ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

ΑΝΑΓΝΩΡΙΣΗ ΚΑΙ ΕΝΤΟΠΙΣΜΟΣ ΚΡΑΝΟΥΣ ΜΕ ΤΗ ΧΡΗΣΗ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΟΡΑΣΗΣ ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ CORAL TPU

Φοιτητής: Μιχαλακέας Σωτήριος
ΑΜ: 19387171

Επιβλέπων Καθηγητής

Κάχρης Χριστόφορος
Επίκουρος Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΣΕΠΤΕΜΒΡΙΟΣ 2024

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

HELMET DETECTION USING COMPUTER VISION ON THE CORAL TPU PLATFORM

Student: Michalakeas Sotirios
Registration Number: 19387171

Supervisor

Kachris Christoforos
Assistant Professor

ATHENS-EGALEO, September 2024

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Χριστόφορος Κάχρης, Επίκουρος Καθηγητής	Ευστάθιος Κυριάκης - Μπιτζάρος, Καθηγητής	Μαρία Ραγκούση, Καθηγήτρια
ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ Digitally signed by ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ Date: 2024.10.11 11:02:48 +03'00' (Υπογραφή)	Efstathios Kyriakis-Bitzaros 2024.10.10 18:09:02 +03'00' (Υπογραφή)	MARIA RAGKOUSI Digitally signed by MARIA RAGKOUSI Date: 2024.10.11 10:20:20 +03'00' (Υπογραφή)

Copyright © Με επιφύλαξη παντός δικαιώματος. All rights reserved.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ και Μιχαλακέας Σωτήριος,

Σεπτέμβριος, 2024

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον/την συγγραφέα του και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις θέσεις του επιβλέποντος, της επιτροπής εξέτασης ή τις επίσημες θέσεις του Τμήματος και του Ιδρύματος.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος/η Μιχαλακέας Σωτήριος του Ηλίας, με αριθμό μητρώου 19387171 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

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

Ο Δηλών

Μιχαλακέας Σωτήριος

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ολόψυχα τον επιβλέποντα καθηγητή Κύριο Κάχρη για την αμέριστη ηθική και έμπρακτη υποστήριξη την οποία μου παρείχε καθ' όλο το διάστημα εκπόνησης της εργασίας, τα λοιπά μέλη της επιτροπής αξιολόγησης, καθώς και όλους τους πανεπιστημιακούς δασκάλους μου για τις επιστημονικές γνώσεις που μου παρείχαν, οι οποίες αποτέλεσαν τη βάση της παρούσης.

Επίσης ευχαριστώ θερμά την οικογένειά μου και τους φίλους μου για την ηθική τους συμπαράσταση και την υπομονή τους, κυρίως στις δύσκολες περιόδους του τελευταίου εξαμήνου.

Περίληψη

Στην Διπλωματική εργασία, παρουσιάζεται ένα ενσωματωμένο σύστημα αναγνώρισης και εντοπισμού κράνους, χρησιμοποιώντας την υπολογιστική όραση. Συγκεκριμένα, το Object Detector εφαρμόζεται σε μία πλατφόρμα Google Coral με επιταχυντή Edge TPU, ώστε να επιτυγχάνεται ο εντοπισμός των αντικειμένων σε πραγματικό χρόνο, με μεγαλύτερη ταχύτητα και με σχετικά χαμηλή κατανάλωση ενέργειας. Τα αποτελέσματα των εντοπισμένων κλάσεων μεταφέρονται ως δυαδικά σήματα (λογικό ‘0’ ή λογικό ‘1’) σε έναν εξωτερικό μικροελεγκτή ο οποίος τα επεξεργάζεται και οδηγεί μία LCD οθόνη και ένα παθητικό piezo buzzer, ώστε να παράγεται θετική ή αρνητική οπτική και ακουστική ανατροφοδότηση στους οδηγούς ή συνεπιβάτες. Η εκπαίδευση των μοντέλων γίνεται μέσω της μεθόδου transfer learning που επιτρέπει το Tensorflow Object Detection API, στο προγραμματιστικό περιβάλλον Google Colab. Επιλέχθηκαν δύο αρχιτεκτονικές από το Tensorflow 2 Detection Model Zoo με διαφορετικές δυνατότητες, εκπαιδεύτηκαν με το ίδιο σύνολο δεδομένων και μετά τις δοκιμές τους στο Google Coral, συγκρίθηκαν και αξιολογήθηκαν οι επιδόσεις τους ως προς την ακρίβειά τους και την ταχύτητά τους. Στην συνέχεια, γίνεται λεπτομερής περιγραφή της δημιουργίας του συστήματος ανατροφοδότησης. Τέλος, θα συζητηθούν μέθοδοι για την πιθανή βελτιστοποίηση των μοντέλων και θα παρουσιαστεί και η κατανάλωση ενέργειας που έχει το καθένα.

Λέξεις – κλειδιά

Ενσωματωμένο Σύστημα, Μηχανική Μάθηση, Υπολογιστική Όραση, Εντοπισμός Κράνους, Google Coral, EdgeTPU, Tensorflow Lite, Tensorflow Object Detection, Google Colab

Abstract

In this Diploma Thesis, an embedded system is presented, which utilizes machine learning and computer vision to detect helmets. Furthermore, the Object Detector is deployed in the Google Coral platform, using the EdgeTPU accelerator to achieve real time detection, by increasing the model's detection speed while using relatively low power. The detection results are transferred by logic signals (logical '0' or logical '1') to an external microcontroller that processes them and drives an LCD and a piezo buzzer to create positive or negative visual and auditory feedback to the drivers and passengers. The method in which the models are trained is transfer learning, which can be achieved by using the Tensorflow Object Detection API in the Google Colab programming environment. Two different model architectures with different properties are chosen from the Tensorflow 2 Detection Model Zoo, trained with the same image dataset and after the testing process in the Google Coral Dev Board, their performance is evaluated and compared, based on their accuracy and speed. Also, a detailed description of the creation process of the external feedback system is provided. Finally, some model optimization methods are discussed, and the power consumption of each model will be presented.

Keywords

Embedded System, Machine Learning, Computer Vision, Helmet Detection, Google Coral, EdgeTPU, Tensorflow Lite, Tensorflow Object Detection, Google Colab

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

ΕΙΣΑΓΩΓΗ

Είναι πλέον ευρέως γνωστό ότι η οδική ασφάλεια είναι άκρως σημαντική για την ορθή λειτουργία μιας σύγχρονης κοινωνίας. Ωστόσο πολλοί οδηγοί, συνεχίζουν να παραβιάζουν τον Κ.Ο.Κ (κώδικα οδικής κυκλοφορίας) καθημερινά, βάζοντας σε κίνδυνο όχι μόνο τους συμπολίτες τους, αλλά και την ίδια την σωματική ακεραιότητά τους. Σύμφωνα με έρευνα του Ευρωπαϊκού Συμβουλίου Ασφάλειας των Μεταφορών (ETSC), το 2023, τα τροχαία ατυχήματα που έλαβαν χώρα στα 27 κράτη μέλη της Ε.Ε (Ευρωπαϊκής Ένωσης), προκάλεσαν τον θάνατο 20.418 ατόμων. Ο αριθμός αυτός είναι μειωμένος κατά 1% συγκριτικά με τους θανάτους που σημειώθηκαν το 2022, με απώτερο σκοπό το ποσοστό αυτό να φτάσει το 50% μέχρι το 2030, πράγμα που φαίνεται δύσκολο βάσει των δεδομένων των τελευταίων ετών [1].

Στην Ελλάδα, καταγράφηκαν 621 θάνατοι το 2023, με την ετήσια μείωση του αριθμού αυτού να μην δείχνει ενθαρρυντική. Ενώ υπάρχει μείωση των νεκρών κατά 5% σε σχέση με το 2022, ο αριθμός αυτός έχει μικρή σημασία όταν το 2021 οι νεκροί ήταν 624, το 2020 ήταν 584 και το 2019 ήταν 688, με αποτέλεσμα ο μέσος όρος μείωσης των νεκρών από το 2019 έως το 2023 να κυμαίνεται στο 9,7%, με τον αντίστοιχο ευρωπαϊκό μέσο όρο να είναι 10,3%. Επιπλέον, μόλις τέσσερις χώρες της Ε.Ε κατέγραψαν υψηλότερο αριθμό νεκρών ανά εκατομμύριο κατοίκους κατά την διάρκεια του 2023 [1].

Ο Γιώργος Γιαννής, εκπρόσωπος της Ελλάδας στο ETSC, διευθυντής του τμήματος Χωροταξίας Μεταφορών του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΜΠ) και επικεφαλής του Παρατηρητηρίου Οδικής Ασφάλειας του ΕΜΠ, ανέφερε ότι «στα ατυχήματα υπάρχουν οι πέντε δολοφόνοι. Η ταχύτητα, το αλκοόλ, η απόσπαση προσοχής, η μη χρήση ζώνης και η μη χρήση κράνους. Σε όλους αυτούς τους δείκτες βρισκόμαστε στις τελευταίες θέσεις». Αναφέρει μάλιστα ότι το πρώτο που πρέπει να βελτιωθεί είναι η μη χρήση του κράνους μοτοσικλέτας, καθώς οι οδηγοί φοράνε το κράνος τους κατά 80% ενώ οι συνεπιβάτες κατά 65%, με το αντίστοιχο ευρωπαϊκό ποσοστό να κυμαίνεται στο 95% με 100%. Στην συνέχεια τονίζει πως «Αν ανεβάσουμε κι εμείς αυτά τα ποσοστά στο 95%, τότε από τους 211 νεκρούς μοτοσικλετιστές που είχαμε το 2023, θα γλιτώσουμε τουλάχιστον τους μισούς. Στην Ελλάδα, το 32% των νεκρών από τροχαία είναι μοτοσικλετιστές. Το αντίστοιχο ποσοστό στην Ευρώπη κυμαίνεται κατά μέσο όρο στο 18%» [1].

Αντικείμενο της διπλωματικής εργασίας

Το αντικείμενο της διπλωματικής εργασίας είναι η δημιουργία ενός ενσωματωμένου συστήματος, χαμηλής κατανάλωσης ενέργειας, το οποίο θα εντοπίζει σε πραγματικό χρόνο εάν οι οδηγοί/επιβάτες μηχανών/ποδηλάτων χρησιμοποιούν κράνος και θα παρέχει οπτικά και ηχητικά μηνύματα θετικής ή αρνητικής ανατροφοδότησης. Για να γίνει αυτό, θα αξιοποιηθεί η τεχνολογία της υπολογιστικής όρασης σε συνδυασμό με την μηχανική μάθηση, προκειμένου να μπορεί να προσαρμόζεται σε διαφορετικές συνθήκες.

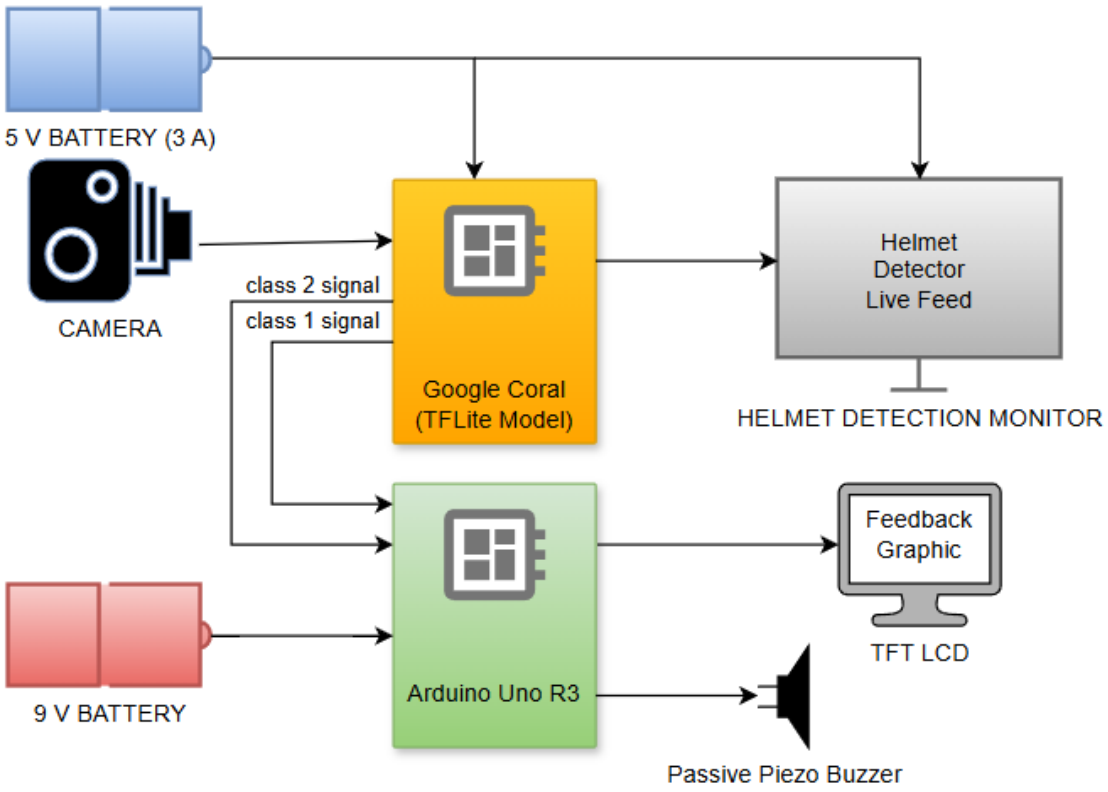
Σκοπός και στόχοι

Είναι σαφές, βάσει των ερευνών που παρουσιάστηκαν στην εισαγωγή, ότι υπάρχει ανάγκη για ένα σύστημα ελέγχου και υπενθύμισης χρήσης κράνους για τους μοτοσικλετιστές και ποδηλάτες, το οποίο θα μπορεί να τοποθετηθεί σε μία πληθώρα χώρων. Σκοπός της διπλωματικής εργασίας είναι η υλοποίηση της ιδέας αυτής, κατασκευάζοντας ένα φορητό ενσωματωμένο σύστημα το οποίο μέσω μηχανικής μάθησης, θα εντοπίζει αν ένας/μία οδηγός/συνεπιβάτης μηχανής ή ποδηλάτου φοράει/δεν φοράει κράνος και θα παρέχει την απαραίτητη ανατροφοδότηση.

Μέσω του συστήματος αυτού, υπάρχει αισιοδοξία ότι θα βελτιωθεί το ποσοστό των ατόμων που φοράνε κράνος στην Ελλάδα, μειώνοντας παράλληλα τον αριθμό των θανάτων των μοτοσικλετιστών και ποδηλατών, καθώς πρόκειται για ένα σύστημα που βασίζεται στην επιβράβευση/τιμωρία. Οι νομοταγείς πολίτες θα επιβραβεύονται με θετική ανατροφοδότηση και οι παραβάτες θα λαμβάνουν προειδοποιητικά ηχητικά και οπτικά μηνύματα.

Μεθοδολογία

Η εφαρμογή αυτή έγινε πάνω στο Google Coral Dev Board το οποίο εκτελεί ένα μοντέλο για object detection, εκπαιδευμένο με την μέθοδο Transfer Learning, με ένα ειδικά δημιουργημένο σύνολο δεδομένων, στο προγραμματιστικό περιβάλλον Google Colab, μέσω του Tensorflow Object Detection API και ανάλογα με το αποτέλεσμα του εντοπισμού, δηλαδή την κλάση που ανιχνεύει από τις εικόνες που παρέχονται από την κάμερα, δίνει τα κατάλληλα σήματα σε έναν εξωτερικό μικροελεγκτή (Arduino Uno R3), το οποίο με την σειρά του ελέγχει το σύστημα παροχής οπτικής και ηχητικής ανατροφοδότησης. Το σύστημα αυτό αποτελείται από ένα TFT LCD (Thin-Film-Transistor Liquid-Crystal Display) και ένα παθητικό piezo buzzer.



