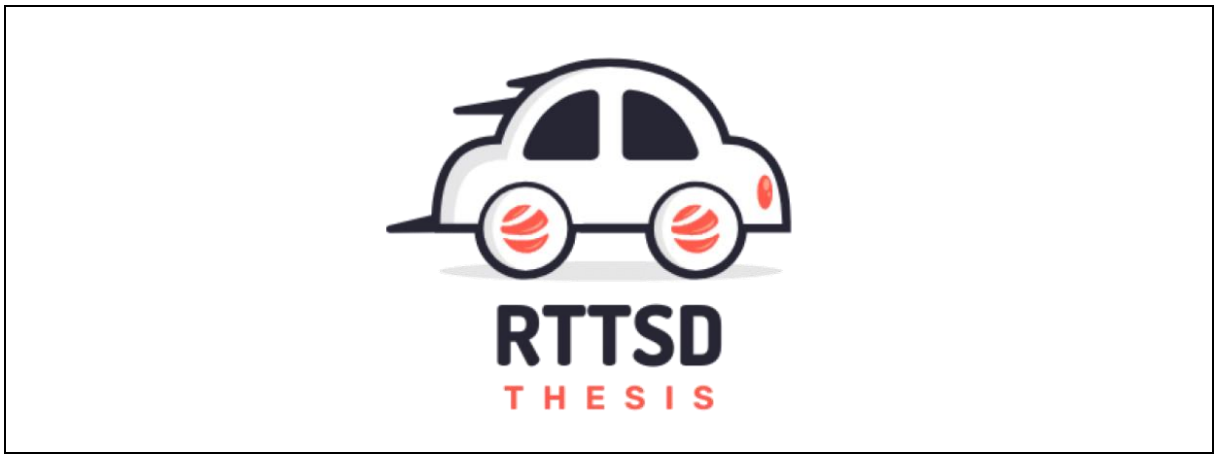
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή Android για την αναγνώριση οδικών σημάτων»



Των φοιτητών
Γεωργιάδη Πολυχρόνη
Ζαχαράκη Κωνσταντίνου
Αρ. Μητρώου: 164648, 164663

Επιβλέπων
Δρ. Διαμαντάρας Κωνσταντίνος
Καθηγητής

Ημερομηνία 6/2/2022

Τίτλος Δ.Ε. Εφαρμογή Android για την αναγνώριση οδικών σημάτων
Κωδικός Δ.Ε. 21245
Ονοματεπώνυμο φοιτητών
Ζαχαράκης Κωνσταντίνος & Γεωργιάδης Πολυχρόνης
Ονοματεπώνυμο εισηγητή Διαμαντάρα Κωνσταντίνος
Ημερομηνία ανάληψης Δ.Ε. 02-04-2021
Ημερομηνία περάτωσης Δ.Ε. 06-02-2022

Βεβαιώνουμε ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Στους αγαπημένους μας»

Πρόλογος

Το ενδιαφέρον που μας ένωσε σε αυτή τη σχολή και η θέληση μας για να αποκτήσουμε νέες γνώσεις και να εξελιχθούμε, συνεχίζει να καλλιεργείται και να ανθίζει μέσα μας. Ένας από τους σπόρους του μας οδήγησε στην απόφαση να ασχοληθούμε με το θέμα αυτής της διπλωματικής. Μέσα από την ανάγκη μας για εξέλιξη, ανακαλύψαμε τις προοπτικές των αλγορίθμων μηχανικής μάθησης, την δυνατότητα να δημιουργούμε εφαρμογές για κινητές συσκευές και την αξία του να συναντάς προβλήματα, να τα επιλύεις και να μαθαίνεις μέσα από αυτά. Στόχος μας, ήταν να φτιάξουμε μία εφαρμογή χρήσιμη για πολλούς συνανθρώπους μας και να βάλουμε το λιθαράκι μας στην βελτίωση της οδηγικής κοινότητας. Όλα όσα συναντήσαμε κατά τη διάρκεια αυτών των μηνών, μας πρόσφεραν διδάγματα σε ακαδημαϊκό, αλλά και προσωπικό επίπεδο και να μας βοηθήσουν να γίνουμε καλύτεροι.

Περίληψη

Σκοπός της διπλωματικής εργασίας μας είναι η δημιουργία μίας εφαρμογής για κινητά τηλέφωνα, η οποία με τη χρήση μοντέλων μηχανικής μάθησης θα μπορεί να αναγνωρίσει σήματα οδικής κυκλοφορίας σε ευρωπαϊκούς δρόμους. Καταγράψαμε πολλές παρόμοιες προσπάθειες, οι οποίες ακολούθησαν διάφορες μεθοδολογίες και τεχνικές που μας βοήθησαν να καταλήξουμε στη δική μας λύση. Αποφασίσαμε να χρησιμοποιήσουμε ένα σύνολο εικόνων που διανέμεται από την ομάδα του Mapillary, το οποίο περιέχει πάνω από 50.000 εικόνες που θα χρησιμοποιήσουμε στην εκπαίδευση δύο μοντέλων μηχανικής μάθησης. Το πρώτο, είναι ένα μοντέλο ανίχνευσης αντικειμένων YOLOv5 που επεξεργάζεται τις εισόδους που του δίνονται και επιστρέφει στην έξοδο την τοποθεσία και την κλάση από τα σήματα που αναγνωρίζει. Έπειτα, παίρνει θέση το δεύτερο μοντέλο, ένας ταξινομητής, ο οποίος θα ελέγχει σε ποια κλάση ανήκει το κάθε αντικείμενο που του δόθηκε. Με το συνδυασμό των παραπάνω αλγορίθμων, θα υλοποιήσουμε μία εφαρμογή για κινητά τηλέφωνα στο λειτουργικό σύστημα Android, η οποία θα εξάγει στιγμιότυπα με τη χρήση της κάμερας του τηλεφώνου, θα τα περνάει μέσα από τους δύο αλγορίθμους και θα εμφανίζει στο χρήστη τα σήματα οδικής κυκλοφορίας που αναγνώρισε στο δρόμο. Επίσης, η εφαρμογή θα παρέχει τη δυνατότητα φωνητικών ενημερώσεων για τα αποτελέσματα που ανίχνευσε και για να προειδοποιήσει τον χρήστη για κάποια παράβαση ταχύτητας. Στα παρακάτω κεφάλαια θα δούμε δείγματα από τις εφαρμογές άλλων χρηστών, ποιες μεθοδολογίες και εργαλεία χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής, την ανάλυση σε όλα τα στάδια της δημιουργίας της και συμπεράσματα που βγάλαμε από το τελικό αποτέλεσμα.

«Android application for road traffic sign detection»

« Polychronis Georgiadis & Konstantinos Zacharakis »

Abstract

The aim of our thesis is to create a mobile phone application, which, using machine learning models, will be able to recognize traffic signs on European roads. We documented many similar efforts, which followed different methodologies and techniques that helped us to come up with our solution. We decided to use a set of images distributed by the Mapillary team, which contains over 50,000 images that we will use in training two machine learning models. The first, is a YOLOv5 object detection model that processes the inputs given to it and returns to the output the location and class from the recognized sign. Then, the second model, a classifier, takes over, which will check which class each object given to it belongs to. By combining the above algorithms, we will implement a mobile phone application on the Android operating system, which will extract snapshots using the phone's camera, pass them through the two algorithms, and display to the user the road signs it recognized on the road. The app will also provide the option of voice notifications of the results it detected and to warn the user of a speeding violation. In the following chapters we will look at samples of other users' applications, what methodologies and tools we used to implement at our application, the analysis at all stages of its creation and conclusions drawn from the final result.

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τους φίλους μας και τις οικογένειες μας για την συμπαράσταση κατά της διάρκεια της εκπόνησης της διπλωματικής, και τον επιβλέπον καθηγητή μας κ. Διαμαντάρα Κωνσταντίνο για τις πολύτιμες γνώσεις και τους πόρους που μας παρείχε και σε όλο αυτό το ταξίδι.

Περιεχόμενα

Πρόλογος 5
Περίληψη 6
Abstract 7
Ευχαριστίες 8
Περιεχόμενα 9
Κατάλογος Σχημάτων 13
Κατάλογος Πινάκων 15
Συντομογραφίες 16
Κεφάλαιο 1ο: Εισαγωγή 1
1.1 Εισαγωγή 1
1.2 Στόχος Διπλωματικής 2
1.3 Δομή Διπλωματικής 2
Κεφάλαιο 2ο: Βιβλιογραφική ανασκόπηση και ιστορικό υπόβαθρο 3
2.1 Ιστορικό υπόβαθρο 3
2.2 Ενσωματωμένα συστήματα αυτόματης οδήγησης 3
2.2.1 Σύστημα αυτόνομης οδήγησης – Tesla, Inc 3
2.2.2 Σύστημα αναγνώρισης σημάτων κυκλοφορίας - Toyota 4
2.3 Εφαρμογές αναγνώρισης αντικειμένων 4
2.3.1 TensorFlow Object Detection App 4
2.3.2 Car Assistant Android 5
2.3.3 Traffic Light Detection and Color Recognition 5
2.4 Σύνοψη 5
Κεφάλαιο 3ο: Θεωρητικό υπόβαθρο 7
3.1 Μηχανική Μάθηση (Machine Learning) 7
3.1.1 Εισαγωγή 7
3.1.2 Νευρωνικά δίκτυα (Neural Networks) 7
3.1.2.1 Νευρώνας McCulloch-Pitts 8
3.1.2.2 Νευρώνας Perceptron 9
3.1.3 Βαθιά Μάθηση (Deep Learning) 10
3.1.3.1 Μάθηση με επίβλεψη 10
3.1.3.2 Μάθηση χωρίς επίβλεψη 11
3.1.3.3 Μάθηση με ενίσχυση 11

3.1.3.4 Ρυθμός Εκπαίδευσης 11
3.1.3.5 Αρχικοποίηση Βαρών 12
3.1.3.6 Overfitting 12
3.1.3.7 Dropout 12
3.1.3.8 Εξαγωγή Συμπερασμάτων 12
3.1.4 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) 13
3.1.4.1 Συνελικτικό στρώμα 14
3.1.4.2 Στρώμα υποδειγματοληψίας 16
3.1.4.3 Πλήρως συνδεδεμένο στρώμα 17
3.1.4.4 Τύποι και εφαρμογές συνελικτικών νευρωνικών δικτυών 18
3.2 You Only Look Once (YOLO) 29
3.3 PyTorch 45
3.4 Keras 46
3.5 TensorFlow 48
3.5.1 Εισαγωγή 48
3.5.2 TensorFlow 1.0 49
3.5.3 TPU 49
3.5.4 CUDA 49
3.5.5 SYCL 49
3.5.6 Edge Computing 50
3.5.7 SavedModel 50
3.5.8 TensorFlow 2.0 50
3.5.9 TensorFlow Intepreter 51
3.5.10 TensorFlow Lite 51
3.5.11 Κβαντισμός (Quantization) 52
3.5.11.1 Κβάντιση float16 μετά την εκπαίδευση 52
3.5.11.2 Κβαντισμός δυναμικού εύρους μετά την εκπαίδευση 53
3.5.11.3 Ακέραιος κβαντισμός μετά την εκπαίδευση 53
3.5.11.4 Εκπαίδευση με γνώση κβαντισμού 54
3.6 Android 54
3.6.1 Εισαγωγή στο Android 54
3.6.2 Τα βασικά στοιχεία του Android 54
3.6.3 Οι εκδόσεις του Android 54
3.6.3.1 Alpha (Android 1.0) 54
3.6.3.2 Beta (Android 1.1) 55

3.6.3.3 Cupcake (Android 1.5) 55
3.6.3.4 Donut (Android 1.6) 55
3.6.3.5 Eclair (Android 2.0 – 2.1) 55
3.6.3.6 Froyo (Android 2.2) 55
3.6.3.7 Gingerbread (Android 2.3-2.3.7) 56
3.6.3.8 Honeycomb (Android 3.0 – 3.2.6) 56
3.6.3.9 Ice Cream Sandwich (Android 4.0 – 4.0.4) 56
3.6.3.10 Jelly Bean (Android 4.1 – 4.3.1) 56
3.6.3.11 KitKat (Android 4.4 - 4.4.4) 56
3.6.3.12 Lollipop (Android 5.0 – 5.1.1) 56
3.6.3.13 Marshmallow (Android 6.0 – 6.0.1) 57
3.6.3.14 Nougat (Android 7.0 – 7.1.2) 57
3.6.3.15 Oreo (Android 8.0 – 8.1) 57
3.6.3.16 Pie (Android 9.0) 57
3.6.3.17 Q (Android 10.0) 57
3.6.3.18 R (Android 11.0) 57
3.6.3.19 Android 12 58
3.6.4 Android Studio 58
3.6.5 Android APK 58
3.6.6 Android SDK 59
3.6.7 Gradle 59
3.6.8 Αρχιτεκτονική Android Εφαρμογών 59
3.6.8.1 Activity και κύκλος ζωής 59
3.6.8.2 Fragment και κύκλος ζωής 61
3.6.8.3 Android Manifest 61
3.7 GitHub 62
3.8 Java 62
3.9 Python 63
3.10 Google Collab 63
Κεφάλαιο 4ο: Ανάλυση Εφαρμογής 64
4.1 Σύνολα Δεδομένων Ανίχνευσης και Αναγνώρισης Αντικειμένων 64
4.1.1 Σύνολο Δεδομένων Ανίχνευσης Αντικειμένων (Object Detection Dataset) 64
4.1.2 Σύνολο Δεδομένων Αναγνώρισης Αντικειμένων (Object Recognition Dataset) 66
4.2 Αναγνώριση πλαισίου οριοθέτησης πινακίδας με YOLOv5 68
4.2.2 Εισαγωγή 68

4.2.3 Προετοιμασία Dataset ... 69
4.2.4 Επεξεργασία εικόνων ... 69
4.2.5 Επεξεργασία ετικετών ... 69
4.2.6 Εκπαίδευση YOLOv5s ... 70
4.2.7 Βασικές παράμετροι εκπαίδευσης ... 70
4.2.8 Επιλογή μοντέλου εκπαίδευσης (YOLOv5s) ... 71
4.2.9 Διαδικασία και αποτελέσματα εκπαίδευσης ... 71
4.2.10 Μετατροπή σε TFLite ... 73
4.3 Περιγραφή εκπαίδευσης του μοντέλου ταξινόμησης αντικειμένων ... 76
4.3.1 Εκπαίδευση μοντέλου ταξινόμησης αντικειμένων ... 76
4.3.2 Εξαγωγή βαρών μοντέλου ... 80
4.4 Εφαρμογή στο Android Studio ... 81
4.4.1 Εισαγωγή ... 81
4.4.2 Δομή ... 82
4.4.3 Android Manifest ... 82
4.4.4 Detector Activity ... 83
4.4.5 CameraActivity ... 86
4.4.6 CameraConnectionFragment & LegacyCameraConnectionFragment ... 88
4.4.7 Περιβάλλον (env) ... 89
4.4.8 CustomView ... 89
4.4.9 MediaPlayer ... 89
4.4.10 Tracking ... 89
4.4.11 Αναγνώριση (Detection) ... 90
4.4.12 TensorFlow Classifier API ... 90
4.4.13 Assets, res, ml ... 91
4.4.14 Gradle – Εξαρτήσεις ... 91
4.4.15 Αποτέλεσμα Εφαρμογής & Ανταπόκριση σε πραγματικές συνθήκες ... 92
Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης ... 94
5.1 Συμπεράσματα ... 94
5.2 Προτάσεις για βελτίωση ... 94
5.3 Σύνοψη ... 95
Βιβλιογραφία ... 96

Κατάλογος Σχημάτων

Σχήμα 2.1: Αριθμός ανθρώπινων θανάτων στους δρόμους ανά 1εκ. κατοίκους (EL). 3
Σχήμα 2.2: Διεπαφή της εφαρμογής Tensorlow Object Detection. 4
Σχήμα 2.3: Διεπαφή της εφαρμογής Car-Assistant. 5
Σχήμα 3.1: Βιολογικός Νευρώνας. 8
Σχήμα 3.2: Νευρώνας McCulloch-Pitts. 8
Σχήμα 3.3: Βηματική Συνάρτηση. 9
Σχήμα 3.4: Σιγμοειδής Συνάρτηση. 9
Σχήμα 3.5: Αρχιτεκτονική ενός συνελικτικού νευρωνικού δικτυού για αναγνώριση αντικειμένων. ... 13
Σχήμα 3.6: Διαδικασία συνέλιξης. 15
Σχήμα 3.7: Παράδειγμα σύγκρισης υποδειγματοληψίας μέσης και μέγιστης τιμής, με μέγεθος φίλτρου 2x2 σε είσοδο υποδεκτικού πεδίου 4x4. 17
Σχήμα 3.8: Αρχιτεκτονική LeNet. 19
Σχήμα 3.9: Αρχιτεκτονική AlexNet. 21
Σχήμα 3.10: Αρχιτεκτονική ZFNet. 23
Σχήμα 3.11: Inception Module του GoogLeNet. 24
Σχήμα 3.12: Αρχιτεκτονική GoogLeNet. 25
Σχήμα 3.13: Παράδειγμα αρχιτεκτονικής του VGG-16. 26
Σχήμα 3.14: ResNet block. 27
Σχήμα 3.15: Παράδειγμα αρχιτεκτονικής μοντέλου ResNet με 34 επίπεδα. 27
Σχήμα 3.16: Διάγραμμα SE-Block. 28
Σχήμα 3.17: Βελτιώσεις του SENet συγκριτικά με τις υπάρχουσες αρχιτεκτονικές. 28
Σχήμα 3.18: Χωρισμός εικόνας σε ισομεγέθη πλέγματα. 30
Σχήμα 3.19: Παράμετροι ενός πλαισίου οριοθέτησης. Το p_c είναι η πιθανότητα ενός αντικειμένου να βρίσκεται μέσα στο πλαίσιο και το c η κλάση του αντικειμένου. 31
Σχήμα 3.20: Στην παραπάνω εικόνα, υπάρχουν δύο πλαίσια οριοθέτησης, ένα πράσινο και ένα κόκκινο. Το κόκκινο πλαίσιο είναι το προβλεπόμενο πλαίσιο ενώ το πράσινο πλαίσιο είναι το πραγματικό πλαίσιο. Το YOLO διασφαλίζει ότι τα δύο πλαίσια οριοθέτησης είναι ίσα. 32
Σχήμα 3.21: Η παραπάνω εικόνα δείχνει πώς εφαρμόζονται οι τρεις τεχνικές για την παραγωγή των τελικών αποτελεσμάτων ανίχνευσης. 32
Σχήμα 3.22: Το δίκτυο ανίχνευσης έχει 24 συνελικτικά επίπεδα ακολουθούμενα από 2 πλήρως συνδεδεμένα επίπεδα. Τα εναλλασσόμενα συνελικτικά στρώματα 1×1 μειώνουν τον χώρο χαρακτηριστικών από τα προηγούμενα επίπεδα. Τα στρώματα συνέλιξης προεκπαιδεύοντε στο σύνολο δεδομένων ταξινόμησης ImageNet, στη μισή ανάλυση (224×224) και στη συνέχεια διπλασιάζεται η ανάλυση για την ανίχνευση. 33
Σχήμα 3.23: Αρχιτεκτονική της απλής μορφής του YOLOv1, με τα 24 στρώματα συνέλιξης και τα 2 πλήρως συνδεδεμένα στρώματα. 34
Σχήμα 3.24: Η συνάρτηση απώλειας του YOLOv1. 35
Σχήμα 3.25: Αρχιτεκτονική Darknet-19. 38
Σχήμα 3.26: Αρχιτεκτονική του Darknet-53 με 5 ResBlock που περιέχουν διαδοχικά στρώματα συνέλιξης 1×1 και στρώματα συνέλιξης 3×3. 39
Σχήμα 3.27: Ο ανιχνευτής πολλαπλής κλίμακας του YOLOv3. 39
Σχήμα 3.28: Σύγκριση των 3 δικτυών Backbone. 41
Σχήμα 3.29: Dense block του YOLOv4. 41
Σχήμα 3.30: SPP block του YOLOv4. 42

Σχήμα 3.31: Ταχύτητα και ακρίβεια του YOLOv4 με εισόδους διαφορετικών μεγεθών στο σύνολο δεδομένων MS COCO. Οι επισημασμένες γραμμές αποτελούν ανιχνευτές πραγματικού χρόνου (καρέ ανά δευτερόλεπτο ≥ 30). 43
Σχήμα 3.32: Οι 5 τύποι προ-εκπαιδευμένων μοντέλων του YOLOv5. 44
Σχήμα 3.33: Μετρικές αξιολόγησης των προ-εκπαιδευμένων μοντέλων του YOLOv5 στο σύνολο δεδομένων COCO. Οι μετρήσεις έγιναν στην GPU V100. 45
Σχήμα 3.34: Δομή παρασκήνιου του Keras. 47
Σχήμα 3.35: Στατιστικά στοιχεία του Keras σε σύγκριση με το PyTorch από την επίσημη σελίδα του. 48
Σχήμα 3.36: Διάγραμμα λειτουργίας ενός μοντέλου TensorFlow. 51
Σχήμα 3.37: Tensorfow Lite Flow. 52
Σχήμα 3.38: Κύκλος ζωής ενός Activity. 60
Σχήμα 3.39: Κύκλος ζωής ενός Fragment. 61
Σχήμα 3.40: Παράδειγμα δήλωσης Activity στο AndroidManifest.xml. 62
Σχήμα 4.1: Παραδείγματα επισημασμένων εικόνων από το σύνολο δεδομένων εκπαίδευσης του MTSD, που καλύπτουν διαφορετικές συνθήκες φωτισμού και καιρού. Στην κορυφή βλέπουμε μια επισκόπηση όλων των σημάτων – κλάσεων που καλύπτονται στο MTSD. 65
Σχήμα 4.2: Παράδειγμα εικόνας αριστερά με το πλαίσιο οριοθέτησης (μωβ τετράγωνο) και την ονομασία της κλάσης. Από δεξιά δίνεται ο αντίστοιχος κώδικας για την επισήμανση του πλαισίου οριοθέτησης σε αρχείο μορφής .json. 66
Σχήμα 4.3: Ραβδόγραμμα με τις ονομασίες των κλάσεων και τον αριθμό των εικόνων από τον οποίο αποτελείται η κάθε μια. Ο οριζόντιος άξονας δείχνει τον αριθμό των κλάσεων με την ονομασία της κάθε μιας και ο κάθετος άξονας τον αριθμό των εικόνων κάθε κλάσης. 68
Σχήμα 4.4: Παράδειγμα αρχείου ετικετών. 69
Σχήμα 4.5: Ποσοστό επιτυχών προβλέψεων. 72
Σχήμα 4.6: Ποσοστό επιτυχών ανακλήσεων. 72
Σχήμα 4.7: Μέσος όρος μέσης ακρίβειας > 0.5. 72
Σχήμα 4.8: Μέσος όρος μέσης ακρίβειας > 0.95. 73
Σχήμα 4.9: Αρχιτεκτονική μοντέλου YOLOv5s. 74
Σχήμα 4.10: Αναγνώριση με best-fp16.tflite. 75
Σχήμα 4.11: Αναγνώριση με best-int8.tflite. 76
Σχήμα 4.12: Κώδικας φόρτωσης των βιβλιοθηκών σε γλώσσα Python στο Google Colab. 77
Σχήμα 4.13: Κώδικας ανάθεσης προορισμών, ορισμού κοινού μεγέθους εικόνων και ορισμού τιμών κανονικοποίησης των τελευταίων. 77
Σχήμα 4.14: Κώδικας ανάκτησης των εικόνων εκπαίδευσης με τις ετικέτες τους και μετατροπή σε πίνακα NumPy. 78
Σχήμα 4.15: Κώδικας χωρισμού του συνόλου δεδομένων για εκπαίδευση και επαλήθευση. 78
Σχήμα 4.16: Αρχιτεκτονική του συνελικτικού μοντέλου εκπαίδευσης. 79
Σχήμα 4.17: Διαγράμματα μεταβολής της ακρίβειας και απώλειας ταξινόμησης κατά τη διάρκεια της εκπαίδευσης. Ο οριζόντιος άξονας δείχνει τις εποχές εκπαίδευσης και ο κάθετος άξονας τη τιμή από 0 έως 1. 80
Σχήμα 4.18: Αριστερά παρουσιάζεται ο κώδικας για τον υπολογισμό του ποσοστού ακριβείας του μοντέλου στο σύνολο των εικόνων δοκιμής. Δεξιά παρουσιάζονται μερικές εικόνες από το σύνολο δοκιμής με την προβλεπόμενη και την πραγματική κλάση τους. 80
Σχήμα 4.19: Κώδικας μετατροπής του αρχείου με τα βάρη μορφής h5 (Keras) σε tflite (Tensorflow). 81

Σχήμα 4.20: Δομή αρχείων Android Studio. 82
Σχήμα 4.21: AndroidManifest.xml. 83
Σχήμα 4.22: Detector Activity 84
Σχήμα 4.23: Κώδικας Αναγνώρισης Σημάτων. 85
Σχήμα 4.24: Η μέθοδος Classify. 86
Σχήμα 4.25: Η μέθοδος onClick() του CameraActivity. 88
Σχήμα 4.26: ML BInding Feature. 91
Σχήμα 4.27: Gradle Dependencies. 92
Σχήμα 4.28: Αποτέλεσμα τελικής διεπαφής της εφαρμογής. 93

Κατάλογος Πινάκων

Πίνακας 3.1: Περίληψη αρχιτεκτονικής LeNet. 19
Πίνακας 3.2: Περίληψη αρχιτεκτονικής AlexNet. 21
Πίνακας 3.3: Περίληψη αρχιτεκτονικής GoogLeNet. 24
Πίνακας 3.4: Πίνακας με τις μετρήσεις όλων τον προ-εκπαιδευμένων μοντέλων του YOLOv5. Όλα τα μοντέλα εκπαιδευτήκαν για 300 εποχές με τις προκαθορισμένες ρυθμίσεις και υπερπαραμέτρους. Οι τιμές mAP^{val} αντιστοιχούν για το σύνολο δεδομένων COCO val2017. Η επαύξηση χρόνου δοκιμής Test Time Augmentation (TTA) περιλαμβάνει επαυξήσεις αντανάκλασης και κλίμακας. 45
Πίνακας 3.5: Πίνακας μοντέλων κβαντισμού. 52
Πίνακας 4.1: Πίνακας με τα διαθέσιμα σύνολα δεδομένων ταξινόμησης πινακίδων κυκλοφορίας. 66
Πίνακας 4.2: Αποτελέσματα πριν και μετά τον κβαντισμό. 75

Συντομογραφίες

ADAS	Advanced Driver Assistance System
A.I.	Artificial Intelligence
API	Application Programming Interface
APK	Android Package File
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CSP	Cross-stage Partial
FCN	Fully Convolutional Network
FPN	Feature Pyramid Networks
FPS	Frames Per Second
GPU	Graphics Processing Unit
IOU	Intersection Over Union
mAP	mean Average Precision
NDK	Native Development Kit
NMS	Non-Maximum Suppression
PAN	Path Aggregation Network
R-CNN	Region-based Convolutional Neural Networks
ReLU	Leaky Rectified Linear Unit
SDK	Android Software Development Kit
SPP	Spatial Pyramid Pooling
SSD	Single Shot Detector
TPU	Tensor Processing Unit
YOLO	You Only Look Once
Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Σ.Ν.Δ	Συνελικτικά Νευρωνικά Δίκτυα

Κεφάλαιο 1ο: **Εισαγωγή**

1.1 Εισαγωγή

Τα σήματα κυκλοφορίας αποτελούν αναπόσπαστο μέρος της οδικής μας υποδομής. Είναι οδικές εγκαταστάσεις που απαγορεύουν, ενημερώνουν, προειδοποιούν και υποχρεώνουν τους οδηγούς για την σωστή και ασφαλή χρήση των οδών. Με την ανάπτυξη των ευφυών τεχνολογιών τα τελευταία χρόνια στο χώρο της αυτοκινητοβιομηχανίας, εταιρίες όπως η Mercedes-Benz, Tesla, Toyota και άλλες, έχουν επενδύσει στην ανάπτυξη του συστήματος Advanced Driver Assistance System. Το ADAS μέσα στα πολλά συστήματα υποβοήθησης του οδηγού που παρέχει, περιλαμβάνει και την αναγνώριση σημάτων κυκλοφορίας, προκειμένου να λυθούν οι ανησυχίες σχετικά με την οδική ασφάλεια, τόσο των οδηγών όσο και των πεζών[1–3].

Ένα σύστημα αναγνώρισης σημάτων κυκλοφορίας περιλαμβάνει συνήθως δύο στάδια: την ανίχνευση σημάτων κυκλοφορίας και την αναγνώριση πινακίδων κυκλοφορίας[4]. Το σύστημα μπορεί να ανιχνεύσει και να αναγνωρίσει τα σήματα μέσα από εικόνες που έχουν ληφθεί από κάμερες ή αισθητήρες απεικόνισης. Η ανίχνευση λαμβάνει τις εικόνες και εντοπίζει την περιοχή όπου βρίσκεται η πινακίδα κυκλοφορίας, ενώ η αναγνώριση ταξινομεί κάθε πινακίδα κυκλοφορίας σε αντίστοιχη κατηγορία. Οι εικόνες με τις ανιχνευμένες πινακίδες κυκλοφορίας από το στάδιο ανίχνευσης χρησιμοποιούνται ως είσοδοι στο στάδιο αναγνώρισης. Έτσι, η ακρίβεια της ανίχνευσης σημάτων κυκλοφορίας έχει δραματική επίδραση στην ακρίβεια ολόκληρου του συστήματος.

Πολλές προσεγγίσεις έχουν προταθεί για τον εντοπισμό πινακίδων κυκλοφορίας. Μια από αυτές είναι με τη χρήση μοντέλων και αλγορίθμων Βαθιάς Μάθησης[5], [6]. Στην αρχιτεκτονική Βαθιάς Μάθησης χρησιμοποιούνται ευρέως τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNN), καθώς έχει αποδειχθεί πως μπορούν να επιτύχουν καλύτερες επιδόσεις στην αναγνώριση αντικειμένων και ταξινόμηση εικόνων. Αλγόριθμοι όπως ο Region-based Convolutional Neural Networks (R-CNN), Fast R-CNN, Faster R-CNN, Single Shot Detector (SSD), You Only Look Once (YOLO) χρησιμοποιούν ΣΝΔ. Οι προαναφερθέντες αλγόριθμοι χωρίζονται σε δυο κατηγορίες: Πρότασης Περιοχής (Region Proposal) όπου μια-μια οι περιοχές ενός πλαισίου προτείνονται και ταξινομούνται, και Μιας Λήψης (Single Shot) όπου όλα τα αντικείμενα αναγνωρίζονται ταυτόχρονα σε ολόκληρη την εικόνα. Στην πρώτη κατηγορία ανήκει η οικογένεια των νευρωνικών δικτύων R-CNN[7] (R-CNN, Fast R-CNN, Faster R-CNN), ενώ στην δεύτερη περιλαμβάνονται τα SSD[8] και YOLO[9]. Τα νευρωνικά δίκτυα που χρησιμοποιούν αναγνώριση Πρότασης Περιοχής έχουν αρκετά αργό χρόνο αναγνώρισης για ποιοτική ανίχνευση αντικειμένων. Ωστόσο, για πλατφόρμες κινητών τηλεφώνων, τα CNN Single Shot είναι πιο κατάλληλα, καθώς είναι αρκετά πιο γρήγορα.

Για να αποφασίσουμε ποιος αλγόριθμος ταιριάζει καλύτερα σε μια συγκεκριμένη εφαρμογή, είναι σημαντικές όχι μόνο οι τυπικές μετρήσεις ακρίβειας όπως η mean average precision (mAP), αλλά και άλλοι παράγοντες όπως η κατανάλωση μνήμης, οι χρόνοι λειτουργίας και τα καρέ ανά δευτερόλεπτο. Παραδείγματος χάριν, τα αυτόνομα οχήματα απαιτούν καλή ακρίβεια ανίχνευσης και απόδοση σε πραγματικό χρόνο, ενώ οι φορητές συσκευές απαιτούν ελαφρές αρχιτεκτονικές μοντέλων με χαμηλή χρήση μνήμης. Με βάση το προηγούμενο, η καταλληλότερη μέθοδος για την υλοποίηση αυτού του έργου είναι με τη χρήση του YOLO. Αυτό συμβαίνει διότι το YOLO έχει υψηλά mAP και καρέ ανά δευτερόλεπτο (FPS), πληρώντας έτσι τις ανάγκες της εφαρμογής.

1.2 Στόχος Διπλωματικής

Σε αυτήν την εργασία, ο κύριος στόχος μας είναι να παρουσιάσουμε ένα σύστημα ανίχνευσης και αναγνώρισης σημάτων κυκλοφορίας σε πραγματικό χρόνο για λειτουργικά συστήματα Android. Οι οδηγοί πρέπει να προσέχουν συνεχώς διάφορες συνθήκες κατά την ώρα της οδήγησης, όπως για παράδειγμα την ταχύτητα του οχήματος, τον έλεγχο της πορείας του οχήματος, τον έλεγχο διερχόμενων οχημάτων και πολλών άλλων. Έτσι λοιπόν, η εφαρμογή επιδιώκει να παρέχει συμπληρωματική βοήθεια αμφότερα στους νέους και στους πιο έμπειρους οδηγούς, και παράλληλα να είναι πρακτική στη χρήση ανεξαρτήτως του τύπου οχήματος.

Επιπλέον, είναι απαραίτητο να αντιμετωπίσουμε διάφορες προκλήσεις όπως η ανίχνευση μικρών αντικειμένων σε μεγάλες εικόνες, τα περίπλοκα φόντα παρά ένα μονότονο με χαρακτηριστικές περιπτώσεις τον ουρανό, κτήρια, δέντρα και άλλα, και άλλες παρεμβολές όπως διαφημιστικές πινακίδες. Προκειμένου να αντιμετωπίσουμε τα προαναφερθέντα ζητήματα, προτείνουμε τη χρήση της τελευταίας έκδοσης του YOLO (YOLOv5) για την ανίχνευση πινακίδων και ενός εξατομικευμένου Συνελικτικού Νευρωνικού Δικτύου για την ταξινόμηση τους.

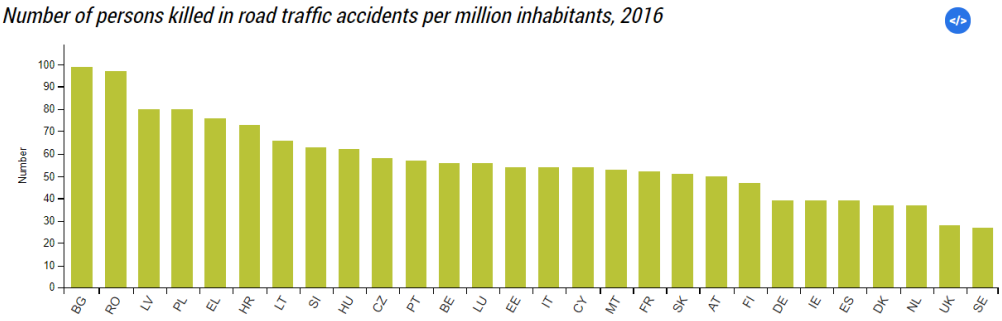
1.3 Δομή Διπλωματικής

Το 1ο κεφάλαιο αποτελεί το εισαγωγικό μέρος της εργασίας, όπου παρουσιάζεται το πρόβλημα προς λύση, καθώς επίσης περιγράφεται ο στόχος της εργασίας. Στο 2ο κεφάλαιο γίνεται μια βιβλιογραφική και θεωρητική ανασκόπηση. Ενδεικτικά, αναφέρονται αυτοκινητοβιομηχανίες που χρησιμοποιούν τη συγκεκριμένη τεχνολογία στη σειρά παραγωγής τους, ερευνήθηκαν θεωρητικές έρευνες για την αποτελεσματικότητα των αλγορίθμων και παρουσιάζονται αντίστοιχες εφαρμογές ανίχνευσης και αναγνώρισης σημάτων σε πραγματικό χρόνο. Στο 3ο κεφάλαιο αναπτύσσεται το θεωρητικό υπόβαθρο στο οποίο βασίστηκε η εργασία. Έπειτα, στο 4ο κεφάλαιο γίνεται μια περιγραφή των συνόλων δεδομένων με τις εικόνες σημάτων που χρησιμοποιήθηκαν, αναλύονται τα μοντέλα για την αναγνώριση και την ταξινόμηση των σημάτων στις κλάσεις και αναλύεται ο σχεδιασμός και η ανάπτυξη της εφαρμογής. Τέλος, στο 5ο κεφάλαιο παρουσιάζονται τα αποτελέσματα που προέκυψαν από τη δοκιμή της εφαρμογής σε πραγματικές συνθήκες οδήγησης, ενώ θα καταλήξουμε και σε διάφορα συμπεράσματα.

Κεφάλαιο 2ο: **Βιβλιογραφική ανασκόπηση και ιστορικό υπόβαθρο**

2.1 Ιστορικό υπόβαθρο

Από τότε που εφευρέθηκαν τα αυτοκίνητα, η ζωή των ανθρώπων έχει αλλάξει αρκετά. Το 1860 δημιουργήθηκε η πρώτη πατέντα από τον Jean J. Lenoir στη Γαλλία. Τα μοντέρνα αυτοκίνητα ωστόσο, όπως τα γνωρίζουμε σήμερα, ξεκίνησαν το 1890 από τη Γερμανία και τη Γαλλία.[10] Μαζί με τις πολλές βελτιώσεις που έφερε αυτό το μέσο μετακίνησης δημιουργήθηκαν και πολλοί κίνδυνοι τους οποίους μέχρι και σήμερα δεν έχουμε καταφέρει να αφανίσουμε. Ένας από αυτούς είναι φυσικά ο κίνδυνος των τροχαίων ατυχημάτων που συμβαίνουν καθημερινά στους δρόμους, μικρά ή μεγάλα, τα οποία πολλές φορές καταλήγουν να είναι ακαριαία για τα εμπλεκόμενα άτομα. Σύμφωνα με την έρευνα της Eurostat [11] μπορεί από το 2001 μέχρι σήμερα να έχουν μειωθεί τα τροχαία ατυχήματα σε μεγάλο βαθμό κατά 50%, αλλά η Ελλάδα συνεχίζει να βρίσκεται ακόμα στην κορυφή των γραφημάτων με τους περισσότερους θανάτους από τροχαία ατυχήματα ανά χώρα [11].



Σχήμα 2.1: Αριθμός ανθρώπινων θανάτων στους δρόμους ανά 1εκ. κατοίκους (EL).

Ένα από τα πιο σημαντικά θέματα της οδήγησης στους δρόμους σήμερα είναι η παρατήρηση των σημάτων οδικής κυκλοφορίας για την έγκαιρη λήψη αποφάσεων και τη διατήρηση της ασφάλειας στο δρόμο. Για να επιτευχθεί αυτό, έχουν δημιουργηθεί αρκετές εφαρμογές αναγνώρισης κυκλοφοριακών σημάτων, οι οποίες χρησιμοποιούν μια πληθώρα τεχνολογιών λογισμικού, αλλά και εξαρτημάτων. Οι εφαρμογές αυτές υπάγονται στην κατηγορία ανίχνευσης και αναγνώρισης σημάτων κυκλοφορίας (ΑΑΣΚ). Στο κεφάλαιο αυτό θα δούμε κάποιες αυτές που έχουν υλοποιηθεί και ενσωματωθεί είτε σε αυτόνομα οχήματα είτε σε μικροϋπολογιστές.[12]

2.2 Ενσωματωμένα συστήματα αυτόματης οδήγησης

2.2.1 Σύστημα αυτόνομης οδήγησης – Tesla, Inc

Η Tesla ξεκίνησε ως εταιρεία η οποία θα δημιουργούσε ηλεκτρικά σπορ οχήματα το 2003. Σήμερα έχει καταφέρει όχι μόνο να πετύχει το στόχο της, αλλά και να δημιουργήσει λύσεις για ένα καλύτερο αστικό περιβάλλον βασισμένες στην ηλεκτρική ενέργεια. [13] Έχει ενδιαφέρον το γεγονός ότι είναι μια από τις πρώτες κατασκευάστριες οχημάτων που προσφέρει αυτόματο οδηγό και μάλιστα χρησιμοποιώντας νευρωνικά δίκτυα. Στον τομέα αυτό έχει χρησιμοποιήσει την τεχνολογία βαθιάς μάθησης για προβλήματα από θέματα οπτικής μέχρι ελέγχου των συστημάτων της. Οι κάμερες γύρω από το

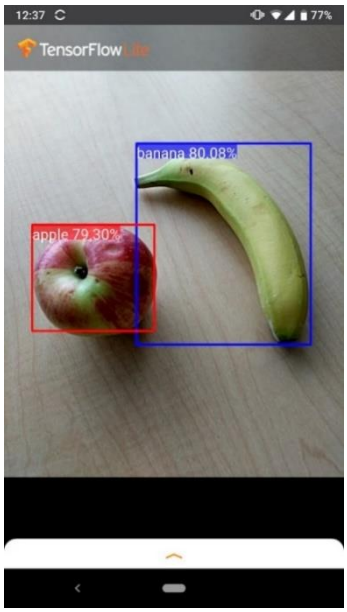
αυτοκίνητο αναλύουν ακατέργαστε εικόνες για να πραγματοποιήσουν σημασιολογική τμηματοποίηση, ανίχνευση αντικειμένων και εκτιμήσεις βάθους. Τα δίκτυα μαθαίνουν από διάφορα και πιο περίπλοκα σενάρια ανά τον κόσμο, παίρνοντας εικόνες από περίπου ένα εκατομμύριο οχήματα σε πραγματικό χρόνο. Το νευρωνικό δίκτυο της αποτελείτε από 48 δίκτυα τα οποία χρειάζονται εβδομήντα χιλιάδες ώρες για εκπαίδευση. Το αποτέλεσμα αυτών είναι ένας χίλιες διαφορετικές έξοδοι σε κάθε στιγμή.[14]

2.2.2 Σύστημα αναγνώρισης σημάτων κυκλοφορίας - Toyota

Η Toyota στα νέα της οχήματα τα τελευταία χρόνια, προσφέρει διάφορες υπηρεσίες υποβοήθησης στην οδήγηση. Το σύστημα της ονομάζεται «Toyota Safety Sense» και περιέχει διάφορα υποσυστήματα, τα οποία στοχεύουν στην αποφυγή ατυχημάτων κατά τη διάρκεια της οδήγησης. Μερικά από αυτά είναι το σύστημα υποβοήθησης διατήρησης λωρίδας, το οποίο με τη χρήση μιας κάμερας με αισθητήρα και ένα ραντάρ χιλιοστομετρικού κύματος χρησιμοποιεί την διαγράμμιση του δρόμου αλλά και τη θέση του οχήματος που βρίσκεται μπροστά του για να διευθύνει και να επαναφέρει το όχημα στην πορεία του σε περίπτωση παρέκκλισης από το δρόμο. Υπάρχει επίσης και ένα σύστημα ειδοποίησης αλλαγής λωρίδας και ελέγχου διεύθυνσης, το οποίο μόλις καταλάβει ότι το όχημα αρχίζει να παρεκκλίνει από τη λωρίδα του, ενεργοποιείται μια φωτεινή ένδειξη και ηχητική ειδοποίηση για τον οδηγό και αν χρειαστεί, αυτόματα στρέφεται ελαφρώς το όχημα προς την άλλη μεριά της λωρίδας. Ένα ακόμη σχετικό υποσύστημα είναι το σύστημα αναγνώρισης σημάτων κυκλοφορίας, όπου με τη χρήση των παραπάνω αισθητήρων, το αυτοκίνητο είναι σε θέση να εντοπίζει τα οδικά σήματα στη διαδρομή και να εμφανίζει πληροφορίες όπως το όριο ταχύτητας ή κάποιον περιορισμό στο δρόμο όπως το να κάνει ο οδηγός προσπέραση επάνω στο καντράν με μια οθόνη ευκρίνειας. Άλλα χρήσιμα συστήματα είναι το σύστημα αυτόματης λειτουργίας προβολέων, αποφυγής πρόσκρουσης και δυνατότητα διατήρησης ταχύτητας.[15]

2.3 Εφαρμογές αναγνώρισης αντικειμένων

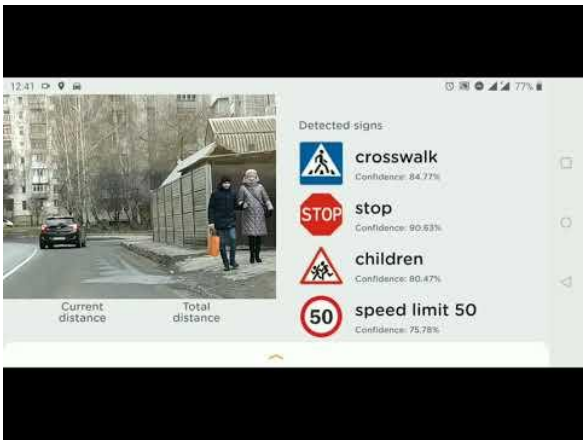
2.3.1 TensorFlow Object Detection App



Σχήμα 2.2: Διεπαφή της εφαρμογής Tensorlow Object Detection.

Το API (Application Programming Interface) του TensorFlow δίνει την δυνατότητα να ενσωματώσουμε οποιοδήποτε μοντέλο ανίχνευσης αντικειμένων μηχανικής μάθησης χρησιμοποιώντας μερικές γραμμές κώδικα. Η ίδια η ομάδα του TensorFlow έχει ετοιμάσει μία εφαρμογή επίδειξης των δυνατοτήτων της, η οποία χρησιμοποιώντας το ήδη εκπαιδευμένο μοντέλο ανίχνευσης και ταξινόμησης αντικειμένων «MobileNet SSD» μας δίνει τη δυνατότητα να αναγνωρίσουμε τα αντικείμενα που βρίσκονται γύρο μας, από ένα τενεκεδάκι αναψυκτικού μέχρι έναν άνθρωπο, χωρίς αυτό να είναι εμπόδιο στο να παρατείνουμε τις δυνατότητες του αλγορίθμου μέσω της τεχνικής transfer learning. [16]

2.3.2 Car Assistant Android



Σχήμα 2.3: Διεπαφή της εφαρμογής Car-Assistant.

Η εφαρμογή «Car Assistant Android» είναι μια ερασιτεχνική προσπάθεια του χρήστη "VladYatsenko" στο GitHub, η οποία χρησιμοποιεί την πίσω κάμερα του κινητού για να ανιχνεύει σε συνεχόμενη κίνηση σήματα στο δρόμο. Έχει χρησιμοποιηθεί ο αλγόριθμος αναγνώρισης MobileNet SSD του TensorFlow, το οποίο έχει εκπαιδευτεί με το σετ δεδομένων του COCO2017 και έχει επιλέξει μερικές κλάσεις για αναγνώριση. Επίσης η εφαρμογή ανιχνεύει την ταχύτητα του χρήστη και επιπλέον έχει επιλογή για ηχητική ειδοποίηση στην περίπτωση που αναγνωρίσει κάποιο σήμα. Είναι σχεδιασμένη σε Android Studio στη γλώσσα Java και λειτουργεί με Android software development kit (SDK) > 21.2.[17]

2.3.3 Traffic Light Detection and Color Recognition

Άλλος ένας χρήστης που επιχείρησε να δημιουργήσει μια εφαρμογή η οποία αναγνωρίζει τους φωτεινούς σηματοδότες στο δρόμο, αλλά και το χρώμα που εμφανίζονται είναι ο «nileshchopda», ο οποίος επίσης με τη χρήση του TensorFlow και της τεχνικής transfer learning.To project τρέχει σε μικροεπεξεργαστή τύπου Raspberry Pi στη μορφή του TensorFlow Lite. Η κύρια λειτουργία του είναι να γνωρίζει με τη χρήση κάποια επέκτασης κάμερας τους φωτεινούς σηματοδότες αλλά και ταυτόχρονα ποια «λαμπάκια» είναι ανάμενα και να παίρνει την απόφαση αν ο χρήστης/οδηγός πρέπει να προχωρήσει ή να σταματήσει. Ένα μειονέκτημα αυτής της εφαρμογής είναι πως για να λειτουργήσει με πραγματικό χρόνο στον πραγματικό κόσμο, πρέπει να έχει την κατάλληλη επεξεργαστική δύναμη, πράγμα που δεν υπάρχει στην πρωτότυπη έκδοση.[18]

2.4 Σύνοψη

Όπως είδαμε, η χρήση της τεχνητής νοημοσύνης δεν είναι κάτι νέο στον τομέα του προγραμματισμού και έχει διάφορες εφαρμογές οι οποίες μπορούν να λύσουν πολλά και πολύπλοκα προβλήματα που αντιμετωπίζει καθημερινά η κοινωνία. Πολλοί χρήστες ξεκίνησαν να δημιουργούν τις δικές τους,

ερασιτεχνικές και μη, εφαρμογές οι οποίες έχουν ως σκοπό την αναγνώριση των σημάτων οδικής κυκλοφορίας είτε για πεζούς είτε για οχήματα. Υπάρχουν επίσης πιο αξιόπιστα συστήματα όπως προαναφέραμε, τα οποία χρησιμοποιούνται από εξειδικευμένες εταιρίες και βιομηχανίες παραγωγής αυτοκινήτων που βοηθάν τον οδηγό να είναι πιο ασφαλής αλλά και πιο επιδέξιος στις κινήσεις του και γενικότερα στη βελτίωσης της οδηγικής συμπεριφοράς.

Κεφάλαιο 3ο: **Θεωρητικό υπόβαθρο**

3.1 Μηχανική Μάθηση (Machine Learning)

3.1.1 Εισαγωγή

Μηχανική μάθηση ονομάζουμε τρόπο με τον οποίο ένας υπολογιστής εκτελεί έναν αλγόριθμο με τη χρήση των κατάλληλων δεδομένων, ώστε να πετύχει τα βέλτιστα αποτελέσματα μιας λειτουργίας. Ξεκίνησε από τον Arthur Samuel το 1959, στα εργαστήρια της IBM (International Business Machines Corporation). Ο A.Samuel ήθελε να δημιουργήσει έναν αλγόριθμο που θα έπαιζε το παιχνίδι checkers και θα μάθαινε από κάθε ήττα του, έτσι ώστε να γίνει ισάξιος ή καλύτερος από αυτούς. Η λειτουργία που προσπάθησε να μιμηθεί είναι αντίστοιχη με αυτή της ανθρώπινης νοημοσύνης. Ξεκίνησε την υλοποίηση χρησιμοποιώντας δύο υπάρχουσες μεθόδους εκείνης της εποχής, την Rote learning και Μάθηση μέσω γενίκευσης.

Η τεχνική της απομνημόνευσης (Rote Learning) που χρησιμοποιείται στη μηχανική μάθηση αποθηκεύει ένα ιστορικό εισόδων και των αντίστοιχων εξόδων της, ανακαλώντας την αποθηκευμένη έξοδο αν αυτή υπάρχει. Χρησιμοποιεί δηλαδή μια συνάρτηση, όπου για κάθε είσοδο παράγει την ίδια έξοδο και το αντίστροφο.

Η μέθοδος μάθησης μέσω γενίκευσης (Learning by generalization) παίρνει τα δεδομένα που γνωρίζει και μέσω των της διαδικασίας εκπαίδευσης θέλουμε να γενικεύει, δηλαδή να μπορεί να υπολογίσει εξόδους για εισόδους που βλέπει πρώτη φορά. Για να δοκιμάσουμε την αποτελεσματικότητα αυτού του αλγορίθμου, χρησιμοποιούμε ένα δοκιμαστικό σετ δεδομένων και αναλύουμε τις εξόδους τους.

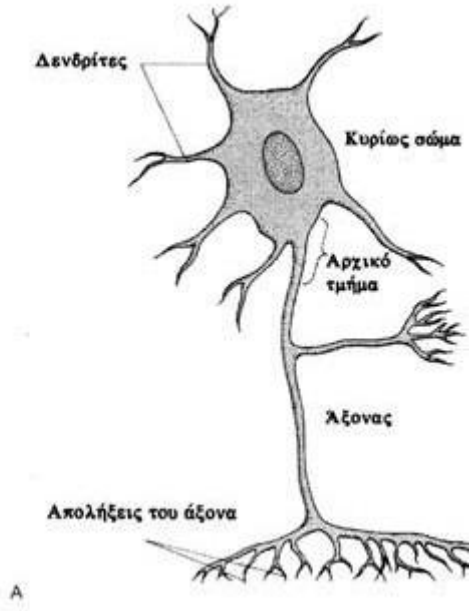
Με τις παραπάνω δύο μεθόδους, ο A.Samuel κατάφερε να πάρει το βραβείο Πρωτοπόρου Υπολογιστών το 1987 από την IEEE και να ξεκινήσει αυτό που ονομάζουμε σήμερα τεχνητή νοημοσύνη και μηχανική μάθηση. [19]–[21]

Ένας κύκλος ζωής της μηχανικής μάθησης μπορεί να χωριστεί σε δύο κύρια, διακριτά μέρη. Το πρώτο είναι η φάση εκπαίδευσης, κατά την οποία δημιουργείται ή ¨εκπαιδεύεται¨ ένα μοντέλο με την εκτέλεση ενός συγκεκριμένου υποσυνόλου δεδομένων στο μοντέλο. Η εξαγωγή συμπερασμάτων είναι η δεύτερη φάση, κατά την οποία το μοντέλο τίθεται σε εφαρμογή σε ζωντανά δεδομένα για να παράγει αποτελέσματα που μπορούν να αξιοποιηθούν. Η επεξεργασία των δεδομένων από το μοντέλο μηχανικής μάθησης αναφέρεται συχνά ως ¨βαθμολόγηση¨, οπότε μπορεί κανείς να πει ότι το μοντέλο βαθμολογεί τα δεδομένα και η έξοδος είναι μια βαθμολογία.

3.1.2 Νευρωνικά δίκτυα (Neural Networks)

Τα νευρωνικά δίκτυα είναι ένα από τα πιο δημοφιλή μοντέλα της μηχανικής μάθησης, το οποίο ειδικεύεται στο να λύνει πολύπλοκα προβλήματα τεχνητής νοημοσύνης. Ονομάζεται έτσι, διότι ο τρόπος λειτουργίας του μοιάζει πολύ με τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου και κατά συνέπια της ανθρώπινης νοημοσύνης. Η επιστήμη που μελετάει το ανθρώπινο νευρολογικό σύστημα, ονομάζεται νευροφυσιολογία και μας έχει βοηθήσει στο να κατανοήσουμε τον εγκέφαλο και τον τρόπο λειτουργίας του σε μεγαλύτερο βάθος.

