοτήτων, χαρακτηριστικά MFCCs και LPCs τα οποία ταξινομούνται από ένα γκρουπ Logistic Regression, Random Forests και εξαιρετικά τυχαιοποιημένων δέντρων. Αξιοσημείωτα, οι Xie et al. [27] χρησιμοποίησαν οπτικά όσο και ηχητικά χαρακτηριστικά για την εκπαίδευση ενός CNN και συνδύασαν τα αποτελέσματα για την ταξινόμηση ήχων πτηνών. Τέλος, η χρήση fine-tuned pre-trained τοπολογιών CNNs (Inception, MobileNet, ResNet) [28-30], αλλά και συνδυαστικά με MLPs [31] και δεδομένα τοποθεσίας έχει αποδειχθεί εξίσου επιτυχής.

Οι επιλογές είναι πολλές και ποικίλες, λαμβάνοντας όμως υπόψη τις περιορισμένες υπολογιστικές δυνατότητες της διαθέσιμης συσκευής, αποφασίστηκε να εφαρμοστεί μία σχετικά απλή τοπολογία CNN για την εκπαίδευση φασματογραφημάτων, ακολουθώντας βήματα της εργασίας των Á. Incze et al. [32]. Παρά την απλότητα της επιλογής, τα CNN με την κατάλληλη ρύθμιση αποτελούν ικανότατη τοπολογία και είναι ισχυρά εργαλεία, ταιριαστά σε μεγάλης κλίμακας dataset. Αυτό υποστηρίζεται και από την εκτεταμένη χρήση τους στη λογοτεχνία.

Επιγραμματικά, θα χρησιμοποιηθούν αρχεία ήχου μεταβλητής διάρκειας σε μορφή .mp3, στα οποία θα γίνει ανάλογη κανονικοποίηση και εφαρμογή φίλτρων. Στη συνέχεια θα τμηματοποιηθούν σε ίσα κομμάτια μεγέθους 5 δευτερολέπτων, και αφού αφαιρεθούν τα κομμάτια που δεν περιέχουν πληροφορία (βάσει συγκρίσεων), θα μετατραπούν σε φασματογραφήματα με τη χρήση STFT. Στο σημείο αυτό θα είναι έτοιμα για να φορτωθούν στο μοντέλο.

2.2.2 – Το Dataset

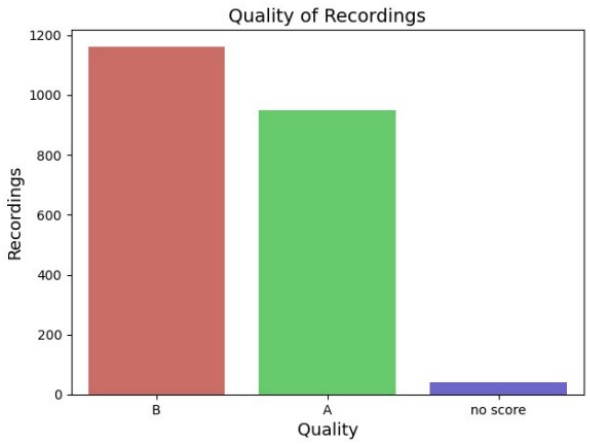
Σαν αρχικό Dataset και βάση του μοντέλου επιλέχθηκε το «**Bird songs from Europe (xeno-canto)**» το οποίο μπορεί να βρεθεί στο Kaggle [33]. Πρόκειται για ένα ισορροπημένο dataset με υψηλής ποιότητας τραγούδια αρσενικών πτηνών από τα 50 επικρατέστερα είδη της Ευρώπης. Ειδικότερα, τα δεδομένα συλλέχθηκαν από το Xeno-Canto API με τη βοήθεια του πακέτου warbleR γλώσσας R [34].

Το API query για το κατέβασμα των ηχογραφήσεων ήταν το ακόλουθο: **type:song type:male len_gt:30 q_gt:C area:Europe.** Με απλά λόγια, το query ζητάει ηχογραφήσεις που είναι τραγούδια, αρσενικών πτηνών, μήκους άνω των 30 δευτερολέπτων, ποιότητας A και B (σε κλίμακα A-D) και με τοποθεσία την Ευρώπη. Στη συνέχεια, φιλτραρίστηκαν τα 50 επικρατέστερα είδη και υποδειγματολήφθηκαν όλα στη μικρότερη συχνότητα που ήταν n = 43. Σαν αποτέλεσμα, το dataset κατέληξε να αποτελείται από 43 x 50 = 2150 ηχογραφήσεις μεταβλητού μήκους. Το query που εκτελείται από το warbleR επιστρέφει metadata σχετικά με τις ηχογραφήσεις τα οποία εμπεριέχονται σε ένα αρχείο .csv.

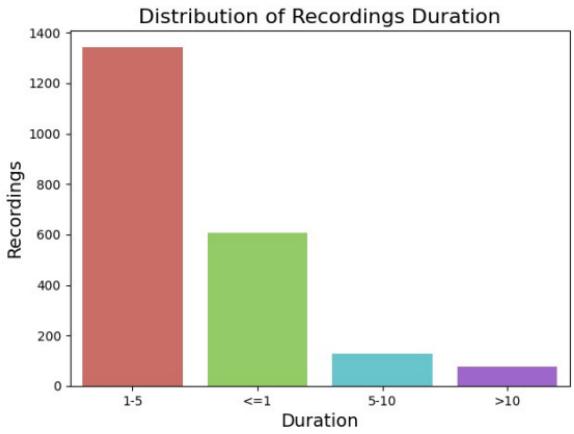
Επισημαίνεται ότι:

- Οι ηχογραφήσεις ενδέχεται να μην περιέχουν αποκλειστικά αρσενικά τραγούδια, αλλά, επί παραδείγματι, μίξεις από καλέσματα και τραγούδια που ανήκουν σε αρσενικά και θηλυκά πτηνά του ίδιου είδους ή και διαφορετικών ειδών.
- Η ποιότητα των ηχογραφήσεων είναι περιορισμένη σε A και B όμως περιέχει και ηχογραφήσεις χωρίς αξιολόγηση.

Η συνολική ποιότητα και διάρκεια των ηχογραφήσεων παρουσιάζεται οπτικοποιημένη παρακάτω. Τα δεδομένα προήλθαν από το αρχείο .csv και οπτικοποιήθηκαν με το πακέτο Matplotlib.



Εικόνα 2.3 – Διανομή της ποιότητας του Dataset



Εικόνα 2.4 – Διανομή της διάρκειας των αρχείων

2.3 Προετοιμασία του Dataset

Προκειμένου τα δεδομένα του dataset να φτάσουν στη μορφή με την οποία θα εισαχθούν στο μοντέλο πρέπει να περάσουν από κάποια στάδια προ επεξεργασίας. Τα στάδια αυτά είναι χωρισμένα σε ξεχωριστά αρχεία python και η λειτουργία τους με τη σειρά περιγράφεται στις ακόλουθες γραμμές:

I. LOAD.PY

Στο πρώτο στάδιο φορτώνονται ξεχωριστά όλα τα αρχεία με μορφή .mp3 και επιστρέφεται ο ήχος μαζί με τον αρχικό ρυθμό δειγματοληψίας (sampling rate) με τη συνάρτηση load() του πακέτου Librosa.

```
21          name, ext = os.path.splitext(filename)
22          if (ext == '.mp3'):
23
24              try:
25                  filepath = [path, filename]
26                  filepath = "".join(filepath)
27
28                  #load audiofile and return original sample rate
29                  audio, sr = librosa.load(path=filepath, sr=None)
```

Στη συνέχεια, γίνεται μετατροπή του αρχείου σε μονοφωνικό κανάλι από στέρεο με τη συνάρτηση to_mono() του πακέτου Librosa. Αυτό συμβαίνει για τη μείωση κατά το ήμισυ του μεγέθους αρχείων και ως αποτέλεσμα, τη γρηγορότερη επεξεργασία του κάθε αρχείου. Η βελτιωμένη ποιότητα που προσφέρει ο στέρεο ήχος κυρίως για αισθητικούς σκοπούς και δεν έχει αποδειχθεί να οδηγεί στην δημιουργία καλύτερων φασματογραφημάτων.

```
31              #convert all files to mono channel
32              audio = librosa.to_mono(audio)
```

Ύστερα, το αρχείο δειγματοληπτείται ξανά με ρυθμό 24kHz και αποθηκεύεται σε αρχείο με το ίδιο όνομα σε μορφή .wav. Η δειγματοληψία στα 24kHz συμβαίνει επειδή το εύρος ακοής των περισσότερων πτηνών είναι από 50Hz έως 12kHz [35] και, σύμφωνα με το θεώρημα Nyquist, ο ρυθμός δειγματοληψίας πρέπει να είναι διπλάσιος της μέγιστης συχνότητας του σήματος. Η μετατροπή σε wav γίνεται κυρίως επειδή η βιβλιοθήκη matplotlib που εκτελεί STFT και εξάγει το φασματογράφημα από τον κάθε ήχο δε δουλεύει με αρχεία mp3.

```
34              #resample audiofile at 24kHz
35              resampled = librosa.core.resample(audio, orig_sr=sr, target_sr=SAMPLE_RATE)
36
37              sf.write(f"{name}.wav", resampled, samplerate=SAMPLE_RATE)
38              os.remove(filename)
```

Εδώ να σημειωθεί πως το κάθε αρχείο είναι ονομασμένο με το όνομα του πτηνού που ακούγεται σε αυτό, ακολουθούμενο από έναν τυχαίο αριθμό, παραδείγματος χάρη «Sonus-naturalis_178159.mp3». Αφού το αρχείο έχει δειγματοληφθεί επιτυχώς, το αρχικό .mp3 αρχείο διαγράφεται. Τα .wav αρχεία επιστρέφονται με τη μορφή λίστας, ενώ σε μία δεύτερη λίστα εντάσσεται το είδος του πτηνού που περιέχεται σε κάθε αρχείο. Οι δύο αυτές λίστες επιστρέφονται για χρήση από τα επόμενα στάδια.

```
45      wavs = os.listdir(path)
46      for filename in wavs:
47          genus = "-".join(filename.split("-", 2)[:2])
48          labels.append(genus)
49
50      return(wavs, labels)
```

II. SPLIT_WAV.PY

Σε αυτό το αρχείο κώδικα φορτώνεται το κάθε αρχείο .wav ξεχωριστά με χρήση ενός for loop. Οι μεταβλητές «name» και «id» κρατάνε την ονομασία του είδους και τον τυχαίο αριθμό αντίστοιχα, ενώ το αρχείο ήχου μετατρέπεται σε μορφή AudioSegment του πακέτου Pydub προκειμένου να εφαρμοστούν κάποιες συναρτήσεις της βιβλιοθήκης πάνω του.

```
18          name = "-".join(file.split("-", 2)[2:3])
19          id = name.split(".")[0]
20          song = AudioSegment.from_file(file)
```

Μόλις το αρχείο μετατραπεί σε μορφή AudioSegment, εφαρμόζεται πάνω του υψιπερατό φίλτρο συχνότητας 220Hz επειδή η πλειοψηφία των πτηνών κελαηδούν σε συχνότητες υψηλότερες από αυτήν [36]. Έγιναν δοκιμές εφαρμογής και χαμηλοπερατών φίλτρων με συχνότητα 8kHz, όμως αποδείχθηκε πως αφαιρούνταν σημαντική πληροφορία από τους ήχους και έτσι δε συμπεριλήφθηκε. Ακολουθεί κανονικοποίηση των ήχων στα 3dB, αριθμός που επιλέχθηκε ύστερα από δοκιμές. Μετά από την κανονικοποίηση, το κάθε αρχείο χωρίζεται σε ίσα τμήματα 5 δευτερολέπτων για να γίνει η αφαίρεση του θορύβου.

```
22          # Most bird vocalizations are above 220Hz, this filter attenuates noise #
23          song = high_pass_filter(song, 220, 5)
24          # amplitude of 3dB was selected after trials
25          song = match_target_amplitude(song, -3.0)
26          chunks = make_chunks(song, 5000)
```

Σειρά έχει ο υπολογισμός της ενέργειας και της ισχύος κάθε αρχείου, τιμές που θα χρειαστούν στη σύγκριση των τμημάτων και το διαχωρισμό εκείνων που περιέχουν πληροφορία από τα κενά.

```
28          # compute song energy and power for every recording to compare with chunks
29          song_array = song.get_array_of_samples()
30          song_array = np.array(song_array)
31          song_energy = np.sum(song_array.astype(float)**2)
32          song_power = song_energy / len(song)
```

Ένα nested for loop χρησιμοποιείται για να γίνει σύγκριση του κάθε κομματιού με το αρχείο από το οποίο προήλθε. Πολύ απλά, αν η ισχύς του κομματιού είναι μεγαλύτερη ή ίση της συνολικής ισχύος του αρχείου, κρίνεται πως το κομμάτι αυτό περιέχει κελάηδισμα και εξάγεται σε έναν τοπικό φάκελο, διατηρώντας το «name» και το «id», συν έναν ακόμα μοναδικό αριθμό

για την αποφυγή διπλότυπων αρχείων. Διαφορετικά, το κομμάτι αγνοείται. Αφού ελεγχθούν όλα τα κομμάτια από ένα αρχείο, αυτό διαγράφεται.

```
        for i, ch in enumerate(chunks):
            ch_array = ch.get_array_of_samples()
            ch_array = np.array(ch_array)
            ch_energy = np.sum(ch_array.astype(float)**2)
            ch_power = ch_energy / len(ch)

            # if chunks power is greater than average power of complete recording,
            # it contains useful bird song
            if (ch_power >= song_power and len(ch) == 5000):
                label_id = [labels[index], '_', id, '_', str(i)]
                label_id_str = "".join(label_id)
                chunk_path = [path, "/{0}.wav".format(label_id_str)]
                full_path = "".join(chunk_path)
                ch.export(full_path, format="wav")

        os.remove(file)
```

Στο τέλος αυτού του σταδίου το Dataset αποτελείται από 26440 εικόνες.

III. PREPROCESSING.PY

Εδώ πραγματοποιείται η μετατροπή των αρχείων ήχου - όσων έχουν φιλτραριστεί από το προηγούμενο στάδιο - σε φασματογραφήματα. Στην αρχή φορτώνονται οι συναρτήσεις των παραπάνω σταδίων και ελέγχουν αν υπάρχουν αρχεία που δεν έχουν μετατραπεί σε μορφή .wav. Αν υπάρχουν, οι συναρτήσεις εκτελούνται με τη σειρά, προετοιμάζοντας τα αρχεία για το μετασχηματισμό Fourier.

```
#Call functions only if new mp3 data has been entered
#in the directory
for file in os.listdir(data_path):
    if not file.endswith(".wav"):
        train_audio, train_labels = load_and_label(data_path)
        split(train_audio, train_labels, split_path)
```

Στη συνέχεια προσδιορίζονται οι τιμές των μεταβλητών που θα χρησιμοποιηθούν στο STFT. Αυτές είναι οι fft και hop_len και περιγράφουν το μήκος του σήματος μετά από την εφαρμογή συνάρτησης παραθύρου και τον αριθμό δειγμάτων ήχου μεταξύ γειτονικών στηλών STFT. Σύμφωνα με την τεκμηρίωση της Librosa, η τιμή fft = 512 είναι η προτεινόμενη για εφαρμογή σε επεξεργασία ομιλίας [37]. Το hop_len είναι από προεπιλογή ίσο με *win_length / 4*. Η μεταβλητή win_length στην περίπτωση μας δεν έχει προσδιοριστεί, οπότε παίρνει την τιμή του fft. Εξ 'ου και hop_len = 128. Η μεταβλητή win_length περιγράφει το μήκος του παραθύρου που εφαρμόζεται στο σήμα. Μικρότερες τιμές οδηγούν σε βελτιωμένη χρονική ανάλυση, ενώ

μεγαλύτερες τιμές βελτιώνουν την ανάλυση της συχνότητας [37]. Εδώ επιλέχθηκε η μέση τιμή 512. Παράλληλα ο ήχος φορτώνεται σε μορφή AudioSegment για να εφαρμοστούν οι σχετικές συναρτήσεις.

```
32          fft = 512
33          # Determines time scale in x-axis
34          hop_len = 128
35
36          sound = AudioSegment.from_file(file)
37          samples = sound.get_array_of_samples()
```

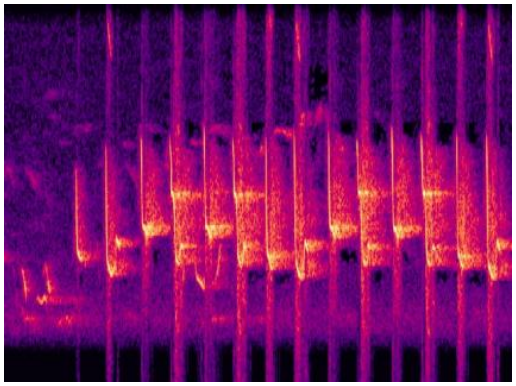
Με τις απαραίτητες μεταβλητές έτοιμες, γίνεται μία κανονικοποίηση του ήχου σε μορφή numpy array, καθώς είναι η απαιτούμενη μορφή από τη συνάρτηση librosa.stft(). Γίνεται ο μετασχηματισμός Fourier και ύστερα κατασκευάζεται το φασματογράφημα με τη συνάρτηση librosa.amplitude_to_db(). Υπάρχουν αρκετές επιλογές παραθύρων για το μετασχηματισμό Fourier, με τα πιο αρμόζοντα να είναι τα παράθυρα «Blackman» και «Kaiser». Στη συνάρτηση amplitude_to_db () μπορεί να προσδιορισθεί και ο χρωματικός χάρτης (colormap) του φασματογραφήματος. Αυτό αναφέρεται διότι ακόμα και η επιλογή του χάρτη επηρεάζει την εκπαίδευση του μοντέλου, όπως θα δούμε και στη συνέχεια.

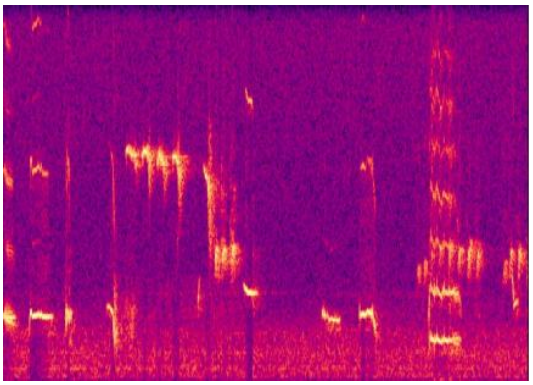
```
39          # normalize samples
40          snd_arr = np.array(samples).astype(np.float32)/32768
41
42          # Apply STFT with the selected parameters, best windows are blackman and kaiser
43          fourier = np.abs(librosa.stft(snd_arr, n_fft=fft, hop_length=hop_len, window='blackman'))
44          spec = librosa.amplitude_to_db(fourier, ref=np.max)
```

Τέλος, τα φασματογραφήματα εμφανίζονται στην οθόνη και αποθηκεύονται σε τοπικό φάκελο και σε μορφή εικόνας .png μέσω της βιβλιοθήκης matplotlib. Το ηχητικό αρχείο διαγράφεται καθώς έχει τελέσει το σκοπό του.

```
46          fig = plt.figure()
47          fig.add_subplot(111)
48
49          # Default colormap is the best one in terms of model accuracy
50          librosa.display.specshow(spec)
51
52          spec_path = [data_path, os.path.splitext(file)[0]]
53          spec_path = "".join(spec_path)
54
55          plt.subplots_adjust(top=1, bottom=0, right=1, left=0,
56                              hspace=0, wspace=0)
57
58          plt.savefig("{}".format(spec_path, dpi=500, bbox_inches='tight', pad_inches=0, format='png'))
59
60          plt.close(fig)
61
62          os.remove(file)
```

Εδώ παρουσιάζονται μερικά φασματογραφήματα τα οποία κατασκευάστηκαν με την παραπάνω μέθοδο. Ο χρωματικός χάρτης είναι ο προεπιλεγμένος της βιβλιοθήκης matplotlib και οι διαστάσεις των αρχείων είναι 640x480.





IV. PNG2JPG.PY

Αυτό το βήμα προστέθηκε για να συμπιέσει και να μειώσει τις διαστάσεις των αρχείων των φασματογραφημάτων, ώστε να μειωθεί σημαντικά και η κατανάλωση μνήμης στην εκπαίδευση του μοντέλου. Με τη σειρά:

Φορτώνονται τα αρχεία .png μέσω for loop και αφαιρείται από το καθένα από αυτά το κανάλι Α από το RGBA, το οποίο αντιστοιχεί στο Alpha ή αλλιώς τη διαφάνεια του αρχείου και είναι χαρακτηριστικό των png εικόνων. Αυτό συμβαίνει για να μετατραπεί το αρχείο σε μορφή jpg, η οποία δε διαθέτει το κανάλι Α. Να σημειωθεί πως η μετατροπή από png σε jpg γίνεται με τη χρήση συναρτήσεων της βιβλιοθήκης OpenCV και η μορφή των εικόνων, είτε αυτή είναι png είτε jpg δεν επηρεάζει την εκπαίδευση του μοντέλου. Ο κύριος λόγος μετατροπής είναι το μικρότερο μέγεθος αρχείων που προσφέρει η μορφή jpg.

```
15      for spectrogram in tqdm(spectrograms, desc='Resizing images'):
16          img_path = os.path.join(directory, spectrogram)
17          img = cv2.imread(img_path)
18
19          # PNG images have 4 channels (RGBA), so to convert them
20          #to JPEG we must remove the 4th channel(A:transparency)
21          rgb = cv2.cvtColor(img, cv2.COLOR_RGBA2RGB)
```

Στη συνέχεια οι διαστάσεις της εικόνας μειώνονται από το αρχικό 640x480 σε 224x168, για να επισπευσθεί η εκπαίδευση του μοντέλου χωρίς την απώλεια ακρίβειας. Οι συγκεκριμένες διαστάσεις επιλέχθηκαν προκειμένου να διατηρηθεί το aspect ratio της εικόνας, καθώς άλλες επιλογές θα παραμόρφωναν την εικόνα και θα οδηγούσαν σε εξ ολοκλήρου λάθος εκπαίδευση.

```
23          # Resize images accordingly to preserve
24          # aspect ratio and information
25          rgb = cv2.resize(rgb, (224, 168), interpolation=cv2.INTER_AREA)
26
27          jpg_path = [directory, os.path.splitext(spectrogram)[0], '.jpg']
28          jpg_path = "".join(jpg_path)
```

Τελευταία ακολουθεί αποθήκευση της εικόνας jpg στην οποία γίνεται μείωση της ποιότητας στο 75% όπως φαίνεται και στην εικόνα. Αυτή η πράξη επιτυγχάνει ακόμα μικρότερο μέγεθος αρχείου και μάλιστα όχι εις βάρος της ποιότητας της εικόνας, αφού η διαφορά είναι μηδαμινή. Μετά την αποθήκευση της εικόνας, το κάθε αρχείο png διαγράφεται από το σύστημα.

```
27          jpg_path = [directory, os.path.splitext(spectrogram)[0], '.jpg']
28          jpg_path = "".join(jpg_path)
29
30          # A quality factor of 75 reduces the file size without having
31          # any particular effect on image quality
32          cv2.imwrite(jpg_path, rgb, [int(cv2.IMWRITE_JPEG_QUALITY), 75])
33          os.remove(spectrogram)
```

Ενδεικτικά, το μέγεθος της αρχικής png εικόνας ήταν 920kB. Μετά τη μετατροπή σε jpg, το μέγεθος κατέβηκε στα 112kB.

Με το παραπάνω αρχείο τελειώνει και η προ επεξεργασία των δεδομένων του dataset προτού αυτά εισαχθούν στο μοντέλο για εκπαίδευση. Στις παραγράφους που ακολουθούν θα γίνει ανάλυση των τοπολογιών CNN που δοκιμάστηκαν, καταλήγοντας στο τελικό μοντέλο που χρησιμοποιείται στην εφαρμογή.

2.4 Κατασκευή Αρχικών Μοντέλων

Το Dataset χωρίστηκε σε 2 τμήματα, τα train_data και test_data με το καθένα να έχει το δικό του μονοπάτι στο σύστημα. Τα train_data αποτελούνταν από το 85% του dataset, δηλαδή 22,474

εικόνες και τα test_data από τις υπόλοιπες 3,966. Ο διαχωρισμός αυτός γίνεται ώστε το μοντέλο να εκπαιδευτεί με τα train_data και μετά να ελεγχθεί η απόδοση του στα test_data, τα οποία δεν έχει δει και έτσι δεν μπορεί να κάνει προκατειλημμένες προβλέψεις.

Αρχικά έγιναν δοκιμές εκπαίδευσης ενός απλού σχετικά μοντέλου CNN στο οποίο θα τροφοδοτούνταν απευθείας όλα τα δεδομένα του dataset σε numpy arrays. Σύντομα έγινε αντιληπτό πως λόγω του μεγάλου μεγέθους του dataset η απευθείας τροφοδότηση των δεδομένων δεν ήταν δυνατή και η RAM του συστήματος δεν μπορούσε να την υποστηρίξει.

Ακολούθησε συγγραφή 2 βοηθητικών προγραμμάτων, των load_specs_2D.py και load_specs_3D.py στα οποία ανατέθηκε η φόρτωση των δεδομένων του dataset με βάση τη λιγότερη δυνατή χρήση μνήμης RAM, ώστε να επιτευχθεί η εκπαίδευση του μοντέλου. Το load_specs_2D φόρτωνε τα φασματογραφήματα σε ασπρόμαυρη μορφή, με τη βοήθεια του αρχείου png2jpg.py, ενώ το load_specs_3D τα φόρτωνε στην προβλεπόμενη RGB μορφή.

Στο load_specs_2D έγινε χρήση συμπιεσμένων αραιών πινάκων (compressed sparse matrices - CSRs), οι οποίοι, λαμβάνοντας ένα διάνυσμα με τις θέσεις των μαύρων pixels στο φασματογράφημα, αγνοούσαν τα συγκεκριμένα pixels και δεν τα φόρτωναν. Ύστερα, οι κατασκευασμένες εικόνες φορτώνονταν σε κανονικοποιημένους πίνακες numpy float16 για περαιτέρω μείωση του μεγέθους τους.

```
    for spectrogram in os.listdir(directory):
        img_path = os.path.join(directory, spectrogram)
        img = Image.open(img_path)
        img = np.array(img)

        # return numpy array with rows and columns containing black pixels
        row = np.nonzero(img==0)[0]
        col = np.nonzero(img==0)[1]

        # construct 2D array with black pixels and their positions
        cellValue = np.array([img[i][j] for i,j in zip(row, col)])

        # create compressed sparse row matrix
        img_sparse = csr_matrix((cellValue, (row, col)), shape=(img.shape[0], img.shape[1]))
        img_sparse = img_sparse.toarray()

        images.append(img_sparse)
        labels.append(spectrogram.split('_', 1)[0])


    train_images, train_labels = shuffle(images, labels)

    # preprocess data for training
    train_images = np.array(train_images, dtype='float16')/255.0
    train_images = train_images.reshape(train_images.shape[0], 168, 224, 1)
    train_labels = np.array(train_labels)
```

Αντίστοιχα, στο load_specs_3D προσπάθησε να γίνει άνοιγμα των φασματογραφημάτων σε πίνακες numpy τύπου float16, αντί του προεπιλεγμένου float32 και περαιτέρω φόρτωση σε κανονικοποιημένο πίνακα numpy float16 για τροφοδότηση στο μοντέλο.

```
    for spectrogram in os.listdir(directory):
        img_path = os.path.join(directory, spectrogram)
        img = cv2.imread(img_path)
        img = np.array(img, 'float16')
        images.append(img)
        labels.append(spectrogram.split('_', 1)[0])

    train_images, train_labels = shuffle(images, labels)

    train_images = np.array(train_images, dtype='float16')/255.0
    train_images = train_images.reshape(train_images.shape[0], 168, 224, 3)
    train_labels = np.array(train_labels)
```

Παρά τις προσπάθειες, οι παραπάνω ενέργειες δεν κατάφεραν να μειώσουν τη χρήση μνήμης αρκετά ώστε να γίνει η εκπαίδευση του μοντέλου. Ακόμα όμως και αν το κατάφερναν, συγκεκριμένα το πιο υποσχόμενο load_specs_2D, η ακρίβεια του μοντέλου θα μειωνόταν αισθητά, όπως ισχύει για τα ασπρόμαυρα φασματογραφήματα [32].