Εικόνα 1: Το block diagram του συστήματος

Καινοτομία

Η ενίσχυση της ασφάλειας μέσω ενός συστήματος εντοπισμού κρανών με την χρήση υπολογιστικής όρασης και μηχανικής μάθησης γίνεται ολοένα και πιο δημοφιλής στις βιομηχανίες και στα εργοτάξια. Ωστόσο, όσον αφορά τις εφαρμογές για την οδική ασφάλεια, υπάρχει απαίτηση για αρκετά μεγαλύτερες ταχύτητες και κατά συνέπεια, περισσότερη υπολογιστική ισχύς, προκειμένου να γίνεται αποτελεσματικά ο εντοπισμός σε πραγματικό χρόνο. Για την περίπτωση της διπλωματικής εργασίας, όπου η ανίχνευση γίνεται για οδηγούς και επιβάτες μηχανών και ποδηλάτων, το σύστημα χρειάζεται επιταχυντές υλικού για να μπορέσει να δουλέψει χωρίς προβλήματα. Συγκεκριμένα, η εργασία αυτή καινοτομεί:

- με την δημιουργία ενός συστήματος που αξιοποιεί τον χαμηλού κόστους επιταχυντή, EdgeTPU.
- με την παρουσίαση προτάσεων για την βελτιστοποίηση του συνόλου δεδομένων και της διαδικασίας της εκπαίδευσης μοντέλων χαμηλής υπολογιστικής ισχύος.
- την ενσωμάτωση του ανιχνευτή με ένα εξωτερικό σύστημα θετικής και αρνητικής οπτικοακουστικής ανατροφοδότησης για την προώθηση της χρήσης κράνους στους δρόμους της Ελλάδας.

Δομή

Η διπλωματική εργασία χωρίζεται σε 7 βασικά κεφάλαια:

- 1ο κεφάλαιο - Θεωρητικό Υπόβαθρο: Στο κεφάλαιο αυτό γίνεται συνοπτικά μία εισαγωγή στις βασικές έννοιες που χρειάζονται για την κατανόηση του αντικειμένου της εργασίας.
- 2ο κεφάλαιο - Βιβλιογραφική επισκόπηση: Στο κεφάλαιο αυτό θα γίνει μία επισκόπηση της τεχνολογικής αιχμής, παρουσιάζοντας σχετικά επιστημονικά άρθρα και έρευνες προκειμένου να γίνει η επιβεβαίωση της επιλεγμένης μεθοδολογίας για την διπλωματική εργασία.
- 3ο κεφάλαιο - Εργαλεία Μηχανικής Μάθησης: Στο κεφάλαιο αυτό καλύπτονται οι απαραίτητες γνώσεις σχετικά με το υλικό και το λογισμικό που χρησιμοποιήθηκε για την εκπαίδευση των μοντέλων μηχανικής μάθησης.
- 4ο κεφάλαιο - Δημιουργία και εκπαίδευση του TFlite μοντέλου: Σε αυτό το κεφάλαιο περιγράφεται αναλυτικά η διαδικασία της δημιουργίας των TFlite μοντέλων για object detection.
- 5ο κεφάλαιο - Μεταφορά και εφαρμογή του μοντέλου στο Google Coral Dev Board: Σε αυτό το κεφάλαιο περιέχεται η διαδικασία της μεταφοράς των εκπαιδευμένων μοντέλων στο Google Coral Dev Board, καθώς και της εκτέλεσής τους.
- 6ο κεφάλαιο - Δημιουργία οπτικού και ηχητικού συστήματος ανατροφοδότησης: Το κεφάλαιο αυτό έχει ως στόχο την λεπτομερής περιγραφή της δημιουργίας του εξωτερικού οπτικού και ηχητικού συστήματος ανατροφοδότησης σε μικροελεγκτή ATmega328P.
- 7ο κεφάλαιο - Δοκιμές και Αποτελέσματα: Σε αυτό το κεφάλαιο θα γίνει η ανάλυση της διαδικασίας των δοκιμών των μοντέλων. Έπειτα, θα ακολουθήσει η περιγραφή και η οπτικοποίηση των αποτελεσμάτων.
- 8ο κεφάλαιο - Ενεργειακή Αναφορά: Το τελευταίο από τα βασικά κεφάλαια. Εκεί θα γίνει μία αναφορά στην ενεργειακή κατανάλωση του συστήματος σε όλες τις πιθανές καταστάσεις λειτουργίας του και θα αναφερθούν και οι πιθανές επιλογές τροφοδοσίας.
- Τέλος, θα παρουσιαστούν τα τελικά συμπεράσματα για την διαδικασία της κατασκευής του συστήματος. Επιπλέον, θα γίνουν σχολιαστούν τα δύο μοντέλα, καθώς και μερικές ιδέες για την περαιτέρω βελτίωση της απόδοσής τους.

1 ΚΕΦΑΛΑΙΟ 1ο : Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό θα γίνει μία εισαγωγή στο απαραίτητο θεωρητικό υπόβαθρο, προκειμένου να γίνει κατανοητή η διαδικασία υλοποίησης της διπλωματικής εργασίας. Συγκεκριμένα, θα γίνει μία σταδιακή αύξηση της πολυπλοκότητας της θεωρίας, ξεκινώντας από τις βασικές έννοιες της μηχανικής μάθησης και καταλήγοντας σε πιο εξειδικευμένες γνώσεις για τον εντοπισμό αντικειμένων μέσω υπολογιστικής όρασης. Επιπλέον, θα δοθεί μεγαλύτερη έμφαση στις μεθόδους και τους ορισμούς που αφορούν το αντικείμενο της διπλωματικής εργασίας.

1.1 Μηχανική Μάθηση

Ως **Μάθηση** ορίζεται η διαδικασία βελτίωσης της επίδοσης ενός συστήματος σε μία συγκεκριμένη εργασία μετά από την παρατήρηση πολλών παραδειγμάτων. Για να μπορέσει να επιτευχθεί μάθηση, πρέπει να υπάρχουν τρία βασικά συστατικά:

1. ένα περιβάλλον το οποίο προσφέρει δεδομένα στο σύστημα
2. ένα κριτήριο αξιολόγησης της επίδοσης του συστήματος
3. μία συγκεκριμένη εργασία που καλείται να εκτελέσει το σύστημα

Η **Μηχανική Μάθηση** είναι ένας τομέας της Τεχνητής Νοημοσύνης που ασχολείται με την ανάπτυξη αλγορίθμων μάθησης, δηλαδή αλγορίθμων που βελτιώνουν την επίδοση ενός συστήματος σε προβλήματα όπως η αναγνώριση αντικειμένων, η πρόβλεψη της τιμής ή αξίας μετρήσιμων ποσοτήτων, η ομαδοποίηση ομοειδών αντικειμένων και η ανάπτυξη στρατηγικών (π.χ. σε παιχνίδια). Για να επιτευχθεί η βελτίωση, δίνεται στον αλγόριθμο ένα ικανό πλήθος παραδειγμάτων ώστε να μπορέσει να εκπαιδευθεί. Η βελτίωση συνήθως γίνεται σταδιακά επειδή ο αλγόριθμος τις περισσότερες περιπτώσεις είναι επαναληπτικός, δηλαδή εξετάζει τα παραδείγματα πολλές φορές [2].

Φυσικά, η διαδικασία αυτή δεν μπορεί να πραγματοποιηθεί μέσω απομνημόνευσης όλων των πιθανών εκδοχών ενός αντικειμένου. Στις περισσότερες περιπτώσεις αυτό δεν είναι εφικτό, καθώς μπορεί να υπάρχουν άπειρα διαφορετικά παραδείγματα. Για παράδειγμα, για την εφαρμογή της εργασίας αυτής, θα έπρεπε να βρεθούν εικόνες για όλα τα διαφορετικά κράνη που υπάρχουν, με όλα τα διαφορετικά επίπεδα φωτισμού, όλες τις πιθανές αποστάσεις και όλες τις πιθανές γωνίες. Στόχος λοιπόν της μάθησης είναι η δυνατότητα παραγωγής σωστών εκτιμήσεων σχετικά με δεδομένα τα οποία αντιμετωπίζονται για πρώτη φορά από το σύστημα [2].

Για να μπορέσει να ανταπεξέλθει στην παραπάνω απαίτηση, είναι απαραίτητη η παρουσίαση παρόμοιων αντικειμένων ή δεδομένων με τέτοιον τρόπο ώστε να αποκαλύπτεται η κρυμμένη σχέση μεταξύ των μεταβλητών [2].

Υπάρχουν τρεις βασικοί **τύποι μάθησης** [2]:

1. μάθηση με επίβλεψη
2. μάθηση χωρίς επίβλεψη
3. μάθηση με ενίσχυση

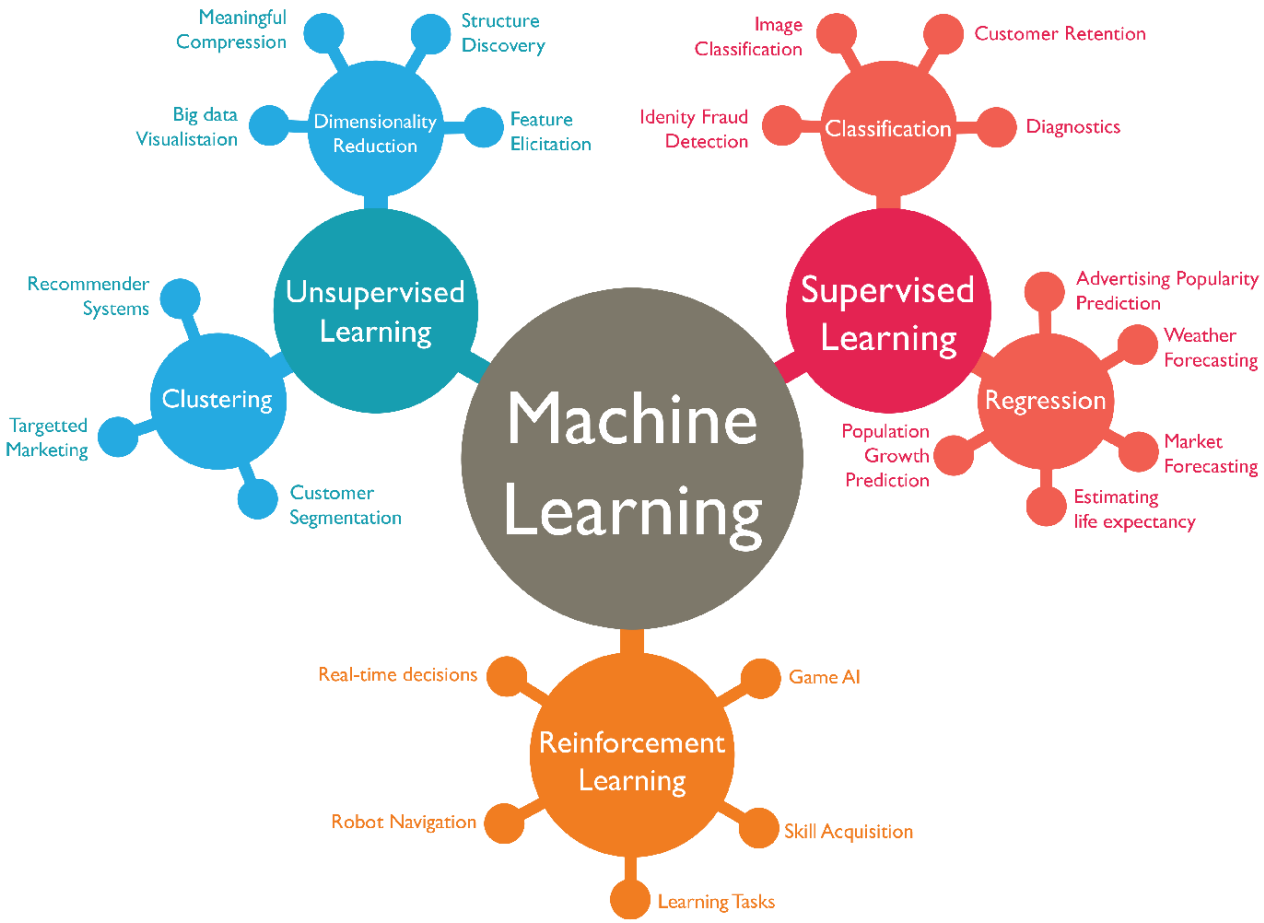
Στην **μάθηση με επίβλεψη** (supervised learning), το μοντέλο χρησιμοποιεί μια σειρά από πρότυπα εισόδου x1, x2, x3,..., μαζί με τους αντίστοιχους στόχους t1, t2, t3,... Τα πρότυπα είναι συνήθως διανύσματα σε κάποιον χώρο n διαστάσεων. Τα προβλήματα μάθησης με επίβλεψη είναι ίσως τα πιο συνηθισμένα και χωρίζονται σε δύο μεγάλες κατηγορίες, τα προβλήματα **ταξινόμησης** και τα προβλήματα **παλινδρόμησης** [2].

Τα προβλήματα **ταξινόμησης** (classification problems) έχουν σαν στόχους διακριτές τιμές, π.χ. 0, 1, 2,... Οι στόχοι αντιστοιχούν σε κλάσεις αντικειμένων [2]. Για παράδειγμα, στην εργασία αυτή θα υπάρχουν δύο κλάσεις για τις περιπτώσεις που το άτομο φοράει ή δεν φοράει κράνος. Αυτές ορίζονται ως κλάση 0 και κλάση 1 αντίστοιχα.

Τα προβλήματα **παλινδρόμησης** (regression problems) έχουν σαν στόχους συνεχείς τιμές ή απεριόριστο πλήθος διακριτών τιμών. Οι στόχοι αυτοί αντιστοιχούν σε τιμές ή αξίες κάποιων ποσοτήτων [2].

Στην **μάθηση χωρίς επίβλεψη** (unsupervised learning), το μοντέλο χρησιμοποιεί τα πρότυπα εισόδου, χωρίς όμως να διαθέτει πληροφορίες σχετικά με τους στόχους. Μία αρκετά συχνή εφαρμογή αυτού του τύπου μάθησης είναι τα προβλήματα **συσταδοποίησης** (clustering) **ή ομαδοποίησης** αντικειμένων [2].

Τέλος, στην **μάθηση με ενίσχυση** (reinforcement learning), το μοντέλο χρησιμοποιεί μια ακολουθία προτύπων εισόδου x1, x2, x3, ...και οι στόχοι είναι συνήθως τιμές ανταμοιβής ή τιμωρίας. Συχνά (όχι όμως πάντα), υπάρχει μία μόνο ανταμοιβή και αυτή γίνεται γνωστή στο τέλος της ακολουθίας. Αυτό γίνεται ιδίως όταν το σύστημα μαθαίνει να παίζει παιχνίδια [2].