Ο ανθρώπινος νευρώνας αποτελείται από τους δενδρίτες, δηλαδή τις εισόδους ηλεκτρικών σημάτων οι οποίοι έρχονται από άλλους νευρώνες, τον άξονα, ο οποίος έχει μήκος μερικά χιλιοστά και ο ρόλος του είναι να στέλνει ηλεκτρικούς παλμούς με ίδιο πλάτους και συχνότητας η οποία μεταβάλλεται προς τα έξω, δηλαδή λειτουργεί σαν έξοδος. Και οι συνάψεις, οι οποίες βρίσκονται στην έξοδο του άξονα και είναι σημεία όπου ενώνονται οι διακλαδώσεις μεταξύ τους δικού τους άξονα και των δενδριτών από άλλους νευρώνες.

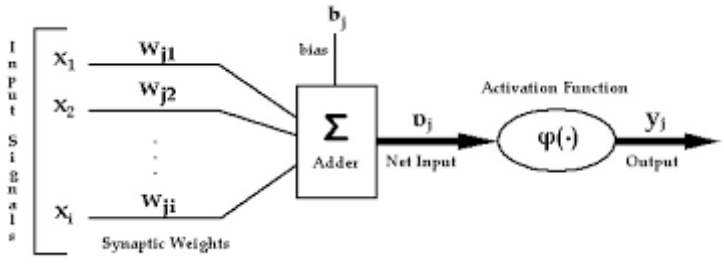


Σχήμα 3.1: Βιολογικός Νευρώνας.

Παρακάτω θα δούμε πως μεταφέροντας το σχήμα του βιολογικού νευρώνα σε ψηφιακή μορφή, μπορέσαμε να μιμηθούμε τις διαδικασίες που κάνει ο ανθρώπινος εγκέφαλος και να δημιουργήσουμε την τεχνητή νοημοσύνη.

3.1.2.1 Νευρώνας McCulloch-Pitts

Ο τεχνητός νευρώνας αποτελεί τη δομική μονάδα του τεχνητού νευρωνικού δικτύου. Σε αυτόν γίνεται όλη η επεξεργασίας και πληροφορίας, δίνοντας για κάθε συνδυασμό εισόδων μία έξοδο, η οποία έπειτα μπορεί να αποτελέσει είσοδο για κάποιο άλλο νευρώνα. Στην αρχή της εξέλιξης της επιστήμης των νευρωνικών δικτύων, οι Warren McCulloch και Walter Pitts το 1943, δημιουργήσαν ένα μοντέλο το οποίο χρησιμοποιεί κάποιες εισόδους x_n οι οποίες πολλαπλασιάζονταν με κάποια βάρη w_n και πολώνονταν με μια μεταβλητή b και όλα μαζί με μια συνάρτηση (Βηματική συνάρτηση) έδιναν ως έξοδο τη συνολική διέγερση u του νευρώνα.



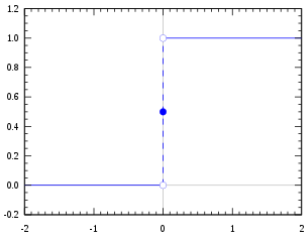
Σχήμα 3.2: Νευρώνας McCulloch-Pitts.

Ο τρόπος λειτουργίας του παραπάνω νευρώνα θεωρείται γραμμικός, μιας και οι πληροφορίες κινούνται προς μια κατεύθυνση και δεν υπάρχει κάποια επιστροφή των εισόδων στις εξόδους. Αναλυτικότερα, οι είσοδοι πολλαπλασιάζονται με τα βάρη των συνάψεων και μαζί με την πόλωση περνάνε στη συνάρτηση ενεργοποίησης του νευρώνα όπου και γίνεται η κύρια επεξεργασία. Η έξοδος του μοντέλου καθορίζει το αν είναι ενεργό ή όχι. Δηλαδή, αν η έξοδος είναι μεγαλύτερη από την τιμή κατωφλιού που έχουμε ορίσει, τότε ο νευρώνας ενεργοποιείται, ενώ αντίθετα, αν είναι αρνητική, τότε ο νευρώνας παραμένει αδρανής ή απενεργοποιημένος.

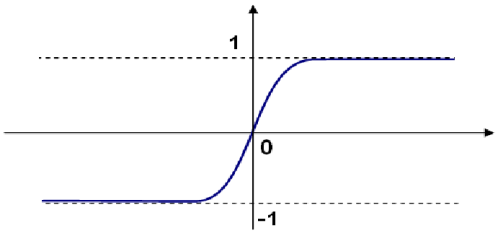
$$u = f(u) = \begin{cases} 1 \text{ αν } u > 0 \\ 0 \text{ αν } u < 0 \end{cases}$$

$$u = w_1 x_1 + \ldots + w_n x_n + b$$

Αν το αποτέλεσμα της συνολικής διέγερσης είναι θετικό τότε ο νευρώνας είναι ενεργός, ενώ αν το αποτέλεσμα είναι αρνητικό, τότε ο νευρώνας είναι ανενεργός ή βρίσκεται σε αδράνεια. Η συνάρτηση βήματος χρησιμοποιήθηκε κυρίως στα πρώτα μοντέλα λόγω της απλότητας της. Ωστόσο σήμερα, τα περισσότερα μοντέλα χρησιμοποιούν τη σιγμοειδής συνάρτηση, η οποία είναι συνεχής και φραγμένη με θετική παράγωγο. Η σιγμοειδής συνάρτηση θεωρητικά έχει ως πεδίο ορισμού όλο το σύνολο των πραγματικών αριθμών, αλλά συνήθως περιορίζεται λόγω των ορίων που θέτουμε στα συναπτικά βάρη τους στο διάστημα [0,1] ή [-1,1].



Σχήμα 3.3: Βηματική Συνάρτηση.



Σχήμα 3.4: Σιγμοειδής Συνάρτηση.

3.1.2.2 Νευρώνας Perceptron

Το 1962, ο Frank Rosenblatt ακολουθώντας τα ευρήματα του νευροψυχολόγου Donald Hebb στο βιβλίο του The Organization of Behavior, ο οποίος προτείνει την ιδέα πως όταν δύο νευρώνες πυροδοτούνται μαζί, η σύνδεση μεταξύ τους ενισχύεται και αυτό αποτελεί τη βασική λειτουργία που είναι απαραίτητη για τη μάθηση και την μνήμη του ανθρώπου, καταφέρνει να δημιουργήσει τον πρώτο νευρώνα Perceptron. Στόχος τους ήταν να μπορούν να κάνουν ταξινόμηση των δεδομένων σε διάφορες κλάσεις.

Το δίκτυο του νευρώνα Perceptron αποτελείται από δύο στρώματα ή επίπεδα. Στο πρώτο επίπεδο βρίσκονται οι είσοδοι του μοντέλου οι οποίοι πάνε απευθείας στο δεύτερο. Το δεύτερο επίπεδο απαρτίζεται από διάφορους νευρώνες McCulloch-Pitts, από τους οποίους και παράγονται οι έξοδοι του δικτύου. Έπειτα αυτές τις εξόδους τις ταξινομεί ανάμεσα σε διάφορες κλάσεις.

Τα πλεονεκτήματα αυτού του αλγορίθμου είναι πως όταν εκπαιδεύεται σωστά, τα αποτελέσματα του είναι εξίσου επιτυχημένα. Ένα όμως μειονέκτημα το οποίο απέδειξαν οι Misky και Papert είναι πως ο Perceptron δεν μπορεί να λύσει προβλήματα που δεν είναι γραμμικά διαχωρίσιμα. Αυτό το απέδειξαν χρησιμοποιώντας το πρόβλημα της λογικής πράξης XOR. Για δύο εισόδους τα αποτελέσματα θα έπρεπε να αντιστοιχούν σε τέσσερεις κλάσεις. Η λύση της εξίσωσης της συνάρτησης ενεργοποίησης δίνει μια ευθεία, η οποία αδυνατεί να χωρίσει παραπάνω από δύο κλάσεις.

Αυτά τα προβλήματα πλέον έχουν λυθεί με τη χρήση πολυεπίπεδων Perceptron αλλά και διαφορετικών συναρτήσεων ενεργοποίησης όπως η ReLU ή Softmax τις οποίες θα δούμε παρακάτω και χρησιμοποιούμε στην εργασία.

3.1.3 Βαθιά Μάθηση (Deep Learning)

Με τη χρήση την τεχνητής νοημοσύνης, πολλά απλά προβλήματα ταξινόμησης μπορούν να λυθούν με δίκτυα Perceptron ή πολυεπίπεδα Perceptron και τις διάφορες τεχνικές μάθησης που είδαμε παραπάνω και μάλιστα με πολύ μεγάλη πιθανότητα επιτυχίας. Ωστόσο για την λύση των πιο περίπλοκων προβλημάτων, χρειάζονται περισσότεροι νευρώνες πράγμα το οποίο μας οδηγεί σε αυτό που ονομάζουμε βαθιά μάθηση (Deep Learning).

Η βαθιά μάθηση αποτελεί ένα ακόμη είδος νευρωνικού δικτύου, όμως σε σχέση με τα πιο απλά νευρωνικά δίκτυα που είδαμε, η διαφορά της είναι πως ανάμεσα στο στρώμα εισόδου και εξόδου υπάρχουν πολλά περισσότερα κρυφά στρώματα νευρώνων, τα οποία κάνουν περισσότερους και πιο σύνθετους υπολογισμούς, με αποτέλεσμα να δίνουν τη δυνατότητα να λύσουμε ακόμη πιο σύνθετα προβλήματα.

Τα πλεονεκτήματα του μοντέλου είναι η αποδοτικότερη αναπαράσταση της πληροφορίας, αφού χρησιμοποιούμε περισσότερους νευρώνες, φτάνουμε όλο και πιο κοντά στον τρόπο που λειτουργεί ο ανθρώπινος εγκέφαλος και στην αρχιτεκτονική του, η οποία έχει εξίσου μεγάλο βάθος. Οι είσοδοι περνάνε ιεραρχικά από κάθε επίπεδο, σπάζοντας την πολυπλοκότητα της αναγνώρισης ενός χαρακτηριστικού σε μικρότερα κομμάτια, έτσι όπως λειτουργεί και η ανθρώπινη λογική επίλυσης προβλημάτων. Το μειονέκτημα αυτής της μεθόδου είναι πως ο χρόνος εκπαίδευσης αυξάνεται μαζί με τον χρόνο που απαιτείται για να βγάλει το μοντέλο συμπεράσματα, όταν χρησιμοποιηθεί μετά την εκπαίδευση. Παρακάτω θα δούμε πως ένας συνδυασμός διάφορων μοντέλων μπορεί να βοηθήσει σε κάθε περίπτωση.

3.1.3.1 Μάθηση με επίβλεψη

Σε αυτή την παράγραφο θα αναλύσουμε τον τρόπο με τον οποίο λειτουργεί ένας αλγόριθμος μάθησης με επίβλεψη. Η ανάγκη που μπορεί να προκύψει για να χρησιμοποιήσει κάποιος τον συγκεκριμένο αλγόριθμο είναι η ταξινόμηση των δεδομένων σε γνωστές τάξεις αλλά με άγνωστα μοτίβα, τα οποία στόχος του αλγορίθμου είναι να αναγνωρίσει. Πριν ξεκινήσει η εκπαίδευση, πρέπει να υπάρχει ένα σύνολο δεδομένων, τα οποία έχουν προ κατηγοριοποιηθεί στις ομάδες που θέλουμε ο αλγόριθμος να ταξινομεί και έπειτα όλα αυτά τα στοιχεία να περαστούν σαν είσοδο στον αλγόριθμο. Κατά διάρκεια της εκπαίδευσης, ο αλγόριθμος, ξέροντας σε ποια ομάδα ανήκει η είσοδος που επεξεργάζεται, βρίσκει διάφορα κοινά χαρακτηριστικά μεταξύ των υπόλοιπων στοιχείων της ομάδας που έχουν αναλυθεί και ρυθμίζει τα βάρη κατάλληλα.

Όσο μεγαλύτερο το σύνολο δεδομένων για κάθε ομάδα, τόσα περισσότερα χαρακτηριστικά θα μπορέσει ο αλγόριθμος να αναγνωρίσει με πολύ μεγάλη λεπτομέρεια, το οποίο θα βοηθήσει στο μεγαλύτερο ποσοστό επιτυχίας της αναγνώρισης. Μόλις ο αλγόριθμος ολοκληρώσει την εκπαίδευση, τότε

χρησιμοποιεί τα υπάρχοντα δεδομένα που συλλέχθηκαν, για να αναλύσει την είσοδο που του δίνεται και να την ταξινομήσει στις κλάσεις που της ταιριάζουν.

3.1.3.2 Μάθηση χωρίς επίβλεψη

Σε αυτή τη μορφή μάθησης, δεν χρειάζεται να έχουμε χωρίσει από πριν το σύνολο δεδομένων, αρκεί όμως να έχουμε την κατάλληλη ποικιλία εισόδων για τον σκοπό που θέλουμε να πετύχουμε. Ο αλγόριθμος διαβάζει τα ωμά δεδομένα χωρίς να ξέρει τι αυτά αντιπροσωπεύουν και σκοπός του είναι να βρει όσα περισσότερα επαναλαμβανόμενα μοτίβα υπάρχουν μεταξύ των δεδομένων που προσπελαύνει κατά τη διάρκεια της εκπαίδευσης. Όταν ανακαλύψει πως το αποτέλεσμα μιας από τις εισόδους ταιριάζει με κάποιο που έχει προηγηθεί το προσθέτει σε μία ομάδα ή δημιουργεί μία καινούρια. Υπάρχουν διάφορα ήδη μάθησης χωρίς επίβλεψη, όπως η μέθοδος Clustering και Association Rule Learning. Αντίστοιχα υπάρχουν και διάφοροι αλγόριθμοι οι οποίοι κάνουν αυτή την ταξινόμηση όπως ο K-means clustering, ο αλγόριθμος ανάπτυξης συχνών μοτίβων (Frequent pattern growth algorithm) και ο αλγόριθμος ανάλυσης κύριας συνιστώσας (Principal component analysis).[22][23]

Οι λόγοι για τους οποίους κάποιος θα διάλεγε μια τέτοια μέθοδο εκμάθησης είναι γιατί μπορεί να επιτύχει κατηγοριοποίηση και ταξινόμηση με βάση μεταβλητών με μεγάλη πολυπλοκότητα και δυσκολία ανίχνευσης και από τον ίδιο τον άνθρωπο. Το μοντέλο αφού εκπαιδευτεί, σε διάφορους τομείς μπορεί να προβλέπει το μέλλον, χρησιμοποιώντας τα στοιχεία που έχει επεξεργαστεί. Μερικά παραδείγματα χρήσης μοντέλων που έχουν εκπαιδευτεί με μάθηση χωρίς επίβλεψη είναι η αναγνώριση ανωμαλιών, αναγνώριση εμποδίων, σύστημα προτάσεων και δημιουργία πιθανών χαρακτήρων πελατών για μια εταιρεία.

3.1.3.3 Μάθηση με ενίσχυση

Ο τρίτος τρόπος εκπαίδευσης αλγορίθμου μηχανικής μάθησης είναι με ενίσχυση. Η μάθηση με ενίσχυση μοιάζει αρκετά με την προηγούμενη κατηγορία, όπου κι εδώ έχουμε μία βάση δεδομένων που τα δεδομένα δεν είναι κατηγοριοποιημένα. Η διαφορά που έχει από τις υπόλοιπες, είναι πως κατά την έξοδο, τα αποτελέσματα βαθμολογούνται με θετικούς ή αρνητικούς αριθμούς. Ανάλογα με το ποσοστό επιτυχίας που έχει κάθε φορά ο αλγόριθμος, επηρεάζονται οι παράμετροι της επόμενης εκπαίδευσης. Η λογική πίσω από αυτό τον τρόπο είναι πως ο αλγόριθμος μαθαίνει δοκιμάζοντας και κάνοντας λάθη, όπως ακριβώς και ο άνθρωπος, μέχρι να φτάσει σε ένα αποδεκτό ποσοστό επιτυχίας. Ο λόγος που κάποιος θα χρησιμοποιούσε αυτό τον αλγόριθμο είναι για τα προσομοιώσει τη λύση ενός παιχνιδιού ή τη λύση σε ένα στρατηγικό πρόβλημα, να μειώσει το ρίσκο και να δει τα πιθανά σενάρια αποτυχίας. Παρακάτω θα δούμε, πως ένας αλγόριθμος σαν και αυτόν, κατέληξε να νικήσει τους καλύτερους παίκτες σε διάφορα παιχνίδια στρατηγικής όπως το σκάκι, checker και go.

3.1.3.4 Ρυθμός Εκπαίδευσης

Ο ρυθμός εκπαίδευσης, είναι μία σταθερή τιμή, η οποία προσδιορίζει το ρυθμό με τον οποίο θα αλλάζουν οι μεταβλητές των βαρών εισόδου και του πολωτή. Όσο πιο μεγάλη είναι η τιμή, τόσο πιο μεγάλη διαφορά μπορεί να έχει το ποσοστό πετυχημένων προβλέψεων κατά τη διάρκεια μιας εποχής, ενώ όσο πιο μικρός είναι, τόσο πιο αργά θα αλλάζει. Ο σκοπός αυτής της τιμής είναι να βοηθάει τον αλγόριθμο να αλλάζει τα βάρη των εισόδων με τέτοιο ρυθμό, ώστε να βοηθάει στην μεγαλύτερη επιτυχία της εκπαίδευσης.

3.1.3.5 Αρχικοποίηση Βαρών

Τα βάρη σε ένα νευρωνικό δίκτυο χρησιμοποιούνται στην εκπαίδευση του, δίνοντας βαρύτητα στις εισόδους κάθε νευρώνα σε κάθε επίπεδο. Οι αρχικές τιμές που θα τους δοθούν κατά τη διάρκεια της αρχικοποίησης τους, δημιουργούνται τυχαία από τον αλγόριθμο μάθησης και συνήθως οι τιμές είναι μικρές. Μετά από κάθε επανάληψη, οι τιμές των βαρών αυξομειώνονται σύμφωνα με τον ρυθμό εκπαίδευσης, αλλά και το αποτέλεσμα της επαλήθευσης του αλγορίθμου.

3.1.3.6 Overfitting

Όταν κατά τη διάρκεια της εκπαίδευσης ενός μοντέλου υπερτροφοδοτούμε με δεδομένα το μοντέλο και δεν έχουμε αρκετές ή διαφορετικές εισόδους για την επαλήθευση, τότε είναι πολύ πιθανό οι πιθανότητες σωστής πρόβλεψης να είναι πολύ υψηλές. Όσο και αν αυτός είναι ο σκοπός της εκπαίδευσης του μοντέλου, στην πραγματικότητα, οι τόσο υψηλές τιμές μας προβληματίζουν διότι σημαίνει πως το μοντέλο έχει μάθει καλά τα δεδομένα που του δίνονται, αλλά αποτυγχάνει στη γενίκευση. Αυτή η κατάσταση ονομάζεται overfitting και μπορεί να αποφευχθεί, αλλάζοντας την αναλογία των δεδομένων προς εκπαίδευση και έλεγχο, αφαιρώντας κάποια στρώματα νευρώνων ή με την μέθοδο Dropout.

3.1.3.7 Dropout

Ο όρος Dropout, που σημαίνει αφαίρεση, αναφέρεται στην προσωρινή αφαίρεση μερικών κρυφών ή φανερών κόμβων κατά τη διάρκεια της εκπαίδευσης ενός νευρωνικού δικτύου. Αφαιρώντας έναν κόμβο, αφαιρούμε και όλες τις εισόδους και εξόδους που σχετίζονται μαζί του, με σκοπό να κάνουμε την εκπαίδευση του δικτύου δυσκολότερη, έτσι ώστε να αποφύγουμε την κατάσταση υπερπροσαρμογής (overfitting). Η επιλογή αυτή γίνεται τυχαία σε κάθε επανάληψη, χρησιμοποιώντας μία σταθερά πιθανότητας p. Κατά τη διάρκεια του ελέγχου ή επαλήθευσης της εκπαίδευσης του δικτύου, όλοι οι κόμβοι χρησιμοποιούνται, χωρίς να υπάρχουν αφαιρετέοι.

3.1.3.8 Εξαγωγή Συμπερασμάτων

Η εξαγωγή συμπερασμάτων στη μηχανική μάθηση είναι η διαδικασία της εκτέλεσης ζωντανών σημείων δεδομένων σε έναν αλγόριθμο μηχανικής μάθησης για τον υπολογισμό μιας εξόδου, όπως μια απλή αριθμητική βαθμολογία. Όταν ένα μοντέλο μηχανικής μάθησης εκτελείται σε εφαρμογές, συχνά περιγράφεται στη συνέχεια ως τεχνητή νοημοσύνη (AI), δεδομένου ότι εκτελεί λειτουργίες παρόμοιες με την ανθρώπινη σκέψη και ανάλυση. Η εξαγωγή συμπερασμάτων μηχανικής μάθησης συνεπάγεται ουσιαστικά με την ανάπτυξη μιας εφαρμογής λογισμικού, καθώς το μοντέλο μηχανικής μάθησης είναι συνήθως απλώς κώδικας λογισμικού που υλοποιεί έναν μαθηματικό αλγόριθμο. Αυτός ο αλγόριθμος κάνει υπολογισμούς με βάση τα χαρακτηριστικά των δεδομένων, γνωστά ως "χαρακτηριστικά" στην καθομιλουμένη.

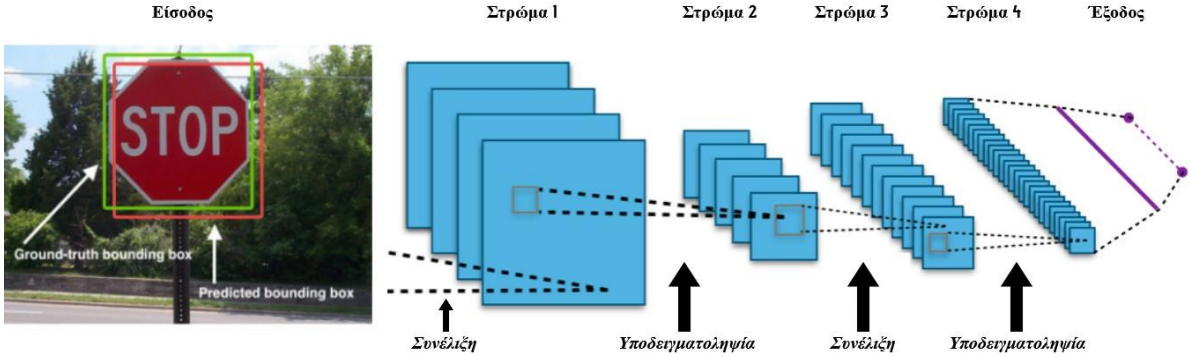
Η εξαγωγή συμπερασμάτων μηχανικής μάθησης αναπτύσσεται γενικά από μηχανικούς DevOps ή μηχανικούς δεδομένων. Μερικές φορές ζητείται από τους επιστήμονες δεδομένων, οι οποίοι είναι υπεύθυνοι για την εκπαίδευση των μοντέλων, να αναλάβουν τη διαδικασία εξαγωγής συμπερασμάτων μηχανικής μάθησης. Αυτή η τελευταία κατάσταση συχνά προκαλεί σημαντικά εμπόδια στο να φτάσει κανείς στο στάδιο της εξαγωγής συμπερασμάτων, δεδομένου ότι οι επιστήμονες δεδομένων δεν είναι απαραίτητα ειδικευμένοι στην ανάπτυξη συστημάτων. Οι επιτυχείς αναπτύξεις μοντέλων μηχανικής μάθησης είναι συχνά το αποτέλεσμα του στενού συντονισμού μεταξύ διαφορετικών ομάδων, ενώ συχνά αναπτύσσονται επίσης νεότερες τεχνολογίες λογισμικού για να προσπαθήσουν να απλοποιήσουν τη διαδικασία.

3.1.4 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)

Ως συνελικτικά νευρωνικά δίκτυα ορίζονται τα δίκτυα πολλών στρωμάτων κατάλληλα για λειτουργίες αναγνώρισης. Οι Kunihiko Fukushima και Yann LeCun έθεσαν τα θεμέλια της έρευνας γύρω από τα συνελικτικά νευρωνικά δίκτυα στην εργασία τους το 1980 [24]. Αξιοσημείωτη είναι επίσης, η εφαρμογή οπισθοδιάδοσης (backpropagation) του LeCun για την εκπαίδευση νευρωνικών δικτύων ώστε να αναγνωρίζουν μοτίβα μέσα σε μια σειρά χειρόγραφων ταχυδρομικών κωδικών [25]. Συμπληρωματικά, έμπνευση για αυτά τα μοντέλα αποτελέσαν και τα ευρήματα της έρευνας των David Hubel και Torsten Wiesel σχετικά με τον οπτικό φλοιό της γάτας, ο οποίος περιέχει δυο ειδών νευρώνες: α) τους απλούς νευρώνες οι οποίοι εξάγουν τοπικά χαρακτηριστικά από μια περιοχή του αμφιβληστροειδούς και β) τους σύνθετους νευρώνες οι οποίοι συνδυάζουν τις εξόδους πολλών απλών νευρώνων από μια μικρή γειτονιά και παράγουν χαρακτηριστικά υψηλότερου επιπέδου [26].

Ένα ΣΝΔ αποτελείται από εναλλασσόμενα στρώματα συνέλιξης και υποδειγματοληψίας, όπου έπειτα από το τελευταίο στρώμα ακολουθούν ένα ή περισσότερα πλήρως συνδεδεμένα στρώματα που λειτουργούν ως ένα υποδίκτυο ταξινομητή. Η αρχιτεκτονική ενός τέτοιου μοντέλου αποτελείται από τα εξής μέρη:

- Είσοδος – μια εικόνα ως δισδιάστατος πίνακας.
- Στρώμα 1 – στρώμα συνέλιξης.
- Στρώμα 2 – στρώμα υποδειγματοληψίας.
- Στρώμα 3 – στρώμα συνέλιξης.
- Στρώμα 4 – στρώμα υποδειγματοληψίας.
- Πλήρως συνδεδεμένο στρώμα ταξινόμησης με το τελευταίο στρώμα υποδειγματοληψίας.



Σχήμα 3.5: Αρχιτεκτονική ενός συνελικτικού νευρωνικού δικτυού για αναγνώριση αντικειμένων.

Η αρχιτεκτονική αυτή σχεδιάζεται με σκοπό να εκμεταλλευθεί τη δισδιάστατη δομή των εικόνων εισόδου. Το συγκεκριμένο επιτυγχάνεται με τοπικές συνδέσεις και συναπτικά βάρη ακολουθούμενα από υποδειγματοληψία, προκειμένου να δημιουργηθούν χαρακτηριστικά ανεξάρτητα μετατοπίσεων.

Ο συνδυασμός στρωμάτων συνέλιξης και υποδειγματοληψίας υλοποιεί έναν μετασχηματισμό της εικόνας εισόδου σε χάρτη χαρακτηριστικών. Η έξοδος ενός συνελικτικού στρώματος ή ενός στρώματος υποδειγματοληψίας είναι ουσιαστικά μια σειρά από χάρτες χαρακτηριστικών. Ειδικότερα, κάθε χάρτης χαρακτηριστικών είναι ένα δισδιάστατο πλέγμα από νευρώνες όπου ο κάθε νευρώνας διεγείρεται από μια μικρή περιοχή στο προηγούμενο στρώμα, η οποία καλείται τοπικό υποδεκτικό πεδίο. Μάσκα ονομάζονται τα συναπτικά βάρη που συνδέουν έναν νευρώνα με τους νευρώνες του τοπικού πεδίου [27].

Όπως είπαμε παραπάνω, τα συνελικτικά δίκτυα αποτελούνται από τρεις τύπους στρωμάτων: τα συνελικτικά στρώματα, τα στρώματα υποδειγματοληψίας και τα πλήρως συνδεδεμένα στρώματα. Στη συνέχεια θα αναλύσουμε τα προαναφερθέντα αυτά 3 στρώματα.

3.1.4.1 Συνελικτικό στρώμα

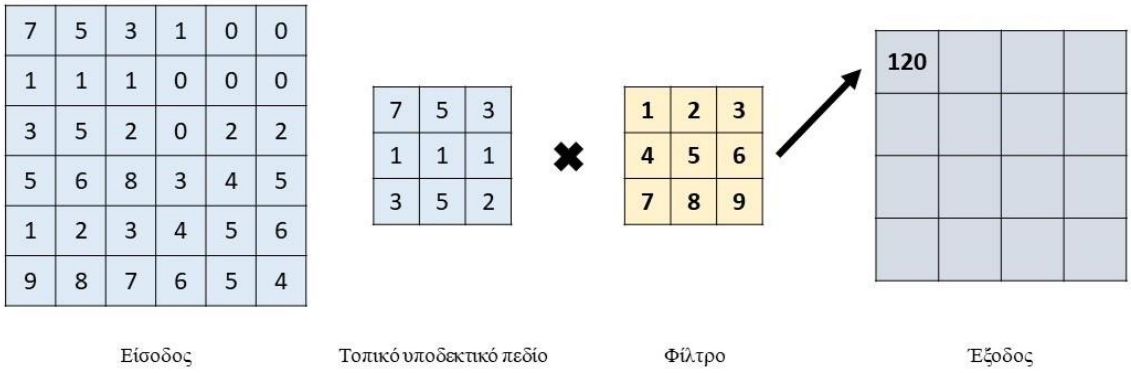
Το συνελικτικό επίπεδο αποτελεί τη βασική δομή ενός CNN, με την πλειοψηφία τον υπολογισμών να συμβαίνουν σε αυτό. Όπως αναφέρθηκε νωρίτερα, το στρώμα αυτό απαιτεί συγκεκριμένα στοιχεία, τα οποία είναι τα δεδομένα εισόδου (εικόνα, βίντεο, ήχος), τα φίλτρα ή μάσκα και τους χάρτες χαρακτηριστικών.

Η είσοδος σε αυτό το επίπεδο μπορεί να αναπαρασταθεί ως ένας πίνακας τριών διαστάσεων – πλάτος, ύψος, βάθος - W×H×C, με το πλάτος να αντιστοιχεί στα pixel του οριζόντιου άξονα, το ύψος στα pixel του κάθετου άξονα και το βάθος να αναφέρεται στα κανάλια όπως για παράδειγμα το RGB για τις έγχρωμες εικόνες. Τα φίλτρα είναι μια δισδιάστατη διάταξη βαρών η οποία αντιπροσωπεύει ένα μέρος της εικόνας. Συνήθως το μέγεθος ενός φίλτρου είναι ένας πίνακας διαστάσεων 3×3, το οποίο καθορίζει επίσης και το τοπικό υποδεκτικό πεδίο. Η είσοδος συνελίσσεται με μια σειρά από φίλτρα παράγοντας έτσι τους χάρτες χαρακτηριστικών, καθένας από τους οποίους είναι ένα δισδιάστατο πλέγμα νευρώνων. Έστω για μια εικόνα I (Image) και ένα φίλτρο K (Kernel) τότε έχουμε την παρακάτω μαθηματική σχέση για τη συνέλιξη:

$$conv(I,K)_{x,y} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} \sum_{k=1}^{n_C} K_{i,j,k} I_{x+i-1,y+j-1,k} \qquad (3.1.4.1)$$

Όπου τα n_H,n_W,n_C δηλώνουν το μέγεθος του ύψους, το μέγεθος του πλάτους και τον αριθμό των καναλιών της εικόνας, αντίστοιχα. Τα x, y καθορίζουν τη θέση του νευρώνα στον k-οστό χάρτη χαρακτηριστικών.

Όπως μπορούμε να δούμε στην παρακάτω εικόνα (Σχήμα 3.2), κάθε τιμή εξόδου στον χάρτη χαρακτηριστικών δεν χρειάζεται να συνδέεται άμεσα με κάθε τιμή pixel της εικόνας εισόδου. Απαραίτητο είναι μόνο να συνδεθεί το υποδεκτικό πεδίο όπου εφαρμόζεται το φίλτρο. Δεδομένου ότι ο πίνακας εξόδου δεν χρειάζεται να αντιστοιχηθεί απευθείας σε κάθε τιμή εισόδου, τα συνελικτικά επίπεδα αναφέρονται συνήθως και ως επίπεδα ‘μερικώς συνδεδεμένα’.



Σχήμα 3.6: Διαδικασία συνέλιξης.

Χρειάζεται επίσης, να σημειωθεί ότι τα συναπτικά βάρη (βλέπετε εξίσωση 3.1.4.1) $K_{i,j,k}$ στον χάρτη χαρακτηριστικών παραμένουν σταθερά και είναι ανεξάρτητα της θέσης x, y. Ορισμένες παράμετροι όμως, όπως είναι και οι τιμές του βάρους, μπορούν να προσαρμόζονται κατά της διάρκεια της εκπαίδευσης μέσω της διαδικασίας της οπισθοδιάδοσης (backpropagation) ή της απότομης καθόδου (gradient descent). Ωστόσο, υπάρχουν τρεις υπερπαράμετροι που επηρεάζουν το μέγεθος της εξόδου και μπορούν να μειώσουν σημαντικά την πολυπλοκότητα του μοντέλου μέσω της ρύθμισης τους πριν από την εκπαίδευση.

Η πρώτη υπερπαράμετρος είναι το βάθος της εξόδου που παράγεται από τα συνελικτικά στρώματα. Το βάθος της εξόδου επηρεάζεται από τον αριθμό των φίλτρων. Για παράδειγμα, τρία διαφορετικά φίλτρα θα έδιναν τρεις διαφορετικούς χάρτες χαρακτηριστικών, δημιουργώντας ένα βάθος τριών. Η μείωση αυτής της υπερπαραμέτρου μπορεί να ελαχιστοποιήσει σημαντικά τον συνολικό αριθμό των νευρώνων του δικτύου, αλλά μπορεί επίσης να μειώσει σημαντικά τις δυνατότητες αναγνώρισης προτύπων του μοντέλου.

Η δεύτερη υπερπαράμετρος είναι το βήμα ή διασκελισμός (stride), και είναι απόσταση ή ο αριθμός των εικονοστοιχείων που κινείται το φίλτρο πάνω από τον πίνακα εισόδου. Αυτό σημαίνει ότι το φίλτρο δεν εφαρμόζεται σε όλες τις δυνατές θέσεις x, y της εικόνας, αλλά σε θέσεις που απέχουν μεταξύ τους απόσταση ίση με όση έχει οριστεί το βήμα $s_{x,y}$. Για παράδειγμα, αν θέσουμε το βήμα ως 1, τότε θα έχουμε ένα εξαιρετικά επικαλυμμένο πεδίο που παράγει μεγάλες ενεργοποιήσεις. Παρακάτω ορίζεται η σχέση για τη συνέλιξη με βήμα:

$$strconv(I,K)_{x,y} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} K_{i,j} I_{xs_x - i, ys_y - j} \qquad (3.1.4.2)$$

Σπανίως δίνεται και τιμή βήματος μεγαλύτερη του ενός, η οποία μειώνει την ποσότητα της επικάλυψης, οπότε και επιτυγχάνεται μείωση των υπολογισμών, παρ' όλ' αυτά αρχίζει να διακρίνεται μια μικρή μείωση στην επίδοση.

Η τρίτη και τελευταία υπερπαράμετρος είναι το μηδενικό γέμισμα (Zero-padding), όπου χρησιμοποιείται όταν τα φίλτρα δεν ταιριάζουν ακριβώς στην εικόνα εισόδου. Ουσιαστικά είναι μια διαδικασία συμπλήρωσης του περιγράμματος της εισόδου. Θέτει όλα τα στοιχεία που εμπίπτουν εκτός του πίνακα εισόδου ίσα με μηδέν, παράγοντας έτσι μια μεγαλύτερη ή ίσου μεγέθους έξοδο. Υπάρχουν τρεις τύποι γεμίσματος:

i. Έγκυρο γέμισμα (Valid-padding): Επίσης γνωστό ως χωρίς γέμισμα (No-padding). Σε αυτήν την περίπτωση, η τελευταία συνέλιξη απορρίπτεται εάν οι διαστάσεις δεν ευθυγραμμίζονται.
ii. Ίδιο γέμισμα (Same-padding): Διασφαλίζει ότι το στρώμα εξόδου έχει το ίδιο μέγεθος με το στρώμα εισόδου.
iii. Πλήρες γέμισμα (Full-padding): Αυξάνει το μέγεθος της εξόδου προσθέτοντας μηδενικά στο περίγραμμα της εισόδου.

Ιδιαίτερα σημαντικό είναι ότι με τη χρήση των παραπάνω υπερπαραμέτρων αλλάζουν οι διαστάσεις της εξόδου. Η αλλαγή αυτή μπορεί να υπολογιστεί με τον ακόλουθο τύπο:

$$\frac{(V-R)+2Z}{S+1} \tag{3.1.4.3}$$

Όπου το V αντιπροσωπεύει το μέγεθος της εισόδου (πλάτος×ύψος×βάθος - W×H×C), το R αντιπροσωπεύει το μέγεθος του υποδεκτικού πεδίου, το Z αντιστοιχεί στην ποσότητα του μηδενικού σετ γεμίσματος και το S αναφέρεται στο βήμα. Εάν το αποτέλεσμα της εξίσωση δεν είναι ίσο με έναν ολόκληρο ακέραιο αριθμό, τότε το βήμα έχει οριστεί εσφαλμένα, καθώς οι νευρώνες δεν θα μπορούν να χωρέσουν σωστά στη δεδομένη είσοδο.

3.1.4.2 Στρώμα υποδειγματοληψίας

Τα επίπεδα συγκέντρωσης (pooling layers), επίσης γνωστά ως στρώματα υποδειγματοληψίας (sub-sampling), πραγματοποιούν μείωση των διαστάσεων του μοντέλου μειώνοντας τον αριθμό των παραμέτρων εισόδου, οδηγώντας έτσι σε περαιτέρω μείωση της υπολογιστικής πολυπλοκότητας του μοντέλου. Ένα στρώμα υποδειγματοληψίας εφαρμόζεται συνήθως μετά από κάθε στρώμα συνέλιξης. Παρόμοια με το συνελικτικό στρώμα, σαρώνει με ένα φίλτρο ολόκληρη την είσοδο, με μόνη διαφορά ότι αυτό το φίλτρο δεν έχει βάρη.

Στο στρώμα υποδειγματοληψίας, το φίλτρο εφαρμόζει μια συνάρτηση συνάθροισης, γνωστή και ως «σύνοψη», στις τιμές εντός του υποδεκτικού πεδίου, συμπληρώνοντας τον πίνακα εξόδου. Υπάρχουν δύο κύριοι τύποι συναρτήσεων:

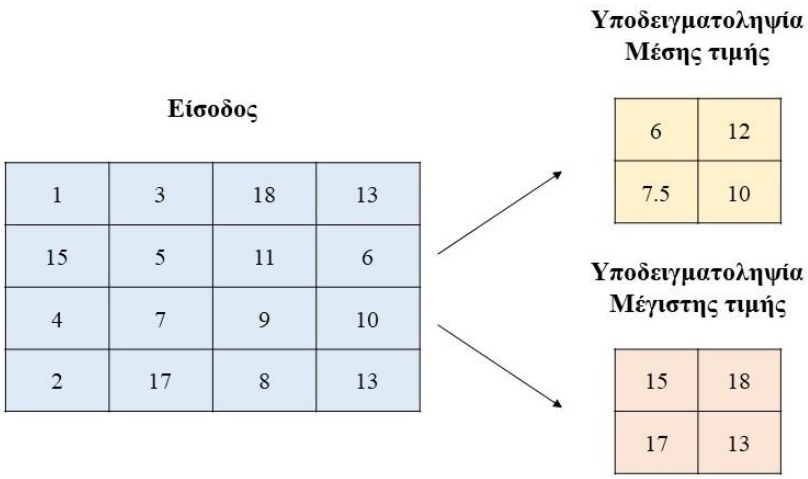
i. Μέσης τιμής: Καθώς το φίλτρο μετακινείται κατά μήκος της εισόδου, υπολογίζει τη μέση τιμή εντός του υποδεκτικού πεδίου και την επιστρέφει στον πίνακα εξόδου. Η μαθηματική περιγραφή της συνάρτησης αυτής για έναν χάρτη χαρακτηριστικών εισόδου διαστάσεων N×N με χρήση μάσκας διαστάσεων m×m είναι:

$$output_{x,y} = \frac{1}{m^2}\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} I_{xm+i,ym+j} \tag{3.1.4.4}$$

ii. Μέγιστης τιμής: Καθώς το φίλτρο μετακινείται κατά μήκος της εισόδου, επιλέγει το εικονοστοιχείο με τη μέγιστη τιμή από το υποδεκτικό πεδίο και το στέλνει στη διάταξη εξόδου. Δηλαδή, εφαρμόζεται η εξής σχέση:

$$output_{x,y} = \max_{0 \le i,\, j \le m-1} I_{xm+i,\, ym+j} \tag{3.1.4.5}$$

Η υποδειγματοληψία μέγιστης τιμής είναι μη γραμμική, προσφέροντας καλύτερη αναπαράσταση των χαρακτηριστικών σε μειωμένες διαστάσεις. Εύλογα, λοιπόν, αυτή η προσέγγιση τείνει να χρησιμοποιείται πιο συχνά σε σύγκριση με τη μέση τιμή.



Σχήμα 3.7: Παράδειγμα σύγκρισης υποδειγματοληψίας μέσης και μέγιστης τιμής, με μέγεθος φίλτρου 2x2 σε είσοδο υποδεκτικού πεδίου 4x4.

Τέλος, μετά από κάθε λειτουργία συνέλιξης, ένα CNN εφαρμόζει έναν μετασχηματισμό Rectified Linear Unit (ReLU) ή μια άλλη μη γραμμική συνάρτηση ενεργοποίησης, όπως η tanh και η sigmoid στον χάρτη χαρακτηριστικών, εισάγοντας μη γραμμικότητα στο μοντέλο. Κατά κύριο λόγο προτιμάται η ReLU, καθώς απλοποιεί τον αλγόριθμο οπισθοδιάδοσης και τον επιταχύνει, αφού υλοποιεί λιγότερες πράξεις [28]–[30].

3.1.4.3 Πλήρως συνδεδεμένο στρώμα

Το πλήρως συνδεδεμένο στρώμα περιέχει νευρώνες από τους οποίους συνδέονται άμεσα νευρώνες των δυο γειτονικών στρωμάτων. Οι τιμές των pixel της εικόνας εισόδου δεν συνδέονται απευθείας με το επίπεδο εξόδου σε μερικώς συνδεδεμένα στρώματα. Ωστόσο, στο πλήρως συνδεδεμένο στρώμα, κάθε κόμβος στο επίπεδο εξόδου συνδέεται απευθείας με έναν κόμβο στο προηγούμενο στρώμα.

Σκοπός του στρώματος είναι να χρησιμοποιήσει τα χαρακτηριστικά από την έξοδο των προηγούμενων στρωμάτων συνέλιξης και υποδειγματοληψίας, προκειμένου να κατηγοριοποιήσει την εικόνα εισόδου. Ταξινομεί δηλαδή με βάση τα χαρακτηριστικά που εξάγονται μέσω των προηγούμενων επιπέδων και των διαφορετικών φίλτρων τους. Ενώ τα στρώματα συνέλιξης και υποδειγματοληψίας τείνουν να χρησιμοποιούν συναρτήσεις ReLU, τα πλήρως συνδεδεμένα στρώματα χρησιμοποιούν συνήθως μια συνάρτηση ενεργοποίησης Softmax για την κατάλληλη ταξινόμηση των εισόδων, παράγοντας μια πιθανότητα από 0 έως 1.

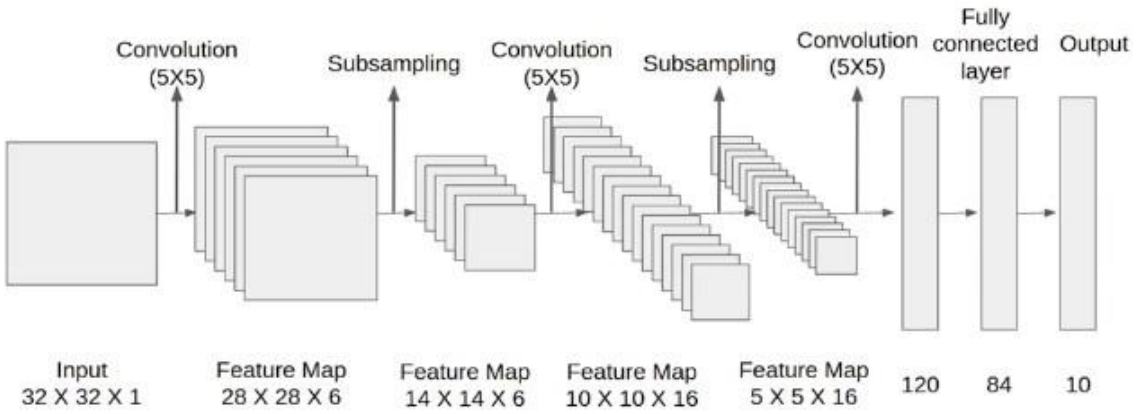
3.1.4.4 Τύποι και εφαρμογές συνελικτικών νευρωνικών δικτυών

Τα συνελικτικά νευρωνικά δίκτυα χρησιμοποιούνται ως επί το πλείστων για την επίλυση προβλημάτων τεχνητής νοημοσύνης και ειδικότερα σε συστήματα αναγνώρισης εικόνων. Στη συνέχεια, παρουσιάζονται μερικές αξιοσημείωτες αρχιτεκτονικές CNN:

- **LeNet**: Το LeNet είναι η πρώτη επιτυχημένη εφαρμογή ενός συνελικτικού δικτύου που προτάθηκε από τον Yann LeCun, όντας μέλος των εργαστήριων της Bell, το 1989 [25]. Αναφέρεται στο LeNet-5, ένα από τα πρώτα συνελικτικά νευρωνικά δίκτυα που προώθησε την ανάπτυξη της Βαθιάς Μάθησης [31]. Στην έρευνα του, συνδύασε ένα CNN που εκπαιδεύτηκε με αλγορίθμους οπισθοδιάδοσης για την αναγνώριση χειρόγραφων αριθμών και το εφάρμοσε με επιτυχία στον εντοπισμό χειρόγραφων ταχυδρομικών αριθμών. Η αρχιτεκτονική του δικτύου αποτελείται από 7 επίπεδα:
    - Είσοδος: Εισαγωγή δεδομένων εικόνας μεγέθους 32×32 pixels.
    - Στρώμα 1: Στρώμα συνέλιξης με 6 φίλτρα διαστάσεων 5×5. Γίνεται χρήση της συνάρτησης ενεργοποίησης tanh και η έξοδος είναι ένας χάρτης χαρακτηριστικών μεγέθους 28×28×6.
      Το μέγεθος της εξόδου ενός επιπέδου συνέλιξης ορίζεται από τον τύπο:

$$output = \frac{Input - filter\ size}{stride} + 1 \tag{3.1.4.6}$$

    - Στρώμα 2: Στρώμα υποδειγματοληψίας που εξάγει 6 γραφήματα χαρακτηριστικών μεγέθους 14×14. Εφαρμόζεται συνάρτηση μέσης τιμής, γίνεται χρήση φίλτρου διαστάσεων 2×2 και το βήμα ορίζεται ίσο με 2. Επομένως, η έξοδος μειώνεται σε 14×14×6.
    - Στρώμα 3: Δεύτερο στρώμα συνέλιξης με 16 φίλτρα διαστάσεων 5×5. Συνάρτηση ενεργοποίησης παραμένει η tanh. Σε αυτό το επίπεδο, μόνο οι 10 από τους 16 χάρτες συνδέονται με τους 6 χάρτες χαρακτηριστικών του προηγούμενου επιπέδου. Με τον τρόπο αυτό σπάει η συμμετρία στο δίκτυο και διατηρείται ο αριθμός των συνδέσεων σε λογικά όρια.
    - Στρώμα 4: Δεύτερο στρώμα υποδειγματοληψίας παρόμοιο με το προηγούμενο του (Στρώμα 2). Εφαρμόζεται συνάρτηση μέσης τιμής, γίνεται χρήση φίλτρου διαστάσεων 2×2 και το βήμα ορίζεται ίσο με 2. Σε αντίθεση με το προηγούμενο, έχει 16 χάρτες χαρακτηριστικών, οπότε η έξοδος μειώνεται σε 5×5×16.
    - Στρώμα 5: Τρίτο στρώμα συνέλιξης με 120 φίλτρα διαστάσεων 5×5. Κάθε χάρτης συνδέεται με τη γειτονιά 5×5 και με τα 16 γραφήματα χαρακτηριστικών του στρώματος 4. Εδώ, δεδομένου ότι το μέγεθος του γραφήματος χαρακτηριστικών του προηγούμενου στρώματος είναι επίσης διαστάσεων 5x5, το μέγεθος της εξόδου αυτό του στρώματος είναι 1×1. Άρα, τα δυο στρώματα είναι πλήρως συνδεδεμένα. Το στρώμα 5 επισημαίνεται ως συνελικτικό αντί για πλήρως συνδεδεμένο επίπεδο, επειδή εάν η είσοδος γίνει μεγαλύτερη και η δομή παραμείνει αμετάβλητη, τότε το μέγεθος της εξόδου θα είναι μεγαλύτερο από 1×1, δηλαδή όχι πλήρως συνδεδεμένο.
    - Στρώμα 6: Πλήρως συνδεδεμένο στρώμα με 84 νευρώνες. Η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η tanh.
    - Έξοδος: Πλήρως συνδεδεμένο στρώμα με 10 νευρώνες και χρήση συνάρτησης ενεργοποίησης softmax. Η softmax δίνει την πιθανότητα ότι ένα σημείο δεδομένων ανήκει σε μια συγκεκριμένη κλάση.



Σχήμα 3.8: Αρχιτεκτονική LeNet.

Πίνακας 3.1: Περίληψη αρχιτεκτονικής LeNet.

Στρώμα		**Χάρτης Χαρακτηριστικών**	**Μέγεθος**	**Φίλτρο**	**Βήμα**	**Συνάρτηση Ενεργοποίησης**
Είσοδος	Εικόνα	1	32×32	-	-	-
1	Συνέλιξη	6	28×28	5×5	1	tanh
2	Μέση Υποδειγματοληψία	6	14×14	2×2	2	tanh
3	Συνέλιξη	16	10×10	5×5	1	tanh
4	Μέση Υποδειγματοληψία	16	5×5	2×2	2	tanh
5	Συνέλιξη	120	1×1	5×5	1	tanh
6	Πλήρως συνδεδεμένο	-	84	-	-	tanh
Έξοδος	Πλήρως συνδεδεμένο	-	10	-	-	Softmax

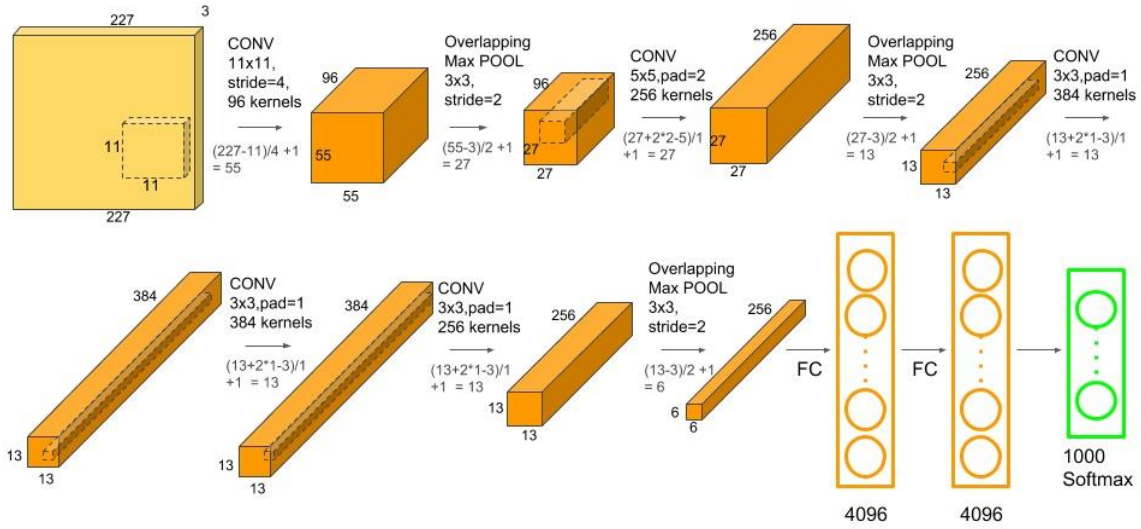
Η έρευνα αυτή σημείωσε μεγάλη επιτυχία και κέντρισε το ενδιαφέρον των μελετητών για τη μελέτη των νευρωνικών δικτύων. Η αναγνώριση απλών ψηφιακών εικόνων είναι η πιο κλασική εφαρμογή του LeNet. Το LeNet-5 σήμανε την εμφάνιση του CNN και ορίζει τα κύρια δομικά στοιχεία του, με τα σημερινά μοντέλα να το χρησιμοποιούν ως πρότυπο.

- **AlexNet**: Το AlexNet είναι ένα βαθύ συνελικτικό μοντέλο νευρωνικού δικτύου, που σχεδιάστηκε από τον Alex Krizhevsky σε συνεργασία με τους Ilya Sutskever και Geoffrey Hinton [32]. Το μοντέλο κατέλαβε την πρώτη θέση στον ετήσιο διαγωνισμό του ImageNet "ImageNet Large Scale Visual Recognition Challenge" (ILSVRC) το 2012, καθώς πέτυχε σφάλμα Top-5 15.3%, με διαφορά 10.8% από το δεύτερο. Το βάθος του μοντέλου ήταν απαραίτητο για την υψηλή του απόδοση, η οποία ενώ ήταν υπολογιστικά ακριβή, κατάφερε να γίνει εφικτή με τη χρήση μονάδων επεξεργασίας γραφικών (GPU) κατά την εκπαίδευση. Η αρχιτεκτονική του μοντέλου αποτελείται από 8 επίπεδα:

- 5 συνελικτικά στρώματα,
- 3 στρώματα υποδειγματοληψίας μέγιστης τιμής (max pooling),
- 2 στρώματα κανονικοποίησης,
- 2 πλήρως συνδεδεμένα στρώματα.