2.4.1 – Η χρήση DataFrames

Από τα παραπάνω έγινε εμφανές πως υπήρχε ανάγκη για μια διαφορετική μέθοδο φόρτωσης του Dataset που δε θα επιβάρυνε τόσο σοβαρά τη μνήμη RAM. Η μέθοδος που επιλέχθηκε για αυτό ήταν η χρήση ImageDataGenerators, γεννητριών που πέρα από τη σύνθεση νέων δεδομένων από ήδη υπάρχοντα, επιτρέπουν την τμηματική φόρτωση και εκπαίδευση τους, γνωστή και ως batch training. Το batch training εφαρμόζεται όταν τα δεδομένα είναι μεγάλα και η εκτέλεση αλγορίθμων που απαιτεί το πλήρες σύνολο τους αποδεικνύεται ακριβές. Αναλυτικότερα, ο κώδικας του μοντέλου είναι ο ακόλουθος.

Αρχικά, ελέγχεται αν έχει γίνει μετατροπή όλων των αρχείων png σε jpg και, αν δεν έχει γίνει, εκτελείται η συνάρτηση του png2jpg.py. Τα jpg αρχεία φορτώνονται σε λίστα μαζί με τις αντίστοιχες ετικέτες τους (το είδος πτηνού κάθε αρχείου), προκειμένου να μετατραπούν σε Pandas DataFrames.

```
# Check ALL data for compression
for file in os.listdir(data_path):
    if not file.endswith(".jpg"):
        compress(data_path)
    break

filenames = os.listdir(data_path)
categories = []
for filename in filenames:
    category = filename.split('_', 1)[0]
    categories.append(category)
```

Τα DataFrames είναι δισδιάστατες δομές δεδομένων και μοιάζουν με 2D πίνακες με σειρές και στήλες. Η μετατροπή των δεδομένων σε αυτή τη μορφή είναι υποχρεωτική για να γίνει η εκπαίδευση του μοντέλου με ImageDataGenerators. Στην εικόνα που ακολουθεί κατασκευάζεται ένα DataFrame από τις 2 λίστες, όπου η μεταβλητή 'image' αντιστοιχεί στον άξονα x και η 'label' στον άξονα y. Το DataFrame χωρίζεται στο train_df για την εκπαίδευση και το validate_df για έλεγχο της απόδοσης του μοντέλου, όπως ορίζουν οι σωστές πρακτικές εκπαίδευσης μοντέλων μηχανικής μάθησης. Το μέγεθος τους είναι αντίστοιχα το 80% και το 20% του dataset. Ξεχωριστό DataFrame χτίστηκε και για τα test_data.

```
df = pd.DataFrame({
    'image':filenames,
    'label':categories
})

train_df, validate_df = train_test_split(df, test_size=0.2, stratify=df['label'])
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
```

Να επισημανθεί πως η παράμετρος stratify της συνάρτησης train_test_split () έχει ως στόχο τη διατήρηση της ισορροπίας στα τμήματα δεδομένων, ειδικότερα την ίση κάλυψη όλων των ετικετών, για να αποφευχθεί το ενδεχόμενο εκμάθησης λάθος βαρών και ως συνέπεια, λανθασμένης εκπαίδευσης.

Ακολούθως, ορίζονται παράμετροι που διατηρούν τιμές των DataFrames απαραίτητες για την εκπαίδευση. Τέτοιες είναι το μέγεθος των 2 DataFrames, ο αριθμός των batches και οι διαστάσεις των φασματογραφημάτων.

```
total_train = train_df.shape[0]
total_valid = validate_df.shape[0]
batch_size = 32
IMG_SIZE = (168, 224)
```

Σειρά έχει η αρχικοποίηση των ImageDataGenerators. Όπως φαίνεται από το παρακάτω απόσπασμα, για το κάθε DataFrame πραγματοποιείται κανονικοποίηση και ορίζονται οι σχετικές παράμετροι. Συγκεκριμένα, συν των παραπάνω, ορίζονται το όνομα του κάθε DataFrame, το μονοπάτι στο σύστημα που τροφοδοτεί τα δεδομένα, οι πίνακες που αναλογούν στον κάθε άξονα, το χρώμα των εικόνων, το είδος ταξινόμησης (δυαδική ή κατηγορική) και ένα σήμα που ελέγχει την ανάμιξη ή μη των δεδομένων, για αποδοτικότερη εκπαίδευση.

```
# Training Generator
train_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    data_path, #train_path,
    x_col='image',
    y_col='label',
    target_size=IMG_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=batch_size,
    shuffle=True
)
```

Επόμενο κομμάτι στον κώδικα είναι η τοπολογία του μοντέλου. Στο σημείο αυτό γίνονταν τακτικές αλλαγές και πειραματισμοί μέχρι να βρεθεί η τοπολογία που προσφέρει την καλύτερη εκπαίδευση, Ενδεικτικά, ο κώδικας έχει την εξής μορφή:

```
# Build Model
model = Sequential()

model.add(Conv2D(32, (3,3), input_shape=(168, 224, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(2,2, padding='same'))

model.add(Conv2D(64, (3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(2,2, padding='same'))

model.add(Conv2D(128, (3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(2,2, padding='same'))

model.add(Flatten())
model.add(Dense(50, activation='softmax'))
```

Πριν προχωρήσουμε στην εκπαίδευση του μοντέλου, ορίζονται ο ρυθμός εκμάθησης του μοντέλου, η συνάρτηση υπολογισμού της απώλειας του και πραγματοποιείται η μεταγλώττιση του. Αυτές οι παράμετροι επηρεάζουν σημαντικά την εκπαίδευση και η επιλογή τους είναι κρίσιμη.

```
#Compile Model
opt = Adam(lr=0.001)
model.compile(optimizer=opt,
loss='categorical_crossentropy', metrics=['accuracy'])
```

Ταυτόχρονα, εισάγονται κάποια callbacks, μέθοδοι που ελέγχουν και επεμβαίνουν στην εκπαίδευση όποτε χρειάζεται. Η early_stop είναι υπεύθυνη για διακοπή της εκπαίδευσης εάν παρατηρήσει πως το μοντέλο αποκλίνει ύστερα από συγκεκριμένο αριθμό εποχών, ενώ η lr_redux υποπολλαπλασιάζει το ρυθμό εκπαίδευσης στην περίπτωση έλλειψης προόδου, ενέργεια που μπορεί να προσφέρει σημαντική ώθηση στην εκάστοτε εκπαίδευση.

```
#Callbacks
early_stop = EarlyStopping(monitor='val_loss', patience=4)
lr_redux = ReduceLROnPlateau(monitor='val_loss', patience=3,
                             verbose=1,
                             factor=0.3,
                             min_lr=9e-5)

callbacks = [early_stop, lr_redux]
```

Τέλος, καλείται η συνάρτηση tensorflow.keras.fit () με την οποία ξεκινάει η εκπαίδευση.

```
185  #Fit Model
186  history = model.fit(train_generator,
187                      epochs=20,
188                      steps_per_epoch = total_train//batch_size,
189                      validation_data=validation_generator,
190                      validation_steps=total_valid//batch_size,
191                      verbose=2,
192                      callbacks=callbacks)
```

Στη συγκεκριμένη περίπτωση το μοντέλο εκπαιδεύεται για 10 εποχές ή γύρους. Ακολουθεί ο υπολογισμός της ακρίβειας και απώλειας του μοντέλου στα test_data και η αποθήκευση του εκάστοτε μοντέλου.

```
195  test_loss, test_acc = model.evaluate(test_generator, steps=total_test//batch_size)
196
197  print("Test loss", test_loss)
198  print("Test accuracy", test_acc*100)
199
200  model.save(f"first_model.h5")
```

Έχοντας αναλύσει τη μορφή και τη διαδικασία εκπαίδευσης ενός βασικού μοντέλου, σειρά έχει η ανάλυση των αποτελεσμάτων κάποιων αρχικών τοπολογιών.

Πρώτα παρουσιάζονται τα αποτελέσματα από την εκπαίδευση ενός απλού μοντέλου με την προ ηγηθείσα τοπολογία. Ειδικότερα, το μοντέλο αποτελείται από 3 συνελικτικά στρώματα, ακολουθούμενα από στρώματα που εκτελούν BatchNormalization (), MaxPooling () και τέλος από ένα πλήρως συνδεδεμένο στρώμα στην έξοδο. Ο αριθμός των batches ανέρχεται στα 32.

Να σημειωθεί πως οι μεταβλητές «Loss», «Acc» αντιστοιχούν στα Training Accuracy και Training Loss του train_df, ενώ οι «Val_loss», «Val_acc» αντιστοιχούν στα Validation Accuracy και Validation Loss του validate_df.

Epoch	Loss	Acc	Val_loss	Val_acc
1	7.32	41%	9.99	9.60%
2	6.68	59%	7.21	44.70%
3	6.57	66%	7.25	47.30%
4	6.51	71%	7.02	49.60%
5	6.47	74%	7.09	49%
6	6.45	76%	7.05	50%
7	6.46	77%	7.28	47.40%
8	6.42	78%	6.92	52.40%
9	6.38	81%	6.91	52.60%
10	6.39	82%	6.89	52.90%

Εικόνα 2.5 - Αποτελέσματα εκπαίδευσης με Jet χρωματικό χάρτη

Epoch	Loss	Acc	Val_loss	Val_acc
1	5.68	52%	10.42	5.90%
2	5.62	55%	5.93	44.90%
3	5.14	62%	5.64	53.50%
4	5.1	65%	5.31	57.50%
5	4.99	67%	5.45	56%
6	4.94	68%	5.42	57%
7	4.95	69%	5.63	55.40%
8	4.88	72%	5.22	60.50%
9	4.83	71%	5.2	61.00%
10	4.84	71%	5.19	61.30%

Εικόνα 2.6 - Αποτελέσματα εκπαίδευσης με τον προεπιλεγμένο χρωματικό χάρτη

Αρχικά εξετάστηκε η διαφορά στην εκπαίδευση δεδομένων με διαφορετικούς χρωματικοί χάρτες. Είναι φανερό, όπως αναφέρθηκε και προηγουμένως, πως η επιλογή του χρωματικός χάρτης επιδρά αισθητά στην εκπαίδευση του μοντέλου. Παρά την υψηλότερη ακρίβεια εκπαίδευσης που παρουσιάζει ο χρωματικός χάρτης Jet, ο προεπιλεγμένος χρωματικός χάρτης φαίνεται να επιτυγχάνει καλύτερη συνολική πρόοδο, όπως φαίνεται από τις χαμηλότερες απώλειες και την σχετικά υψηλότερη επικυρωμένη ακρίβεια, το λεγόμενο validation accuracy. Εφεξής σε όλες τις τοπολογίες θα χρησιμοποιείται ο προεπιλεγμένος χρωματικός χάρτης.

Στη συνέχεια δοκιμάστηκε η εισαγωγή ενός στρώματος Dropout πριν από το πλήρως συνδεδεμένο στρώμα, με συντελεστή 0.3. Το Dropout απενεργοποιεί τυχαία βάρη του μοντέλου, αναγκάζοντας το να συνεχίσει την εκπαίδευση με μειωμένους πόρους.

Χρησιμοποιείται συνήθως για να μειώσει την πολυπλοκότητα του μοντέλου και για να γίνει αποφυγή του overfitting, κατά το οποίο το μοντέλο αποδίδει πολύ καλά στην εκπαίδευση, αλλά λόγω προκατειλημμένων βαρών, αποδίδει πολύ φτωχότερα σε άγνωστα δεδομένα.

Epoch	Loss	Acc	Val_loss	Val_acc
1	3.93	41%	8.54	3.00%
2	3.67	59%	3.9	52.00%
3	3.48	66%	3.65	60.90%
4	3.29	71%	3.61	62.30%
5	3.24	74%	3.49	65%
6	3.2	76%	3.61	64.00%
7	3.19	77%	3.7	63.00%
8	3.19	78%	3.7	63.00%
9	3.15	81%	3.31	70.00%
10	3.14	82%	3.31	69.87%

Εικόνα 2.7 - Αποτελέσματα εκπαίδευσης με Dropout (0.3)

Σύμφωνα με τα αποτελέσματα, η εισαγωγή του στρώματος Dropout κατάφερε να βελτιώσει αρκούντως το validation accuracy και την απώλεια του μοντέλου. Επομένως, έγινε και αυτό μέρος του μοντέλου.

Μετά και από την ενσωμάτωση του Dropout, αποφασίστηκε να γίνει πειραματισμός με διαφορετικές διαστάσεις στα kernel των MaxPooling στρωμάτων. Το kernel αναγνωρίζεται ως η μικρή παρένθεση με τους 2 αριθμούς οι οποίοι αντιστοιχούν σε διαστάσεις στους άξονες. Λειτουργεί ως ανιχνευτής χαρακτηριστικών (feature detector) για το μοντέλο και ουσιαστικά είναι ένα τετραγωνικό φίλτρο που ολισθαίνει πάνω στην εικόνα, ψάχνοντας χρήσιμα χαρακτηριστικά για την εκπαίδευση. Επισημαίνεται πως όσο αυξάνεται ο αριθμός των

διαστάσεων βελτιώνεται η ακρίβεια του μοντέλου, εις βάρος όμως της ταχύτητας. Στα παραπάνω μοντέλα το kernel είχε διαστάσεις (2, 2) και εδώ αυξήθηκαν στο (5, 5).

Epoch	Loss	Acc	Val_loss	Val_acc
1	2.45	59%	7.88	2.00%
2	2.37	63%	2.39	63.80%
3	2.12	72%	2.31	67.66%
4	2.14	78%	2.25	69.95%
5	2.03	80%	2.21	71%
6	1.97	83%	2.17	72%
7	1.87	85%	2.24	72.41%
8	1.72	87%	2.13	74.29%
9	1.76	87%	2.24	72.68%
10	1.69	88%	2.2	73.93%

Εικόνα 2.8 - Αποτελέσματα εκπαίδευσης με kernel 5x5

Τα αποτελέσματα της αύξησης των διαστάσεων του kernel αποδείχθηκαν πολύ θετικά, με βελτίωση στην καθεμία από τις παραμέτρους της εκπαίδευσης. Ωστόσο, η ταχύτητα εκπαίδευσης για το μοντέλο έφτασε σε εξουθενωτικά χαμηλά επίπεδα. Για παράδειγμα, ενώ αρχικά η κάθε εποχή χρειαζόταν 50 λεπτά για να ολοκληρωθεί, στη συγκεκριμένη περίπτωση επιβαρύνθηκε με επιπλέον 30 λεπτά. Σε βάθος 10 εποχών αυτή η επιβάρυνση μεταφράζεται σε 5 επιπρόσθετες ώρες εκπαίδευσης.

Συν τοις άλλοις, υποστηρίζεται πως η αύξηση των διαστάσεων του φίλτρου καθιστά την ανίχνευση λεπτομερέστερων χαρακτηριστικών δυσκολότερη έως αδύνατη. Αυτός ο παράγοντας δύναται να προκαλέσει δυσχέρειες στην πρόβλεψη πτηνών με τέτοια χαρακτηριστικά. Έτσι, επιλέχθηκε να μη γίνει χρήση μεγαλύτερων φίλτρων και έγινε αναζήτηση πιο βιώσιμων εναλλακτικών.

Η αναζήτηση μεθόδων για βελτίωση της εκπαίδευσης οδήγησαν στην προσθήκη ενός επιπλέον κρυφού στρώματος στο μοντέλο. Η ενέργεια αυτή επιλέχθηκε, επειδή βλέποντας τα παραπάνω αποτελέσματα, συμπεραίνεται πως το μοντέλο συνεχίζει να μαθαίνει και να

εκπαιδεύεται μέχρι και το πέρας της εκπαίδευσης, χωρίς να παρουσιάζει στοιχεία κορεσμού. Η εισαγωγή νέων κρυφών στρωμάτων δίνει στο μοντέλο τη δυνατότητα να μάθει περισσότερο και καλύτερα.

Epoch	Loss	Acc	Val_loss	Val_acc
1	4.17	42.00%	8.34	1.00%
2	1.39	63.62%	1.39	63.49%
3	0.92	73%	0.89	76%
4	0.8	79.76%	0.86	76.39%
5	0.62	83.44%	0.79	79.15%
6	0.52	87.25%	0.83	78.81%
7	0.49	89.70%	0.66	83.00%
8	0.46	92.10%	0.62	83.42%
9	0.44	93.40%	0.57	85.67%
10	0.43	94.50%	0.62	82.93%

Εικόνα 2.9 - Αποτελέσματα εκπαίδευσης με 4 κρυφά στρώματα

Όπως ήταν αναμενόμενο, η προσθήκη 4ου κρυφού στρώματος ανέβασε την εκπαίδευση σε νέα ύψη, συνολικά αλλά και ξεχωριστά για κάθε παράμετρο. Πόσο μάλλον, το μοντέλο συνεχίζει να παρουσιάζει περιθώρια βελτίωσης πριν την ολοκλήρωση των 10 εποχών εκπαίδευσης.

Συμπερασματικά, σαν επόμενο βήμα ακολούθησε η αύξηση της εκπαίδευσης από 10 σε 20 εποχές, ώστε να εμφανιστούν τα πραγματικά όρια του μοντέλου.

Epoch	Loss	Acc	Val_loss	Val_acc
1	1.24	49%	5.77	1.00%
2	1.12	56%	1.31	65.00%
3	0.89	66%	1.04	71.68%
4	0.77	73%	0.86	76.75%
5	0.74	77%	0.77	79%
6	0.66	81%	0.74	81%
7	0.54	83%	0.64	82.50%
8	0.47	86%	0.59	83.10%
9	0.44	88%	0.68	81.36%
10	0.42	89%	0.6	84.20%
11	0.38	91%	0.51	86.00%
12	0.37	92%	0.52	85.70%
13	0.34	93%	0.52	85.84%
14	0.36	94%	0.48	86.88%
15	0.33	94%	0.52	86.20%
16	0.3	95%	0.51	86.20%
17	0.28	96%	0.43	88.74%
18	0.25	96%	0.46	87.33%
19	0.27	97%	0.48	87.60%
20	0.26	97%	0.41	89.16%

Εικόνα 2.10 - Αποτελέσματα εκπαίδευσης με 20 εποχές

Ακόμα και στις 20 εποχές, το μοντέλο συνεχίζει να εκπαιδεύεται συνεχώς και με αυξημένη ακρίβεια. Η διαφορές στην απώλεια και την ακρίβεια του μοντέλου ανάμεσα στα 2 διαφορετικά DataFrames είναι προσδοκώμενες και δεν είναι απαραίτητα κακοί δείκτες για την εκπαίδευση. Συνήθως οφείλονται στο μικρότερο μέγεθος και τη μικρότερη ποικιλία του σετ επικύρωσης δεδομένων. Ενδέχεται όμως και να οφείλονται σε πιθανό overfitting, κάτι το οποίο μπορεί να διορθωθεί με την προσθήκη περαιτέρω στρωμάτων Dropout ή L2 Regularizers, μεταξύ άλλων.

Εν συνεχεία, έγιναν δοκιμές μοντέλων που περιείχαν ένα επιπλέον πλήρως συνδεδεμένο στρώμα πριν το τελικό που είναι υπεύθυνο για τις προβλέψεις. Η προσθήκη αυτών των στρωμάτων εμβαθύνει το μοντέλο και σε πολλές περιπτώσεις οδηγεί σε καλύτερες προβλέψεις. Η χρήση >1 πλήρως συνδεδεμένων στρωμάτων συστήνεται και από αρκετούς Data Scientists.

Epoch	Loss	Acc	Val_loss	Val_acc
1	1.7	57%	2.7	33.00%
2	1.23	68%	1.77	52.60%
3	0.97	74%	1.39	62.00%
4	0.76	79%	1.13	68.60%
5	0.72	80%	0.96	73%
6	0.63	83%	0.82	77%
7	0.62	82%	0.73	79.00%
8	0.5	86%	0.63	81.60%

Εικόνα 2.11 - Αποτελέσματα εκπαίδευσης με Dense Layer 256 νευρώνων

Epoch	Loss	Acc	Val_loss	Val_acc
1	2.12	49%	3.07	24.00%
2	1.66	57%	2.27	41.30%
3	1.28	68%	1.85	50.60%
4	1.12	72%	1.58	57.10%
5	1.01	74%	1.37	62%
6	0.86	78%	1.22	66%
7	0.84	79%	1.11	68.20%
8	0.69	82%	0.99	71.60%

Εικόνα 2.12 - Αποτελέσματα εκπαίδευσης με Dense Layer 512 νευρώνων

Τα αποτελέσματα είναι θετικά για την περίπτωση των 256 νευρώνων, όχι όμως και για τους 512. Όπως φαίνεται και στην Εικόνα 15, η απόκλιση μεταξύ της ακρίβειας και της απώλειας έχει μειωθεί, γεγονός που δείχνει πως το ενδεχόμενο overfitting έχει αντιμετωπισθεί. Συγκρίνοντας τις εικόνες 13 και 15 παρατηρούμε πως διαφέρουν μόνο στο Training Accuracy, κάτι που ενισχύει την πεποίθηση περί αντιμετώπισης του overfitting.

Πέρα από τη χειρωνακτική αλλαγή τοπολογιών και γενικότερων πειραματισμών εξετάσθηκαν τα ενδεχόμενα κατασκευής μοντέλων μέσω Hyperparameter Tuning [38] και Transfer Learning [39].

Αναφορικά, το Hyperparameter tuning είναι μία μέθοδος κατά την οποία τροφοδοτούνται από το χρήστη διάφορες τιμές παραμέτρων βασικών για το μοντέλο. Ο χρήστης μπορεί να ορίσει πίνακες με τιμές για τον αριθμό των batches, τον αριθμό και το μέγεθος των στρωμάτων, το ρυθμό εκμάθησης, μεταξύ άλλων. Ύστερα, το μοντέλο δοκιμάζει όλους τους πιθανούς συνδυασμούς των παραπάνω παραμέτρων, παρουσιάζοντας στο τέλος την απόδοση τους και κάνοντας την επιλογή ενός μοντέλου εύκολη υπόθεση. Το Transfer Learning εστιάζει στη χρήση ισχυρών προ εκπαιδευμένων μοντέλων για εκπαίδευση και πρόβλεψη πάνω σε δεδομένα στα οποία τα μοντέλα αυτά δεν έχουν εκπαιδευθεί στο παρελθόν. Γνωστά μοντέλα που χρησιμοποιούνται συχνά σε αυτήν τη μέθοδο είναι τα MobileNet, ResNet, Inception.

Δυστυχώς, δεν υπήρξε η δυνατότητα εφαρμογής αυτών των μεθόδων διότι αποδείχθηκαν εξαιρετικά χρονοβόρες με τις δυνατότητες του διαθέσιμου συστήματος.

2.4.2 – Η εφαρμογή cross-validation

Εξετάζοντας τα δεδομένα του Dataset, παρατηρήθηκε πολύ σημαντική διαφορά ανάμεσα στο πλήθος εικόνων για διαφορετικά είδη. Το πλήθος δεδομένων για τα 3 συχνότερα είδη ήταν:

- 2830 (Sonus Naturalis)
- 1294 (Acrocephalus Dumetorum)
- 1086 (Luscinia Megarynchos)

Για τα 3 σπανιότερα είδη, οι αριθμοί αυτοί ήταν:

- 334 (Emberiza Schoeniclus)
- 350 (Parus Major)

- 356 (Carpodacus Erythrinus)

Το Dataset, πέρα από μικρό (μέσος όρος 620 δείγματα για 50 κλάσεις) ήταν μη ισορροπημένο, που σημαίνει πως το μοντέλο χρειάζεται διαφορετική μέθοδο εκπαίδευσης από αυτήν που ακολουθήθηκε έως τώρα, διότι οι υπάρχουσες παράμετροι εκπαίδευσης είναι αξιόπιστες μόνο για ισορροπημένα και περιεκτικά Datasets. Η μέθοδος που εφαρμόστηκε ήταν το stratified k-fold cross-validation, υποκατηγορία του cross-validation γνωστού και ως Διασταυρωμένη Επικύρωση.

Η διασταυρούμενη επικύρωση είναι μια στατιστική μέθοδος που χρησιμοποιείται για την εκτίμηση της ικανότητας μοντέλων μηχανικής μάθησης σε περιορισμένα δείγματα δεδομένων [40].

Χρησιμοποιείται συνήθως στην εφαρμοσμένη μηχανική μάθηση για τη σύγκριση και την επιλογή ενός μοντέλου για ένα δεδομένο πρόβλημα μοντελοποίησης πρόβλεψης, επειδή είναι εύκολο να κατανοηθεί, να εφαρμοστεί εύκολα και οδηγεί σε εκτιμήσεις δεξιοτήτων που έχουν γενικά χαμηλότερη προκατάληψη από άλλες μεθόδους [41].

Η διαδικασία έχει μια ενιαία παράμετρο που ονομάζεται k και αναφέρεται στον αριθμό των ομάδων στις οποίες πρόκειται να χωριστεί ένα δεδομένο δείγμα δεδομένων. Ως εκ τούτου, η διαδικασία ονομάζεται συχνά k-fold cross-validation. Όταν επιλέγεται μια συγκεκριμένη τιμή για το k, μπορεί να χρησιμοποιηθεί στη θέση του k στην αναφορά στο μοντέλο, όπως το k=10 να γίνει δεκαπλάσια διασταυρούμενη επικύρωση. Οι πρώτες αναδιπλώσεις k-1 χρησιμοποιούνται για την εκπαίδευση ενός μοντέλου, και η τελευταία χρησιμοποιείται ως το σετ δοκιμής. Αυτή η διαδικασία επαναλαμβάνεται και σε κάθε πτυχή δίνεται η ευκαιρία να χρησιμοποιηθεί ως σετ δοκιμής κράτησης. Ένα σύνολο k μοντέλων ταιριάζουν και αξιολογούνται και η απόδοση του μοντέλου υπολογίζεται ως ο μέσος όρος αυτών των εκτελέσεων [42].