Εικόνα 1.1: Γραφική αναπαράσταση της Μηχανικής Μάθησης και των υποκατηγοριών της [3]

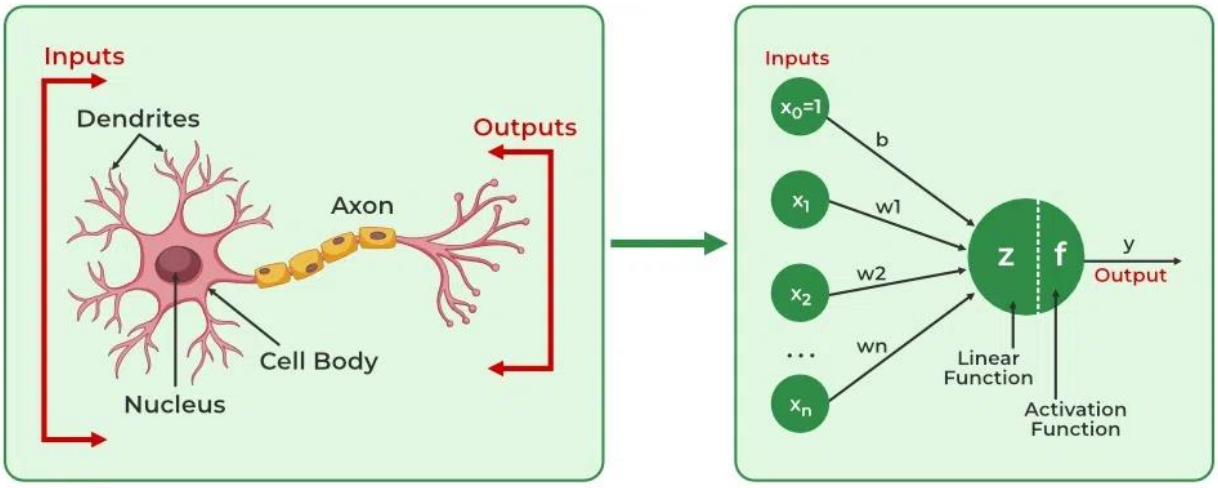
1.2 Νευρωνικά Δίκτυα

Τα τυπικά τεχνητά **νευρωνικά δίκτυα** χρησιμοποιούν πολύ απλοποιημένα μοντέλα νευρώνων. Ενώ η τεχνολογία αυτή είναι βασισμένη στα βιολογικά νευρωνικά δίκτυα, τα τεχνητά μοντέλα έχουν ελάχιστη σχέση με αυτά, καθώς οι τεχνητοί νευρώνες διατηρούν μόνο τα πολύ αδρά χαρακτηριστικά των λεπτομερών μοντέλων που χρησιμοποιούνται στην νευρολογία. Ωστόσο αυτό το γεγονός δεν έχει και πολύ μεγάλη σημασία, γιατί ακόμη και αυτά τα απλά μοντέλα νευρώνων μπορούν να δημιουργήσουν ιδιαιτέρως ενδιαφέροντα δίκτυα, αρκεί να πληρούν δύο βασικά χαρακτηριστικά [2]:

1. οι νευρώνες πρέπει να έχουν ρυθμιζόμενες παραμέτρους ώστε να διευκολύνεται η διαδικασία της μάθησης (**πλαστικότητα νευρώνων**).
2. το δίκτυο πρέπει να αποτελείται από μεγάλο πλήθος νευρώνων ώστε να επιτυγχάνεται **παραλληλισμός της επεξεργασίας** και **κατανομή των πληροφοριών**.

Η πρόκληση που αντιμετωπίζει η θεωρία των τεχνητών νευρωνικών δικτύων είναι η εύρεση κατάλληλων αλγορίθμων εκπαίδευσης των δικτύων και ανάκλησης των πληροφοριών που αυτά περιέχουν, ώστε να προσομοιάζονται ευφυείς διαδικασίες όπως αυτές που αναφέρθηκαν παραπάνω.

Η πιο απλή μορφή ενός τεχνητού νευρώνα παρουσιάστηκε από τους επιστήμονες McCulloch και Pitts το 1940, η κατάσταση του οποίου περιγράφεται από έναν δυαδικό αριθμό y, όπου όταν το y = 0, ο νευρώνας είναι αδρανής και όταν το y = 1, ο νευρώνας πυροδοτεί παλμούς στην μέγιστη δυνατή συχνότητα. Οι αντίστοιχες τεχνητές συνάψεις περιγράφονται από τα **συναπτικά βάρη** (synaptic weights) w, που είναι πραγματικοί αριθμοί. Τα συναπτικά βάρη λαμβάνουν θετικές τιμές για τις ενισχυτικές συνάψεις και αρνητικές τιμές για τις ανασταλτικές συνάψεις. Έστω λοιπόν οι είσοδοι x1, x2, x3,..., xn. Τότε ελέγχεται αν το άθροισμα $w1 \cdot x1 + w2 \cdot x2 + \ ... \ + wn \cdot xn$ του φορτίου που δέχεται ο νευρώνας είναι μεγαλύτερο από ένα κατώφλι θ. Αν είναι μεγαλύτερο, τότε ο νευρώνας πυροδοτεί παλμούς. Διαφορετικά, ο νευρώνας παραμένει αδρανής. Μαθηματικά, ορίζεται η ποσότητα $u = \sum_{i=1}^{n} wixi - \theta$ (1) και η αντίστοιχη έξοδος του νευρώνα προκύπτει ως $y = f(u) = 0 \, αν \, u \leq 0$ (2) και $y = f(u) = 1 \, αν \, u > 0$ (3). Η ποσότητα u καλείται **διέγερση** του νευρώνα και προκύπτει από το περιβάλλον, δηλαδή τις εισόδους xi. Η συνάρτηση f() είναι γνωστή ως βηματική συνάρτηση 0/1 (step function 0/1) [2].



Εικόνα 1.2: Από τον βιολογικό, στον τεχνητό νευρώνα [4]

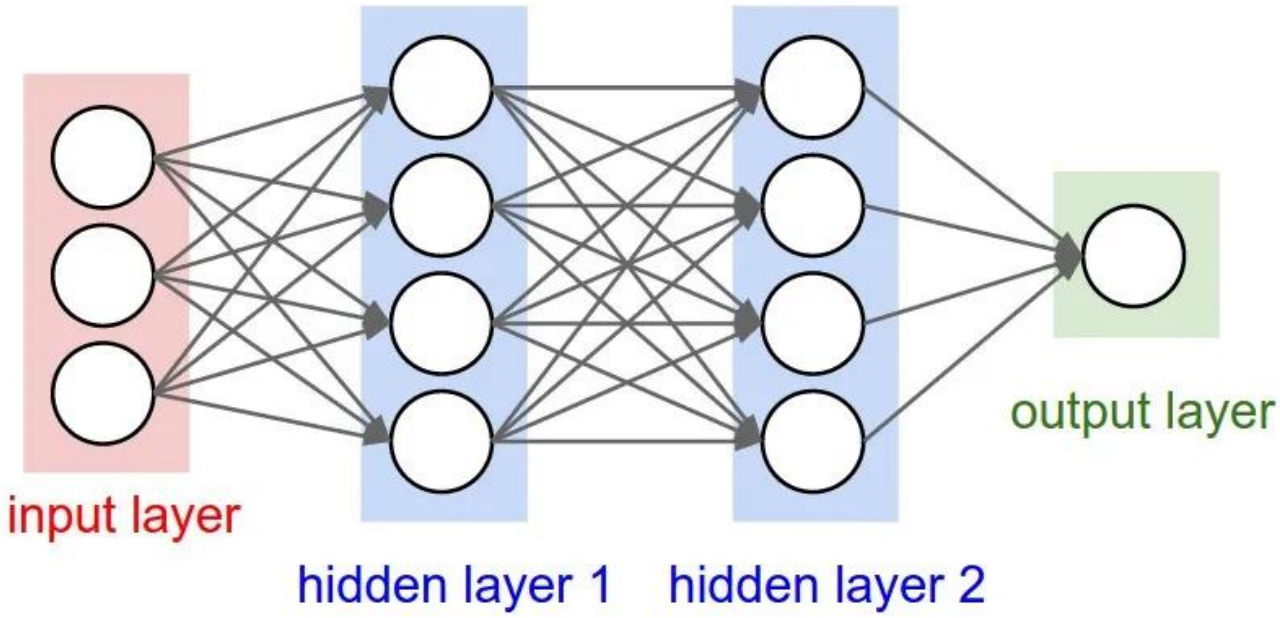
Προφανώς υπάρχουν πολλές διαφορετικές μοντελοποιήσεις του τεχνητού νευρώνα πλέον, που διαφέρουν από το μοντέλο McCulloch και Pitts. Η πιο σημαντική διαφορά εντοπίζεται στην μορφή της μη-γραμμικής συνάρτησης f() που είναι γνωστή και ως συνάρτηση ενεργοποίησης του νευρώνα (neuron activation function) και μπορεί να πάρει διάφορες μορφές.

Οι πιο βασικές είναι [2]:

- Η βηματική -1/1 (step function –1/1)
- Σιγμοειδής (sigmoid)
- Υπερβολική εφαπτομένη (hyperbolic tangent)
- Συνάρτηση κατωφλίου (threshold function)
- Συνάρτηση ράμπας (ramp function) ή ανορθωμένη γραμμική μονάδα (rectified linear unit – ReLU)
- Γραμμική (linear)

Η βασική δομή ενός νευρωνικού δικτύου είναι η εξής [2]:

1. ένα **στρώμα εξόδου** (output layer) που αποτελείται από έναν νευρώνα Y για κάθε έξοδο y.
2. το **κρυφό στρώμα** (hidden layer) που αποτελείται από νευρώνες Ai με εξόδους αi και ανάλογα με την εφαρμογή του δικτύου, μπορεί να περιέχει μία πληθώρα από διαφορετικά στρώματα νευρώνων με διαφορετικές ιδιότητες και συναρτήσεις ενεργοποίησης, η έξοδοι των οποίων λειτουργούν ως είσοδοι των νευρώνων του επόμενου στρώματος.
3. το **στρώμα εισόδων** (input layer) xi, το οποίο δεν λογίζεται ως στρώμα νευρώνων, διότι οι κόμβοι της εισόδου απλώς μεταδίδουν τις τιμές τους στο επόμενο στρώμα.



Εικόνα 1.3: Η βασική δομή ενός τεχνητού νευρωνικού δικτύου [5]

Η διαδικασία κατά την οποία καθορίζονται όλα τα συναπτικά βάρη σε ένα νευρωνικό δίκτυο πολλαπλών στρωμάτων, ώστε να ικανοποιείται κάποιο κριτήριο καταλληλότητας λέγεται **εκπαίδευση** (training). Ίσως ο πιο σημαντικός και ευρέως χρησιμοποιημένος αλγόριθμος είναι ο **Back-Propagation**. Φυσικά πρόκειται για έναν πολύ παλιό αλγόριθμο, που προτάθηκε αρχικά από τον Paul Werbos την δεκαετία του 1970, οπότε είναι και λογικό να έχει υποστεί αρκετές βελτιώσεις κατά την πάροδο των χρόνων. Η κεντρική ιδέα ωστόσο παραμένει η ίδια. Το βασικό χαρακτηριστικό της μεθόδου είναι η ύπαρξη στόχων, άρα κατατάσσεται στην κατηγορία της μάθησης με επίβλεψη [2].

Έστω λοιπόν:

- $x^{(p)} = \left[x_1^{(p)}, \ \dots, x_n^{(p)}\right]^T$ το p-οστό διάνυσμα εισόδου
- $y^{(p)} = \left[y_1^{(p)}, \ \dots, y_m^{(p)}\right]^T$ το p-οστό διάνυσμα εξόδου
- $t^{(p)} = \left[t_1^{(p)}, \ \dots, t_m^{(p)}\right]^T$ το p-οστό διάνυσμα στόχων

Οι είσοδοι είναι τα ζεύγη των διανυσμάτων εισόδων και στόχων $\left\{x^{(p)}, \ t^{(p)}\right\}_{p=1}^{N}$ και οι έξοδοι είναι τα εκπαιδευμένα βάρη $w_{ij}(l)$, τα οποία αρχικά αρχικοποιούνται με μερικές τυχαίες τιμές, κατά προτίμηση μικρές, για κάθε στρώμα l. Το βάρος αντιστοιχεί στο κατώφλι του νευρώνα i για το στρώμα l [2].

Το n τίθεται αρχικά ως 1.

Για το κάθε πρότυπο p = 1, ... , P

Στην **φάση της ανάκλησης** (forward phase), υπολογίζεται η έξοδος $\alpha_i(1)$ για το στρώμα 1, στην συνέχεια το $\alpha_i(2)$ για το στρώμα 2 και τελειώνει όταν βρεθεί και το $\alpha_i(L)$ για το στρώμα L.

Στην **φάση του υπολογισμού δ** (backward phase) υπολογίζονται τα σφάλματα $\delta_i(L)$ του στρώματος L, μετά τα $\delta_i(L-1)$ για το στρώμα L-1 και τελειώνει όταν βρεθούν και τα σφάλματα $\delta_i(1)$ του πρώτου στρώματος. Για το στρώμα L, το δ προκύπτει από την εξίσωση$\delta_i^p(L) = f'\left(u_i^p(L)\right) \cdot \left(t_i^{(p)} - y_i^{(p)}\right)$(4), ενώ για τα υπόλοιπα στρώματα χρησιμοποιείται η εξίσωση $\delta_i^p(l) = f'\left(u_i^p(l)\right) \cdot \sum_{j=1}^{N(l+1)} w_{ji}^p(l+1) \cdot \delta_j^p \cdot (l+1)$ (5).

Στην φάση της **ενημέρωσης βαρών** (update phase), ενημερώνονται τα βάρη, όλων των στρωμάτων μέσω της εξίσωσης $w_{ij}^{k+1}(l) = w_{ij}^k(l) + \beta \cdot \delta_i^p(l) \cdot \alpha_j^p(l-1)$ (6) όπου j = 0, 1, ... , N και l = 1, ... , L

Αφού ολοκληρωθούν αυτές οι φάσεις, το n τίθεται ως n+1 (αυξάνεται κατά 1) και επαναλαμβάνεται η παραπάνω διαδικασία μέχρι το συνολικό σφάλμα J να είναι μικρότερο από κάποιο κατώφλι ε ή να έχει φτάσει στον μέγιστο αριθμό επαναλήψεων [2].

Ο αλγόριθμος αυτός είναι μία προσέγγιση της βαθμωτής κατάβασης. Η παράμετρος **β**, λέγεται **βήμα εκπαίδευσης** και είναι περίπου ίσο με το ελάχιστο βήμα χρόνου dt. Άρα για πολύ μικρές τιμές β, ο αλγόριθμος θα συγκλίνει στο κοντινότερο τοπικό ελάχιστο, ωστόσο η σύγκλιση θα είναι πολύ αργή. Για μεγαλύτερες τιμές, η σύγκλιση θα είναι πιο γρήγορη αλλά η προσέγγιση δεν θα είναι και τόσο καλή. Μία καλή λύση αυτού του προβλήματος είναι να οριστεί διαφορετικό β για κάθε βάρος ξεχωριστά, καθώς τα βάρη έχουν διαφορές μεταξύ τους και ενώ ένα μπορεί να χρειάζεται ταχύτερους ρυθμούς εκπαίδευσης, ένα άλλο μπορεί να απαιτεί βραδύτερους [2].