Στη συνέχεια θα γίνει περιγραφή της αρχιτεκτονικής του AlexNet:

1. Είσοδος: Εισαγωγή δεδομένων 3 εικόνων μεγέθους 224×224, που λόγω χρήσης γεμίσματος (padding) μπορεί να γίνει και 227×227×3.
2. Στρώμα 1: Στρώμα συνέλιξης με 96 φίλτρα διαστάσεων 11×11. Γίνεται χρήση της συνάρτησης ενεργοποίησης ReLU και ορίζεται βήμα ίσο με 4. Η έξοδος είναι ένας χάρτης χαρακτηριστικών διαστάσεων 55×55×96.
3. Στρώμα 2: Στρώμα υποδειγματοληψίας που εξάγει 96 γραφήματα χαρακτηριστικών μεγέθους 27×27. Εφαρμόζεται συνάρτηση μέγιστης τιμής, γίνεται χρήση φίλτρου μεγέθους 3×3 και βήματος ίσου με 2. Επομένως, η έξοδος θα είναι μεγέθους 27×27×96.
4. Στρώμα 3: Δεύτερο στρώμα συνέλιξης με 256 φίλτρα διαστάσεων 5×5. Ως συνάρτηση ενεργοποίησης παραμένει η ReLU, όμως αλλάζει το βήμα σε 1 και προστίθεται padding ίσο με 2. Η έξοδος που παίρνουμε είναι διαστάσεων 27×27×256.
5. Στρώμα 4: Δεύτερο στρώμα μέγιστης υποδειγματοληψίας που εξάγει 256 γραφήματα χαρακτηριστικών μεγέθους 13×13. Χρησιμοποιείται φίλτρο μεγέθους 3×3 με βήμα 2. Ο χάρτης χαρακτηριστικών που προκύπτει είναι διαστάσεων 13×13×256.
6. Στρώμα 5: Τρίτο στρώμα συνέλιξης με 384 φίλτρα διαστάσεων 3×3. Ως συνάρτηση ενεργοποίησης παραμένει η ReLU, όμως αλλάζουν το βήμα και το padding σε 1. Οπότε, ο χάρτης χαρακτηριστικών έχει διαστάσεις 13×13×384.
7. Στρώμα 6: Τέταρτο στρώμα συνέλιξης ίδιο με το προηγούμενο του. Η έξοδος παραμένει αμετάβλητη, διαστάσεων 13×13×384.
8. Στρώμα 7: Τελικό στρώμα συνέλιξης με 256 φίλτρα διαστάσεων 3×3. Η συνάρτηση ενεργοποίησης είναι η ReLU, με το βήμα και το padding να παραμένουν 1. Ο χάρτης χαρακτηριστικών που προκύπτει είναι διαστάσεων 13×13×256.
9. Στρώμα 8: Τελικό στρώμα μέγιστης υποδειγματοληψίας που εξάγει 256 γραφήματα χαρακτηριστικών μεγέθους 6×6. Χρησιμοποιείται φίλτρο μεγέθους 3×3 με βήμα 2. Ο χάρτης χαρακτηριστικών που προκύπτει είναι διαστάσεων 6×6×256.
10. Στρώμα 9: Έπειτα από όλα αυτά, έχουμε το πρώτο στρώμα αραίωσης (Dropout), με ποσοστό dropout ίσο με 0.5.
11. Στρώμα 10: Πλήρως συνδεδεμένο στρώμα με 4096 νευρώνες. Η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η ReLU.
12. Στρώμα 11: Ακολουθεί ένα ακόμα στρώμα αραίωσης με σταθερό ποσοστό dropout 0.5.
13. Στρώμα 12: Πλήρως συνδεδεμένο στρώμα με 4096 νευρώνες. Η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η ReLU.
14. Στρώμα 13: Τελευταίο πλήρως συνδεδεμένο στρώμα με 1000 νευρώνες. Η συνάρτηση ενεργοποίησης που χρησιμοποιείται σε αυτό το επίπεδο είναι η softmax.



Σχήμα 3.9: Αρχιτεκτονική AlexNet.

Πίνακας 3.2: Περίληψη αρχιτεκτονικής AlexNet.

Στρώμα		**Μέγεθος Χάρτη Χαρακτηριστικών**	**Φίλτρο**	**Βήμα**	**Γέμισμα**	**Συνάρτηση Ενεργοποίησης**
Είσοδος	3 εικόνες	227×227×3	-	-	-	-
1	Συνέλιξη	55×55×96	11×11	4	-	ReLU
2	Μέγιστη Υποδειγματοληψία	27×27×96	3×3	2	-	-
3	Συνέλιξη	27×27×256	5×5	1	2	ReLU
4	Μέγιστη Υποδειγματοληψία	13×13×256	3×3	2	-	-
5	Συνέλιξη	13×13×384	3×3	1	1	ReLU
6	Συνέλιξη	13×13×384	3×3	1	1	ReLU
7	Συνέλιξη	13×13×256	3×3	1	1	ReLU
8	Μέγιστη Υποδειγματοληψία	6×6×256	3×3	2	1	-
9	Αραίωση	6×6×256	Dropout rate 0.5			
10	Πλήρως συνδεδεμένο	4096	-	-	-	ReLU
11	Αραίωση	4096	Dropout rate 0.5			
12	Πλήρως συνδεδεμένο	4096	-	-	-	ReLU

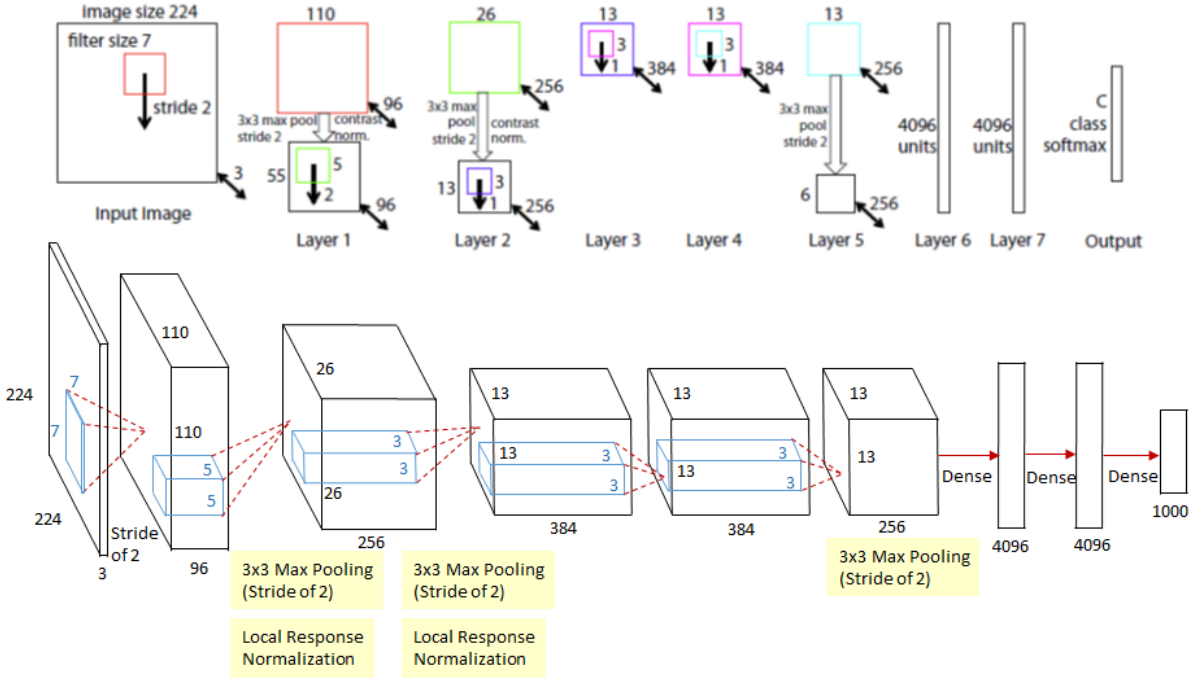
Έξοδος	Πλήρως συνδεδεμένο	1000	-	-	-	Softmax

Αλλά όλα αυτά δεν είναι τα μόνα που κάνουν το AlexNet ξεχωριστό. Παρακάτω είναι μερικά από τα χαρακτηριστικά που χρησιμοποιήθηκαν και ήταν νέες προσεγγίσεις στα συνελικτικά νευρωνικά δίκτυα:

- Rectified Linear Unit (ReLU): Το AlexNet χρησιμοποιεί τη μη γραμμική συνάρτηση ReLU αντί της tanh, που ήταν συνηθισμένη εκείνη την εποχή. Το πλεονέκτημα της συνάρτησης είναι ο χρόνος της εκπαίδευσης. Το AlexNet με τη χρήση της ReLU κατάφερε να επιτύχει ποσοστά σφάλματος 25% στο σύνολο δεδομένων CIFAR-10, 6 φορές πιο γρήγορα από ένα ισοδύναμο CNN που χρησιμοποιεί την tanh.
- Επικαλυπτόμενη Μέγιστη Υποδειγματοληψία: Τα CNN παραδοσιακά «συγκεντρώνουν» εξόδους γειτονικών ομάδων νευρώνων χωρίς επικάλυψη. Τα επικαλυπτόμενα επίπεδα μέγιστης υποδειγματοληψίας είναι παρόμοια με τα επίπεδα μέγιστης υποδειγματοληψίας, εκτός από τα παρακείμενα παράθυρα πάνω από τα οποία υπολογίζεται το μέγιστο και επικαλύπτονται το ένα με το άλλο. Οι συντάκτες του AlexNet, χρησιμοποίησαν παράθυρα συγκέντρωσης μεγέθους 3×3 με βήμα 2 μεταξύ των παρακείμενων παράθυρων. Όταν εισήγαγαν την επικάλυψη, είδαν μια μείωση του σφάλματος κατά περίπου 0,5% και διαπίστωσαν ότι τα μοντέλα με επικαλυπτόμενη συγκέντρωση γενικά δυσκολεύονται περισσότερο να παρουσιάσουν υπερ-μοντελοποίηση.
- Χρήση πολλαπλών GPU: Το AlexNet επιτρέπει την εκπαίδευση σε πολλαπλές GPU τοποθετώντας τους μισούς νευρώνες του μοντέλου σε μια GPU και τους άλλους μισούς σε μια άλλη. Αυτό σημαίνει ότι μπορεί να εκπαιδευτεί σε ένα μεγαλύτερο μοντέλο, αλλά και να μειώσει τον χρόνο εκπαίδευσης. Το AlexNet εκπαιδεύτηκε με την GPU GTX 580 με μόλις 3GB μνήμης.
- Επαύξηση Δεδομένων: Οι εισηγητές του μοντέλου AlexNet χρησιμοποίησαν τις αυξήσεις δεδομένων με κατοπτρισμό (Mirroring) και τυχαίων περικοπών (Random Crops) αυξάνοντας έτσι το μέγεθος των δεδομένων εκπαίδευσης κατά 2048. Εξήγαγαν τυχαίες περικοπές μεγέθους 227×227 από το εσωτερικό εικόνων μεγέθους 256×256 και το χρησιμοποίησαν ως είσοδο στο δίκτυο. Επίσης, πραγματοποίησαν ανάλυση βασικών στοιχείων (Principle Component Analysis) στις τιμές των εικονοστοιχείων RGB για να αλλάξουν τις εντάσεις των καναλιών RGB, γεγονός που μείωσε το σφάλμα top-1 με ποσοστό άνω του 1%.
- Αφαίρεση: Με την τεχνική “Dropout”, ένας νευρώνας απορρίπτεται από το δίκτυο με προκαθορισμένη πιθανότητα (π.χ. 0.5 ή 50%). Όταν ένας νευρώνας «απενεργοποιείται», δεν συμβάλλει στη διάδοση προς τα εμπρός ή προς τα πίσω. Αυτό σημαίνει ότι κάθε επανάληψη χρησιμοποιεί ένα διαφορετικό δείγμα των παραμέτρων του μοντέλου, το οποίο αναγκάζει κάθε νευρώνα να έχει πιο ισχυρά χαρακτηριστικά που μπορούν να χρησιμοποιηθούν με άλλους τυχαίους νευρώνες. Ωστόσο, η αφαίρεση αυξάνει τον χρόνο εκπαίδευσης που απαιτείται για τη σύγκλιση του μοντέλου, αλλά έχει αποδειχθεί ότι μειώνει σημαντικά τον κίνδυνο υπερ-μοντελοποίησης.

- **ZFNet (Zeiler-Fergus)**: Το μοντέλο των Matthew D. Zeiler και Rob Fergus αποτελεί μια βελτίωση του AlexNet [33]. Θεωρείται ως ο νικητής του διαγωνισμού ILSVRC το 2013, παρόλο που το μοντέλο Clarifai, επίσης του Zeiler και ιδρυτή της ομώνυμης εταιρίας, κατέκτησε την πρώτη θέση. Το ZFNet έχει βελτιώσει σημαντικά το ποσοστό σφαλμάτων

ταξινόμησης εικόνων σε σύγκριση με το AlexNet, καταγράφοντας σφάλμα Top-5 14.8%. Πιο συγκεκριμένα, έχει το ίδιο πλήθος στρωμάτων και νευρώνων με το AlexNet, αλλά το μέγεθος του φίλτρου του πρώτου επιπέδου και το βήμα είναι μικρότερα, ενώ παράλληλα αυξήθηκε το μέγεθος του μεσαίου συνελικτικού επιπέδου. Επιπρόσθετα με την τεχνική μέγιστης υποδειγματοληψίας, χρησιμοποιεί και την τεχνική κανονικοποίησης τοπικής αντίθεσης (Local Contrast Normalization). Είναι ένας τύπος κανονικοποίησης που εκτελεί τοπικές κανονικοποιήσεις αφαίρεσης και διαίρεσης, επιβάλλοντας ένα είδος τοπικού ανταγωνισμού μεταξύ γειτονικών χαρακτηριστικών σε έναν χάρτη χαρακτηριστικών και μεταξύ χαρακτηριστικών στην ίδια χωρική θέση σε διαφορετικούς χάρτες χαρακτηριστικών.

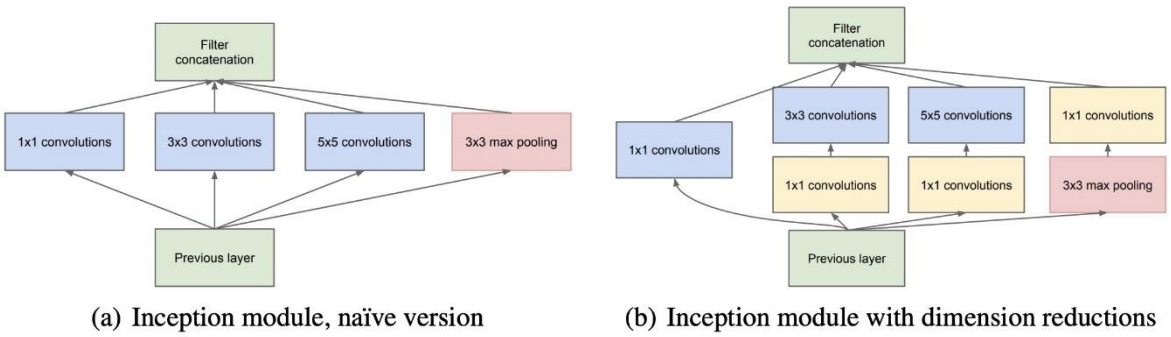


Σχήμα 3.10: Αρχιτεκτονική ZFNet.

- **GoogLeNet**: Το μοντέλο GoogLeNet προτάθηκε από ερευνητές της Google για τον διαγωνισμό ILSVRC του 2014, όπου και κατέλαβε την πρώτη θέση [34]. Πέτυχε σφάλμα Top-5 6.67%, αρκετά μειωμένο σε σύγκριση με των προηγούμενων νικητών AlexNet και ZFNet. Είναι ένα βαθύ συνελικτικό νευρωνικό δίκτυο, αποτελούμενο από 22 επίπεδα (27 με της συγκέντρωσης), που είναι μια παραλλαγή του Inception Network. Το "Inception Module" είναι μια αρχιτεκτονική νευρωνικών δικτύων που αξιοποιεί την ανίχνευση χαρακτηριστικών σε διαφορετικές κλίμακες μέσω συνελίξεων με διαφορετικά φίλτρα και μειώνει το υπολογιστικό κόστος της εκπαίδευσης ενός εκτεταμένου δικτύου μέσω μείωσης διαστάσεων.
  Η αρχιτεκτονική GoogLeNet είναι πολύ διαφορετική από τις προηγούμενες αρχιτεκτονικές αιχμής όπως το AlexNet και το ZFNet. Χρησιμοποιεί πολλά διαφορετικά είδη μεθόδων, όπως η συνέλιξη 1×1, το Inception Module που αναφέραμε, και το Global Average Pooling. Στη συνέχεια, θα αναλύσουμε τις τεχνικές:
  - Συνέλιξη 1×1: Αυτές οι συνελίξεις χρησιμοποιήθηκαν για να μειώσουν τον αριθμό των παραμέτρων (βάρη και πόλωση) της αρχιτεκτονικής. Μειώνοντας τις παραμέτρους αυξάνουμε και το βάθος της αρχιτεκτονικής. Για παράδειγμα, εάν θέλουμε να εκτελέσουμε μια συνέλιξη 5×5 με 48 φίλτρα, χωρίς να χρησιμοποιήσουμε την συνέλιξη 1×1, τότε θα έχουμε συνολικά (14×14×48) × (5×5×480) = 112.9 εκατομμύρια

παραμέτρους. Ενώ για το ίδιο, αλλά με συνέλιξη 1×1, θα έχουμε (14×14×16) × (1×1×480) + (14×14×48) × (5×5×16) = 1.5M + 3.8M = 5.3 εκατομμύρια παραμέτρους.

- Global Average Pooling: Σε προηγούμενες αρχιτεκτονικές όπως το AlexNet, τα πλήρως συνδεδεμένα επίπεδα χρησιμοποιούνται στο τέλος του δικτύου. Αυτά τα πλήρως συνδεδεμένα επίπεδα περιέχουν την πλειονότητα των παραμέτρων πολλών αρχιτεκτονικών που προκαλούν αύξηση του κόστους υπολογισμού. Στην αρχιτεκτονική GoogLeNet, υπάρχει μια μέθοδος που ονομάζεται Γενική Μέση Συγκέντρωση που χρησιμοποιείται στο τέλος του δικτύου. Αυτό το επίπεδο παίρνει έναν χάρτη χαρακτηριστικών 7×7 και τον υπολογίζει κατά μέσο όρο σε 1×1. Αυτό μειώνει επίσης τον αριθμό των εκπαιδεύσιμων παραμέτρων στο 0 και βελτιώνει την ακρίβεια top-1 κατά 0,6%.
- Inception Module: Σε αυτήν την αρχιτεκτονική, υπάρχει ένα σταθερό μέγεθος συνέλιξης για κάθε στρώμα. Στο Inception Module, η συνέλιξη 1×1, 3×3, 5×5 και η μέγιστη υποδειγματοληψία 3×3, που εκτελούνται με παράλληλο τρόπο στην είσοδο και την έξοδο, στοιβάζονται μαζί για να παραχθεί το τελικό αποτέλεσμα. Η ιδέα είναι πως τα φίλτρα συνέλιξης διαφορετικών μεγεθών θα χειρίζονται καλύτερα τα αντικείμενα σε πολλαπλή κλίμακα.
- Βοηθητικός ταξινομητής για εκπαίδευση: Η αρχιτεκτονική Inception χρησιμοποιεί μερικούς ενδιάμεσους κλάδους ταξινομητή στη μέση, οι οποίοι χρησιμοποιούνται μόνο κατά την εκπαίδευση. Αυτοί οι κλάδοι αποτελούνται από ένα επίπεδο υποδειγματοληψίας μέσης τιμής 5×5 με βήμα 3, μια συνέλιξη 1×1 με 128 φίλτρα, δύο πλήρως συνδεδεμένα στρώματα των 1024 και 1000 εξόδων αντίστοιχα, και ένα στρώμα ταξινόμησης Softmax.



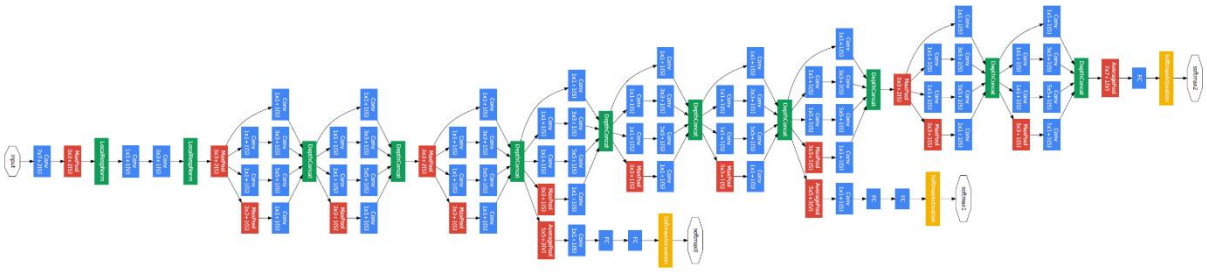
(a) Inception module, naïve version (b) Inception module with dimension reductions

Σχήμα 3.11: Inception Module του GoogLeNet.

Πίνακας 3.3: Περίληψη αρχιτεκτονικής GoogLeNet.

Στρώμα		**Μέγεθος Χάρτη Χαρακτηριστικών**	**Φίλτρο/Βήμα**
Είσοδος	3 εικόνες	224×224×3	-
1	Συνέλιξη	112×112×64	(7×7)/2
2	Μέγιστη Υποδειγματοληψία	56×56×64	(3×3)/2

3	Συνέλιξη	56×56×192	(1×1)/1
4	Συνέλιξη	56×56×192	(3×3)/1
5	Μέγιστη Υποδειγματοληψία	28×28×192	(3×3)/2
6	Inception	28×28×256	-
7	Inception	28×28×480	-
8	Μέγιστη Υποδειγματοληψία	14×14×480	(3×3)/2
9	Inception	14×14×512	-
10	Inception	14×14×512	-
11	Inception	14×14×512	-
12	Inception	14×14×528	-
13	Inception	14×14×832	-
14	Μέγιστη Υποδειγματοληψία	7×7×832	(3×3)/2
15	Inception	7×7×832	-
16	Inception	7×7×1024	-
17	Μέση Υποδειγματοληψία	1×1×1024	(7×7)/1
18	Αραίωση	1×1×1024	Dropout rate 0.4
19	Πλήρως συνδεδεμένο	1000	-
Έξοδος	Πλήρως συνδεδεμένο (Softmax)	1000	-



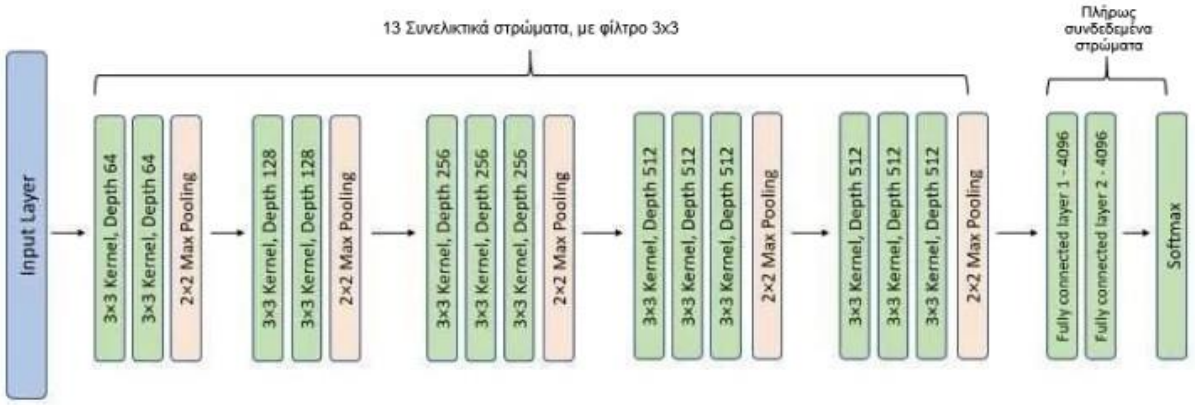
Σχήμα 3.12: Αρχιτεκτονική GoogLeNet.

- **VGGNet**: Το μοντέλο VGGNet αναπτύχθηκε από την ομάδα Visual Geometry Group για τον διαγωνισμό ILSVRC του 2014, όπου και απέσπασε την δεύτερη θέση με μικρή διαφορά από το GoogLeNet [35]. Στην ταξινόμηση πέτυχε σφάλμα Top-5 6.8%. Υπάρχουν 4 παραλλαγές του μοντέλου ανάλογα με το βάθος του, τα VGG-11, VGG-13, VGG-16 και VGG-19, με τα τελευταία δυο να είναι τα πιο δημοφιλή. Ο αριθμός έπειτα από το όνομα δείχνει το συνολικό

πλήθος των συνελικτικών στρωμάτων που χρησιμοποιούνται στο δίκτυο, χωρίς να προσμετρώνται τα στρώματα υποδειγματοληψίας.

Το μοντέλο VGG βασίζεται στα πιο βασικά χαρακτηριστικά των συνελικτικών νευρωνικών δικτύων και είναι κατασκευασμένο με πολύ μικρά συνελικτικά στρώματα. Για παράδειγμα το VGG-16 αποτελείται από 13 συνελικτικά στρώματα και τρία πλήρως συνδεδεμένα στρώματα. Ας ρίξουμε μια σύντομη ματιά στην αρχιτεκτονική του VGG:

- Είσοδος: Εισαγωγή δεδομένων 3 εικόνων μεγέθους 224×224.
- Συνελικτικά στρώματα: Τα συνελικτικά στρώματα του VGG αξιοποιούν ένα ελάχιστο υποδεκτικό πεδίο, όπως 3×3. Επιπλέον, υπάρχουν και φίλτρα συνέλιξης 1×1 που λειτουργούν ως γραμμικός μετασχηματισμός της εισόδου. Ακολουθεί η συνάρτηση ενεργοποίησης ReLU. Το βήμα συνέλιξης είναι σταθερό στο 1 για να διατηρείται η χωρική ανάλυση μετά τη συνέλιξη.
- Κρυφά επίπεδα: Όλα τα κρυφά επίπεδα στο δίκτυο VGG χρησιμοποιούν τη συνάρτηση ενεργοποίησης ReLU. Επιπλέον, γίνεται χρήση στρωμάτων υποδειγματοληψίας μέγιστης τιμής.
- Πλήρως συνδεδεμένα στρώματα: Το VGGNet έχει 3 πλήρως συνδεδεμένα στρώματα, με τα πρώτα δυο να έχουν 4096 νευρώνες, ενώ το τελευταίο να είναι ένα στρώμα Softmax με 1000 νευρώνες.



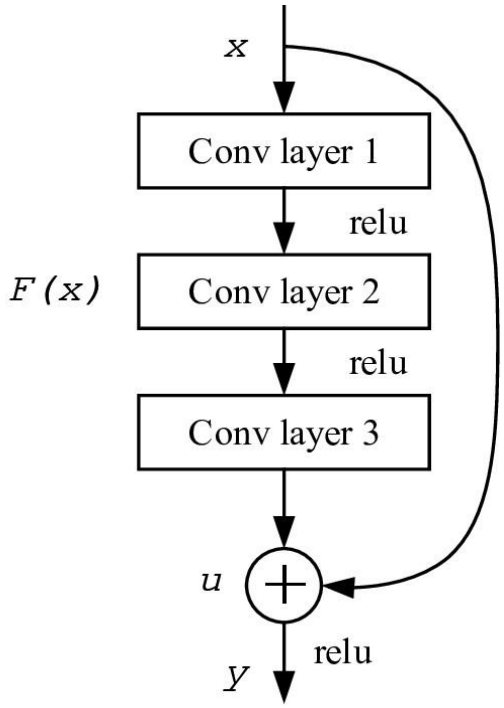
Σχήμα 3.13: Παράδειγμα αρχιτεκτονικής του VGG-16.

- **ResNet**: Το Residual Network (ResNet) είναι ένα μοντέλο Βαθιάς Μάθησης που σχεδιάστηκε από τους Shaoqing Ren, Kaiming He, Jian Sun και Xiangyu Zhang, μέλη της εταιρίας Microsoft Research Asia, για τον διαγωνισμό ILSVRC του 2015, λαμβάνοντας την πρώτη θέση με επίδοση σφάλματος Top-5 3.57% [36]. Το μοντέλο αυτό είναι το πρώτο που κατάφερε να πέτυχει σφάλμα μικρότερο του 5%.

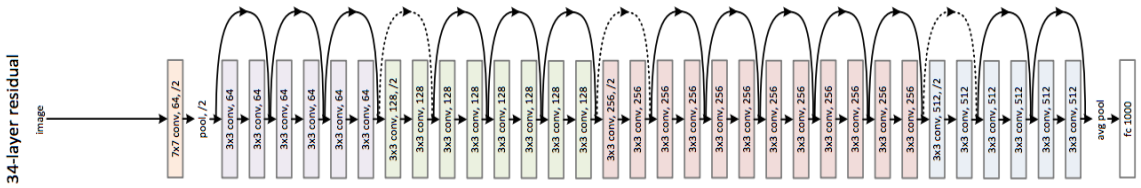
  Η βασική ιδέα του ResNet είναι η εισαγωγή μιας «σύνδεσης συντόμευσης ταυτότητας» (identity shortcut connection) που παρακάμπτει ένα ή περισσότερα επίπεδα. Το συνελικτικό υποδίκτυο $F(\mathbf{x})$ του ResNet μπλοκ μοντελοποιεί το υπόλοιπο (residual) $F(\mathbf{x}) = \mathbf{u} - \mathbf{x}$.

  Η αρχιτεκτονική του μοντέλου αποτελείται από:

  - 1 απλό συνελικτικό στρώμα,
  - 4 στρώματα υποδειγματοληψίας μέγιστης τιμής,
  - 1 στρώμα υποδειγματοληψίας μέσης τιμής,
  - 50, 100, ή 152 ResNets και
  - 1 στρώμα εξόδου Softmax με 1000 νευρώνες.



Σχήμα 3.14: ResNet block.

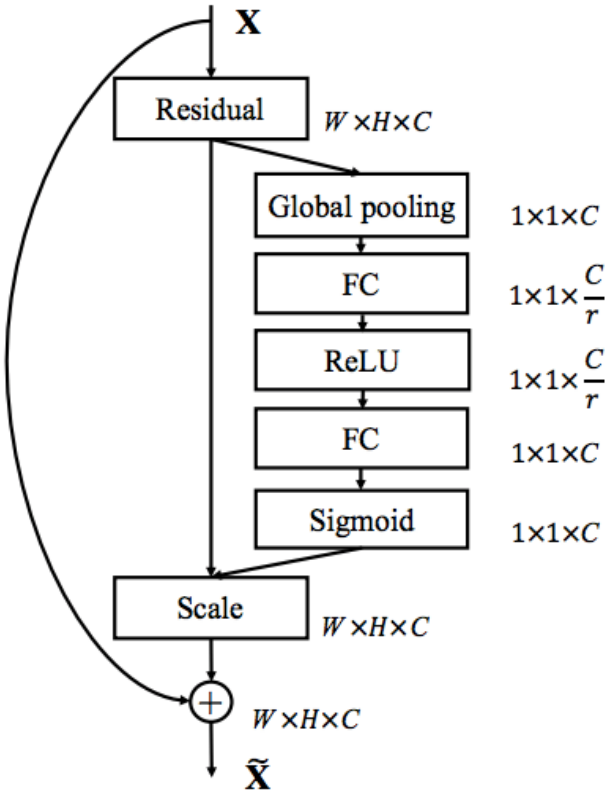


Σχήμα 3.15: Παράδειγμα αρχιτεκτονικής μοντέλου ResNet με 34 επίπεδα.

- **SENet**: Το Squeeze & Excitation (SENet) προτάθηκε από τους Jie Hu, Li Shen, Samuel Albanie, Gang Sun και Enhua Wu για τον διαγωνισμό ILSVRC του 2017, λαμβάνοντας την πρώτη θέση με επίδοση σφάλματος Top-5 2.25%. Το δίκτυο SENet αποτελεί μια βελτίωση του ResNet, με την διαφορά τους να βρίσκεται στον τύπο του βασικού μπλοκ.
  Το βασικό δομικό στοιχείο του μοντέλου είναι το SE-Block που επεκτείνει το ResBlock και αποτελείται από τρεις λειτουργίες:
  - Συμπίεση πληροφοριών: Χρησιμοποιείται για την εξαγωγή καθολικών χαρακτηριστικών από τον χάρτη χαρακτηριστικών. Ο χάρτης χαρακτηριστικών είναι η έξοδος του προηγουμένου στρώματος συνέλιξης διαστάσεων B×W×H×C. Στη λειτουργία αυτή γίνεται χρήση ενός καθολικού στρώματος υποδειγματοληψίας μέσης τιμής που μειώνει τις διαστάσεις του χάρτη χαρακτηριστικών σε B×1×1×C.
  - Διέγερση: Για τη λειτουργία διέγερσης, χρησιμοποιείται ένα πλήρως συνδεδεμένο πολυστρωματικό Perceptron (MLP) με δομή συμφόρησης. Το MLP αποτελείται από τρία επίπεδα:
    i. Την είσοδο που έχει μέγεθος B×1×1×C, το οποίο μειώνεται σε B×C. Έτσι, το στρώμα εισόδου έχει τον αριθμό C των νευρώνων,
    ii. Το κρυφό στρώμα που μειώνει τον αριθμό των νευρώνων κατά συντελεστή r. Έτσι, το κρυφό στρώμα έχει έναν αριθμό C/r νευρώνων,

iii. Το επίπεδο εξόδου, οπού ο αριθμός των νευρώνων αυξάνεται ξανά στο C.

- Κλιμάκωση: Ο τανυστής μεγέθους B×1×1×C διέρχεται από μια σιγμοειδή συνάρτηση ενεργοποίησης. Η συνάρτηση μετατρέπει τις τιμές του τανυστή σε 0 και 1. Εάν η τιμή είναι κοντά στο 0 σημαίνει ότι το κανάλι είναι λιγότερο σημαντικό, επομένως, οι τιμές του καναλιού χαρακτηριστικών θα μειωθούν, ενώ εάν η τιμή είναι κοντά στο 1, αυτό σημαίνει ότι το κανάλι είναι σημαντικό. Στη συνέχεια, εκτελούμε έναν κατά στοιχείο πολλαπλασιασμό μεταξύ της εξόδου της συνάρτησης ενεργοποίησης και του χάρτη χαρακτηριστικών εισόδου.



Σχήμα 3.16: Διάγραμμα SE-Block.

Οι συγγραφείς δείχνουν ότι προσθέτοντας SE-block στο ResNet-50 έχουμε σχεδόν την ίδια ακρίβεια με το ResNet-101. Αυτό είναι εντυπωσιακό για ένα μοντέλο που απαιτεί μόνο το μισό του υπολογιστικό κόστους. Η εργασία διερευνά περαιτέρω άλλες αρχιτεκτονικές όπως οι Inception, Inception-ResNet και ResNeXt.

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [9]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [9]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [9]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [43]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [43]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
BN-Inception [14]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [38]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Σχήμα 3.17: Βελτιώσεις του SENet συγκριτικά με τις υπάρχουσες αρχιτεκτονικές.

Με βάση όλα τα παραπάνω, τα Συνελικτικά Νευρωνικά Δίκτυα αποτελούν έως σήμερα την αιχμή της τεχνολογίας σε πολλές εφαρμογές, ειδικότερα στην αναγνώριση εικόνων και βίντεο. Την τελευταία δεκαετία, έχουν επιδείξει σημαντικά αποτελέσματα σε μια σειρά προβλημάτων αναγνώρισης και ταξινόμησης. Στον χώρο της αυτοκίνησης, παρόλο που η εποχή των αυτόνομων οχημάτων δεν έχει αναδυθεί πλήρως, η υποκείμενη τεχνολογία έχει αρχίσει να εισχωρεί στα αυτοκίνητα, βελτιώνοντας την ασφάλεια του οδηγού και των επιβατών μέσω λειτουργιών, όπως η αναγνώριση των σημάτων οδικής κυκλοφορίας και της ανίχνευσης λωρίδας.

3.2 You Only Look Once (YOLO)

Τις προηγούμενες δεκαετίες, οι μέθοδοι που χρησιμοποιήθηκαν για την αντιμετώπιση της ανίχνευσης αντικειμένων αποτελούνταν από δύο στάδια: i) την εξαγωγή διαφορετικών περιοχών από την εικόνα χρησιμοποιώντας συρόμενα παράθυρα διαφορετικών μεγεθών και ii) την εφαρμογή του προβλήματος ταξινόμησης για τον προσδιορισμό της κλάσης των αντικειμένων. Αυτές οι προσεγγίσεις έχουν το μειονέκτημα ότι απαιτούν μεγάλο όγκο υπολογισμών και αναλύονται σε πολλαπλά στάδια. Κατά συνέπεια, καθίσταται δύσκολη η βελτιστοποίηση του συστήματος από άποψη ταχύτητας.

Το 2016, ο Joseph Redmon μαζί με τους συνεργάτες του από το πανεπιστημίου της Γουάσινγκτον, παρουσίασαν ένα σύστημα ανίχνευσης αντικειμένων που εκτελεί όλα τα βασικά στάδια της ανίχνευσης χρησιμοποιώντας ένα μόνο νευρωνικό δίκτυο για πρώτη φορά, τον αλγόριθμο YOLO (You Only Look Once) [9]. Πρόκειται για έναν αλγόριθμο που ανιχνεύει και αναγνωρίζει διάφορα αντικείμενα σε μια εικόνα, σε πραγματικό χρόνο. Επαναπροσδιορίζει την ανίχνευση αντικειμένων ως πρόβλημα παλινδρόμησης και παρέχει τις πιθανότητες κλάσεων των ανιχνευόμενων εικόνων.

Ο αλγόριθμος YOLO χρησιμοποιεί συνελικτικά νευρωνικά δίκτυα (CNN) για την ανίχνευση αντικειμένων σε πραγματικό χρόνο. Όπως υποδηλώνει το όνομα, ο αλγόριθμος απαιτεί μόνο μία προς τα εμπρός διάδοση μέσω ενός νευρωνικού δικτύου για την ανίχνευση αντικειμένων. Αυτό σημαίνει ότι η πρόβλεψη σε ολόκληρη την εικόνα γίνεται με μία μόνο εκτέλεση του αλγορίθμου. Το ΣΝΔ χρησιμοποιείται για την πρόβλεψη διαφόρων πιθανοτήτων κλάσης και οριοθέτησης πλαισίων ταυτόχρονα.

Τα τελευταία 6 χρόνια, ο αλγόριθμος έχει αναβαθμιστεί σε 5 εκδόσεις συμπεριλαμβανομένης και της αρχικής. Κάθε έκδοση έχει αναβαθμιστεί και ενσωματωθεί με τις πιο προηγμένες ιδέες που προέρχονται από την ερευνητική κοινότητα της Μηχανικής Όρασης υπολογιστών. Οι πρώτες τρεις εκδόσεις αναπτυχθήκαν από τον δημιουργό του YOLO, Joseph Redmon [9], [37], [38]. Η τέταρτη έκδοση δημοσιεύτηκε από τον Alexey Bochkovskiy το 2020 και χρησιμοποιεί την αρχιτεκτονική Darknet του Joseph Redmon [39]. Η πέμπτη έκδοση αναπτύχθηκε ένα μηνά αργότερα από τον ερευνητή Glenn Jocher και την ομάδα του Ultralytics, με τον αλγόριθμο να βασίζεται στο PyTorch [40]. Το YOLOv5 έχει μερικές διαφορές και βελτιώσεις σε σχέση με την προηγούμενη έκδοση, προκαλώντας τις εντυπώσεις με την απόδοση του στην ανίχνευση αντικειμένων σε πραγματικό χρόνο.

Λειτουργία αλγορίθμου YOLO

Ο αλγόριθμος YOLO λειτουργεί χρησιμοποιώντας τις ακόλουθες τρεις τεχνικές:

- Υπολειπόμενα μπλοκ (Residual blocks)
- Παλινδρόμηση πλαισίου οριοθέτησης (Bounding box regression)
- Διασταύρωση πάνω από την ένωση (Intersection Over Union (IOU))

Υπολειπόμενα μπλοκ

Η εικόνα εισόδου χωρίζεται σε διάφορα πλέγματα όπου κάθε πλέγμα είναι διαστάσεων $S \times S$. Κάθε κελί πλέγματος θα ανιχνεύει αντικείμενα που εμφανίζονται μέσα του. Εάν το κέντρο ενός αντικειμένου εμφανίζεται σε ένα κελί πλέγματος, τότε αυτό το κελί είναι υπεύθυνο για την ανίχνευση του συγκεκριμένου αντικειμένου. Επομένως, όλα τα άλλα κελιά αγνοούν ακόμη και αυτή την εμφάνιση αντικειμένου που αποκαλύπτεται σε πολλαπλά κελιά. Το παρακάτω σχήμα (3.18) δείχνει πως μια εικόνα εισόδου χωρίζεται σε ισομεγέθη πλέγματα.

Σχήμα 3.18: Χωρισμός εικόνας σε ισομεγέθη πλέγματα.

Παλινδρόμηση πλαισίου οριοθέτησης

Ένα πλαίσιο οριοθέτησης είναι ένα περίγραμμα που επισημαίνει ένα αντικείμενο σε μια εικόνα. Προκειμένου να εφαρμοστεί η ανίχνευση αντικειμένων, κάθε κελί πλέγματος προβλέπει B οριοθετημένα πλαίσια με τις παραμέτρους τους και τους βαθμούς εμπιστοσύνης για αυτά τα πλαίσια. Αυτή η βαθμολογία αντικατοπτρίζει την παρουσία ή την απουσία ενός αντικειμένου στο πλαίσιο οριοθέτησης. Η βαθμολογία εμπιστοσύνης ορίζεται ως εξής:

$$confidence\ score = p(Object) * IOU_{pred}^{truth} \quad (3.2.1)$$

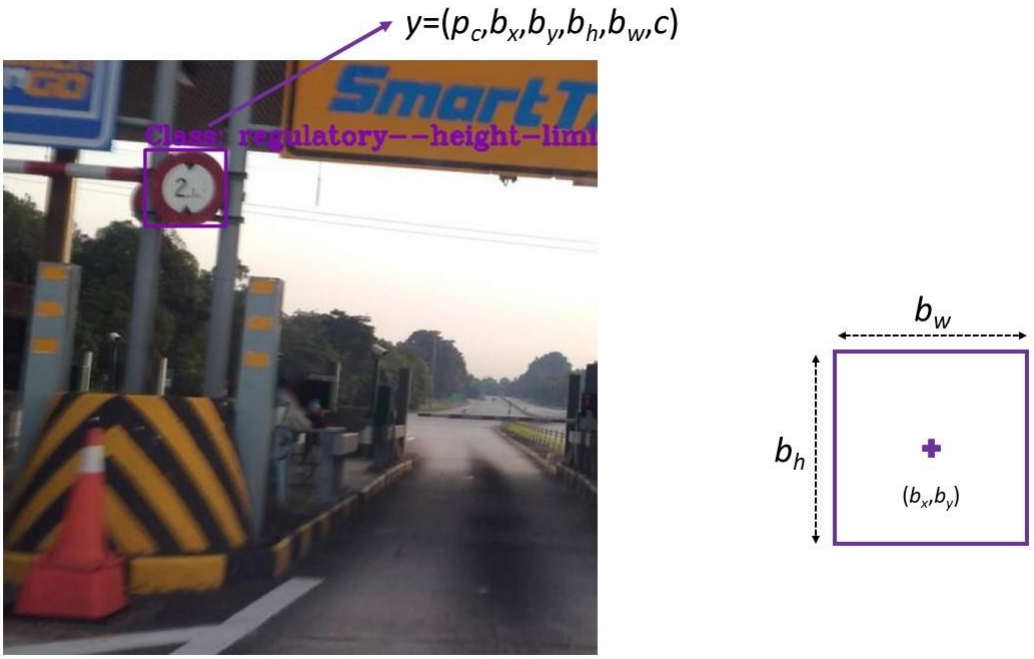
Όπου το *p(Object)* είναι η πιθανότητα να υπάρχει ένα αντικείμενο μέσα στο κελί και το IOU_{pred}^{truth} είναι η τομή πάνω από την ένωση του πλαισίου πρόβλεψης και του πλαισίου αληθείας βάσης. Το *p(Object)* βρίσκεται μεταξύ 0 και 1, επομένως η βαθμολογία εμπιστοσύνης θα είναι κοντά στο 0 εάν δεν υπάρχει αντικείμενο σε αυτό το κελί, ενώ διαφορετικά θα ισούται με IOU_{pred}^{truth}.

Εκτός αυτού, κάθε πλαίσιο οριοθέτησης στην εικόνα αποτελείται από τα ακόλουθα χαρακτηριστικά:

- Πλάτος (b_w),
- Ύψος (b_h),
- Κέντρο οριοθέτησης (b_x,b_y),

- Κλάση

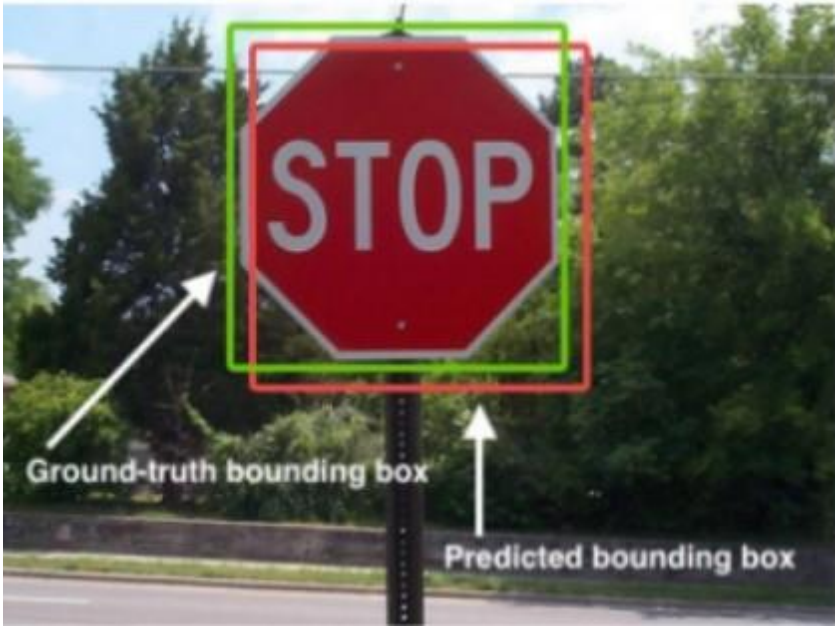
Το YOLO χρησιμοποιεί μια παλινδρόμηση μονής οριοθέτησης για να προβλέψει το ύψος, το πλάτος, το κέντρο και την κατηγορία των αντικειμένων. Η πιθανότητα ενός αντικειμένου που προβλέπεται για κάθε κλάση σε ένα κελί πλέγματος συμβολίζεται ως $p(Class_i|Object)$. Οι τιμές πιθανότητας για την κλάση *C* θα παράγουν έξοδο *C* για κάθε κελί πλέγματος. Το πλαίσιο οριοθέτησης *B* του ίδιου κελιού πλέγματος μοιράζεται ένα κοινό σύνολο προβλέψεων σχετικά με την κλάση του αντικειμένου, που σημαίνει ότι όλα τα πλαίσια οριοθέτησης στο ίδιο κελί πλέγματος έχουν την ίδια κλάση. Η παρακάτω εικόνα αντιπροσωπεύει την πιθανότητα εμφάνισης ενός αντικειμένου στο πλαίσιο οριοθέτησης.



Σχήμα 3.19: Παράμετροι ενός πλαισίου οριοθέτησης. Το p_c είναι η πιθανότητα ενός αντικειμένου να βρίσκεται μέσα στο πλαίσιο και το c η κλάση του αντικειμένου.

Διασταύρωση πάνω από την ένωση

Η διασταύρωση πάνω από την ένωση (IOU) είναι ένα φαινόμενο στην ανίχνευση αντικειμένων που περιγράφει τον τρόπο επικάλυψης των πλαισίων. Το YOLO χρησιμοποιεί το IOU για να παρέχει ένα πλαίσιο εξόδου που περιβάλλει τέλεια τα αντικείμενα. Κάθε κελί πλέγματος είναι υπεύθυνο για την πρόβλεψη των οριοθετημένων πλαισίων και των βαθμολογιών εμπιστοσύνης τους. Το IOU είναι ίσο με 1 εάν το προβλεπόμενο πλαίσιο οριοθέτησης είναι το ίδιο με το πραγματικό πλαίσιο. Αυτός ο μηχανισμός εξαλείφει τα οριοθετημένα πλαίσια που δεν είναι ίσα με το πραγματικό κουτί.

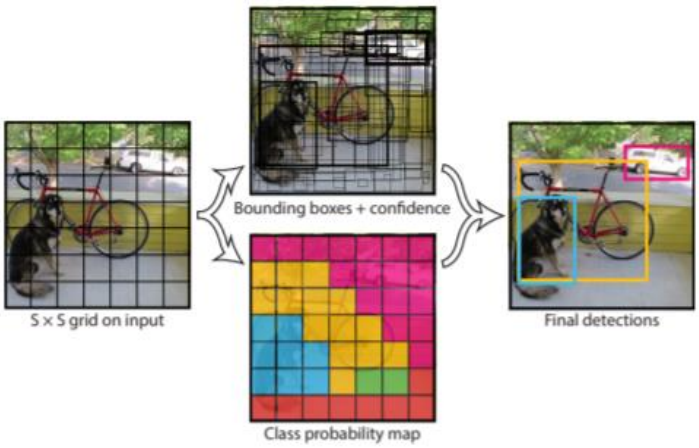


Σχήμα 3.20: Στην παραπάνω εικόνα, υπάρχουν δύο πλαίσια οριοθέτησης, ένα πράσινο και ένα κόκκινο. Το κόκκινο πλαίσιο είναι το προβλεπόμενο πλαίσιο ενώ το πράσινο πλαίσιο είναι το πραγματικό πλαίσιο. Το YOLO διασφαλίζει ότι τα δύο πλαίσια οριοθέτησης είναι ίσα.

Συνδυασμός των τεχνικών

Πρώτα, η εικόνα χωρίζεται σε κελιά πλέγματος. Κάθε κελί πλέγματος προβλέπει τα όρια *B* και παρέχει τις βαθμολογίες εμπιστοσύνης τους. Τα κελιά προβλέπουν τις πιθανότητες κλάσης για να καθορίσουν την κλάση κάθε αντικειμένου. Όλες οι προβλέψεις γίνονται ταυτόχρονα χρησιμοποιώντας ένα ενιαίο συνελικτικό νευρωνικό δίκτυο.

Η τομή πάνω από την ένωση διασφαλίζει ότι τα προβλεπόμενα οριοθετημένα πλαίσια είναι ίσα με τα πραγματικά πλαίσια των αντικειμένων. Αυτό το φαινόμενο εξαλείφει τα περιττά πλαίσια οριοθέτησης που δεν πληρούν τα χαρακτηριστικά των αντικειμένων (όπως ύψος και πλάτος). Η τελική ανίχνευση θα αποτελείται από μοναδικά πλαίσια οριοθέτησης που ταιριάζουν τέλεια στα αντικείμενα.



Σχήμα 3.21: Η παραπάνω εικόνα δείχνει πώς εφαρμόζονται οι τρεις τεχνικές για την παραγωγή των τελικών αποτελεσμάτων ανίχνευσης.

Τέλος, το YOLO εφαρμόζει Non-Maximum Suppression (NMS) για τον καθαρισμό όλων των πλαισίων οριοθέτησης που δεν περιέχουν κάποιο αντικείμενο ή περιέχουν το ίδιο αντικείμενο με άλλα πλαίσια

οριοθέτησης. Επιλέγοντας μια τιμή κατωφλίου, το NMS αφαιρεί όλα τα πλαίσια οριοθέτησης επικάλυψης που έχουν τιμή IOU μεγαλύτερη από την τιμή κατωφλίου.

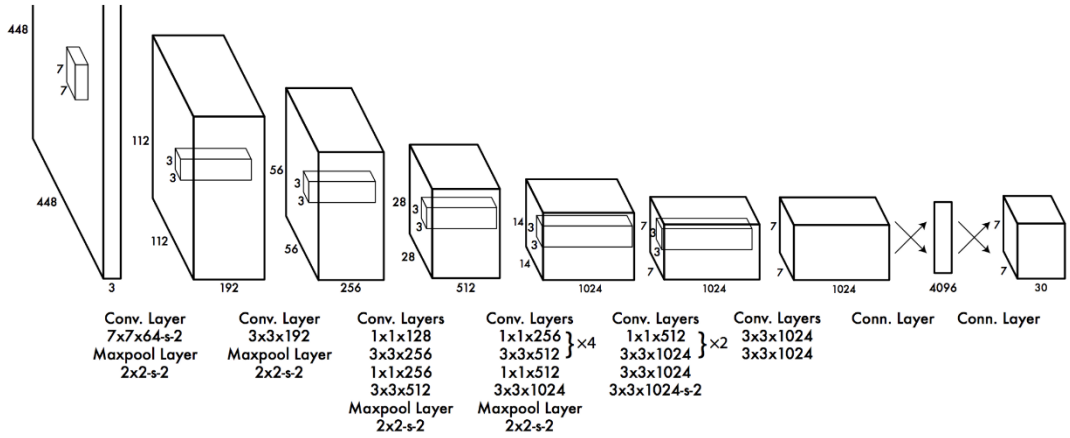
YOLOv1

Η αρχιτεκτονική της πρώτης έκδοσης του μοντέλου YOLO είναι εμπνευσμένη από το μοντέλο GoogLeNet [9]. Η ανίχνευση αντικειμένων στην εργασία αυτή πλαισιώνεται ως ένα πρόβλημα παλινδρόμησης σε χωρικά διαχωρισμένα πλαίσια οριοθέτησης και σχετικές πιθανότητες κλάσεων. Τα αρχικά συνελικτικά επίπεδα του δικτύου εξάγουν χαρακτηριστικά από την εικόνα, ενώ τα πλήρως συνδεδεμένα επίπεδα προβλέπουν τις πιθανότητες και τις συντεταγμένες εξόδου. Το βασικό μοντέλο της πρώτης έκδοσης του YOLO, επεξεργάζεται εικόνες σε πραγματικό χρόνο με ταχύτητα 45 καρέ ανά δευτερόλεπτο.

Η αρχιτεκτονική που χρησιμοποιείται στο μοντέλο ονομάστηκε Darknet από τους συγγράφεις και αποτελείται από 24 συνελικτικά επίπεδα, ακολουθούμενα από 2 πλήρως συνδεδεμένα επίπεδα. Έχοντας ως βάση το Inception Module του GoogLeNet, χρησιμοποιούνται επίπεδα μείωσης 1×1 ακολουθούμενα από επίπεδα συνέλιξης 3×3. Η τελική έξοδος του δικτύου είναι ένας τανυστής διαστάσεων 7×7×30 με τις προβλέψεις. Αυτό το επίπεδο προβλέπει τις πιθανότητες κλάσης και τις συντεταγμένες του πλαισίου οριοθέτησης. Παράλληλα, γίνεται κανονικοποίηση του πλάτους και του ύψους του πλαισίου οριοθέτησης με το πλάτος και το ύψος της εικόνας, έτσι ώστε οι τιμές να είναι μεταξύ 0 και 1.