Ενώ η Διασταυρωμένη επικύρωση αντιμετωπίζει το πρόβλημα προκατειλημμένων προβλέψεων σε μικρά και ισορροπημένα Datasets, το stratified k-fold cross-validation την καθιστά κατάλληλη μέθοδο για αξιόπιστες προβλέψεις πάνω σε μη ισορροπημένα δεδομένα. Το κάνει αυτό με την ισόποση διανομή δειγμάτων για κάθε αναδίπλωση, όπως έκανε και η

παράμετρος stratify της συνάρτησης train_test_split που χρησιμοποιήθηκε στην εκπαίδευση των DataFrames.

Ακολουθεί σύντομη περιγραφή του αρχείου model_kfold.py που κάνει εκπαίδευση του μοντέλου με την παραπάνω μέθοδο.

Πρώτον, αρχικοποιείται η μέθοδος StratifiedKFold και περνάει στη μεταβλητή kfold. Ακολουθεί η αρχικοποίηση των λιστών που θα περιέχουν τα δεδομένα του σετ εκπαίδευσης και του σετ επαλήθευσης για κάθε αναδίπλωση, καθώς και δύο λίστες που θα αποθηκεύουν την ακρίβεια και την απώλεια της κάθε δοκιμής.

```
37  # initialize stratified kfold with k=3
38  kfold = StratifiedKFold(n_splits=3, shuffle=True)
39
40  # initialize data lists
41  train_x = []
42  train_y = []
43  test_x = []
44  test_y = []
45
46  # lists to compute average metrics
47  acc_per_fold = []
48  loss_per_fold = []
49
```

Στη συνέχεια, τα δεδομένα του Dataset χωρίζονται σε train και test σετ μέσω της μεταβλητής kfold και, αφού οι εικόνες με τις αντίστοιχες ετικέτες τους εισαχθούν σε νέες λίστες, μετατρέπονται σε train και test DataFrames. Το df της εκπαίδευσης χωρίζεται ξανά σε train_df και validate_df ισόποσων διανομών στα δεδομένα χάρη στην παράμετρο stratify της train_test_split.

```
fold_no = 1
for train_idx, test_idx in kfold.split(filenames, categories):

    for i, j in enumerate(train_idx):
        train_x.append(filenames[j])
        train_y.append(categories[j])

    for i, j in enumerate(test_idx):
        test_x.append(filenames[j])
        test_y.append(categories[j])


    df = pd.DataFrame({
        'image': train_x,
        'label': train_y
    })

    test_df = pd.DataFrame({
        'image': test_x,
        'label': test_y
    })

    # create validation set with balanced labels
    train_df, validate_df = train_test_split(df, test_size=0.20, stratify=df['label'])
```

Αφού εκτελεστεί η μετατροπή σε ImageDataGenerators και η τοπολογία του μοντέλου, το μοντέλο περνάει από την ίδια εκπαίδευση με τα προγενέστερα του.

Ύστερη της εκπαίδευσης είναι η εκτίμηση του μοντέλου πάνω στα ξεχωριστά κάθε φορά δεδομένα του test_df. Υπολογίζονται τα Test Accuracy και Test Loss της κάθε αναδίπλωσης και προστίθενται σε λίστες για σύγκριση με τις υπόλοιπες. Γίνεται εκκαθάριση των λιστών που περιείχαν μέχρι τώρα τα δεδομένα για εκπαίδευση και εκτίμηση και το μοντέλο αποθηκεύεται, προχωρώντας τελικά στην εκπαίδευση των επόμενων αναδιπλώσεων.

```
    scores = model.evaluate(test_generator, steps=total_test//batch_size)
    print(f'Score for fold {fold_no}: {model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of {scores[1]*100}%')
    acc_per_fold.append(scores[1]*100)
    loss_per_fold.append(scores[0])

    train_x.clear()
    train_y.clear()
    test_x.clear()
    test_y.clear()

    # save model
    model.save(f"k_fold_{fold_no}.h5")

    fold_no = fold_no + 1
```

Τέλος, εκτυπώνεται η συνολική ακρίβεια και απώλεια κάθε αναδίπλωσης μαζί με τους μέσους όρους τους για το σύνολο του cross-validation, ώστε να γίνει ευκολότερη η σύγκριση διαφορετικών μοντέλων στο μέλλον.

```
# Provide average scores
print('------------------------------------------------------------------------')
print('Score per fold')
for i in range(0, len(acc_per_fold)):
  print('------------------------------------------------------------------------')
  print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} - Accuracy: {acc_per_fold[i]}%')
print('------------------------------------------------------------------------')
print('Average scores for all folds:')
print(f'> Accuracy: {np.mean(acc_per_fold)} (+- {np.std(acc_per_fold)})')
print(f'> Loss: {np.mean(loss_per_fold)}')
print('------------------------------------------------------------------------')
```

Τα σημεία του αρχείου που δε συμπεριλήφθηκαν έχουν την ίδια ακριβώς δομή με το αρχείο model_dataframe.py.

Αποτελέσματα αυτής της μεθόδου παρουσιάζονται παρακάτω. Η αρχική τοπολογία για τα μοντέλα του cross-validation συνίσταται από ένα στρώμα εισόδου 32 νευρώνων, 3 κρυφά στρώματα 64, 128 και 128 νευρώνων αντίστοιχα και ένα στρώμα εξόδου. Ύστερα από κάθε συνελικτικό στρώμα υπάρχουν με τη σειρά στρώματα BatchNormalization και MaxPooling2D. Ως αλγόριθμος βελτιστοποίησης χρησιμοποιήθηκε ο Adam και συμπεριλήφθηκαν τα callbacks EarlyStopping και ReduceLROnPlateau.

Η διάρκεια της εκπαίδευσης είναι 10 εποχές και τα μοντέλα διαφοροποιούνται ως προς τον αριθμό των batches και την αρχική τιμή του ρυθμού εκμάθησης. Αναλυτικότερα:

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.31	90%	0.71	81.66%	0.72	79.78%
2	0.3	91%	0.74	80.72%	0.73	81.03%
3	0.29	91%	0.69	82.41%	0.74	80.56%
Average					0.73	81.1+-0.66

Εικόνα 2.13 - Αποτελέσματα εκπαίδευσης με batch_size=16, lr=1e-4

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.35	90%	0.78	79.80%	0.8	79.50%
2	0.4	88%	0.71	81.30%	0.71	81.70%
3	0.36	89%	0.72	81.20%	0.72	80.90%
Average					0.74	80.7+-0.9

Εικόνα 2.14 - Αποτελέσματα εκπαίδευσης με batch_size=32, lr=1e-4

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.49	84%	0.73	78.70%	0.72	80.85%
2	0.55	86%	0.79	78.80%	0.78	78.70%
3	0.47	86%	0.71	80.60%	0.75	79.90%
Average					0.75	79.9+-0.85

Εικόνα 2.15 - Αποτελέσματα εκπαίδευσης με batch_size=64, lr=1e-4

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.05	98%	0.61	86.10%	0.65	85.67%
2	0.11	96%	0.61	84.62%	0.63	84.96%
3	0.04	99%	0.6	87.00%	0.56	87.20%
Average					0.61	85.9+-0.93

Εικόνα 2.16 - Αποτελέσματα εκπαίδευσης με batch_size=16, lr=1e-3

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.15	95%	1	78.50%	1.04	77.60%
2	0.15	95%	1.18	73.86%	1.2	73.68%
3	0.06	98%	0.56	86.60%	0.6	86.18%
Average					0.95	79.1+-5.2

Εικόνα 2.17 - Αποτελέσματα εκπαίδευσης με batch_size=32, lr=1e-3

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.17	93%	1.03	78.00%	1.03	77.00%
2	0.09	97%	0.56	85.45%	0.59	85.60%
3	0.09	97%	0.58	86.90%	0.54	86.60%
Average					0.72	83.1+-4.3

Εικόνα 2.18 - Αποτελέσματα εκπαίδευσης με batch_size=64, lr=1e-3

Τα αρχικά μοντέλα μπορούν να χωριστούν σε 2 ομάδες. Αυτά που έχουν αρχικό ρυθμό εκμάθησης 1e-4 και εκείνα με ρυθμό εκμάθησης 1e-3. Για την πρώτη ομάδα μπορούμε να παρατηρήσουμε πως καλύτερα αποτελέσματα έχει το μοντέλο με αριθμό batches 16. Διαθέτει τη χαμηλότερη απώλεια και στα 3 σετ (training, validation, test), αλλά και την καλύτερη σχετική ακρίβεια στις προβλέψεις. Μοναδικό μειονέκτημα είναι η αισθητή απόκλιση στην απώλεια του training σετ και του test σετ.

Στο δεύτερο γκρουπ την πρώτη θέση έχει και πάλι το batch_size=16. Είναι γνωστό πως τα CNN μοντέλα είναι πολύ ευαίσθητα σε αλλαγές στον αριθμό των batches, χάριν όμως στη χαμηλή προκατάληψη της μεθόδου k-fold CV και τη χαμηλή απόκλιση στο κατά μέσο όρο Test Accuracy μπορούμε να είμαστε σίγουροι πως η απόδοση του μοντέλου δεν οφείλεται σε μία τυχαία κορύφωση αλλά σε σταθερή αποτελεσματική εκπαίδευση.

Συνήθως, όσον αφορά τους ρυθμούς εκμάθησης, ισχύει το ότι ένας μεγάλος ρυθμός εκμάθησης επιτρέπει στο μοντέλο να μαθαίνει γρηγορότερα, με κόστος να φτάσει σε ένα υποβέλτιστο τελικό σύνολο βαρών. Αντίστοιχα, ένας μικρότερος ρυθμός εκμάθησης μπορεί να επιτρέψει στο μοντέλο να μάθει ένα πιο βέλτιστο ή ακόμα και συνολικά βέλτιστο σύνολο βαρών, αλλά μπορεί να χρειαστεί πολύ περισσότερος χρόνος για να εκπαιδευτεί [43].

Το επιχείρημα αυτό αποδεικνύεται και στην περίπτωση μας, όπου ο μεγαλύτερος ρυθμός εκμάθησης έφτασε γρηγορότερα στον υπολογισμό βέλτιστων βαρών και συμπερασματικά, βέλτιστης ακρίβειας, έναντι του μικρότερου ρυθμού. Μοναδική παρατήρηση είναι η σχετικά μεγαλύτερη απόκλιση (0.93 έναντι 0.66) στη συνολική ακρίβεια του test set, που ενδέχεται να προδίδει μια αναμενόμενη αλλά αμελητέα αστάθεια. Η αστάθεια αυτή είναι αισθητή εάν συγκρίνουμε όλες τις αποκλίσεις στην ακρίβεια ανάμεσα στα δύο γκρουπ.

Τα πειράματα συνεχίστηκαν στο κυρίαρχο μοντέλο, αυτό της εικόνας 20, αλλά και στο μοντέλο της εικόνας 22, για να γίνει περαιτέρω σύγκριση. Σειρά έχει η προσθήκη στρώματος Dropout με συντελεστή 0.2.

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.01	99. 50%	0.81	83.50%	0.79	84.40%
2	0.01	99. 70%	0.69	86.80%	0.67	86.40%
3	0.02	99. 44%	0.75	85.05%	0.83	84.32%
Average					0.76	85.1+-0.9

Εικόνα 2.19 - Αποτελέσματα εκπαίδευσης με batch_size=16, Dropout(0.2)

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.03	99. 10%	0.63	86.25%	0.59	86.40%
2	0.1	96. 92%	1.09	79.32%	1.1	79.16%
3	0.02	99. 50%	0.55	87.57%	0.59	86.84%
Average					0.76	84.2+-3.5

Εικόνα 2.20 - Αποτελέσματα εκπαίδευσης με batch_size=64, Dropout(0.2)

Τα αποτελέσματα είναι θετικά και στα δύο μοντέλα, όμως σε αυτό της δεύτερης εικόνας υπάρχει σημαντική απόκλιση στην ακρίβεια αλλά και την απώλεια, που προδίδει αστάθεια. Επομένως, από τα δύο επιλέχθηκε το σταθερότερο πρώτο, το οποίο παρουσίαζε και την καλύτερη συνολική ακρίβεια.

Δεδομένων των πολύ κοντινών αποτελεσμάτων των μοντέλων στις εικόνες 2.19 και 2.20, αποφασίστηκε να γίνει μία πιο ολοκληρωμένη εκπαίδευση διάρκειας 20 εποχών, ώστε να εντοπισθούν οι διαφορές ανάμεσα στην εφαρμογή και μη στρώματος Dropout. Ο συντελεστής του Dropout ανέβηκε στο 0.5 για να είναι πιο έντονη η επιρροή του στην εκπαίδευση.

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.01	99. 66%	0.55	89.83%	0.62	89.07%
2	0.01	99. 75%	0.88	85.85%	0.91	84.47%
3	0.01	99. 60%	1.02	83.82%	1	83.50%
Average					0.85	85.7+-2.4

Εικόνα 2.21 - Αποτελέσματα εκπαίδευσης 20 εποχών, χωρίς Dropout

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	0.0017	99. 99%	0.77	86.80%	0.76	87.06%
2	9.32E-04	99. 99%	0.79	86.53%	0.75	86.14%
3	0.001	100%	0.8	86.20%	0.83	85.90%
Average					0.78	86.4+-0.48

Εικόνα 2.22 - Αποτελέσματα εκπαίδευσης 20 εποχών, με Dropout(0.5)

Αμφότερα τα μοντέλα εκπαιδεύθηκαν για 20 εποχές, όμως σε ορισμένες περιπτώσεις η εκπαίδευση τερματίσθηκε λόγω κορεσμού. Στην περίπτωση του μοντέλου χωρίς στρώμα Dropout, και στις 3 αναδιπλώσεις η εκπαίδευση τερματίσθηκε στις 13, 14 και 13 εποχές αντίστοιχα, μαρτυρώντας την αδυναμία του μοντέλου να φτάσει σε υψηλότερα επίπεδα και μένοντας σε τοπικό μέγιστο. Αντιθέτως, το μοντέλο με Dropout με συντελεστή 0.5 κατάφερε να ολοκληρώσει την εκπαίδευση, με εξαίρεση την πρώτη αναδίπλωση, όπου σταμάτησε στις 18 εποχές.

Συμπερασματικά, το μοντέλο που διαθέτει στρώμα Dropout αποδείχθηκε ως το πιο ισχυρό και πιο πολλά υποσχόμενο, με αρκετό χώρο για ρυθμίσεις που μπορούν να βελτιώσουν τις δυνατότητες του. Το αδύναμο μοντέλο της εικόνας 2.21 δε διαθέτει χώρο για βελτίωση και έτσι απορρίφθηκε.

Παρατηρώντας τα πειραματικά αποτελέσματα των μοντέλων σε άγνωστα δεδομένα στις εικόνες 6.1 και 6.2 του κεφαλαίου 6, μπορούμε να συμπεράνουμε πως δεν είναι ιδανικά. Αιτία για αυτό μπορεί να είναι η κακή ποιότητα των δεδομένων στα οποία γίνονται προβλέψεις, πιθανότερα όμως είναι η έλλειψη αντιπροσωπευτικού και μεγάλου δείγματος για τα είδη στο συνολικό Dataset. Τα δεδομένα προερχόμενα εξ ολοκλήρου από χρήστες και σε εξωτερικό περιβάλλον φυσικά και δε μοιράζονται την ίδια τάξη και άσπιλη φύση με Datasets φτιαγμένα από επιστήμονες σε ελεγχόμενα περιβάλλοντα. Η παραπάνω υπόθεση ενισχύεται από το

γεγονός πως αρκετά πτηνά έχουν τραγούδια με μεγάλη πολυπλοκότητα και όχι μόνο ένα απλό ξεχωριστό μοτίβο. Ακόμα και στα φασματογραφήματα παρατηρήθηκαν έως και πέντε διαφορετικά μοτίβα για το ίδιο είδος.

2.5 Εφαρμογή ισχυρότερου Dataset

Άμεση λύση για το παραπάνω πρόβλημα ήταν η σύνθεση ενός μεγαλύτερου και πιο αντιπροσωπευτικού Dataset. Για τη συλλογή δεδομένων στραφήκαμε στην πλατφόρμα Xeno – Canto, στην οποία υπάρχουν 714235 ηχογραφήσεις για 10360 είδη πτηνών από επιστήμονες και λάτρεις του χώρου.

Για τη λήψη των δεδομένων χρησιμοποιήθηκε το notebook «Download birds songs recording metadata» από το Kaggle [44]. Γενικά, ο κώδικας του notebook στέλνει αιτήματα στο Xeno – Canto API, προσδιορίζοντας το είδος των δεδομένων που χρειαζόμαστε. Μετά, προχωράει σε λήψη της πρώτης σελίδας δεδομένων και, με βάση το περιεχόμενο της, εκτελείται επανειλημμένα, για λήψη όλων των υπόλοιπων σελίδων. Τέλος, κάνει μετατροπή των δεδομένων από μορφή JSON σε DataFrames, τα οποία μετατρέπει ξανά σε ένα αρχείο .csv.

Το query που χρησιμοποιήθηκε για τη συλλογή των δεδομένων ήταν το:

type: song type: male len_gt:30 q_gt:C area: Europe,

το ίδιο που χρησιμοποιήθηκε για το αρχικό Dataset.

Στο επόμενο στάδιο, τα δεδομένα του .csv φιλτράρονται ώστε να παραμείνουν 160 αρχεία αποκλειστικά από τα 50 αρχικά είδη. Τα αρχεία εντάσσονται σε μία λίστα με τη μορφή του συνδέσμου τους στο Xeno – Canto και μέσω αυτής της λίστας, κάθε σύνδεσμος γράφεται σε ένα νέο αρχείο birds_europe.txt. Ακολουθεί η εκτέλεση στην κονσόλα της εντολής

```
wget -P /home/tasos/target_dir --trust-server-names -i birds_europe.txt
```

με την οποία γίνεται λήψη όλων των αρχείων και αποθήκευση τους σε ξεχωριστό φάκελο στο σύστημα.

Τέλος, με την εφαρμογή ενός απλού σκριπτ όλα τα αρχεία μετονομάζονται στην ίδια μορφή με αυτά του αρχικού Dataset, η οποία είναι και η απαιτούμενη από τα στάδια προεπεξεργασίας.

Με την επιτυχή προεπεξεργασία όλων των αρχείων, το τελικό Dataset αποτελούνταν από 71693 φασματογραφήματα, έναντι των αρχικών 26440. Η έλλειψη ισορροπίας στα δεδομένα ήταν εμφανής και εδώ, με το κυριότερο είδος, «Sonus Naturalis» να φτάνει τα 5637 δείγματα και το σπανιότερο, «Loxia Curvivostra» να μένει στα 567. Αυτό ήταν αναπόφευκτο, καθώς η ισορροπία του Dataset επηρεάζεται άμεσα από το πλήθος και τη διάρκεια των ηχογραφήσεων, η οποία σύμφωνα με το query ήταν ίδια για όλα τα αρχεία. Ωστόσο, δε θεωρήθηκε αρνητική, αλλά αντιπροσωπευτική της συχνότητας εμφάνισης κάθε πληθυσμού και έτσι δεν έγιναν ενέργειες χειρωνακτικής αφαίρεσης ή πρόσθεσης δειγμάτων για κάλυψη της διαφοράς.

Επίσης, πραγματοποιήθηκε εκπαίδευση μοντέλων που περιείχαν L2 regularizer, μεθόδου που στοχεύει στην καλύτερη γενίκευση του μοντέλου, όπως το προαναφερθέν Dropout. Τα δύο μοντέλα που εκπαιδεύθηκαν χρησιμοποιούσαν τον L2 regularizer, με το ένα μόνο να κάνει και χρήση στρώματος Dropout συντελεστή 0.5. Τα αποτελέσματα της εκπαίδευσης παρουσιάζονται παρακάτω:

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	2.63	70. 1%	2.6	71.11%	2.61	70.91%
2	2. 59	72. 51%	2.51	73.67%	2.5	74.02%
3	4.39	68%	4.5	67.50%	4.55	67.53%
Average					3.22	70.82+-2.6

Εικόνα 2.23 - Αποτελέσματα εκπαίδευσης με L2 Regularizer, χωρίς Dropout

Fold	Loss	Acc	Val_loss	Val_acc	Test_loss	Test_acc
1	1.24	81. 45%	1.49	75.56%	1.49	75.57%
2	1. 23	82. 29%	1.58	73.16%	1.58	73.15%
3	1.23	81. 86%	1.45	76.92%	1.45	76.93%
Average					1.51	75.2+-1.55

Εικόνα 2.24 - Αποτελέσματα εκπαίδευσης με L2 Regularizer και στρώμα Dropout

Τα πειραματικά αποτελέσματα για όλα τα παραπάνω μοντέλα, όπως και για το τελικό που χρησιμοποιείται στην εφαρμογή παρουσιάζονται στην υποενότητα 6.1 του κεφαλαίου 6.

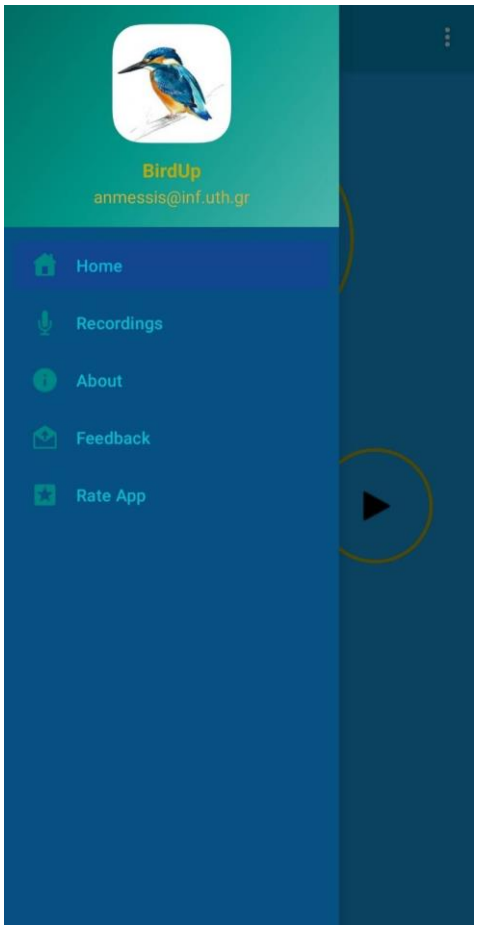
ΚΕΦΑΛΑΙΟ 3

ΛΕΙΤΟΥΡΓΙΚΗ ΠΕΡΙΓΡΑΦΗ

Σε αυτό το κεφάλαιο παρουσιάζονται όλες οι διεπαφές που μπορεί να συναντήσει ο χρήστης μέσα από την εφαρμογή. Έπειτα, περιγράφεται κάθε ξεχωριστή λειτουργία, από την ηχογράφηση μέχρι την αποθήκευση προβλέψεων.

3.1 Κυρίως Μενού

Στο κυρίως μενού παρέχονται πέντε βασικές επιλογές, όπως φαίνεται και στην ακόλουθη εικόνα.

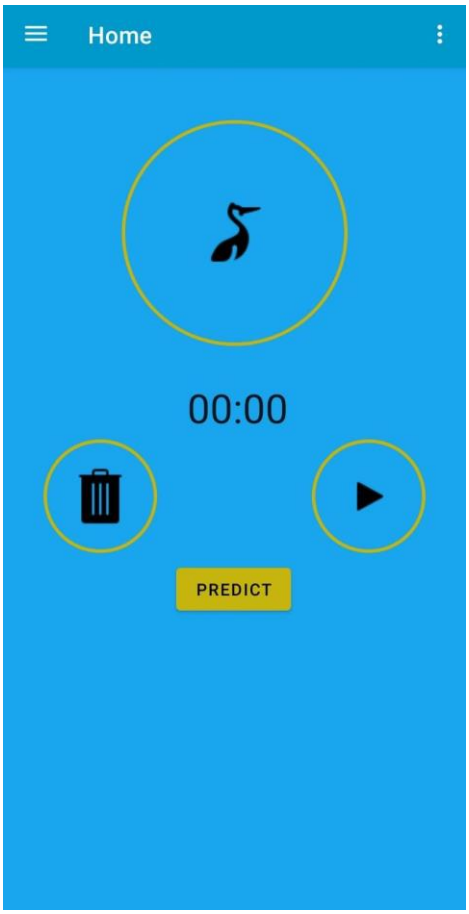


Εικόνα 3.1 - Το κυρίως μενού της εφαρμογής

Το μενού αυτό εμφανίζεται με σύρσιμο του δαχτύλου από την αριστερή άκρη της οθόνης και προς τα δεξιά, ή πατώντας το κουμπί  που βρίσκεται στα αριστερά της μπάρας της εφαρμογής όταν το μενού είναι κλειστό.

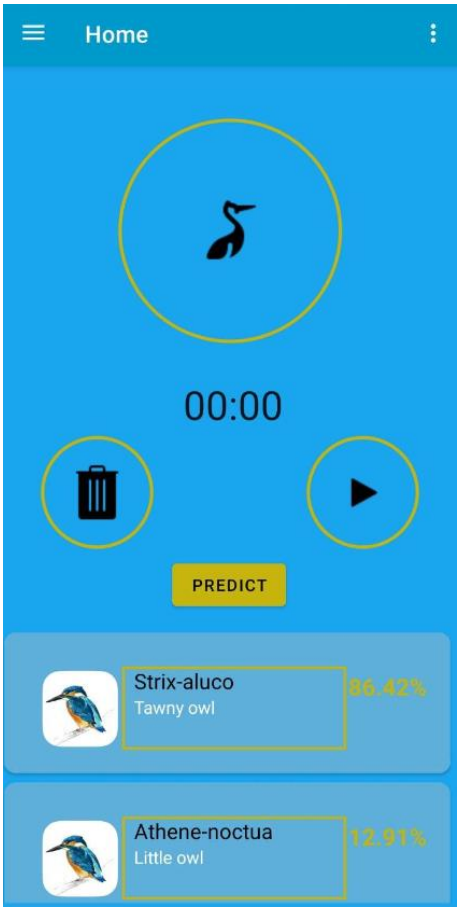
3.1.1 – Home

Η πρώτη επιλογή στο κυρίως μενού και η πρώτη διεπαφή που βλέπει ο χρήστης όταν ανοίγει την εφαρμογή. Το Home αποτελείται από κουμπιά που εκτελούν ηχογράφηση, αναπαραγωγή και διαγραφή της εκάστοτε ηχογράφησης και τέλος κουμπί για την τροφοδότηση της ηχογράφησης στο μοντέλο και τη διενέργεια προβλέψεων. Μόλις υπολογιστούν οι προβλέψεις, οι κυριότερες εμφανίζονται στην οθόνη (εικόνα 3.3) και ο χρήστης μπορεί να αποθηκεύσει την ηχογράφηση με την πρόβλεψη που προτιμά (εικόνα 3.4).