1.3 Βαθιά Μάθηση

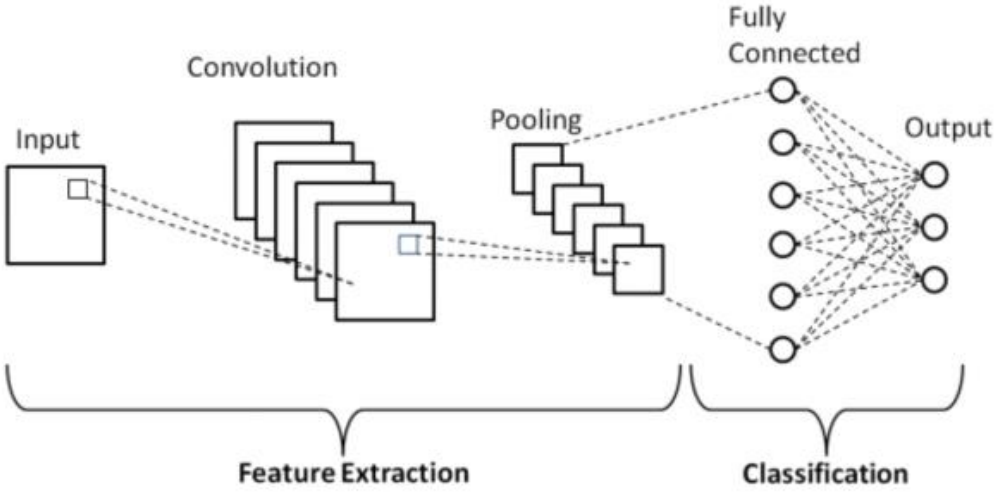
Η **βαθιά μάθηση** (deep learning) αφορά τα νευρωνικά δίκτυα πολλών στρωμάτων. Ένα ρηχό νευρωνικό δίκτυο δύο στρωμάτων μπορεί να αναπαραστήσει οποιαδήποτε συνεχής συνάρτηση. Τα μοντέλα με πολλά στρώματα έχουν ορισμένα πολύ σημαντικά πλεονεκτήματα. Ένα βασικό είναι ότι σε πολλές περιπτώσεις, ένα βαθύ δίκτυο μπορεί να αναπαραστήσει συναρτήσεις της ίδιας πολυπλοκότητας με ένα ρηχό, χρησιμοποιώντας παράλληλα λιγότερους νευρώνες. Ένα άλλο πλεονέκτημα προκύπτει από βιολογικές παρατηρήσεις. Συγκεκριμένα, υπάρχουν ενδείξεις ότι ο ανθρώπινος εγκέφαλος έχει και αυτός βαθιά αρχιτεκτονική, με βασικό παράδειγμα τον οπτικό φλοιό, που αποτελείται από πολλαπλά διαφορετικά στρώματα, τα οποία σταδιακά αναγνωρίζουν όλο και πιο σύνθετα σχήματα, οδηγώντας στην τελική εικόνα [2]. Με αυτή την λογική λειτουργούν και τα τεχνητά βαθιά δίκτυα. Παρακάτω θα γίνει μία συνοπτική παρουσίαση των νευρωνικών δικτύων που βασίζονται στην βαθιά μάθηση και αποτέλεσαν την βάση με την οποία έγινε εφικτή αυτή η εργασία.

1.4 Συνελικτικά Νευρωνικά Δίκτυα

Για την επίτευξη της ανίχνευσης και κατηγοριοποίησης αντικειμένων, γίνεται χρήση **συνελικτικών νευρωνικών δικτύων** (convolutional neural networks – CNN). Τα CNN είναι δίκτυα πολλών στρωμάτων, κατάλληλα για την αναγνώριση εικόνας. Η πρώτη πρόταση ενός τέτοιου δικτύου ήταν από τον LeCun το 1980 και η δομή του είναι συνήθως βαθιά. Αποτελούνται από εναλλασσόμενα στρώματα **συνέλιξης** (convolution) και **υποδειγματοληψίας** (sub-sampling/pooling). Μετά το τελευταίο στρώμα συνέλιξης ή υποδειγματοληψίας, ακολουθούν ένα ή περισσότερα **πλήρως συνδεδεμένα στρώματα** (fully-connected layers), τα οποία λειτουργούν ως ένα υποδίκτυο ταξινομητή [2].

Η συνεχής χρήση στρωμάτων συνέλιξης και υποδειγματοληψίας οδηγεί στην ουσία στον μετασχηματισμό της εικόνας εισόδου σε **χάρτη χαρακτηριστικών** (feature map). Ο χάρτης αυτός

στην συνέχεια λειτουργεί σαν είσοδος στο υποδίκτυο ταξινόμησης. Το τελευταίο στρώμα είναι συνήθως ένα στρώμα softmax [2].



Εικόνα 1.4: Η βασική δομή ενός συνελικτικού νευρωνικού δικτύου [6]

Η έξοδος του κάθε συνελικτικού στρώματος ή στρώματος υποδειγματοληψίας είναι μία σειρά από χάρτες χαρακτηριστικών. Ο κάθε χάρτης είναι ένα δισδιάστατο πλέγμα από νευρώνες, όπου ο καθένας διεγείρεται από μία μικρή περιοχή στο προηγούμενο στρώμα. Η περιοχή αυτή, καλείται τοπικό υποδεκτικό πεδίο (local receptive field) του συγκεκριμένου νευρώνα. Τα συναπτικά βάρη που συνδέουν έναν νευρώνα με τους νευρώνες του τοπικού πεδίου του είναι συλλογικά γνωστά ως μάσκα. Όλοι οι νευρώνες που βρίσκονται στον ίδιο χάρτη χαρακτηριστικών έχουν την ίδια μάσκα, οπότε υπάρχει επανάληψη των βαρών [2].

Έστω C ο αριθμός χαρτών χαρακτηριστικών και ένα δισδιάστατο πλέγμα νευρώνων με διαστάσεις NxN. Τότε ο νευρώνας που βρίσκεται στην θέση i,j στον k-οστό χάρτη χαρακτηριστικών, υλοποιεί την παρακάτω συνάρτηση:

$y_i^k = f\left(\sum_{i=1}^{C'} \sum_{\alpha=1}^{m} \sum_{\beta=1}^{m} w_{\alpha,\beta,l}^k \cdot x_{i-\alpha,j-\beta}^l + b^k\right)$ (7) [2].

Η πράξη μέσα στην παρένθεση της παραπάνω εξίσωσης (7) ονομάζεται **συνέλιξη**. Υπολογίζεται το άθροισμα των τιμών $x_{i-\alpha,j-\beta}^l$ των νευρώνων για όλους τους χάρτες χαρακτηριστικών $l = 1, \dots, C'$ του προηγούμενου στρώματος. Ο νευρώνας δέχεται είσοδο μόνο από μία μικρή περιοχή μεγέθους m x m με κέντρο το σημείο i,j από τον κάθε χάρτη του προηγούμενου στρώματος. Αυτή η περιοχή ονομάζεται **τοπικό υποδεκτικό πεδίο** (local receptive field). Ο τρισδιάστατος πίνακας $W^k = \left[w_{\alpha,\beta,l}^k\right]$ καλείται **συνελικτική μάσκα** ή απλώς **μάσκα** ή **φίλτρο** για τον k-οστό χάρτη

χαρακτηριστικών. Το αποτέλεσμα της συνέλιξης περνάει μέσα από μία μη γραμμική συνάρτηση ενεργοποίησης νευρώνα f() όπως η ReLU [2].

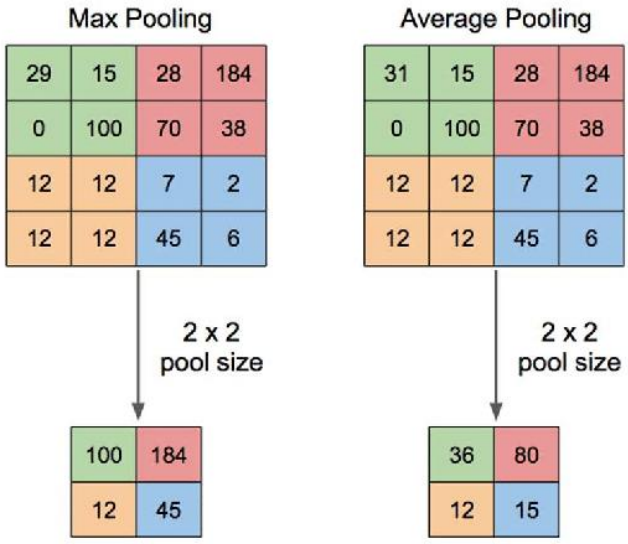
Στην ουσία η συνέλιξη λειτουργεί ως ένα φίλτρο, το οποίο σαρώνει την εικόνα εισόδου και δίνει την μέγιστη απόκριση εκεί όπου εμφανίζεται ένα τοπικό χαρακτηριστικό όμοιο με το σχήμα της μάσκας [2].

Όπως αναφέρθηκε προηγουμένως, μετά από κάθε στρώμα συνέλιξης, ακολουθεί συνήθως ένα **στρώμα υποδειγματοληψίας**. Σκοπός του στρώματος αυτού είναι να κάνει το σύστημα λιγότερο ευαίσθητο σε μικρές μετατοπίσεις των αντικειμένων της εικόνας, καθώς επίσης να αφαιρεί τις λεπτομέρειες, χωρίς ωστόσο να προκληθεί απώλεια της ικανότητας διαχωρισμού των αντικειμένων. Με αυτόν τον τρόπο, επιτυγχάνεται μία συμπίεση των δεδομένων και κατά συνέπεια μείωση των πράξεων [2].

Στο στρώμα αυτό η έξοδος yi,j του κάθε νευρώνα είναι ουσιαστικά η "σύνοψη" των εξόδων των νευρώνων από μία συγκεκριμένη γειτονιά m x m του προηγούμενου στρώματος, δηλαδή από το υποδεκτικό πεδίο του νευρώνα [2]. Δύο από τις πιο συνήθεις συναρτήσεις είναι:

- η **υποδειγματοληψία μέσης τιμής** (average pooling)
  $y_{ij} = \frac{1}{m^2}\sum_{a=0}^{m-1}\sum_{b=0}^{m-1} x_{im+a,\, jm+b}$ (8)
- η **υποδειγματοληψία μέγιστης τιμής** (max-pooling)
  $y_{ij} = \max x_{im+a,\, jm+b}\ \ με\ (0 \le a, b \le m-1)$ (9)

Η διαφορά μεταξύ των δύο αυτών μεθόδων είναι ότι η υποδειγματοληψία μέγιστης τιμής δημιουργεί χάρτες που αποτυπώνουν πιο καθαρά τα χαρακτηριστικά της αρχικής εικόνας [2].



Εικόνα 1.5: Παράδειγμα Max Pooling και Average Pooling [7]

Αξίζει να σημειωθεί, ότι σε αντίθεση με τα νευρωνικά δίκτυα τύπου MLP (multilayer perceptron), στα στρώματα συνέλιξης και υποδειγματοληψίας, οι νευρώνες δεν συνδέονται με όλους τους νευρώνες του προηγούμενου στρώματος. Αυτό έχει ως αποτέλεσμα την μείωση του πλήθους των παραμέτρων που απαιτούνται για την εκπαίδευση και την βελτίωση της επίδοσης του μοντέλου, διότι επιτρέπει την αναγνώριση χαρακτηριστικών οπουδήποτε και αν βρίσκονται στην εικόνα [2].

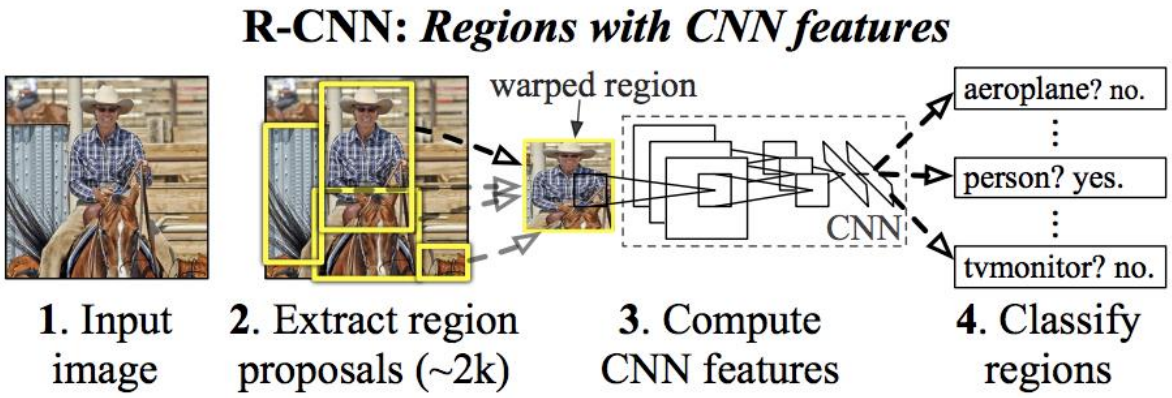
1.5 Object Detection

Ο **εντοπισμός αντικειμένων** (object detection) είναι μία εφαρμογή που αξιοποιεί την **υπολογιστική όραση** (computer vision - CV), ώστε να εντοπίζει την τοποθεσία και τον τύπο πολλαπλών αντικειμένων ταυτόχρονα. Για παράδειγμα, στην διπλωματική εργασία, δεν θα γίνεται απλά ο εντοπισμός της ύπαρξης ή μη, κράνους, αλλά και η εύρεση της τοποθεσίας που βρίσκονται τα αντικείμενα. Συνήθως γίνεται και η απαρίθμηση των εντοπισμένων αντικειμένων. Το object detection απαιτεί αρκετά πολύπλοκα συνελικτικά νευρωνικά δίκτυα και μεγάλη υπολογιστική ισχύς για να μπορεί να δουλεύει με αξιοπιστία και με μεγάλη ταχύτητα, ειδικά στις περιπτώσεις **εντοπισμού σε πραγματικό χρόνο** (real time detection). Η μεγάλη πρόκληση λοιπόν, είναι η δημιουργία δικτύων που μεγιστοποιούν την ακρίβεια (accuracy) του object detector, ενώ παράλληλα να ελαχιστοποιούν τις παραμέτρους που πρέπει να εκπαιδευτούν, όπως και το σύνολο των μαθηματικών πράξεων που πρέπει να γίνουν από τον υπολογιστή. Όπως θα αναλυθεί και παρακάτω, αυτό δεν είναι εύκολο, με τις περισσότερες αρχιτεκτονικές δικτύων να δημιουργούνται για συγκεκριμένου τύπου εφαρμογές. Συνεπώς, δεν υπάρχει κάποιο δίκτυο που να υπερισχύει σε όλες τις κατηγορίες ταυτόχρονα. Μερικά δίκτυα είναι μεγαλύτερης ακρίβειας, αλλά απαιτούν περισσότερη υπολογιστική ισχύς, ενώ άλλα είναι πιο κατάλληλα για εφαρμογές σε κινητά ή ενσωματωμένα συστήματα, υπολείποντας παράλληλα στην ακρίβεια. Υπάρχουν δύο μεγάλες κατηγορίες object detector: **α) two-stage/proposal** και **β) one-stage/proposal free**.

Για την περίπτωση των two-stage object detectors, η διαδικασία ανίχνευσης και εντοπισμού των αντικειμένων περνάει από δύο στάδια. Το πρώτο στάδιο ονομάζεται region proposal και ευθύνεται για την πρόταση περιοχών που υπάρχει πιθανότητα να βρίσκονται αντικείμενα ενδιαφέροντος. Το δεύτερο στάδιο αποτελείται από ένα ακόμα συνελικτικό νευρωνικό δίκτυο, που λαμβάνοντας τις προτάσεις από το προηγούμενο, τις επεξεργάζεται και τις αποδίδει στον ταξινομητή (classifier), όπου ολοκληρώνεται η διαδικασία. Το μοντέλο που λειτούργησε σαν την βάση για την εξέλιξη των two-stage object detectors είναι το R-CNN [8].