Το τελικό επίπεδο χρησιμοποιεί μια γραμμική συνάρτηση ενεργοποίησης, αντί για την παρακάτω συνάρτηση Leaky Rectified Linear Unit (leaky ReLU) των υπόλοιπων επιπέδων:

$$\varphi(x) = \begin{cases} x, & x > 0 \\ 0.1x, & x \leq 0 \end{cases} \tag{3.2.2}$$



Σχήμα 3.22: Το δίκτυο ανίχνευσης έχει 24 συνελικτικά επίπεδα ακολουθούμενα από 2 πλήρως συνδεδεμένα επίπεδα. Τα εναλλασσόμενα συνελικτικά στρώματα 1×1 μειώνουν τον χώρο χαρακτηριστικών από τα προηγούμενα επίπεδα. Τα στρώματα συνέλιξης προεκπαιδεύοντε στο σύνολο δεδομένων ταξινόμησης ImageNet, στη μισή ανάλυση (224×224) και στη συνέχεια διπλασιάζεται η ανάλυση για την ανίχνευση.

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

Σχήμα 3.23: Αρχιτεκτονική της απλής μορφής του YOLOv1, με τα 24 στρώματα συνέλιξης και τα 2 πλήρως συνδεδεμένα στρώματα.

Το μοντέλο αξιολογήθηκε στο σύνολο δεδομένων ανίχνευσης PASCAL VOC [41]. Το δίκτυο εκπαιδεύτηκε για περίπου 135 εποχές στα σύνολα δεδομένων εκπαίδευσης και επικύρωσης από το PASCAL VOC 2007 και 2012.

Το τετραγωνικό σφάλμα αθροίσματος (sum-squared error) αποτελεί τη ραχοκοκαλιά της συνάρτησης απώλειας του YOLOv1. Σε κάθε εικόνα εισόδου υπάρχουν πολλά κελιά πλέγματος που δεν περιέχουν κανένα αντικείμενο, των οποίων η βαθμολογία εμπιστοσύνης είναι μηδέν, υπερκαλύπτοντας συχνά τη διαβάθμιση από κελιά που περιέχουν αντικείμενα. Αυτό μπορεί να οδηγήσει σε αστάθεια του μοντέλου, με αποτέλεσμα η εκπαίδευση να αποκλίνει νωρίς.

Για να αποφευχθεί μια τέτοια συντριπτική απόκλιση που οδηγεί σε απόκλιση εκπαίδευσης και αστάθεια μοντέλου, το YOLOv1 επιβάλλει την υψηλότερη ποινή για προβλέψεις από οριοθετημένα πλαίσια που περιέχουν αντικείμενα και τη χαμηλότερη για προβλέψεις όταν δεν υπάρχει αντικείμενο. Χρησιμοποιούνται δυο παράμετροι, οι λ_{coord} και λ_{noobj} για να επιτευχθεί αυτό, με την $\lambda_{coord} = 5$ και $\lambda_{noobj} = 0.5$.

Η συνάρτηση απώλειας του YOLOv1 υπολογίζεται λαμβάνοντας το ποσό της συνάρτησης απώλειας όλων των παραμέτρων των πλαισίων οριοθέτησης, συμπεριλαμβανομένων των b_x, b_y, b_w, b_h, p_c και βαθμολογία εμπιστοσύνης.

$$\mathcal{L} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$
$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{I}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

Σχήμα 3.24: Η συνάρτηση απώλειας του YOLOv1.

Ο πρώτος όρος της συνάρτησης υπολογίζει την απώλεια που σχετίζεται με την προβλεπόμενη θέση του πλαισίου οριοθέτησης και τη θέση του πλαισίου ground truth με βάση συντεταγμένες (x_{center}, y_{center}). Το $\mathbb{I}_{ij}^{obj}$ ορίζεται ως 1 εάν υπάρχει αντικείμενο μέσα στο j-οστό προβλεπόμενο πλαίσιο οριοθέτησης στο κελί i, και 0 διαφορετικά.

Ο δεύτερος όρος της συνάρτησης υπολογίζει το σφάλμα στην πρόβλεψη του πλάτους και του ύψους του πλαισίου οριοθέτησης παρόμοια με τον πρώτο όρο. Ωστόσο, το μέγεθος του σφάλματος στα μεγάλα πλαίσια επηρεάζει την εξίσωση λιγότερο από ότι στα μικρά. Καθώς το πλάτος και το ύψος κανονικοποιούνται μεταξύ 0 και 1, οι τετραγωνικές ρίζες τους αυξάνουν τις διαφορές για μικρότερες τιμές περισσότερο από τις μεγαλύτερες τιμές. Ως εκ τούτου, χρησιμοποιείται η τετραγωνική ρίζα του πλάτους και του ύψους του πλαισίου οριοθέτησης αντί για το πλάτος και το ύψος απευθείας.

Στους επόμενους δυο όρους υπολογίζεται η βαθμολογία απώλειας εμπιστοσύνης, είτε το αντικείμενο υπάρχει στο πλαίσιο οριοθέτησης είτε όχι. Η συνάρτηση απώλειας τιμωρεί το σφάλμα συντεταγμένων πλαισίου οριοθέτησης, μόνο εάν ο προγνωστικός παράγοντας είναι ¨υπεύθυνος¨ για το πλαίσιο ground truth (δηλαδή έχει το υψηλότερο IOU από οποιοδήποτε προγνωστικό σε αυτό το κελί πλέγματος).

Ο τελευταίος όρος της συνάρτησης είναι παρόμοιος με την κανονική απώλεια ταξινόμησης που υπολογίζει την απώλεια πιθανότητας κλάσης, εκτός από τον όρο $\mathbb{I}_{ij}^{obj}$. Αυτός ο όρος χρησιμοποιείται επειδή το YOLOv1 δεν τιμωρεί τα σφάλματα ταξινόμησης ακόμη και όταν δεν υπάρχουν αντικείμενα στο κελί.

Το YOLOv1 επιβάλλει ισχυρούς χωρικούς περιορισμούς στις προβλέψεις πλαισίων οριοθέτησης, αφού κάθε κελί πλέγματος προβλέπει μόνο δύο πλαίσια και μπορεί να έχει μόνο μία κλάση. Αυτός ο χωρικός περιορισμός περιορίζει τον αριθμό των κοντινών αντικειμένων που μπορεί να προβλέψει το μοντέλο. Το μοντέλο δυσκολεύεται να βρει μικρά αντικείμενα αν εμφανίζονται ως σύμπλεγμα. Επίσης, εφόσον το μοντέλο μαθαίνει να προβλέπει οριοθετημένα πλαίσια από δεδομένα, δυσκολεύεται να γενικεύσει σε αντικείμενα σε νέους ή ασυνήθιστους λόγους διαστάσεων. Τέλος, η συνάρτηση απώλειας αντιμετωπίζει τα σφάλματα το ίδιο σε μικρά πλαίσια οριοθέτησης, έναντι σε μεγάλα πλαίσια οριοθέτησης. Ένα μικρό σφάλμα σε ένα μεγάλο πλαίσιο είναι γενικά αποδεκτό, όμως ένα μικρό σφάλμα σε ένα μικρό πλαίσιο έχει πολύ μεγαλύτερη επίδραση στο IOU.

Τέλος, οι συγγραφείς εισήγαγαν επίσης το μοντέλο fast-YOLO, ένα νευρωνικό δίκτυο με λιγότερα συνελικτικά στρώματα και με λιγότερα φίλτρα σε αυτά τα επίπεδα. Το fast-YOLO αποτελείται από 9

επίπεδα συνέλιξης. Εκτός από το μέγεθος του δικτύου, όλες οι παράμετροι εκπαίδευσης και δοκιμής είναι ίδιες μεταξύ του YOLO και του fast-YOLO. Η μικρότερη αυτή έκδοση μπορεί να επεξεργάζεται 155 καρέ ανά δευτερόλεπτο, πετυχαίνοντας το διπλάσιο mean Average Precision (mAP) από άλλους ανιχνευτές σε πραγματικό χρόνο.

YOLOv2

Στα τέλη της ίδιας χρονιάς, οι Redmon και Farhadi παρουσίασαν τη νέα έκδοση του YOLO, ονόματι YOLO9000 ή YOLOv2 [37]. Αυτό το μοντέλο μπορεί να ανιχνεύσει περισσότερες από 9000 κατηγορίες αντικειμένων σε πραγματικό χρόνο. Χρησιμοποιώντας μια νέα, πολλαπλής κλίμακας μέθοδο εκπαίδευσης, το μοντέλο μπορεί να τρέξει σε διάφορα μεγέθη εικόνων εισόδου, προσφέροντας μια εύκολη αντιστάθμιση μεταξύ ταχύτητας και ακρίβειας. Στα 67 καρέ ανά δευτερόλεπτο, το YOLOv2 πετυχαίνει 76,8 mAP στο VOC 2007, ενώ στα 40 καρέ ανά δευτερόλεπτο πετυχαίνει 78,6 mAP, υπερτερώντας των υπόλοιπων μεθόδων αιχμής, όπως το Faster R-CNN με ResNet και SSD, ενώ εξακολουθεί να λειτουργεί σημαντικά πιο γρήγορα.

Οι συγγραφείς προτείνουν ακόμη, έναν αλγόριθμο για την από κοινού εκπαίδευση στον εντοπισμό και την ταξινόμηση αντικειμένων. Με τη μέθοδο αυτή, το μοντέλο εκπαιδεύεται ταυτόχρονα και στο σύνολο δεδομένων ανίχνευσης COCO και στο σύνολο δεδομένων ταξινόμησης ImageNet. Αυτό σημαίνει ότι το YOLO9000 μπορεί να προβλέπει ανιχνεύσεις για κατηγορίες αντικειμένων που δεν έχουν δεδομένα ανίχνευσης με ετικέτα.

Η προηγούμενη έκδοση του YOLO έπασχε από μια ποικιλία ελλείψεων σε σχέση με τα άλλα μοντέλα ανίχνευσης. Το YOLO έχει σχετικά χαμηλή ανάκληση σε σύγκριση με τις μεθόδους που βασίζονται σε προτάσεις περιοχής για την αναγνώριση των αντικειμένων. Έπειτα, το YOLO κάνει σημαντικό αριθμό σφαλμάτων εντοπισμού. Οπότε, κύριος στόχος της δεύτερης έκδοσης είναι η βελτίωση της ανάκλησης και του εντοπισμού, διατηρώντας παράλληλα την ακρίβεια ταξινόμησης. Οι βελτιώσεις από το YOLOv1 στο YOLOv2 είναι οι ακόλουθες:

- **Batch Normalization**: Η κανονικοποίηση συνόλου ομαλοποιεί το επίπεδο εισόδου αλλάζοντας ελαφρά και κλιμακώνοντας τις ενεργοποιήσεις. Είναι μια από τις πιο δημοφιλείς μεθόδους κανονικοποίησης στα μοντέλα Βαθιάς Μάθησης, καθώς επιτρέπει την ταχύτερη και πιο σταθερή εκπαίδευση σε βαθιά νευρωνικά δίκτυα, σταθεροποιώντας την κατανομή των στρωμάτων εισόδου κατά τη διάρκεια της εκπαίδευσης [42]. Αυτή η τεχνική μειώνει τον χρόνο εκπαίδευσης, αυξάνει τη γενίκευση του δικτύου και βοηθά στην τακτοποίηση του μοντέλου. Με την προσθήκη κανονικοποίησης συνόλου σε όλα τα συνελικτικά επίπεδα στο YOLO, έχει βελτιωθεί το mAP κατά τουλάχιστον 2%. Επίσης, δεν χρειάζεται να γίνει χρήση πρόσθετων στρωμάτων Dropout στο δίκτυο για να αποφευχθεί η υπερπροσαρμογή.
- **High Resolution Classifier**: Το αρχικό YOLO εκπαιδεύει το δίκτυο ταξινομητή σε ανάλυση 224×224 και αυξάνει την ανάλυση σε 448×448 για την ανίχνευση. Ειδικότερα, στο YOLOv1 τα πρώτα 20 επίπεδα συνέλιξης χρησιμοποιήθηκαν για την εκπαίδευση του δικτύου ταξινόμησης με εικόνα εισόδου 224×224, και στη συνέχεια προστέθηκαν τα υπόλοιπα 4 στρώματα συνέλιξης με τα 2 πλήρως συνδεδεμένα στρώματα. Αυτό σημαίνει ότι το δίκτυο πρέπει ταυτόχρονα να μεταβεί στην ανίχνευση αντικειμένων εκμάθησης και να προσαρμοστεί στη νέα ανάλυση εισόδου. Αντιθέτως, στο YOLOv2 μετά την ολοκλήρωση της εκπαίδευσης με εικόνα εισόδου 224×224, το μοντέλο συνεχίζει την εκπαίδευση του εξαγωγέα χαρακτηριστικών για 10 εποχές με εικόνα εισόδου 448×448 στο ImageNet, δίνοντας χρόνο στο δίκτυο να προσαρμόσει τα φίλτρα του ώστε να λειτουργούν καλύτερα σε είσοδο υψηλότερης ανάλυσης. Αυτό το δίκτυο ταξινόμησης υψηλής ανάλυσης δίνει μια αύξηση σχεδόν 4% στο mAP.

- **Convolutional with Anchor Boxes**: Μια από τις πιο αξιοσημείωτες αλλαγές στο YOLOv2 είναι τα Anchor Boxes. Το YOLO προβλέπει τις συντεταγμένες των πλαισίων οριοθέτησης απευθείας, χρησιμοποιώντας πλήρως συνδεδεμένα στρώματα πάνω από τον συνελικτικό εξαγωγέα χαρακτηριστικών. Η ιδέα του YOLOv1 είναι η χρήση ενός κελιού πλέγματος, υπεύθυνο για την ανίχνευση ενός αντικειμένου που έχει το κέντρο μέσα σε αυτό το κελί. Έτσι, όταν δύο ή περισσότερα αντικείμενα έχουν το κέντρο τους μέσα στο ίδιο κελί πλέγματος, η πρόβλεψη μπορεί να είναι εσφαλμένη.
  Για να λύσουν αυτό το πρόβλημα, οι συγγραφείς προσπάθησαν να επιτρέψουν σε ένα κελί πλέγματος να προβλέψει περισσότερα από ένα αντικείμενα. Το anchor box είναι μια λίστα προκαθορισμένων πλαισίων που ταιριάζουν καλύτερα με τα επιθυμητά αντικείμενα. Χρησιμοποιώντας τα anchor boxes παρουσιάζεται μια μικρή μείωση στην ακρίβεια, όπου το μοντέλο να λαμβάνει 69,2 mAP με ανάκληση 88%. Χωρίς αυτά λαμβάνει 69,5 mAP με ανάκληση 81%. Παρόλο που το mAP μειώνεται, η αύξηση της ανάκλησης σημαίνει ότι το μοντέλο έχει περισσότερο περιθώριο βελτίωσης.
- **Dimension Clusters**: Με την εφαρμογή των anchor boxes παρουσιάστηκαν δυο προβλήματα με το πρώτο να είναι ότι οι διαστάσεις του πλαισίου επιλέγονται με το χέρι. Η λύση που προτείνουν είναι ότι αντί να επιλέγουμε με το χέρι τα καλύτερα προσαρμοσμένα πλαίσια, να εκτελείται ο αλγόριθμος ομαδοποίησης k-means στα πλαίσια οριοθέτησης του συνόλου εκπαίδευσης για να ομαδοποιηθούν τα πλαίσια οριοθέτησης που έχουν παρόμοια σχήματα και, στη συνέχεια, να σχεδιαστεί ο μέσος όρος IOU με το πλησιέστερο κέντρο. Επίσης, αντί να χρησιμοποιηθεί η Ευκλείδεια απόσταση, η μέτρηση της απόστασης γίνεται ως εξής:
$$d(box, centroid) = 1 - \text{IOU}(box, centroid)$$
- **Direct location prediction**: Το δεύτερο πρόβλημα που προκύπτει από την εφαρμογή των anchor boxes είναι η αστάθεια του μοντέλου, ειδικά κατά τις πρώτες επαναλήψεις της εκπαίδευσης. Το μεγαλύτερο μέρος της αστάθειας προέρχεται από την πρόβλεψη των θέσεων (b_x, b_y) του πλαισίου. Έτσι, η θέση του πλαισίου οριοθέτησης μπορεί να απέχει από το κελί του πλέγματος που είναι υπεύθυνο για την πρόβλεψη αυτού του πλαισίου οριοθέτησης.
  Κάθε κελί πλέγματος στο YOLO καθορίζεται σε μια κλίμακα 0-1, με τις συντεταγμένες του επάνω αριστερού σημείου να είναι (0,0) και του κάτω δεξιά να είναι (1,1). Το YOLOv2 χρησιμοποίει τη Σιγμοειδή συνάρτηση ενεργοποιήσεις (σ) για τον περιορισμό της τιμής του κέντρου του πλαισίου οριοθέτησης στο εύρος 0-1, το οποίο με τη σειρά του μπορεί να ορίσει τις προβλέψεις του πλαισίου οριοθέτησης γύρω από το κελί του πλέγματος.
  Το δίκτυο προβλέπει 5 οριοθετημένα πλαίσια σε κάθε κελί στον χάρτη χαρακτηριστικών εξόδου. Το δίκτυο προβλέπει επίσης, 5 συντεταγμένες για κάθε πλαίσιο οριοθέτησης, t_x, t_y, t_w, t_h και t_o. Εάν το κελί έχει μετατοπιστεί από την επάνω αριστερή γωνία της εικόνας κατά (c_x, c_y) και το πλαίσιο οριοθέτησης πριν έχει πλάτος και ύψος p_w, p_h, τότε οι προβλέψεις αντιστοιχούν σε:
$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$
- **Fine-Grained Features**: Το YOLOv2 διαιρεί τον χάρτη χαρακτηριστικών σε κελιά πλέγματος 13×13. Με την προσθήκη ενός επιπέδου διέλευσης, που συνδυάζει τα χαρακτηριστικά υψηλότερης ανάλυσης με τα χαρακτηριστικά χαμηλής ανάλυσης στοιβάζοντας γειτονικά

χαρακτηριστικά σε διαφορετικά κανάλια αντί για χωρικές τοποθεσίες, ο χάρτης χαρακτηριστικών διαστάσεων 26×26×512 μετατρέπεται σε έναν χάρτη χαρακτηριστικών 13×13×2048.

Αν και αυτό είναι αρκετό για μεγάλα αντικείμενα, μπορεί να επωφεληθεί από τα πιο λεπτομερή χαρακτηριστικά για την αναγνώριση ή τον εντοπισμό μικρότερων αντικειμένων στην εικόνα.

Multi-Scale Training: Η αρχική έκδοση του YOLO χρησιμοποιεί ανάλυση εισόδου 448×448, η οποία με τα anchor boxes στο YOLOv2 έχει αλλάξει σε 416×416. Το YOLOv1 έχει αδυναμία στην ανίχνευση αντικειμένων με διαφορετικά μεγέθη εισόδου που σημάνει ότι αν εκπαιδευτεί με μικρές εικόνες ενός συγκεκριμένου αντικειμένου θα έχει πρόβλημα στην ανίχνευση του ίδιου αντικειμένου σε εικόνα μεγαλύτερου μεγέθους. Αυτό έχει επιλυθεί σε μεγάλο βαθμό σε αυτή την έκδοση, αφού το δίκτυο εκπαιδεύεται με τυχαίες εικόνες σε διαφορετικές διαστάσεις που κυμαίνονται μεταξύ 320×320 και 608×608. Συγκεκριμένα, κάθε 10 batches το δίκτυο επιλέγει τυχαία μια νέα διάσταση για την εικόνα, μειώνοντας το δείγμα κατά 32 όπου και αντλούνται τα ακόλουθα πολλαπλάσια του {320, 352, …, 608}.

Κατά συνέπεια, το δίκτυο μαθαίνει και προβλέπει τα αντικείμενα από διάφορες διαστάσεις εισόδου με ακρίβεια. Το δίκτυο λειτουργεί πιο γρήγορα σε μικρότερα μεγέθη, έτσι το YOLOv2 προσφέρει μια εύκολη αντιστάθμιση μεταξύ ταχύτητας και ακρίβειας.

- **Darknet-19**: Το YOLOv2 βασίζεται στην αναβαθμισμένη αρχιτεκτονική του Darknet (Darknet-19) που χρησιμοποιεί 19 συνελικτικά στρώματα, 5 στρώματα υποδειγματοληψίας μέγιστης τιμής και ένα επίπεδο Softmax.

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Σχήμα 3.25: Αρχιτεκτονική Darknet-19.

YOLOv3

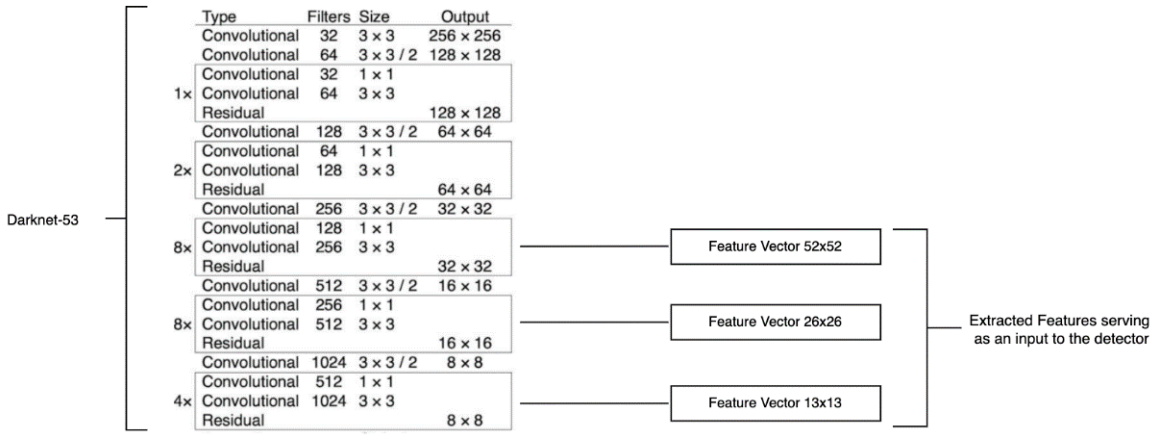
Το YOLOv3 απαρτίζει την τελευταία έκδοση του αλγορίθμου που αναπτύχθηκε από τους αρχικούς δημιουργούς Redmon και Farhadi [38]. Η αρχιτεκτονική αυτής της έκδοσης βασίζεται στα ResBlock του ResNet και το νέο Darknet-53 που αποτελείται από 53 συνελικτικά στρώματα. Το δίκτυο χρησιμοποιεί διαδοχικά στρώματα συνέλιξης 1×1 και στρώματα συνέλιξης 3×3 μέσα σε κάθε ResBlock.

Χάρη στα υπολειπόμενα μπλοκ του ResNet, τα επίπεδα επικάλυψης δεν υποβαθμίζουν την απόδοση του δικτύου. Τα 53 στρώματα του Darknet στοιβάζονται περαιτέρω με 53 ακόμη στρώματα για την κεφαλή ανίχνευσης, καθιστώντας το YOLOv3 συνολικά μια πλήρως συνελικτική υποκείμενη αρχιτεκτονική με 106 στρώματα.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Σχήμα 3.26: Αρχιτεκτονική του Darknet-53 με 5 ResBlock που περιέχουν διαδοχικά στρώματα συνέλιξης 1×1 και στρώματα συνέλιξης 3×3.

Στις 2 προηγούμενες εκδόσεις του YOLO, μετά την εκπαίδευση του εξαγωγέα χαρακτηριστικών με την αρχιτεκτονική Darknet, η εικόνα εισόδου προωθούταν σε μερικά ακόμα επίπεδα μέχρι να γίνουν οι προβλέψεις στα τελευταία επίπεδα του ανιχνευτή αντικειμένων. Ωστόσο, το YOLOv3 προσάρτησε το δίκτυο των επιπέδων πρόβλεψης αντί να το στοιβάζει στα τελευταία επίπεδα όπως πριν. Το πιο αξιοσημείωτο χαρακτηριστικό του YOLOv3 είναι ότι κάνει προβλέψεις σε 3 διαφορετικές κλίμακες: στο 82ο, στο 94ο και το 106ο στρώμα.



Σχήμα 3.27: Ο ανιχνευτής πολλαπλής κλίμακας του YOLOv3.

Στη συνέχεια θα δούμε μερικές ακόμα βελτιώσεις του YOLOv3:

- **Προβλέψεις πλαισίου οριοθέτησης**: Σε αντίθεση με το YOLOv1 όπου τα πλαίσια οριοθέτησης προβλέπονται από το ίδιο κελί πλέγματος και μοιράζονται ένα σύνολο πιθανοτήτων πρόβλεψης κλάσεων *C*, το YOLOv3 προβλέπει μια βαθμολογία για κάθε πλαίσιο οριοθέτησης χρησιμοποιώντας λογιστική παλινδρόμηση.
- **Προβλέψεις κλάσης**: Το YOLOv3 χρησιμοποιεί λογιστικούς ταξινομητές για κάθε κλάση αντί για Softmax που είχε το προηγούμενο μοντέλο. Με αυτό τον τρόπο γίνεται να έχουμε ταξινόμηση πολλαπλών ετικετών.
  Για παράδειγμα στο σύνολο δεδομένων Open Images υπάρχουν πολλές επικαλυπτόμενες ετικέτες όπως το Άτομο (Person) και η Γυναικάς (Woman). Η χρήση Softmax επιβάλλει την υπόθεση ότι κάθε πλαίσιο έχει ακριβώς μία κλάση, κάτι που συχνά δεν συμβαίνει.
  Οπότε, μια προσέγγιση πολλαπλών ετικετών μοντελοποιεί καλύτερα τα δεδομένα.
- **Προβλέψεις κλίμακας**: Το YOLOv3 κάνει προβλέψεις παρόμοιες με τα Δίκτυα Πυραμίδας Δυνατοτήτων (Feature Pyramid Networks - FPN) [43], όπου γίνονται 3 προβλέψεις για κάθε θέση της εικόνας εισόδου και εξάγονται χαρακτηριστικά από κάθε πρόβλεψη.
  Στο βασικό σύστημα εξαγωγής χαρακτηριστικών προστίθενται διάφορα επίπεδα συνέλιξης, όπου το τελευταίο από αυτά προβλέπει έναν τρισδιάστατο τανυστή με το πλαίσιο οριοθέτησης, την αντικειμενικότητα και την τιμή της πρόβλεψης κλάσης. Η προσθήκη αυτών των στρωμάτων συνέλιξης βελτιώνει την έξοδο.
  Τέλος, κάνοντας upsampling από τα προηγούμενα στρώματα επιτρέπεται η απόκτηση πλήρους σημασιολογικής πληροφορίας και πιο λεπτομερούς πληροφορίας από τον προηγούμενο χάρτη χαρακτηριστικών.

YOLOv4

Το Φεβρουάριο του 2020, έπειτα από 5 χρόνια ανάπτυξης των YOLO και Darknet, ο Joseph Redmon ανακοίνωσε την αποχώρηση του από τον τομέα της Υπολογιστικής Όρασης. Τον Απρίλιο της ιδίας χρονιάς, ο ερευνητής Alexey Bochkovskiy που ανέπτυξε το πλαίσιο Darknet και τις προηγούμενες εκδόσεις του YOLO σε γλώσσα C για τα λειτουργικά συστήματα Windows [22], δημοσιεύεσαι τη νέα έκδοση του YOLO (YOLOv4) σε συνεργασία με τους Chien-Yao Wang και Hong-Yuan Mark Liao.

Το YOLO ως ανιχνευτής ενός σταδίου βασίζεται στα εξής κύρια στοιχεία:

- **Backbone**: Πρόκειται για ένα CNN που εξάγει χαρακτηριστικά. Χρησιμοποιούνται μοντέλα, όπως το ResNet, που προ-εκπαιδεύονται σε σύνολα δεδομένων ταξινόμησης, όπως το ImageNet, και στη συνέχεια προσαρμόζονται στο σύνολο δεδομένων ανίχνευσης.
- **Neck**: Επιπλέον στρώματα που παρεμβάλλονται μεταξύ των Backbone και Head. Χρησιμοποιούνται για την εξαγωγή διαφορετικών χαρτών χαρακτηριστικών κατά τα διάφορα στάδια του Backbone. Το Neck μπορεί να είναι ένα FPN, όπως για παράδειγμα στο YOLOv3.
- **Head**: Πρόκειται για ένα δίκτυο που είναι υπεύθυνο για την εκτέλεση της ανίχνευσης (ταξινόμηση και παλινδρόμηση) των πλαισίων οριοθέτησης. Οι ανιχνευτές αντικειμένων που χρησιμοποιούν anchor boxes, όπως το YOLO, εφαρμόζουν το δίκτυο αυτό σε κάθε anchor box.

Το κοινό σημείο όλων των αρχιτεκτονικών ανίχνευσης αντικειμένων είναι ότι τα χαρακτηριστικά της εικόνας εισόδου συμπιέζονται μέσω του εξαγωγέα χαρακτηριστικών (Backbone) και στη συνέχεια προωθούνται στον ανιχνευτή αντικειμένων (Neck και Head). Το Neck λειτουργεί ως ένας αθροιστής χαρακτηριστικών που ανακατεύει και συνδυάζει τα χαρακτηριστικά που σχηματίζονται στο Backbone,

με σκοπό την προετοιμασία τους για το βήμα ανίχνευσης στο Head. Το Head είναι υπεύθυνο για την πραγματοποίηση της ανίχνευσης, του εντοπισμού και της ταξινόμησης για κάθε πλαίσιο οριοθέτησης.

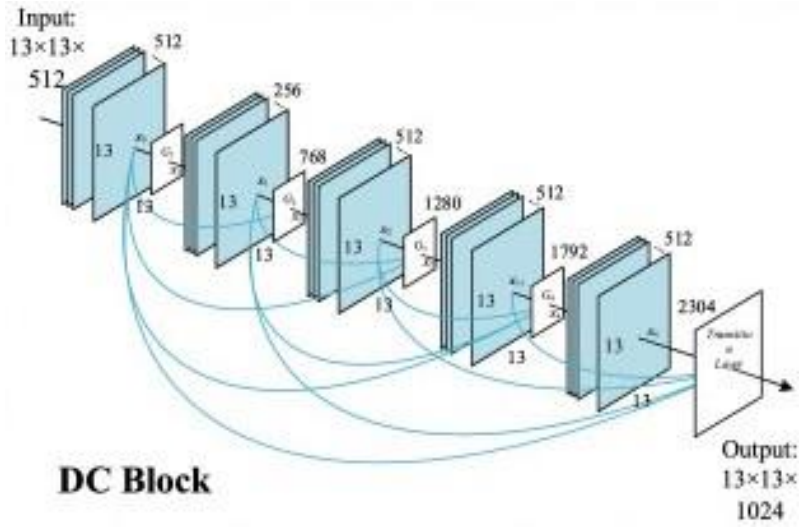
Οι συγγράφεις του YOLOv4 πραγματοποίησαν μια σειρά πειραμάτων, καταλήγοντας στις παρακάτω μεθόδους για κάθε τμήμα της αρχιτεκτονικής της νέας έκδοσης του YOLO, ώστε να επιτύχουν βελτίωση της ακρίβειας του μοντέλου:

- **Backbone**: Οι συγγράφεις κατέληξαν σε 3 επιλογές για τον εξαγωγέα χαρακτηριστικών του μοντέλου: CSPResNext50, CSPDarknet53 και EfficientNet-B3. Κάποιες από τις επιλογές είναι περισσότερο κατάλληλες για ταξινόμηση παρά για ανίχνευση. Για παράδειγμα, το CSPDarknet53 έδειξε να είναι καλύτερο από το CSPResNext50 όσον αφορά την ανίχνευση αντικειμένων και το CSPResNext50 καλύτερο από το CSPDarknet53 για την ταξινόμηση εικόνων.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

Σχήμα 3.28: Σύγκριση των 3 δικτυών Backbone.

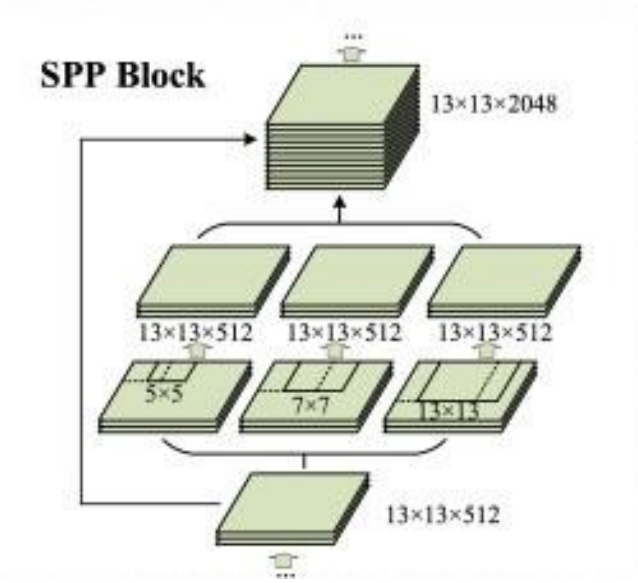
Τα CSPResNext50 και CSPDarknet53 προέρχονται από την αρχιτεκτονική DenseNet, η οποία χρησιμοποιεί την προηγούμενη είσοδο και τη συνθέτει με την τρέχουσα είσοδο πριν μεταβεί στο πυκνό (dense) στρώμα. Το Cross Stage Partial (CSP) [45] βασίζεται στην ίδια αρχή με το DenseNet, με τη διαφορά ότι αντί να χρησιμοποιείται ο χάρτης χαρακτηριστικών εισόδου πλήρους μεγέθους στο στρώμα βάσης, η είσοδος διαχωρίζεται σε 2 τμήματα. Ένα τμήμα που θα προωθείται μέσω του dense block και ένα άλλο που θα στέλνεται κατευθείαν στο επόμενο στάδιο χωρίς επεξεργασία.



Σχήμα 3.29: Dense block του YOLOv4.

Συνδυάζοντας αυτές τις ιδέες με την αρχιτεκτονική Darknet-53 του YOLOv3, η καλύτερη επιλογή ήταν αυτή του CSPDarknet53, καθώς παρατηρήθηκε ότι το συγκεκριμένο μοντέλο επιδεικνύει μεγαλύτερη ικανότητα αύξησης στην ακρίβεια του ανιχνευτή.

- **Neck**: Οι χάρτες χαρακτηριστικών εξόδου του CSPDarknet53 στέλνονται πρώτα σε ένα πρόσθετο Spatial Pyramid Pooling (SPP) block για να αυξηθεί το υποδεκτικό πεδίο και να διαχωριστούν τα πιο σημαντικά χαρακτηριστικά.
  Πολλά μοντέλα CNN περιέχουν πλήρως συνδεδεμένα επίπεδα, τα οποία δέχονται μόνο εικόνες εισόδου συγκεκριμένων διαστάσεων. Δεδομένου ότι τέτοια στρώματα αφαιρέθηκαν από το YOLOv2 και έπειτα, κάνοντας τον αλγόριθμο YOLO ένα πλήρως συνελικτικό δίκτυο (FCN), επιτρέπεται πλέον η είσοδος εικόνων διαφορετικών διαστάσεων. Το SPP [46] δημιουργήθηκε για αυτόν τον σκοπό, αφού μπορεί να παράγει έξοδο σταθερού μεγέθους ανεξάρτητα από το μέγεθος της εισόδου. Πέρα από αυτό, το SPP βοηθά στην εξαγωγή σημαντικών χαρακτηριστικών με τη συγκέντρωση εκδόσεων πολλαπλών κλίμακων του εαυτού του. Έτσι, το SPP block εξάγει ταυτόχρονα διαφορετικούς τύπους σημαντικών χαρακτηριστικών.
  Το SPP block ακόμη, έχει τροποποιηθεί έτσι ώστε να διατηρεί τη χωρική διάσταση της εξόδου, αφού το YOLO κάνει ταυτόχρονα τις προβλέψεις και τους εντοπισμούς των πλαισίων οριοθέτησης. Για τη μείωση του αριθμού των χαρτών χαρακτηριστικών εισόδου που αποστέλλονται στο SPP block χρησιμοποιείται μια συνέλιξη 1×1 μεταξύ του Backbone και του block. Στη συνέχεια, οι χάρτες χαρακτηριστικών εισόδου διπλασιάζονται και συγκεντρώνονται σε διαφορετικές κλίμακες, με μόνη διαφορά στο συμπλήρωμα (padding) που χρησιμοποιείται για να διατηρηθεί σταθερό το μέγεθος των χαρτών χαρακτηριστικών της εξόδου.



Σχήμα 3.30: SPP block του YOLOv4.

Επίσης, προκειμένου να διατηρηθούν τα πιο λεπτομερή χαρακτηριστικά (fine-grained), ο Redmon είχε εφαρμόσει την αρχιτεκτονική FPN στο YOLOv3. Στο YOLOv4 όμως, γίνεται χρήση μιας προηγμένης έκδοσης της FPN, η Path Aggregation Network (PAN) [47], την οποία τροποποίησαν οι συγγράφεις αλλάζοντας την πράξη της πρόσθεσης κατά στοιχείο με την πράξη της συνένωσης για τους χάρτες χαρακτηριστικών.
Στην FPN, οι προβλέψεις γίνονταν χωριστά και ανεξάρτητα σε διαφορετικά επίπεδα κλίμακας. Αυτό μπορεί να παράγει διπλές προβλέψεις και να μην αξιοποιεί πληροφορίες από άλλους χάρτες χαρακτηριστικών. Η PAN συγχώνευσε όλους τους χάρτες χαρακτηριστικών εξόδου της πυραμίδας αύξησης από κάτω προς τα πάνω χρησιμοποιώντας την ευθυγράμμιση ROI (Region of Interest) και τα πλήρως συνδεδεμένα στρώματα με τη λειτουργία element-wise max. Έτσι,

όλες οι μεταβλητότητες των χαρτών χαρακτηριστικών συγκεντρώνονται και χρησιμοποιούνται για προβλέψεις.

- **Head**: Το YOLOv4 χρησιμοποιεί το ίδιο Head με το YOLOv3 για την ανίχνευση. Η λειτουργία του Head είναι να εκτελεί πυκνές προβλέψεις, όπου η τελική πρόβλεψη αποτελείται από ένα διάνυσμα με τις συντεταγμένες του προβλεπόμενου πλαισίου οριοθέτησης.

Επιπλέον, οι συγγράφεις του YOLOv4 διακρίνουν δυο κατηγορίες μεθόδων που χρησιμοποιούνται για τη βελτίωση της ακρίβειας του ανιχνευτή αντικειμένων. Αυτές οι κατηγορίες αναφέρονται με τους παρακάτω δυο όρους:

- **Bag of Freebies (BoF)**: Μέθοδοι που μπορούν να κάνουν τον ανιχνευτή αντικειμένων να έχει καλύτερη ακρίβεια χωρίς να αυξάνεται το κόστος για την εξαγωγή συμπερασμάτων. Ουσιαστικά, οι βελτιώσεις αυτές μπορούν να βοηθήσουν στην αύξηση της απόδοσης και της ακρίβειας του μοντέλου χωρίς κανένα κόστος σε υλικό. Στο YOLOv4 εφαρμόστηκαν οι ακόλουθες μέθοδοι βελτίωσης:
  - **Backbone**: CutMix [48] και Mosaic (Glenn Jocher) επαυξήσεις δεδομένων, κανονικοποίηση DropBlock [49] και εξομάλυνση ετικέτας κλάσεων.
  - **Neck και Head**: Complete Intersection over Union (CIoU-loss), Cross-mini–Batch Normalization (CmBN), κανονικοποίηση DropBlock, Mosaic επαύξηση δεδομένων, Self-Adversarial Training (SAT), εξάλειψη ευαισθησίας πλέγματος, χρήση πολλαπλών anchors για μια μόνο βασική αλήθεια, χρονοπρογραμματιστής ανόπτησης συνημίτονου, βέλτιστες υπερ-παράμετροι και τυχαία σχήματα εκπαίδευσης.
- **Bag of Specials (BoS)**: Προηγμένες μέθοδοι βελτιστοποίησης που απαιτούν από την αρχιτεκτονική ένα μικρό κόστος, ώστε να επιτευχθεί βελτίωση της ακρίβειας απόδοσης ανίχνευσης. Στο YOLOv4 εφαρμόστηκαν οι ακόλουθες μέθοδοι βελτίωσης:
  - **Backbone**: Ενεργοποίηση Mish [50], Cross-stage partial connections (CSP) [45], Multi-input weighted residual connections (MiWRC) [51].
  - **Neck και Head**: Ενεργοποίηση Mish [50], Spatial Pyramid Pooling block (SPP-block) [46], Spatial Attention Module block (SAM-block) [52], Path Aggregation Network block (PAN-block) [47], Distance Intersection over Union Non-Maximum Suppression (DIoU-NMS) [53].

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	38 (M)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	31 (M)	**43.0%**	**64.9%**	**46.5%**	**24.3%**	**46.1%**	**55.2%**
YOLOv4	CSPDarknet-53	608	23 (M)	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%

Σχήμα 3.31: Ταχύτητα και ακρίβεια του YOLOv4 με εισόδους διαφορετικών μεγεθών στο σύνολο δεδομένων MS COCO. Οι επισημασμένες γραμμές αποτελούν ανιχνευτές πραγματικού χρόνου (καρέ ανά δευτερόλεπτο ≥ 30).

YOLOv5

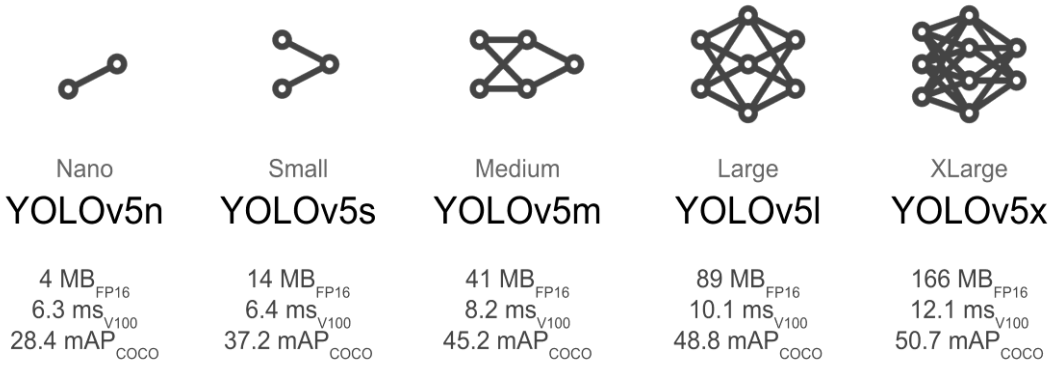
Σε διάστημα μικρότερο των 2 μηνών από την κυκλοφορία του YOLOv4, ο ερευνητής και ιδρυτής της ομάδας Ultralytics LLC Glenn Jocher, δημοσίευσε την νέα έκδοση του YOLO [40]. Η μεγαλύτερη συνεισφορά του Glenn Jocher, και ουσιαστικά του YOLOv5, είναι η μετατροπή της αρχιτεκτονικής Darknet στο πλαίσιο PyTorch σε γλώσσα Python. Το YOLOv5 ξεκίνησε ως μια επέκταση του repository YOLOv3 PyTorch.

Η αρχιτεκτονική του YOLOv5 είναι παρόμοια με αυτή του YOLOv4, καθώς και οι δυο ερευνητές εφάρμοσαν τις πιο σύγχρονες καινοτομίες στον τομέα της αναγνώρισης αντικειμένων. Ο Glenn Jocher παρόλο που δεν έχει δημοσιεύσει ακόμα ένα επιστημονικό άρθρο σχετικά με τη δομή του YOLOv5, από τον κώδικα των αρχείων τύπου .yaml διαπιστώνεται η δομή του μοντέλου να είναι ως εξής:

- **Backbone**: Δίκτυο Cross-stage partial (CSP).
- **Neck**: Spatial Pyramid Pooling block και Path Aggregation Network.
- **Head**: Ίδιο με το YOLOv3 και χρήση συνάρτησης απώλειας Generalized Intersection over Union (GIoU-loss) [54].

Εκτός από αυτά, το YOLOv5 παρέχει τις παρακάτω επιλογές για την εκπαίδευση:

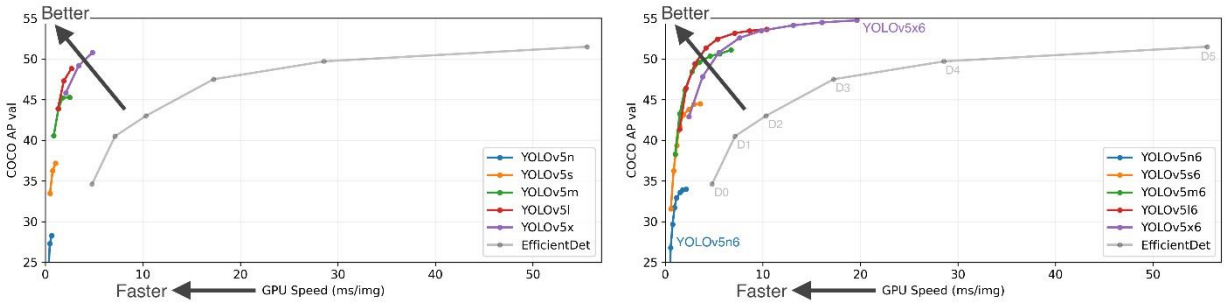
- **Συνάρτηση ενεργοποίησης**: Η συνάρτηση ενεργοποίησης Leaky ReLU χρησιμοποιείται στα ενδιάμεσα (κρυφά) επίπεδα, ενώ η σιγμοειδής (Sigmoid) συνάρτηση στο τελικό επίπεδο ανίχνευσης.
- **Συνάρτηση βελτιστοποίησης**: Η προεπιλεγμένη συνάρτηση βελτιστοποίησης για την εκπαίδευση είναι η Stochastic Gradient Descent (SGD), ωστόσο δίνεται και η δυνατότητα επιλογής της Adaptive Moment Estimation (Adam) [55].
- **Συνάρτηση απώλειας**: Η συνάρτηση Binary Cross-Entropy με απώλεια Logits (BCEWithLogitsLoss) [56] από το PyTorch χρησιμοποιείται για τον υπολογισμό των απωλειών της πιθανότητας κλάσης και της βαθμολογίας αντικειμένου.



Σχήμα 3.32: Οι 5 τύποι προ-εκπαιδευμένων μοντέλων του YOLOv5.

Παράλληλα, το YOLOv5 διαθέτει πολλαπλά προ-εκπαιδευμένα μοντέλα. Η διαφορά μεταξύ τους είναι ο συμβιβασμός ανάμεσα στο μέγεθος του μοντέλου και στον χρόνο εξαγωγής συμπερασμάτων. Τα μεγαλύτερα μοντέλα όπως τα YOLOv5x και YOLOv5x6 παράγουν καλύτερα αποτελέσματα σε όλες σχεδόν τις περιπτώσεις, αλλά έχουν περισσότερες παραμέτρους, απαιτούν περισσότερη μνήμη CUDA για την εκπαίδευση και είναι πιο αργά στην εκτέλεση. Τα πιο μικρά μοντέλα όπως τα YOLOv5n/s και YOLOv5n6/s6 συστήνονται για εφαρμογές σε κινητές συσκευές.

Οι μετρικές αξιολόγησης που παρουσιάζονται παρακάτω είναι προκαταρτικές, μέχρις ότου να ολοκληρωθεί και να δημοσιευθεί η επίσημη ερευνητική εργασία του YOLOv5. Οι μετρικές αυτές βασίζονται στις επιδόσεις στο σύνολο δεδομένων COCO, το οποίο περιέχει ένα ευρύ φάσμα εικόνων με 80 κλάσεις αντικειμένων.



Σχήμα 3.33: Μετρικές αξιολόγησης των προ-εκπαιδευμένων μοντέλων του YOLOv5 στο σύνολο δεδομένων COCO. Οι μετρήσεις έγιναν στην GPU V100.

Πίνακας 3.4: Πίνακας με τις μετρήσεις όλων τον προ-εκπαιδευμένων μοντέλων του YOLOv5. Όλα τα μοντέλα εκπαιδευτήκαν για 300 εποχές με τις προκαθορισμένες ρυθμίσεις και υπερπαραμέτρους. Οι τιμές mAP^{val} αντιστοιχούν για το σύνολο δεδομένων COCO val2017. Η επαύξηση χρόνου δοκιμής Test Time Augmentation (TTA) περιλαμβάνει επαυξήσεις αντανάκλασης και κλίμακας.

Model	**size (pixels)**	**mAP^{val} 0.5:0.95**	**mAP^{val} 0.5**	**Speed CPU b1 (ms)**	**Speed V100 b1 (ms)**	**Speed V100 b32 (ms)**	**params (M)**	**FLOPs @640**
YOLOv5n	640	28.4	46.0	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.2	56.0	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.2	63.9	224	8.2	1.7	21.2	49.0
YOLOv5l	640	48.8	67.2	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	34.0	50.7	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.5	63.0	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.0	69.0	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.6	71.6	1784	15.8	10.5	76.7	111.4
YOLOv5x6 + TTA	1280 1536	54.7 **55.4**	**72.4** 72.3	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

3.3 PyTorch

Το PyTorch είναι μια βιβλιοθήκη Βαθιάς Μηχανικής Μάθησης ανοιχτού κώδικα για την Python [57]. Αναπτύχθηκε από την ερευνητική ομάδα τεχνητής νοημοσύνης του Facebook (Facebook's AI Research lab - FAIR) το 2016. Είναι μια από τις ευρέως χρησιμοποιούμενες βιβλιοθήκες μηχανικής μάθησης με χρήση GPU και CPU.

Το PyTorch είναι κατασκευασμένο με βάση τη βιβλιοθήκη Torch, η οποία υποστηρίζει τους υπολογισμούς τανυστών σε GPU. Οι τανυστές είναι μια εξειδικευμένη δομή δεδομένων που μοιάζει πολύ με τους πίνακες και τις μήτρες. Οι τανυστές του PyTorch είναι παρόμοιοι με τους πίνακες της NumPy, με τη διαφορά ότι μπορούν να εκτελούνται σε GPU ή άλλο εξειδικευμένο υλικό για την

επιτάχυνση των υπολογισμών. Στο PyTorch, χρησιμοποιούμε τανυστές για να κωδικοποιήσουμε τις εισόδους και τις εξόδους ενός μοντέλου, καθώς και τις παραμέτρους του.

Το PyTorch περιλαμβάνει τρία σημαντικά modules:

1. **Automatic Differentiation (Autograd)**: Κατά την εκπαίδευση των νευρωνικών δικτυών, ο πιο συχνά χρησιμοποιούμενος αλγόριθμος είναι ο αλγόριθμος οπισθοδιάδοσης. Σ' αυτόν τον αλγόριθμο οι παράμετροι - βάρη του μοντέλου - προσαρμόζονται σύμφωνα με την κλίση της συνάρτησης απώλειας ως προς τη δεδομένη παράμετρο.
   Για τον υπολογισμό αυτών των κλίσεων, διατίθεται η Αυτόματη Διαφοροποίηση (*torch.autograd*) που υποστηρίζει τον αυτόματο υπολογισμό της κλίσης για οποιοδήποτε υπολογιστικό γράφημα. Αυτή η μέθοδος είναι ιδιαίτερα ισχυρή κατά τη δημιουργία νευρωνικών δικτύων για την εξοικονόμηση χρόνου σε μία εποχή με τον υπολογισμό της διαφοροποίησης των παραμέτρων στο εμπρόσθιο πέρασμα.
2. **Optimization (Optim)**: Η βελτιστοποίηση είναι η διαδικασία προσαρμογής των παραμέτρων του μοντέλου για τη μείωση του σφάλματος σε κάθε βήμα της εκπαίδευσης. Οι αλγόριθμοι βελτιστοποίησης καθορίζουν τον τρόπο με τον οποίο εκτελείται αυτή η διαδικασία.
   Το PyTorch παρέχει το *torch.optim* για την υλοποίηση των διάφορων αλγορίθμων βελτιστοποίησης.
3. **nn**: Τα νευρωνικά δίκτυα στο PyTorch μπορούν να κατασκευαστούν με το πακέτο *torch.nn*. Το nn module εξαρτάται από το autograd για τον ορισμό των μοντέλων και τη διαφοροποίηση τους. Ένα nn module περιέχει διάφορα επίπεδα και μια μέθοδο προς τα εμπρός διάδοσης που επιστρέφει την έξοδο.

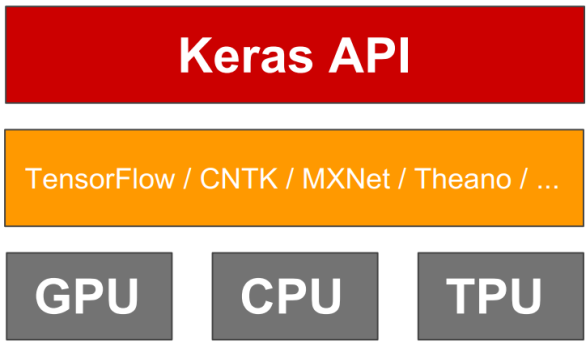
3.4 Keras

Το Keras είναι επίσης μια βιβλιοθήκη Βαθιάς Μηχανικής Μάθησης ανοιχτού κώδικα για την Python [58]. Δημιουργήθηκε από τον μηχανικό της Google François Chollet, με σκοπό τον εύκολο πειραματισμό σε Βαθιά Νευρωνικά Δίκτυα και την φιλική προς τον χρήστη λειτουργία του. Σκοπός του είναι εύκολα να μπορεί να χρησιμοποιηθεί από προγραμματιστές για να εκπαιδεύσουν μοντέλα μηχανικής μάθησης, με τη χρήση ενός API.

Το Keras υποστηρίζει τα παρακάτω πλαίσια:

- Tensorflow,
- Theano,
- PlaidML,
- MXNet,
- Microsoft Cognitive Toolkit

Ο τρόπος που χρησιμοποιεί το Tensorflow το API του Keras, φαίνεται στο παρακάτω σχήμα (Σχήμα 3.34).