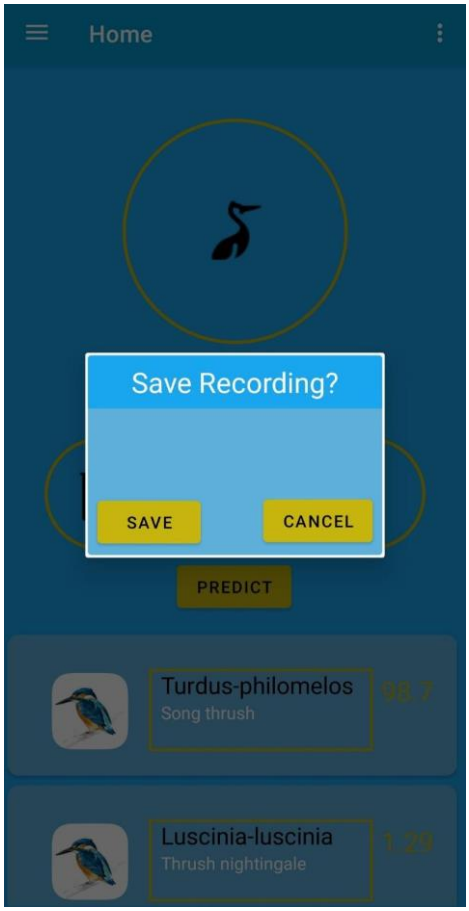


Εικόνα 3.2 – Home

Η μέθοδος λειτουργίας του Home Fragment περιγράφεται αναλυτικά στο υποκεφάλαιο 3.2, «Home Fragment».



Εικόνα 3.3 - Εμφάνιση των αποτελεσμάτων

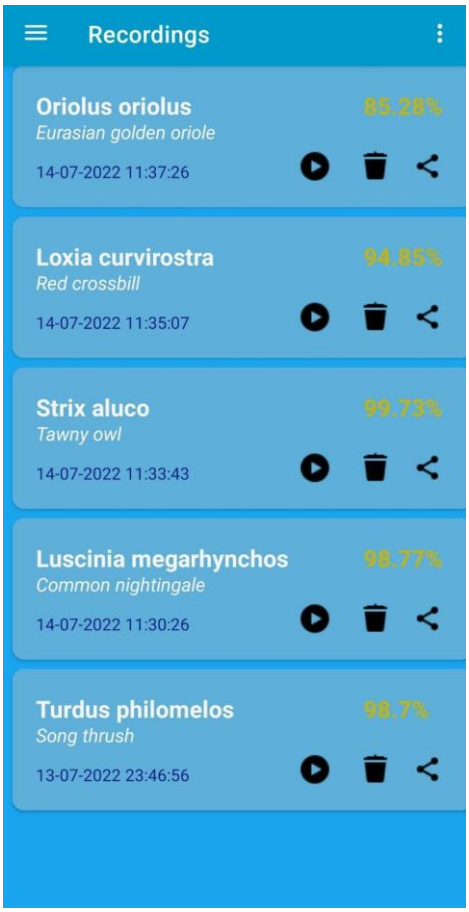


Εικόνα 3.4 - Αποθήκευση της Ηχογράφησης

3.1.2 – Recordings

Η δεύτερη επιλογή του κυρίως μενού και ο χώρος όπου εμφανίζονται όλες οι αποθηκευμένες ηχογραφήσεις του χρήστη, με σχετικές πληροφορίες, όπως την επιστημονική και την κοινή

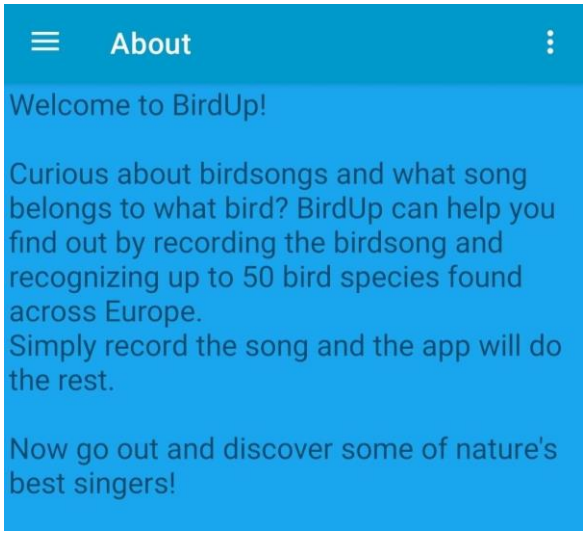
ονομασία του προβλεπόμενου είδους, το ποσοστό βεβαιότητας της πρόβλεψης και την ημερομηνία και ώρα πρόβλεψης. Ακόμα, για κάθε ηχογράφηση υπάρχει η δυνατότητα διαγραφής και αναπαραγωγής της μέσω των ανάλογων κουμπιών.



Εικόνα 3.5 - Recordings

3.1.3 – About

Η τρίτη επιλογή του μενού εμφανίζει ένα απλό κείμενο καλωσορίσματος που περιγράφει σύντομα την ουσία της εφαρμογής.



Εικόνα 3.6 - About

3.1.4 – Feedback

Η επιλογή feedback δίνει στο χρήστη τη δυνατότητα να επικοινωνήσει με το σχεδιαστή της εφαρμογής. Ο χρήστης μπορεί, μέσω e-mail, να αφήσει θετικά ή αρνητικά σχόλια, να μεταφέρει παράπονα και προτάσεις για βελτίωση της εφαρμογής και πολλά άλλα.

Πατώντας την επιλογή Feedback από το μενού, η εφαρμογή προτρέπει το χρήστη να επιλέξει από μία λίστα εφαρμογών αυτήν μέσω της οποίας θα στείλει την κριτική του. Μόλις επιλέξει, η εφαρμογή εκκινεί, με τα πεδία του παραλήπτη και του θέματος ήδη συμπληρωμένα. Το μόνο που έχει να κάνει ο χρήστης είναι να συμπληρώσει το σώμα και να πατήσει αποστολή.

3.1.5 – Rate App

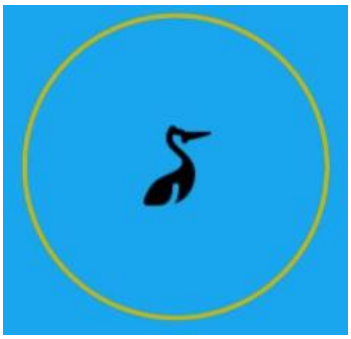
Η επιλογή αυτή ανακατευθύνει το χρήστη στη σελίδα της εφαρμογής στο Play Store, από όπου μπορεί να αφήσει την αξιολόγηση του με βάση την εμπειρία του.

3.2 Home Fragment

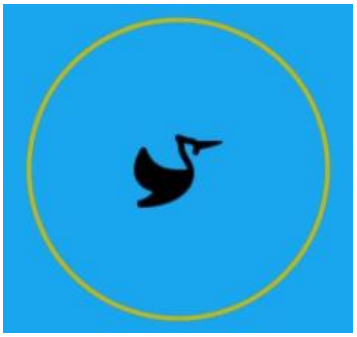
Παρατηρώντας τη διεπαφή του Home στην εικόνα 2.2 εντοπίζουμε τέσσερα κουμπιά και ένα χρονόμετρο. Τα κουμπιά με τη σειρά είναι τα recordButton για ηχογράφηση, playButton για αναπαραγωγή του ήχου, trashButton για διαγραφή και analyzeButton για τροφοδότηση της ηχογράφησής στα στάδια προ επεξεργασίας και τελικά στο μοντέλο. Ακολουθεί περιγραφή της λειτουργίας του καθενός. Ακόμα, υπάρχει η λίστα predictionList στην οποία εμφανίζονται τα αποτελέσματα των τριών κυριότερων προβλέψεων και αναδυόμενο παράθυρο για την αποθήκευση των προβλέψεων.

3.2.1 – recordButton

Το recordButton σε κατάσταση αδράνειας έχει τη μορφή της εικόνας 3.7. Πατώντας πάνω του μία φορά ξεκινάει η ηχογράφηση και η χρονομέτρηση και το κουμπί μεταβαίνει στην κατάσταση της εικόνας 3.8. Πατώντας το ξανά ενώ ηχογραφεί οδηγεί την ηχογράφηση και τη χρονομέτρηση σε παύση. Η κατάσταση παύσης προστέθηκε για να δώσει στο χρήστη τη δυνατότητα να αποφύγει την παράλληλη ηχογράφηση απρόσμενων θορύβων πάνω από το στόχο και να διασφαλίσει την αποφυγή κηλίδωσης του δείγματος. Ο χρήστης μπορεί να συνεχίσει την ηχογράφηση πατώντας το κουμπί και ούτω καθεξής.



Εικόνα 3.7 – recordButton



Εικόνα 3.8 - recordButton κατά την ηχογράφηση

Η ηχογράφηση πραγματοποιείται με συναρτήσεις του MediaRecorder Android class [45]. Αρχικά ορίζονται παράμετροι της ηχογράφησης, όπως ο τύπος του αρχείου, ο αριθμός των καναλιών, το είδος του κωδικοποιητή, το bitrate της ηχογράφησης και ο ρυθμός δειγματοληψίας. Αναλυτικότερα, η ηχογράφηση αποθηκεύεται σε αρχείο τύπου 3GPP, με κωδικοποιητή AAC_ELD ο οποίος είναι σχεδιασμένος για να προσφέρει υψηλή ποιότητα ομιλίας και ήχου, καλύπτοντας ένα εύρος ζώνης ήχου των 20kHz [46] και με καλή υποστήριξη για bitrates μεσαίας και μεγάλης κλίμακας.

Ο ήχος χρησιμοποιεί μονοφωνικό κανάλι με ρυθμό δειγματοληψίας τα 24kHz, σύμφωνα με τις ρυθμίσεις προ επεξεργασίας στο κεφάλαιο 2, ενώ το bitrate υπολογίζεται στα 384kbps για την αναπαραγωγή του αρχείου. Οι συναρτήσεις του MediaRecorder λειτουργούν για συσκευές Android 7 και πάνω (API 24).

3.2.2 – playButton

Τη στιγμή που το recordButton βρίσκεται σε κατάσταση παύσης, ο χρήστης μπορεί να πατήσει το playButton και να ακούσει την ηχογράφηση προτού αποφασίσει να τη διαγράψει ή να την περάσει για πρόβλεψη. Να σημειωθεί πως το πάτημα του playButton προκαλεί απελευθέρωση των πόρων της ηχογράφησης και τερματισμό της, που είναι απαραίτητο για την αναπαραγωγή

του δείγματος. Το playButton είναι απενεργοποιημένο κατά τη διάρκεια της ηχογράφησης και κατά τη διεξαγωγή των προβλέψεων, όπως επίσης και όταν δεν υπάρχει αρχείο προς αναπαραγωγή. Χρησιμοποιεί το MediaPlayer class.

Εικόνα 3.9 - playButton

Εικόνα 3.10 - playButton κατά την αναπαραγωγή

3.2.3 – trashButton

Δουλειά του trashButton είναι να διαγράφει την εκάστοτε ηχογράφηση. Όπως και με το playButton, το πάτημα του τερματίζει την ηχογράφηση (για την περίπτωση που ο χρήστης δεν έχει πατήσει πρωτύτερα το playButton) και ύστερα ελέγχει τον εσωτερικό αποθηκευτικό χώρο της εφαρμογής για αρχεία τύπου .3gp ή .wav. Εάν εντοπίσει τέτοια αρχεία τα διαγράφει και παράλληλα μηδενίζει τη χρονομέτρηση, αλλιώς εμφανίζει το μήνυμα «Nothing in trash».

Η διαγραφή της ηχογράφησης δύναται να γίνει και μετά από το πέρασμα της στο μοντέλο και την εμφάνιση των προβλέψεων, στην περίπτωση που ο χρήστης δεν είναι ευχαριστημένος με την πρόβλεψη ή απλά δε θέλει να την αποθηκεύσει.

Εικόνα 3.11 – trashButton

Πάτημα του κουμπιού σε εκείνη την περίπτωση εκτελεί, συν τοις άλλοις, εκκαθάριση της λίστας προβλέψεων και απόκρυψη της από τη διεπαφή.

3.2.4 – analyzeButton

Αφού ο χρήστης έχει ηχογραφήσει ένα δείγμα και επιθυμεί να δει τις πιθανές προβλέψεις για αυτό, πατάει το κουμπί analyzeButton. Με αυτήν την κίνηση, εκτελείται πρωτίστως απελευθέρωση των πόρων και τερματισμός της ηχογράφησης, όπως παραπάνω και μετά ακολουθεί αρχικοποίηση του πακέτου Python για Android ώστε να τροφοδοτηθεί το δείγμα στα στάδια προ επεξεργασίας. Για ενσωμάτωση του κώδικα Python στην εφαρμογή χρησιμοποιήθηκε το πακέτο Chaquopy.

Εικόνα 3.12 – analyzeButton

3.2.4.α – Επεξεργασία δειγμάτων

Ύστερα από την αρχικοποίηση του πακέτου Python, το δείγμα μετατρέπεται σε μορφή .wav μέσω της εξωτερικής βιβλιοθήκης ffmpeg για επεξεργασία αρχείων πολυμέσων. Το αρχικό πλάνο για χρήση εξαρχής αρχείων wav αντί για 3gp δε λειτούργησε λόγω αδιευκρίνιστης φθοράς στο αρχείο. Η εναλλακτική παρόλα αυτά προτιμήθηκε λόγω του μικρότερου μεγέθους αρχείων που παρείχε η μορφή 3gp συγκριτικά με τη wav.

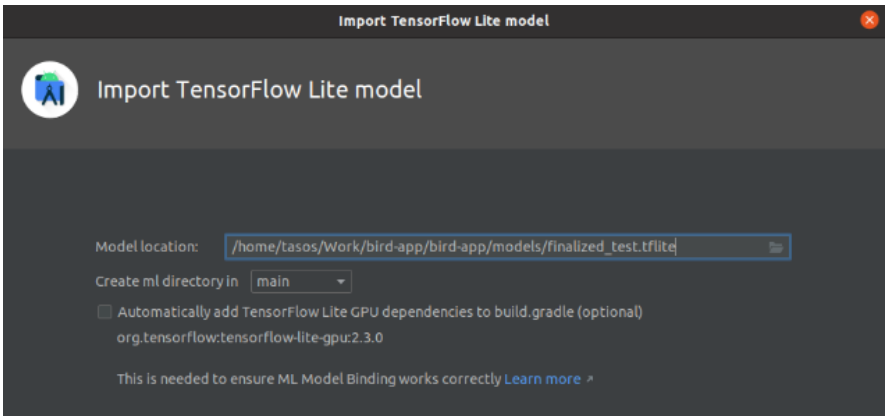
Συνεχίζοντας, δημιουργείται νέος φάκελος «/samples» για την ξεχωριστή αποθήκευση των χωρισμένων αρχείων και των εικόνων και ύστερα εκτελούνται τα στάδια προ επεξεργασίας με τη σειρά, με εξαίρεση το «load.py» που έχει αντικατασταθεί από κώδικα Kotlin.

Τελευταία ενέργεια πριν την αρχικοποίηση του μοντέλου είναι ο έλεγχος του παραπάνω φακέλου για την ύπαρξη εικόνων. Είναι απολύτως πιθανό να υπάρχει έντονη παρουσία θορύβου στο σύνολο του δείγματος, ή αντίθετα, η πηγή του ήχου που προσπαθούμε να καταγράψουμε να είναι αδύναμη και μη εύκολα διαχωρίσιμη από το περιβάλλον Σε αυτήν την περίπτωση που οι περιεχόμενες συχνότητες στο αρχείο ανήκουν σε πολύ κοντινές τιμές, ενδέχεται η σύγκριση των δειγμάτων στο στάδιο split_wav.py να μην τις ξεχωρίσει και επομένως να τις αφαιρέσει όλες. Ο έλεγχος αυτός καλύπτει τη συγκεκριμένη, μολονότι σπάνια περίπτωση.

3.2.4.β – Ενσωμάτωση του Μοντέλου

Η ενσωμάτωση μοντέλων μηχανικής μάθησης σε κινητές συσκευές Android επιτρέπεται μόνο υπό τη μορφή .tflite που αντιστοιχεί στο Tensorflow Lite [47], μία ελαφρύτερη και γρηγορότερη εναλλακτική του Tensorflow σχεδιασμένη για αυτόν ακριβώς το σκοπό. Επομένως, είμαστε υποχρεωμένοι να μετατρέψουμε το μοντέλο μας από τη μορφή H5 στην tflite. Αυτό το κάνουμε με το πρόγραμμα tflite_convert.py που ακολουθεί τα απαραίτητα βήματα όπως περιγράφονται στο [48].

Έχοντας πλέον μετατρέψει το μοντέλο μας στην κατάλληλη μορφή για τη συσκευή, το εισάγουμε στο περιβάλλον της εφαρμογής μέσω του AndroidStudio. Πατώντας δεξί κλικ στο φάκελο «ml» που έχουμε δημιουργήσει για αποθήκευση των μοντέλων κάτω από το φάκελο «app», επιλέγουμε με τη σειρά New > Other > TensorFlow Lite Model, εμφανίζοντας το παρακάτω παράθυρο:



Επιλέγουμε το μοντέλο και στη συνέχεια το αρχικοποιούμε φορτώνοντας το με τη μορφή tflite.Interpreter [49]. Μαζί του φορτώνουμε και δύο αρχεία txt που περιέχουν τις επιστημονικές και τις κοινές ονομασίες των ειδών αντίστοιχα, από τα οποία θα τραβήξουμε τα απαραίτητα δεδομένα για κάθε πρόβλεψη. Τέλος αρχικοποιούμε και ένα αντικείμενο ImageProcessor [50] για κανονικοποίηση των δειγμάτων πριν την τροφοδότηση τους.

3.2.4.γ – Υπολογισμός Αποτελεσμάτων

Για να υπολογισμό των προβλέψεων, φορτώνουμε ένα προς ένα όλα τα δείγματα μέσω ενός for-loop και τα διαβάζουμε ως bitmaps. Παράλληλα αρχικοποιούμε δύο TensorBuffers, ένα για τα δεδομένα εισόδου στο μοντέλο και ένα για τα δεδομένα της εξόδου. Οι TensorBuffers έχουν τύπο δεδομένων Float 32 και διαστάσεις (1, 168, 224, 3) για την είσοδο και (1, 50) για την έξοδο. Το bitmap φορτώνεται σε μία μεταβλητή τύπου TensorImage, ή οποία στη συνέχεια κανονικοποιείται από τον ImageProcessor και φορτώνεται με τη σειρά της στον TensorBuffer της εισόδου. Ύστερα, καλείται ο Interpreter στον οποίον έχουν φορτωθεί οι δύο TensorBuffers και υπολογίζει τα αποτελέσματα.

Τα αποτελέσματα διαβάζονται ως πίνακας float και ταξινομούνται κατά φθίνουσα σειρά. Ταυτόχρονα, οι δείκτες των μη ταξινομημένων αποτελεσμάτων μεταφέρονται σε μία λίστα, η

οποία θα τους αντιστοιχίσει στις κατάλληλες σειρές στα αρχεία με τις ονομασίες. Γίνεται ένας τελευταίος έλεγχος κατά τον οποίον εξετάζεται εάν η καλύτερη πρόβλεψη έχει ακρίβεια μικρότερη ή ίση με 50%. Εάν ναι, οι προβλέψεις για το παρόν δείγμα αγνοούνται καθώς δεν είναι καλύτερες από τυχαίες προβλέψεις και φορτώνεται το επόμενο δείγμα της ηχογράφησης.

Αν όχι, προχωράμε σε στρογγυλοποίηση των προβλέψεων έως δύο δεκαδικά ψηφία και από το σύνολο τους κρατάμε τις τρεις καλύτερες. Με τη βοήθεια της παραπάνω λίστας δεικτών μεταφέρουμε τα στοιχεία της κάθε πρόβλεψης σε τρεις περαιτέρω ξεχωριστές λίστες, που αποθηκεύουν τις επιστημονικές ονομασίες, τις κοινές ονομασίες και τα ποσοστά ακρίβειας τους.

Επαναλαμβάνουμε την παραπάνω διαδικασία για κάθε δείγμα και στο τέλος υπολογίζουμε τους μέσους όρους των τριών λιστών. Κλείνουμε το μοντέλο που έχει τελέσει το έργο του και φορτώνουμε τα τρία καλύτερα αποτελέσματα στη λίστα που εμφανίζει τις προβλέψεις.

3.2.5 – predictionList

Εδώ εμφανίζονται οι τρεις κυριότερες προβλέψεις του μοντέλου. Ο χρήστης έχει την επιλογή απλά να εξετάσει τα αποτελέσματα και να τα διαγράψει για να ηχογραφήσει κάτι διαφορετικό, ή να πατήσει πάνω σε μία από τις προβλέψεις εάν θέλει να αποθηκεύσει την ηχογράφηση του. Σε αυτό το ενδεχόμενο, εμφανίζεται το αναδυόμενο παράθυρο που παρουσιάζεται στη συνέχεια.

Εικόνα 3.13 - Αποτέλεσμα από το predictionList

Το δείγμα στην εικόνα 3.13 περιέχει όλα τα δεδομένα που εξάγομε από τη διαδικασία υπολογισμού των αποτελεσμάτων. Με μαύρα γράμματα εμφανίζεται η επιστημονική ονομασία του πτηνού, με λευκά η κοινή ονομασία του, ενώ η ακρίβεια της πρόβλεψης είναι το ποσοστό στα δεξιά του παραθύρου. Η εικόνα στα αριστερά θα αντικατασταθεί σε μελλοντικό update με εικόνα του αντίστοιχου είδους.

3.2.6 – Custom Popup

Πατώντας πάνω σε μία πρόβλεψη εμφανίζεται το ακόλουθο αναδυόμενο παράθυρο, το οποίο ζητάει την επιβεβαίωση του χρήστη για την αποθήκευση της συγκεκριμένης πρόβλεψης. Πάτημα του κουμπιού «SAVE» αποθηκεύει την ηχογράφηση με τα στοιχεία της επιλεγμένης πρόβλεψης, ενώ πάτημα του «CANCEL» ή οπουδήποτε αλλού στην οθόνη εξαφανίζει το παράθυρο χωρίς καμία άλλη ενέργεια.

Εικόνα 3.14 - Custom Popup

Όταν ο χρήστης πατάει «SAVE», το αρχείο .wav της ηχογράφησης διαγράφεται και κατασκευάζεται ένα νέο αρχείου τύπου .rec ίδιου ονόματος. Σε αυτό μεταφέρονται όλα τα στοιχεία της πρόβλεψης από το predictionList, συμπεριλαμβανομένων της ημερομηνίας και ώρας της ηχογράφησης αλλά και της τοποθεσίας της στο χώρο της εφαρμογής. Η ημερομηνία υπολογίζεται κάθε φορά στην αρχή μίας ηχογράφησης και επισυνάπτεται στο όνομα του αρχείου ώστε να μπορεί στη συνέχεια να εκλεχθεί ευκολότερα. Εφόσον το αρχείο rec έχει κατασκευαστεί

επιτυχώς και εμπεριέχει όλες τις απαραίτητες μεταβλητές, αποθηκεύεται στο Recordings Fragment μαζί με τις υπόλοιπες αποθηκευμένες ηχογραφήσεις του χρήστη.

Το αναδυόμενο παράθυρο κατασκευάζεται μέσω του Dialog class [51], που χρησιμοποιείται ως βάση για το σχεδιασμό όλων των αναδυόμενων παραθύρων σε εφαρμογές Android.

3.3 Αρχικές Δοκιμές

Εδώ ολοκληρώνεται η λειτουργική περιγραφή της εφαρμογής. Όπως μπορεί να γίνει αντιληπτό, οι λειτουργίες της εφαρμογής είναι οι απολύτως απαραίτητες ούτως ώστε ο χρήστης να πετύχει το σκοπό του, χωρίς να βομβαρδίζεται από περίσσειες πληροφορίες. Οι αρχικές δοκιμές της εφαρμογής έγιναν με την ηχογράφηση πτηνών μέσω του YouTube, όπου το κάθε είδος ήταν ήδη γνωστό. Τα αποτελέσματα ήταν αισιόδοξα, υπήρχαν όμως και περιπτώσεις στις οποίες η απόδοση του μοντέλου ανταποδείχθηκε ανεπαρκής. Για την αντιμετώπιση αυτού του προβλήματος και παράλληλα την ενδυνάμωση των προβλέψεων και την ευχρηστία της εφαρμογής, το μοντέλο εκπαιδεύθηκε να κάνει προβλέψεις σε δείγματα του 1 second έναντι των αρχικών 5, τα αποτελέσματα της οποίας βρίσκονται στην υποενότητα 6.1.

ΚΕΦΑΛΑΙΟ 4

ΤΑ ΕΡΓΑΛΕΙΑ

Το BirdUp κατασκευάστηκε για χρήση από το λειτουργικό Android και ο προγραμματισμός του έγινε στη γλώσσα Kotlin και μέσα από το περιβάλλον Android Studio. Αντιστοίχως, το μοντέλο κατασκευάστηκε με τη βοήθεια της βιβλιοθήκης TensorFlow και του πακέτου Conda. Στο κεφάλαιο αυτό παρουσιάζονται τα εργαλεία αυτά, ενώ συμπεριλαμβάνονται επιμέρους πακέτα και εργαλεία που βοήθησαν στη δημιουργία της εφαρμογής. Συμπεριλαμβάνεται επίσης σύντομη περιγραφή της γλώσσας Kotlin και των δυνατοτήτων της.

4.1 Το λειτουργικό Android

Το Android είναι ένα λειτουργικό σύστημα σχεδιασμένο για κινητές συσκευές, πιο συγκεκριμένα smartphones και tablets, και βασίζεται σε μια τροποποιημένη έκδοση του Linux kernel και άλλου λογισμικού ανοιχτού κώδικα. Η εφαρμογή του μπορεί να επεκταθεί και σε ένα εύρος διαφορετικών ηλεκτρονικών συσκευών, μεταξύ των οποίων υπολογιστές, όπου έχει τη μορφή Emulator, ψηφιακές φωτογραφικές μηχανές και κονσόλες βιντεοπαιχνιδιών.

Εικόνα 4.1 - Το λογότυπο του Android (2019 - σήμερα)

Το Android αναπτύχθηκε από μια κοινοπραξία προγραμματιστών γνωστή ως Open Handset Alliance και εμπορικά χορηγείται από την Google [52]. Παρουσιάστηκε τον Νοέμβριο του 2007,

με την πρώτη εμπορική συσκευή Android, το HTC Dream, να κυκλοφόρησε τον Σεπτέμβριο του 2008 [53].

Πάνω από το 70 τοις εκατό των smartphone Android τρέχουν το οικοσύστημα της Google. Οι περισσότερες συσκευές Android διαθέτουν πρόσθετο προ εγκατεστημένο ιδιόκτητο λογισμικό, κυριότερα, το Google Mobile Services(GMS [54]), το οποίο περιέχει εφαρμογές όπως τα Google Chrome, Google Play και άλλα.