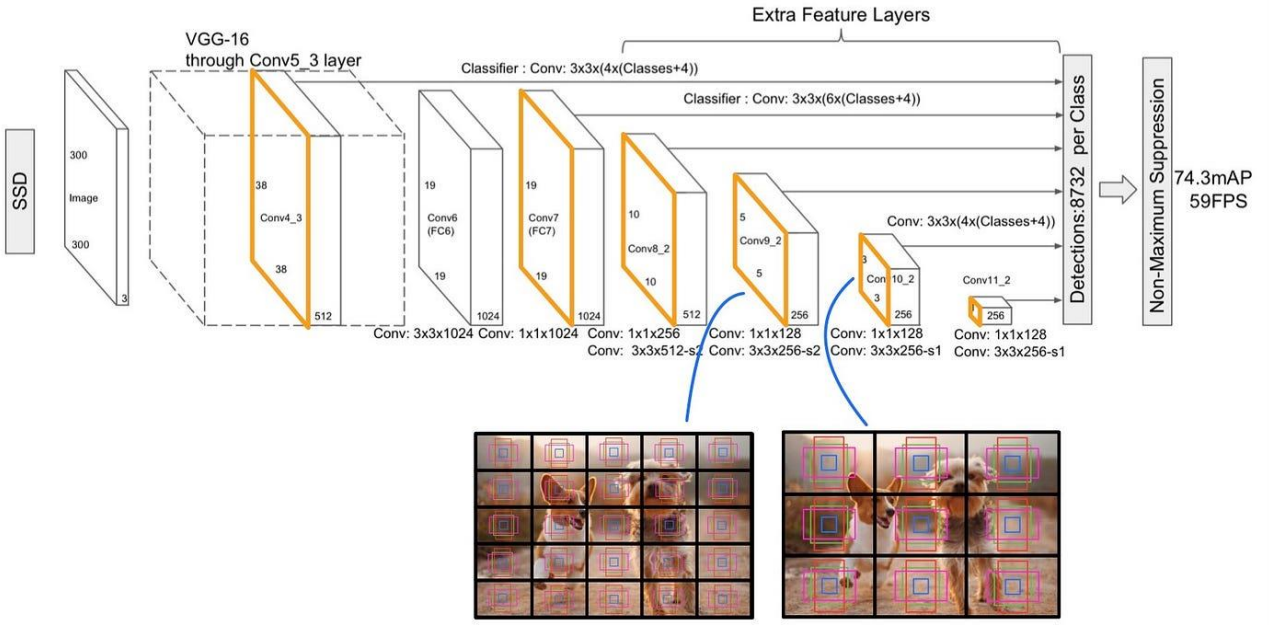
Οι προτάσεις περιοχών (region proposals) μπορούν να πραγματοποιηθούν με διάφορους αλγορίθμους, ωστόσο η πιο γνωστή μέθοδος είναι το **selective search**. Μέσω αυτής της μεθόδου, ένας **αλγόριθμος τμηματοποίησης** (segmentation algorithm) εφαρμόζεται στην εικόνα και δημιουργεί έναν χάρτη τμηματοποίησης, στον οποίο σχεδιάζονται και τα πιθανά **bounding boxes** (παραλληλόγραμμα που περικλείουν τα εντοπισμένα αντικείμενα). Ο χάρτης αυτός συγχωνεύεται

μετά από κάθε επανάληψη και οι προτάσεις όλο και μεγαλύτερων περιοχών αντλούνται από τον βελτιωμένο χάρτη [8]. Το υπόλοιπο δίκτυο, επεξεργάζεται τις περιοχές που έχουν δημιουργηθεί και εμφανίζει τα αποτελέσματα τις κατηγοριοποίησης για την καθεμία, σχεδιάζοντας παράλληλα και τα αντίστοιχα bounding boxes.



Εικόνα 1.6: Απεικόνιση της λειτουργίας ενός R-CNN [9]

Στην περίπτωση των one-stage object detectors, η διαδικασία ολοκληρώνεται σε ένα στάδιο, χωρίς να υπάρχει δηλαδή κάποια πρόταση. Τα μοντέλα αυτά είναι συνήθως πιο γρήγορα και προτιμώνται σε εφαρμογές πραγματικού χρόνου, ωστόσο δεν έχουν την ίδια απόδοση στο κομμάτι της ακρίβειας με τα two-stage. Οι δύο πιο δημοφιλείς μέθοδοι είναι το **SSD** (single shot detection) και το **YOLO** (You Only Look Once). Και οι δύο βασίζονται στην ίδια ιδέα. Στους χάρτες χαρακτηριστικών που δημιουργούνται από το δίκτυο, δημιουργείται ένα πλέγμα από ίσα σε διαστάσεις κελιά (SxS grid). Ένα δευτερεύον συνελικτικό νευρωνικό δίκτυο πραγματοποιεί μία πληθώρα από προβλέψεις για το κάθε κελί, οι οποίες διαφέρουν ποσοτικά από μοντέλο σε μοντέλο, όπου δίνονται τιμές σχετικά με το ποια κλάση έχει την μεγαλύτερη πιθανότητα να βρίσκεται σε αυτό (class scores). Παράλληλα γίνονται και οι ανάλογες προβλέψεις για τα πιθανά bounding boxes των αντικειμένων. Πλέον, για να γίνει πιο αποδοτική η πρόβλεψη των bounding boxes, ορίζονται προκαθορισμένα κουτιά που ονομάζονται **anchor boxes** και λειτουργούν σαν σημεία αναφοράς ως προς το σχήμα, το μέγεθος και την τοποθεσία των αντικειμένων. Ένα ακόμα πλεονέκτημά τους είναι ότι επιτρέπουν τον εντοπισμό πολλαπλών αντικειμένων μέσα σε ένα κελί. Επειδή ο αριθμός των bounding boxes που προκύπτουν για κάθε αντικείμενο είναι τεράστιος, ενώ χρειάζεται μόλις ένα, δημιουργείται μια ανάγκη για την αφαίρεση των περιττών κουτιών [10].



Εικόνα 1.7: Βασική δομή ενός Single-Shot Detector [11]

Μπορεί λοιπόν να εφαρμοστεί ένας αλγόριθμος που ονομάζεται **non-maximum suppression.** Κάθε κουτί έχει τα εξής δεδομένα: α) το **confidence score** (τιμή από το 0 έως το 1 που δείχνει την σιγουριά της πρόβλεψης) για την κλάση και β) το όνομα της κλάσης που αντιπροσωπεύει. Για την κάθε κλάση:

Είσοδοι: Μία λίστα (B) με τα bounding boxes που έχουν προταθεί, τα confidence scores τους (S) και η τιμή κατωφλίου επικάλυψης (overlap threshold) (N).

Έξοδος: Μία λίστα με τις φιλτραρισμένες προτάσεις (D), η οποία είναι αρχικά άδεια.

Αλγόριθμος:

1. Επιλέγεται η πρόταση με το μεγαλύτερο S και μεταφέρεται από την λίστα B στην λίστα D.
2. Η πρόταση αυτή συγκρίνεται με τις υπόλοιπες. Αν η τιμή IoU (Intersection over Union) είναι μεγαλύτερη από την τιμή N, τότε αφαιρείται το αντίστοιχο κουτί από την λίστα B.
3. Από τις προτάσεις που έχουν μείνει στην λίστα B, μεταφέρεται η πρόταση με το μεγαλύτερο S και μεταφέρεται στην λίστα D.
4. Ξαναγίνεται σύγκριση της πρότασης που μόλις μεταφέρθηκε με τις υπόλοιπες. Για κάθε πρόταση με IoU μεγαλύτερο από το N, γίνεται η αφαίρεσή της από την λίστα B.
5. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να μην υπάρχουν άλλες προτάσεις στην λίστα B.

Το IoU μεταξύ δύο bounding boxes υπολογίζεται ως εξής:

$$IoU = \frac{\text{ΚΟΙΝΗ ΕΠΙΦΑΝΕΙΑ ΤΩΝ ΔΥΟ ΚΟΥΤΙΩΝ}}{\text{ΣΥΝΟΛΙΚΗ ΕΠΙΦΑΝΕΙΑ ΤΩΝ ΔΥΟ ΚΟΥΤΙΩΝ}} (10)$$ [12].

Εικόνα 1.8: Πριν και μετά το non-maximum suppression [13]

Προφανώς υπάρχουν πάρα πολλές διαφορετικές αρχιτεκτονικές για object detection, οι οποίες εξελίσσονται συνεχώς με νέες επαναστατικές ιδέες για την βελτίωση τόσο της ταχύτητας του εντοπισμού, όσο και της ακρίβειάς του. Όσον αφορά τα two-stage detectors, δημιουργήθηκαν πολλά βελτιωμένα μοντέλα πάνω στο R-CNN, όπως το Fast R-CNN και το Faster R-CNN, ενώ για τα one-stage detectors, πολλαπλές εκδοχές του YOLO και μία τεράστια ποικιλία από SSD μοντέλα είναι πλέον διαθέσιμα [10].

Εφόσον στην εργασία αυτή είναι άκρως σημαντική η ταχύτητα του μοντέλου και η χαμηλή απαίτηση υπολογιστικής ισχύος, θα πρέπει να χρησιμοποιηθεί η μέθοδος εντοπισμού με ένα στάδιο (one-stage) και μία αρχιτεκτονική δικτύου που πληροί τις παραπάνω προϋποθέσεις.

2 ΚΕΦΑΛΑΙΟ 2ο : Βιβλιογραφική επισκόπηση

Στο κεφάλαιο αυτό θα γίνει μία επισκόπηση της τεχνολογικής αιχμής μέσω αναζήτησης δημοσιευμένων επιστημονικών άρθρων και άλλων αναφορών από διεθνείς βάσεις δεδομένων, προκειμένου να υπάρξει επιβεβαίωση της εγκυρότητας της επιλεγμένης μεθοδολογίας, βάσει των τάσεων στον κλάδο της μηχανικής μάθησης. Τα τελευταία χρόνια, οι εφαρμογές αναγνώρισης κράνους μέσω της υπολογιστικής όρασης έχουν γίνει αρκετά δημοφιλείς, ιδιαίτερα στον τομέα της εργατικής ασφάλειας. Αρκετές έρευνες έχουν πραγματοποιηθεί, στις οποίες δοκιμάζονται διάφορες μεθοδολογίες για τον αποτελεσματικό εντοπισμό κρανών σε εργάτες, συνεισφέροντας έτσι στην δημιουργία πιο ασφαλών εργατικών συνθηκών. Παρακάτω θα παρουσιαστούν συνοπτικά μερικές από αυτές, που παρουσιάζονται σε σχετικά επιστημονικά άρθρα.

Στην [14], παρουσιάζεται μία έρευνα πολλαπλών προσπαθειών για την αυτόματη ανίχνευση κρανών με την χρήση υπολογιστικής όρασης. Μία αξιοσημείωτη προσέγγιση είναι αυτή των Wu και Zhao [15], οι οποίοι δημιούργησαν ένα ευφυές σύστημα που ενσωματώνει τον εντοπισμό των πεζών με την αναγνώριση κράνους. Η μέθοδός τους χρησιμοποιεί τεχνολογίες αναγνώρισης εικόνας για να διασφαλίσει την τήρηση των μέτρων ασφαλείας στα εργοτάξια, με σκοπό την μείωση των εργασιακών ατυχημάτων και κατά συνέπεια την ελάττωση των θανάτων.

Άλλη μία μεγάλη συνεισφορά προέρχεται από μία μελέτη, όπου προτείνεται ένα σύστημα διπλής λειτουργίας για την αναγνώριση κράνους και ταυτοποίησης [16]. Η έρευνα αυτή αναδεικνύει την ενσωμάτωση του εντοπισμού κράνους, με την ταυτοποίηση των εργαζομένων, που είναι άκρως σημαντική για την παρακολούθηση της συμμόρφωσής τους σε πραγματικό χρόνο. Οι συγγραφείς πραγματοποίησαν πολλαπλές δοκιμές σε διάφορα επίπεδα ορατότητας και κατέληξαν ότι τέτοιου είδους συστήματα μπορούν να ενισχύσουν την εποπτεία των μέτρων ασφαλείας στα εργοτάξια. Με την ανάπτυξη της βαθιάς μάθησης, μπόρεσαν να δημιουργηθούν μοντέλα για την ανίχνευση κρανών σε πραγματικό χρόνο. Για παράδειγμα, σε μία έρευνα αξιοποιήθηκαν συνελικτικά νευρωνικά δίκτυα τα οποία μπόρεσαν να πετύχουν υψηλή ακρίβεια κατά τον εντοπισμό των κρανών ασφαλείας σε διαφορετικές συνθήκες, αναδεικνύοντας την αποτελεσματικότητα αυτών των τεχνολογιών σε πρακτικές εφαρμογές πραγματικού χρόνου [14].

Η πρόοδος αυτών των τεχνολογιών δείχνει τον σημαντικό ρόλο που μπορεί να έχει η υπολογιστική όραση στην προώθηση της ασφάλειας, μέσω του αποτελεσματικού εντοπισμού των κρανών, ανοίγοντας τον δρόμο για περισσότερη έρευνα σε αυτόν τον τομέα. Ωστόσο, στις παραπάνω περιπτώσεις, ο εντοπισμός έγινε σε βιομηχανικές περιοχές, όπου δεν υπάρχει ανάγκη για μεγάλη υπολογιστική ισχύς.

Μία πιθανή λύση στο πρόβλημα αυτό προτάθηκε στο [17], όπου οι συγγραφείς κατάφεραν να δημιουργήσουν ένα μοντέλο που απαιτεί χαμηλότερη υπολογιστική ισχύς, αντικαθιστώντας το

μοντέλο **YOLOv4**, με ένα που αξιοποιεί το συνελικτικό νευρωνικό δίκτυο **CSPDarknet53-Tiny** σε συνδυασμό με ένα **FPN** (feature pyramid network) αντί για το **PAN** (Path Aggregation Network) που υπήρχε εξαρχής. Επίσης αφαιρούν τελείως την δομή **SPP** (Spatial Pyramid Pooling) για να αυξήσουν περαιτέρω την ταχύτητα του μοντέλου. Για να αντιμετωπίσουν την μείωση της ακρίβειας, χρησιμοποιούν τον μηχανισμό **CBAM** (Convolutional Block Attention Module).

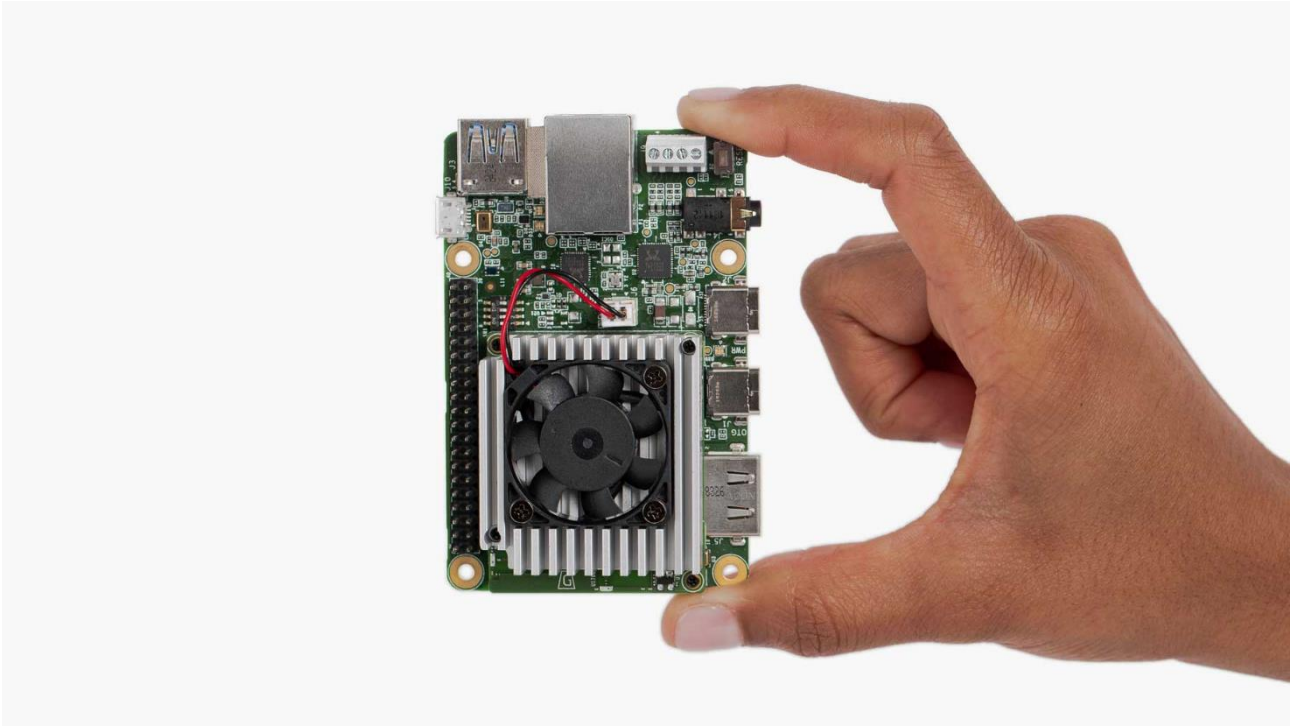
Τέλος, στο [18], χρησιμοποιείται ένα σύνολο δεδομένων με 5626 video (24465 bounding boxes) από δρόμους στο Dhaka και το Bangladesh, με ανάλυση 720p και ρυθμό ανανέωσης καρέ 30 fps, για να εκπαιδευτούν μοντέλα **SSD Mobilenet V2** και **Faster R-CNN inception V2**, προκειμένου να γίνεται εντοπισμός των μηχανών, των κρανών και των αντίστοιχων πινακίδων κυκλοφορίας σε πραγματικό χρόνο. Τα μοντέλα είχαν αρκετά καλές επιδόσεις, τόσο στην ακρίβεια, όσο και στην ταχύτητά τους, πετυχαίνοντας 45 fps στην **NVIDIA RTX2080 GPU**. Το πρόβλημα με αυτή την προσέγγιση είναι το υψηλό κόστος των υλικών και της ενέργειας για την εκπαίδευση των μοντέλων.