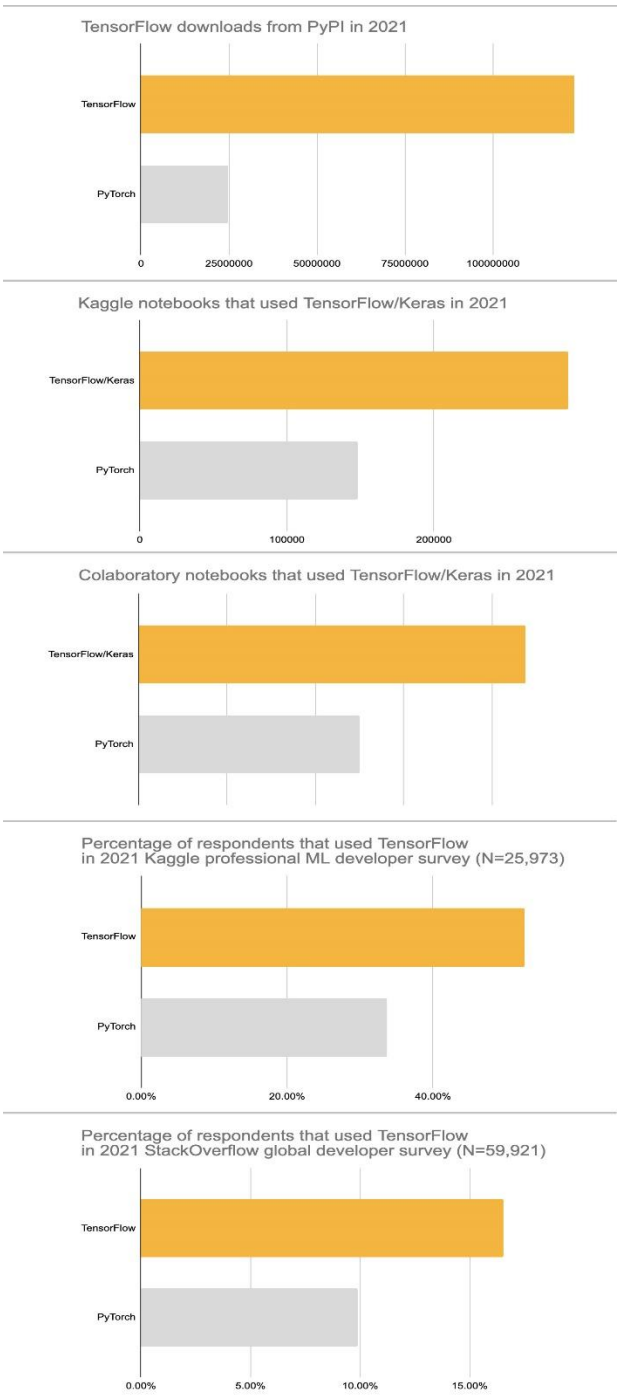


Σχήμα 3.34: Δομή παρασκήνιου του Keras.

Βασικά χαρακτηριστικά του Keras είναι:

- Επιτρέπει την εκτέλεση του ίδιου κώδικα σε CPU, GPU ή TPU χωρίς διαφορά,
- Το φιλικό προς το χρήστη API του, καθιστά εύκολη τη γρήγορη προτυποποίηση μοντέλων βαθιάς μάθησης,
- Μπορεί να γίνει εξαγωγή των μοντέλων Keras σε TFLite ώστε να εκτελούνται απευθείας σε Android και iOS συσκευές,
- Υποστηρίζει σχεδόν όλα τα μοντέλα νευρωνικών δικτυών,
- Περιέχει μια μεγάλη ποικιλία στρωμάτων (συνελικτικά, συγκεντρωτικά, κανονικοποίησης και αφαίρεσης), συναρτήσεις ενεργοποίησης και μηχανισμούς μάθησης,
- Η αρθρωτή δομή και η ευελιξία του το καθιστούν κατάλληλο για καινοτόμες έρευνες.

Τέλος, το 2021 το Keras κατέγραψε πάνω από ένα εκατομμύριο μεμονωμένους χρήστες λαμβάνοντας ευρεία αποδοχή από τη βιομηχανία και την ερευνητική κοινότητα. Χρησιμοποιείται από τις πλατφόρμες όπως το Netflix, Uber, Yelp, Instacart και είναι ιδιαίτερα δημοφιλής μεταξύ των νεοσύστατων επιχειρήσεων που τοποθετούν τη βαθιά μάθηση στον πυρήνα των προϊόντων τους. Το Keras μαζί με το TensorFlow 2 είναι επίσης αγαπημένα μεταξύ των ερευνητών, καταλαμβάνοντας την 1η θέση όσον αφορά τις αναφορές σε επιστημονικές εργασίες, ενώ το Keras έχει υιοθετηθεί από ερευνητές σε μεγάλους επιστημονικούς οργανισμούς, όπως το CERN και η NASA.



Σχήμα 3.35: Στατιστικά στοιχεία του Keras σε σύγκριση με το PyTorch από την επίσημη σελίδα του.

3.5 TensorFlow

3.5.1 Εισαγωγή

Το TensorFlow είναι μία πλατφόρμα ανοιχτού κώδικα, δημιουργημένη από την εσωτερική ομάδα της Google, Google Brain, η οποία κυκλοφόρησε το 2017 [59]. Επιτρέπει εύκολα το προγραμματισμό και την δημιουργία αλγορίθμων μηχανικής μάθησης και τεχνητής νοημοσύνης με τη χρήση ενός API. Δίνει τη δυνατότητα στους προγραμματιστές να συνδεθούν από οποιαδήποτε πλατφόρμα χρησιμοποιούν και υποστηρίζει διάφορες γλώσσες, μερικές εκ των οποίων είναι η Python, C και C++. Πολλές από τις χρήσεις του TensorFlow είναι η ανάπτυξη ερευνητικών εφαρμογών, μοντέλων βαθιάς μάθησης,

αναγνώρισης φυσικής γλώσσας, όραση υπολογιστών και πολλά άλλα. Έχει υψηλή ευελιξία λόγο της δομής που χρησιμοποιεί, η οποία μοιάζει με σχεδιάγραμμα ροής, το οποίο έπειτα μπορεί να μεταφραστεί σε διάφορες μορφές και να χρησιμοποιηθεί από διάφορες συσκευές και πλατφόρμες.

3.5.2 TensorFlow 1.0

Το TensorFlow 1.0 προσφέρει ένα API γραμμένο σε Python, που μπορεί να αναπτυχθεί σε πολλά περιβάλλοντα και συσκευές όπως CPU, GPU Android και άλλα λειτουργικά συστήματα κινητών συσκευών όπως iOS ή σε javascript (tensorflow 1.x+) χρησιμοποιώντας το πρόγραμμα περιήγησης. Το προϊόν ήταν από την αρχή του έτοιμο για χρήση και αυτό είναι κάτι που το βοήθησε στο να πετύχει και να γίνει δημοφιλέστερο από την αντίστοιχη έκδοση Pytorch. Τόσο το TensorFlow όσο και το PyTorch εστιάζουν στη βαθιά μάθηση και είναι βελτιστοποιημένα για βαθιά μάθηση.

3.5.3 TPU

Για την επίλυση προβλημάτων βαθιάς μάθησης, πολλές φορές χρειάζονται αρκετά περίπλοκοι και μεγάλοι υπολογισμοί. Λόγω της φύσης της, οι υπολογισμοί αυτοί χρειάζονται και την ανάλογη υπολογιστική δύναμη. Πέρα από τη χρήση μίας CPU, η οποία μπορεί να μην είναι όσο αποδοτική όσο θα θέλαμε, μια λύση ήταν η χρήση των μεμονωμένων GPU, οι οποίες ουσιαστικά ήταν μονάδες επεξεργασίας με μεγαλύτερες δυνατότητες από τις CPU και μπορούσαν να χρησιμοποιηθούν με μεγαλύτερη αποκλειστικότητα από τους αλγορίθμους βαθιάς μάθησης. [60] Ωστόσο, για την καλύτερη διαχείριση και πιο βελτιστοποιημένη χρήση, η Google δημιούργησε αποκλειστικά διαμορφωμένες επεξεργαστικές μονάδες, τις TPU (Tensor Process Unit) οι οποίες είναι εξειδικευμένες στο να λύνουν γρήγορα και ευέλικτα πράξεις πολλαπλασιασμού μεταξύ πινάκων, κάνοντας τα πιο σύνθετα μοντέλα μηχανικής μάθησης να εκπαιδεύονται ταχύτερα, μέσα σε λίγους μήνες ή και βδομάδες.

3.5.4 CUDA

Το CUDA είναι η πλατφόρμα της NVIDIA, η οποία χρησιμοποιείται για την υλοποίηση υπολογιστικών πράξεων με ένα διαφορετικό μοντέλο προγραμματισμού, το οποίο επιτυγχάνει καλύτερα αποτελέσματα. Για να γίνει αυτό, χρησιμοποιείται η επεξεργαστική ισχύ μιας κάρτας γραφικών (GPU) και η μέθοδος του παράλληλου προγραμματισμού , έτσι ώστε οι υπολογισμοί να εκτελούνται με μεγαλύτερες ταχύτητες και με καλύτερη εκμετάλλευση [61]. Πριν το CUDA, οι εφαρμογές μπορούσαν να τρέξουν κατά κύριο λόγο στην κεντρική μονάδα επεξεργασίας (CPU) ενός υπολογιστή, η οποία ήταν βελτιστοποιημένη έτσι ώστε να τρέχει αρκετά αποτελεσματικά σε μονοπύρηνα νήματα. Ωστόσο, χρησιμοποιώντας μια GPU, μπορούμε να εκμεταλλευτούμε καλύτερα τον πολυπύρηνο σχεδιασμό της και να τρέξουμε παράλληλα χιλιάδες υπολογισμούς. Με τη χρήση των βιβλιοθηκών της CUDA οι προγραμματιστές μπορούν να συνεχίσουν να γράφουν σε γλώσσες όπως C, C++, Python, MATLAB και πολλές ακόμη, ενώ ταυτόχρονα με να εκμεταλλεύονται τις νέες ιδιότητες της GPU. [62] Στην μηχανική μάθηση, η οποία απαιτεί πολλούς και πολλές φορές δύσκολους ή χρονοβόρους υπολογισμούς, ένα μεμονωμένο σύστημα το οποίο θα εξειδικευόταν στο να λύνει πολλά συστήματα και μαθηματικούς πίνακες ή συναρτήσεις, μπορεί να εξοικονομήσει πολύ χρόνο και πόρους. Αυτό έρχεται να εκμεταλλευτεί και το TensorFlow, βελτιστοποιώντας συνεχώς τους αλγορίθμους του για να μπορέσει να αξιοποιήσει όσο το δυνατόν περισσότερο το CUDA και τις δυνατότητες του.[59]

3.5.5 SYCL

Το SYCL, είναι ένα αφαιρετικό επίπεδο γενικού προγραμματισμού των εταιρειών Khronos , το οποίο χρησιμοποιεί πρότυπα και γενικές λειτουργίες «lambda», έτσι ώστε επιτρέπει τους προγραμματιστές να διατηρούν ένα καθαρό, υψηλού επιπέδου, κώδικα μεταξύ διάφορων πλατφορμών με βελτιστοποιημένη

επιτάχυνση και διάφορα φάσματα όπως το OpenCL και API. συνδυάζεται με τις βιβλιοθήκες των γλωσσών C και C++ ή κάποια frameworks, όπως το OpenCV και το OpenMP[63].

3.5.6 Edge Computing

Το Edge Computing, είναι ένα κατανεμημένο σύστημα, το οποίο φέρνει την επεξεργασία δεδομένων και την αποθήκευση τους, όσο πιο κοντά γίνεται στην πηγή αυτών των δεδομένων. Δηλαδή, αυτή η μορφή αρχιτεκτονικής, δίνει λύση στον τρόπο με τον οποίο τα δεδομένα επεξεργάζονται, διανύοντας την μικρότερη δυνατή απόσταση και ταυτόχρονα παρέχοντας γρήγορα αποτελέσματα. Έτσι μειώνεται η καθυστέρηση στην επικοινωνία, το εύρος ζώνης και γίνεται καλύτερη διατήρηση τον δεδομένων τοπικά, αφού εκεί γίνονται και οι περισσότεροι υπολογισμοί.[64], [65]

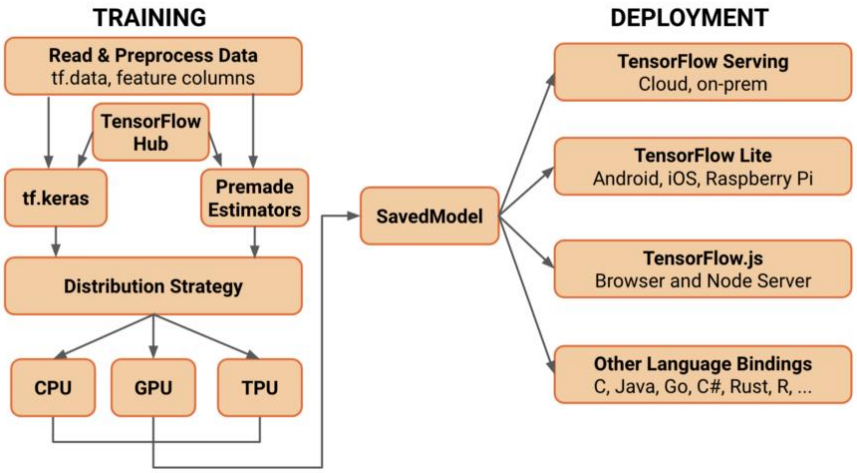
3.5.7 SavedModel

Το SavedModel είναι η καθολική μορφή σειριοποίησης για τα μοντέλα TensorFlow και περιέχει την πλήρης έκδοση εκπαιδευμένου μοντέλου, συμπεριλαμβανομένων των παραμέτρων και των επιπέδων υπολογισμών. Το SavedModel παρέχει μια ουδέτερη ως προς τη γλώσσα μορφή για την αποθήκευση μοντέλων μηχανικής μάθησης, η οποία είναι ανακτήσιμη και ερμητική. Επιτρέπει σε συστήματα και εργαλεία υψηλότερου επιπέδου να παράγουν, να χρησιμοποιούν και να μετασχηματίζουν μοντέλα TensorFlow. Δε χρειάζεται την αρχική έκδοση του μοντέλου για την εκτέλεση του, γεγονός που το καθιστά χρήσιμο για κοινή χρήση και ανάπτυξη με το TFLite, το Tensorflow.js, το TensorFlow Serving ή το TensorFlow Hub. Με τη χρήση του API που δίνεται από το TensorFlow, μπορούμε να αποθηκεύσουμε και να φορτώσουμε ένα μοντέλο οποιαδήποτε στιγμή, χρησιμοποιώντας το SavedModel του. [66]

3.5.8 TensorFlow 2.0

Η έκδοση 2.0 του TensorFlow έγινε διαθέσιμη 3 χρόνια μετά την πρώτη, τον Ιανουάριο του 2019. Υπήρχαν αρκετές αλλαγές και διαφοροποιήσεις συγκριτικά με τον προκάτοχο του, όπως η ευκολότερη χρήση του API Keras για τη δημιουργία μοντέλων, η χρήση ανάμεσα σε διαφορετικές πλατφόρμες και γλώσσες, η δυνατότητα εξαγωγής του μοντέλου σε SavedModel και η χρήση του από διάφορες συσκευές και περιβάλλοντα, αλλά και ο γενικότερος τρόπος που δουλεύει το μοντέλο πιο αποτελεσματικά και απλοποιημένα.

Η ροή του νέου μοντέλου ξεκινάει φορτώνοντας τα δεδομένα από τον αποθηκευτικό χώρο, έπειτα έρχεται η δημιουργία, εκπαίδευση και επαλήθευση του μοντέλου και στη συνέχεια η αποσφαλμάτωση. Το API του TensorFlow υποστηρίζει διάφορα εύρη επεξεργαστικών μονάδων όπως CPU, GPU και TPU, έτσι ώστε η εκπαίδευση να μπορεί να γίνει σε κατανεμημένα συστήματα, με πολλαπλά νήματα και γενικότερα αρκετά ευέλικτα. Τέλος, το μοντέλο εξάγεται σε μορφή SavedModel, από την οποία μπορεί να χρησιμοποιηθεί από διάφορα συστήματα, όπως περιηγητές με τη χρήση του TensorFlow.js, σε κινητές συσκευές και ενσωματωμένα συστήματα με τη χρήση του TensorFlow Lite, αλλά και Cloud συστήματα μέσω του Tensorflow Serving.[16]



Σχήμα 3.36: Διάγραμμα λειτουργίας ενός μοντέλου TensorFlow.

3.5.9 TensorFlow Intepreter

Ένας διερμηνέας (Intepreter) ενσωματώνει ένα προ-εκπαιδευμένο μοντέλο TensorFlow Lite, στο οποίο εκτελούνται λειτουργίες για την εξαγωγή συμπερασμάτων του μοντέλου. Οι σειρές των εισόδων και των εξόδων καθορίζονται κατά τη μετατροπή του μοντέλου TensorFlow σε μοντέλο TensorFlowLite με τον TFLiteConverter, όπως και τα προεπιλεγμένα σχήματα των εισόδων. Όταν οι είσοδοι παρέχονται ως (πολυδιάστατοι) πίνακες, οι αντίστοιχοι τένσορες εισόδου θα αλλάξουν σιωπηρά το μέγεθος σύμφωνα με το σχήμα αυτού του πίνακα. Όταν οι είσοδοι παρέχονται ως τύποι Buffer, δεν γίνεται σιωπηρή αλλαγή μεγέθους και καλών πρέπει να διασφαλίσει ότι το μέγεθος byte του Buffer είτε ταιριάζει με αυτό του αντίστοιχου τένσορα, είτε ότι πρώτα θα αλλάξει το μέγεθος του τένσορα μέσω της συνάρτησης resizeInput(). Η βιβλιοθήκη TFLite είναι κατασκευασμένη σύμφωνα με το NDK API 19, το οποίο είναι και το ελάχιστο SDK που χρειάζεται, για να λειτουργήσει σωστά σε μία Android εφαρμογή.

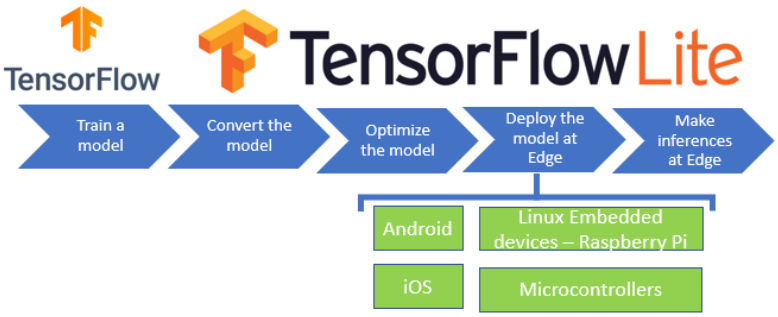
3.5.10 TensorFlow Lite

Το TensorFlow Lite είναι ένα framework του TensorFlow, το οποίο μετατρέπει το ήδη εκπαιδευμένο μοντέλο από την αρχική του μορφή SavedModel, σε μια πιο ελαφριά έκδοση για κινητές συσκευές, όπως Android ή iOS, αλλά και ενσωματωμένα συστήματα όπως Raspberry Pie και μικροελεγκτές.[67] Η κύρια χρήση αυτής της μορφής του μοντέλου είναι η απόδοση του όταν χρησιμοποιείται για Edge Computing. Βοηθάει στην χαμηλό μέγεθος των δεδομένων και τη γρήγορη μετακίνηση τους μεταξύ συσκευής-server, την ασφάλεια των δεδομένων, αφού δεν ξεφεύγουν από το τοπικό δίκτυο και επιλογές και άμεση χρήση του λόγο του ότι είναι ήδη εκπαιδευμένο σε κάποιο τοπικό server ή στο cloud.

Η διαδικασία για να χρησιμοποιήσουμε ένα μοντέλο τύπου TFLite είναι να εκπαιδεύσουμε έναν αλγόριθμο είτε με προ-εκπαιδευμένα βάρη είτε από την αρχή. Έπειτα, το αποτέλεσμα του εκπαιδευμένου μοντέλου χρειάζεται να βελτιστοποιηθεί με κάποιες τεχνικές και στη συνέχεια, από SavedModel να εξαχθεί σε TFLite για να χρησιμοποιηθεί στην εκάστοτε εφαρμογή.[68]

Με το να βελτιστοποιήσουμε ένα μοντέλο πετυχαίνουμε πολλά πράγματα, εκ των οποίον ένα από αυτά είναι να μειώσουμε το μέγεθος του. Έχοντας ένα μικρότερο μέγεθος χωρίς να μειώσουμε την αποτελεσματικότητα του μοντέλου, καταφέρνουμε να το κάνουμε πιο ελαφρύ, για να μπορούν τα το τρέξουν συσκευές οι οποίες χρησιμοποιούν την αρχιτεκτονική του edge computing, αλλά και για να είναι γρηγορότερη η προσπέλαση της RAM και ο χρόνος που χρειάζεται για να κατέβει το μοντέλο από το διαδίκτυο. Οι δύο βασικοί τρόπου που χρησιμοποιούνται για τη βελτιστοποίηση ενός μοντέλου είναι

το Quantization και το Weight Pruning.[68] Ο πρώτος, επεξεργάζεται την μορφή του μοντέλου, η οποία είναι σε SavedModel, και μετατρέπει την συνάρτηση ενεργοποίησης, τα βάρη και τις κλίσεις από 32-bit αριθμούς κινητής υποδιαστολής σε 16-bit ή 8-bit κινητής υποδιαστολής ή σε ακέραιους, μειώνοντας έτσι το συνολικό μέγεθος του αλγορίθμου.[69] Ο δεύτερος τρόπος λειτουργεί όπως το κλάδεμα που κάνουμε στο κήπο μας, δηλαδή αφαιρούμε τις παραμέτρους οι οποίες δεν επηρεάζουν την απόδοση του μοντέλου, δημιουργώντας μια συμπιεσμένη έκδοση με ίδια αποτελέσματα, αλλά ταχύτερη εκτέλεση.[70]



Σχήμα 3.37: Tensorfow Lite Flow.

3.5.11 Κβαντισμός (Quantization)

Πίνακας 3.5: Πίνακας μοντέλων κβαντισμού.

Τεχνική	Απαιτούμενα δεδομένα	Μείωση μεγέθους	Επιτυχία	Υποστηρίζεται από
Κβαντισμός Float-16	Δε χρειάζονται	Μέχρι 50%	Αμελητέα μείωση	CPU, GPU
Δυναμικός κβαντισμός	Δε χρειάζονται	Μέχρι 75%	Ελάχιστη μείωση	CPU, GPU (Android)
Κβαντισμός ακεραίων	Δείγμα αντιπροσωπευτικών δεδομένων	Μέχρι 75%	Ελάχιστη μείωση	CPU, GPU (Android), EdgeTPU, Hexagon DSP
Κβαντισμός με επίγνωση	Δεδομένα με ετικέτες	Μέχρι 75%	Η μικρότερη δυνατή	CPU, GPU (Android), EdgeTPU, Hexagon DSP

3.5.11.1 Κβάντιση float16 μετά την εκπαίδευση

Το TensorFlow Lite υποστηρίζει τη μετατροπή των βαρών σε τιμές κινητής υποδιαστολής 16 bit κατά τη μετατροπή του μοντέλου από το TensorFlow στη μορφή flat buffer του TensorFlow Lite. Αυτό έχει ως αποτέλεσμα τη μείωση του μεγέθους του μοντέλου κατά 2 φορές. Ορισμένο υλικό, όπως οι GPU, μπορεί να υπολογίζει εγγενώς σε αυτή την αριθμητική μειωμένης ακρίβειας, πραγματοποιώντας μια επιτάχυνση σε σχέση με την παραδοσιακή εκτέλεση με κινητές μονάδες. Ο αντιπρόσωπος GPU του

TensorFlow Lite μπορεί να ρυθμιστεί ώστε να εκτελείται με αυτόν τον τρόπο. Ωστόσο, ένα μοντέλο που έχει μετατραπεί σε βάρη float16 μπορεί ακόμα να εκτελεστεί στη CPU χωρίς πρόσθετη τροποποίηση. Τα βάρη float16 αναβαθμίζονται σε float32 πριν από την πρώτη εξαγωγή συμπερασμάτων. Αυτό επιτρέπει σημαντική μείωση του μεγέθους του μοντέλου με αντάλλαγμα ελάχιστες επιπτώσεις στην καθυστέρηση και την ακρίβεια.

3.5.11.2 Κβαντισμός δυναμικού εύρους μετά την εκπαίδευση

Το TensorFlow Lite υποστηρίζει τη μετατροπή των βαρών σε ακρίβεια 8 bit ως μέρος της μετατροπής του μοντέλου από τα tensorflow graphdefs στη μορφή flat buffer του TensorFlow Lite. Η κβάντιση δυναμικού εύρους επιτυγχάνει μείωση του μεγέθους του μοντέλου κατά 4 φορές. Επιπλέον, το TFLite υποστηρίζει "on the fly" κβάντιση και αποκβάντιση των ενεργοποιήσεων για να επιτρέπει:

1. Χρήση κβαντισμένων πυρήνων για ταχύτερη υλοποίηση όταν είναι διαθέσιμοι.
2. Ανάμειξη πυρήνων κινητής υποδιαστολής με κβαντισμένους πυρήνες για διαφορετικά τμήματα του γραφήματος.

Οι ενεργοποιήσεις αποθηκεύονται πάντα σε κινητή μονάδα υποδιαστολής. Για τις λειτουργίες που υποστηρίζουν κβαντισμένους πυρήνες, οι ενεργοποιήσεις κβαντίζονται δυναμικά σε ακρίβεια 8 bits πριν από την επεξεργασία και απο-κβαντίζονται σε ακρίβεια κινητής υποδιαστολής μετά την επεξεργασία. Ανάλογα με το μοντέλο που μετατρέπεται, αυτό μπορεί να δώσει μια επιτάχυνση σε σχέση με τον υπολογισμό με καθαρή κινητή μονάδα.

Σε αντίθεση με την εκπαίδευση με επίγνωση της κβάντισης , τα βάρη κβαντίζονται μετά την εκπαίδευση και οι ενεργοποιήσεις κβαντίζονται δυναμικά κατά την εξαγωγή συμπερασμάτων σε αυτή τη μέθοδο. Επομένως, τα βάρη του μοντέλου δεν επανεκπαιδεύονται για να αντισταθμίσουν τα σφάλματα που προκαλούνται από την κβάντιση. Είναι σημαντικό να ελέγχεται η ακρίβεια του κβαντισμένου μοντέλου για να διασφαλιστεί ότι η υποβάθμιση είναι αποδεκτή.

3.5.11.3 Ακέραιος κβαντισμός μετά την εκπαίδευση

Η κβάντιση ακέραιων αριθμών είναι μια στρατηγική βελτιστοποίησης που μετατρέπει αριθμούς κινητής υποδιαστολής 32 bit (όπως τα βάρη και οι έξοδοι ενεργοποίησης) στους πλησιέστερους αριθμούς σταθερής υποδιαστολής 8 bit. Αυτό έχει ως αποτέλεσμα ένα μικρότερο μοντέλο και αυξημένη ταχύτητα εξαγωγής συμπερασμάτων, η οποία είναι πολύτιμη για συσκευές χαμηλής κατανάλωσης ενέργειας, όπως οι μικροελεγκτές ή τα κινητά τηλέφωνα. Αυτή η μορφή δεδομένων απαιτείται επίσης από επιταχυντές που χρησιμοποιούν μόνο ακέραιους αριθμούς, όπως η Edge TPU.

Για να κβαντοποιήσουμε τα μεταβλητά δεδομένα (όπως η είσοδος/έξοδος του μοντέλου και τα ενδιάμεσα επίπεδα μεταξύ των στρωμάτων), πρέπει να παρέχουμε ένα σύνολο αντιπροσωπευτικών δεδομένων στη συνάρτηση RepresentantiveDataset(). Πρόκειται για μια συνάρτηση γεννήτριας που παρέχει ένα μικρό σύνολο δεδομένων για τη βαθμονόμηση ή την εκτίμηση του εύρους, δηλαδή του (ελάχιστου, μέγιστου) όλων των πινάκων κινητής υποδιαστολής στο μοντέλο (όπως η είσοδος του μοντέλου, οι έξοδοι ενεργοποίησης των ενδιάμεσων στρωμάτων και η έξοδος του μοντέλου) για κβάντιση. Συνήθως, πρόκειται για ένα μικρό υποσύνολο μερικών εκατοντάδων δειγμάτων που επιλέγονται τυχαία, χωρίς συγκεκριμένη σειρά, από το σύνολο δεδομένων εκπαίδευσης ή αξιολόγησης.[71]

3.5.11.4 Εκπαίδευση με γνώση κβαντισμού

Η εκπαίδευση με επίγνωση της κβάντισης προσομοιώνει την κβάντιση σε χρόνο εξαγωγής συμπερασμάτων, δημιουργώντας ένα μοντέλο που τα επόμενα εργαλεία θα χρησιμοποιήσουν για να παράγουν πραγματικά κβαντισμένα μοντέλα. Τα κβαντισμένα μοντέλα χρησιμοποιούν χαμηλότερη ακρίβεια (π.χ. 8-bit αντί για 32-bit κινητής υποδιαστολής), οδηγώντας σε διάφορα οφέλη κατά την ανάπτυξη.

3.6 Android

3.6.1 Εισαγωγή στο Android

Τον Νοέμβριο του 2007 η Google παρουσίασε για πρώτη φορά, την πλατφόρμα της για κινητά τηλέφωνα ονόματι “Android”. Ένα λογισμικό ανοιχτού κώδικα το οποίο θα έμπαινε σε κάθε έξυπνη συσκευή όπως τάμπλετ, έξυπνοι υπολογιστές, έξυπνα κινητά κ.α.[72] Το Android έκανε μεγάλη επιτυχία στην αγορά καταφέρνοντας να έχει το 84% του μεριδίου της αγοράς το 2021[73] και να συνεχίζει να φέρνει νέες και πρωτοπόρες τεχνολογίες για τους χρήστες του.

Το 2003 όπου πρωτοξεκίνησε η Android Inc, ήταν μια αμερικάνικη εταιρεία που έφτιαχνε λειτουργικά συστήματα για τις κάμερες των κινητών τηλεφώνων. Το 2004 το project αγοράστηκε από την Google και στη συνέχεια, την ανάπτυξη της πλατφόρμας την ανέλαβε η Open Handset Alliance (OHA), η οποία την συνεχίζει μέχρι και σήμερα.

Η OHA αποτελείται από ένα γκρουπ εταιρειών όπως η Google, Intel, NVIDIA, Qualcomm, Motorola, HTC και T-Mobile, οι οποίες είναι ο λόγος για τον οποίο το Android είναι η ναυαρχίδα των λειτουργικών συστημάτων για έξυπνες συσκευές.

3.6.2 Τα βασικά στοιχεία του Android

Το λειτουργικό του είναι βασισμένο σε πυρήνα Linux. Ο τρόπος με τον οποίο διανέμεται δίνει τους προγραμματιστές την ελευθερία και την ευελιξία για να σχεδιάζουν τα δικά τους προϊόντα, επιτρέποντας οποιαδήποτε εφαρμογή να μπορεί να εκμεταλλευτεί πλήρως τις δυνατότητες των κινητών τηλεφώνων χρησιμοποιώντας την κάμερα, την δυνατότητα αποστολής μηνυμάτων, την πραγματοποίηση κλήσεων και τη χρήση διάφορων μετρητών περιβάλλοντος υλικού. Επιπλέον χρησιμοποιεί μια προσαρμοσμένη εικονική μηχανή που έχει σχεδιαστεί για τη βελτιστοποίηση των πόρων μνήμης. Έτσι, μπορεί να επεκταθεί και να ενσωματώσει νέες τεχνολογίες αιχμής αλλά και επεκταθεί ελεύθερα.

3.6.3 Οι εκδόσεις του Android

Σε αυτή την ενότητα θα δούμε τις πιο σημαντικές αλλαγές και προσθήκες στις εκδόσεις του λειτουργικού συστήματος Android, από την πρώτη κυκλοφορία του μέχρι και σήμερα.

3.6.3.1 Alpha (Android 1.0)

Η πρώτη έκδοση του λειτουργικού συστήματος Android δημοσιεύθηκε το Σεπτέμβρη του 2004. Η πρώτη συσκευή που υιοθέτησε και χρησιμοποίησε το λειτουργικό ήταν το HTC Dream (G1) της εταιρείας T-Mobile.[74]

3.6.3.2 Beta (Android 1.1)

Δεν υπάρχουν πολλές πληροφορίες για την beta έκδοση του Android, πέρα από το ότι υποστήριζε την αναπαραγωγή βίντεο, τη χρήση πληκτρολογίων από τρίτους, την εγγραφή βίντεο και την αντιγραφή επικόλληση στον βασικό ενσωματωμένο περιηγητή.

3.6.3.3 Cupcake (Android 1.5)

Αυτή ήταν η πρώτη επίσημη έκδοση, η οποία έγινε διαθέσιμη εγγενώς σε διάφορα κινητά τηλέφωνα και έφερε βελτίωση όλων των βασικών στοιχείων διεπαφής, χρήση του επιταχυνσιόμετρου για τις εφαρμογές που χρησιμοποιούν εφέ περιστροφής, προσθήκη δυνατότητας κλήσεων, αποθήκευση επαφών, περιηγητή, υπηρεσιών email, ημερολόγιο κάμερα και συλλογή φωτογραφιών αλλά και διαχειριστή εφαρμογών. Βελτιώθηκαν οι επιδώσεις του λειτουργικού, προστέθηκαν τα widgets, το Bluetooth, αναβαθμίστηκε ο πυρήνας του Linux και αναβαθμίστηκε το API και οι λειτουργίες του (API 3)[75, σ. 5]

3.6.3.4 Donut (Android 1.6)

Στην επόμενη έκδοση με όνομα «Donut» προστέθηκαν λειτουργίες οι οποίες θα βοηθούσαν τον χρήστη να μπορέσει να περιηγηθεί και να εκμεταλλευτεί τις δυνατότητες του λειτουργικού καλύτερα. Ιδιαίτερα σημαντική είναι η προσθήκη της γρήγορης αναζήτησης, το αναβαθμισμένο UI της συλλογής φωτογραφιών αλλά και της εφαρμογής για λήψη φωτογραφίας, η αναβάθμιση του πρωτοκόλλου VPN σε 802.1x, η προσθήκη του εικονιδίου μπαταρίας για να καταλαβαίνει ο χρήστης ποιο εύκολα την κατάσταση της μπαταρίας του κινητού του αλλά και αλλαγές στην εφαρμογή του καταστήματος της Google, το «Google Play». Επιπλέον, προστέθηκαν κάποιες λειτουργίες για τους προγραμματιστές, αναβαθμίζοντας την έκδοση του API την 4η.[76, σ. 6]

3.6.3.5 Eclair (Android 2.0 – 2.1)

Η έκδοση Eclair παρουσιάστηκε από την Google τον Οκτώβρη του 2009, προσθέτοντας σημαντικές αλλαγές στην προκάτοχο της (Android 1.6). Σημαντικές αλλαγές προστέθηκαν στην αρχική οθόνη όπου το Google Search πλέον εμφανίζεται στην κορυφή της αρχικής οθόνης, επανασχεδιάστηκαν κάποιες λειτουργίες της κάμερας, δημιουργήθηκαν νέες λειτουργίες για καλύτερες λήψεις, όπως ψηφιακό zoom, εφέ χρώματος και εργαλεία επεξεργασίας φωτογραφιών.

Στην πλευρά της πλατφόρμας, έχει προστεθεί η υποστήριξη της τεχνολογίας NFC (Near Field Communication), αναζήτηση στα αποθηκευμένα μηνύματα, βελτιώσεις στην εφαρμογή Google Maps (Έκδοση 3.1.2) και υποστήριξη Exchange στην εφαρμογή Email. Επίσης σημαντικές βελτιώσεις έγιναν στον προκαθορισμένο περιηγητή με τη πιο σημαντική, την υποστήριξη της HTML 5.[77, σ. 0]

3.6.3.6 Froyo (Android 2.2)

Η επόμενη αναβάθμιση του λειτουργικού ήρθε με την κωδική ονομασία “Froyo” τον Μάϊο του 2010 στο συνέδριο Google I/O. Σε αυτή την έκδοση προστέθηκαν η λειτουργία Wi-Fi Hotspot και δυνατότητα ανταλλαγής δεδομένων μέσω σύνδεσης με καλώδιού USB. Πλέον δίνεται η δυνατότητα στις εφαρμογές να δημιουργούν ειδοποιήσεις μέσω της λειτουργίας «Push Notifications». Επιπρόσθετα, βελτιώθηκε η ταχύτητα εκτέλεσης εφαρμογών και λειτουργικού περίπου. [78] Το API πλέον βρίσκεται στην έκδοση 8. [79], [80]

3.6.3.7 Gingerbread (Android 2.3-2.3.7)

Τον Δεκέμβρη του 2010 κυκλοφόρησε η έκδοση Gingerbread, η οποία ξεκίνησε την υποστήριξη των ηλεκτρονικών πληρωμών με τη χρήση του NFC, και την χρήση VoIP (Voice Over IP) για τηλεφωνία μέσω ίντερνετ. Στην έκδοση Gingerbread η διεπαφή χρήστη είχε υποστεί πολλές βελτιστοποιήσεις με πολλούς τρόπος, με αποτέλεσμα να είναι πιο αποδοτική, πιο γρήγορη και να εξοικονομεί περισσότερη ενέργεια. Η παλέτα χρωμάτων ανανεώθηκε αποδίδοντας καλύτερα και με ζωντάνια τα στοιχεία στην οθόνη του χρήστη.[81, σ. 3]

3.6.3.8 Honeycomb (Android 3.0 – 3.2.6)

Πρώτη φορά το Android 3.0 εμφανίστηκε στο κινητό Motorola Xoom τον Φευρουάριο του 2011, ωστόσο έγινε διαθέσιμο πολύ σύντομα και για άλλες συσκευές όπως τα τάμπλετς της Samsung και της LG. Οι δυνατότητες που παρουσιάστηκαν ήταν η επιλογές που είχε για να προσαρμόσουν οι χρήστες τις εφαρμογές τους έχοντας να επιλέξουν ανάμεσα σε 5 διαθέσιμες αρχικές οθόνες. Πλέον η μπάρα στην αρχή την οθόνης κατεβαίνει μέχρι κάτω παρουσιάζοντας τις ειδοποιήσεις των εφαρμογών που ενημερώνουν τον χρήστη. Επίσης δίνεται η δυνατότητα να προσαρμοστεί η ίδια η μπάρα προσθέτοντας ή αφαιρώντας widgets. Επανασχεδιάστηκε το βασικό πληκτρολόγιο και πλέον υποστηρίζονται πολλαπλοί επεξεργαστές στον σχεδιασμό υλικού. Το API βρίσκεται πλέον στο επίπεδο 11 με 13, ανάλογα την έκδοση του Honeycomb για την οποία αναφερόμαστε.[82]

3.6.3.9 Ice Cream Sandwich (Android 4.0 – 4.0.4)

Η 4η έκδοση του Android, έφερε αλλαγές στο UI, βελτιώνοντας την είσοδο φωνητικών εντολών, τον έλεγχο ορθογραφικών, και άλλες επιλογές για την προσβασιμότητα των χρηστών. Προστέθηκε επίσης η δυνατότητα του ξεκλειδώματος με τη χρήση της αναγνώρισης προσώπου, και η επικοινωνία μέσω Wi-Fi P2P (Peer-To-Peer) μια δυνατότητα που επιτρέπει στις συσκευές οι οποίες είναι κοντά να επικοινωνούν μέσω Wi-Fi.[83]

3.6.3.10 Jelly Bean (Android 4.1 – 4.3.1)

Στην έκδοση Jelly Bean, προστέθηκε ο γονικός έλεγχος, η έξυπνη λειτουργία Bluetooth η οποία βελτιστοποίησε την κατανάλωση μπαταρίας σε διάφορες συσκευές, και βελτιώθηκαν μερικές εφαρμογές όπως το πληκτρολόγιο κλήσης και η χρήση της Αραβικής γλώσσας. [84]

3.6.3.11 KitKat (Android 4.4 - 4.4.4)

Σε αυτή την έκδοση, βρισκόμαστε πλέον στο API 19 και πλέον οι δυνατότητες που προσφέρονται στον χρήστη είναι πάρα πολλές. Έχει φρεσκαριστεί ο σχεδιασμός της διεπαφής, έχει προστεθεί αναγνώριση φωνητικών εντολών, έχει βελτιστοποιηθεί ο τρόπος που το λογισμικό οργανώνει την παράλληλη λειτουργία πολλών εφαρμογών, με στόχο την καλύτερη αξιοποίηση των πόρων. Βελτιώθηκε επίσης η υπηρεσία email αλλά και μηνυμάτων καθώς και προστέθηκε η δυνατότητα εκτύπωσης εγγράφων από οπουδήποτε ασύρματα. Άλλες τεχνικές αλλαγές είναι η εξοικονόμηση ενέργειας σε λειτουργίες όπως αναπαραγωγή ήχου μέσω του Audiojack, η καλύτερη επεξεργασία φωτογραφιών σε μεγαλύτερη ανάλυση, βελτίωση στο touchscreen και στον τρόπο ανέπαφων πληρωμών. [85]

3.6.3.12 Lollipop (Android 5.0 – 5.1.1)

Αυτή η έκδοση, χρησιμοποιεί την γλώσσα σχεδιασμου “Material Design”, η οποία διατηρεί τη αίσθηση χαρτιού στο περιβάλλον διεπαφής του χρήστη. Μια πολύ σημαντική αλλαγή έγινε στην πλατφόρμα η οποία πρότινος χρησιμοποιούσε την εικονική μηχανή του Dalvik, ενώ πλέον ήρθε η ART (Android

Runtime) το οποίο υπόσχεται περισσότερες βελτιστοποιήσεις και καλύτερη χρήση της μπαταρίας. Τον Νοέμβριο του 2016 όλες οι συσκευές Android φαίνεται να τρέχουν την έκδοση Lollipop σε ποσοστό 34%, ενώ η έκδοση του API βρίσκεται στην 21η με 22η.[86, σ. 1]

3.6.3.13 Marshmallow (Android 6.0 – 6.0.1)

Στο Android 6.0 με κωδική ονομασία "Marshmallow" εξασφαλίστηκε η επιλογή του χρήστη να μπορεί να ελέγχει τα δικαιώματα που δίνει στις εφαρμογές αυξάνοντας το την αξία των προσωπικών δικαιωμάτων που μοιράζεται ο χρήστης, έκανε πιο εύκολη την αναζήτηση αλλάζοντας κάποια σημεία στην κύρια διεπαφή του λειτουργικού και έδωσε τη δυνατότητα αποθήκευσης προγραμμάτων και εφαρμογών σε μια εξωτερική μνήμη τύπου microSD για την επέκταση του διαθέσιμου χώρου. Σε αυτή την έκδοση ήρθε για πρώτη φορά η ταχεία φόρτιση μέσω USB Type-C και πλέον οι χρήστες μπορούν να ξεκλειδώνουν το κινητό τους με τη χρήση του δακτυλικού τους αποτυπώματος.[87]

3.6.3.14 Nougat (Android 7.0 – 7.1.2)

Στην έκδοση Android Nougat (ή αλλιώς έκδοση N), η οποία κυκλοφόρησε τον Αύγουστο του 2016, εισάγονται αλλαγές στον τρόπο προβολής των εφαρμογών, δηλαδή προστίθεται η λειτουργία διαίρεσης οθόνης, βελτιωμένη χρήση μπαταρίας, προσθήκη emoji, και νέο περιβάλλον Java βασισμένο στο OpenJDK. Πλέον τα γραφικά υποστηρίζονται από το Vulkan API και το API του ίδιου του Android βρίσκεται στην έκδοση 24. [88]

3.6.3.15 Oreo (Android 8.0 – 8.1)

Η έκδοση Oreo έρχεται με καλύτερα γραφικά και πιο εξυπηρετικό UI κάνοντας όπως μας αναφέρει η Google δύο φορές ταχύτερη τη συσκευή. Νέες δυνατότητες προστέθηκαν όπως η «Εικόνα μέσα σε εικόνα», η οποία επιτρέπει 2 εφαρμογές να χρησιμοποιούν την οθόνη και να εμφανίζονται η μία πάνω στην άλλη. Το API έχει αναπτυχθεί και βρίσκεται στην έκδοση 26. [89]

3.6.3.16 Pie (Android 9.0)

Τα νέα χαρακτηριστικά σε αυτή την έκδοση, είναι η αλλαγή στη διεπαφή χρήστη, η προσθήκη της λειτουργίας στιγμιότυπου , υποστήριξη τεχνολογίας HEIF, προσθήκη αισθητήρα φωτεινότητας για να προσαρμόζεται η συσκευή με τις ανάγκες του περιβάλλοντος, δημιουργία πολλαπλών χρηστών σε μία συσκευή και άλλες αλλαγές για την προσβασιμότητα. [90]

3.6.3.17 Q (Android 10.0)

Από την έκδοση 10 και μετά η Google έπαψε να δίνει κώδικά ονόματα γλυκών στις συσκευές της και πλέον συνεχίζουμε μόνο με τα γράμματα της αλφαβήτου. Κυκλοφόρησε τον Μάρτιο του 2019 φέρνοντας αλλαγές όπως το σκοτεινό θέμα στις εφαρμογές, τεχνολογία η οποία δημιουργεί ζωντανά υπότιτλους σε κάθε λογής βίντεο χρησιμοποιώντας αλγορίθμους της Google, προετοιμάζει το έδαφος για το 5G και συνεχίζει με διάφορες βελτιώσεις συστήματος. [38]

3.6.3.18 R (Android 11.0)

Η τελευταία δημοσιευμένη έκδοση του Android κυκλοφορεί με το όνομα "Android R", μιας και είναι η 11η έκδοση του λογισμικού από την Google τον Φεβρουάριο του 2020. Σε αυτή την έκδοση δεν έχει προστεθεί κάτι καινούριο, πέρα από τις βελτιώσεις στις υπάρχουσες δυνατότητες των προηγούμενων εκδόσεων. Ωστόσο, με την αύξηση του IoT, οι συσκευές μπορούν να συνδέονται μεταξύ τους, αλλά και να συνδεθεί μια συσκευή με κάποιο αμάξι και να αντικατοπτρίζονται όλες οι λειτουργίες στην εκάστοτε

διεπαφή, κάνοντας το λογισμικό ακόμη πιο ευέλικτο για κάθε είδους χρήση. Το API της πλατφόρμας πλέον βρίσκεται στην 30η του έκδοση.

3.6.3.19 Android 12

Το Android 12 είναι η επόμενη έκδοση, για την οποία δεν ξέρουμε ακόμη το όνομα της, αλλά ξέρουμε ότι θα φέρει αλλαγές στοχεύοντας στην καλύτερη εμπειρία χρήστη, καλύτερα gestures, τεχνικές αναβαθμίσεις στις μεθόδους του κώδικα, βελτιωμένη απόδοση, υψηλότερη ασφάλεια και προσωπικό απόρρητο αλλά και γρηγορότερη ανάδραση μεταξύ διεπαφής και χρήστη. Η Google στοχεύει στο να φτιάξει μια πιο προσωποποιημένη εμπειρία, μαθαίνοντας για τον χρήστη και κάνοντας διάφορες προτάσεις με βάση τους αλγορίθμους τεχνητής νοημοσύνης που έχει δημιουργήσει μαζί με τους συνεργάτες της. [39]

3.6.4 Android Studio

Το Android Studio είναι ένα περιβάλλον ανάπτυξης, το οποίο δημιουργήθηκε από την Google σε συνεργασία με την JetBrains επάνω στο υπάρχον λειτουργικό του IntelliJ IDEA. Η Google ανακοίνωσε την ανάπτυξη του προγράμματος τον Μάιο του 2013 και προχώρησε με την πρώτη δημοσίευση τον Ιούνιου του επόμενου έτους. Η πρώτη σταθερή έκδοση (1.0) έγινε διαθέσιμη τον Δεκέμβρη του 2014, ενώ μέχρι σήμερα, έχουν πραγματοποιηθεί αρκετές αναβαθμίσεις, με αποτέλεσμα να βρισκόμαστε στην έκδοση «Artic Fox», η οποία έγινε διαθέσιμη τον Ιούλιο του 2021. [93]

Ένα πρόγραμμα IDE, δίνει τη δυνατότητα στον χρήστη να γράψει πηγαίο κώδικα μέσα από ένα γραφικό περιβάλλον, ο οποίος μεταγλωττίζεται σε γλώσσα μηχανής και εκτελείται τοπικά, δίνοντας τη δυνατότητα της ζωντανής προβολής του προγράμματος από τον προγραμματιστή εύκολα και γρήγορα. Επίσης, υπάρχει η δυνατότητα για εντοπισμό σφαλμάτων η οποία μπορεί να εμφανίσει γραφικά τη θέση ενός σφάλματος στον κώδικα.[94]

Σε συνδυασμό με το λειτουργικό Android, η Google μαζί με την JetBrains, δημιούργησαν ένα προσαρμοσμένο IDE το οποίο θα έδινε τη δυνατότητα στους προγραμματιστές να δημιουργήσουν δικά τους προγράμματα και εφαρμογές για κινητά τηλέφωνα, τάμπλετ, έξυπνα ρολόγια χειρός ή οποιαδήποτε άλλη συσκευή τρέχει το λειτουργικό Android.

Μερικά από τα χαρακτηριστικά της τελευταίας έκδοσης είναι η υποστήριξη της πλατφόρμας του Gradle για να ‘χτιστεί’ μια εφαρμογή, προσαρμοσμένες διορθώσεις και συμβουλές για τις ιδιαιτερότητες που έχουν οι γλώσσες προγραμματισμού και οι τεχνικές για το περιβάλλον του Android. Επίσης το Android Studio υποστηρίζει διάφορες γλώσσες προγραμματισμού, όπως Java, C++, Go, αλλά πιο πρόσφατες γλώσσες, όπως η Kotlin.

3.6.5 Android APK

Ένα αρχείο APK (Android Package File) είναι μια συμπιεσμένη μορφή αρχείου, η οποία χρησιμοποιείται από το λειτουργικό σύστημα Android και περιέχει διάφορες βιβλιοθήκες, αρχεία και πηγαίο κώδικα, όλα αυτά μεταγλωττισμένα σε γλώσσα μηχανής η οποία έπειτα θα εκτελεσθεί από τις έξυπνες συσκευές που θα τα εγκαταστήσουν. Μερικά αρχεία που μπορούμε να βρούμε σε ένα APK είναι το “META-INF” το οποίο μπορεί να θεωρηθεί ως ένας ο οποίος περιέχει τις πιστοποιήσεις εγκυρότητας του APK, με υπογραφές οι οποίες είναι κρυπτογραφημένες, το AndroidManifest.xml, το οποίο περιέχει πληροφορίες όπως ποια δικαιώματα χρειάζεται η εφαρμογή για να λειτουργήσει, πιο υλικό θα χρησιμοποιήσει αλλά και ποιες είναι οι διεπαφές εκκίνησης της εφαρμογής και το αρχείο classes.dex, το οποίο είναι ένα ήδη μεταγλωττισμένο εκτελέσιμο αρχείο για την εικονική μηχανή Dalvik

ή Android Runtime (ART) και περιέχει τα κύρια κομμάτια του προγράμματος, τα οποία μπορούν πολύ εύκολα να αναστραφούν και να γίνουν γλώσσα υψηλού επιπέδου.

Η διαδικασία μετατροπής του αρχείου σε υπογεγραμμένο APK, ξεκινάει από τον πηγαίο κώδικα, ο οποίος είναι γραμμένος είτε σε Java είτε σε Kotlin, που περνάει από τον αντίστοιχο μεταγλωττιστή της κάθε γλώσσας (javac και kotlinc) και μετατρέπεται στο διάδικό σύστημα. Για να τρέξει στο λειτουργικό σύστημα Android, το μεταγλωττισμένο αρχείο περνάει από έναν ακόμη μεταγλωττιστή, τον DEX, του οποίου το τελικό αρχείο είναι και αυτό που θα διανεμηθεί στους τελικούς χρήστες.

Μόλις μαζευτούν και μεταγλωττιστούν σωστά όλα τα αρχεία που χρειάζονται, συμπιέζονται σε αυτό που ονομάζουμε αρχείο APK. Για να διανεμηθεί διαδικτυακά και με ασφάλεια, το τελευταίο βήμα είναι το αρχείο να υπογραφεί με κρυπτογράφηση, έτσι ώστε να διασφαλιστεί η ακεραιότητα της εφαρμογής και των δεδομένων της. [95]

3.6.6 Android SDK

Το Android SDK είναι ένα κιτ ανάπτυξης λογισμικού που περιλαμβάνει μια συλλογή από εργαλεία προγραμματισμού. Σε αυτά υπάγονται ο έλεγχος σφαλμάτων, βιβλιοθήκες κώδικα, προσομοιωτής, βιβλιοθήκες, παραδείγματα από κώδικα και μαθήματα. Μέχρι το 2014 το μόνο πρόγραμμα προγραμματισμού που υποστήριζε το Android SDK ήταν το Eclipse, μέχρι που το 2015 με το Android Studio, είχε δημοσιευθεί το Android Studio. [96]

3.6.7 Gradle

Το Gradle είναι ένα ανοιχτού κώδικα λογισμικό, το οποίο υιοθέτησε η Google, ως προεπιλεγμένο εργαλείο για τη δημιουργία αυτοματοποιημένης κατασκευής εφαρμογών για Android λειτουργικά συστήματα μέσω του Android Studio. Είναι αρκετά ευέλικτο και να δημιουργεί κάθε τύπο λογισμικού υποστηρίζοντας πολλά λογισμικά IDEs.

Επιτρέπει την δημιουργία του APK μαζί με τη χρήση βιβλιοθηκών εύκολα και γρήγορα, διασφαλίζοντας τη σωστή λειτουργεία της εφαρμογής και έλεγχο των βέλτιστων επιδόσεων χρησιμοποιώντας αποθηκευμένα αρχεία από προηγούμενες δημιουργίες και αξιοποιώντας μόνο ότι είναι αναγκαίο για να τρέξει το πρόγραμμα ανά περίπτωση. [97]

3.6.8 Αρχιτεκτονική Android Εφαρμογών

3.6.8.1 Activity και κύκλος ζωής

Ένα Activity είναι μια κλάση και ένα βασικό συστατικό για να δημιουργηθεί μια εφαρμογή Android. Στον κλασσικό προγραμματισμό, τα κύρια προγράμματα ξεκινάν με μια μέθοδο “main()” η οποία καλείτε με την εκκίνηση του προγράμματος. Στα συστήματα Android ωστόσο, καλείτε το Activity, το οποίο απαρτίζεται από τις μεθόδους του κύκλου ζωής του και την όψη του, που είναι ένα αρχείο xml.

Μια εφαρμογή Android μπορεί να έχει διάφορα Activities, τα οποία διαχειρίζονται μέσω των κύκλων ζωής του και πρέπει να δηλώνονται στο AndroidManifest.xml.

Στην αρχή της εκκίνησης ενός Activity καλείτε η μέθοδος onCreate() όπου περιέχει όλα τα συστατικά και τα δεδομένα που χρειάζεται να φορτώσουν με τελευταία την setContentView() για να ξεκινήσει να βλέπει τη διεπαφή ο χρήστης.

Στην έξοδο της προηγούμενης μεθόδου καλείτε η onStart(), η οποία εμφανίζει την διεπαφή που ετοιμάστηκε από την setContentView() στην οθόνη και πλέον ο χρήστης μπορεί να αρχίσει την διάδραση με αυτή.