Οι εφαρμογές στο Android χρησιμοποιούν τη μορφή APK και διανέμονται μέσω ιδιόκτητων καταστημάτων εφαρμογών όπως το Google Play Store, το Samsung Galaxy Store, το Huawei AppGallery και το Café Bazaar, ή πλατφόρμες ανοιχτού κώδικα όπως τα Aptoide και F-Droid [52].

Το Android είναι το λειτουργικό με τις περισσότερες πωλήσεις παγκοσμίως σε smartphones από το 2011 και σε tablets από το 2013. Από το Μάιο του 2021 μετράει πάνω από τρία δισεκατομμύρια ενεργούς χρήστες μηνιαίως, τη μεγαλύτερη εγκατεστημένη βάση οποιουδήποτε λειτουργικού συστήματος [52], και από τον Ιανουάριο του 2021, το Google Play Store διαθέτει πάνω από τρεις εκατομμύρια εφαρμογές [52, 55]. Το Android 12, που κυκλοφόρησε στις 4 Οκτώβρη 2021, είναι η πιο πρόσφατη έκδοση.

Μοντέλο Εφαρμογών Android

Οι εφαρμογές Android μπορούν να γραφτούν χρησιμοποιώντας γλώσσες Kotlin, Java και C++. Τα εργαλεία Android SDK συντάσσουν τον κώδικά μαζί με τυχόν αρχεία δεδομένων και πόρων σε ένα APK ή ένα πακέτο εφαρμογής Android.

Ένα πακέτο Android, το οποίο είναι ένα αρχείο αρχειοθέτησης με επίθημα .apk, περιέχει τα περιεχόμενα μιας εφαρμογής Android που απαιτούνται κατά το χρόνο εκτέλεσης και είναι το αρχείο που χρησιμοποιούν οι συσκευές που τροφοδοτούνται με Android για την εγκατάσταση της εφαρμογής.

Τα βασικά δομικά στοιχεία κάθε εφαρμογής Android, αποτελούν σημεία εισόδου μέσω των οποίων γίνεται η επικοινωνία του συστήματος ή ενός χρήστη με την εφαρμογή. Διακρίνονται σε τέσσερις διαφορετικούς τύπους:

1. Activities – αλληλεπίδραση του χρήστη με την εφαρμογή.
2. Services – επικοινωνία της εφαρμογής με το σύστημα για τη διατήρηση της λειτουργίας της στο background. Χρησιμοποιείται συνήθως για την εκτέλεση μακροχρόνων λειτουργιών ή εργασιών εξ αποστάσεως.
3. Broadcast receivers – επιτρέπουν την παράδοση συμβάντων που παρακάμπτουν τη λειτουργία της εφαρμογής και της επιτρέπουν να ανταποκριθεί στα συμβάντα αυτά (π.χ. εισερχόμενη κλήση).
4. Content Providers – τελούν καθήκοντα διαχείρισης ενός κοινού συνόλου δεδομένων εφαρμογών που μπορούν να αποθηκευτούν στο σύστημα, μία βάση SQLite και σε οποιαδήποτε άλλη μόνιμη τοποθεσία αποθήκευσης στην οποία έχει πρόσβαση η εφαρμογή [56].

Ο σχεδιασμός της παρούσας εφαρμογής έγινε με βάση το στοιχείο Activity, πάνω στο οποίο φιλοξενούνται επιμέρους Fragments.

Το Activity είναι μία ενιαία, εστιασμένη δραστηριότητα που μπορεί να εκτελέσει ο χρήστης, η οποία δημιουργεί δικό της παράθυρο στην εφαρμογή για τη διευκόλυνση της επικοινωνίας του χρήστη με αυτήν. Το παράθυρο ενός Activity περιέχει όλα τα απαραίτητα στοιχεία σχετικά με τη διεπαφή του και τον κώδικα που αυτό εκτελεί. Μία εφαρμογή συνηθίζεται να έχει πολλαπλές κλάσεις Activity, δηλαδή πολλαπλές οθόνες. Από αυτές, μία καθορίζεται ως τη main Activity, και αποτελεί την πρώτη οθόνη που εμφανίζεται όταν ο χρήστης εκκινεί την εφαρμογή. Κάθε επόμενη κλάση Activity ξεκινάει από αυτήν και μπορεί με τη σειρά της να ξεκινήσει κι άλλες.

Ξεκινώντας από το Android 11, τα Activities έχουν αντικατασταθεί, ως επί το πλείστον, από τα Fragments. Τα Fragments επιτρέπουν τη διαίρεση της γενικής διεπαφής χρήστη ενός Activity σε

διακριτά κομμάτια, προσφέροντας αρθρωτότητα και τη δυνατότητα επαναχρησιμοποίησης UI και κώδικα στα διάφορα στάδια της εφαρμογής. Η δυνατότητα αυτή τα καθιστά την καλύτερη επιλογή για τη σχεδίαση εφαρμογών για smartphones και tablets. Τα Fragments δε μπορούν να χρησιμοποιηθούν αυτόνομα – πρέπει να φιλοξενούνται από ένα Activity ή ένα άλλο Fragment.

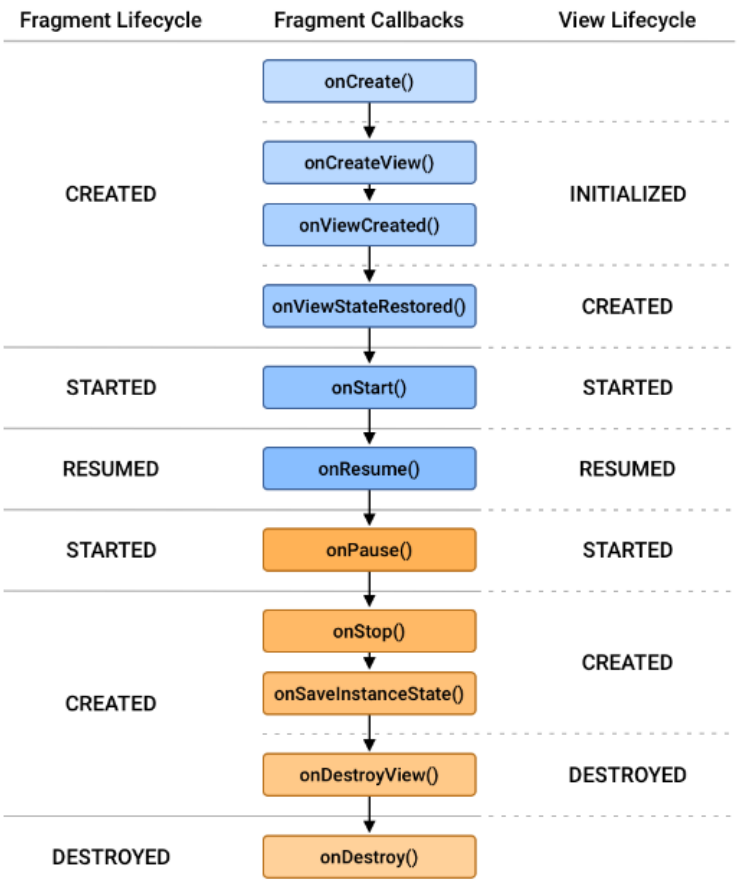
Τα Activities είναι ιδανικά για την τοποθέτηση καθολικών στοιχείων γύρω από το UI της εφαρμογής, όπως ένα navigation drawer, ενώ τα Fragments είναι καλύτερα κατάλληλα για τον ορισμό και τη διαχείριση του UI μίας μεμονωμένης οθόνης.

Η οργάνωση των Activities και Fragments στο περιβάλλον της εφαρμογής γίνεται με τη χρήση στοιβών (back stacks). Η εκκίνηση ενός νέου Fragment το τοποθετεί στην κορυφή της στοίβας και το κάνει το τρέχον Fragment. Όταν αυτό εκπληρώσει το σκοπό του, π.χ. αποστολή ενός e-mail, εξέρχεται από τη στοίβα και στη θέση του μπαίνει το προηγούμενο Fragment, π.χ. η λίστα των απεσταλμένων e-mail.

Τα Fragments συνδυαστικά με τα UI τους έχουν πέντε ουσιαστικές καταστάσεις:

1. INITIALIZED – Το Fragment έχει κατασκευαστεί όμως δεν έχει αντιστοιχηθεί σε κάποιο UI.
2. CREATED – Το Fragment έχει κατασκευαστεί και συνδυαστεί με το αντίστοιχο UI.
3. STARTED – Το Fragment έχει κληθεί από την εφαρμογή και ετοιμάζεται να εκτελέσει τη λειτουργία του.
4. RESUMED – Το Fragment είναι εν λειτουργία και βρίσκεται στην κορυφή της στοίβας. Ο χρήστης αλληλοεπιδρά μαζί του τη συγκεκριμένη στιγμή.
5. DESTROYED – Το Fragment δε χρησιμοποιείται πλέον από την εφαρμογή και αφαιρείται από τη μνήμη τη στοίβα.

Το παρακάτω διάγραμμα παρουσιάζει τις κύριες διαδρομές κατάστασης ενός Fragment. Στα αριστερά παρουσιάζεται η κατάσταση του Fragment, οι μέθοδοι στο κέντρο καλούνται όταν το Fragment μετακινείται μεταξύ καταστάσεων ή εντός μίας κατάστασης και στα δεξιά είναι η κατάσταση του UI που αντιστοιχεί στο Fragment.



Εικόνα 4.2 - Android Fragment Lifecycle

4.2 Το Android Studio

Το Android Studio είναι το επίσημο, ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για το λειτουργικό σύστημα Android της Google, βασισμένο στο λογισμικό IntelliJ IDEA της JetBrains και σχεδιασμένο ειδικά για ανάπτυξη Android. Διατίθεται για λήψη στα λειτουργικά Windows, macOS και Linux [57].

Εικόνα 4.3 -Το λογότυπο του Android Studio

Το Android Studio ανακοινώθηκε στις 16 Μαΐου 2013, στο συνέδριο Google I/O. Ήταν σε στάδιο προεπισκόπησης πρώιμης πρόσβασης ξεκινώντας από την έκδοση 0.1 το Μάιο του 2013 και μετά εισήλθε στο στάδιο beta ξεκινώντας από την έκδοση 0.8 που κυκλοφόρησε τον Ιούνιο του 2014 [58]. Η πρώτη σταθερή έκδοση κυκλοφόρησε το Δεκέμβρη του 2014, ξεκινώντας από την έκδοση 1.0 [59].

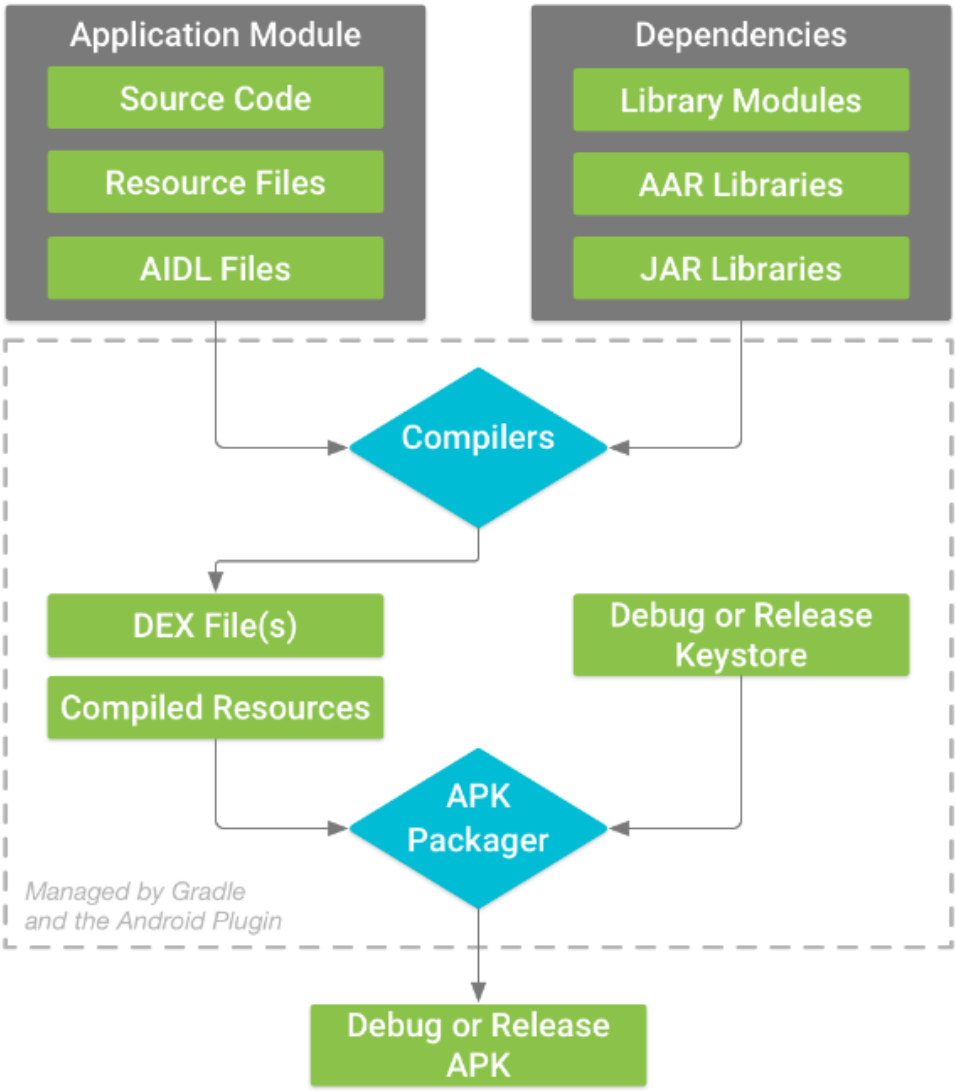
Στις 7 Μαΐου 2019, η Kotlin αντικατέστησε την Java ως προτιμώμενη γλώσσα της Google για την ανάπτυξη εφαρμογών Android [60]. Η Java εξακολουθεί να υποστηρίζεται, όπως και η C++ [61].

Ορισμένες από τις σημαντικότερες δυνατότητες του είναι οι εξής:

- Υποστήριξη Gradle-based κατασκευών. Το Gradle είναι ένα εργαλείο αυτοματισμού κατασκευής για ανάπτυξη λογισμικού σε διάφορες γλώσσες, μεταξύ των οποίων Java, Kotlin, Groovy, Scala, C/C++ και JavaScript. Ελέγχει τη διαδικασία ανάπτυξης από τα στάδια μεταγλώττισης και ενσωμάτωσης πακέτων κώδικα έως τη δοκιμή και δημοσίευση της εφαρμογής. Αυτά τα κάνει στο αρχείο build.gradle, στο οποίο θα γίνει αναφορά αργότερα.
- Ευκολία Σχεδιασμού. Ο προγραμματιστής έχει την επιλογή να σχεδιάσει το UI της εφαρμογής του τόσο με τη χρήση κώδικα (XML), όσο και μέσω του Layout Editor που επιτρέπει προεπισκόπηση του layout και σχεδιασμό με απλό drag & drop των επιθυμητών

στοιχείων. Φυσικά, η κατασκευή και προεπισκόπηση του layout μπορεί να γίνει σε πολλές διαμορφώσεις οθόνης.

- Έξυπνο Auto-complete. Σε πολλές περιπτώσεις όπου χρειάζεται επαναχρησιμοποίηση κώδικα, το Android Studio παρέχει το ζητούμενο κομμάτι κώδικα ακολουθώντας τα πιο πρόσφατα πρότυπα.
- Δημιουργία αρχείων APK (.apk), τα οποία χρησιμοποιούνται από το λειτουργικό Android για τη διανομή και εγκατάσταση εφαρμογών [62]



Εικόνα 4.4 - Η διαδικασία κατασκευής ενός τυπικού Android app module

- Εργαλεία ανακατασκευής (Refactoring Tools). Αποσκοπούν στη βέλτιστη και έξυπνη επιδιόρθωση του κώδικα της εφαρμογής. Για παράδειγμα, επιλέγοντας το εργαλείο μετονομασίας (refactor) για μια μεταβλητή τη μετονομάζει σε κάθε αρχείο κώδικα και πακέτο στο οποίο εμφανίζεται και χρησιμοποιείται.
- Σύνδεση με Version Control Systems. Για έναν προγραμματιστή, η χρήση συστημάτων Version Control είναι αναπόσπαστο κομμάτι της δουλείας του, τόσο σε ατομικό επίπεδο, όσο και σε συλλογικό. Το Android Studio προσφέρει τη δυνατότητα σύνδεσης με τέτοιες πλατφόρμες, όπως το GitHub, Subversion, CVS και διασφαλίζει την εύκολη διαχείριση, οργάνωση και μεταφόρτωση του κώδικα και των επιμέρους εκδόσεων του μέσα από αυτές.
- Επιθεώρηση Βάσης Δεδομένων. Αυτή η λειτουργία επιτρέπει την αναζήτηση και τροφοδότηση της βάσης κατά την εκτέλεση της εφαρμογής και βοηθάει σημαντικά στον εντοπισμό σφαλμάτων εντός της βάσης δεδομένων.
- Εικονική συσκευή Android (Emulator) για την εκτέλεση και τη διόρθωση σφαλμάτων της εφαρμογής χωρίς την απαίτηση φυσικής συσκευής. Ο προγραμματιστής μπορεί να επιλέξει ανάμεσα από πολλές διαθέσιμες συσκευές, διαφορετικών κατασκευαστών και διαστάσεων και να ελέγξει τη συμπεριφορά της εφαρμογής του υπό αυτές τις διατάξεις, εξασφαλίζοντας έτσι την καθολικότητα της.
- Έξυπνος Debugger. Μπορεί να εκτελεστεί αποτελεσματικά σε πληθώρα διαφορετικών συσκευών εντός του emulator, διαθέτει τη δυνατότητα παύσης της ροής του κώδικα (breakpoints) και προσφέρει προεπισκόπηση τιμών για μεταβλητές και εκφράσεις για εύκολα sanity checks.

4.3 Η γλώσσα Kotlin

Η Kotlin είναι μια γλώσσα προγραμματισμού επόμενης γενιάς που αναπτύχθηκε από την JetBrains, με συνεισφορές προγραμματιστών ανοιχτού κώδικα. Συνδυάζει τόσο

αντικειμενοστραφή όσο και λειτουργικά χαρακτηριστικά. Επικεντρώνεται στη διαλειτουργικότητα, την ασφάλεια, τη σαφήνεια και την υποστήριξη εργαλείων. Μπορεί να χρησιμοποιηθεί για server, desktop και mobile εφαρμογές [63].

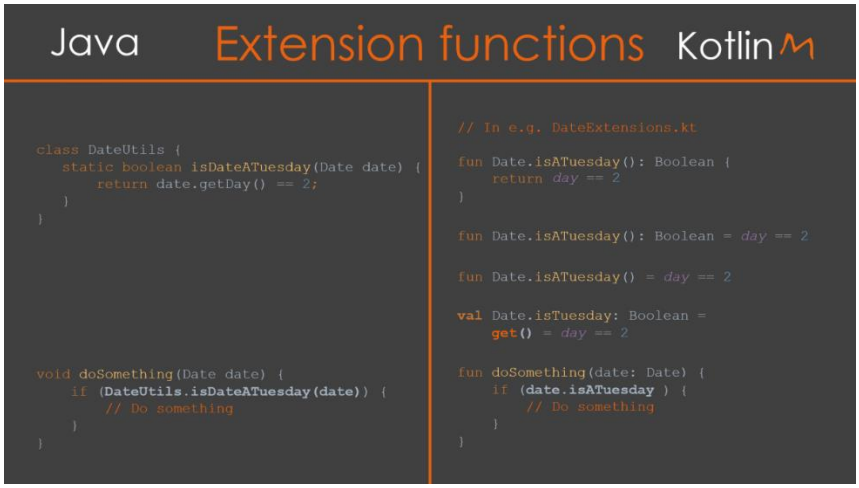
Τα κύρια χαρακτηριστικά της Kotlin που την κάνουν να ξεχωρίζει είναι:

- High-order Functions: Οι ίδιες οι συναρτήσεις μπορούν να είναι παράμετροι συναρτήσεων ή να επιστρέφονται από άλλες συναρτήσεις.
- Null Safety: Τα αντικείμενα της Kotlin ως προεπιλογή δε μπορούν να λάβουν τιμές null, παρά μόνο εάν δηλωθούν με το χαρακτήρα «?» να έπεται του τύπου τους. Έτσι αποφεύγονται τα Null Exceptions.
- Data Class: Οι κλάσεις της Kotlin απαλλάσσουν τον προγραμματιστή από τους πολυάριθμους getters και setters Java, καθώς τους δημιουργεί αυτομάτως.

Εικόνα 4.5 – Kotlin Data Class vs Java Class

- Διαλειτουργικότητα με Java: Κώδικας Kotlin μπορεί να κληθεί από κώδικα Java και αντίστροφα.
- Smart Cast: Τα αντικείμενα αντιστοιχίζονται αυτόματα στον επιθυμητό τύπο εάν δεν έχουν ήδη αντιστοιχηθεί κατά την αρχικοποίηση τους.
- Default Arguments: Εισαγωγή προκαθορισμένων τιμών στις παραμέτρους μίας συνάρτησης κατά τη δήλωση της. Έτσι, εάν μια συνάρτηση κληθεί χωρίς συγκεκριμένες παραμέτρους, χρησιμοποιούνται οι προκαθορισμένες τιμές.

- Named Arguments: Ο προγραμματιστής μπορεί να θέσει τιμές σε παραμέτρους συναρτήσεων κατά την κλήση τους με το να εισάγει το όνομα της παραμέτρου και θέτοντας της μία τιμή με χρήση του συμβόλου «=». Έτσι ο κώδικας γίνεται πιο ευανάγνωστος και η σειρά των παραμέτρων μπορεί να αλλάξει.
- Multi-value Returns: Όταν μια συνάρτηση επιστρέφει πολλές τιμές, αυτές μπορούν να αποσυμπιεστούν σε ξεχωριστές μεταβλητές με τη χρήση δήλωσης αποδομής (destructive declaration).
- Extensions: Πρόσθεση επιπλέον λειτουργικότητας σε υπάρχοντα στοιχεία.



Εικόνα 4.6 - Kotlin Extensions

Εικόνα 4.7 - Το Λογότυπο της Kotlin

4.4 Εξαρτήσεις της Εφαρμογής

Στον κώδικα της BirdUp έχουν δοκιμαστεί και ενσωματωθεί εξωτερικές βιβλιοθήκες και APIs, είτε διότι ήταν απαραίτητα για την επίτευξη του έργου της, είτε για να προσφέρουν μία πιο ολοκληρωμένη εμπειρία στο χρήστη. Αυτά τα dependencies, όπως είναι γνωστά, περιγράφονται στην παρούσα υποενότητα.

Να σημειωθεί πως η εγκατάσταση και φόρτωση των dependencies σε μία εφαρμογή γίνεται με τη μορφή εντολών που προστίθενται στο αρχείο build.gradle, το οποίο προαναφέρθηκε παραπάνω. Οι εντολές αυτές καθορίζουν τις επιμέρους βιβλιοθήκες που απαιτούνται και το build.gradle αναλαμβάνει τη λήψη και αρμονική εγκατάσταση τους στο περιβάλλον της εφαρμογής.

4.4.1 - FFmpeg

Το FFmpeg είναι ένα project λογισμικού ελεύθερου και ανοιχτού κώδικα που αποτελείται από μια σουίτα βιβλιοθηκών και προγραμμάτων για το χειρισμό βίντεο, ήχου και άλλων αρχείων και ροών πολυμέσων. Στον πυρήνα του βρίσκεται το ίδιο το εργαλείο γραμμής εντολών ffmpeg, σχεδιασμένο για την επεξεργασία αρχείων βίντεο και ήχου. Χρησιμοποιείται ευρέως για δια κωδικοποίηση μορφής, βασική επεξεργασία (περικοπή και συνένωση), κλιμάκωση βίντεο, εφέ

βίντεο μετά την παραγωγή και άλλα. Περιλαμβάνει επιπλέον εργαλεία, όπως τα ffprobe και libavcoded τα οποία χρησιμοποιούνται στο παρασκήνιο της BirdUp [64].

Εικόνα 4.8 - Το λογότυπο του FFmpeg

Το FFmpeg στη BirdUp χρησιμοποιήθηκε στη μετατροπή του αρχικού αρχείου 3gp της ηχογράφησης σε wav, καθώς επίσης και στο διάβασμα του αρχείου και την τροφοδότηση του σε παρασκηνιακές διεργασίες.

4.4.2 - Chaquopy

Το Chaquopy είναι ένα Python SDK σχεδιασμένο για να προσφέρει εύκολη ενσωμάτωση κώδικα Python σε εφαρμογές Android. Παρέχει πλήρη ενοποίηση με το σύστημα Gradle του Android Studio, προσφέρει απλά APIs για την κλήση κώδικα Python από Java/Kotlin και αντιστρόφως, ενώ παράλληλα διαθέτει ένα ευρύ φάσμα third-party πακέτων Python, συμπεριλαμβανομένων των SciPy, OpenCV, TensorFlow και πολλών άλλων [65].

Εικόνα 4.9 - Το λογότυπο του Chaquopy

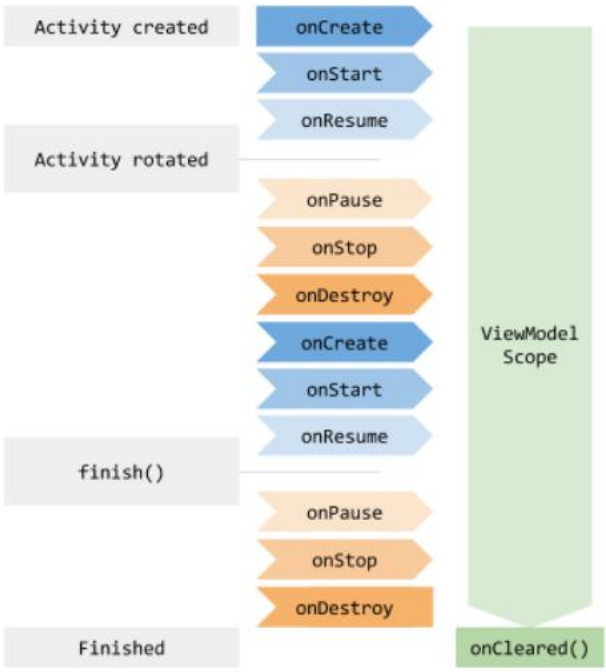
Το Chaquopy έπαιξε σημαντικό ρόλο στην κατασκευή της BirdUp, επειδή όπως απορρέει από την παραπάνω περιγραφή, χρησιμοποιήθηκε για την αδιάλειπτη ενσωμάτωση όλων των σταδίων προ επεξεργασίας του κεφαλαίου 2 και την κατάλληλη επικοινωνία τους με την υπόλοιπη εφαρμογή. Να αναφερθεί πως ενώ η λήψη του είναι δωρεάν, η έκδοση εφαρμογής που χρησιμοποιεί το Chaquopy επιτρέπεται μόνο με τη χρήση άδειας, η οποία λαμβάνεται δωρεάν για εφαρμογές ανοιχτού κώδικα εφόσον τηρεί συγκεκριμένες προδιαγραφές.