3 ΚΕΦΑΛΑΙΟ 3ο : Εργαλεία Μηχανικής Μάθησης

Στο κεφάλαιο αυτό θα γίνει μία εισαγωγή σε όλα τα εργαλεία που χρησιμοποιήθηκαν για την εκπαίδευση και την εφαρμογή των μοντέλων. Στην ουσία θα αναλυθεί το **υλισμικό** (hardware) και το **λογισμικό** (software) που χρειάστηκε καθώς και οι λόγοι για τους οποίους επιλέχθηκαν.

3.1 Το Google Coral Dev Board

Το **Google Coral Dev Board** είναι ένας μικρός υπολογιστής τοποθετημένος σε μία πλακέτα και είναι κατάλληλος για εφαρμογές μηχανικής μάθησης. Σχεδιασμένο από την Google, μπορεί πολύ εύκολα να συνδυαστεί με επιπλέον υλισμικό για την κατασκευή ενσωματωμένων συστημάτων με την δυνατότητα χρήσης μοντέλων μηχανικής μάθησης, όπως ένα object detector [19].



Εικόνα 3.1: Το Google Coral Dev Board [20]

Το Coral System-on-Module είναι ένα πλήρως ολοκληρωμένο σύστημα που περιέχει το iMX 8M system-on-chip της NXP, μνήμη eMMC, LPDDR4 RAM, Wi-Fi και Bluetooth. Ωστόσο το μεγάλο του πλεονέκτημα είναι το **EdgeTPU** (Tensor Processing Unit – TPU), το οποίο είναι ένα Application-specific integrated circuit (ASIC) σχεδιασμένο από την Google και παρέχει την δυνατότητα γρήγορων υπολογισμών με τανυστές (tensors), δηλαδή γεωμετρικά αντικείμενα που περιγράφουν γραμμικές σχέσεις ανάμεσα σε διανύσματα, βαθμωτά μεγέθη και άλλους τανυστές [21] και αποτελούν βασικό κομμάτι για την υλοποίηση μοντέλων μηχανικής μάθησης με χαμηλότερη κατανάλωση ενέργειας. Επιπλέον, παρέχει υποστήριξη αρκετών περιφερειακών συνδέσεων όπως

θύρες USB 2.0/3.0, θύρα για Ethernet, διεπαφή οθόνης DSI, διεπαφή κάμερας CSI-2, ακροδέκτες ηχείων και 40 pins για την περαιτέρω σύνδεση με άλλες συσκευές [19].

Αναλυτικά, τα χαρακτηριστικά του Google Coral Dev Board, όπως παρουσιάζονται στο documentation [19]:

- Edge TPU System-on-Module (SoM)
- NXP i.MX 8M SoC (Quad-core Arm Cortex-A53, plus Cortex-M4F)
- Google Edge TPU ML accelerator coprocessor
- Cryptographic coprocessor
- Wi-Fi 2x2 MIMO (802.11b/g/n/ac 2.4/5 GHz)
- Bluetooth 4.2
- 8 or 16 GB eMMC
- 1 or 4 GB LPDDR4
- USB connections
- USB Type-C power port (5 V DC)
- USB 3.0 Type-C OTG port
- USB 3.0 Type-A host port
- USB 2.0 Micro-B serial console port
- Audio connections
- 3.5 mm audio jack (CTIA compliant)
- Digital PDM microphone (x2)
- 2.54 mm 4-pin terminal for stereo speakers
- Video connections
- HDMI 2.0a (full size)
- 39-pin FFC connector for MIPI DSI display (4-lane)
- 24-pin FFC connector for MIPI CSI-2 camera (4-lane)
- MicroSD card slot
- Gigabit Ethernet port
- 40-pin GPIO expansion header
- Supports Mendel Linux (derivative of Debian)

Όλα τα I/O pin λειτουργούν με λογική 3,3 V. Έχουν προγραμματιζόμενη σύνθετη αντίσταση 40-255 Ω και μπορούν να παρέχουν ρεύμα μέχρι 82 mA όταν βρίσκονται σε λειτουργία εξόδου. Ο χρήστης έχει την δυνατότητα να επιλέξει και να προγραμματίσει κάποιο pin μέσω **Python APIs** (Application Programming Interface – API), όπως το python-periphery, το Adafruit Blinka και το libgpiod [19].

Συγκεκριμένα υπάρχουν:

- 2x 5V pins (+5V)
- 2x 3,3V pins (+3,3V)
- 4x I2C pins (Inter-Integrated Circuit)
    - 2x I2C SDA pins
    - 2x I2C SCL pins
- 4x UART pins (Universal Asynchronous Receiver / Transmitter)
    - 2x UART TX pins
    - 2x UART RX pins
- 5x SPI pins (Serial Peripheral Interface)
    - 1x SPI MOSI pin
    - 1x SPI MISO pin
    - 1x SPI SCLK pin
    - 2x SPI SS pins
- 3x PWM pins (Pulse Width Modulation)
- 4x SAI pins (Serial Audio Interface)
    - 1x SAI TXC pin
    - 1x SAI RXD0 pin
    - 1x SAI TXD0 pin
    - 1x SAI TXFS pin
- 8x GPIO pins (General Purpose Input/Output)
- 8x GND pins (Ground)

Το Google Coral χρησιμοποιεί ένα λειτουργικό σύστημα που ονομάζεται **Mendel Linux** και περιέχει όλες τις απαραίτητες βιβλιοθήκες και τα Python APIs για την ορθή λειτουργία του EdgeTPU και των TFlite μοντέλων (θα μελετηθούν στην επόμενη ενότητα) που εκτελούνται σε αυτό. Ο χρήστης μπορεί να επικοινωνήσει με το board μέσω κονσόλας από έναν host υπολογιστή με δύο βασικούς τρόπους. Ο πρώτος που είναι μέσω ενός καλωδίου USB-C που συνδέει το Coral με τον host υπολογιστή. Ο host πρέπει να έχει εγκατεστημένη την γλώσσα προγραμματισμού Python, το **Git Bash** και το **MDT** (Mendel Development Tool). Ο δεύτερος τρόπος είναι μέσω σειριακής κονσόλας αφού γίνει σύνδεση με καλώδιο Micro-USB. Ο host υπολογιστής, μέσω μίας εφαρμογής όπως το **PuTTY**, μπορεί να ενεργοποιήσει μία σειριακή σύνδεση με baud rate = 115200 [19].

3.2 Tensorflow - TFlite

Το **Tensorflow** είναι μία δωρεάν βιβλιοθήκη λογισμικού, η οποία αυτή την στιγμή είναι μία από τις πιο δημοφιλείς επιλογές για την δημιουργία μοντέλων μηχανικής μάθησης και τεχνητής

νοημοσύνης. Είναι γραμμένο σε **Python και C++** και δημιουργήθηκε από την ομάδα Google Brain της Google το 2015, με την έκδοση 1.0 να γίνεται διαθέσιμη για το κοινό το 2017 και την έκδοση 2.0 το 2019. Το Tensorflow μπορεί να χρησιμοποιηθεί με διάφορες γλώσσες προγραμματισμού, όπως την Java, την Javascript, την C++ και την Python, ωστόσο προτείνεται η χρήση της τελευταίας καθώς είναι πιο ολοκληρωμένες οι αντίστοιχες βιβλιοθήκες της [22], [23]. Μέσω του ενσωματωμένου **Keras API**, δίνεται η δυνατότητα ανάπτυξης άπειρων διαφορετικών νευρωνικών δικτύων με μεγάλη ευκολία, λόγω του υψηλού προγραμματιστικού επιπέδου του (high-level programming). Επίσης διαθέτει τεράστια ποικιλία σε στρώματα (π.χ. συνελικτικά/max-pooling, Dense κλπ), συναρτήσεις ενεργοποίησης, συναρτήσεις απωλειών (loss functions), μεθόδους βελτιστοποίησης και μεθόδους κανονικοποίησης των δεδομένων, η ρύθμιση των οποίων μπορεί να γίνει με λεπτομέρεια μέσω των αντίστοιχων συναρτήσεων που παρέχονται από το API [23].

Σε περίπτωση πολύ μεγάλων νευρωνικών δικτύων, υπάρχει η δυνατότητα κατανομής της διαδικασίας της εκπαίδευσής του σε πολλαπλές συσκευές, χωρίς την πραγματοποίηση αλλαγής στο μοντέλο, μέσω του Distribution Strategy API [23].

Το μεγάλο πλεονέκτημα του Tensorflow που το καθιστά ως την νούμερο ένα επιλογή για την δημιουργία της εργασίας αυτής είναι το **TensorflowLite** ή **TFlite**. Πρόκειται για ένα σύνολο εργαλείων που δίνουν την δυνατότητα δημιουργίας μοντέλων που θα λειτουργούν σε συσκευές χαμηλής υπολογιστικής ισχύος, όπως τα κινητά ή τα ενσωματωμένα συστήματα, με μικρότερο όγκο, υψηλότερη απόδοση και χαμηλότερη κατανάλωση ενέργειας. Μπορεί να εφαρμοστεί σε πολλαπλές πλατφόρμες όπως, το Android, το IOS, το embedded Linux και τους μικροελεγκτές. Επιπλέον, υποστηρίζει και αρκετές γλώσσες προγραμματισμού όπως, την Java, την Objective C, την C++ και την Python. Προσφέρει επιλογές αύξησης της απόδοσης του μοντέλου, μέσω **επιτάγχυνσης υλικού** (hardware acceleration) και **βελτιστοποίησης μοντέλου** (model optimization) [24].

Για την δημιουργία ενός TFlite μοντέλου υπάρχουν διάφοροι τρόποι. Ο πρώτος είναι η χρήση ενός ήδη υπάρχοντος μοντέλου. Ο δεύτερος είναι μέσω του **Tensorflow Lite Model Maker**, για την δημιουργία μοντέλων με δεδομένα του χρήστη. Ο τρίτος είναι μέσω της μετατροπής ενός κανονικού Tensorflow μοντέλου. Για να γίνει αυτό, αξιοποιείται ένα εργαλείο που ονομάζεται **Tensorflow Lite Converter**. Επιπρόσθετα, το εργαλείο αυτό διαθέτει και την επιλογή εφαρμογής μεθόδων βελτιστοποίησης που μειώνουν τον συνολικό όγκο του μοντέλου χωρίς να επιβαρύνεται ιδιαίτερα η απόδοσή του.

Στην περίπτωση που είναι διαθέσιμο ένα EdgeTPU, μπορεί να χρησιμοποιηθεί το edgetpu compiler, το οποίο θα μεταφέρει τις περισσότερες δυνατές διαδικασίες του μοντέλου στην TPU. Ωστόσο πριν μπορέσει να γίνει αυτό, είναι απαραίτητο να πραγματοποιηθεί το **quantization** του μοντέλου. Κατά την διάρκεια του quantization, τα συναπτικά βάρη μετατρέπονται από τύπου float32 (float 32 bit) σε int8 (integer 8 bit), στρογγυλοποιώντας δηλαδή τις τιμές τους. Τις περισσότερες

φορές υπάρχει μία μικρή μείωση στην ακρίβεια, ωστόσο η αύξηση της ταχύτητας του μοντέλου είναι πολύ μεγάλη.

3.3 Tensorflow Object Detection

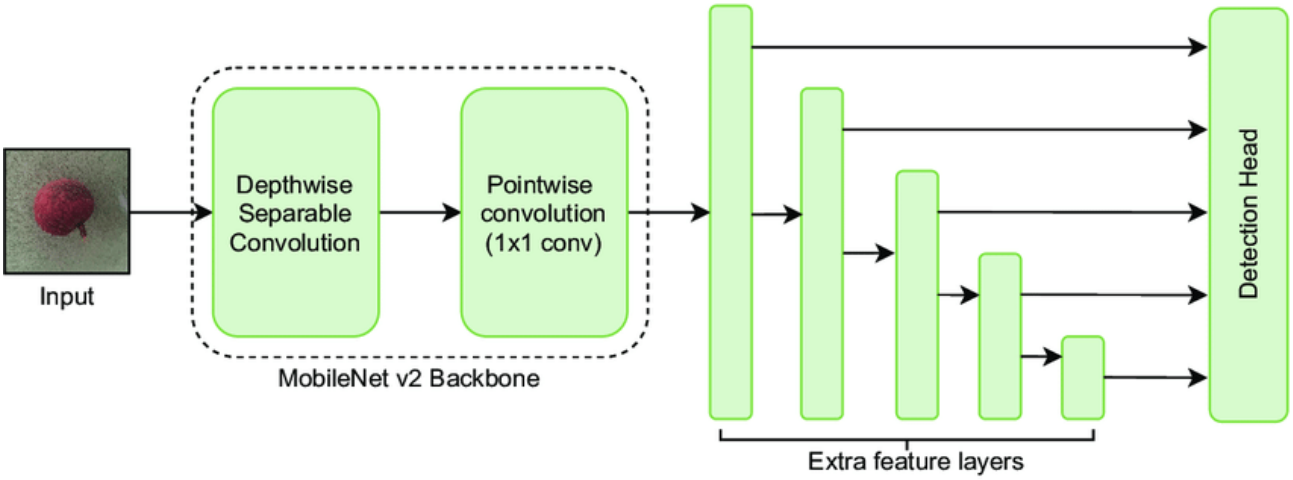
Το Tensorflow Object Detection API ή απλώς **Tensorflow Object Detection**, είναι μία δωρεάν βιβλιοθήκη που περιέχει όλα τα απαραίτητα εργαλεία για την επίτευξη της δημιουργίας ενός μοντέλου εντοπισμού αντικειμένων, μέσω μίας μεθόδου που λέγεται **Transfer Learning**. Η μέθοδος αυτή βασίζεται στην επανεκπαίδευση των τελευταίων (συνήθως) στρωμάτων ενός ήδη εκπαιδευμένου νευρωνικού δικτύου με κάποιο μεγάλο σύνολο δεδομένων. Ο λόγος που είναι τόσο χρήσιμη αυτή η διαδικασία είναι το γεγονός ότι απαιτούνται πολύ λιγότερα δεδομένα και κατά συνέπεια λιγότερος χρόνος εκπαίδευσης, για την δημιουργία μοντέλων με πολύ ικανοποιητική απόδοση. Έτσι, το Transfer Learning συνεχίζει να είναι η καλύτερη μέθοδος δημιουργίας μοντέλων υψηλών επιδόσεων για χρήστες με περιορισμένους πόρους. Το ερώτημα είναι το ίδιο πάντα για τους χρήστες. Ποια αρχιτεκτονική είναι η καλύτερη για την εφαρμογή τους; Σε αυτό έρχεται να βοηθήσει το **TensorFlow 2 Detection Model Zoo** (παρέχεται από το Tensorflow Object Detection), που παρέχει μία μεγάλη λίστα από μοντέλα για εντοπισμό αντικειμένων (ενός και δύο σταδίων), δίνοντας παράλληλα και πληροφορίες σχετικά με την ταχύτητα και την ακρίβειά τους κατά την διάρκεια δοκιμών στο σύνολο δεδομένων COCO (Common Objects in Context), καθώς και την μορφή της εξόδους τους (Boxes/Keypoints/Masks).