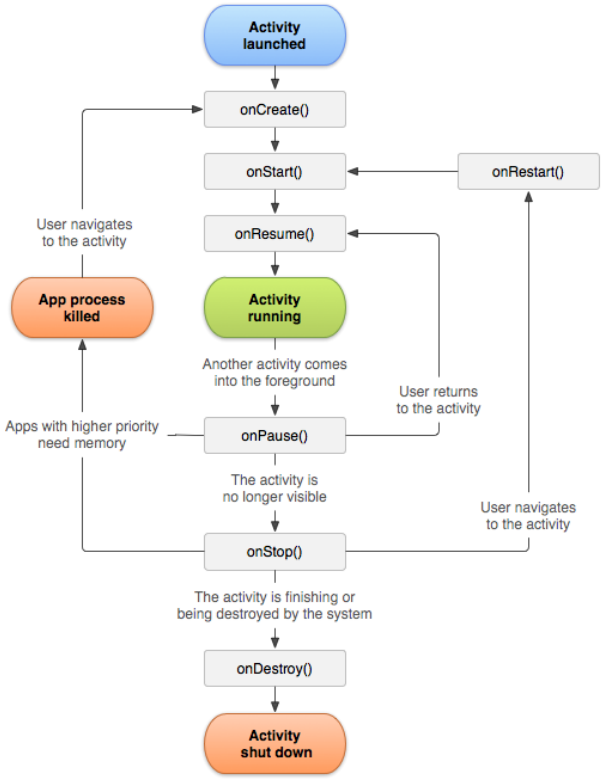
Η onResume() εισέρχεται εμβόλιμα από το σύστημα λίγο πριν η διεπαφή αρχίσει να αντιδρά με τον χρήστη και η δουλεία της είναι να συλλαμβάνει κάθε είσοδο του χρήστη. Οι πιο βασικές λειτουργίες της εφαρμογής έχουν υλοποιηθεί σε αυτή τη μέθοδο.

Όταν ο χρήστης αλλάζει εφαρμογή, ή χρησιμοποιεί τη λειτουργία πολλών παραθύρων και κατά κάποιο τρόπο το σύστημα καταλαβαίνει ότι πάει να φύγει. Αυτή η μέθοδος χρησιμοποιείται για να ορίσουμε ποιες λειτουργίες δεν χρειάζεται να συνεχίσουν να λειτουργούν όταν η εφαρμογή λειτουργεί στο παρασκήνιο. Όλα τα συστατικά που συνδέονται με αυτή την διεπιφάνια μπαίνουν σε κατάσταση onPause επίσης, περιμένοντας πότε θα επιστρέψουν με την onResume.

Όταν δεν υπάρχει το Activity στην οθόνη του χρήστη, τότε βρισκόμαστε στην κατάσταση onStop, όπου εδώ η Activity έχει ολοκληρωθεί και σύντομα θα καταστραφεί. Η μέθοδος onStop() μας δίνει τη δυνατότητα να αλλάξουμε παραμέτρους και λειτουργίες η οποίες δεν επηρεάζονται από το αν η διεπαφή είναι στο προσκήνιο σε αντίθεση με την onPause().

Το επόμενο στάδιο είναι η καταστροφή με την onDestroy() η οποία καλείτε μετά την onStop(), όπου όλες οι απαραίτητες ενέργειες έχουν πραγματοποιηθεί και πλέον είναι ασφαλές να τερματιστεί το Activity. Όταν συμβεί αυτό, είτε έγινε από επιλογή του χρήστη, είτε από το σύστημα. Γι' αυτό οι πληροφορίες του Activity περνάνε σε ένα αντικείμενο της κλάσης ViewModel, έτσι ώστε αν ξαναδημιουργηθεί το Activity, οι πληροφορίες να είναι διαθέσιμες άμεσα.

Στην παρακάτω εικόνα φαίνεται όλος ο κύκλος ενός Activity, από την αρχή της δημιουργίας του, μέχρι και την καταστροφή του. Επιπλέον φαίνεται άλλη μία μέθοδος, η Restart(), η οποία βρίσκεται ανάμεσα στην onStop() και onStart(), και καλείται όταν ο χρήστης αποφασίσει να επιστρέψει σε μια διεπιφάνια, η οποία δεν βρισκόταν στο προσκήνιο. [98], [99]



Σχήμα 3.38: Κύκλος ζωής ενός Activity.

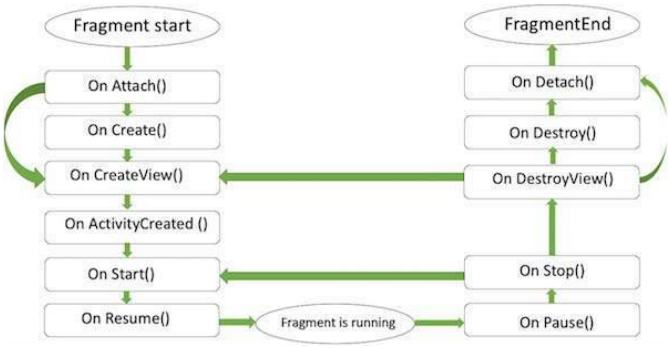
3.6.8.2 Fragment και κύκλος ζωής

Fragment είναι μία εξαρτημένη οντότητα κάτω από ένα ή περισσότερα Activities, το οποίο έχει τον δικό του κύκλο ζωής, τη δική του διάδραση και μπορεί να προστεθεί ή να αφαιρεθεί δυναμικά κατά τη διάρκεια εκτέλεσης ενός Activity.

Το Fragment μπορεί να χρησιμοποιηθεί από διαφορετικά Activities, ενώ όταν είναι ενεργό, τότε προστίθεται στο Back Stack του εκάστοτε Activity. Ένα παράδειγμα χρήσης του Back Stack είναι σε περιπτώσεις όπου ο χρήστης περιστρέφει την οθόνη, και έχουμε ετοιμάσει μια διαφορετική διάταξη για την οριζόντια ή κάθετη εφαρμογή.

Η φιλοσοφία του Fragment, είναι ότι μπορεί να ενσωματωθεί μέσα σε ένα activity, ή μέσα σε ένα άλλο Fragment, δημιουργώντας έτσι μια διεπαφή. Κάθε Fragment έχει μια μέθοδο onAttach() η οποία καλείται όταν ένα fragment συσχετιστεί με ένα Activity, μια onCreate() (Όπως και τα Activities) η οποία δημιουργεί το Fragment, μια onCreateView() για να δημιουργήσει τη διεπαφή με τον χρήστη και την onActivityCreated() που εκτελείται μόλις ολοκληρωθεί η onCreate().

Πέρα από αυτές, υπάρχουν και η onStart() που κάνει ορατό το fragment, η onResume() που ξεκινάει την διάδραση και αλληλεπίδραση με τον χρήση, η onPause() για να σταματήσει τη διάδραση ενώ όμως το fragment παραμένει ορατό στην οθόνη, η onStop() που σαν πλέον σταματάει τη διεπαφή από το να είναι ορατή, η onDestoyView() που σκοτώνει τα αντικείμενα που σχετίζονται με το fragment, η onDestroy() που το καταστρέφει και η onDetach() που ακυρώνει τη συσχέτιση του fragment με το σχετικό activity.[100]



Σχήμα 3.39: Κύκλος ζωής ενός Fragment.

3.6.8.3 Android Manifest

Το AndroidManifest.xml βρίσκεται στην ρίζα όλων των αρχείων και φακέλων ενός APK, περιέχει τις πιο σημαντικές πληροφορίες μιας Android εφαρμογής και είναι απαραίτητο, τόσο για την ίδια την εφαρμογή, το λειτουργικό και το Google Play που θα χρησιμοποιηθεί για να διανεμηθεί το υπογεγραμμένο APK.

Μερικές από τις πληροφορίες που βρίσκονται σε αυτό το αρχείο είναι το όνομα της εφαρμογής, το οποίο χρησιμοποιείται για να εντοπιστούν τα υπόλοιπα αρχεία που συνθέτουν το πρόγραμμα, μαζί και ο μοναδικός κωδικός ID που το ξεχωρίζει. Κάθε συστατικό της εφαρμογής δηλώνεται στο αρχείο με τις αντίστοιχες σημασιολογικές έννοιες που το περικλείουν.

```
<manifest ... >
    <application ... >
        <activity android:name="com.example.myapp.MainActivity" ... >
        </activity>
    </application>
</manifest>
```

Σχήμα 3.40: Παράδειγμα δήλωσης Activity στο AndroidManifest.xml.

3.7 GitHub

Το GitHub είναι μια πλατφόρμα υποστήριξης και φιλοξενίας λογισμικού, η οποία επιτρέπει τη συνεργασία μεταξύ προγραμματιστών, την αποθήκευση κώδικα και τον έλεγχο έκδοσης των προγραμμάτων που αποθηκεύονται. Το ηλεκτρονικό αυτό αποθετήριο δίνει τη δυνατότητα να μπορούν τα Project να μοιράζονται με το κοινό η να κρατιούνται ιδιωτικά αλλά και να οργανώνεται καλύτερα μια ομάδα που δουλεύει ταυτόχρονα σε κοινά αρχεία. [101]

Στο GitHub έχουμε αποθήκευση τα αρχεία της εργασίας μας, τον πηγαίο κώδικα, διάφορες εικόνες ή αρχεία ήχου και το χρησιμοποιούσαμε για να συγχρονίζουμε μεταξύ μας τη δουλειά μας, να αποτρέψουμε διάφορα λάθη ή αστοχίες και να βελτιστοποιήσουμε την πρόοδο μας εξ αποστάσεως.

3.8 Java

Η Java ξεκίνησε από την Sun Microsystems το 1991 από την ανάγκη να καλύψει κενά που άφηνε η γλώσσα C++. Ο James Gosling, στα πειράματα του, κατάφερε να υιοθετήσει αρκετές λειτουργίες της γλώσσας C++ και σε συνδυασμό με μερικές νέες λειτουργίες, η Java είναι ικανή να προγραμματίζει με επιτυχία μικροσυσκευές.

Τα βασικά χαρακτηριστικά της γλώσσας είναι η απλότητα της σε σύγκριση με την C++, η οποία είχε μια αρκετά περίπλοκη λογική με τη χρήση των pointers, οι οποίοι έχουν πλέον καταργηθεί. Με τη Java, η διαδικασία παραγωγής εκτελέσιμου κώδικα γίνεται με τη χρήση του μεταγλωττιστή και του ερμηνευτή, οι οποίοι μετατρέπουν τη γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής και έπειτα, διενεργούν εκτέλεση σε μια εικονική μηχανή (JVM).

Αυτό που διαφοροποιεί την Java από τις υπόλοιπες γλώσσες προγραμματισμού της γενιάς της είναι ότι υποστηρίζει αμιγώς τον αντικειμενοστραφή προγραμματισμό, δηλαδή τη δημιουργία αυτόνομων οντοτήτων με δική της ταυτότητα και χαρακτηριστικά. Οι οντότητες αυτές λέγονται αντικείμενα, τα οποία καθορίζονται από κλάσεις (τύπους δεδομένων), οι οποίες προδιαγράφουν τη δομή των δεδομένων και τις διαδικασίες που επιδρούν πάνω σε αυτά.

Άλλες υποστηρικτικές λειτουργίες της Java είναι η πολυνηματική επεξεργασία (multi-threading), η ενσωμάτωση πολυμέσων, η αυτόματη διαχείριση μνήμης, η δυναμική διασύνδεση των δεδομένων, η καταλληλόλητα της για διάφορες εφαρμογές όπως web ή android και το υψηλό επίπεδο ασφαλείας που προσφέρει.

Μία απαίτηση που έχει είναι η σωστή χρήση των τύπων και των εντολών, μιας και κάνει αυστηρό έλεγχο πάνω σε αυτούς κατά τη διάρκεια της μεταγλώττισης, ωστόσο αυτό διευκολύνει τον προγραμματιστή, μιας και άλλη μια δυνατότητα της Java είναι η μεταφερσιμότητα της, κάνοντας ικανή τη χρήση ενός μεταγλωττισμένου αρχείου από ένα λειτουργικό σύστημα (πχ. Windows) να τρέχει σε κάποιο άλλο χωρίς πρόβλημα (πχ. Unix). [102]

3.9 Python

Η Python δημιουργήθηκε από τον Guido van Rossum το 1989 στο Κέντρο Μαθηματικών και Πληροφορικής (Centrum Wiskunde & Informatica) της Ολλανδίας και πήρε το όνομα μιας ομάδας παραγωγής τηλεοπτικών σειρών και ταινιών, τους Monty Python. Είναι μία γλώσσα ανοικτού λογισμικού, η οποία χρησιμοποιεί την αγγλική γλώσσα για τις ορολογίες της αποτελώντας μία από τις πιο εύκολες γλώσσες προγραμματισμού για δημιουργία προγραμμάτων αλλά και εκπαίδευση νέων προγραμματιστών. Είναι φορητή για να τρέχει σε διαφορετικά περιβάλλοντα και συστήματα. Η Python είναι διερμηνευόμενη, αλληλεπιδραστική και αντικειμενοστραφείς γλώσσα γενικής χρήσεως. Αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί για τη δημιουργία προγραμμάτων (scripts) διαχείρισης εργασιών ενός λειτουργικού συστήματος, εφαρμογές δικτύου, αλλά και σε εφαρμογές όπως η τεχνητή μάθηση. [103]

3.10 Google Collab

Το Colaboratory, ή εν συντομία "Colab", είναι ένα προϊόν της Google Research. Το Colab επιτρέπει σε οποιονδήποτε να γράφει και να εκτελεί αυθαίρετο κώδικα python μέσω του προγράμματος περιήγησης και είναι ιδιαίτερα κατάλληλο για μηχανική μάθηση, ανάλυση δεδομένων και εκπαίδευση. Πιο τεχνικά, το Colab είναι μια φιλοξενούμενη υπηρεσία σημειωματάριου Jupyter που δεν απαιτεί καμία εγκατάσταση για να χρησιμοποιηθεί, ενώ παρέχει δωρεάν πρόσβαση σε υπολογιστικούς πόρους, συμπεριλαμβανομένων των GPU.

Κεφάλαιο 4ο: **Ανάλυση Εφαρμογής**

4.1 Σύνολα Δεδομένων Ανίχνευσης και Αναγνώρισης Αντικειμένων

Με την ανάπτυξη και την εφαρμογή της Βαθιάς Μάθησης στην Υπολογιστική Όραση (Computer Vision), η ανίχνευση και αναγνώριση αντικειμένων έχει μελετηθεί εκτενώς σε σύνολα δεδομένων γενικής κατανόησης σκηνών. Όσον αφορά τη λεπτομερή ανίχνευση και ταξινόμηση, υπάρχουν επίσης σύνολα δεδομένων που επικεντρώνονται σε γενικές ιεραρχικές κλάσεις αντικειμένων ή σε ειδικότερα σύνολα δεδομένων.

Στην παρούσα εργασία επικεντρωνόμαστε στην ανίχνευση και τη ταξινόμηση πινακίδων κυκλοφορίας. Οι πινακίδες κυκλοφορίας είναι βασικά στοιχεία του οδικού δικτύου για τον έλεγχο της κυκλοφορίας και για την οδική ασφάλεια. Ως κατηγορία αντικειμένων, έχουν συγκεκριμένα χαρακτηριστικά στην εμφάνιση τους. Πρώτα από όλα οι πινακίδες κυκλοφορίας είναι επίπεδες και άκαμπτες. Έπειτα, σχεδιάζονται με σκοπό να είναι διακριτές από το γύρω περιβάλλον τους. Για παράδειγμα, στις ευρωπαϊκές χώρες όπου ανήκει και η Ελλάδα, οι ρυθμιστικές πινακίδες είναι συνήθως κυκλικές με κόκκινο περίγραμμα. Τα προαναφερθέντα χαρακτηριστικά περιορίζουν σε ένα βαθμό την ποικιλομορφία και αυξάνουν την διακριτότητα των πινακίδων κυκλοφορίας.

Όπως προκύπτει, η ανίχνευση και ταξινόμηση των πινακίδων εξακολουθεί να αποτελεί ένα πολύ δύσκολο πρόβλημα για τους εξής λόγους: α) οι πινακίδες κυκλοφορίας συγχέονται εύκολα με άλλες κατηγορίες αντικειμένων όπως είναι διαφημιστικές πινακίδες, επιγραφές LED κ.α., β) η αντανάκλαση, οι συνθήκες χαμηλού φωτισμού, οι φθορές και καιρικά φαινόμενα όπως βροχή και ομίχλη επηρεάζουν την απόδοση της ταξινόμησης και γ) η πλειονότητα των πινακίδων κυκλοφορίας όταν εμφανίζονται σε εικόνες επίπεδου δρόμου είναι σχετικά μικρές σε μέγεθος.

Ως προσπάθεια άμβλυνσης του προβλήματος αυτού παρουσιάζουμε τη χρήση δυο ξεχωριστών συνόλων δεδομένων:

- Ένα σύνολο δεδομένων για την ανίχνευση των πινακίδων κυκλοφορίας μέσα από εικόνες.
- Ένα σύνολο δεδομένων για την αναγνώριση – ταξινόμηση των ανιχνευμένων πινακίδων.

4.1.1 Σύνολο Δεδομένων Ανίχνευσης Αντικειμένων (Object Detection Dataset)

Σε αυτήν την ενότητα παρουσιάζουμε ένα σύνολο δεδομένων για την ανίχνευση των πινακίδων κυκλοφορίας. Για τον σκοπό αυτό, επιλέχθηκε το σύνολο δεδομένων Mapillary Traffic Sign (MTSD) [104], το οποίο αποτελείται από 100.000 εικόνες υψηλής ανάλυσης από όλο τον κόσμο. Το MTSD παρέχει ομοιομορφία και ποικιλομορφία, καθώς έχει ομοιόμορφη γεωγραφική κατανομή των εικόνων από όλον τον κόσμο και έχει εικόνες διαφορετικής ποιότητας, οι οποίες έχουν ληφθεί υπό διαφορετικές συνθήκες (καιρός και ώρα της ημέρας, διαφορετικοί αισθητήρες και σημεία θέασης των καμερών).

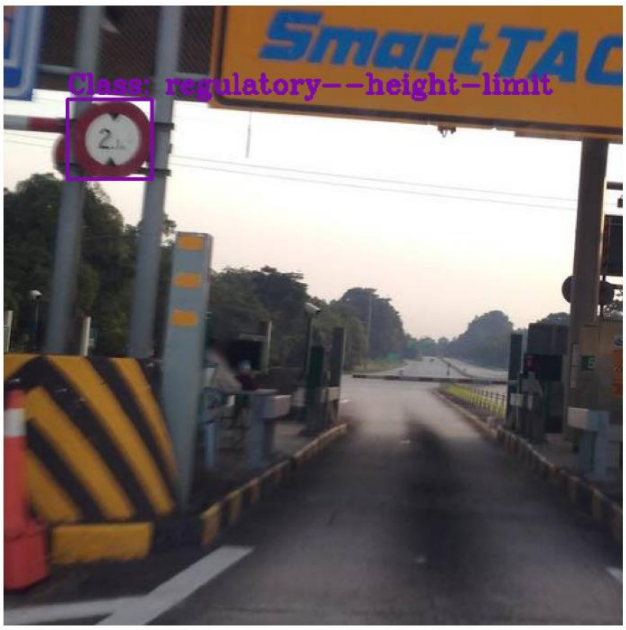
Οι 52.000 από αυτές τις εικόνες είναι πλήρως επισημασμένες από ανθρώπους, με περισσότερες από 320.000 επισημασμένες πινακίδες κυκλοφορίας (πλαίσια οριοθέτησης) σε όλες τις εικόνες. Το MTSD καλύπτει περισσότερα από 300 διαφορετικά σήματα – κλάσεις.

Σχήμα 4.1: Παραδείγματα επισημασμένων εικόνων από το σύνολο δεδομένων εκπαίδευσης του MTSD, που καλύπτουν διαφορετικές συνθήκες φωτισμού και καιρού. Στην κορυφή βλέπουμε μια επισκόπηση όλων των σημάτων – κλάσεων που καλύπτονται στο MTSD.

Προκειμένου επίσης να υπάρχει ποικιλομορφία στα σήματα, τα δείγματα των εικόνων συλλέχθηκαν από όλες τις ηπείρους με κατανομή 20% από την Ευρώπη, τη Βόρεια Αμερική και την Ασία αντίστοιχα, 15% από τη Νότια Αμερική και την Ωκεανία αντίστοιχα, και 10% από την Αφρική.

Επιπλέον, σε αντίθεση με άλλα σύνολα δεδομένων, το MTSD περιέχει πλαίσια οριοθέτησης για όλους τους τύπους πινακίδων που σχετίζονται με την κυκλοφορία, συμπεριλαμβανομένων πινακίδων κατεύθυνσης, πληροφοριών, πινακίδων αυτοκινητοδρόμων κ.λπ. Οι επισημάνσεις για κάθε εικόνα είναι αποθηκευμένες σε αρχεία μορφής *JSON* με τα ακόλουθα κλειδιά:

- **width**: Το πλάτος της αντίστοιχης εικόνας.
- **height**: Το ύψος της αντίστοιχης εικόνας.
- **ispano**: Μια δυαδική τιμή που δείχνει αν η εικόνα είναι πανοραμική. Τα πανοράματα αποθηκεύονται ως τυπικές εικόνες με ισογωνική προβολή και μπορούν να φορτωθούν ως οποιαδήποτε εικόνα.
- **objects**: Λίστα με τις πινακίδες κυκλοφορίας στην εικόνα. Κάθε αντικείμενο αποτελείται από τα ακόλουθα κλειδιά:
    - **bbox**: Προσδιορισμός του πλαισίου οριοθέτησης της πινακίδας κυκλοφορίας εντός της εικόνας.
    - **key**: Ένα μοναδικό αναγνωριστικό της πινακίδας κυκλοφορίας.
    - **label**: Η κατηγορία της πινακίδας κυκλοφορίας.
    - **properties**: Ένα λεξικό που ορίζει τις ειδικές ιδιότητες της πινακίδας.




```
width:                          4160
height:                         3120
ispano:                         false
objects:
  0:
    key:                        "cj3jihnkk1536l51w0a2p7"
    label:                      "regulatory--height-limit--g1"
    bbox:
      xmin:                     1052.1875
      ymin:                     1131.15234375
      xmax:                     1273.59375
      ymax:                     1340.625
    properties:
      barrier:                  false
      occluded:                 false
      out-of-frame:             false
      exterior:                 false
      ambiguous:                false
      included:                 false
      direction-or-information: false
      highway:                  false
      dummy:                    false
```


Σχήμα 4.2: Παράδειγμα εικόνας αριστερά με το πλαίσιο οριοθέτησης (μωβ τετράγωνο) και την ονομασία της κλάσης. Από δεξιά δίνεται ο αντίστοιχος κώδικας για την επισήμανση του πλαισίου οριοθέτησης σε αρχείο μορφής .json.

Τέλος το σύνολο δεδομένων είναι χωρισμένο στις 3 ακόλουθες κατηγορίες:

- Σύνολο εκπαίδευσης 36.589 εικόνων.
- Σύνολο επικύρωσης 5.320 εικόνων.
- Σύνολο δοκιμής 10.544 εικόνων.

4.1.2 Σύνολο Δεδομένων Αναγνώρισης Αντικειμένων (Object Recognition Dataset)

Στην υποενότητα αυτή θα παρουσιάσουμε το σύνολο δεδομένων αναγνώρισης αντικειμένων – ταξινόμησης των πινακίδων κυκλοφορίας. Υπάρχουν διαθέσιμα αρκετά σύνολα δεδομένων, όπως είναι το German Traffic Sign Recognition Benchmark (GTSRB) [105], [106], το LISA [107], το KUL [108], το Stereopolis [109], [110], τα MASTIF από το 2009 έως το 2011 [111], το Russian Traffic Sign Dataset (RTSD) [112], το RUG [113], τα STS [114] και το Ευρωπαϊκό [115] που αποτελεί συλλογή των GTSRB, KUL, Stereopolis, MASTIF, RUG και STS, με μερικές τροποποιήσεις.

Πίνακας 4.1: Πίνακας με τα διαθέσιμα σύνολα δεδομένων ταξινόμησης πινακίδων κυκλοφορίας.

Σύνολο δεδομένων	**Χώρα**	**Κλάσεις**	**Αριθμός εικόνων**	
			Εκπαίδευσης	**Δοκιμής**
LISATSD	Η.Π.Α	47	7.855	
KUL	Βέλγιο	62	4.561	2.528
Stereopolis	Γαλλία	10	203	68
GTSRB	Γερμανία	43	39.209	12.630
MASTIF-2009	Κροατία	97	4.568	1.843
MASTIF-2010		88	3.694	1.490
MASTIF-2011		53	1.037	389
RUG	Ολλανδία	3	58	17

RTSD	Ρωσία	156	104.358	
STS-Set1	Σουηδία	19	2.092	829
STS-Set2		19	2.409	1.033
ETSD	Ευρωπαϊκό	164	82.476	

Μετά από έρευνα, αποφασίσαμε πως η χρήση ενός από τα παραπάνω σύνολα δεδομένων δεν θα αποτελούσε την κατάλληλη επιλογή για διαφορετικούς λόγους το καθένα:

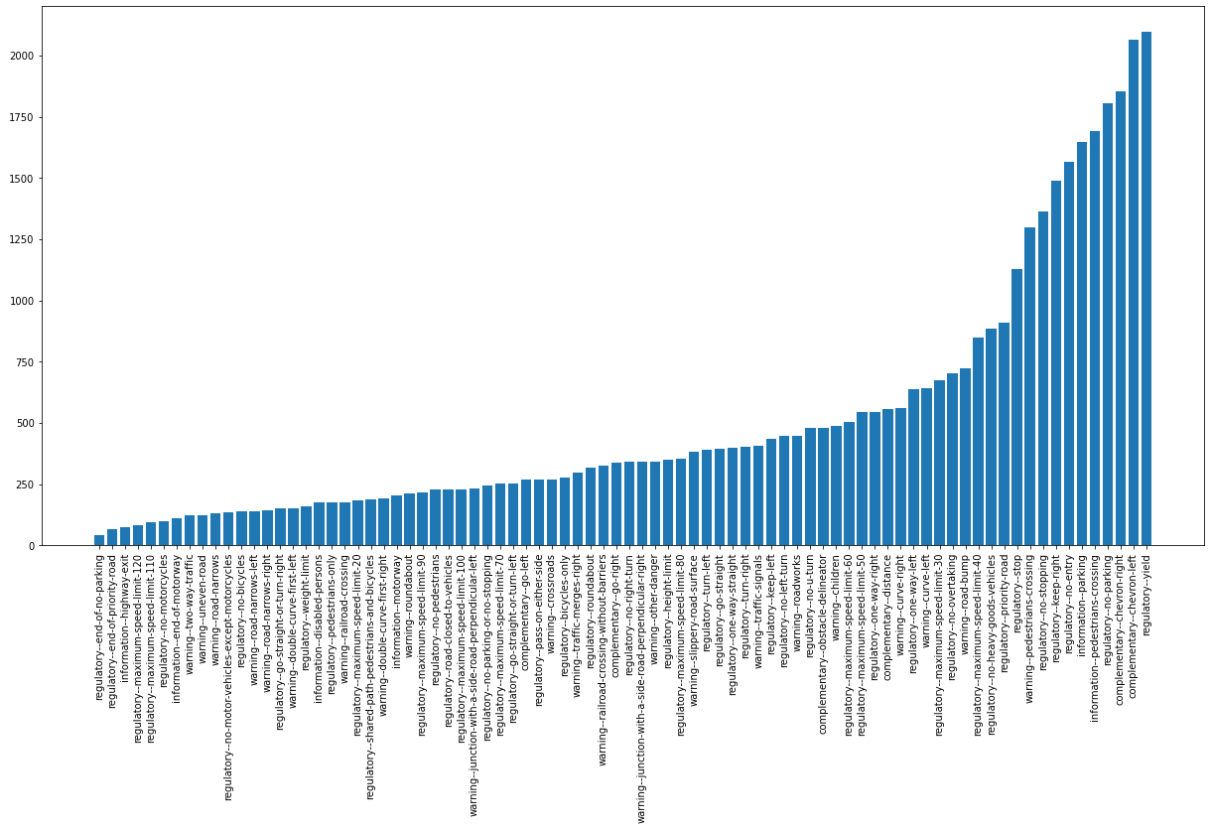
- Το σύνολο δεδομένων των Η.Π.Α δεν μπορεί να χρησιμοποιηθεί καθώς τα σήματα του διαφέρουν αρκετά από αυτά της Ευρώπης και της Ελλάδας.
- Τα σύνολα δεδομένων της Γαλλίας, της Ολλανδίας και της Σουηδίας έχουν περιορισμένο αριθμό κλάσεων που δεν είναι επιθυμητό και καθιστούν αδύνατη τη μέτρηση της γενίκευσης του αλγορίθμου ταξινόμησης.
- Παράλληλα, τα σύνολα δεδομένων του Βελγίου και της Κροατίας έχουν λίγες εικόνες για την εκπαίδευση του μοντέλου ταξινόμησης.
- Το σύνολο δεδομένων της Γερμανίας ενώ έχει ικανοποιητικό αριθμό κλάσεων και εικόνων, θεωρούμε πως δεν έχει τον επιθυμητό αριθμό κλάσεων και πως λείπουν μερικά σημαντικά και συχνά εμφανιζόμενα σήματα.
- Αντίθετα, τα σύνολα δεδομένων της Ρωσίας και της Ευρώπης είναι αρκετά μεγάλα και με πολλές κλάσεις, γεγονός που θα επηρέαζε την απόδοση της εφαρμογής σε πραγματικό χρόνο.

Οπότε καταλήξαμε στη δημιουργία του δικού μας συνόλου δεδομένων, εξάγοντας από τις εικόνες του Mapillary τα αποκόμματα των σημάτων κυκλοφορίας που περιέχονται σε κάθε μια εικόνα.

Όπως αναφέραμε, το Mapillary παρέχει τις επισημάνσεις για κάθε εικόνα σε αρχεία μορφής *JSON*, όπου στο καθένα περιέχονται τα κλειδιά για τις διαστάσεις του πλαισίου οριοθετήσεις κάθε πινακίδας και η κλάση στην οποία ανήκει. Με βάση τις 4 τιμές (*xmin, ymin, xmax, ymax*) των πλαισίων οριοθέτησης, δημιουργήσαμε νέες εικόνες διαστάσεων *(xmax- xmin)×(ymax- ymin*) με το απόκομμα του συγκεκριμένου ορθογωνίου από την αντίστοιχη αρχική εικόνα.

Επίσης, για κάθε κλάση του συνόλου Mapillary μετρήθηκαν πόσες επισημάνσεις αντιστοιχούν στην κάθε μια και κρατήσαμε όσες κλάσεις είχαν περισσότερες από 200. Συνολικά υπήρχαν 95 κλάσεις με λιγότερες από 200 επισημάνσεις, όμως προστέθηκαν μερικές κλάσεις από αυτές στο τελικό σύνολο δεδομένων, καθώς θεωρήθηκαν σημαντικές ώστε να μην συμπεριληφθούν.

Συνεπώς, το σύνολο δεδομένων για την αναγνώριση αποτελείται από 82 κλάσεις με 41.360 εικόνες για την εκπαίδευση και 581 για τη δοκιμή. Παρακάτω παρουσιάζεται ένα ραβδόγραμμα με τις ονομασίες των κλάσεων και τον αριθμό των εικόνων από τον οποίο αποτελείται η κάθε μια.



Σχήμα 4.3: Ραβδόγραμμα με τις ονομασίες των κλάσεων και τον αριθμό των εικόνων από τον οποίο αποτελείται η κάθε μια. Ο οριζόντιος άξονας δείχνει τον αριθμό των κλάσεων με την ονομασία της κάθε μιας και ο κάθετος άξονας τον αριθμό των εικόνων κάθε κλάσης.

4.2 Αναγνώριση πλαισίου οριοθέτησης πινακίδας με YOLOv5

4.2.2 Εισαγωγή

Για τη δημιουργία της εφαρμογής χρησιμοποιήσαμε διάφορες τεχνολογίες, κάποιες από αυτές βρίσκονται στη αιχμή τους, ενώ άλλες είναι αρκετά εδραιωμένες στο χώρο του προγραμματισμού.

Χωρίσαμε την υλοποίηση της εφαρμογής σε 3 βασικά βήματα. Το πρώτο από αυτά είναι να βρούμε ένα τρόπο να αναγνωρίζουμε αν υπάρχουν στο δρόμο πινακίδες όσο το δυνατόν μεγαλύτερο ποσοστό επιτυχίας. Αυτό έγινε με τη χρήση του “Traffic Sign” συνόλου δεδομένων της ομάδας του Mapillary, το οποίο μας βοήθησε να τροφοδοτήσουμε τον αλγόριθμο του YOLOv5 με περίπου 40.000 εικόνες τραβηγμένες από ευρωπαϊκούς δρόμους σε διάφορες συνθήκες φωτισμού, απόστασης και ευκρίνειας. Έπειτα χρησιμοποιήσαμε το ίδιο dataset, για να γράψουμε τον δικό μας ταξινομητή, ο οποίο αναγνωρίζει πάνω από 80 σήματα οδικής κυκλοφορίας. Στο τρίτο στάδιο, συνδυάσαμε τους παραπάνω αλγορίθμους και δημιουργήσαμε μία Android εφαρμογή, έτσι ώστε οι εικόνες που τραβάμε από την κάμερα της συσκευής να γίνεται είσοδος στον αλγόριθμο του YOLOv5, ο οποίος με τη σειρά του αναγνωρίζει και οριοθετεί πλαίσια πινακίδων και τα στέλνει στον ταξινομητή. Από εκεί, η έξοδος του ταξινομητή επιστρέφει στον κύριο ελεγκτή της διεργασίας της Android εφαρμογής και ζωγραφίζεται ένα πλαίσιο στον καμβά (την οθόνη) του χρήστη, με την πινακίδα που αναγνώρισαν οι αλγόριθμοι σε ένα πλαίσιο και μαζί το όνομα της πινακίδας με την πιθανότητα επιτυχίας.

Επιπλέον, η εφαρμογή παίρνει μετρικά από τον χρόνο που διαρκεί η αναγνώριση πινακίδας, την ταχύτητα του χρήστη εκείνη τη στιγμή και με τη χρήση του ηχείου του κινητού στέλνει φωνητικά

μηνύματα τα οποία καθοδηγούν το χρήστη με κάποιες προτεινόμενες ενέργειες, ή τον ενημερώνουν για τους τρέχοντες κανόνες που ισχύουν στον δρόμο σύμφωνα με τον κώδικα οδικής κυκλοφορίας.

4.2.3 Προετοιμασία Dataset

Για να γίνει σωστά η εκπαίδευση τη αλγορίθμου, πρέπει πρώτα να έχουμε προετοιμάσει σωστά το σύνολο δεδομένων. Οι απαιτήσεις που έπρεπε να καλύψουμε είναι το μέγεθος των εικόνων να είναι έχει αναλογία 1:1, οι πληροφορίες των ετικετών να βρίσκονται σε μορφή txt και τα αρχεία να έχουν ταξινομηθεί κατάλληλα.

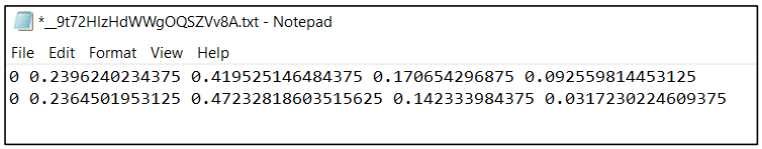
4.2.4 Επεξεργασία εικόνων

Η πληθώρα των εικόνων του Dataset του Mapillary περιέχει εικόνες σημάτων σε υψηλή ανάλυση διάφορων διαστάσεων. Μία απαίτηση του YOLOv5 είναι όλες οι εικόνες που θα δοθούν στην είσοδο να έχουν τις ίδιες διαστάσεις ύψους και πλάτους. Ένα από τα προβλήματα που χρειάστηκε να λύσουμε πρώτού προχωρήσουμε στην εκπαίδευση, ήταν η αλλαγή μεγέθους των εικόνων και η μείωση της ανάλυσης στο τελικό αποτέλεσμα για να μειώσουμε τον χώρο αποθήκευσης στον διακομιστή που χρησιμοποιήσαμε και να ελαττώσουμε τον χρόνο επεξεργασίας και τη μνήμη που θα χρειαζόταν ο αλγόριθμος για κάθε εικόνα κατά τη διάρκεια της εκπαίδευσης.

Χρησιμοποιήσαμε τη βιβλιοθήκη Pillow (PIL) στην Python, η οποία παρέχει μέσω της κλάσης Image τη δυνατότητα επεξεργασίας εικόνων η οποία κάλυπτε την ανάγκη μας. Η πρώτη αλλαγή ήταν να μετατρέψουμε τις εικόνες σε μέγεθος $640px * 640px$. Η μετατροπή έγινε κρατώντας ίδια την αναλογία διαστάσεων μέχρι να φτάσει η μεγαλύτερη από τις δύο διαστάσεις στο επιθυμητό μέγεθος και προσθέτοντας στην άλλη συμπλήρωμα μαύρων πλαισίων. Το μέγεθος είναι το προτεινόμενο από την ομάδα του YOLO και ήταν το ελάχιστο για να μπορούν όλα τα αντικείμενα που θα αναγνωρίζαμε να έχουν μέγεθος μεγαλύτερο από 3pixels. Αποθηκεύσαμε τις εικόνες σε JPEG, έτσι ώστε μέσω του αλγορίθμου συμπίεσης να κερδίσουμε χωρητικότητα χωρίς να υπάρχει σημαντική αλλαγή στο αποτέλεσμα τις εικόνας, παρά τη μείωση της πληροφορίας. Όλες οι φωτογραφίες αποθηκεύτηκαν στο φάκελο “train”, ο οποίος θα χρησιμοποιηθεί κατά τη διάρκεια της εκπαίδευσης.

4.2.5 Επεξεργασία ετικετών

Το YOLOv5 απαιτεί οι ετικέτες του να βρίσκονται σε ένα αρχείο txt ίδιου ονόματος με την εικόνα που αντιστοιχεί και μέσα σε ξεχωριστές σειρές κάθε αντικείμενο που υπάρχει για αναγνώριση. Οι τιμές που αποθηκεύονται είναι η κλάση που ανήκει το αντικείμενο, η τοποθεσία του πλαισίου μέσα στην εικόνα και το μέγεθος του πλαισίου.



Σχήμα 4.4: Παράδειγμα αρχείου ετικετών.

Οι πληροφορίες που φαίνονται στην παραπάνω εικόνα με τη σειρά είναι η κλάση, η οποία επειδή ταξινομούμε μεταξύ μιας κλάσης (δυαδική ταξινόμηση) παίρνει πάντα την τιμή μηδέν. Η επόμενη τιμή είναι η θέση του πλαισίου στον άξονα x και μετά η θέση του στον άξονα y. Οι τελευταίες δύο τιμές είναι το συνολικό πλάτος (width) και ύψος (height) του πλαισίου αντίστοιχα.

Τα στοιχεία των εικόνων που πήραμε από το Mapillary, δίνονται μέσω των αρχείων τύπου json, μια μορφή δομής αποθήκευσης δεδομένων Javascript, η οποία χρησιμοποιείται για την ανταλλαγή και αποθήκευση δεδομένων σε διάφορες διαδικτυακές εφαρμογές. Για την εξαγωγή των στοιχείων χρησιμοποιήσαμε κώδικα python, όπου διαβάσαμε και μετατρέψαμε τις πληροφορίες που χρειαζόμαστε για την εκπαίδευση, στη μορφή των δύο σημείων για την τοποθεσία του πλαισίου και το μέγεθός του. Αντίστοιχα υπολογίσαμε τις τιμές των πλαισίων οι οποίες άλλαξαν όταν αλλάξαμε και το μέγεθος των εικόνων για να κρατήσουμε σωστή την αναλογία.

4.2.6 Εκπαίδευση YOLOv5s

Για να γίνει η εκπαίδευση χρειάζεται μια ένα υπολογιστικό σύστημα και μια κάρτα γραφικών η οποία τρέχει Cuda της Nvidia και μπορεί να ανταπεξέλθει σε πολύωρη χρήση λόγο της φύσης της εκπαίδευσης. Χρησιμοποιήσαμε μία κάρτα γραφικών TITAN Xp της Nvidia με 12 GB μνήμης, η οποία μας έδωσε τη δυνατότητα να τρέξουμε αρκετά πειράματα, μαζί και την τελική εκπαίδευση του μοντέλου που επιλέξαμε σε λειτουργικό σύστημα Linux με Python 3.8.3 Anaconda.

Πρώτα χρειάστηκε να κατεβάσουμε το τη βιβλιοθήκη του YOLOv5 της ομάδας των Ultralitics μέσω του GitHub. Η έκδοση που επιλέξαμε ήταν η πιο πρόσφατη. Αφού είχαμε ετοιμάσει τις εικόνες σε ανάλυση 640×640 pixel, της φορτώσαμε στον αλγόριθμο για να ξεκινήσει η εκπαίδευση.

$$!\,python\ train.py\ --img\ 640\ --batch\ 40\ --epochs\ 3000\ --data\ '../../../Dataset-640v4/data.yaml'\ --weights\ ''\ --cfg\ yolov5s.yaml\ --name\ 640v4$$

Για να τρέξουμε την εκπαίδευση του αλγορίθμου YOLOv5, κάνουμε κλήση στο βασικό αρχείο $train.py$ και περνάμε τις απαραίτητες τιμές στις παραμέτρους που χρειάζεται για την εκκίνηση.

4.2.7 Βασικές παράμετροι εκπαίδευσης

$-img$: Με την παράμετρο $-img$ δηλώνουμε το μέγεθος των εικόνων τις οποίες θα διαβάσει ο YOLOv5, το οποίο είναι πολύ σημαντικό να είναι ίδιου μήκους και πλάτους, για να διατηρηθούν σωστά τα πλαίσια και οι ετικέτες που θα βρεθούν.

$-batch$: Έπειτα το μέγεθος του batch, με το οποίο θέτουμε πόσα πακέτα από φωτογραφίες θα χρησιμοποιούνται ξεχωριστά για κάθε εποχή με το $-batch$, όπου δώσαμε την τιμή 40, για να διαιρέσουμε το πλήθος όλων των εικόνων έτσι ώστε να εκμεταλλευτούμε όσο το δυνατόν περισσότερο τα 12GB της μνήμης που είχαμε στη διάθεση μας.

$-epochs$: Κατά τη διάρκεια των πειραμάτων με διαφορετικές ρυθμίσεις και διάφορες παραλλαγές στα δεδομένα μας, είδαμε ότι η πορεία της μάθησης του μοντέλου αρχίζει να παίρνει μία οριζόντια φορά, ένα φαινόμενο που ονομάζεται plateau, και σημαίνει πως οι εξέλιξη του αλγορίθμου γίνεται με πολύ μικρότερα βήματα, χωρίς μεγάλη διαφορά στα αποτελέσματα. Αυτό το σημείο το φτάσαμε στις 500 επαναλήψεις, όπου είναι και ο καταληκτικών αριθμός των εποχών που επιλέξαμε.

$-data$: Σε αυτή την παράμετρο, ορίζουμε το αρχείο τύπου $yaml$ όπου προσδιορίζουμε βασικές πληροφορίες όπως τους φακέλους με τις εικόνες για εκπαίδευση, τις ετικέτες επαλήθευσης και τις κλάσεις που χωρίζουμε τα δεδομένα. Στη δική μας περίπτωση η ταξινόμηση θεωρείτε δυαδική και ο λόγος είναι πως υπάρχει μία γενική κλάση, της πινακίδας, όπου ο αλγόριθμος όταν βρει ένα αντικείμενο, έχει να διαλέξει μεταξύ του ότι είναι πινακίδα ή όχι και να σχηματίσει το πλαίσιο. Αυτό γίνεται, γιατί ο

σκοπός της χρήσης του YOLOv5 είναι η αναγνώριση πλαισίων ενός είδους αντικειμένου, καθώς η ταξινόμηση θα γίνει από άλλο αλγόριθμο σε επόμενο επίπεδο.

$-weights$: Σε αυτή την παράμετρο η τιμή που δίνεται είναι κενή. Σκοπός αυτής της παραμέτρου είναι τα βάρη και η αρχιτεκτονική του αλγορίθμου, να εκμεταλλευτούν τα προ-εκπαιδευμένα μοντέλα ή μοντέλα που έχουμε εμείς εκπαιδεύσει και να ξεκινήσουν έχοντας ένα προβάδισμα. Η επιλογή να μη δώσουμε κάποια προ-εκπαιδευμένα βάρη στον αλγόριθμο ήταν γιατί η αναγνώριση που θέλαμε να κάνει είναι απλή και τα βάρη του Yolov5 δεν είναι κατάλληλα, καθώς αναγνωρίζουν πιο περίπλοκα αντικείμενα και μέσα σε αυτά δεν ήταν οι πινακίδες.

$-cfg$: Στο σημείο αυτό επιλέγουμε πιο είδος από τα 5 (nano, small, medium, large & extra large) μοντέλο του YOLOv5 επιλέγουμε. Στα μοντέλα τα οποία χρησιμοποιούν μεγάλα dataset, προτείνεται η επιλογή της εκκίνησης με κενά βάρη, όπως ήδη αναλύσαμε, και η χρήση μιας από τις παραπάνω αρχιτεκτονικές του YOLO. Φυσικά δίνεται η δυνατότητα να φτιάξουμε προσαρμοσμένα μοντέλα, μέσω ενός yaml αρχείου, αλλά οι ανάγκες μας καλύφθηκαν από το YOLOv5s, την 2η μικρότερη επιλογή, η οποία κάλυπτε τις απαιτήσεις σε διάφορα επίπεδα που θα δούμε παρακάτω.

$-name$: Η τελευταία μεταβλητή, χρησιμοποιείται για να θέσουμε χειροκίνητα το όνομα του πειράματος, όπου θα αποθηκευτούν όλα τα αρχεία της εκπαίδευσης του μοντέλου, για να τα χρησιμοποιήσουμε αργότερα, να συνεχίσουμε την εκπαίδευση αν υπάρξει κάποια διακοπή ή κάποιο σφάλμα, αλλά και για να κρατάμε ένα αποθετήριο για οποιαδήποτε άλλη χρήση ή σύγκριση.

4.2.8 Επιλογή μοντέλου εκπαίδευσης (YOLOv5s)

Η επιλογή του μοντέλου που χρησιμοποιήσαμε έγινε σε συνδυασμό με πολλές παραμέτρους, ζυγίζοντας τα πλεονεκτήματα και μειονεκτήματα ανάμεσα στις υπόλοιπες εκδόσεις αλλά και με βάση το τελικό αποτέλεσμα που θέλαμε να πετύχουμε.

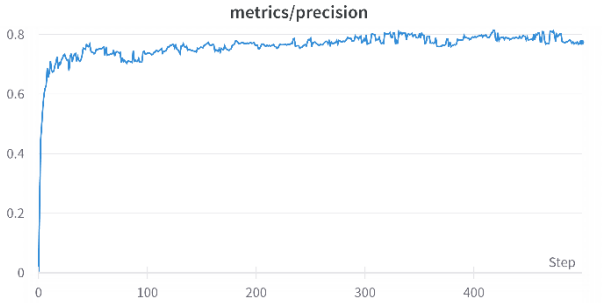
Οι συνθήκες όπου δοκιμάστηκαν τα μοντέλα εκπαίδευσης του YOLOv5 ήταν ίδιες, ωστόσο τα αποτελέσματα διαφέρουν λόγο της πολυπλοκότητας του κάθε μοντέλου, τη διαφορά στους νευρώνες που χρησιμοποιεί και στο μέγεθος των εικόνων που τροφοδοτείται. Για να επιλέξουμε κάποιο από τα μοντέλα, οι παράμετροι που βασιστήκαμε είναι η ταχύτητα εκπαίδευσης, η ταχύτητα του μοντέλου για να βγάλει συμπεράσματα (Inference Time), το ποσοστό επιτυχίας των αναγνωρίσεων και το μέγεθος του τελικού εκπαιδευμένου μοντέλου.

Για να πάρουμε απόφαση ποιο μοντέλο θα χρησιμοποιήσουμε, χρησιμοποιήσαμε τον πίνακα που είδαμε σε προηγούμενο κεφάλαιο, με συγκρίσεις των μοντέλων ως προς την ταχύτητα, την επιτυχία, την πολυπλοκότητα και τα μεγέθη των εικόνων στην είσοδο. Στην περίπτωση μας, το πρόβλημα που θα λύσουμε με τον αλγόριθμο είναι η γρήγορη εύρεση πινακίδων, καθώς η ταξινόμηση θα γίνει σε επόμενο επίπεδο από έναν εξειδικευμένο αλγόριθμο ταξινόμησης. Την περίοδο

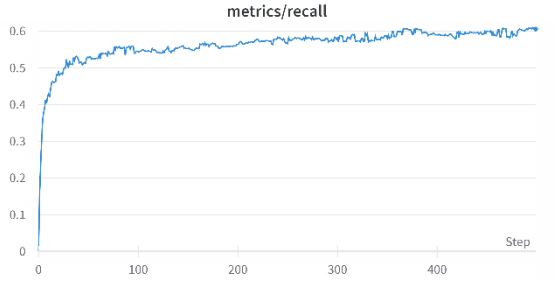
4.2.9 Διαδικασία και αποτελέσματα εκπαίδευσης

Το μοντέλο που επιλέξαμε ήταν το YOLOv5s, το μικρότερο μοντέλο της κατηγορίας, με 500 επαναλήψεις, εκ των οποίων η κάθε μια θα είχε 40 παρτίδες (Batches), με πάνω από 35.000 εικόνες στο σύνολο. Η εκπαίδευση χρειάστηκε περίπου 2,5 ημέρες για να ολοκληρωθεί και τα αποτελέσματα ήταν αρκετά ενθαρρυντικά. Το ποσοστό επιτυχίας της ακρίβειας του αποτελέσματος (Precision) στο σύνολο επαλήθευσης έφτασε το 77% και το ποσοστό ανάκλησης (Recall) στο 60%. Το ποσοστό του μέσου όρου μέσης ακρίβειας (mAP_0.5) με κατώφλι το 0.5, φτάνει το ποσοστό του 67%, και αντίστοιχα με κατώφλι 0.95 (mAP_0.95), φτάνει το 45%, που σημαίνει πως τα πλαίσια των αντικειμένων

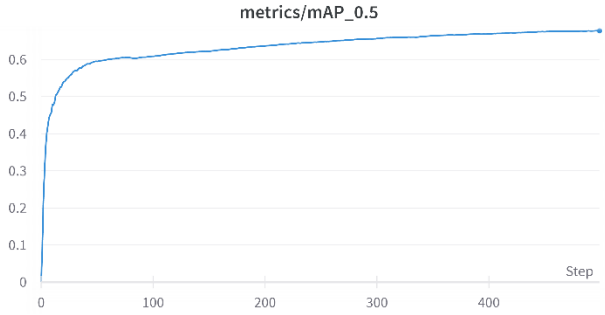
τοποθετούνται στις εικόνες με πολύ καλή ακρίβεια, πράγμα αρκετά καλό για τον μοντέλο μας και πάνω από τις τιμές που βρέθηκαν στα πειράματα των Ultralytics.



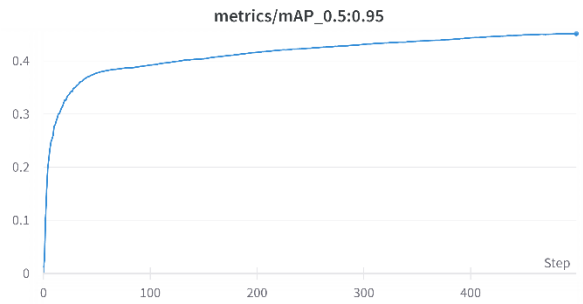
Σχήμα 4.5: Ποσοστό επιτυχών προβλέψεων.



Σχήμα 4.6: Ποσοστό επιτυχών ανακλήσεων.



Σχήμα 4.7: Μέσος όρος μέσης ακρίβειας > 0.5.



Σχήμα 4.8: Μέσος όρος μέσης ακρίβειας > 0.95.

Το συμπέρασμα από τα παραπάνω δεδομένα της εκπαίδευσης (Σχήματα 4.5 – 4.8), είναι πως το μοντέλο μπορεί να καταφέρει με μεγάλη επιτυχία να αναγνωρίζει την ύπαρξη πινακίδων σε δρόμους υπό διάφορες συνθήκες και να ορίζει ένα πλαίσιο γύρο από αυτές, περικλείοντας σωστά όλη την απαραίτητη πληροφορία. Έτσι ολοκληρώνεται το πρώτο βήμα της εφαρμογής αναγνώρισης σημάτων οδικής κυκλοφορίας και σειρά έχει, τα αντικείμενα που ανιχνεύονται από τον αλγόριθμο που εκπαιδεύσαμε, να δοθούν ως είσοδο σε ένα άλλο δίκτυο νευρώνων, έτσι ώστε να ταξινομηθούν σωστά.

Αφού τέλειωσε η εκπαίδευση, η έξοδος του μοντέλου περιείχε δυο αρχεία τύπου PyTorch, το $last.pt$ και το $best.pt$. Το πρώτο είναι το μοντέλο με τα αποτελέσματα τις τελευταίας εκπαίδευσης, που συνήθως χρησιμοποιείται όταν θέλουμε να συνεχίσουμε την εκπαίδευση με περισσότερες εποχές και το δεύτερο αρχείο ήταν αυτό με την καλύτερη απόδοση από όλες, σύμφωνα με το ποσοστό επιτυχίας που είχε στην αναγνώριση του συνόλου επαλήθευσης. Για τη συνέχεια της εφαρμογής, χρειαζόμαστε το μοντέλο με τις καλύτερες επιδόσεις, δηλαδή το best.pt, το οποίο είναι αρκετά βαρύ και ασύμβατο με το Android, οπότε πρέπει να το μετατρέψουμε σε μια πιο κατάλληλη μορφή, η οποία είναι το TensorFlow Lite (TFLite).

4.2.10 Μετατροπή σε TFLite

Για να χρησιμοποιήσουμε το μοντέλο στο Android Studio, πρέπει πρώτα να το μετατρέψουμε σε μορφή είναι συμβατή με το λειτουργικό. Η βιβλιοθήκη του TensorFlow έχει ετοιμάσει έναν converter τον οποίο και χρησιμοποιήσαμε, που μετατρέπει το $best.pt$ σε αρχείο τύπου TFLite. Με τον Intepreter του TensorFlow Lite, μπορούμε να δώσουμε τα στιγμιότυπα που θα λάβουμε από την κάμερα του κινητού για να μας επιστραφεί ένα πολυδιάστατο πίνακα στην έξοδο, ο οποίος περιέχει το πλήθος των αναγνωρίσεων, την τοποθεσία μέσα στην εικόνα, το μέγεθος του πλαισίου και τον αριθμό το όνομα της κλάσης που ανήκει, ο οποίος στη δική μας περίπτωση είναι πάντα μηδέν, με τίτλο «sign».