4.4.3 - ViewModel

Η κλάση ViewModel έχει σχεδιαστεί για να αποθηκεύει και να διαχειρίζεται δεδομένα που σχετίζονται με τη διεπαφή χρήστη με γνώμονα την ανενόχλητη δράση του κύκλου ζωής. Επιτρέπει στα δεδομένα να επιβιώνουν από αλλαγές διαμόρφωσης, όπως περιστροφές οθόνης ή την απόκρυψη ενός Activity από κάποιο άλλο.

Με πιο απλά λόγια, όταν η εφαρμογή μεταβαίνει στην εκκίνηση ενός Activity ενώ υπάρχει ήδη ένα που εκτελείται, το εκτελούμενο Activity μεταφέρεται στο παρασκήνιο και από εκεί, ανάλογα με τις ανάγκες της εφαρμογής, τερματίζεται από την εφαρμογή για την εξοικονόμηση μνήμης. Εάν ο χρήστης προσπαθήσει να αποκτήσει πρόσβαση στα δεδομένα που παράχθηκαν

από αυτό το Activity θα δει ότι έχουν χαθεί. Η κλάση ViewModel αποτρέπει τέτοιες παρενέργειες και διατηρεί τα αποθηκευμένα δεδομένα της, ακόμα και στο συμβάν τερματισμού ολόκληρης της εφαρμογής. Η χρήση της ViewModel ήταν απαραίτητη για τη διατήρηση των αποθηκευμένων ηχογραφήσεων στο Recordings Fragment. Η εικόνα 4.7 δείχνει τον κύκλο ζωής ενός ViewModel παράλληλα με τον κύκλο ζωής ενός Activity που πρώτα περιστρέφεται και ύστερα τερματίζεται με την εντολή finish().



Εικόνα 4.10 - Ο κύκλος ζωής ενός ViewModel

4.4.4 - Navigation UI

Το Navigation UI επιτρέπει στο χρήστη την εύκολη περιήγηση στο περιβάλλον της εφαρμογής και τα διάφορα περιεχόμενα της. Στη BirdUp χρησιμοποιήθηκε στο Main Menu για την πλοήγηση μεταξύ των κυρίων επιλογών Home, Recordings, About, Feedback και Rate App, όπως παρουσιάστηκε και στην εικόνα 3.1. Ο συγκεκριμένος τρόπος πλοήγησης ονομάζεται «Navigation Drawer» και συναντάται σε πολλές εφαρμογές Android.

4.5 Εξαρτήσεις του Νευρωνικού Δικτύου

Όπως έγινε και με την εφαρμογή , έτσι και η κατασκευή του μοντέλου και των σταδίων προ επεξεργασίας απαίτησε τη χρήση εξωτερικών βιβλιοθηκών και πακέτων. Σε αυτήν την υποενότητα παρουσιάζονται τα κύρια εργαλεία Conda και TensorFlow, μαζί με μερικές από τις βασικότερες βιβλιοθήκες που βοήθησαν στην επίτευξη της λειτουργίας όσων περιεγράφηκαν στο κεφάλαιο 2. Σημειώνεται πως ο κώδικας για τα στάδια προ επεξεργασίας και για το μοντέλο συντάχθηκε σε Python 3.8.

4.5.1 – Conda

Το Conda είναι ένα σύστημα ανοιχτού κώδικα για διαχείριση πακέτων και περιβαλλόντων που τρέχει σε Windows, macOS, Linux και z/OS. Αρχικά αναπτύχθηκε για την επίλυση δύσκολων προκλήσεων διαχείρισης πακέτων που αντιμετωπίζουν οι Data Scientists της Python και σήμερα είναι ένας δημοφιλής διαχειριστής πακέτων για Python και R [66].

Εικόνα 4.11 - Το Λογότυπο του Conda

Στο project της BirdUp το Conda χρησιμοποιήθηκε για τη δημιουργία διαφορετικών περιβαλλόντων στα οποία έτρεχε πειραματικός κώδικας. Τα περιβάλλοντα αυτά χρησιμοποιούσαν διαφορετικές εκδόσεις Python, συγκεκριμένα 3.6 έως 3.9 αλλά και TensorFlow, στο οποίο έτρεχαν πειράματα στις εκδόσεις 1.12 και 2.1. Φυσικά, διαφορετικές

εκδόσεις Python και TensorFlow απαιτούσαν και την εγκατάσταση συμβατών εκδόσεων των πολυάριθμων σχετικών εργαλείων, διαδικασία που ήταν πολύ απλή με τη χρήση του Conda.

4.5.2. – TensorFlow

Το TensorFlow [67] είναι μια δωρεάν βιβλιοθήκη λογισμικού ανοιχτού κώδικα για μηχανική μάθηση και τεχνητή νοημοσύνη. Μπορεί να χρησιμοποιηθεί σε μια σειρά εργασιών, αλλά έχει ιδιαίτερη εστίαση στην εκπαίδευση και την εξαγωγή συμπερασμάτων σε βαθιά νευρωνικά δίκτυα (DNNs). Αναπτύχθηκε από την ομάδα Google Brain για εσωτερική χρήση της Google στην έρευνα και την παραγωγή. Η αρχική έκδοση κυκλοφόρησε το 2015 και η ενημερωμένη έκδοση με το όνομα TensorFlow 2.0 τον Σεπτέμβριο του 2019 [68].

Το TensorFlow μπορεί να χρησιμοποιηθεί σε μια μεγάλη ποικιλία γλωσσών προγραμματισμού, κυρίως στην Python, καθώς και σε JavaScript, C++ και Java. Αυτή η ευελιξία προσφέρεται για μια σειρά εφαρμογών σε πολλούς διαφορετικούς τομείς.

Εικόνα 4.12 - Το Λογότυπο του TensorFlow

Στην κατασκευή του μοντέλου της BirdUp χρησιμοποιήθηκε το TensorFlow 2.1 συνδυαστικά με τη βιβλιοθήκη λογισμικού ανοιχτού κώδικα Keras [69], η οποία παρέχει μία διεπαφή Python για νευρωνικά δίκτυα και συγκεκριμένα λειτουργεί ως διεπαφή για το TensorFlow. Σχεδιασμένη

για να επιτρέπει γρήγορο πειραματισμό με DNNs, εστιάζει στο να είναι φιλικό προς το χρήστη, αρθρωτό και επεκτάσιμο [70].

Εικόνα 4.13 - Το Λογότυπο του Keras

4.5.3 – Λοιπές Βιβλιοθήκες

- Librosa [71]: Βιβλιοθήκη γραμμένη σε Python για ανάλυση μουσικής και ήχου. Στο BirdUp χρησιμοποιήθηκε στο στάδιο load.py όπου μετατρέπει τα αρχεία ήχου σε μονοφωνικό κανάλι, εφαρμόζει ξανά δειγματοληψία με ρυθμό 24kHz και τα αποθηκεύει μέσω της soundfile, αλλά και στο στάδιο preprocessing.py, όπου εφάρμοσε STFT στα αρχεία για την παραγωγή φασματογραφημάτων και για μετατροπή του πλάτους σε κλίμακα Decibel.
- Pydub [72]: Πακέτο γραμμένο σε Python που επιτρέπει την επεξεργασία ήχου μέσω μίας διεπαφής υψηλού επιπέδου. Στη BirdUp χρησιμοποιήθηκε στο στάδιο split_wav.py όπου έκανε κανονικοποίηση των αρχείων ήχου, εφάρμοσε υψιπερατό φίλτρο 220Hz για την αφαίρεση θορύβου και χώρισε τα αρχεία σε ίσα τμήματα 5 seconds για να γίνει η μεταξύ τους σύγκριση.
- Numpy [73]: Η βασική βιβλιοθήκη για επιστημονικούς υπολογισμούς στην Python. Παρέχει ένα πολυδιάστατο αντικείμενο πίνακα μαζί με διάφορα παράγωγα και μια ποικιλία από ρουτίνες για γρήγορες πράξεις σε πίνακες, μεταξύ των οποίων διακριτούς μετασχηματισμούς Fourier, γραμμική άλγεβρα, στατιστική, τυχαία προσομοίωση και άλλα. Η εκπαίδευση νευρωνικών δικτύων είναι αδύνατη χωρίς τη χρήση των λειτουργιών που

προσφέρει. Στη BirdUp η χρήση του συναντάται στα στάδια split_wav.py, preprocessing.py, στην εκπαίδευση του μοντέλου και στη διενέργεια προβλέψεων, οπουδήποτε δηλαδή υπάρχει ανάγκη μαθηματικών πράξεων και χρήσης πινάκων.

Εικόνα 4.14 - Το Λογότυπο του NumPy

- Matplotlib [74]: Είναι βιβλιοθήκη σχεδίασης και οπτικοποίησης για την Python και την αριθμητική της επέκταση μαθηματικών NumPy. Το pyplot, το οποίο χρησιμοποιείται στη BirdUp, είναι μια λειτουργική μονάδα του Matplotlib που παρέχει μία διεπαφή παρόμοια του MATLAB, με τη δυνατότητα χρήσης Python και το πλεονέκτημα ότι είναι δωρεάν και ανοιχτού κώδικα [75]. Στη BirdUp ήταν υπεύθυνο για την οπτικοποίηση των φασματογραφημάτων και την αποθήκευση τους στο preprocessing.py, το σχεδιασμό του πίνακα σύγχυσής για τα αποτελέσματα στο κεφάλαιο 2 και για τις οπτικοποιήσεις που έγιναν στα δεδομένα του Dataset σχετικά με τη διάρκεια και την ποιότητα των δεδομένων.
- Pandas [76]: Δωρεάν βιβλιοθήκη λογισμικού γραμμένη για τη γλώσσα Python για χειρισμό και ανάλυση δεδομένων. Ειδικότερα, προσφέρει δομές δεδομένων και λειτουργίες για το χειρισμό αριθμητικών πινάκων και χρονοσειρών [77]. Στη BirdUp χρησιμοποιήθηκε για να φορτώσει τα δεδομένα του Dataset στην απαραίτητη μορφή DataFrames για τροφοδότηση στο μοντέλο και στη διαδικασία κατασκευής του τελικού Dataset από τα δεδομένα της σελίδας Xeno-Canto.
- Scikit-learn [78]: Δωρεάν βιβλιοθήκη μηχανικής μάθησης λογισμικού για την Python. Διαθέτει διάφορους αλγόριθμους ταξινόμησης, παλινδρόμησης και ομαδοποίησης, όπως SVMs, Random Forests, Gradient Boosting, k-means και έχει σχεδιαστεί για να λειτουργεί

με τις αριθμητικές και επιστημονικές βιβλιοθήκες Python NumPy και SciPy [79]. Στη BirdUp παρείχε τις συναρτήσεις για τη δημιουργία του μοντέλου Stratified k-fold Cross Validation και για το χωρισμό του Dataset κατά την εκπαίδευση, όπως επίσης και τον υπολογισμό των μεταβλητών Balanced Accuracy, top-k, MCC, F1 και του πίνακα σύγχυσης για την εξέταση της αποδοτικότητας του κάθε μοντέλου.

Εικόνα 4.15 - Το Λογότυπο του Scikit-Learn

- OpenCV [80]: Βιβλιοθήκη με λειτουργίες επικεντρωμένες στο real-time computer vision. Είναι ένα εξαιρετικό εργαλείο για την επεξεργασία εικόνας και την εκτέλεση εργασιών Computer Vision. Είναι ανοιχτού κώδικα και μπορεί να εφαρμοστεί σε εργασίες ανίχνευσης προσώπου, τμηματοποίηση αντικειμένων και άλλα. Στο project της BirdUp εφαρμόστηκε για τη μετατροπή των φασματογραφημάτων από μορφή PNG σε JPG και για την αλλαγή μεγέθους τους και μείωσης της ποιότητας εικόνας για την επίτευξη μικρότερων μεγεθών αρχείων.

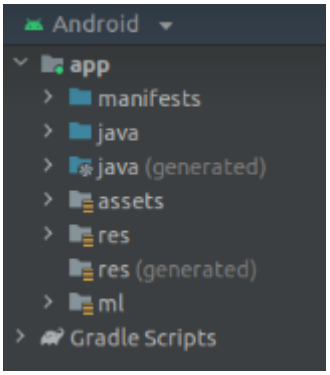
ΚΕΦΑΛΑΙΟ 5

ΛΕΠΤΟΜΕΡΗΣ ΤΕΧΝΙΚΗ ΠΕΡΙΓΡΑΦΗ

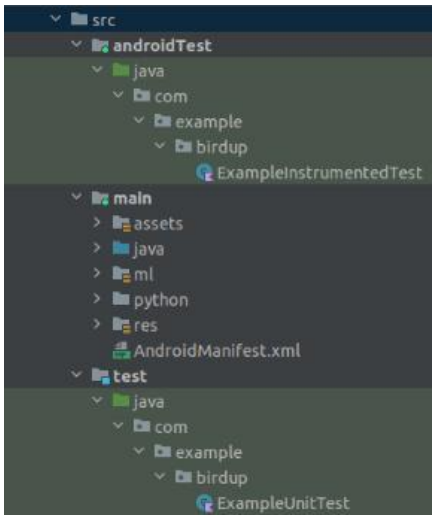
Η οργάνωση του κώδικα ενός project και των πόρων που χρησιμοποιεί είναι πρωταρχικό μέλημα του προγραμματιστή, καθώς επιτρέπει την εύκολη κατανόηση των περιεχομένων του από αυτόν αλλά και από τρίτους που θα χρειαστεί να τον εξετάσουν ή συμβουλευθούν. Ανεξάρτητα από το μέγεθος του project, η οργάνωση και τακτοποίηση του κώδικα σε φακέλους είναι ένδειξη σωστής προγραμματιστικής συμπεριφοράς και πρέπει πάντα να τηρείται.

Στο κεφάλαιο αυτό λοιπόν, αναλύονται οι κλάσεις και το σύνολο των πόρων που χρησιμοποιεί η εφαρμογή BirdUp, καθώς και η μεταξύ τους αλληλεπίδραση που οδηγεί στην κατάλληλη λειτουργία της εφαρμογής. Για να γίνει ευκολότερα κατανοητή η διαδικασία, πρέπει να προηγηθεί η περιγραφή της βασικής δομής ενός Android Studio project.

Για κάθε project του Android Studio υπάρχει το μανιφέστο, ο κώδικας του, τα resources που χρησιμοποιεί και τέλος τα Gradle Scripts. Το Android Studio διαθέτει δύο τρόπους αναπαράστασης των στοιχείων αυτών που παρουσιάζονται στις εικόνες 5.1 και 5.2 αντίστοιχα. Η αναπαράσταση «Android» της εικόνας 5.1 είναι απλούστερη από την αναπαράσταση «Project Files» της εικόνας 5.2, και έτσι προτιμάται από τους προγραμματιστές κατά τη δημιουργία μίας εφαρμογής.



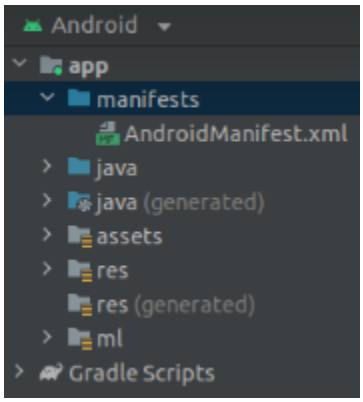
Εικόνα 5.1 - Αναπαράσταση «Android»



Εικόνα 5.2 – Αναπαράσταση «Project Files»

5.1 Android Manifest

Στο φάκελο «manifests» της εικόνας 5.3 βρίσκεται το αρχείο AndroidManifest.xml, το οποίο περιέχει βασικές πληροφορίες της εφαρμογής που είναι απαραίτητες για την εκτέλεση της και τη διαχείριση της από το λειτουργικό και την πλατφόρμα Google Play.



Εικόνα 5.3 - Τοποθεσία του αρχείου AndroidManifest.xml

Συγκεκριμένα, το αρχείο αυτό πρέπει να περιέχει τα εξής στοιχεία:

1. Το όνομα πακέτου της εφαρμογής. Κατά την δημιουργία του αρχείου APK, τα αρχεία Gradle αντικαθιστούν αυτό το όνομα με ένα αναγνωριστικό (application ID), το οποίο έπειτα χρησιμοποιείται ως μοναδικό αναγνωριστικό της εφαρμογής στο σύστημα και στο Google Play.

2. Τα δομικά στοιχεία που χρησιμοποιούνται από την εφαρμογή, συγκεκριμένα τα προαναφερθέντα activities, services, broadcast receivers και content providers. Κάθε στοιχείο πρέπει να συνοδεύεται από λεπτομέρειες που περιγράφουν τον σκοπό του και τον τρόπο χρήσης του. Στην BirdUp τα δομικά στοιχεία που έχουν δηλωθεί στο μανιφέστο είναι ένα Activity και τέλος ένας content provider που επιτρέπει την κοινοποίηση ηχογραφήσεων.

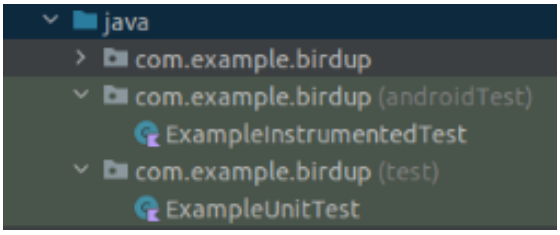
3. Τα δικαιώματα (permissions) που χρειάζεται η εφαρμογή για να αποκτήσει πρόσβαση σε συγκεκριμένες λειτουργίες του συστήματος ή άλλων εφαρμογών. Δηλώνονται επίσης οι προϋποθέσεις υπό τις οποίες μπορεί μια άλλη εφαρμογή να αποκτήσει πρόσβαση σε περιεχόμενο αυτής της εφαρμογής. Στο manifest της BirdUp χρησιμοποιείται το δικαίωμα RECORD_AUDIO, το οποίο επιτρέπει τη χρήση του μικροφώνου για τη διενέργεια ηχογραφήσεων.

4. Τα hardware και software χαρακτηριστικά που απαιτεί η εφαρμογή ώστε να μπορεί να χρησιμοποιηθεί. Αυτά επηρεάζουν ποιες συσκευές μπορούν να εγκαταστήσουν την εφαρμογή από το Google Play. Δεν υπάρχουν τέτοιες απαιτήσεις για την BirdUp.

5.2 Πακέτα Κώδικα

Όλος ο κώδικας Kotlin του project αποθηκεύεται μέσα στο φάκελο «Java» και είναι οργανωμένος σε πακέτα. Πέρα από τα αρχεία κώδικα, σε αυτόν το φάκελο συναντώνται όλα τα test files, τα οποία μπορούν να εκτελούν διαφορετικές εκδοχές του κώδικα και τα αποτελέσματα

τους μπορούν να βοηθήσουν τον προγραμματιστή στον εντοπισμό σφαλμάτων και bugs που ενδέχεται να παρουσιάσει η εφαρμογή.

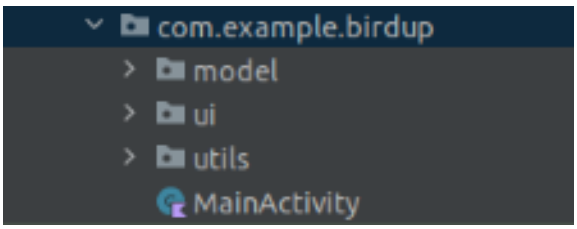


Εικόνα 5.4 - Περιεχόμενα φακέλου «Java»

Στον πρώτο φάκελο είναι ο κώδικας και στους δύο επόμενους τα test files

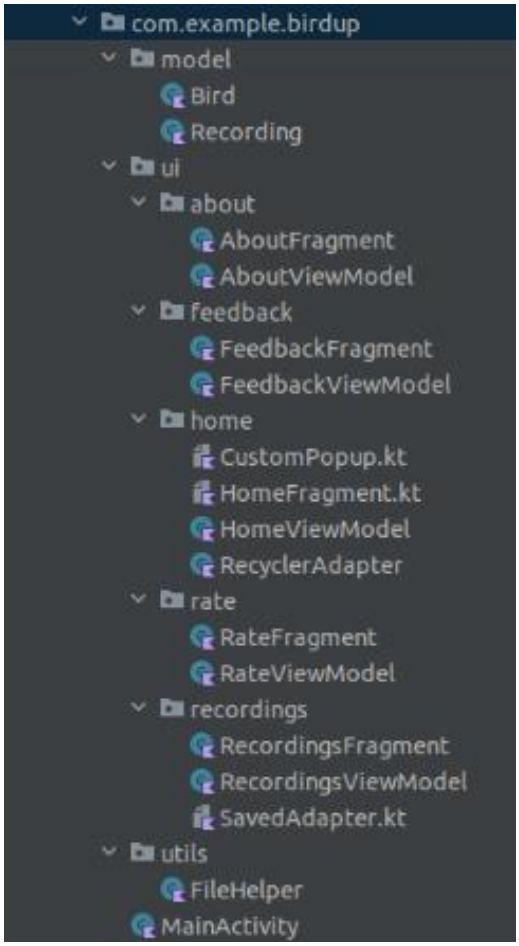
Επιστρέφοντας στα αρχεία κώδικα, εκεί τοποθετούνται οι κλάσεις Activity ή Fragment που έχουν κατασκευαστεί από τον προγραμματιστή, συμπεριλαμβανομένων και όσων άλλων κλάσεων είναι απαραίτητες για την εφαρμογή. Στην αρχή κάθε νέου project που συγγράφεται με Kotlin το Android Studio δημιουργεί αυτόματα το αρχείο MainActivity.kt, το οποίο αντιπροσωπεύει το κύριο Activity που εκτελείται στην εκκίνηση της εφαρμογής και από το οποίο ξεκινούν όλα τα μεταγενέστερα Activities ή Fragments. Στην αρχική του μορφή, το MainActivity περιέχει τον απαραίτητο κώδικα για την αρχικοποίηση του βασικού layout, μαζί με κώδικα που αρχικοποιεί τον εκάστοτε τρόπο πλοήγησης – εδώ το προαναφερθέν «Navigation Drawer» – ώστε ο χρήστης να μπορεί να περιηγηθεί στην εφαρμογή. Ο προγραμματιστής μπορεί στη συνέχεια να κάνει τις δικές του προσθήκες, όπως για παράδειγμα, την προσθήκη ενός Intent για την αποστολή e-mail.

Ο κώδικας της εφαρμογής αποτελείται από 17 κλάσεις (εικόνα 5.6), οι οποίες χωρίζονται στις κλάσεις μοντέλου, κλάσεις UI, μία βοηθητική κλάση «utils» και την κύρια, «top-level» κλάση.



Εικόνα 5.5 - Ο φάκελος του κώδικα της BirdUp

- Οι κλάσεις μοντέλου αντιστοιχούν στα μοντέλο των προβλέψεων, δηλαδή εμπεριέχουν όλα τα απαραίτητα στοιχεία του, όπως τις ονομασίες των πτηνών, την ημερομηνία ηχογράφησης, το ποσοστό ακρίβειας της πρόβλεψης και την τοποθεσία του αρχείου της ηχογράφησης στο σύστημα.
- Οι κλάσεις UI καθορίζουν τη λειτουργία του κάθε στοιχείου UI της εφαρμογής.
- Η βοηθητική κλάση περιέχει συναρτήσεις που στοχεύουν στην εύκολη αποθήκευση των προβλέψεων, την κατάλληλη αντιστοίχιση ηχογραφήσεων και προβλέψεων και την ανάκτηση των αποθηκευμένων ηχογραφήσεων.
- Η «top-level» κλάση είναι η γνωστή MainActivity που, όπως έχει ήδη αναφερθεί, είναι υπεύθυνη για την αρμονική σύνδεση των κλάσεων των παραπάνω κατηγοριών μέσα στο περιβάλλον της εφαρμογής.



Εικόνα 5.6 - Όλα τα αρχεία κώδικα της εφαρμογής

MainActivity.kt

Το αρχείο MainActivity.kt περιέχει την κλάση Activity που καλείται με το άνοιγμα της εφαρμογής. Είναι υπεύθυνο για τη δημιουργία του Κυρίως Μενού και των Fragments κάθε επιλογής του. Επιπλέον, εντός του MainActivity συμπεριλαμβάνεται κώδικας που κάνει αρχικοποίησή δύο Intents για την αποστολή e-mail και τη βαθμολόγηση της εφαρμογής μέσω ανακατεύθυνσης στη σελίδα της στο Google Play αντίστοιχα.

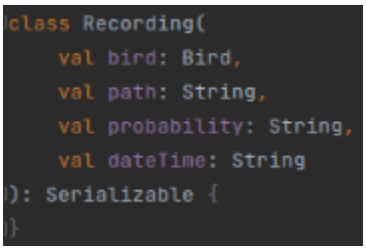
Model

Τα μοντέλα είναι ένα πολυχρησιμοποιούμενο εργαλείο τόσο στην κατασκευή εφαρμογών αλλά και γενικότερα στην ανάπτυξη λογισμικού. Πρόκειται για μία κλάση η οποία περικλείει ορισμένες οντότητες που εμφανίζονται συχνά κατά τη λειτουργία της εφαρμογής, και αποθηκεύει τα βασικά δεδομένα τους. Στην BirdUp υπάρχουν δύο απλά μοντέλα, το Bird και Recording. Το πρώτο αντιπροσωπεύει μεμονωμένα στοιχεία της πρόβλεψης, ενώ το δεύτερο συμπεριλαμβάνει όλα τα απαραίτητα στοιχεία της πρόβλεψης και ενσωματώνει το πρώτο.

```
class Bird(val name: String,
           val latinName: String
): Serializable
```

Εικόνα 5.7 - Η δήλωση της κλάσης Bird

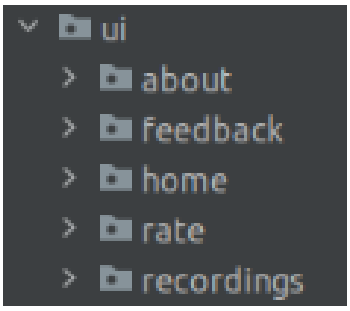
Αμφότερες οι κλάσεις μπορούν να μετατραπούν σε Serializable κλάσεις και αντίστροφα. Αυτό τους δίνει τη δυνατότητα να αποθηκευτούν ως αρχείο και να ανακτηθούν από το σύστημα απαράλλαχτες. Αυτή η λειτουργία επιτρέπει στην BirdUp να αποθηκεύει προβλέψεις και να τις ανακτά μετά τον τερματισμό της εφαρμογής.



Εικόνα 5.8 - Η δήλωση της κλάσης Recording

User Interface

Οι περισσότερες κλάσεις της εφαρμογής είναι συγκεντρωμένες στον υπό-φάκελο ui. Εντός αυτού του φακέλου βρίσκονται όλες οι κλάσεις που είναι σχετικές με το UI της εφαρμογής και ειδικότερα την προβολή και λειτουργία των αντίστοιχων διεπαφών.



Εικόνα 5.9 - Ο υπό-φάκελος UI της εφαρμογής

Η κύρια λειτουργία της εφαρμογής διεξάγεται στους φακέλους home και recordings, στους οποίους βρίσκονται τα αντίστοιχα Fragments, συμπεριλαμβανομένων μερικών βοηθητικών κλάσεων. Οι φάκελοι about, feedback και rate αποσκοπούν στην παροχή μίας βελτιωμένης και ολοκληρωμένης εμπειρίας για το χρήστη.