3.4 Το SSD-MobileNetV2

Το **SSD-MobileNetV2** είναι ένα εξαιρετικό μοντέλο για τον σκοπό της εργασίας αυτής, καθώς παρέχει ικανοποιητική ακρίβεια σε κοντινές αποστάσεις, όπως και την δυνατότητα εντοπισμού πολλαπλών αντικειμένων, ενώ παράλληλα απαιτεί χαμηλή υπολογιστική ισχύς. Επιπλέον, οι περισσότερες πράξεις που εκτελούνται μέσα στο δίκτυο αυτό, υποστηρίζονται πλήρως από το EdgeTPU που θα χρησιμοποιηθεί. Έτσι ο edgetpu compiler αποδίδει σχεδόν όλες τις διαδικασίες στην TPU, με αποτέλεσμα να είναι εφικτές πολύ μεγαλύτερες ταχύτητες. Άλλο ένα μεγάλο πλεονέκτημα είναι ότι το μοντέλο είναι γραμμένο στο Tensorflow, κάνοντας την αναπόφευκτη μετατροπή του σε TFlite μοντέλο, αρκετά εύκολη.

Το SSD-MobileNetV2 είναι ένα single shot detector (SSD) και η αρχιτεκτονική του αποτελείται από δύο βασικά σκέλη. Αρχικά, το στρώμα εισόδου του δέχεται εικόνες διαστάσεων 300x300 pixel (προεπιλογή - μπορεί να το αλλάξει ο χρήστης στις ρυθμίσεις). Οι εικόνες αυτές τροφοδοτούνται στο συνελικτικό νευρωνικό δίκτυο MobileNetV2, το οποίο είναι υπεύθυνο για την δημιουργία χαρτών χαρακτηριστικών. Μερικά ακόμα συνελικτικά στρώματα (στρώματα SSD) είναι τοποθετημένα ώστε να δημιουργηθούν περισσότεροι χάρτες με διαφορετικά μεγέθη. Το δεύτερο σκέλος λαμβάνει τα δεδομένα από το δίκτυο, κάνει τις προβλέψεις για όλους τους χάρτες χαρακτηριστικών και μετά την πραγματοποίηση non-maximum suppression, εμφανίζει τα

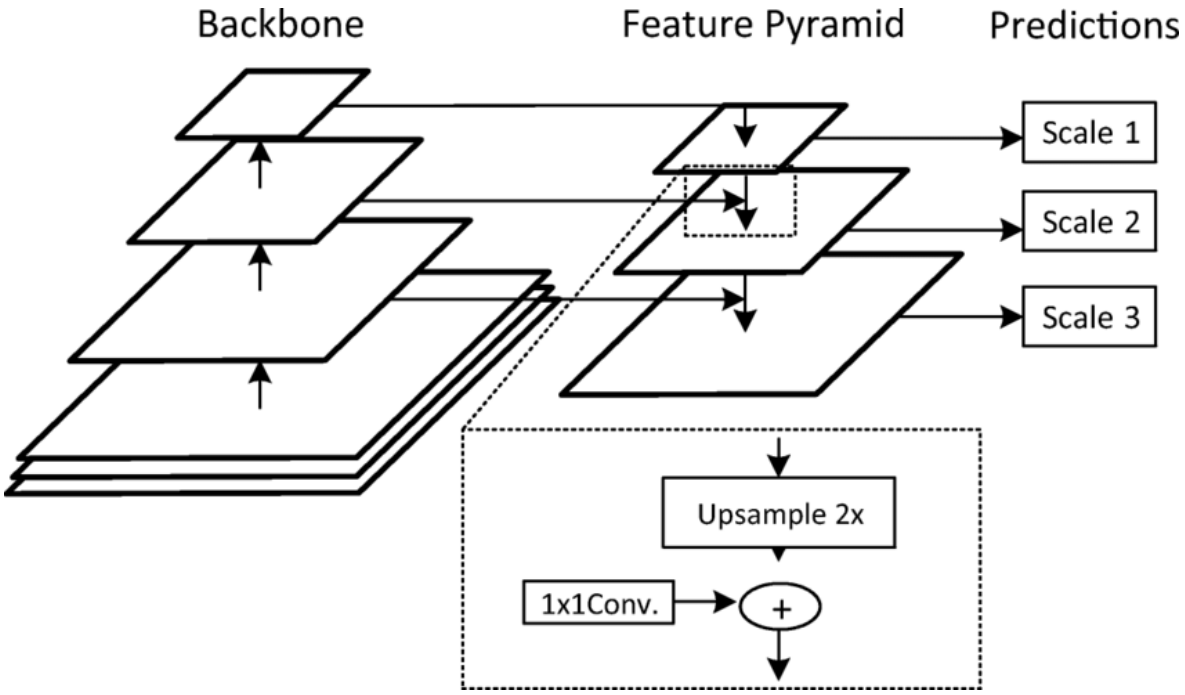
αποτελέσματα του εντοπισμού, δηλαδή την κατηγοριοποίηση του αντικειμένου και την σχεδίαση του αντίστοιχου bounding box.



Εικόνα 3.2: Η αρχιτεκτονική του SSD-MobileNetV2 [25]

3.5 Το SSD-MobileNetV2-FPNlite

Άλλο ένα καταπληκτικό υποψήφιο μοντέλο για την διπλωματική εργασία είναι το **SSD-MobileNetV2-FPNlite**. Χρησιμοποιεί και εκείνο το συνελικτικό νευρωνικό δίκτυο MobileNetV2 ως το βασικό κομμάτι του και στην ουσία η μόνη διαφορά που έχει με το SSD-MobileNetV2, είναι η προσθήκη του Feature Pyramid Network Lite (FPNlite). Αυτό το δίκτυο είναι μία πιο απλή μορφή της αρχιτεκτονικής FPN, όπου χάρτες χαρακτηριστικών που δημιουργούνται από το MobileNetV2 και τα στρώματα SSD, συνδυάζονται για την δημιουργία ακόμη περισσότερων χαρτών, δημιουργώντας μία "πυραμίδα" από χάρτες χαρακτηριστικών πολλαπλών διαστάσεων. Τα FPN θεωρητικά βελτιώνουν την ακρίβεια και την ικανότητα εντοπισμού ενός μοντέλου, ωστόσο συνήθως προκαλούν και μία πτώση στην ταχύτητά του.



Εικόνα 3.3: Δομή ενός FPN [26]

3.6 Η Python

Η **Python** είναι μία **διερμηνευόμενη** (interpreted) γλώσσα προγραμματισμού υψηλού επιπέδου (high level) που δίνει έμφαση στην εύκολη κατανόηση από τον άνθρωπο. Πρόκειται για μία δυναμική γλώσσα που ανήκει στις γλώσσες προστακτικού προγραμματισμού (imperative programming) και υποστηρίζει **αντικειμενοστραφές** (object-oriented) και **διαδικαστικό** (procedual) **προγραμματιστικό υπόδειγμα** (programming paradigm). Η Python επίσης έχει την δυνατότητα να εκτελείται σε πολλά λειτουργικά συστήματα, οπότε σπάνια υπάρχουν προβλήματα συμβατότητας. Επίσης είναι open source και διαθέτει πολλά δωρεάν **εργαλεία, modules και βιβλιοθήκες** (συλλογή από modules) που περιέχουν όλες τις απαραίτητες συναρτήσεις για την δημιουργία πολύπλοκων εφαρμογών, συμπεριλαμβανομένου και της μηχανικής μάθησης. Το μεγάλο μειονέκτημά της είναι ότι είναι αρκετά πιο αργή από άλλες γλώσσες προγραμματισμού όπως η C και η C++ [27].

Οι τύποι δεδομένων που χρησιμοποιούνται στην Python είναι [27]:

- **int** - ακέραιος
- **float** - κινητής υποδιαστολής
- **str (string)** - συμβολοσειρά
- **bool (boolean)** - λογικός τύπος (True/False)
- **NoneType** - Δεν έχει κάποια τιμή

Κατά την δήλωση μίας μεταβλητής δεν δηλώνεται ο τύπος της. Για να μπορέσει ο χρήστης να δει τον τύπο πρέπει να χρησιμοποιήσει μία συνάρτηση που λέγεται type().

Οι ενσωματωμένες δομές δεδομένων στην Python είναι [27]:

- η **Λίστα** (List): μοιάζει με πίνακα αλλά μπορεί να περιέχει ανομοιογενή στοιχεία.
- η **Πλειάδα** (Tuple): είναι σαν μία λίστα που τα στοιχεία της είναι αμετάβλητα.
- το **Σύνολο** (Set): μία συλλογή από μοναδικά δεδομένα.
- το **Λεξικό** (Dictionary): δομή δεδομένων που περιέχει ζευγάρια τιμών.

Άλλη μία ενδιαφέρουσα ιδιαιτερότητα που έχει, είναι η χρήση αριστερών εσοχών για την διαφοροποίηση των **μπλοκ του κώδικα** (code blocks), ενώ άλλες γλώσσες όπως η C και η Java χρησιμοποιούν αγκύλες ({}). Στην Python το μπλοκ κώδικα αρχίζει στο σημείο που η αριστερή εσοχή αυξάνεται προς τα δεξιά (συνήθως με το πάτημα του πλήκτρου tab) και τελειώνει εκεί που καταργείται η εσοχή (επιστρέφει στην προηγούμενη θέση). Αν η εσοχή βρίσκεται μέσα σε μία άλλη, τότε το μπλοκ αυτό είναι μέσα σε ένα άλλο μπλοκ κώδικα (nested block) [27].

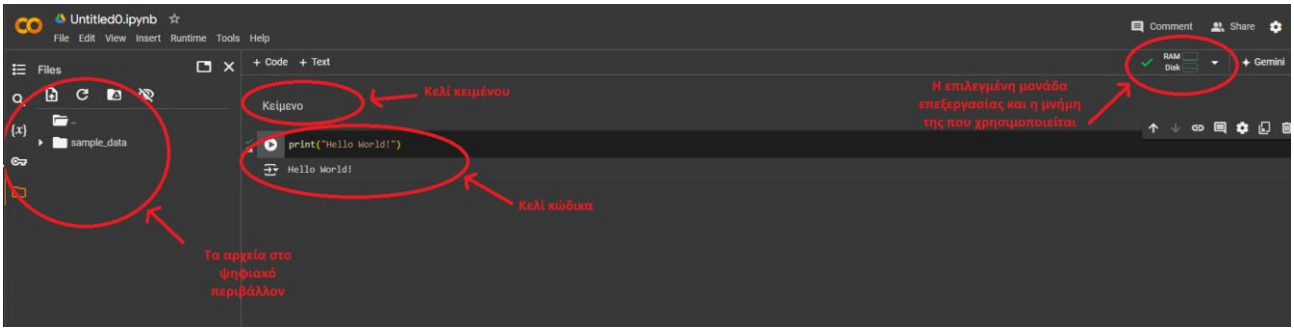
Όσον αφορά τους βρόχους επανάληψης (loops), η Python δίνει την επιλογή για [27]:

- **For** loops: επανάληψη του μπλοκ κώδικα σύμφωνα με όρια που έχει ορίσει ο χρήστης
- **While** loops: επανάληψη του μπλοκ κώδικα μέχρι να ικανοποιηθεί μία συνθήκη

Όπως και σε πολλές άλλες γλώσσες προγραμματισμού, υπάρχει και η δυνατότητα δημιουργίας **συναρτήσεων** (functions) [27]. Οι συναρτήσεις περιέχουν κώδικα που μπορεί να εκτελεστεί μόλις κληθούν με το όνομά τους. Για να οριστεί μία συνάρτηση αρκεί να χρησιμοποιηθεί το def και μετά να δοθεί το όνομά της και μέσα σε παρένθεση, οι απαιτούμενες τιμές εισόδου που χρειάζονται για να προκύψει το επιθυμητό αποτέλεσμα. Είναι ιδιαιτέρως χρήσιμες σε περιπτώσεις που ο κώδικας πρέπει να εκτελεστεί πολλαπλές φορές με διαφορετικά δεδομένα.

3.7 Το Google Colab

Το **Google Colab** είναι μία υπηρεσία cloud της Google, μέσω της οποίας επιτρέπεται η εκτέλεση κώδικα Python σε ένα προσωρινό εικονικό περιβάλλον. Επιτρέπει την δημιουργία πολλαπλών κελιών με κώδικα που μπορούν να εκτελεστούν ξεχωριστά με την εντολή του προγραμματιστή. Επιπλέον, υπάρχουν και κελιά κειμένου, ώστε να γίνεται εύκολη η επεξήγηση της διαδικασίας που ακολουθείται. Με τον συνδυασμό αυτών των δύο λειτουργιών, μπορεί εύκολα να δημιουργηθεί ένα **Colab Notebook**, το οποίο μπορεί να μοιραστεί και να εκτελεστεί από άλλους χρήστες [28].



Εικόνα 3.4: Το User Interface (UI) του Google Colab

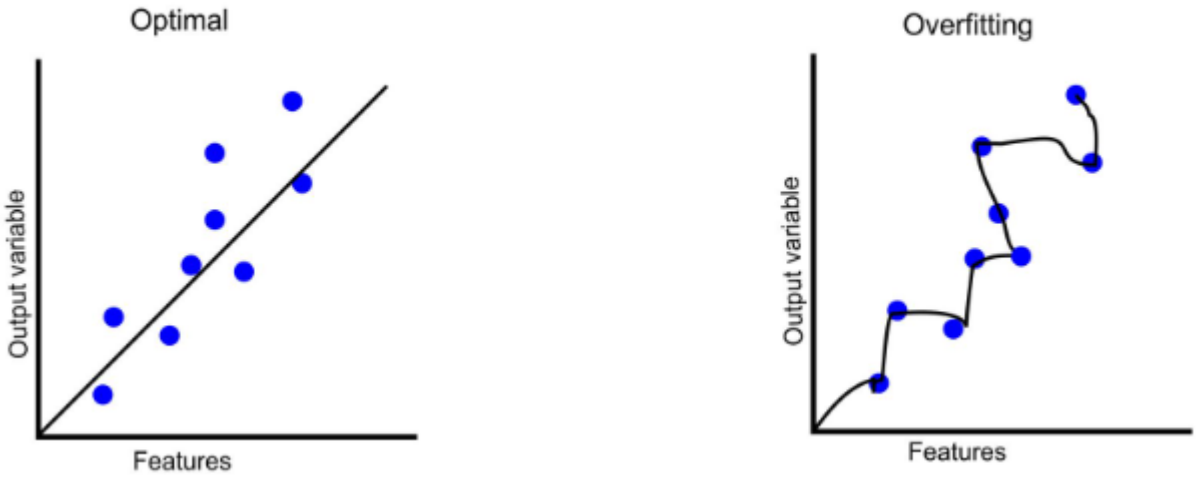
Εκτός αυτών, η Google παρέχει πολλές επιλογές **μονάδων επεξεργασίας** (processing unit - PU). Η μόνη δωρεάν επιλογή είναι μία τυπική κεντρική μονάδα επεξεργασίας (Central Processing Unit – CPU), ενώ με την αγορά υπολογιστικών πόρων (computing units) μπορούν να αξιοποιηθούν και διάφορες **μονάδες επεξεργασίας γραφικών** (Graphics Processing Unit – GPU) **και τανυστών** (Tensor Processing Unit – TPU). Τις περισσότερες φορές, όταν πρόκειται να γίνει η εκπαίδευση ενός μοντέλου μηχανικής μάθησης, είναι επιθυμητή μία GPU, καθώς μειώνει δραστικά τον συνολικό χρόνο που απαιτείται για την ολοκλήρωση της διαδικασίας. Για παράδειγμα, ένα μοντέλο που μπορεί να χρειαστεί μόλις 4 ώρες για την εκπαίδευσή του (με Transfer Learning) με την χρήση μίας GPU, θα χρειαστεί περίπου 58 ώρες με την CPU.