Για να μετατρέψουμε το αρχείο από PyTorch σε tflite χρησιμοποιήσαμε τον μετατροπέα του του χρήστη $zldrobit$ στο github [116], ο οποίος έχει φτιάξει μία παραλλαγή του αρχείου $export.py$ των Ultralitics, και μετατρέπει το μοντέλο από pytorch σε TensorFlow Lite. Τα στάδια της μετατροπής είναι η αναπαράσταση του αρχικού μοντέλου σε TensorFlow GraphDef, ένα γράφο ροής δεδομένων του TensorFlow και έπειτα σε TensorFlow Lite της μορφής των 16bit κινητής υποδιαστολής. Επίσης, μα δίνει τη δυνατότητα να βελτιστοποιήσουμε το μοντέλο με την τεχνική του κβαντισμού, η οποία μας επιτρέπει να μειώσουμε αρκετά το μέγεθος του τελικού αρχείου tflite.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(1, 640, 640, 3)]	0	
tf__focus (tf_Focus)	(1, 320, 320, 32)	3584	input_1[0][0]
tf__conv_1 (tf_Conv)	(1, 160, 160, 64)	18688	tf__focus[0][0]
tf__c3 (tf_C3)	(1, 160, 160, 64)	19200	tf__conv_1[0][0]
tf__conv_7 (tf_Conv)	(1, 80, 80, 128)	74240	tf__c3[0][0]
tf__c3_1 (tf_C3)	(1, 80, 80, 128)	158208	tf__conv_7[0][0]
tf__conv_17 (tf_Conv)	(1, 40, 40, 256)	295936	tf__c3_1[0][0]
tf__c3_2 (tf_C3)	(1, 40, 40, 256)	627712	tf__conv_17[0][0]
tf__conv_27 (tf_Conv)	(1, 20, 20, 512)	1181696	tf__c3_2[0][0]
tf_spp (tf_SPP)	(1, 20, 20, 512)	658432	tf__conv_27[0][0]
tf__c3_3 (tf_C3)	(1, 20, 20, 512)	1185792	tf_spp[0][0]
tf__conv_35 (tf_Conv)	(1, 20, 20, 256)	132096	tf__c3_3[0][0]
tf__upsample (tf_Upsample)	(1, 40, 40, 256)	0	tf__conv_35[0][0]
tf__concat (tf_Concat)	(1, 40, 40, 512)	0	tf__upsample[0][0] tf__c3_2[0][0]
tf__c3_4 (tf_C3)	(1, 40, 40, 256)	363520	tf__concat[0][0]
tf__conv_41 (tf_Conv)	(1, 40, 40, 128)	33280	tf__c3_4[0][0]
tf__upsample_1 (tf_Upsample)	(1, 80, 80, 128)	0	tf__conv_41[0][0]
tf__concat_1 (tf_Concat)	(1, 80, 80, 256)	0	tf__upsample_1[0][0] tf__c3_1[0][0]
tf__c3_5 (tf_C3)	(1, 80, 80, 128)	91648	tf__concat_1[0][0]
tf__conv_47 (tf_Conv)	(1, 40, 40, 128)	147968	tf__c3_5[0][0]
tf__concat_2 (tf_Concat)	(1, 40, 40, 256)	0	tf__conv_47[0][0] tf__conv_41[0][0]
tf__c3_6 (tf_C3)	(1, 40, 40, 256)	297984	tf__concat_2[0][0]
tf__conv_53 (tf_Conv)	(1, 20, 20, 256)	590848	tf__c3_6[0][0]
tf__concat_3 (tf_Concat)	(1, 20, 20, 512)	0	tf__conv_53[0][0] tf__conv_35[0][0]
tf__c3_7 (tf_C3)	(1, 20, 20, 512)	1185792	tf__concat_3[0][0]
tf__detect (tf_Detect)	((1, 25200, 6), [(1, 16182		tf__c3_5[0][0] tf__c3_6[0][0] tf__c3_7[0][0]

Σχήμα 4.9: Αρχιτεκτονική μοντέλου YOLOv5s.

Τα επίπεδα του παραπάνω σχήματος (Σχήμα 4.9) μετατράπηκαν σε μορφή TFLite, με τελευταίο το νευρώνα της αναγνώρισης (tf_detect) ο οποίος μα δίνει τον πίνακα με τα τελικά αποτελέσματα. Από την παραπάνω εικόνα βλέπουμε ότι από το αρχικό επίπεδο, το οποίο δέχεται ως είσοδο μία εικόνα, μεγέθους 640x640, το μέγεθος δηλαδή με το οποίο εκπαιδεύσαμε τον αλγόριθμο, σε 3 χρωματικά επίπεδα, για το κόκκινο, πράσινο και μπλε (RGB), εξ 'ου και οι διαστάσεις (1,640,640,3). Μετά από την επεξεργασία της εισόδου και εξαγωγή των χαρακτηριστικών, σε κάθε επίπεδο μέχρι το τελευταίο της εξόδου βλέπουμε πως οι παράμετροι αυξάνονται. Αυτό γίνεται, διότι στο μοντέλο του YOLOv5s, υπάρχουν διάφορα κρυφά στρώματα, εκ των οποίων κάποια είναι εκπαιδεύσιμα και κάποια όχι. Στο σύνολο τους, οι παράμετροι είναι 7.082.806, από τους οποίους αυτοί που μπορούν να εκπαιδευτούν καλύτερα ή να συνεχίσουν να βελτιστοποιούνται, δηλαδή μπορούν να επηρεαστούν τα βάρη τους κατά τη διάρκεια της εκπαίδευσης, είναι οι 7.063.542 και μη βελτιστοποιήσιμοι, δηλαδή αυτοί που δεν μπορούμε να τους επηρεάσουμε και μένουν ανέγγιχτοι κατά τη διάρκεια της εκπαίδευσης, είναι 19.264. Στο τέλος έχουμε τον πολυδιάστατο πίνακα της εξόδου, ο οποίος έχει τις διαστάσεις (1,25200,6). Οι τιμές αυτές αντιπροσωπεύουν με τη σειρά που τις βλέπουμε, τον αριθμό τις εξόδου, που είναι ίσος με την είσοδο, τον αριθμό των αγκυρών (anchors) οι οποίες είναι οι διάφοροι συνδυασμοί των χαρακτηριστικών των μοντέλων, με τον κάθε ένα να εξυπηρετεί κάποιο αποτέλεσμα. Τέλος, ο αριθμός έξι, αντιπροσωπεύει τις κλάσεις που έχουμε, στη δική μας περίπτωση μόνο μία, τα 4 σημεία για τη δημιουργία πλαισίου και ο βαθμός αντικειμενικότητας για το κάθε πλαίσιο.

Για να πετύχουμε ακόμη καλύτερα αποτελέσματα, περάσαμε το μοντέλο από κβαντισμό ακεραίων αριθμών(integer quantization). Επιλέξαμε 100 εικόνες από το σύνολο δοκιμών για να τροφοδοτήσουμε τον μετατροπέα και το αποτέλεσμα ήταν να το ένα μοντέλο tflite, που τα βάρη, οι τένσορες, οι είσοδοι και οι έξοδοι του να παίρνουν τιμές ακεραίων αριθμών και το μέγεθος του να είναι περίπου 7,3MB.

Πίνακας 4.2: Αποτελέσματα πριν και μετά τον κβαντισμό.

Μοντέλο	**Τύπος**	**Μέγεθος**	**Μείωση αξιοπιστίας**
Best.pt	SavedModel	13.7 MB	-
Best-fp16.tflite	TFLite – Float16	13.5 MB	< 1%
Best-int8.tflite	TFLite - Quantized	7.3 MB	1-2%

Για να δοκιμάσουμε το ποσοστό επιτυχίας των μοντέλων πριν και μετά τον κβαντισμό, χρησιμοποιήσαμε τον κώδικα των Ultralytics, μέσω του αρχείου $detect.py$, ο οποίος μας επιτρέπει να τεστάρουμε σε τις εικόνες που τις τοποθετήσαμε στη θέση */yolov5/data/images* τα μοντέλα και μετά την εκτέλεση του πειράματος, δημιουργεί τις εικόνες με περασμένα τα πλαίσια με τα αντικείμενα που αναγνωρίστηκαν, την κλάση και το ποσοστό αξιοπιστίας τους.

Τα αποτελέσματα του μοντέλου πριν τον κβαντισμό:

Σχήμα 4.10: Αναγνώριση με best-fp16.tflite.

Τα αποτελέσματα του μοντέλου μετά τον κβαντισμό:

Σχήμα 4.11: Αναγνώριση με best-int8.tflite.

Όπως φαίνεται στα παραπάνω σχήματα (Σχήμα 4.10 – 4.11), το ποσοστό αξιοπιστίας των αποτελεσμάτων έπεσε κατά 1-2 μονάδες, ωστόσο το ποσό είναι αμελητέο, καθώς από τη μετατροπή εξοικονομήσαμε χώρο, λόγο της μείωσης του μεγέθους του αρχείου, το οποίο αρχικά ήταν 13.7 MB και κατέληξε να είναι 46.8% μικρότερο.

Τη στιγμή που γράφεται αυτή η διπλωματική, η ομάδα των Ultralytics έχει ενσωματώσει μέσα στη βιβλιοθήκη του YOLOv5, και την επιλογή για εξαγωγή για TensorFlow Lite κατευθείαν από το export.py. Ωστόσο, με μερικές γραμμές κώδικα, μπορεί εύκολα αυτό το μοντέλο να μετατραπεί σε κβαντισμένο και να έχει το ίδιο αποτέλεσμα.

4.3 Περιγραφή εκπαίδευσης του μοντέλου ταξινόμησης αντικειμένων

4.3.1 Εκπαίδευση μοντέλου ταξινόμησης αντικειμένων

Η εκπαίδευση του μοντέλου ταξινόμησης των πινακίδων κυκλοφορίας υλοποιήθηκε στο Google Colab με χρήση της βιβλιοθήκης Βαθιάς Μάθησης Keras. Η εκπαίδευση έγινε με το σύνολο δεδομένων αναγνώρισης των 82 κλάσεων που δημιουργήσαμε και τη χρήση ενός εξατομικευμένου Νευρωνικού Δικτύου, την αρχιτεκτονική του οποίου θα αναλύσουμε στη συνέχεια.

Πρώτο βήμα πριν την εκπαίδευση είναι η φόρτωση των απαραίτητων βιβλιοθηκών, όπως το Keras για την κατασκευή του μοντέλου, το sklearn για το διαχωρισμό των δεδομένων εκπαίδευσης και δοκιμής, το PIL για τη μετατροπή των εικόνων σε πίνακα αριθμών και άλλες βιβλιοθήκες για υπολογισμούς, δημιουργία πινάκων – τανυστών και τη σχεδίαση διαγραμμάτων, όπως οι Pandas, NumPy, Matplotlib και Tensorflow.

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import tensorflow_hub as hub
import tensorflow as tf
from tensorflow import keras
from PIL import Image
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score
np.random.seed(82)
```

Σχήμα 4.12: Κώδικας φόρτωσης των βιβλιοθηκών σε γλώσσα Python στο Google Colab.

Το επόμενο βήμα είναι η ανάθεση των προορισμών για τους φακέλους με τις εικόνες εκπαίδευσης και δοκιμής του συνόλου δεδομένων, καθώς και ο ορισμός ενός κοινού μεγέθους εικόνων. Έπειτα, γίνεται ανάκτηση των εικόνων με τις ετικέτες τους και εφαρμόζεται η αλλαγή μεγέθους σε 64×64, αφού οι εικόνες πρέπει να έχουν το ίδιο μέγεθος για την αναγνώριση. Παράλληλα, μετατρέπουμε τη λίστα με τις εικόνες σε πίνακα της NumPy λόγω της ταχύτητας υπολογισμών και του κέρδους σε υπολογιστική μνήμη.

```
data_dir = '/content/classification_dataset'
train_path = '/content/classification_dataset/Train'
test_path = '/content/classification_dataset/Test'

# Resizing the images to 64x64x3
IMAGE_SHAPE = (64, 64)

IMG_HEIGHT = 64
IMG_WIDTH = 64
MODEL_INPUT_MEAN = 0
MODEL_INPUT_STD = 255

channels = 3
```

Σχήμα 4.13: Κώδικας ανάθεσης προορισμών, ορισμού κοινού μεγέθους εικόνων και ορισμού τιμών κανονικοποίησης των τελευταίων.

```
TRAINING_DATA_DIR = '/content/classification_dataset/Train'
image_generator = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)
image_data = image_generator.flow_from_directory(str(TRAINING_DATA_DIR), target_size=IMAGE_SHAPE)
```

```
Found 41360 images belonging to 82 classes.
```

```
for image_batch, label_batch in image_data:
  print("Image batch shape: ", image_batch.shape)
  print("Label batch shape: ", label_batch.shape)
  break
```

```
Image batch shape:  (32, 64, 64, 3)
Label batch shape:  (32, 82)
```

```
image_data = []
image_labels = []

for i in range(NUM_CATEGORIES):
    path = data_dir + '/Train/' + str(i)
    images = os.listdir(path)

    for img in images:
        try:
            image = cv2.imread(path + '/' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
            image_data.append(np.array(resize_image))
            image_labels.append(i)
        except:
            print("Error in " + img)

# Changing the list to numpy array
image_data = np.array(image_data)
image_labels = np.array(image_labels)

print(image_data.shape, image_labels.shape)
```

```
(41360, 64, 64, 3) (41360,)
```

Σχήμα 4.14: Κώδικας ανάκτησης των εικόνων εκπαίδευσης με τις ετικέτες τους και μετατροπή σε πίνακα NumPy.

Στη συνέχεια λαμβάνει μέρος ο χωρισμός του συνόλου δεδομένων, σε δεδομένα εκπαίδευσης και επαλήθευσης με ποσοστά 70% και 30% αντίστοιχα. Τα τελευταία χρησιμοποιούνται κατά τη διάρκεια της εκπαίδευσης. Στο σύνολο δεδομένων παρέχονται εικόνες για τη δοκιμή του εκπαιδευμένου μοντέλου, οπότε δεν χρειάζεται ο χωρισμός του συνόλου για τη δοκιμή. Ακόμη, εφαρμόζεται κανονικοποίηση των εικόνων διαιρώντας με το 255, ώστε όλες οι τιμές του πίνακα να είναι μεταξύ -1 και 1.

```
X_train, X_val, y_train, y_val = train_test_split(image_data, image_labels, test_size=0.3, random_state=82, shuffle=True)

X_train = X_train/255
X_val = X_val/255

print("X_train.shape", X_train.shape)
print("X_valid.shape", X_val.shape)
print("y_train.shape", y_train.shape)
print("y_valid.shape", y_val.shape)
```

```
X_train.shape (28952, 64, 64, 3)
X_valid.shape (12408, 64, 64, 3)
y_train.shape (28952,)
y_valid.shape (12408,)
```

Σχήμα 4.15: Κώδικας χωρισμού του συνόλου δεδομένων για εκπαίδευση και επαλήθευση.

Για την κατασκευή του μοντέλου εκπαίδευσης χρησιμοποιούμε το διαδοχικό (sequential) μοντέλο της βιβλιοθήκης Keras. Το διαδοχικό μοντέλο είναι κατάλληλο για μια απλή στοίβα στρωμάτων, όπου κάθε στρώμα έχει ακριβώς έναν τανυστή εισόδου και έναν τανυστή εξόδου. Παρακάτω περιγράφεται η αρχιτεκτονική του συνελικτικού μοντέλου:

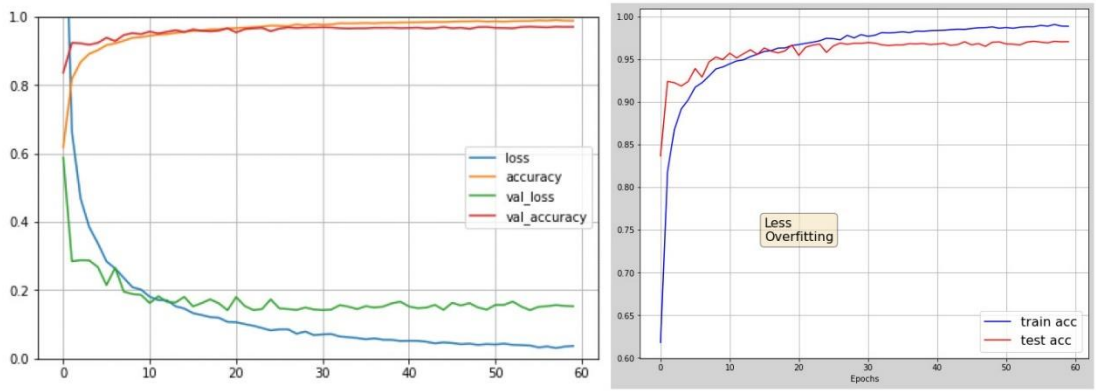
- Αρχικά προστίθενται δυο στρώματα συνέλιξης με 16 και 32 φίλτρα αντίστοιχα. Ταυτόχρονα, γίνεται χρήση της συνάρτησης ενεργοποίησης ReLU και ενός πυρήνα διαστάσεων (3,3) που καθορίζει ένα υποδεκτικό πεδίο 3×3.

- Έπειτα, προστίθεται ένα στρώμα υποδειγματοληψίας μέγιστης τιμής με παράθυρο συγκέντρωσης διαστάσεων 2×2. Με αυτόν τον τρόπο οι διαστάσεις της εικόνας μειώνονται κατά παράγοντα 2.
- Κατόπιν, προστίθεται ένα στρώμα κανονικοποίησης για τον μετασχηματισμό της εισόδου, διατηρώντας τη μέση τιμή εξόδου κοντά στο 0 και την τυπική απόκλιση εξόδου κοντά στο 1.
- Τα παραπάνω στρώματα εφαρμόζονται άλλη μια φορά, με μόνη διαφορά την αλλαγή στις παραμέτρους των στρωμάτων συνέλιξης. Το τρίτο και τέταρτο στρώμα συνέλιξης αποτελούνται από 64 και 128 φίλτρα αντίστοιχα.
- Στη συνέχεια, εφαρμόζεται ένα στρώμα επιπεδοποίησης για τη μετατροπή των δισδιάστατων δεδομένων σε ένα μονοδιάστατο διάνυσμα. Αυτό το στρώμα ακολουθείται από ένα πυκνό στρώμα με 512 νευρώνες με συνάρτηση ενεργοποίησης ReLU. Ακόμη, προστίθεται ένα στρώμα κανονικοποίησης και ένα στρώμα αφαίρεσης με ρυθμό .25, δηλαδή το 25% των νευρώνων αφαιρούνται τυχαία.
- Τέλος, το μοντέλο συμπληρώνει ένα πλήρως συνδεδεμένο στρώμα που εξάγει 82 κόμβους, όσες και οι κλάσεις του συνόλου δεδομένων. Ως συνάρτηση ενεργοποίησης του στρώματος ορίζεται η Softmax.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 62, 62, 16)        448

 conv2d_5 (Conv2D)           (None, 60, 60, 32)        4640

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 batch_normalization_3 (Batc  (None, 30, 30, 32)       128
 hNormalization)

 conv2d_6 (Conv2D)           (None, 28, 28, 64)        18496

 conv2d_7 (Conv2D)           (None, 26, 26, 128)       73856

 max_pooling2d_3 (MaxPooling  (None, 13, 13, 128)      0
 2D)

 batch_normalization_4 (Batc  (None, 13, 13, 128)      512
 hNormalization)

 flatten_1 (Flatten)         (None, 21632)             0

 dense_2 (Dense)             (None, 512)               11076096

 batch_normalization_5 (Batc  (None, 512)              2048
 hNormalization)

 dropout_1 (Dropout)         (None, 512)               0

 dense_3 (Dense)             (None, 82)                42066

=================================================================
Total params: 11,218,290
Trainable params: 11,216,946
Non-trainable params: 1,344
```

Σχήμα 4.16: Αρχιτεκτονική του συνελικτικού μοντέλου εκπαίδευσης.

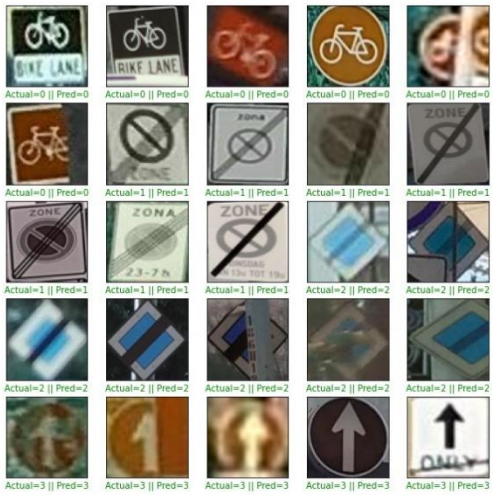
Μετά από την επιτυχή εκπαίδευση με χρήση της GPU Tesla T4 που παρέχεται από το Google Colab, η ακρίβεια του μοντέλου υπολογίστηκε με βάση τις εικόνες από το σύνολο δεδομένων δοκιμής. Έπειτα από αρκετές επαναλήψεις, η υψηλότερη τιμή του ποσοστού ακρίβειας που υπολογίστηκε ήταν 97.76%.



Σχήμα 4.17: Διαγράμματα μεταβολής της ακρίβειας και απώλειας ταξινόμησης κατά τη διάρκεια της εκπαίδευσης. Ο οριζόντιος άξονας δείχνει τις εποχές εκπαίδευσης και ο κάθετος άξονας τη τιμή από 0 έως 1.

```
test = pd.read_csv(data_dir + '/Test.csv')

labels = test["ClassId"].values
imgs = test["Path"].values

data =[]

for img in imgs:
    try:
        image = cv2.imread(data_dir + '/' +img)
        image_fromarray = Image.fromarray(image, 'RGB')
        resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
        data.append(np.array(resize_image))
    except:
        print("Error in " + img)
X_test = np.array(data)
X_test = X_test/255

#pred = model.predict_classes(X_test)
pred = np.argmax(model.predict(X_test), axis=-1)

#Accuracy with the test data
print('Test Data accuracy: ',accuracy_score(labels, pred)*100)
```

```
Test Data accuracy:  97.76247848537005
```



Σχήμα 4.18: Αριστερά παρουσιάζεται ο κώδικας για τον υπολογισμό του ποσοστού ακριβείας του μοντέλου στο σύνολο των εικόνων δοκιμής. Δεξιά παρουσιάζονται μερικές εικόνες από το σύνολο δοκιμής με την προβλεπόμενη και την πραγματική κλάση τους.

4.3.2 Εξαγωγή βαρών μοντέλου

Για να χρησιμοποιήσουμε τα βάρη αναγνώρισης στην εφαρμογή για συστήματα Android, απαραίτητο είναι τα βάρη να είναι σε μορφή TFLite που είναι συμβατή με το λειτουργικό σύστημα. Από τη βιβλιοθήκη Keras η εξαγωγή των βαρών γίνεται σε ένα αρχείο Ιεραρχικής Μορφής Δεδομένων (Hierarchical Data Format - HDF) *.h5*. Το αρχείο αυτό περιέχει πολυδιάστατους πίνακες με τα δεδομένα. Σε αντίθεση, η βιβλιοθήκη Tensorflow δίνει τη δυνατότητα εξαγωγής βαρών σε ένα αρχείο μορφής *SavedModel*. Το *SavedModel* είναι μια πιο ολοκληρωμένη μορφή που αποθηκεύει την αρχιτεκτονική του μοντέλου, τα βάρη, τις απώλειες και τα ανιχνευμένα υπογράμματα Tensorflow. Αυτό επιτρέπει στο Keras να επαναφέρει τόσο τα ενσωματωμένα επίπεδα όσο και τα προσαρμοσμένα αντικείμενα.

Σε σύγκριση με τη μορφή *SavedModel*, υπάρχουν δύο πράγματα που δεν περιλαμβάνονται στο αρχείο *h5*:

1. Οι εξωτερικές απώλειες και οι μετρήσεις δεν αποθηκεύονται σε αντίθεση με το SavedModel. Εάν υπάρχουν τέτοιες απώλειες και μετρικές στο μοντέλο και θέλουμε να συνεχίσουμε την εκπαίδευση, θα πρέπει να προσθέσουμε αυτές τις απώλειες πίσω μόνοι μας μετά τη φόρτωση του μοντέλου.

2. Το γράφημα υπολογισμού των προσαρμοσμένων αντικειμένων, όπως τα προσαρμοσμένα στρώματα, δεν περιλαμβάνεται στο αποθηκευμένο αρχείο. Κατά τη φόρτωση, το Keras θα χρειαστεί πρόσβαση στις κλάσεις αυτών των αντικειμένων, προκειμένου να ανακατασκευάσει το μοντέλο.

Η μορφή *h5* αποτελεί την καλύτερη επιλογή για τον σκοπό της εργασίας, αφού είναι μικρότερη και ελαφρύτερη από τη *SavedModel*, ενώ οι παραπάνω παραλήψεις της δεν επηρεάζουν το αποτέλεσμα της εξαγωγής συμπερασμάτων του αρχείου *tflite*. Το τελικό αρχείο θα είναι μορφής 32bit float-point για τη χρήση σε GPU, όπως οι ARM Mali και Qualcomm Adreno που υπάρχουν στις κινητές συσκευές. Η μετατροπή από *h5* σε *tflite* γίνεται μέσω της βιβλιοθήκης Tensorflow με τον παρακάτω κώδικα στο Σχήμα 4.19.

```
OUTPUT_TFLITE_MODEL_FROM_KERAS = "/content/gdrive/MyDrive/Colab Notebooks/mtsd-h5.tflite"

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TF Lite model.
with tf.io.gfile.GFile(OUTPUT_TFLITE_MODEL_FROM_KERAS, 'wb') as f:
  f.write(tflite_model)
```

Σχήμα 4.19: Κώδικας μετατροπής του αρχείου με τα βάρη μορφής h5 (Keras) σε tflite (Tensorflow).

Το τελικό αρχείο *tflite* δεν πέρασε από τη διαδικασία κβαντισμού, διότι θέλουμε το μεγαλύτερο δυνατό ποσοστό επιτυχίας. Το μέγεθος και η πολυπλοκότητα του μοντέλου δεν είναι τόσο μεγάλα ώστε να χρειαστεί να τα μειώσουμε, γι' αυτό και οι περαιτέρω βελτιστοποιήσεις δεν κρίνονται αναγκαίες.

4.4 Εφαρμογή στο Android Studio

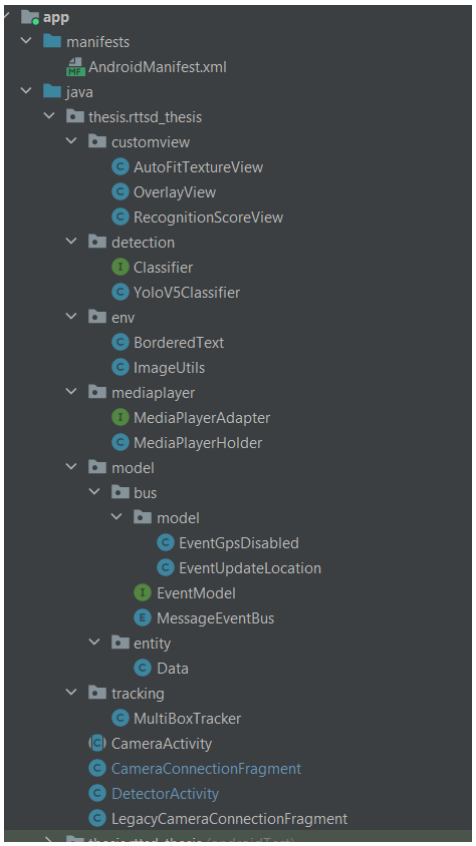
4.4.1 Εισαγωγή

Αφού ολοκληρώσαμε την εκπαίδευση ενός αλγορίθμου για την αναγνώριση των σημάτων στο δρόμο και ενός για την ταξινόμηση αυτών, χρειάζεται να δημιουργήσουμε μια εφαρμογή που θα συνδυάσουμε αυτούς τους δύο αλγορίθμους και θα τους δοκιμάσουμε στον πραγματικό κόσμο. Το εργαλείο που θα χρησιμοποιήσουμε είναι το Android Studio, με το οποίο δημιουργούνται εφαρμογές Android. Μέσω των κατάλληλων βιβλιοθηκών και τη χρήση της γλώσσας προγραμματισμού Java, θα στήσουμε μία εφαρμογή που θα χρησιμοποιεί την κάμερα του κινητού για να αναγνωρίσει σήματα στους δρόμους. Για να το πετύχουμε αυτό, θα περνάμε τις εικόνες της κάμερας μέσα από τα δύο μοντέλα, πρώτα το μοντέλο αναγνώρισης αντικειμένων και έπειτα, την κάθε έξοδο, από τον ταξινομητή. Τα αποτελέσματα των αντικειμένων που αναγνωρίστηκαν και το σήμα που αντιπροσωπεύουν, θα φαίνονται συνδυαστικά πάνω στην οθόνη του χρήστη, ζωγραφίζοντας ένα πλαίσιο γύρο από την πινακίδα και εμφανίζοντας τον τίτλο της πινακίδας και το ποσοστό σιγουριάς που προκύπτει από τον ταξινομητή. Μια ακόμη λειτουργία είναι η αναγνώριση της ταχύτητας του χρήστη, η οποία θα εμφανίζεται στο επάνω μέρος της οθόνης και η σύγκριση της με το τελευταίο ανώτατο όριο που αναγνωρίστηκε από τον αλγόριθμο. Σε περίπτωση που ο χρήστης το επιλέξει, θα αναπαράγονται φωνητικές ειδοποιήσεις και στη διεπιφάνια της εφαρμογής, μαζί με την προεπισκόπηση της κάμερας θα υπάρχουν πληροφορίες για την λειτουργία της εφαρμογής και κάποιες ρυθμίσεις που μπορεί να επηρεάσει ο χρήστης.

Παρακάτω, θα δούμε πιο αναλυτικά μερικά από τα πιο σημαντικά αρχεία της εφαρμογής από το Android Studio, αλλά και το τελικό αποτέλεσμα.

4.4.2 Δομή

Η δομή των αρχείων της εφαρμογής έχει γίνει με αρκετά απλό τρόπο. Πρώτο βρίσκεται το AndroidManifest.xml το οποίο περιέχει τις βασικές πληροφορίες για την εφαρμογή και έπειτα ξεκινάει η ομαδοποίηση των αρχείων συμβολίζει τον σκοπό τους. Το customview ρυθμίζει τη συμπεριφορά του παραθύρου προεπισκόπησης της διεπιφάνιας, το detection περιέχει τις κλάσεις για το μοντέλο αναγνώρισης αντικειμένων της εφαρμογής, ο φάκελος env περιέχει τις κλάσεις για τη διαχείριση των εικόνων αλλά και ενσωματώνει τα κομμάτια της απόδοσης ευανάγνωστου, πλαισιωμένου κειμένου σε έναν καμβά. Στο mediaplayer υπάρχουν τα αρχεία για τη διαχείριση και αναπαραγωγή ειδοποιήσεων ήχου και στο model υπάρχουν οι κλάσεις για τη διαχείριση των ενημερώσεων τοποθεσίας και ταχύτητας του χρήστη μέσω του GPS. Ο φάκελος tracking περιέχει τη κλάση MultiBoxTracker, η οποία ανιχνεύει και χειρίζεται τα υπάρχοντα αντικείμενα με τα νέα που έχουν αναγνωριστεί από τον αλγόριθμο αναγνώρισης. Τέλος τα αρχεία που βρίσκονται εκτός κατηγορίας είναι αυτά που καλούνται άμεσα από την εφαρμογή κατά τη διάρκεια της εκκίνησης και περιέχουν όλες τις βασικές μεθόδους και κώδικα για την αρχικοποίηση και την περεταίρω λειτουργία της εφαρμογής.



Σχήμα 4.20: Δομή αρχείων Android Studio.

4.4.3 Android Manifest

Στο AndroidManifest.xml βρίσκονται οι κύριες πληροφορίες της εφαρμογής. Αυτές είναι τα δικαιώματα πρόσβασης που ζητάμε από τον χρήστη όπως η χρήση της κάμερας, η πρόσβαση στο διαδίκτυο, η χρήση του στίγματος μέσω GPS ή μέσω IP σε περίπτωση που το GPS δεν είναι ενεργοποιημένο.

Ταυτόχρονα, στο ίδιο αρχείο ορίζουμε την ονομασία της εφαρμογής «RTTSD Thesis», το εικονίδιο έναρξης, και το αρχικό Activity που θα ενεργοποιηθεί κατά την έναρξη της εφαρμογής, το οποίο είναι το DetectorActivity. Προαιρετικά μπορούμε να προσθέσουμε μία προσαρμοσμένη οθόνη, που συνήθως

είναι μια εικόνα με κάποιο μικρό εφέ, το οποίο εμφανίζεται όση ώρα παίρνει για να φορτωθούν τα πρώτα στοιχεία της εφαρμογής. Αυτό ονομάζεται «Theme», κάτι που προσθέσαμε κι εμείς, μια λευκή οθόνη με το λογότυπο της εφαρμογής με την ονομασία του θέματος “SplashTheme”.

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

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission. ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="RTTSD Thesis"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:theme="@style/SplashTheme"
        android:supportsRtl="true">
        <activity android:name=".DetectorActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Σχήμα 4.21: AndroidManifest.xml.

4.4.4 Detector Activity

Το πρώτο Activity που συναντάμε είναι το DetectorActivity, το οποίο ορίζεται στο AndroidManifest, ως το αρχικό αρχείο που πρέπει να κληθεί κατά την αρχικοποίηση της εφαρμογής. Επειδή ένα από τα δυνατά σημεία της γλώσσας Java είναι η κληρονομικότητα, το DetectoActivity είναι ουσιαστικά υποκλάση του CameraActivity, το οποίο έχει τις μεθόδους onStart() και onCreate() οι οποίες ευθύνονται για την αρχικοποίηση των στοιχείων της διεπιφάνιας. Παρακάτω θα αναλύσουμε τα περιεχόμενα του Activity, καθώς και τις βασικές μεταβλητές του, τις οποίες θα χωρίσουμε σε τρείς κατηγορίες. Αυτές για που ευθύνονται για το μοντέλο αναγνώρισης αντικειμένων που δημιουργήσαμε με το YOLOv5 και μετατρέψαμε σε TFLite για να είναι συμβατό και πιο αποδοτικό, τις μεταβλητές για την αρχικοποίηση της διεπιφάνιας του χρήστη και της οθόνης προεπισκόπησης, όπου θα μπορεί να δει σε ζωντανό χρόνο τις πινακίδες και τα σήματα που αναγνωρίζονται, και τις μεταβλητές για την αρχικοποίηση του ταξινομητή, που θα παίρνει τις εξόδους του μοντέλου αναγνώρισης αντικειμένων, ως είσοδο και θα επιστρέφει ποιο σήμα αναγνωρίζει.

```
public class DetectorActivity extends CameraActivity implements OnImageAvailableListener {

 // Variables for Object Detection
 private static final int TF_OD_API_INPUT_SIZE = 640;
 private static final boolean TF_OD_API_IS_QUANTIZED = true;
 private static final String TF_OD_API_MODEL_FILE = "sign_recognitionQ.tflite";
 public static final String TF_OD_API_LABELS_FILE = "sign_recognition.txt";
 public static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.6f;
 private int maximumResults = 3;

 // Variables for user's camera preview
 private static final boolean MAINTAIN_ASPECT = true;
 private static final Size DESIRED_PREVIEW_SIZE = new Size( width: 640,  height: 480);
 private static final boolean SAVE_PREVIEW_BITMAP = false;
 private static final float TEXT_SIZE_DIP = 10;

 //Variables for Classifier
 public static float CLASSIFICATION_THRESHOLD = 0.6f;
 public static String MODEL_FILENAME = "model82Q.tflite";
 private static SwitchCompat notification;
 public static final int INPUT_IMG_SIZE_WIDTH = 64;
 public static final int INPUT_IMG_SIZE_HEIGHT = 64;
```

Σχήμα 4.22: Detector Activity.

Οι πιο βασικές από την πρώτη κατηγορία είναι η τιμή εισόδου στο μοντέλο αναγνώρισης αντικειμένων η οποία είναι τα 640px με το οποίο έχει εκπαιδευτεί ο αλγόριθμος Yolov5s, η μετατροπή του μοντέλου στην κβαντοποιημένη εκδοχή του (quantized) και οι διαδρομές των αρχείων με τα βάρη του μοντέλου και της κλάσεις, όπου στην πραγματικότητα περιέχει μόνο την τιμή ‘sign’. Επίσης για το YOLOv5 έχουμε θέσει το ελάχιστο κατώφλι αναγνώρισης πινακίδας στο 60% και μέγιστο αριθμό αναγνώρισης πινακίδων στις τρείς, τις οποίες αποφασίσαμε ότι θα μπορούν να αλλάξουν από ένα μενού επιλογών που θα δίνεται στο χρήστη.

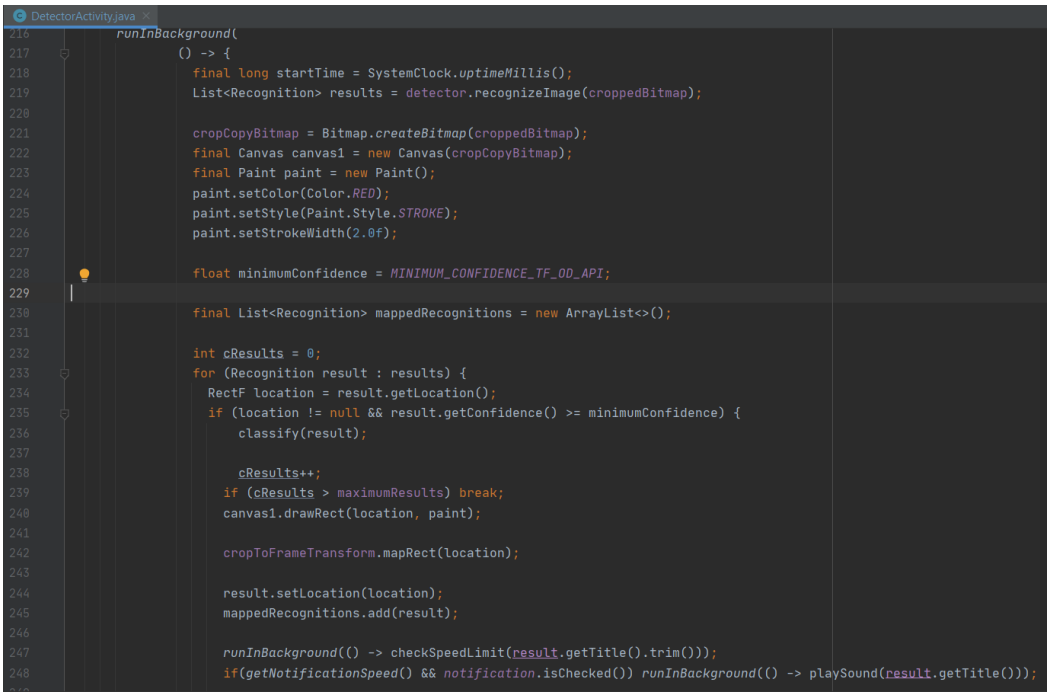
Στη δεύτερη κατηγορία, βρίσκονται οι ρυθμίσεις της διεπαφής που χρησιμοποιεί την κάμερα, η οποία ιδανικά θα θέλαμε να έχει διαστάσεις 640x640px και για τις εικόνες που θα «τραβάει», αλλά και για την προεπισκόπηση που θα βλέπει ο χρήστης. Η επιλογή αυτή έγινε για να μη χρειαστεί να επεξεργαστούμε το μέγεθος της φωτογραφίας πρώτού την δώσουμε ως είσοδο στον αλγόριθμο αναγνώρισης, αλλά και για να κρατήσουμε την οπτική του χρήστη κοντά στην πραγματικότητα. Στην περίπτωση που η οθόνη του χρήστη δεν επαρκεί για να στηθεί σωστά η προεπισκόπηση, τότε ο χώρος της εφαρμόζεται σε όσο το δυνατών μεγαλύτερη έκταση μπορεί. Κατά τη λήψη του στιγμιότυπου θα γίνουν οι κατάλληλες μετατροπές για να δοθεί στον αλγόριθμο αναγνώρισης αντικειμένων του YOLOv5.

Η τρίτη κατηγορία μεταβλητών αφορά τις τιμές που ρυθμίζουν τη συμπεριφορά του ταξινομητή που θα παίρνει ως είσοδο το αποτέλεσμα του αλγορίθμου αναγνώρισης αντικειμένων και στην έξοδο θα επιστρέφει την κλάση που ανήκει η εικόνα που του δόθηκε. Πριν τροφοδοτήσουμε με εικόνες τον αλγόριθμο, πρέπει να αλλάξουμε το μέγεθος των εικόνων, το οποίο και δηλώνουμε με δύο μεταβλητές, μία για το ύψος και μία για το πλάτος, οι οποίες θα χρησιμοποιηθούν στη μέθοδο prepareImageForClassification(). Επίσης, ορίζουμε το κατώφλι του ποσοστού αξιοπιστίας που θέλουμε να έχει ο ταξινομητής, μια τιμή η οποία μπορεί και να αλλάζει δυναμικά μέσω της διεπιφάνιας, και τη διαδρομή για το αρχείο του μοντέλου που είναι σε TensorFlow Lite. Ο μέγιστος αριθμός αποτελεσμάτων δε χρειάζεται να δηλωθεί σε κάποια μεταβλητή, καθώς θέλουμε να πάρουμε μόνο το πρώτο αποτέλεσμα, το οποίο είναι και αυτό που θα αναγνωριστεί με το μεγαλύτερο ποσοστό.

Πέρα από τις μεταβλητές και σταθερές οι οποίες παίζουν πολύ σημαντικό ρόλο στην αρχικοποίηση των συστατικών της εφαρμογής, υπάρχουν και σημαντικά κομμάτια κώδικα, τα οποία είναι μέσα σε μεθόδους και θα τα περιγράψουμε σύντομα. Μερικές μέθοδοι που βρίσκονται σε αυτό το Activity είναι

η setupViews() η οποία κληρονομείται από την κλάση CameraActivity, αλλά υλοποιείται στην υποκλάση της, δηλαδή την DetectorActivity, και εμφανίζει τα κείμενα με τις πληροφορίες στη διεπαφή. Για να εμφανίσουμε μπάρες τις οποίες θα μπορεί ο χρήστης να σύρει το δάχτυλο του και να αλλάξει κάποιες ρυθμίσεις, όπως το κατώφλι της τιμής αξιοπιστίας του ταξινομητή, χρησιμοποιούμε την κλάση SeekBar της βιβλιοθήκης android.widget.

Μιας και η DetectorActivity είναι το βασικό Activity της εφαρμογής, όλες οι διαδικασίες αναγνώρισης και ταξινόμησης των αποτελεσμάτων ξεκινούν μέσα από τις μεθόδους της. Η onPreviewSizeChosen() αρχικοποιεί τον Detector και προετοιμάζει την προεπισκόπηση οθόνης για την στιγμή όπου θα αρχίσει να εξάγει στιγμιότυπα και θα τα αποθηκεύει σε ένα πίνακα Matrix. Οι λήψεις εικόνων γίνονται μέσω της processImage(), μια βασική μέθοδος, που δημιουργεί ένα στιγμιότυπο από την κάμερα του χρήστη, το δίνει στον αλγόριθμο ανίχνευσης αντικειμένων ο οποίος τρέχει το παρασκήνιο και επιστρέφει στην έξοδο του όλα να αντικείμενα που αναγνώρισε. Πριν τελειώσει η διεργασία, επεξεργάζεται τα πλαίσια των αντικειμένων δίνοντας τα ως είσοδο στον ταξινομητή μέσω της μεθόδου classify(), η οποία αναγνωρίζει το σήμα της πινακίδας που της δόθηκε και αλλάζει τις τιμές “title” και “confidence”, δηλαδή τον τίτλο και το ποσοστό αξιοπιστίας, που στην αρχή ορίστηκε από το YOLOv5, με τα νέα δεδομένα που συμπέρανε. Έτσι, στην περίπτωση που ο αλγόριθμος αναγνώρισης βρήκε κάποια πινακίδα και ο ταξινομητής επιτυχώς αναγνώρισε σε ποια από τις 82 κλάσεις ανήκει, έχουμε ένα αντικείμενο τύπου Recognition που περιέχει την τοποθεσία του πλαισίου στο στιγμιότυπο που αναγνωριστικέ, τον τίτλο του σήματος και το ποσοστό που θεωρεί ο αλγόριθμος ταξινόμησης ότι βρέθηκε με επιτυχία. Στην περίπτωση που ο ταξινομητής δεν αναγνωρίσει σε ποιο σήμα αντιστοιχεί η είσοδος που του δόθηκε, επειδή το αντικείμενο αρχικοποιήθηκε με τις τιμές του αλγόριθμου αναγνώρισης, ο τίτλος θα παραμείνει “Sign” και το ποσοστό αξιοπιστίας αυτό που έθεσε ο YOLOv5 ότι αντιστοιχεί στο αποτέλεσμα. Ένας παράγοντας που μπορεί ο ταξινομητής να μην έχει κάποιο αποτέλεσμα, είναι γιατί είτε από προεπιλογή είτε από κάποια ρύθμιση του χρήστη, το κατώφλι αξιοπιστίας είναι αρκετά υψηλό και δεν εγκρίνεται η τιμή για να περαστεί στο τελικό αποτέλεσμα. Το κατώφλι του μοντέλου αναγνώρισης είναι 60% και δεν μπορεί να πειραχτεί, ωστόσο αποφασίσαμε ότι θα είχε ενδιαφέρον και θα ήταν χρήσιμο, να εμφανίζεται η πινακίδα που αναγνωρίστηκε ακόμη και χωρίς να έχει ταξινομηθεί.



Σχήμα 4.23: Κώδικας Αναγνώρισης Σημάτων.

Όταν ολοκληρωθεί το κομμάτι της αναγνώρισης της πινακίδας και του σήματος, τότε καλείται η μέθοδος checkSpeedLimit(), η οποία ελέγχει την τιμή που δόθηκε από τον ταξινομητή και την ορίζει ως το πιο πρόσφατο όριο ταχύτατης. Στις υπόλοιπες περιπτώσεις που θα βρεθεί κάποιο άλλο σήμα και ο χρήστης έχει ενεργές τις ηχητικές ειδοποιήσεις, καλείται η playSound(), η οποία μέσω της switch-case, επιλέγει πιο ηχητικό απόσπασμα θα ακουστεί σύμφωνα με το σήμα που δέχθηκε. Όταν ακούγεται κάποιο μήνυμα πινακίδας, τότε οι ειδοποιήσεις απενεργοποιούνται προσωρινά και ξανά ενεργοποιούνται από το CameraActivity σε χρόνο 5 δευτερολέπτων, έτσι ώστε να μην ακούγονται ταυτόχρονα πολλά μηνύματα και σημάνσεις.

```
DetectorActivity.java
492 @    private void classify (Recognition result){
493          Matrix matrix = new Matrix();
494          matrix.postRotate( degrees: 0);
495
496          Bitmap crop = Bitmap.createBitmap(croppedBitmap,
497                  (int) result.getLocation().left,
498                  (int) result.getLocation().top,
499                  (int) result.getLocation().width(),
500                  (int) result.getLocation().height(),
501                  matrix,
502                   filter: true);
503
504          if (crop != null) {
505              try {
506                  ImageView view = findViewById(R.id.signImg);
507                  crop = prepareImageForClassification(crop);
508                  view.setImageBitmap(crop);
509
510                  // Initialization
511                  ImageClassifier.ImageClassifierOptions options =
512                          ImageClassifier.ImageClassifierOptions.builder().setMaxResults(1).setScoreThreshold(CLASSIFICATION_THRESHOLD).setNumThreads(4).build();
513
514                  ImageClassifier imageClassifier = ImageClassifier.createFromFileAndOptions(
515                          getApplicationContext(), MODEL_FILENAME, options);
516
517                  // Run inference
518                  List<Classifications> results2 = imageClassifier.classify(
519                          TensorImage.fromBitmap(crop));
520
521                  result.setTitle(results2.get(0).getCategories().get(0).getLabel());
522                  result.setConfidence(results2.get(0).getCategories().get(0).getScore());
523              } catch (Exception e) {
524                Log.e( tag: "SLClassifier error:", e.getMessage(),e);
```

Σχήμα 4.24: Η μέθοδος Classify.

Πριν ξεκινήσει να τρέχει στο παρασκήνιο η διαδικασία αναγνώρισης αντικειμένων και η μετέπειτα ταξινόμηση τους, αποθηκεύουμε σε μία μεταβλητή την χρονική στιγμή που ξεκινάει η διεργασία και στο τέλος κάνουμε μία πράξη αφαίρεσης για να βρούμε το χρόνο που πέρασε, έτσι ώστε να έχουμε τον χρόνο που πέρασε για την εξαγωγή συμπερασμάτων, από τη στιγμή της λήψης του στιγμιότυπου. Αυτό απεικονίζεται στην κύρια διεπαφή της εφαρμογής και ονομάζεται χρόνος συμπεράσματος (Inference Time).

Έχουν επίσης προστεθεί κουμπιά με τα οποία μπορεί ο χρήστης να αυξήσει ή να μειώσει τον αριθμό των μέγιστων πινακίδων που επιτρέπεται να αναγνωριστούν την κάθε στιγμή. Καλώντας την μέθοδο setMaximumResults() αλλάζει η μεταβλητή των μέγιστων αναγνωρίσεων έτσι ώστε στην επόμενη έξοδο να δοθούν περισσότερα αντικείμενα προς ταξινόμηση. Άλλη μία επιλογή που λειτουργεί με αυτό τον τρόπο είναι ο αριθμός των νημάτων (Threads) για να προσαρμόσει ο χρήστης, πως θα διαχειριστεί η εφαρμογή τον φόρτο της κατά τη διάρκεια της εκτέλεσης. Ανάλογα τις δυνατότητες τις κάθε συσκευής, αυτή η επιλογή μπορεί να αυξήσει ή να μειώσει την ταχύτητα της αναγνώρισης σημάτων.

4.4.5 CameraActivity

Σε αυτό το Activity υλοποιείται η μέθοδος onCreate(), η οποία περιέχει όλες τις διεργασίες που καλούνται κατά τη δημιουργία της εφαρμογής. Συνήθως αυτή η μέθοδος υπάρχει στο Activity απ' όπου ξεκινάει η εφαρμογή, ωστόσο επειδή η DetectorActivity κληρονομεί όλες τις μεθόδους της CameraActivity, καλείται μέσα από αυτή και έχουμε το ίδιο αποτέλεσμα.

Η μέθοδος onCreate() ενημερώνει το λειτουργικό σύστημα να κρατήσει ανοιχτή την οθόνη, ώστε να λειτουργεί η εφαρμογή συνέχεια, καλεί τη διεπαφή στο προσκήνιο και δημιουργεί ένα Dispodable αντικείμενο, το οποίο τρέχει σε επανάληψη και σε χρονικό διάστημα 5 δευτερολέπτων ενεργοποιεί τη λειτουργεία ειδοποίησης ορίου ταχύτητας που αναλύσαμε παραπάνω εφ' όσον έχει απενεργοποιηθεί από το σύστημα ελέγχου ειδοποιήσεων.

Μέσα στην onCreate καλείται η setCallback(), η οποία ενημερώνει το σύστημα σε περίπτωση που η τοποθεσία του χρήστη αλλάξει για να πάρει τις νέες πληροφορίες, και ελέγχει αν έχει απενεργοποιηθεί το GPS. Έπειτα καλείται η setupViews() η οποία είδαμε ότι υλοποιείται από την DetectorActivity και επεξεργάζεται τις πληροφορίες που εμφανίζονται στη διεπαφή που αρχικοποιήθηκε πριν.

Στη συνέχεια, ελέγχουμε αν ο χρήστης έχει δώσει τα απαραίτητα δικαιώματα για να μπορέσουμε να χρησιμοποιήσουμε την κάμερα, την τοποθεσία και το διαδίκτυο, αρχικοποείται η διαδικασία εύρεσης της τοποθεσίας του χρήστη μέσω του LocationManager. Κάθε φορά που αλλάζει η τοποθεσία του χρήστη, παίρνουμε νέο στίγμα το οποίο αν είναι διαφορετικό με το προηγούμενο ξέρουμε ότι ο χρήστης κινείται και υπολογίζουμε την ταχύτητα που έχει βάση των δεδομένων του χρόνου που πέρασε από το προηγούμενο στίγμα και τη διαφορά της απόστασης που διένυσε και ενημερώνουμε την διεπαφή για να γίνουν οι ενέργειες που χρειάζονται. Σε περίπτωση που ο χρήστης δεν έχει επιτρέψει στην εφαρμογή να χρησιμοποιήσει όλα τα παραπάνω χαρακτηριστικά, εμφανίζεται ένα αναδυόμενο παράθυρο, που τον προτρέπει να δώσει την έγκρισή του.

Αφού ολοκληρωθεί η εκτέλεση της προηγούμενης μεθόδου, καλείται η setFragment(), που αρχικοποιεί το fragment της κάμερας, η οποία βρίσκεται στο κεντρικό σημείο της διεπαφής και αποτελεί το μέσο λήψης στιγμιότυπων και αναγνώρισης σημάτων.

Στη συνέχεια ορίζουμε τους ενεργούς listeners, οι οποίοι «ακούν» τα στοιχεία που βρίσκονται στην οθόνη του χρήστη και μόλις υπάρξει κάποια διάδραση του χρήστη με την εφαρμογή και εκτελούν τον αντίστοιχο κώδικα που βρίσκεται στην μέθοδο onClick(). Η onClick(), ελέγχει όλα να αντικείμενα που έχουν συνδεθεί με τον onClickListener και εκτελεί τον κώδικα που αντιστοιχεί στη λειτουργία που θέλουμε να εκτελέσει το κάθε αντικείμενο, σύμφωνα με το έναν κωδικό ID, όπου είναι μοναδικός για το καθένα και μας επιτρέπει να τα ξεχωρίζουμε μεταξύ τους.