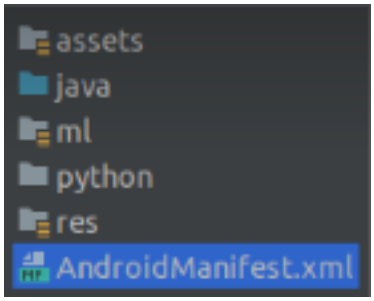
Όπως παρατηρείται στην εικόνα 5.6, ο φάκελος home περιέχει δύο επιπλέον αρχεία πέρα των βασικών HomeFragment και HomeViewModel. Αυτά είναι τα CustomPopup.kt και RecyclerAdapter.kt τα οποία διαχειρίζονται τη λειτουργία του αναδυόμενου παραθύρου για την

αποθήκευση μίας πρόβλεψης και την προβολή της λίστας πιθανών προβλέψεων αντίστοιχα. Παρόμοιο αρχείο βρίσκεται και στο φάκελο recordings, το SavedAdapter.kt, το οποίο αναλαμβάνει τη διαχείριση και προβολή των προβλέψεων στο περιβάλλον της εφαρμογής.

FileHelper.kt

Η κλάση FileHelper του υπό-φακέλου utils περιέχει σημαντικές συναρτήσεις οι οποίες, με τη βοήθεια των παραπάνω μοντέλων, επιτυγχάνουν τη σωστή αποθήκευση και ανάκτηση των προβλέψεων του χρήστη. Λειτουργεί στο παρασκήνιο, καθώς οι συναρτήσεις της καλούνται από άλλα Fragments στα σημεία που χρειάζονται.

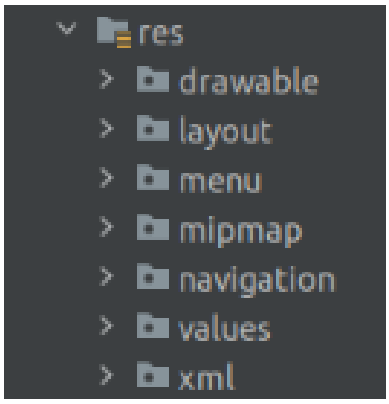
Για τον κώδικα της Python δημιουργείται ξεχωριστός φάκελος ονόματι «python» ο οποίος ιεραρχικά βρίσκεται στο ίδιο επίπεδο με το φάκελο «Java». Επίσης από τον προγραμματιστή δημιουργούνται οι φάκελοι «assets» και «ml». Στον πρώτο τοποθετούνται τα αρχεία txt που περιέχουν τις ετικέτες και τις κοινές ονομασίες των δεδομένων του Dataset, καθώς επίσης και μερικά φασματογραφήματα για τη διενέργεια πειραμάτων. Ο δεύτερος φάκελος είναι η περιοχή όπου αποθηκεύονται όλα τα μοντέλα αρχιτεκτονικής CNN που δοκιμάστηκαν κατά την κατασκευή της εφαρμογής. Η ιεραρχία των φακέλων αυτών φαίνεται στην εικόνα 5.10.



Εικόνα 5.10 - Τοποθεσία των φακέλων «python», «assets» και «ml»

5.3 Resources

Το σύνολο των πόρων που απασχολεί η εφαρμογή, συμπεριλαμβανομένων των εικονιδίων και των ορισμών layouts και themes μπορούν να βρεθούν στο φάκελο res και τους επιμέρους υπό-φακέλους του, όπως αυτοί παρουσιάζονται στην εικόνα 5.11.



Εικόνα 5.11 - Οι υπό-φάκελοι του «res»

Καθένας από τους υπό-φακέλους περιέχει αρχεία τα οποία ενδέχεται να εμφανίζονται παραπάνω από μια φορά στην εφαρμογή. Αποθηκεύονται σε ξεχωριστή τοποθεσία από τα αρχεία κώδικα αφενός για την εύκολη πρόσβαση και επεξεργασία τους, και αφετέρου για την καλύτερη δυνατή αξιοποίηση τους από το σύστημα του Android, το οποίο μπορεί να επιλέξει διαφορετικά resources ανάλογα με τη διαμόρφωση της εφαρμογής (οριζόντια ή κάθετη προβολή, dark mode). Το μόνο που έχει να κάνει ο προγραμματιστής είναι να επιλέξει τα resources που επιθυμεί για κάθε διαμόρφωση και να τα ομαδοποιήσει. Έτσι προωθείται η καθολικότητα της εφαρμογής σε οπτικό επίπεδο.

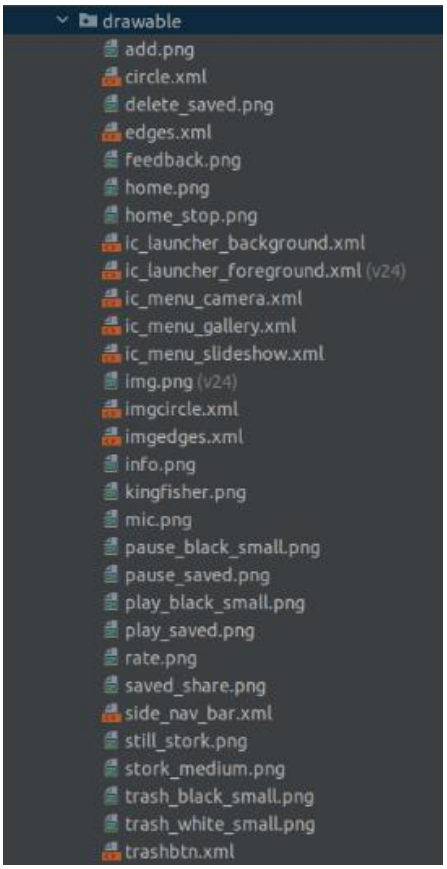
Κάθε αρχείο resource στο σύστημα διαθέτει μοναδικό αναγνωριστικό id, μέσω του οποίου μπορεί να αποκτήσει πρόσβαση ο κώδικας στο resource. Τα αναγνωριστικά ορίζονται από τον προγραμματιστή και καταγράφονται στην κλάση R, η οποία δημιουργείται αυτόματα από το Android Studio.

Ακολουθεί περιγραφή του κάθε υπό-φακέλου του res.

5.3.1 – drawable

Ένα drawable αναπαριστά μία εικόνα ή εικονίδιο που εμφανίζεται στην εφαρμογή. Μπορεί να ενσωματωθεί σε ένα άλλο resource, κυρίως σε layout, αλλά μπορεί να ανακτηθεί από τον κώδικα μέσω APIs όπως το getDrawable(id).

Συνήθως τα drawables συναντώνται σε πολλές διαφορετικές διαστάσεις και αναλύσεις εντός του φακέλου, έτσι ώστε το σύστημα να λαμβάνει κάθε φορά τη μορφή που είναι πιο ταιριαστή για την κάθε περίπτωση.

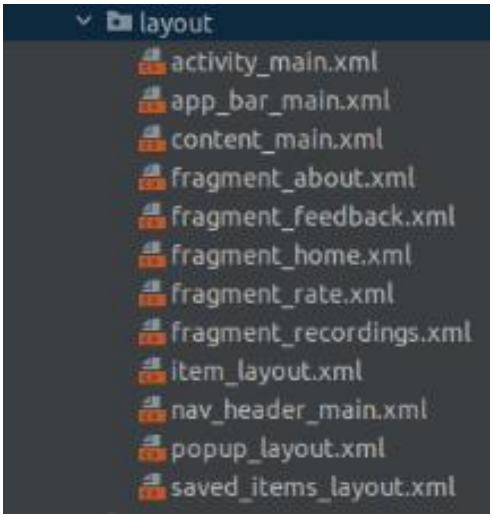


Εικόνα 5.12 - Τα περιεχόμενα του φακέλου «drawables»

Η BirdUp περιέχει άνω των 30 drawables τα οποία έχει λάβει από γνωστό ιστότοπο που τα παρέχει δωρεάν και σε ποικίλες αναλύσεις [81].

5.3.2 – layout

Τα layouts ή αλλιώς διαμορφώσεις οθόνης, είναι κατά βάση containers που συγκρατούν Views ή ViewGroups - κλάσεις στις οποίες υπάγονται - και ορίζουν τη διάταξη αυτών και των περιεχομένων τους. Διατίθενται για επεξεργασία μέσω κώδικα XML ή μέσω του γραφικού περιβάλλοντος «Layout Editor» του Android Studio. Για την BirdUp έχουν κατασκευαστεί 12 layouts τα οποία καλύπτουν όλη την οθόνη ή μέρος αυτής.



Εικόνα 5.13 - Τα περιεχόμενα του φακέλου «layout»