CameraActivity.java

```
@Override
public void onClick(View v) {
  if (v.getId() == R.id.plus) {
    String threads = threadsTextView.getText().toString().trim();
    int numThreads = Integer.parseInt(threads);
    if (numThreads >= 9) return;
    numThreads++;
    threadsTextView.setText(valueOf(numThreads));
    setNumThreads(numThreads);
  }else if (v.getId() == R.id.minus) {
    String threads = threadsTextView.getText().toString().trim();
    int numThreads = Integer.parseInt(threads);
    if (numThreads == 1) return;
    numThreads--;
    threadsTextView.setText(valueOf(numThreads));
    setNumThreads(numThreads);
  }else if(v.getId() == R.id.plus2){
    String signs = signsTextView.getText().toString().trim();
    int numSigns = Integer.parseInt(signs);
    if (numSigns >= 9) return;
    numSigns++;
    signsTextView.setText(valueOf(numSigns));
    setMaximumResults(numSigns);
  }else if(v.getId() == R.id.minus2){
    String signs = signsTextView.getText().toString().trim();
    int numSigns = Integer.parseInt(signs);
    if (numSigns == 1) return;
    numSigns--;
    signsTextView.setText(valueOf(numSigns));
    setMaximumResults(numSigns);
  }
}
```

Σχήμα 4.25: Η μέθοδος onClick() του CameraActivity.

4.4.6 CameraConnectionFragment & LegacyCameraConnectionFragment

Το CameraConnectionFragment είναι ένα fragment το οποίο ελέγχει τις λειτουργίες της κάμερας. Η Legacy έκδοση του fragment, αφορά τη χρήση του camera2API του android, το οποίο χρησιμοποιείται σε περίπτωση που υπάρχει μία εξωτερική κάμερα συνδεμένη στο android και όχι η ενσωματωμένη. Πέρα από κάποιες διαφορετικές λειτουργίες, η σημασία τους για την εφαρμογή είναι ίδια. Επιλέγει το μέγεθος προεπισκόπησης της κάμερας έτσι ώστε να είναι το μικρότερο καρέ ανά pixel που μπορεί να περιέχει ένα τετράγωνο ύψους επί πλάτος. Ελέγχει τη σωστή μετατροπή του στιγμιότυπου και την σωστή περιστροφή της οθόνης σε αρχείο JPEG. Όταν η οθόνη απενεργοποιείται και ενεργοποιείται ξανά, το SurfaceTexture, η κλάση που καταγράφει τα στιγμιότυπα, είναι ήδη διαθέσιμο και δεν θα κληθεί το "onSurfaceTextureAvailable". Σε αυτή την περίπτωση, μπορούμε να ανοίξουμε μια κάμερα και να ξεκινήσουμε την προεπισκόπηση από εδώ (διαφορετικά, περιμένουμε μέχρι να είναι έτοιμη η επιφάνεια στο το SurfaceTextureListener).

4.4.7 Περιβάλλον (env)

Σε αυτή την κατηγορία υπάρχουν δύο κλάσεις οι οποίες σχετίζονται με τον χειρισμό των εικόνων. Η ImageUtils περιέχει βοηθητικές μεθόδους που υπολογίζουν τον αριθμό των Bytes μιας εικόνας για κάποιες διαστάσεις, αποθηκεύει τα αντικείμενα τύπου Bitmap στο δίσκο για ανάλυση και επεξεργασία και χειρίζεται την περικοπή μιας εικόνας και τον μετασχηματισμό της σε πίνακα. Η δεύτερη κλάση στον φάκελο είναι η BorderedText, η οποία διαχειρίζεται τον σχεδιασμό πλαισίων, την δημιουργία κειμένου πάνω από αυτά, την στοίχιση και τον χρωματισμό.

4.4.8 CustomView

Οι κλάσεις σε αυτή τη κατηγορία υποστηρίζουν τη λειτουργία της προεπισκόπησης κάμερας. Συγκεκριμένα, η AutoFitTextureView είναι μία υποκλάση η οποία κληρονομεί την TextureView και σκοπός της είναι να προσαρμόζει την προεπισκόπηση της κάμερας σε μια καθορισμένη αναλογία διαστάσεων. Με τις μεθόδους τις, ορίζει την αναλογία διαστάσεων για την προβολή ελέγχει. Το μέγεθος της οποία υπολογίζεται με βάση τις παραμέτρους που τις έχουμε δώσει. Στη δική μας περίπτωση η αναλογία είναι 1:1, αφού το μέγεθος της προεπισκόπησης θέλουμε να έχει διαστάσεις 640x640px. Ωστόσο αν αυτό δεν γίνεται τότε χρησιμοποιείται το πλησιέστερο δυνατό. Η κλάση OverlayView κληρονομεί την View και προορίζεται για να παρέχει callback κλήσεις σε άλλες κλάσεις που χρησιμοποιούνται στην εφαρμογή.

4.4.9 MediaPlayer

Η ονομασία της κατηγορίας αυτής περιγράφει την λειτουργία της. Περιέχει μέσα την κλάση MediaPlayerHolder και το interface MediaHolderAdapter. Η πρώτη κλάση, κληρονομεί τις μεθόδους του Interface και τις υλοποιεί για να μπορέσει αποτελεσματικά να διαβάζει και να αναπαράγει, κάποιο από τα αρχεία ήχου που του δίνονται όταν ο χρήστης έχει ενεργοποιήσει τη δυνατότητα φωνητικών ειδοποιήσεων. Αρχικοποιείται ένας MediaPlayer της βιβλιοθήκης android.media και στη συνέχεια κάθε φορά που καλείται η loadMedia() διαβάζει το μοναδικό ID του αρχείου mp3, ξεκινάει την αναπαραγωγή και περιμένει για το επόμενο.

4.4.10 Tracking

Ο φάκελος Tracking περιέχει την κλάση MultiBoxTracker, ανιχνεύει τα πλαίσια που ζωγραφίζονται και αποτρέπει την δημιουργία δύο πλαισίων το ένα πάνω στο άλλο στην περίπτωση που αυτά βρίσκονται πολύ κοντά μεταξύ τους.

4.1.1 Model Entity & Bus

Η κατηγορία Model είναι η πιο περίπλοκη από τις άλλες, καθώς περιέχει αρκετές κλάσεις και αρκετούς υποφακέλους μέσα της. Ο φάκελος entiny περιέχει την κλάση Data, η οποία έχει τα δεδομένα που υπολογίζουμε και αποθηκεύουμε όταν στέλνει ενημέρωση τοποθεσίας το GPS του χρήστη. Αυτά είναι η πιο πρόσφατη τοποθεσία, ο χρόνος που πέρασε μέχρι να γίνει η ανανέωση από την τελευταία, η απόσταση μεταξύ των δύο τελευταίων σημείων και η ταχύτητα που μετακινήθηκε ανάμεσα τους.

Ο φάκελος Bus περιέχει τον φάκελο model και 2 αρχεία. Το ένα είναι το interface EventModel και το άλλο ένα αρχείο τύπου Enum, μια ειδική «κλάση», που αντιπροσωπεύει μια ομάδα σταθερών, δηλαδή τιμές που δεν αλλάζουν. Οι κλάσεις στο φάκελο model, είναι η EventGPSDisabled και η EventUpdateLocation. Η πρώτη έχει κενό σώμα, καθώς τη χρησιμοποιούμε για να ξεχωρίσουμε τα events, δηλαδή τα γεγονότα που συμβαίνουν μεταξύ των δύο προαναφερθέντων. Αυτές οι δύο κλάσεις που κληρονομούν την EventModel, περιέχουν ένα αντικείμενο τύπου Data από το φάκελο Entity και

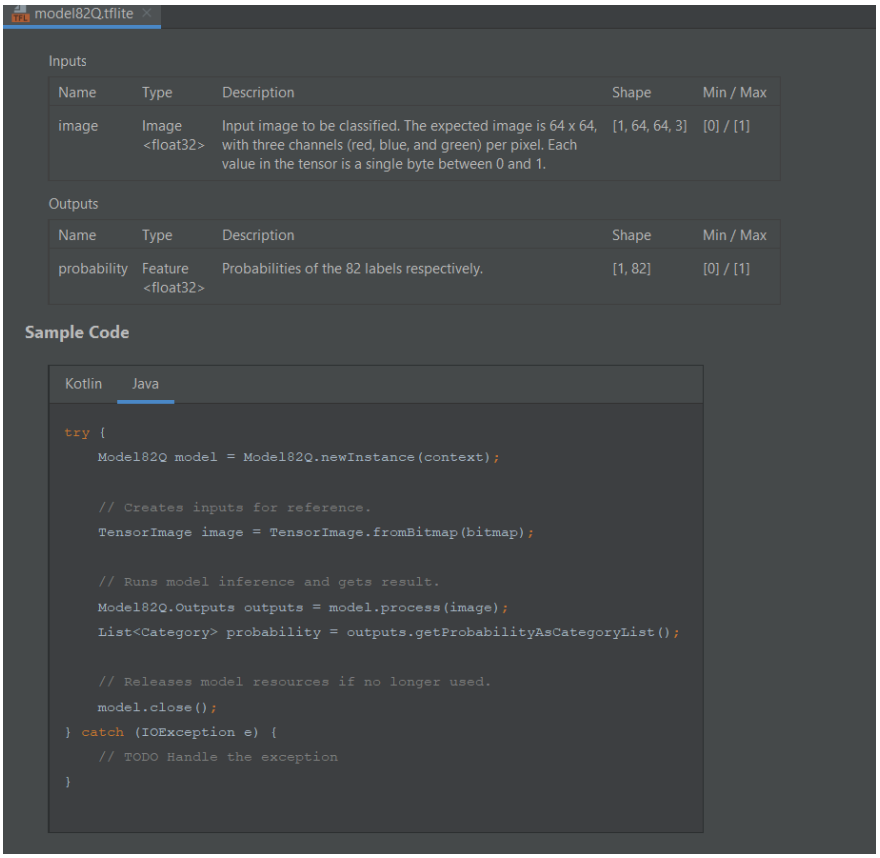
μας βοηθάνε να καταλάβουμε πότε ο χρήστης έχει ενεργό το GPS και πότε το απενεργοποιεί, για να εκτελέσουμε τις απαραίτητες ενέργειες.

4.4.11 Αναγνώριση (Detection)

Η αναγνώριση γίνεται μέσω της κλάσης YOLOv5Classifier. Η κλάση YOLOv5Classifier κληρονομεί το interface Classifier, μία γενική διεπαφή για την αλληλεπίδραση με διαφορετικές μηχανές αναγνώρισης, δημιουργημένη από την ομάδα του TenforFlow. Στη δική μας περίπτωση τη χρησιμοποιούμε για την υποκλάση YOLOv5Classifier, τον αλγόριθμο που αναγνωρίζει πινακίδες στη δοθείσα εικόνα που θα του τροφοδοτήσουμε. η οποία περιέχει τις βασικές μεθόδους που χρησιμοποιούνται για την αναγνώριση σημάτων. Οι τιμές που ορίζονται από τη διεπαφή, είναι ένας ξεχωριστός αριθμός που δίνει σε κάθε αποτέλεσμα, το ποσοστό αξιοπιστίας που το αναγνώρισε, το όνομα του, την τοποθεσία του και τον αύξον αριθμό κλάσης που ανήκει. Η κλάση YOLOv5Classifier υλοποιεί τον διερμηνέα (interpreter) του TensorFlow, ο οποίος φορτώνει το μοντέλο tflite, για να μπορέσει να κάνει τις μετέπειτα αναγνωρίσεις. Αφότου η εφαρμογή αποθηκεύσει προσωρινά ένα στιγμιότυπο και το μετατρέψει σε μορφή Bitmap, το στέλνει στη μέθοδο recognizeImage() που υλοποιείται σε αυτή τη κλάση και ξεκινάει η διαδικασία αναγνώρισης. Η μέθοδος αποθηκεύει τα αποτελέσματα σε μια λίστα τύπου ArrayList<Recognition> με τις πληροφορίες που κληρονομεί από τη διεπιφάνια που αναφέραμε και δημιουργεί μία καταχώριση για κάθε πλαίσιο που ανιχνεύει όσο κρατάει η διεργασία.

4.4.12 TensorFlow Classifier API

Ολοκληρώνοντας τη διαδικασία αναγνώρισης πινακίδων, ακολουθεί η μέθοδος Classify στο κύριο Activity που φορτώνει το αρχείο tflite του ξεχωριστού ταξινομητή που δημιουργήσαμε για να αναγνωρίσουμε σε ποιο από τα 82 σήματα αντιστοιχεί σε κάθε μία από τις πινακίδες. Για να χρησιμοποιήσουμε τον ταξινομητή στην εφαρμογή, χρησιμοποιήσαμε το “ML Binding” που προσφέρεται από το TensorFlow. Ένα εργαλείο το οποίο μας επιτρέπει να εξάγουμε ευκολά τα αποτελέσματα του μοντέλου και να τα επεξεργαστούμε όπως θέλουμε.



Σχήμα 4.26: ML BInding Feature.

4.4.13 Assets, res, ml

Στο Android μπορεί κανείς να αποθηκεύσει το ακατέργαστο αρχείο περιουσιακών στοιχείων (assets) όπως JSON, κείμενο, mp3, HTML, pdf κ.λπ. σε δύο πιθανές τοποθεσίες. Η μία είναι ο φάκελος assets, όπου αποθηκεύουμε αρχεία τα οποία διαβάζουμε με έναν AssetManager, όπως το αρχείο tflite και οι ετικέτες του. Λόγω της ειδικής περίπτωσης του αρχείου tflite του ταξινομητή που δημιουργήθηκε μέσω του TensorFlow ML Binding, τον έχουμε αποθηκεύσει στο φάκελο ml ο οποίος δημιουργήθηκε αυτόματα για αυτή την περίπτωση. Η δεύτερη τοποθεσία είναι η φάκελος res (resources) που τοποθετούμε οποιοδήποτε αρχείο θέλουμε να χρησιμοποιήσουμε μέσω της XML ιδιότητας στη Java, με ένα ξεχωριστό ID για το καθένα. Εκεί βάλαμε τα αρχεία xml των εφαρμογών, φωτογραφίες, λογότυπα και τα αρχεία ήχου mp3 για τις φωνητικές ειδοποιήσεις.

4.4.14 Gradle – Εξαρτήσεις

Στο αρχείο Gradle έχουμε ορίσει τις τιμές της ελάχιστης και της μέγιστης έκδοσης που μπορεί να υποστηρίξει η εφαρμογή, οι οποίες εξαρτώνται από τα στοιχεία που χρησιμοποιήσαμε στην δημιουργία της εφαρμογής. Οι τιμές που επιλέξαμε για την ελάχιστη είναι η 23η και η μεγαλύτερη 30η Android SDK. Η έκδοση του κώδικα παρέμεινε πρώτη, αφού δεν το έχουμε δημοσιεύσει κάπου ακόμα, όπως και το versionName. Η έκδοση Java που χρησιμοποιήθηκε είναι η 1.8 και επιπλέον έχει ενεργοποιηθεί η λειτουργεία mlModelBinding και ρυθμίστηκαν τα aaptoptions να μη συμπιέσουν τα αρχεία tflite κατά τη διάρκεια του build της εφαρμογής. Στο παρακάτω σχήμα (Σχήμα 4.27) φαίνονται οι βιβλιοθήκες που χρησιμοποιήσαμε στην εφαρμογή.

```
dependencies{
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'

    // RX java + android + view binding
    implementation 'io.reactivex.rxjava2:rxjava:2.2.9'
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'

    implementation 'org.tensorflow:tensorflow-lite-metadata:0.1.0'
    implementation 'org.tensorflow:tensorflow-lite-task-vision:0.1.0'
    implementation 'org.tensorflow:tensorflow-lite-gpu:2.3.0'
}
```

Σχήμα 4.27: Gradle Dependencies.

Μέσα σε αυτές ξεχωρίζουν οι βιβλιοθήκες του tflite, οι οποίες είναι κρίσιμες για τη λειτουργία του "ML Binding", την αξιοποίηση της GPU και τις κλάσεις για την χρήση των μηχανισμών αναγνώρισης αντικειμένων.

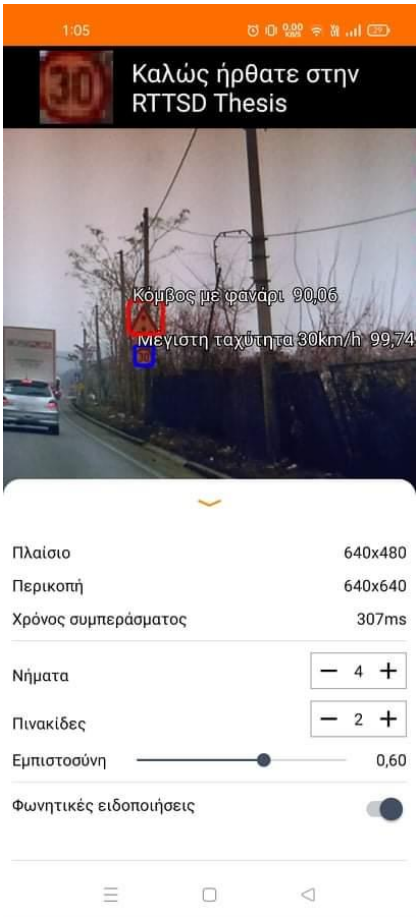
4.4.15 Αποτέλεσμα Εφαρμογής & Ανταπόκριση σε πραγματικές συνθήκες

Η τελική εφαρμογή σχεδιάστηκε έτσι ώστε να είναι εύκολη στη χρήση και γρήγορη στην εκκίνηση, για χρήστες που δεν είναι αρκετά εκπαιδευμένοι στη χρήση έξυπνων κινητών ή βρίσκονται στο αυτοκίνητο ενώ τη χρησιμοποιούν. Στη διεπαφή υπάρχει το παράθυρο προεπισκόπησης της κάμερας και ένα drop-down μενού το οποίο δίνει κάποιες πληροφορίες και επιλογές στο χρήστη. Οι πληροφορίες που δίνονται στο χρήστη είναι το μέγεθος του πλαισίου της κάμερας προεπισκόπησης, το μέγεθος όπου θα γίνει η περικοπή της εικόνας για να περάσει στον αλγόριθμο εντοπισμού αντικειμένων και ο χρόνος που χρειάζεται για να αναγνωρίσει τα ζητούμενα αντικείμενα σε ένα στιγμιότυπο.

Παρακάτω, διαχωρισμένα με μια γραμμή, είναι τα στοιχεία τα οποία ο χρήστης μπορεί να αλλάξει. Τα νήματα (Threads) είναι ο αριθμός των ταυτόχρονων διεργασιών που μπορούν να τρέξουν ταυτόχρονα από τον επεξεργαστή του κινητού. Η τιμή δίπλα από τις Πινακίδες, είναι ο αριθμός των σημάτων οδικής κυκλοφορίας που θέλει ο χρήστης να αναγνωρίζονται από ένα στιγμιότυπο. Η μπάρα δίπλα στη λέξη εμπιστοσύνη υπάρχει για να επιλέξει ο χρήστης το κατώφλι αξιοπιστίας του ταξινομητή για το αποτέλεσμα που θα δώσει. Η προεπιλογή και για τον αλγόριθμο αναγνώρισης αλλά και για τον ταξινομητή είναι 60%, ωστόσο το κατώφλι του αλγόριθμου αναγνώρισης αντικειμένων δε μπορεί να αλλάξει από τον χρήστη.

Τελευταία επιλογή που μπορεί να πειράξει ο χρήστης είναι οι φωνητικές ειδοποιήσεις, οι οποίες είναι απενεργοποιημένες και μπορούν να ενεργοποιηθούν ή να ξανά απενεργοποιηθούν με το πάτημα του switch στην οθόνη. Η επιλογή αυτή διαλέγει την πρώτη από τις τρείς ή όσες πινακίδες επιλέξει ο χρήστης που αναγνωρίζεται και ταξινομείται και εκφωνεί το τίτλο της πινακίδας. Σε περιπτώσεις εκτάκτου ανάγκης όπως η παραβίαση του ορίου ταχύτητας, το αντίστοιχο φωνητικό μήνυμα μπαίνει σε προτεραιότητα. Πάνω από το πλαίσιο προεπισκόπησης της κάμερας, υπάρχει η ένδειξη της ταχύτητας του χρήστη, έτσι ώστε να μπορεί να γίνει εξακρίβωση με την αντίστοιχη ταχύτητα του καντράν του αυτοκινήτου και δίπλα η πινακίδα, όπως αυτή μεταφέρθηκε στον ταξινομητή. Όλες αυτές οι πληροφορίες είναι συνεχώς διαθέσιμες στο χρήστη, ωστόσο μπορούν να αποκρυφθούν για να μείνει μόνο το πλαίσιο προεπισκόπησης ενεργό, σέρνοντας το μενού προς τα κάτω ή προς τα επάνω αγγίζοντας την οθόνη.

Για να δοκιμάσουμε την συσκευή σε φυσικές συνθήκες, βγήκαμε μία βόλτα στο κέντρο της Θεσσαλονίκης, περάσαμε από δρόμους ταχείας κυκλοφορίας, αλλά και κεντρικούς και τα αποτελέσματα ήταν αρκετά ικανοποιητικά. Οι χρόνοι ανταπόκρισης και αναγνώρισης σημάτων κυμαίνονται ανάμεσα στα 300ms – 800ms. Σημαντικό ρόλο σε αυτό παίζει η κατάσταση της συσκευής και των υλικών που χρησιμοποιεί. Η εφαρμογή αναγνωρίζει σωστά τα σήματα σε αρκετά μέτρα απόσταση σε συνθήκες με καλό φωτισμό και μέτρια ταχύτητα μέχρι περίπου τα 70χλμ την ώρα. Σε καταστάσεις χαμηλού φωτισμού ή ταχύτητας που ξεπερνάει τα 80χλμ η εφαρμογή δεν προλαβαίνει να αναγνωρίσει όλα τα σήματα. Στους Ελληνικούς δρόμους που πολλά σήματα είναι ξεθωριασμένα ή διαστρεβλωμένα η εφαρμογή δυσκολεύτηκε να κάνει σωστή αναγνώριση. Ένα μειονέκτημα το οποίο είναι πολύ σημαντικό είναι η έλλειψη πινακίδων, οι οποίες λόγο της απουσίας τους από το σύνολο δεδομένων και της φύσης της διπλωματικής εργασίας δεν δοθήκαν στον αλγόριθμο ταξινόμησης για να εκπαιδευτεί. Το αποτέλεσμα αυτής της έλλειψης είναι να αναγνωρίζονται λανθασμένα κάποιες πινακίδες με υψηλό ποσοστό αξιοπιστίας, διότι ο τρόπος που κατανέμει ο αλγόριθμος τις πιθανότητες για κάθε κλάση πρέπει να έχει ως άθροισμα το 1. Αυτό σημαίνει πως ο αλγόριθμος δεν ξέρει ότι δεν έχει διδαχθεί κάτι και πρέπει οπωσδήποτε να το ταξινομήσει. Υπάρχουν φυσικά πολλά περιθώρια βελτίωσης τα οποία θα δούμε παρακάτω. Όσο η εξέλιξη της τεχνολογίας προχωράει, αυξάνονται οι δυνατότητες μας και οι ευκαιρίες μας για να χρησιμοποιήσουμε τους διαθέσιμους πόρους προς όφελος μας και αυτό είναι κάτι που θέλουμε να εκμεταλλευτούμε και να εντρυφήσουμε περισσότερο, με τη χρήση της μηχανικής μάθησης και των κινητών τηλεφώνων.



Σχήμα 4.28: Αποτέλεσμα τελικής διεπαφής της εφαρμογής.

Τα αρχεία του Android βρίσκονται στο ηλεκτρονικό μας αποθετήριο του GitHub. Το εκτελέσιμο της εφαρμογής έχει αποθηκευτεί στο Google Drive της ομάδας.

Κεφάλαιο 5ο: **Συμπεράσματα και προτάσεις βελτίωσης**

5.1 Συμπεράσματα

Είμαστε αρκετά ικανοποιημένοι από το τελικό αποτέλεσμα. Είδαμε τις δυνατότητες που μπορεί να προσφέρει η χρήση τεχνητής νοημοσύνης στην καθημερινότητα μας και πως μπορούμε να ωφεληθούμε από αυτή στο χώρο της οδικής κυκλοφορίας. Χρειάστηκε αρκετή μελέτη και πολλά πειράματα ώστε να καταφέρουμε τα βέλτιστα αποτελέσματα, αυτό όμως δε σημαίνει ότι δε θα συνεχίσουμε να ερευνούμε και να διευρύνουμε τις γνώσεις μας.

Η εφαρμογή είναι ικανή να βοηθήσει ένα άτομο που δεν έχει γνώση των βασικών σημάτων οδικής κυκλοφορίας, όπως ένας ποδηλάτης που αναγκάζεται να οδηγήσει το όχημα του εκτός των ποδηλατοδρόμων, ένας νέος οδηγός που ακόμα δεν έχει εξοικειωθεί με την απαιτούμενη παρατηρητικότητα και προσοχή που απαιτείται όσο βρίσκεται στο δρόμο, ή ένας μεγαλύτερος και πιο έμπειρος οδηγός που πολύ συχνά από την υπερβολική αίσθηση αυτοπεποίθησης και εξειδίκευσης στην οδήγηση, μπορεί να αγνοήσει κάποιο νέο σήμα.

Οι περιπτωσιολογικές μελέτες είναι πάρα πολλές και η χρησιμότητα της εφαρμογής μπορεί να επεκταθεί σε πολλά επίπεδα, ωστόσο η εφαρμογή έχει συμπληρωματικό χαρακτήρα και σε καμία περίπτωση δεν μπορεί να αντικαταστήσει τον ανθρώπινο παράγοντα, το κύριο οδηγό οποιουδήποτε οχήματος, ο οποίος πρέπει πάντα να έχει το έλεγχο και την ευθύνη των πράξεων του και να παίρνει αποφάσεις με βάσει τις γνώσεις και τις αισθήσεις του.

5.2 Προτάσεις για βελτίωση

Οι μελλοντικές αλλαγές που θέλουμε να κάνουμε στην εφαρμογή, αφορούν την απόδοσης της, την αξιοπιστία των αποτελεσμάτων της, την ποικιλία και την βελτιστοποίηση της σε θέματα λειτουργικότητας. Αναλυτικότερα, θέλουμε να προσθέσουμε κάποιο API, όπως το ROADS API της Google, με το οποίο αναγνωρίζει τον δρόμο που βρίσκεται ο χρήστης και μας επιτρέπει να έχουμε δεδομένα για τις σημάνσεις του δρόμου και τα όρια ταχύτητας. Με αυτό τον τρόπο θα μπορούμε επαληθεύσουμε τα αποτελέσματα που αναγνώρισαν τα μοντέλα και να παίρνουμε πιο αξιόπιστες αποφάσεις.

Παράλληλα, μπορούμε να συλλέξουμε δεδομένα και φωτογραφίες από τις υπόλοιπες σημάνσεις στους οδικούς δρόμους της Ελλάδος και να εκπαιδεύσουμε έναν νέο ταξινομητή, ο οποίος θα μπορεί να αναγνωρίσει περισσότερα σήματα. Με αυτόν τον τρόπο θα μπορέσουμε να εξασφαλίσουμε πως ο ταξινομητής δε θα εξάγει κάποιο λάθος αποτέλεσμα επειδή δεν έχει εκπαιδευτεί με συγκεκριμένα σήματα.

Για να βελτιώσουμε την εμπειρία του χρήστη με την εφαρμογή, θα μπορούσαμε να προσθέσουμε μια ακόμη διεπαφή με τους χάρτες από το Google Maps. Έτσι ο χρήστης θα μπορεί ταυτόχρονα με την αναγνώριση να χρησιμοποιήσει και την πλοήγηση σε άγνωστες περιοχές για να λαμβάνει περισσότερες οδηγίες. Με έναν αλγόριθμο text-to-speech και ένα μοντέλο αναγνώρισης κειμένου, μπορούμε να προσφέρουμε φωνητικές ειδοποιήσεις για τις διάφορες τοπικές πινακίδες που μπορεί να βρεθούν στο δρόμο, οι οποίες δίνουν οδηγίες για έναν πολιτιστικό ή τουριστικό χώρο και είναι εξίσου σημαντικές.

Τέλος, χρίζεται αναγκαία η έρευνα καλύτερων πρακτικών και γρηγορότερων μοντέλων μηχανικής μάθησης, έτσι ώστε οι λειτουργίες της εφαρμογής να εκτελούνται με μεγαλύτερη ταχύτητα. Όσο η εξέλιξη των μοντέλων προχωράει, αυξάνονται οι επιλογές οι και συνδυασμοί που μπορούμε να κάνουμε, έτσι ώστε να πετύχουμε καλύτερα και αξιόπιστα αποτελέσματα.

5.3 Σύνοψη

Οι εφαρμογές τεχνητής νοημοσύνης και μηχανικής μάθησης μπορούν να δώσουν λύση σε πολλά από τα προβλήματα της καθημερινότητας μας. Μέσα από αυτή την εργασία, ερευνήσαμε τα προβλήματα που αντιμετωπίζουν οι νέοι οδηγοί, τις ανάγκες που θα μπορούσαν να καλυφθούν, και καταφέραμε να φτιάξουμε μια πρωτότυπη εφαρμογή, η οποία μπορεί να λειτουργήσει ως βοηθητικό εργαλείο σε κάποιον που είτε έχει άγνοια των σημάτων οδικής κυκλοφορίας, είτε μαθαίνει τώρα. Τα εργαλεία της μηχανικής μάθησης έχουν τεράστιες δυνατότητες και συνεχώς εξελίσσονται. Σε συνδυασμό με αντικείμενα και συσκευές που χρησιμοποιούμε στην καθημερινότητα μας, όπως το κινητό τηλέφωνο, μπορούμε να κάνουμε καλύτερη τη ζωή των ανθρώπων και να μετατρέψουμε τη δημιουργικότητα μας σε αξία. Ωστόσο, είναι σημαντικό να έχουμε κατά νου, πως κανένας αλγόριθμος, ακόμα, δεν είναι σε θέση να αντικαταστήσει πλήρως την ανθρώπινη νοημοσύνη, ειδικά στον τομέα της οδήγησης και της οδηγικής συμπεριφοράς στους δρόμους, όπου είναι ένα εξαιρετικά ευαίσθητο αλλά και επικίνδυνο θέμα, το οποίο μπορούμε να το βελτιώσουμε και ίσως να προλαμβάνουμε πολλές φορές διάφορα ατυχήματα.

Βιβλιογραφία

[1] A. Ziebinski, R. Cupek, H. Erdogan, και S. Waechter, 'A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion', στο *Computational Collective Intelligence*, Cham, 2016, σσ. 135–146. doi: 10.1007/978-3-319-45246-3_13.

[2] M. Lu, K. Wevers, και R. Van Der Heijden, 'Technical Feasibility of Advanced Driver Assistance Systems (ADAS) for Road Traffic Safety', *Transp. Plan. Technol.*, τ. 28, τχ. 3, σσ. 167–187, Ιουνίου 2005, doi: 10.1080/03081060500120282.

[3] M.-Y. Fu και Y.-S. Huang, 'A survey of traffic sign recognition', στο *2010 International Conference on Wavelet Analysis and Pattern Recognition*, Ιουλίου 2010, σσ. 119–124. doi: 10.1109/ICWAPR.2010.5576425.

[4] A. F. Magnussen, N. Le, L. Hu, και W. Eric Wong, 'A Survey of the Inadequacies in Traffic Sign Recognition Systems for Autonomous Vehicles', *Int. J. Perform. Eng.*, τ. 16, τχ. 10, σ. 1588, 2020, doi: 10.23940/ijpe.20.10.p10.15881597.

[5] C. Liu, S. Li, F. Chang, και Y. Wang, 'Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives', *IEEE Access*, τ. 7, σσ. 86578–86596, 2019, doi: 10.1109/ACCESS.2019.2924947.

[6] H. Nguyen, 'Fast Traffic Sign Detection Approach Based on Lightweight Network and Multilayer Proposal Network', *J. Sens.*, τ. 2020, σ. e8844348, Ιουνίου 2020, doi: 10.1155/2020/8844348.

[7] S. Ren, K. He, R. Girshick, και J. Sun, 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks', *ArXiv150601497 Cs*, Ιανουαρίου 2016, Ημερομηνία πρόσβασης: 18 Οκτώβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1506.01497

[8] W. Liu *κ.ά.*, 'SSD: Single Shot MultiBox Detector', *ArXiv151202325 Cs*, τ. 9905, σσ. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[9] J. Redmon, S. Divvala, R. Girshick, και A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', *ArXiv150602640 Cs*, Μαΐου 2016, Ημερομηνία πρόσβασης: 7 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1506.02640

[10] 'Planes, Trains and Automobiles: The History of Transportation', *Leland-West Insurance - www.lelandwest.com*. https://www.lelandwest.com/planes-trains-automobiles-the-history-of-transportation.cfm (ημερομηνία πρόσβασης 12 Σεπτέμβριος 2021).

[11] 'Archive:Road safety statistics - characteristics at national and regional level'. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Archive:Road_safety_statistics_-_characteristics_at_national_and_regional_level (ημερομηνία πρόσβασης 12 Σεπτέμβριος 2021).

[12] S. Maldonado Bascón, J. Acevedo Rodríguez, S. Lafuente Arroyo, A. Fernndez Caballero, και F. López-Ferreras, 'An optimization on pictogram identification for the road-sign recognition task using SVMs', *Comput. Vis. Image Underst.*, τ. 114, τχ. 3, σσ. 373–383, Μαρτίου 2010, doi: 10.1016/j.cviu.2009.12.002.

[13] 'About Tesla | Tesla'. https://www.tesla.com/about (ημερομηνία πρόσβασης 6 Σεπτέμβριος 2021).

[14] 'Artificial Intelligence & Autopilot', *Tesla*. https://www.tesla.com/AI (ημερομηνία πρόσβασης 6 Σεπτέμβριος 2021).

[15] 'Σύστημα Toyota Safety Sense | Toyota Hellas'. https://www.toyota.gr/world-of-toyota/safety/toyota-safety-sense (ημερομηνία πρόσβασης 6 Σεπτέμβριος 2021).

[16] 'Object detection | TensorFlow Lite', *TensorFlow*. https://www.tensorflow.org/lite/examples/object_detection/overview (ημερομηνία πρόσβασης 12 Σεπτέμβριος 2021).

[17] V. Yatsenko, *Car Assistant Android*. 2021. Ημερομηνία πρόσβασης: 12 Σεπτέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/VladYatsenko/car-assistant-android

[18] N. Chopda, *Traffic Light Detection And Color Recognition*. 2021. Ημερομηνία πρόσβασης: 12 Σεπτέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/nileshchopda/Traffic-Light-Detection-And-Color-Recognition

[19] 'Arthur Samuel | IEEE Computer Society'. https://www.computer.org/profiles/arthur-samuel/ (ημερομηνία πρόσβασης 14 Ιανουάριος 2022).

[20] M. Xue και C. Zhu, ‘A Study and Application on Machine Learning of Artificial Intellligence’, στο *2009 International Joint Conference on Artificial Intelligence*, Απριλίου 2009, σσ. 272–274. doi: 10.1109/JCAI.2009.55.
[21] D. B. Fogel, ‘8 - Samuel’s Learning Machine’, στο *Blondie24*, D. B. Fogel, Επιμ. San Francisco: Morgan Kaufmann, 2002, σσ. 129–149. doi: 10.1016/B978-155860783-5/50009-X.
[22] G. Hinton και T. J. Sejnowski, *Unsupervised Learning: Foundations of Neural Computation*. MIT Press, 1999.
[23] ‘What is Unsupervised Learning? | IBM’. https://www.ibm.com/cloud/learn/unsupervised-learning (ημερομηνία πρόσβασης 15 Νοέμβριος 2021).
[24] K. Fukushima, ‘Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position’, *Biol. Cybern.*, τ. 36, τχ. 4, σσ. 193–202, Απριλίου 1980, doi: 10.1007/BF00344251.
[25] Y. LeCun *κ.ά.*, ‘Backpropagation Applied to Handwritten Zip Code Recognition’, *Neural Comput.*, τ. 1, τχ. 4, σσ. 541–551, Δεκεμβρίου 1989, doi: 10.1162/neco.1989.1.4.541.
[26] D. H. Hubel και T. N. Wiesel, ‘Receptive fields of single neurones in the cat’s striate cortex’, *J. Physiol.*, τ. 148, τχ. 3, σσ. 574–591, 1959, doi: 10.1113/jphysiol.1959.sp006308.
[27] Κ. Διαμαντάρας και Δ. Μπότσης, ‘Συνελικτικά νευρωνικά δίκτυα’, στο *Μηχανική Μάθηση*, Εκδόσεις Κλειδάριθμος, 2019, σσ. 276–315.
[28] T. Szandała, ‘Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks’, στο *Bio-inspired Neurocomputing*, τ. 903, A. K. Bhoi, P. K. Mallick, C.-M. Liu, και V. E. Balas, Επιμ. Singapore: Springer Singapore, 2021, σσ. 203–224. doi: 10.1007/978-981-15-5495-7_11.
[29] B. Xu, N. Wang, T. Chen, και M. Li, ‘Empirical Evaluation of Rectified Activations in Convolutional Network’, *ArXiv150500853 Cs Stat*, Νοεμβρίου 2015, Ημερομηνία πρόσβασης: 23 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1505.00853
[30] A. F. Agarap, ‘Deep Learning using Rectified Linear Units (ReLU)’, Μαρτίου 2018.
[31] Y. LeCun, L. Bottou, Y. Bengio, και P. Ha, ‘Gradient-Based Learning Applied to Document Recognition’, σ. 46, 1998.
[32] A. Krizhevsky, I. Sutskever, και G. E. Hinton, ‘ImageNet classification with deep convolutional neural networks’, *Commun. ACM*, τ. 60, τχ. 6, σσ. 84–90, Μαΐου 2017, doi: 10.1145/3065386.
[33] M. D. Zeiler και R. Fergus, ‘Visualizing and Understanding Convolutional Networks’, *ArXiv13112901 Cs*, Νοεμβρίου 2013, Ημερομηνία πρόσβασης: 1 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1311.2901
[34] C. Szegedy *κ.ά.*, ‘Going Deeper with Convolutions’, *ArXiv14094842 Cs*, Σεπτεμβρίου 2014, Ημερομηνία πρόσβασης: 2 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1409.4842
[35] K. Simonyan και A. Zisserman, ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’, *ArXiv14091556 Cs*, Απριλίου 2015, Ημερομηνία πρόσβασης: 2 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1409.1556
[36] K. He, X. Zhang, S. Ren, και J. Sun, ‘Deep Residual Learning for Image Recognition’, *ArXiv151203385 Cs*, Δεκεμβρίου 2015, Ημερομηνία πρόσβασης: 3 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1512.03385
[37] J. Redmon και A. Farhadi, ‘YOLO9000: Better, Faster, Stronger’, *ArXiv161208242 Cs*, Δεκεμβρίου 2016, Ημερομηνία πρόσβασης: 7 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1612.08242
[38] J. Redmon και A. Farhadi, ‘YOLOv3: An Incremental Improvement’, *ArXiv180402767 Cs*, Απριλίου 2018, Ημερομηνία πρόσβασης: 7 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1804.02767
[39] A. Bochkovskiy, C.-Y. Wang, και H.-Y. M. Liao, ‘YOLOv4: Optimal Speed and Accuracy of Object Detection’, *ArXiv200410934 Cs Eess*, Απριλίου 2020, Ημερομηνία πρόσβασης: 7 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/2004.10934
[40] *ultralytics/yolov5*. Ultralytics, 2021. Ημερομηνία πρόσβασης: 17 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ultralytics/yolov5

[41] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, και A. Zisserman, 'The Pascal Visual Object Classes Challenge: A Retrospective', *Int. J. Comput. Vis.*, τ. 111, τχ. 1, σσ. 98–136, Ιανουαρίου 2015, doi: 10.1007/s11263-014-0733-5.
[42] S. Ioffe και C. Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift', σ. 11.
[43] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, και S. Belongie, 'Feature Pyramid Networks for Object Detection', *ArXiv161203144 Cs*, Απριλίου 2017, Ημερομηνία πρόσβασης: 9 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1612.03144
[44] Alexey, *Yolo v4, v3 and v2 for Windows and Linux*. 2022. Ημερομηνία πρόσβασης: 11 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/AlexeyAB/darknet
[45] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, και J.-W. Hsieh, 'CSPNet: A New Backbone that can Enhance Learning Capability of CNN', *ArXiv191111929 Cs*, Νοεμβρίου 2019, Ημερομηνία πρόσβασης: 15 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1911.11929
[46] K. He, X. Zhang, S. Ren, και J. Sun, 'Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition', *ArXiv14064729 Cs*, τ. 8691, σσ. 346–361, 2014, doi: 10.1007/978-3-319-10578-9_23.
[47] S. Liu, L. Qi, H. Qin, J. Shi, και J. Jia, 'Path Aggregation Network for Instance Segmentation', *ArXiv180301534 Cs*, Σεπτεμβρίου 2018, Ημερομηνία πρόσβασης: 15 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1803.01534
[48] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, και Y. Yoo, 'CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features', Μαΐου 2019, Ημερομηνία πρόσβασης: 14 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://arxiv.org/abs/1905.04899v2
[49] G. Ghiasi, T.-Y. Lin, και Q. V. Le, 'DropBlock: A regularization method for convolutional networks', Οκτωβρίου 2018, Ημερομηνία πρόσβασης: 14 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://arxiv.org/abs/1810.12890v1
[50] D. Misra, 'Mish: A Self Regularized Non-Monotonic Activation Function', *ArXiv190808681 Cs Stat*, Αυγούστου 2020, Ημερομηνία πρόσβασης: 15 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1908.08681
[51] M. Tan, R. Pang, και Q. V. Le, 'EfficientDet: Scalable and Efficient Object Detection', *ArXiv191109070 Cs Eess*, Ιουλίου 2020, Ημερομηνία πρόσβασης: 15 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1911.09070
[52] S. Woo, J. Park, J.-Y. Lee, και I. S. Kweon, 'CBAM: Convolutional Block Attention Module', *ArXiv180706521 Cs*, Ιουλίου 2018, Ημερομηνία πρόσβασης: 15 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1807.06521
[53] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, και D. Ren, 'Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression', *ArXiv191108287 Cs*, Νοεμβρίου 2019, Ημερομηνία πρόσβασης: 15 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1911.08287
[54] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, και S. Savarese, 'Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression', στο *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, Ιουνίου 2019, σσ. 658–666. doi: 10.1109/CVPR.2019.00075.
[55] D. P. Kingma και J. Ba, 'Adam: A Method for Stochastic Optimization', *ArXiv14126980 Cs*, Ιανουαρίου 2017, Ημερομηνία πρόσβασης: 25 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1412.6980
[56] 'BCEWithLogitsLoss — PyTorch master documentation'. https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html (ημερομηνία πρόσβασης 26 Ιανουάριος 2022).
[57] 'PyTorch'. https://www.pytorch.org (ημερομηνία πρόσβασης 11 Ιανουάριος 2022).
[58] 'Keras: the Python deep learning API'. https://keras.io/ (ημερομηνία πρόσβασης 11 Ιανουάριος 2022).
[59] M. Abadi *κ.ά.*, 'TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems', σ. 19.

[60] ‘Cloud Tensor Processing Units (TPUs)’, *Google Cloud*. https://cloud.google.com/tpu/docs/tpus (ημερομηνία πρόσβασης 4 Δεκέμβριος 2021).
[61] ‘What Is CUDA | NVIDIA Official Blog’, *The Official NVIDIA Blog*, 10 Σεπτέμβριος 2012. https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/ (ημερομηνία πρόσβασης 29 Νοέμβριος 2021).
[62] ‘CUDA Zone’, *NVIDIA Developer*, 18 Ιούλιος 2017. https://developer.nvidia.com/cuda-zone (ημερομηνία πρόσβασης 29 Νοέμβριος 2021).
[63] ‘SYCL - C++ Single-source Heterogeneous Programming for Acceleration Offload’, *The Khronos Group*, 20 Ιανουάριος 2014. https://www.khronos.org/sycl/ (ημερομηνία πρόσβασης 29 Νοέμβριος 2021).
[64] ‘Wayback Machine’, 9 Αύγουστος 2017. https://web.archive.org/web/20170809231307/http://people.cs.umass.edu/~ramesh/Site/PUBLICATIONS_files/DMPPSW02.pdf (ημερομηνία πρόσβασης 5 Δεκέμβριος 2021).
[65] E. H.-L. Updated: 2018-12-27T08:37:12+00:00, ‘What is Edge Computing: The Network Edge Explained’, *Cloudwards*, 31 Δεκέμβριος 2018. https://www.cloudwards.net/what-is-edge-computing/ (ημερομηνία πρόσβασης 5 Δεκέμβριος 2021).
[66] ‘Using the SavedModel format | TensorFlow Core’, *TensorFlow*. https://www.tensorflow.org/guide/saved_model (ημερομηνία πρόσβασης 26 Ιανουάριος 2022).
[67] TensorFlow, ‘What’s coming in TensorFlow 2.0’, *TensorFlow*, 14 Ιανουάριος 2019. https://medium.com/tensorflow/whats-coming-in-tensorflow-2-0-d3663832e9b8 (ημερομηνία πρόσβασης 5 Δεκέμβριος 2021).
[68] ‘A Basic Introduction to TensorFlow Lite | by Renu Khandelwal | Towards Data Science’. https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292 (ημερομηνία πρόσβασης 5 Δεκέμβριος 2021).
[69] ‘Model optimization | TensorFlow Lite’, *TensorFlow*. https://www.tensorflow.org/lite/performance/model_optimization (ημερομηνία πρόσβασης 5 Δεκέμβριος 2021).
[70] ‘Trim insignificant weights | TensorFlow Model Optimization’, *TensorFlow*. https://www.tensorflow.org/model_optimization/guide/pruning (ημερομηνία πρόσβασης 5 Δεκέμβριος 2021).
[71] ‘Post-training integer quantization | TensorFlow Lite’, *TensorFlow*. https://www.tensorflow.org/lite/performance/post_training_integer_quant (ημερομηνία πρόσβασης 30 Ιανουάριος 2022).
[72] ‘Android Platform’, *Android Developers*. https://developer.android.com/about (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[73] ‘IDC - Smartphone Market Share - Market Share’, *IDC: The premier global market intelligence company*. https://www.idc.com/promo/smartphone-market-share (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[74] ‘Android | Definition, History, & Facts | Britannica’. https://www.britannica.com/technology/Android-operating-system (ημερομηνία πρόσβασης 19 Δεκέμβριος 2021).
[75] ‘Android 1.5 Platform Highlights’, *Android Developers*. https://developer.android.com/about/versions/android-1.5-highlights (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[76] ‘Android 1.6 Platform Highlights’, *Android Developers*. https://developer.android.com/about/versions/android-1.6-highlights (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[77] ‘Android 2.0 Platform Highlights’, *Android Developers*. https://developer.android.com/about/versions/android-2.0-highlights (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[78] ‘EXCLUSIVE: AndroidPolice.com’s Nexus One Is Running Android 2.2 Froyo. How Fast Is It Compared To 2.1? Oh, Only About 450% Faster’, *Android Police*, 12 Μάιος 2010. https://www.androidpolice.com/2010/05/11/exclusive-androidpolice-coms-nexus-one-is-running-android-2-2-froyo-how-fast-is-it-compared-to-2-1-oh-only-about-450-faster/ (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).

[79] C. Sorrel, ‘Android 2.2 'Froyo' Features USB, Wi-Fi Tethering’, *Wired*. Ημερομηνία πρόσβασης: 12 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.wired.com/2010/05/android-22-froyo-features-usb-wi-fi-tethering/
[80] ‘Android 2.2 and developers goodies.’, *Android Developers Blog*. https://android-developers.googleblog.com/2010/05/android-22-and-developers-goodies.html (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[81] F. Graham, ‘What’s new in Google’s Android 2.3 Gingerbread?’, *CNET*. https://www.cnet.com/tech/mobile/whats-new-in-googles-android-2-3-gingerbread/ (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[82] ‘What is Android 3.0 Honeycomb? - Definition from WhatIs.com’, *SearchMobileComputing*. https://searchmobilecomputing.techtarget.com/definition/Android-30-Honeycomb (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[83] ‘Ice Cream Sandwich’, *Android Developers*. https://developer.android.com/about/versions/android-4.0-highlights (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[84] ‘Android – 4.3 Jelly Bean’, *Android*. https://www.android.com/versions/jelly-bean-4-3/ (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[85] ‘Android – 4.4 KitKat’, *Android*. https://www.android.com/versions/kit-kat-4-4/ (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[86] ‘Android 5.1.1 (LMY47V) officially uploaded to AOSP and here’s the full changelog, Nexus Player factory images also available – Phandroid’, 21 Απρίλιος 2015. http://Android%205.1.1%20Lollipop,%20Nexus%20Player,%20AOSP, (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[87] ‘Android – Marshmallow’, *Android*. https://www.android.com/versions/marshmallow-6-0/ (ημερομηνία πρόσβασης 13 Δεκέμβριος 2021).
[88] ‘Android – Nougat’, *Android*. https://www.android.com/versions/nougat-7-0/ (ημερομηνία πρόσβασης 13 Δεκέμβριος 2021).
[89] ‘Android – 8.0 Oreo’, *Android*. https://www.android.com/versions/oreo-8-0/ (ημερομηνία πρόσβασης 14 Δεκέμβριος 2021).
[90] ‘Android 9 Pie’, *Android*. https://www.android.com/versions/pie-9-0/ (ημερομηνία πρόσβασης 14 Δεκέμβριος 2021).
[91] ‘Android 10’, *Android*. https://www.android.com/android-10/ (ημερομηνία πρόσβασης 14 Δεκέμβριος 2021).
[92] ‘Behavior changes: all apps | Android 12’, *Android Developers*. https://developer.android.com/about/versions/12/behavior-changes-all (ημερομηνία πρόσβασης 12 Δεκέμβριος 2021).
[93] ‘Download Android Studio and SDK tools’, *Android Developers*. https://developer.android.com/studio (ημερομηνία πρόσβασης 7 Ιανουάριος 2022).
[94] ‘What is an IDE?’ https://www.redhat.com/en/topics/middleware/what-is-ide (ημερομηνία πρόσβασης 7 Ιανουάριος 2022).
[95] X. Zhang, F. Breitinger, E. Luechinger, και S. O’Shaughnessy, ‘Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations’, *Forensic Sci. Int. Digit. Investig.*, τ. 39, σ. 301285, Δεκεμβρίου 2021, doi: 10.1016/j.fsidi.2021.301285.
[96] ‘SDK Tools release notes’, *Android Developers*. https://developer.android.com/studio/releases/sdk-tools (ημερομηνία πρόσβασης 8 Ιανουάριος 2022).
[97] ‘What is Gradle?’ https://docs.gradle.org/current/userguide/what_is_gradle.html (ημερομηνία πρόσβασης 8 Ιανουάριος 2022).
[98] ‘The Activity Lifecycle’, *Android Developers*. https://developer.android.com/guide/components/activities/activity-lifecycle (ημερομηνία πρόσβασης 9 Ιανουάριος 2022).
[99] ‘Activity’, *Android Developers*. https://developer.android.com/reference/android/app/Activity (ημερομηνία πρόσβασης 9 Ιανουάριος 2022).

[100] ‘Fragments’, *Android Developers*. https://developer.android.com/guide/fragments (ημερομηνία πρόσβασης 9 Ιανουάριος 2022).
[101] ‘Hello World’, *GitHub Docs*. https://docs.github.com/en/get-started/quickstart/hello-world (ημερομηνία πρόσβασης 9 Ιανουάριος 2022).
[102] ‘java_book_EMP.pdf’. Ημερομηνία πρόσβασης: 12 Δεκέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://people.iee.ihu.gr/~sfetsos/java_book_EMP.pdf
[103] ‘Welcome to Python.org’, *Python.org*. https://www.python.org/ (ημερομηνία πρόσβασης 13 Ιανουάριος 2022).
[104] ‘Mapillary’. https://www.mapillary.com/dataset/trafficsign (ημερομηνία πρόσβασης 28 Ιανουάριος 2022).
[105] ‘German Traffic Sign Benchmarks’. https://benchmark.ini.rub.de/gtsrb_dataset.html (ημερομηνία πρόσβασης 1 Φεβρουάριος 2022).
[106] J. Stallkamp, M. Schlipsing, J. Salmen, και C. Igel, ‘Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition’, *Neural Netw.*, τ. 32, σσ. 323–332, Αυγούστου 2012, doi: 10.1016/j.neunet.2012.02.016.
[107] A. Møgelmose, D. Liu, και M. M. Trivedi, ‘Detection of U.S. Traffic Signs’, *IEEE Trans. Intell. Transp. Syst.*, τ. 16, τχ. 6, σσ. 3116–3125, Δεκεμβρίου 2015, doi: 10.1109/TITS.2015.2433019.
[108] R. Timofte, K. Zimmermann, και L. Van Gool, ‘Multi-view traffic sign detection, recognition, and 3D localisation’, *Mach. Vis. Appl.*, τ. 25, τχ. 3, σσ. 633–647, Απριλίου 2014, doi: 10.1007/s00138-011-0391-3.
[109] R. Belaroussi, P. Foucher, J.-P. Tarel, B. Soheilian, P. Charbonnier, και N. Paparoditis, ‘Road Sign Detection in Images: A Case Study’, στο *2010 20th International Conference on Pattern Recognition*, Istanbul, Turkey, Αυγούστου 2010, σσ. 484–488. doi: 10.1109/ICPR.2010.1125.
[110] N. Paparoditis *κ.ά.*, ‘Stereopolis II: A multi-purpose and multi-sensor 3D mobile mapping system for street visualisation and 3D metrology’, *Rev. Franaise Photogrammétrie Télédétection*, τ. 200, σσ. 69–79, Ιανουαρίου 2012, doi: 10.52638/rfpt.2012.63.
[111] S. Segvic *κ.ά.*, ‘A computer vision assisted geoinformation inventory for traffic infrastructure’, στο *13th International IEEE Conference on Intelligent Transportation Systems*, Funchal, Madeira Island, Portugal, Σεπτεμβρίου 2010, σσ. 66–73. doi: 10.1109/ITSC.2010.5624979.
[112] V. I. Shakhuro και A. S. Konushin, ‘Russian traffic sign images dataset’, *Comput. Opt.*, τ. 40, τχ. 2, σσ. 294–300, Ιανουαρίου 2016, doi: 10.18287/2412-6179-2016-40-2-294-300.
[113] C. Grigorescu και N. Petkov, ‘Distance sets for shape filters and shape recognition’, *IEEE Trans. Image Process.*, τ. 12, τχ. 10, σσ. 1274–1286, Οκτωβρίου 2003, doi: 10.1109/TIP.2003.816010.
[114] F. Larsson και M. Felsberg, ‘Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition’, στο *Image Analysis*, Berlin, Heidelberg, 2011, σσ. 238–249. doi: 10.1007/978-3-642-21227-7_23.
[115] C. G. Serna και Y. Ruichek, ‘Classification of Traffic Signs: The European Dataset’, *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2884826.
[116] J. Fang, *zldrobit/yolov5*. 2022. Ημερομηνία πρόσβασης: 25 Ιανουάριος 2022. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/zldrobit/yolov5