5.3.3 – menu

Τα αρχεία του φακέλου «menu» ορίζουν τα διαφορετικά μενού που έχει μία εφαρμογή, τις επιλογές τους και τις ομαδοποιήσεις τους, μαζί με τα εικονίδια κάθε επιλογής. Παράδειγμα αυτής της μορφής είναι το Κυρίως Μενού της BirdUp στην ακόλουθη εικόνα.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_home"
            android:icon="@drawable/home"
            android:title="Home" />
        <item
            android:id="@+id/nav_recordings"
            android:icon="@drawable/mic"
            android:title="Recordings" />
        <item
            android:id="@+id/nav_about"
            android:icon="@drawable/info"
            android:title="About" />
        <item
            android:id="@+id/nav_feedback"
            android:icon="@drawable/feedback"
            android:title="Feedback"/>
        <item
            android:id="@+id/nav_rate"
            android:icon="@drawable/rate"
            android:title="Rate App"/>
    </group>
</menu>
```

Εικόνα 5.14 - Κώδικας XML και προεπισκόπηση του Μενού

5.3.4 – mipmap

Περιλαμβάνει το app icon, ή αλλιώς το λογότυπο της εφαρμογής. Αρχικά, κάθε εφαρμογή Android έχει το προεπιλεγμένο icon της εικόνας 5.15, και ο προγραμματιστής είναι υπεύθυνος για την αντικατάσταση του.

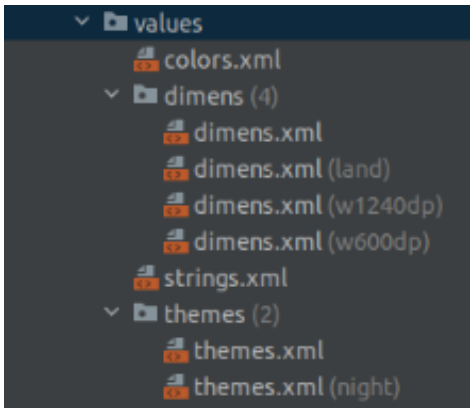
Εικόνα 5.15 - Android default app icon

5.3.5 – navigation

Αποτελείται από αρχεία XML που καθορίζουν την πλοήγηση μεταξύ των διαφόρων διεπαφών και επιλογών της εφαρμογής. Η πλοήγηση περιγράφεται με τη μορφή εντολών ή με γραφήματα πλοήγησης, τα οποία περιέχουν παράθυρα με όλες τις υπάρχουσες διεπαφές της εφαρμογής και βέλη που δείχνουν τις πιθανές κατευθύνσεις μεταξύ τους.

5.3.6 – values

Ο φάκελος «values» περιέχει συλλογές από διαφόρων ειδών τιμές, όπως κωδικούς χρωμάτων, διαστάσεις στοιχείων, strings και themes σε κώδικα XML. Οι τιμές αυτές μπορούν να χρησιμοποιηθούν μέσω των αναγνωριστικών τους, στον κώδικα ή σε αρχεία πόρων και να συμβάλλουν στην απόδοση ομοιομορφίας στο project. Η επεξεργασία των στοιχείων values εκτελείται όπου αυτά εφαρμόζονται, χωρίς την ανάγκη παρέμβασης από τον προγραμματιστή.



Εικόνα 5.16 - Τα περιεχόμενα του φακέλου «values»

Τα συνηθέστερα στοιχεία values είναι τα εξής:

- colors.xml: Συλλογή από χρώματα κωδικοποιημένα σε δεκαεξαδική μορφή. Το Android Studio διαθέτει και color editor για τη δημιουργία custom χρωμάτων και την ευκολότερη επεξεργασία τους.
- dimens: Συλλογή από διαστάσεις που αντιστοιχούν στα διάφορα στοιχεία που ορίζουν τα layouts της εφαρμογής. Στην εικόνα 5.16 οι παρενθέσεις δίπλα στα αρχεία dimens.xml αντιπροσωπεύουν διαφορετικά μεγέθη, πλάτους (width) 1240 και 600 dp (density-independent pixels) αντιστοίχως.
- strings.xml: Συλλογή από όλα τα strings που εμφανίζονται στην εφαρμογή, είτε αυτά είναι απλές λέξεις ή ολόκληρο κείμενο. Η συγκεκριμένη μέθοδος αποθήκευσης των strings διευκολύνει τη μετάφραση της εφαρμογής σε άλλες γλώσσες, αφού ο προγραμματιστής θα χρειαστεί να κάνει αλλαγές μόνο σε αυτό το αρχείο.
- themes: Συλλογή αρχείων XML που καθορίζουν την εμφάνιση του UI σε διαφορετικές πλατφόρμες και διαμορφώσεις. Τα κυριότερα αρχεία themes.xml είναι συνήθως δύο, ένα για Light Mode και ένα για Dark Mode, όπως φαίνεται στην 5.16. Ο προγραμματιστής μπορεί να κατασκευάσει themes μέχρι και για διαφορετικές εκδόσεις Android εάν το επιθυμεί.

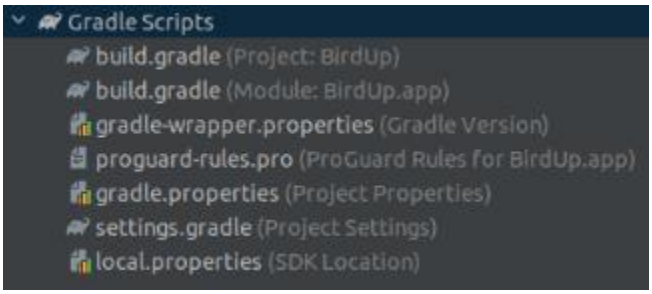
5.3.7 – xml

Εδώ γίνεται η αποθήκευση αυθαίρετων αρχείων XML για χρήσεις εκτός του layout. Η BirdUp χρησιμοποιεί το αρχείο provider_paths.xml, το οποίο παρέχει το path της εφαρμογής στον Content Provider μέσω του οποίου επιτρέπεται η κοινοποίηση ηχογραφήσεων.

5.4 Gradle Scripts

Τα αρχεία Gradle δημιουργούνται αυτόματα με την κατασκευή ενός νέου project από το Android Studio. Περιλαμβάνουν πληροφορίες σχετικές με το σύστημα, τις βιβλιοθήκες που

χρησιμοποιεί η εφαρμογή, και γενικότερα καθορίζουν την κατασκευή της με τη βοήθεια της επέκτασης Android Gradle. Από τα αρχεία της εικόνας 5.17, ο προγραμματιστής συνήθως καλείται να τροποποιήσει τα πρώτα δύο, συγκεκριμένα τα build.gradle(Project) build.gradle(Module). Για αυτά ακολουθεί σύντομη επεξήγηση.



Εικόνα 5.17 - Τα αρχεία του φακέλου «Gradle Scripts»

Εδώ να διευκρινιστεί πως ένα project του Android Studio μπορεί να διαθέτει περισσότερα από ένα modules, τα οποία αναλογούν σε διαφορετικές εκδοχές της εφαρμογής, η καθεμία από τις οποίες μπορεί να κάνει build, να εκτελεστεί στο σύστημα και να υποστεί αποσφαλμάτωση ξεχωριστά από τις άλλες [82]. Για αυτό, το Android Studio δημιουργεί αυτούς τους δύο διαφορετικούς τύπους αρχείων build.gradle:

- build.gradle (Project): Το top-level αρχείο κατασκευής, περιέχει πληροφορίες οι οποίες είναι κοινές για όλα τα modules του project [82].
- build.gradle (Module): Υπό του Project-level αρχείου, διαθέτει πληροφορίες αποκλειστικές για την κατασκευή ενός συγκεκριμένου module. Τέτοιες πληροφορίες και ρυθμίσεις είναι οι εξής:
    - Το όνομα πακέτου της εφαρμογής.
    - Η έκδοση της εφαρμογής.
    - Η παλαιότερη και νεότερη έκδοση συσκευής Android στην οποία μπορεί να λειτουργήσει η εφαρμογή.

- Plugins, τα οποία αντιπροσωπεύουν μία σειρά από διεργασίες που γίνονται κατά την κατασκευή του project. Αυτές περιλαμβάνουν τη μεταγλώττιση κλάσεων, τη δημοσίευση αρχείων κ.α. [83].
- Εξαρτήσεις της εφαρμογής, συμπεριλαμβανομένων εξωγενών βιβλιοθηκών και πακέτων που χρησιμεύουν στη μεταγλώττιση του κώδικα και την κάλυψη των αναγκών της.

Τα υπόλοιπα αρχεία του φακέλου ενημερώνονται αυτόματα με κάθε σχετική αλλαγή στο project και ως αποτέλεσμα δε χρήζουν συχνά τροποποίησης από τον προγραμματιστή [82].

ΚΕΦΑΛΑΙΟ 6

ΠΕΙΡΑΜΑΤΑ, ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Στο τελευταίο αυτό κεφάλαιο παρουσιάζονται τα πειραματικά αποτελέσματα των μοντέλων που εκπαιδεύτηκαν στο κεφάλαιο 2 μέχρι και το τελικό μοντέλο που είναι ενσωματωμένο στην εφαρμογή. Ακολουθούν γενικότερα συμπεράσματα μαζί με πιθανές μελλοντικές επεκτάσεις της BirdUp και κάποια κύρια bugs που εμφανίστηκαν κατά την κατασκευή της, μεταξύ των οποίων κάποια έχουν επιλυθεί ενώ άλλα όχι.

6.1 Πειράματα

Για να είμαστε σίγουροι για τις δυνατότητες ενός μοντέλου δε δύναται να βασιστούμε στα αποτελέσματα μονάχα από τις παραμέτρους accuracy και loss, πόσο μάλλον όταν τα δεδομένα μας χαρακτηρίζονται από ανισορροπία και περιορισμένα δείγματα. Για αυτόν το σκοπό, το κάθε μοντέλο ύστερα από την εκπαίδευση του φορτωνόταν σε ένα πρόγραμμα, μέσα από το οποίο έκανε προβλέψεις σε αφανή δεδομένα και υπολογιζόταν η επίδοση του μέσω πολυπλοκότερων παραμέτρων.

Οι παράμετροι που επιλέχθηκαν για τον επιπρόσθετο έλεγχο των μοντέλων ήταν οι:

- f1-score [84]
- MCC (Matthews Correlation Coefficient) [85]
- Πίνακας Σύγχυσης [86]
- Balanced Accuracy [87]
- Top-k Accuracy score [88]

Οι συγκεκριμένες παράμετροι έχουν ευρεία χρήση στη στατιστική και η παράλληλη χρήση τους εδώ αποσκοπεί στη συγκρότηση μίας ολοκληρωμένης άποψης για τις επιδόσεις του κάθε μοντέλου.

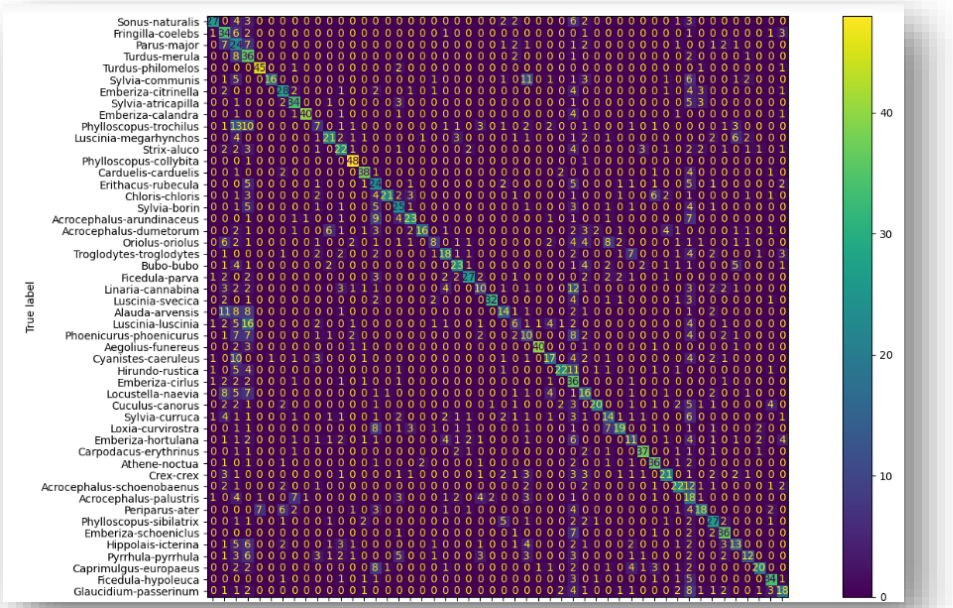
Έχοντας καταλήξει πλέον σε μία τοπολογία με πολύ καλά αποτελέσματα (εικόνα 2.22), αποφασίσθηκε να γίνει σύγκριση του ίδιου μοντέλου σε ελαφρώς διαφορετικά δεδομένα, για να εξετασθεί η επίδραση τους στην εκπαίδευση. Εκπαιδεύθηκαν τρία μοντέλα, όπου στις πρώτες δύο περιπτώσεις η τοπολογία ήταν η ίδια, ενώ στην τρίτη εισάχθηκε ακόμη ένα κρυφό στρώμα 256 νευρώνων για εμβάθυνση της πολυπλοκότητας. Η διαφορά στα δεδομένα ευθύνεται στην εκτέλεση της επαναδειγματοληψίας ύστερα από την προ επεξεργασία και ακριβώς πριν την κατασκευή των φασματογραφημάτων. Αυτό έγινε για να μην αφαιρεθεί καθόλου πληροφορία από τις αρχικές ηχογραφήσεις και να διατηρηθούν όλα τα χαρακτηριστικά που μπορεί να επηρεάσουν την εκπαίδευση.

Τα μοντέλα είναι τα Original (για το αρχικό), Resampled (για την επαναδειγματοληψία) και Extended (για το έξτρα στρώμα). Εκπαιδεύτηκαν χωρίς cross-validation και για τις προβλέψεις χρησιμοποιήθηκαν 50 δείγματα από κάθε είδος (2500 εικόνες συνολικά).

	Original	Resampled	Extended
f1 micro	0.481	0.51	0.474
f1 macro	0.483	0.517	0.486
f1 weighted	0.483	0.517	0.486
MCC	0.472	0.502	0.465
top-k	60.12%	62.48%	58. 48%
Bal_acc	48.08%	51%	47. 36%

Εικόνα 6.1 - Αποτελέσματα προβλέψεων στο 10% του Dataset

Οι παράμετροι top-k και Bal_acc αναλογούν σε ποσοστά με εύρος τιμών το 0% - 100%. Για τις παραμέτρους f1 και MCC το εύρος είναι [0, 1] και [-1, +1] αντίστοιχα. Τα αριστερά άκρα μαρτυρούν μοντέλα που δεν είναι ικανά να τοποθετήσουν καμία πρόβλεψη στη σωστή κατηγορία και τα δεξιά άκρα ανταποκρίνονται σε μοντέλα που εκτελούν σωστά κάθε πρόβλεψη. Η μέση τιμή 0 στην περίπτωση του MCC περιγράφει μοντέλα που παρέχουν τυχαίες προβλέψεις.

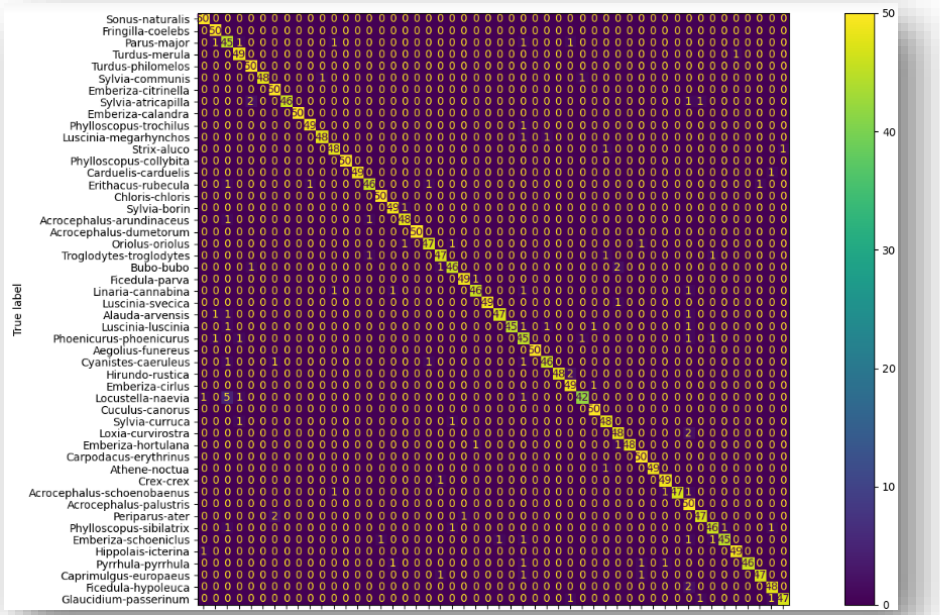


Εικόνα 6.2 - Πίνακας Σύγχυσης του μοντέλου "Resampled"

Τα πειραματικά αποτελέσματα των ίδιων μοντέλων για το τελικό Dataset είναι τα παρακάτω:

	Original	Resampled	Extended
f1 micro	0.88	0.958	0.91
f1 macro	0.88	0.958	0.91
f1 weighted	0.878	0.958	0.909
MCC	0.907	0.957	0.876
top-k	94.52%	98.08%	93%
Bal_acc	90.92%	95. 80%	87. 84%

Εικόνα 6.3 - Αποτελέσματα προβλέψεων με το τελικό Dataset



Εικόνα 6.4 - Πίνακας Σύγχυσης του μοντέλου "Original"

Σειρά έχουν τα αποτελέσματα των προβλέψεων στα μοντέλα με τον L2 regularizer:

	No Dropout	Dropout
f1 micro	0.752	0.764
f1 macro	0.755	0.77
f1 weighted	0.755	0.77
MCC	0.748	0.761
top-k	83.76%	85.92%
Bal_acc	75.16%	76. 44%

Εικόνα 6.5 - Αποτελέσματα προβλέψεων με L2 Regularizer

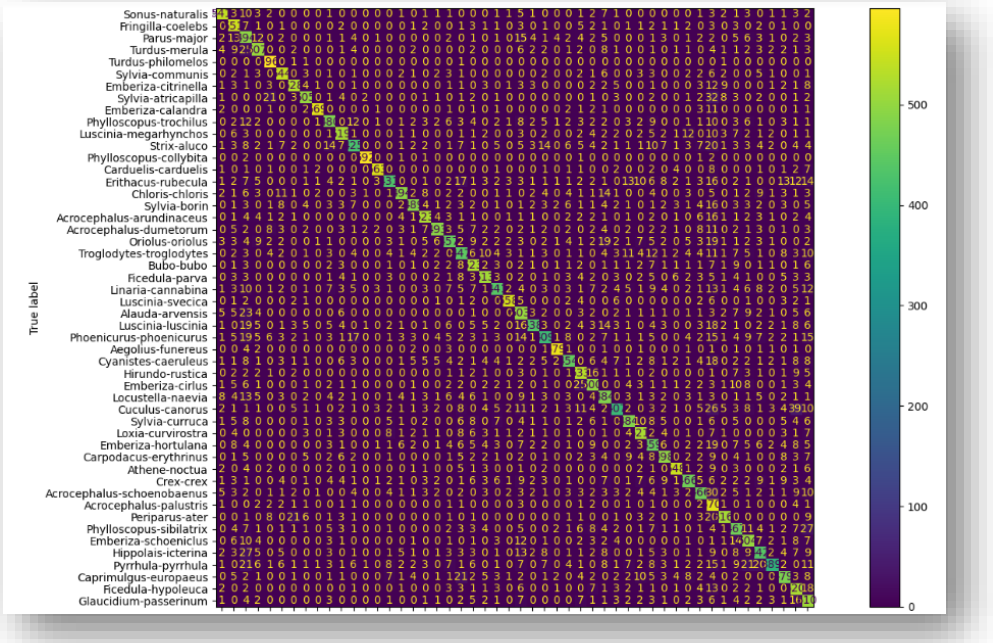
Συγκρίνοντας την απόδοση των μοντέλων πριν και μετά τη χρήση του νέου Dataset, καταλήγουμε στο συμπέρασμα πως η συλλογή νέων και περισσότερων δεδομένων ήταν η σωστή επιλογή. Υπάρχει τρομερή συνολική βελτίωση και στα τρία μοντέλα, με την πρωτιά να δίνεται στο «Resampled». Ωστόσο, εκεί που κέρδιζε σε ακρίβεια, το «Resampled» έχανε στην ταχύτητα της προ επεξεργασίας, σαφέστατα επειδή τα αρχεία καταλάμβαναν μεγαλύτερο χώρο και πληροφορία πριν την επαναδειγματοληψία τους.

Η ενσωμάτωση του μοντέλου και της προ επεξεργασίας στην εφαρμογή οδήγησε στο να μην επιλεχθεί το «Resampled» ως τελικό, για βελτίωση της εμπειρίας του χρήστη. Τη θέση του πήρε το γρηγορότερο μολονότι λιγότερο ακριβές «Original». Πριν την τελική ενσωμάτωση του στην εφαρμογή, το τελικό μοντέλο εκπαιδεύθηκε μία τελευταία φορά, με δείγματα του 1 second έναντι των αρχικών 5, ούτως ώστε να εξυπηρετείται η ηχογράφηση και ταξινόμηση δειγμάτων διάρκειας διάφορης των πολλαπλάσιων των 5 second. Η εκπαίδευση του πραγματοποιήθηκε σύμφωνα με τις οδηγίες εκπαίδευσης τελικών μοντέλων Μηχανικής Μάθησης στο [89].

Τα πειραματικά αποτελέσματα του «Original» σε άγνωστα δείγματα διάρκειας 1 second παρατίθενται παρακάτω:

	1 second
f1 micro	0.831
f1 macro	0.832
f1 weighted	0.832
MCC	0.828
top-k	89.77%
Bal_acc	83.10%

Εικόνα 5.6- Αποτελέσματα προβλέψεων σε δείγματα 1 second



Εικόνα 6.6 - Πίνακας σύγχυσης μοντέλου «Original» για δείγματα 1 second

Η μειωμένη απόδοση του μοντέλου «Original» στα δείγματα του 1 second είναι αισθητή, ωστόσο η αλλαγή αυτή βελτιώνει κατά πολύ την ευελιξία της εφαρμογής με τις ηχογραφήσεις και συνεπώς και την εμπειρία του χρήστη. Από την άλλη, το πρόβλημα της μειωμένης απόδοσης μπορεί να αντιμετωπισθεί εύκολα με την εκπαίδευση νέων, βαθύτερων μοντέλων στο μέλλον, σε συσκευή με ισχυρότερες προδιαγραφές και μεγαλύτερες δυνατότητες.

6.2 Συμπεράσματα

Σκοπός της παρούσας διπλωματικής ήταν ο συνδυασμός εργαλείων και γνώσεων από διαφορετικούς τομείς προγραμματισμού για την επίτευξη ενός βασικού στόχου. Το αντικείμενο του προγραμματιστή πολλές φορές απαιτεί γνώσεις και ικανότητες εργαλείων και πεδίων που με πρώτη ματιά δε φαίνεται να σχετίζονται, εάν όμως συνδυαστούν κατάλληλα μπορούν να φέρουν εις πέρας κάθε λογής έργο. Η κατοχή τέτοιων ποικίλων γνώσεων προσφέρει στον προγραμματιστή μία ευελιξία που αποτελεί σημαντικό προσόν στην επαγγελματική του ζωή αλλά και στην ενασχόληση με προσωπικά projects. Αυτός ήταν και ο κύριος λόγος επιλογής αυτού του θέματος και της μεθόδου που ακολουθήθηκε, μία τελευταία ευκαιρία εκμάθησης νέων γνώσεων και ενίσχυσης του γνωστικού υποβάθρου πριν την αρχή της επαγγελματικής σταδιοδρομίας.

Προσωπικά, ο παραπάνω στόχος επιτεύχθηκε και έφερε μεγάλη εξέλιξη στον προγραμματιστή, τόσο σε επίπεδο απόκτησης γνώσεων και εμπειρίας όσο και στην εφαρμογή του άνωθεν σκεπτικού. Παρότι η εφαρμογή προγραμματιστικά δεν πήρε ιδιαίτερα ρίσκα και εστίασε στην επίτευξη βασικών λειτουργιών, θέτει τα θεμέλια για τη μελλοντική δημιουργία άλλων, ισχυρότερων και πιο ολοκληρωμένων εφαρμογών, εμπορικών και μη.

6.3 Μελλοντικές Επεκτάσεις

Η BirdUp κατασκευάστηκε με τις κύριες λειτουργίες που επιτρέπουν στο χρήστη να πετύχει το στόχο του, αυτό όμως δε συνεπάγεται πως δεν υπάρχουν περιθώρια για εξέλιξη. Ακόμα, οι δυσκολίες που εμφανίστηκαν κατά τον προγραμματισμό της δεν επέμειναν για μεγάλο χρονικό διάστημα, με μικρές εξαιρέσεις. Όπως ήταν αναμενόμενο λόγω και της έλλειψης γνώσεων στην κατασκευή εφαρμογών, η διαδικασία κατασκευής της BirdUp απεδείχθη χρονοβόρα, με αποτέλεσμα να ήταν ανέφικτη η παροχή επιπρόσθετων λειτουργιών και η επίλυση όλων των εμφανιζόμενων σφαλμάτων στο υπάρχον χρονικό διάστημα. Τα σφάλματα και οι περαιτέρω λειτουργίες αναφέρονται στις επόμενες παραγράφους.

6.3.1 – Bugs

Τα σημεία στα οποία εμφανίστηκαν σφάλματα και απρόσμενα αποτελέσματα στην εφαρμογή, είτε αυτά επιλύθηκαν είτε όχι, είναι τα εξής:

- Η προαναφερθείσα αδυναμία χρήσης αρχείων .wav και .mp3 κατευθείαν μέσω του κώδικα της εφαρμογής. Παρά την υποστήριξη των παραπάνω μορφών και την πιστή υπακοή στα απαραίτητα βήματα για τη σωστή χρήση τους, η ηχογράφηση δεν ολοκληρωνόταν σωστά και τερμάτιζε απροσδόκητα την εφαρμογή. Το πρόβλημα αυτό επιλύθηκε με την προσθήκη του λογισμικού ffmpeg, αφού απασχόλησε τον προγραμματιστή για αρκετές μέρες.
- Αδυναμία κοινοποίησης των αποθηκευμένων ηχογραφήσεων. Η λειτουργία αυτή προστέθηκε προς το τέλος της κατασκευής και ακολούθησε τα απαραίτητα βήματα για την κοινοποίηση αρχείων, όπως την κατασκευή αρχείου provider_paths.xml και την παροχή Content Provider στο μανιφέστο της εφαρμογής. Ύστερα από έρευνα στο διαδίκτυο, παρατηρήθηκε πως το παρόν πρόβλημα ήταν συχνό σε συσκευές Android 11, χωρίς την ύπαρξη καθολικής λύσης έως τώρα.

- Ορισμένες φορές όπου υπάρχει μία μόνο αποθηκευμένη ηχογράφηση στην εφαρμογή, η αποθήκευση νέας ηχογράφησης αντικαθιστά την παλιά ηχογράφηση με τη νέα. Παρόμοια συμπεριφορά δεν παρατηρήθηκε στην ύπαρξη πολλών αποθηκευμένων ηχογραφήσεων.
- Τα φασματογραφήματα που παράγονται από την εφαρμογή έχουν μόνο μαύρα pixels για μερικά κλάσματα του δευτερολέπτου στην αρχή της ηχογράφησης. Μολονότι η συμπεριφορά αυτή δεν ήταν προσδοκώμενη, δεν προκάλεσε προβλήματα στη διενέργεια σωστών προβλέψεων.

6.3.2 – Έξτρα Λειτουργίες

Οι λειτουργίες που μπορούν να προστεθούν σε μελλοντικές εκδοχές της εφαρμογής είναι πάρα πολλές. Πολλές από αυτές προέρχονται από ήδη υπάρχουσες εφαρμογές του χώρου, όπως την BirdNet, στην οποία βασίστηκε εν μέρει η BirdUp, ενώ άλλες ήταν προϊόν σκέψης του προγραμματιστή. Οι λειτουργίες αυτές απαριθμούνται εδώ:

- Επέκταση του Dataset. Το αρχικό πλήθος των 50 ειδών επιλέχθηκε μεν για να αποφευχθεί όσο το δυνατόν περισσότερο η καταπόνηση του συστήματος και η καθυστέρηση της εκπαίδευσης, αλλά και ως ένας αποδεκτός αριθμός ειδών για καλές προβλέψεις σε μικρότερη κλίμακα, ώστε να καθιστά την εφαρμογή πρακτική. Το Dataset μπορεί να επεκταθεί αρχικά στα 80 ή 100 είδη, με αυτό να συνεπάγεται μεγαλύτερους χρόνους εκπαίδευσης και αύξηση της πολυπλοκότητας του μοντέλου.
- Ενσωμάτωση δεδομένων τοποθεσίας. Η χρήση της γεωγραφικής τοποθεσίας του χρήστη για τη διενέργεια προβλέψεων ενισχύει σημαντικά τις δυνατότητες πρόβλεψης τέτοιων εφαρμογών, όπως επιτυγχάνεται και στο BirdNet. Είναι αδιαμφισβήτητο πως θα ενισχύσει τα αποτελέσματα και της BirdUp, καθώς οι πληθυσμοί ενός μέρους εξαρτώνται σημαντικά από την εποχή του έτους και της καιρικές συνθήκες στη γύρω περιοχή.
- Προσθήκη Hardware Acceleration στο μοντέλο για τη μείωση του χρόνου προβλέψεων.

- Βελτίωση της μεθόδου αφαίρεσης θορύβου από τις ηχογραφήσεις. Συγκεκριμένα, έχει ήδη εξεταστεί η προσθήκη του πακέτου noisereduce της Python [90], το οποίο χρησιμοποιεί φασματικές πύλες για την αφαίρεση θορύβου και παρουσιάζει πολύ θετικά αποτελέσματα.
- Ανακατεύθυνση στη σελίδα του πτηνού στη Βικιπαίδεια με παρατεταμένο πάτημα στην αποθηκευμένη πρόβλεψη. Με αυτή τη λειτουργία ο χρήστης θα μπορεί εύκολα να βρει πληροφορίες για τα είδη που έχει προβλέψει και θα ενισχύεται παράλληλα η εμπειρία και η φιλικότητα της εφαρμογής.
- Προσθήκη εικόνων του αντίστοιχου είδους στα αποτελέσματα της πρόβλεψης και στα αποθηκευμένα αρχεία, αντί του εικονιδίου της εφαρμογής. Δε συμπεριλήφθηκε στην αρχική έκδοση της εφαρμογής λόγω της χειρωνακτικής δουλείας που απαιτούσε και την εύρεση ξεκάθαρων εικονιδίων μικρών διαστάσεων.
- Διόρθωση της λειτουργίας κοινοποίησης για τη σωστή κοινοποίηση των αποθηκευμένων προβλέψεων.
- Εκπαίδευση του μοντέλου με δείγματα ανθρώπινης ομιλίας. Πολλές φορές στο εξωτερικό περιβάλλον ο θόρυβος του ανθρώπινου παράγοντα επικαλύπτει την πηγή του τραγουδιού, με αποτέλεσμα το μοντέλο να δίνει False Positives που δυσχεραίνουν την εμπειρία του χρήστη και μειώνουν την αξιοπιστία των προβλέψεων. Η προσθήκη αυτή είναι απαραίτητη για να αποφεύγεται το συχνό αυτό φαινόμενο.
- Κβαντοποίηση του μοντέλου tflite. Αυτή η τεχνική μετατροπής πετυχαίνει μείωση του μεγέθους του μοντέλου, βελτιώνοντας παράλληλα τις καθυστερήσεις του CPU και του Hardware Accelerator, με μικρή υποβάθμιση στην ακρίβεια του μοντέλου [91]. Λόγω της τελευταίας παρενέργειας, η προσθήκη αυτή θα εξετασθεί εφόσον το μοντέλο κάνει σταθερά πολύ εύστοχες προβλέψεις, ούτως ώστε να μην επηρεαστεί αισθητά η ακρίβεια του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://ebird.org/home

[2] https://merlin.allaboutbirds.org/

[3] https://www.birds.cornell.edu/home/

[4] https://birdnet.cornell.edu/

[5] https://www.audubon.org/app

[6] http://www.chirpomatic.com/

[7] http://www.songsleuth.com/#/

[8] https://flyinglessons.us/2020/12/22/which-is-the-best-birdsong-id-app-we-tested-them-and-have-a-winner/

[9] https://xeno-canto.org/

[10] https://en.wikipedia.org/wiki/Spectrogram

[11] https://en.wikipedia.org/wiki/Artificial_neural_network

[12] https://www.ranorex.com/blog/ai-code-review/#:~:text=Applying%20AI%20to%20code%20reviews,apply%20AI%20in%20code%20reviews.

[13] Abdous, Kamel. (2020). Automatic Code Optimization With Machine Learning And Combinatorial Optimization. 10.13140/RG.2.2.20966.24647/1.

[14] https://www.augmenta.ag/

[15] https://centaur.ag/centaur-technologies/

[16] https://www.v7labs.com/blog/neural-network-architectures-guide#:~:text=The%20Neural%20Network%20architecture%20is,various%20components%20of

%20a%20neuron.&text=Input%20%2D%20It%20is%20the%20set,model%20for%20the%20learning%20process.

[17] https://en.wikipedia.org/wiki/Recurrent_neural_network

[18] https://en.wikipedia.org/wiki/Long_short-term_memory

[19] Stowell, D. & Plumbley, M. D. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ* **2**, e488 (2014)

[20] Stevens, S. S., Volkmann, J. & Newman, E. B. A scale for the measurement of the psychological magnitude pitch. *J. Acoust. Soc. Am.* **8**, 185–190 (1937)

[21] https://en.wikipedia.org/wiki/Zero-crossing_rate

[22] Graves, A., Mohamed, A.R., and Hinton, G. Speech recognition with deep recurrent neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013), pp. 6645-6649. (2013)

[23] Sainath, T.N., Vinyals, O., Senior, A. and Sak, H., Convolutional, long short-term memory, fully connected deep neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015), pp. 4580-4584. (2015)

[24] https://en.wikipedia.org/wiki/Random_forest

[25] Leng, Y.R. and Dat, T.H., December. Multi-label bird classification using an ensemble classifier with simple features. Asia Pacific Signal and Information Processing Association (APSIPA), pp. 1-5. (2014)

[26] Neal, L., Briggs, F., Raich, R. and Fern, X.Z., Time-frequency segmentation of bird song in noisy acoustic environments. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011), pp. 2012-2015. (2011)

[27] J. Xie, M. Zhu, Handcrafted features and late fusion with deep learning for bird sound classification, Ecological Informatics 52 (2019) 74–81.

[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818– 2826

[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017)

[31] J. Schluter, Bird identification from timestamped, geotagged audio recordings., in: CLEF (Working Notes), 2018.

[32] Á. Incze, H. Jancsó, Z. Szilágyi, A. Farkas and C. Sulyok, "Bird Sound Recognition Using a Convolutional Neural Network," 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), 2018, pp. 000295-000300, doi: 10.1109/SISY.2018.8524677.

[33] https://www.kaggle.com/datasets/monogenea/birdsongs-from-europe

[34] https://marce10.github.io/warbleR/

[35] https://en.wikipedia.org/wiki/Bird_vocalization

[36] https://www.wildlifeacoustics.com/uploads/video-downloads/Video_Script_SM4_Audio_Settings.pdf

[37] https://librosa.org/doc/latest/generated/librosa.stft.html

[38] https://en.wikipedia.org/wiki/Hyperparameter_optimization

[39] https://en.wikipedia.org/wiki/Transfer_learning

[40] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[41] https://machinelearningmastery.com/k-fold-cross-validation/

[42] https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/

[43] https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

[44] https://www.kaggle.com/code/gpreda/download-birds-songs-recording-metadata/notebook

[45] https://developer.android.com/reference/android/media/MediaRecorder

[46] https://www.iis.fraunhofer.de/en/ff/amm/communication/aaceld.html

[47] https://www.tensorflow.org/lite

[48] https://www.tensorflow.org/lite/models/convert

[49] https://www.tensorflow.org/api_docs/python/tf/lite/Interpreter

[50]https://www.tensorflow.org/lite/api_docs/java/org/tensorflow/lite/support/image/ImageProcessor.Builder

[51] https://developer.android.com/guide/topics/ui/dialogs

[52] https://en.wikipedia.org/wiki/Android_(operating_system)

[53] https://en.wikipedia.org/wiki/HTC_Dream

[54] https://www.android.com/gms/

[55] https://www.appbrain.com/stats/number-of-android-apps

[56] https://developer.android.com/guide/components/fundamentals

[57] https://developer.android.com/studio/index.html

[58] https://developer.android.com/studio/releases#older-releases

[59] https://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/

[60] https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?guccounter=1

[61] https://www.androidauthority.com/develop-android-apps-languages-learn-391008/

[62] https://en.wikipedia.org/wiki/Apk_(file_format)

[63] https://devopedia.org/kotlin-language

[64] https://en.wikipedia.org/wiki/FFmpeg

[65] https://chaquo.com/chaquopy/

[66] https://en.wikipedia.org/wiki/Conda_(package_manager)

[67] https://www.tensorflow.org/

[68] https://en.wikipedia.org/wiki/TensorFlow

[69] https://keras.io/

[70] https://en.wikipedia.org/wiki/Keras

[71] https://github.com/librosa/librosa

[72] https://github.com/jiaaro/pydub

[73] https://numpy.org/

[74] https://matplotlib.org/

[75] https://en.wikipedia.org/wiki/Matplotlib

[76] https://pandas.pydata.org/

[77] https://en.wikipedia.org/wiki/Pandas_(software)

[78] https://scikit-learn.org/stable/

[79] https://en.wikipedia.org/wiki/Scikit-learn

[80] https://opencv.org/

[81] https://icons8.com/icons/set/android

[82] https://developer.android.com/studio/build

[83] https://docs.gradle.org/current/userguide/tutorial_using_tasks.html#tutorial_using_tasks

[84] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[85] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

[86] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[87] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html

[88] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.top_k_accuracy_score.html

[89] https://machinelearningmastery.com/train-final-machine-learning-model/

[90] https://pypi.org/project/noisereduce/

[91] https://www.tensorflow.org/lite/performance/post_training_quantization

[92] https://en.wikipedia.org/wiki/API

[93] https://en.wikipedia.org/wiki/Comma-separated_values

[94] https://en.wikipedia.org/wiki/Integrated_development_environment

[95] https://en.wikipedia.org/wiki/XML

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ Α

Συντομογραφίες

- ANN – Artificial Neural Network
- API – Application Interface: Μια διεπαφή προγραμματισμού εφαρμογών είναι ένας τρόπος για δύο ή περισσότερα προγράμματα υπολογιστών να επικοινωνούν μεταξύ τους. Είναι ένας τύπος διεπαφής λογισμικού, που προσφέρει υπηρεσία σε άλλα κομμάτια λογισμικού [92].
- APK – Android Package
- CNN – Convolutional Neural Networks
- CSV – Comma Separated Values: Ένα οριοθετημένο αρχείο κειμένου που χρησιμοποιεί κόμμα για να διαχωρίσει τιμές. Κάθε γραμμή του αρχείου είναι μια εγγραφή δεδομένων. Κάθε εγγραφή αποτελείται από ένα ή περισσότερα πεδία, χωρισμένα με κόμμα. Αποθηκεύει συνήθως δεδομένα πίνακα (αριθμούς και κείμενο) σε απλό κείμενο, οπότε κάθε γραμμή θα έχει τον ίδιο αριθμό πεδίων [93].
- DNN – Deep Neural Networks
- FFMPEG – Fast Forward MPEG
- FFT – Fast Fourier Transform
- GAN – General Adversarial Networks
- IDE – Integrated Development Environment: Εφαρμογή λογισμικού που παρέχει ολοκληρωμένες διευκολύνσεις σε προγραμματιστές υπολογιστών για ανάπτυξη λογισμικού. Ένα IDE συνήθως αποτελείται από τουλάχιστον έναν επεξεργαστή πηγαίου κώδικα, εργαλεία αυτοματισμού κατασκευής και έναν εντοπισμό σφαλμάτων [94].
- JPG – Joint Photographic Experts Group
- LPC – Linear Prediction Coefficients
- LSTM – Long Short-Term Memory
- MFCC – Matthews Correlated Coefficient
- MLP – Multi-Layer Perceptron

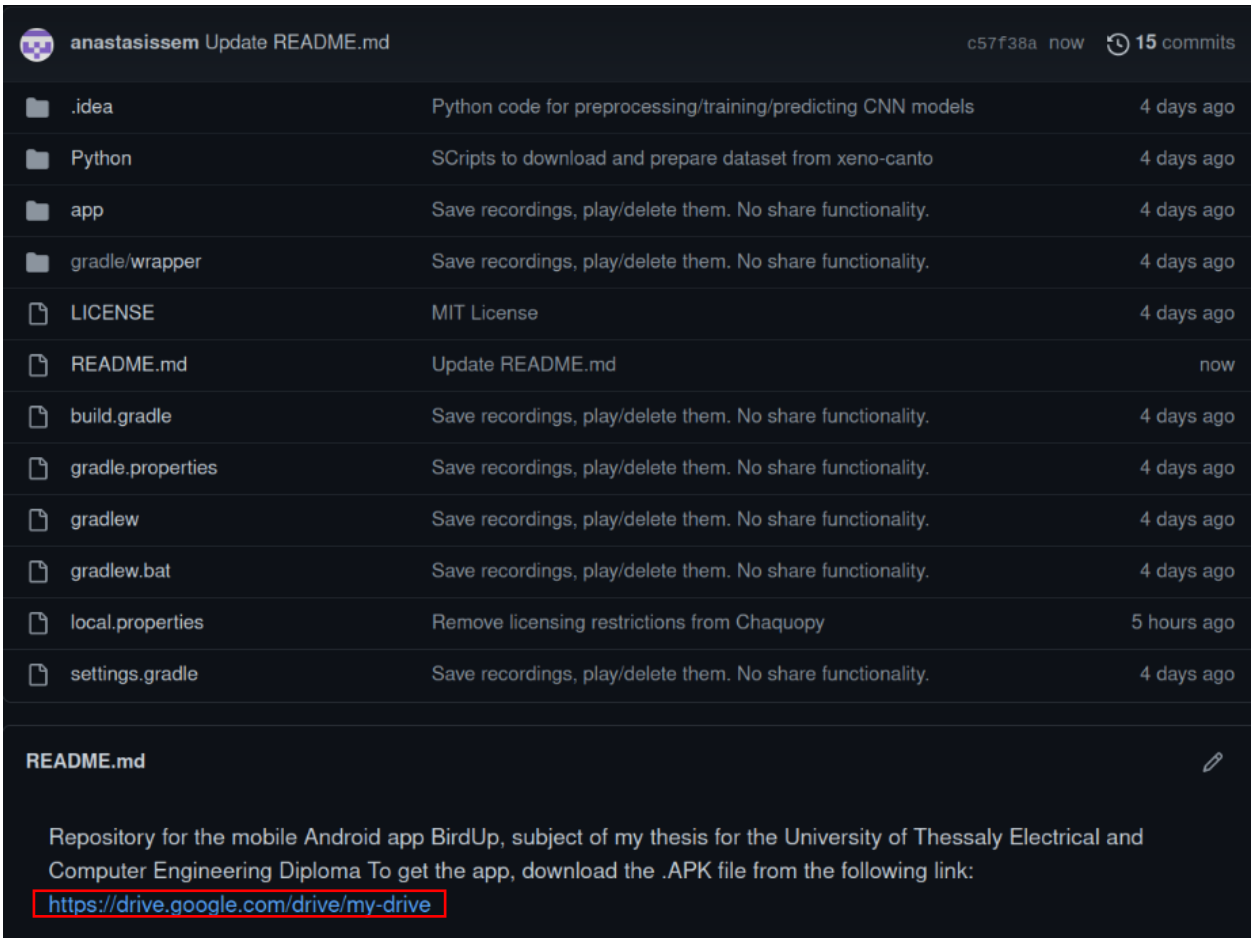
- PNG – Portable Network Graphics
- RNN – Recurrent Neural Networks
- SED – Sound Event Detection
- STFT – Short-Time Fourier Transform
- UI – User Interface
- WAV – Waveform Audio File Format
- XML – Extensible Markup Language: Γλώσσα σήμανσης και μορφή αρχείου για την αποθήκευση, τη μετάδοση και την ανακατασκευή αυθαίρετων δεδομένων. Ορίζει ένα σύνολο κανόνων για την κωδικοποίηση εγγράφων σε μορφή που είναι τόσο αναγνώσιμη από τον άνθρωπο όσο και από μηχανή [95].

ΠΑΡΑΡΤΗΜΑ Β

Οδηγίες εγκατάστασης της εφαρμογής

Ο πηγαίος κώδικας της BirdUp βρίσκεται στο GitHub, στον ακόλουθο σύνδεσμο:

https://github.com/anastasissem/Android-BirdUp



Στο αρχείο README.md του project παρέχεται το link για το Google Drive, όπου εκεί είναι ανεβασμένο το αρχείο app-debug.apk, το οποίο μπορεί κανείς να το κατεβάσει, και ανοίγοντας το, να εγκαταστήσει την εφαρμογή στην προσωπική Android συσκευή του.