Ο τρόπος με τον οποίο είναι δυνατή η χρήση της πολύ μεγαλύτερης υπολογιστικής ισχύος μίας GPU για την εκπαίδευση των μοντέλων, είναι μέσω ενός API της NVIDIA που λέγεται **CUDA** (Compute Unified Device Architecture). Μέσω των βιβλιοθηκών CUDA, οι προγραμματιστές έχουν την δυνατότητα να εκτελούν κομμάτια του κώδικά τους σε επιταχυντές GPU (GPU accelerators), με μόνη απαίτηση την χρήση ορισμένων επεκτάσεων. Οι γλώσσες προγραμματισμού που υποστηρίζονται αυξάνονται συνεχώς, με μερικές από τις ήδη διαθέσιμες να είναι η C, η C++, η Fortran, η Python και το Matlab [29].

Για την εκπαίδευση και την εξαγωγή του μοντέλου της διπλωματικής εργασίας, θα δημιουργηθεί ένα Colab Notebook στο οποίο θα ακολουθείται ολόκληρη η διαδικασία, που περιλαμβάνει την εγκατάσταση των απαραίτητων βιβλιοθηκών και εργαλείων, την προεπεξεργασία των δεδομένων, την εκπαίδευση και δοκιμή του μοντέλου.

4 ΚΕΦΑΛΑΙΟ 4ο : Δημιουργία και εκπαίδευση του TFlite μοντέλου

Στην ενότητα αυτή θα γίνει η λεπτομερής περιγραφή της διαδικασίας που ακολουθήθηκε προκειμένου να δημιουργηθεί το μοντέλο που θα εντοπίζει την ύπαρξη ή μη ύπαρξη κράνους σε οδηγούς και συνεπιβάτες μηχανής ή ποδηλάτου. Όπως αναφέρθηκε και παραπάνω, η εκπαίδευση και η δοκιμή του μοντέλου θα γίνει στο προγραμματιστικό περιβάλλον του Google Colab και συγκεκριμένα, θα αξιοποιηθεί η L4 GPU που παρέχει η Google. Για να γίνει όμως η εκπαίδευση χρειάζονται πολλά δεδομένα υπό την μορφή ενός **συνόλου δεδομένων** (dataset) που θα χωρίζεται ποσοστιαία σε ένα υποσύνολο για την **εκπαίδευση** (training), ένα για την **επαλήθευση** (validation), και ένα για την **δοκιμή** (testing). Ο λόγος που χρειάζεται η επαλήθευση είναι για να αποφευχθεί το φαινόμενο του **overfitting**, στο οποίο το μοντέλο προσαρμόζει τα βάρη του ώστε να έχει πολύ καλές επιδόσεις για το σύνολο δεδομένων της εκπαίδευσης, χάνοντας παράλληλα την ικανότητα να λειτουργεί καλά υπό καινούργιες συνθήκες.



Εικόνα 4.1 Το φαινόμενο του overfitting [30]

4.1 Δημιουργία του συνόλου δεδομένων

Το σύνολο δεδομένων ή dataset που χρειάζεται για να δημιουργηθεί ένα μοντέλο για object detection απαιτεί:

α) έναν μεγάλο αριθμό εικόνων (ή βίντεο) που θα περιέχει όλα τα αντικείμενα των επιθυμητών κλάσεων (εξόδων) σε διάφορες γωνίες, επίπεδα φωτεινότητας και απόστασης. Για την καλύτερη προσαρμογή του μοντέλου, συνιστάται τα δεδομένα των κλάσεων να είναι ποσοτικά και ποιοτικά όμοια.

β) να γίνει σωστά η διαδικασία του labeling των εικόνων, αποδίδοντας στην καθεμία τα βέλτιστα bounding boxes και τα ονόματα των κλάσεων που αντιπροσωπεύουν, για το κάθε αντικείμενο

ενδιαφέροντος που βρίσκεται σε αυτή. Είναι άκρως σημαντικό να γίνει σωστά αυτό το βήμα, καθώς με αυτά τα δεδομένα θα γίνει η εκπαίδευση του μοντέλου.

Το σύνολο δεδομένων που δημιουργήθηκε για την εργασία αυτή, υπέστη πολλές διαφορετικές αλλαγές οι οποίες προφανώς επηρέασαν το τελικό αποτέλεσμα, μερικές φορές θετικά και άλλες αρνητικά. Παρακάτω θα γίνει μια συνοπτική περιγραφή όλων των αλλαγών, ωστόσο τα αποτελέσματά τους θα αναλυθούν σε άλλη ενότητα.

Αρχικά, έγινε χρήση ενός δωρεάν dataset από το Kaggle [31]. Το dataset αυτό περιέχει 764 εικόνες με τα αντίστοιχα labels τους σε μορφή **Pascal VOC** και χωρισμένα σε δύο κλάσεις (With Helmet, Without Helmet).

Το Pascal VOC αποτελεί μία μορφή labeling που αξιοποιεί αρχεία xml, η οποία είναι πολύ φιλική προς το ανθρώπινο μάτι, ωστόσο δεν είναι ιδιαίτερα χρήσιμη για τα μοντέλα εντοπισμού αντικειμένων. Για την κάθε εικόνα υπάρχει και ένα αντίστοιχο αρχείο xml, με το ίδιο όνομα, όπου καταγράφονται όλες οι απαραίτητες πληροφορίες της ίδιας της εικόνας, καθώς και των αντικειμένων που βρίσκονται σε αυτή (κλάση, συντεταγμένες των bounding boxes). Το Pascal VOC κάνει εύκολη την δημιουργία συνόλου δεδομένων για αυτές τις εφαρμογές, αλλά όταν έρθει η στιγμή να γίνει η εισαγωγή του για την εκπαίδευση του μοντέλου, συνήθως μετατρέπεται σε κάποιο άλλο format.

```xml
<annotation>
    <folder>data2</folder>
    <filename>BikesHelmets0.png</filename>
    <path>C:\Users\ \BikesHelmets0.png</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>400</width>
        <height>267</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>With Helmet</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>28</xmin>
            <ymin>104</ymin>
            <xmax>80</xmax>
            <ymax>160</ymax>
        </bndbox>
    </object>
```

Εικόνα 4.2: Δομή ενός αντικειμένου σε ένα xml αρχείο

Μετά από ανάλυση του συνόλου δεδομένων, διαπιστώθηκε ότι υπάρχει ένας αρκετά μεγάλος αριθμών διπλών εγγραφών (duplicates). Συγκεκριμένα, 249 εικόνες ήταν διπλές, τριπλές, ακόμη και τετραπλές. Αυτό προφανώς δεν είναι επιθυμητό καθώς αυξάνεται πολύ η πιθανότητα του overfitting, με το μοντέλο να προσαρμόζεται πολύ καλά σε ορισμένα δεδομένα που εμφανίζονται πολλαπλές φορές. Επίσης κατά την διάρκεια των δοκιμών, επειδή οι εικόνες χωρίζονται τυχαία, μπορεί πολλές από τις επιλεγμένες να μην είναι άγνωστες στο μοντέλο και να δημιουργηθεί μία λάθος εικόνα για την απόδοσή του. Το επόμενο βήμα λοιπόν, ήταν η αφαίρεση αυτών των εικόνων, ώστε να είναι όλες ξεχωριστές.

Το πρόβλημα που προέκυψε μετά από αυτή την αλλαγή ήταν η έλλειψη δεδομένων, καθώς το σύνολο των εικόνων ήταν πλέον μειωμένο κατά 33%. Η λύση ήταν να γίνει η πρόσθεση εικόνων και να δημιουργηθούν τα αντίστοιχα xml αρχεία. Επιλέχθηκαν λοιπόν με προσοχή, 500 συνολικά ποιοτικές εικόνες ατόμων με και χωρίς κράνος, μέσω του Google Images. Για να γίνει μάλιστα πιο εύκολη η οργάνωση της δουλειάς, χωρίστηκαν σε 5 υποσύνολα των 100 εικόνων. Εν τέλει, ο συνολικός αριθμός εικόνων για το νέο σύνολο δεδομένων έφτασε τις 1015, που θεωρητικά θα προσφέρει αρκετά ικανοποιητικά αποτελέσματα. Το επόμενο στάδιο είναι να οριστούν τα bounding

boxes και οι κλάσεις, για καθεμία από τις εικόνες. Πώς όμως μπορεί να γίνει εύκολα και γρήγορα το labeling για το καινούργιο dataset;

Σε αυτές τις περιπτώσεις η χρήση ενός εργαλείου είναι μονόδρομος, καθώς η εγγραφή των συντεταγμένων του κάθε bounding box είναι μία πολύ δύσκολη και χρονοβόρα διαδικασία χωρίς αυτό. Ευτυχώς, υπάρχουν πολλές τέτοιες εφαρμογές στο διαδίκτυο. Αυτή που χρησιμοποιήθηκε στην διπλωματική εργασία είναι το **labelImg** [32].

Το labelImg επιτρέπει στον χρήστη να σχεδιάζει με τον κέρσορά του τα bounding boxes πάνω σε κάθε εικόνα και στην συνέχεια να ορίζει τις κλάσεις που τα αντιπροσωπεύουν. Στο τέλος, αποθηκεύονται τα δεδομένα του labeling σε αρχεία xml με την μορφή Pascal VOC. Κατόπιν ελέγχου του αρχικού συνόλου δεδομένων παρατηρήθηκαν πολλά λάθη στην χάραξη των bounding boxes και στις κλάσεις που είχαν οριστεί. Συνεπώς, η καλύτερη λύση ήταν η πραγματοποίηση της διαδικασίας από την αρχή, ακολουθώντας τους παρακάτω κανόνες:

1. Όλα τα αντικείμενα που φαίνονται σε κάθε εικόνα θα λαμβάνονται υπόψη, εκτός αν είναι πολύ χαμηλή η ορατότητά τους (π.χ. είναι πολύ μακριά και η εικόνα είναι πολύ χαμηλής ανάλυσης).
2. Τα κουτιά της κλάσης ‘Without Helmet’ πρέπει να περιέχουν μόνο το κεφάλι του ατόμου, δηλαδή να έχουν ως άνω άκρο την κορυφή της κεφαλής, ως κάτω άκρο το πηγούνι, ως αριστερό και δεξί άκρο το αντίστοιχο αυτί (αν φαίνεται). Αν δεν είναι ορατό κάποιο αυτί τότε η σχεδίαση γίνεται μέχρι το σημείο που σταματάει να υπάρχει πρόσωπο.
3. Σε περίπτωση που κάποιο αντικείμενο επικαλύπτεται μερικώς από κάποιο άλλο, αλλά η συνολική ορατότητά του είναι επαρκής, τότε το αντικείμενο αυτό λαμβάνεται υπόψη και η σχεδίαση του κουτιού γίνεται σαν να μην υπάρχει κάποιο εμπόδιο.
4. Τα κράνη μηχανής που καλύπτουν όλο το πρόσωπο, θα έχουν bounding box που θα περιλαμβάνει ολόκληρο το κράνος. Ωστόσο, αν το κράνος είναι ποδηλάτου ή μηχανής που όμως δεν καλύπτει όλο το πρόσωπο, τότε περιλαμβάνεται το κράνος μαζί με το υπόλοιπο πρόσωπο.

Το καινούργιο dataset ήδη φαινόταν αρκετά βελτιωμένο συγκριτικά με το προηγούμενο. Μετά από αρκετές ανασκοπήσεις όμως ήταν εμφανές ότι μπορούσαν να γίνουν πολλές ακόμη αλλαγές για την περαιτέρω αύξηση της ποιότητάς του. Ένα πρόβλημα που παρατηρήθηκε ήταν η ύπαρξη εικόνων με άτομα που ήταν με την πλάτη στην κάμερα. Η συσκευή αυτή όμως θα λειτουργεί σε σημεία όπου τα άτομα θα κοιτάνε προς την πλευρά της κάμεράς της. Ήταν σαφές λοιπόν, ότι στο επόμενο βήμα, έπρεπε να αφαιρεθούν αυτές οι εγγραφές, ώστε να μην τροφοδοτούνται στο μοντέλο δεδομένα που δεν θα χρειαστεί να αντιμετωπίσει κατά την διάρκεια της λειτουργίας του.

Η τελευταία αλλαγή που έγινε ήταν η περαιτέρω αφαίρεση ορισμένων εικόνων, όπου τα άτομα φορούσαν κάποιου είδους μπούρκα ή μαντήλα. Διαπιστώθηκε ότι αυτές οι περιπτώσεις ήταν αρκετά

σπάνιες για τα δεδομένα της Ελλάδας και ταυτόχρονα λόγω της ομοιότητάς τους με τα κράνη, θα εισήγαγαν πολλαπλά προβλήματα στον καθορισμό των συναπτικών βαρών κατά την διάρκεια της εκπαίδευσης του μοντέλου. Αξίζει να σημειωθεί ότι σε αντίθεση με την πρώτη περίπτωση αφαίρεσης δεδομένων (αφαίρεση duplicate εικονών), κατά την διάρκεια των τελευταίων δύο βημάτων, οι εικόνες που απορρίφθηκαν ήταν πολύ λίγες, άρα δεν υπήρχε κάποιος λόγος να αντικατασταθούν με νέες. Συγκεκριμένα, 25 εικόνες είχαν άτομα με γυρισμένη την πλάτη τους στην κάμερα και 32 ήταν αυτές που περιείχαν άτομα με μπούρκα ή μαντήλα. Έτσι, το σύνολο δεδομένων κατέληξε στην τελική του μορφή, με τον συνολικό αριθμό εικόνων να είναι 958.

Σημείωση: Για την βέλτιστη δημιουργία ενός συνόλου δεδομένων, προτείνεται η συλλογή εικόνων/δεδομένων από το περιβάλλον που πρόκειται να εφαρμοστεί. Για την περίπτωση αυτή, θα ήταν ιδανικό να υπήρχαν εικόνες από τους δρόμους της Ελλάδας, σε διάφορα επίπεδα φωτισμού, διάφορες αποστάσεις και γωνίες και με διαστάσεις όμοιες με αυτές που θα τροφοδοτούνται στο μοντέλο μέσω της κάμερας. Το τελευταίο είναι σημαντικό, γιατί αν η κάμερα δεν έχει την δυνατότητα να παρέχει εικόνες με τις ίδιες διαστάσεις με αυτές που δέχεται σαν είσοδο το μοντέλο, τότε τουλάχιστον, η κάθε εικόνα εισόδου να υφίσταται την ίδια παραμόρφωση με τις εικόνες που έχει εκπαιδευτεί.

4.2 Εγκατάσταση πακέτων

Με το σύνολο δεδομένων έτοιμο, μπορεί πλέον να ξεκινήσει η διαδικασία της δημιουργίας του μοντέλου. Αρχικά όμως, πρέπει να γίνει η εγκατάσταση όλων των απαραίτητων βιβλιοθηκών και εργαλείων. Όπως αναφέρθηκε προηγουμένως, θα χρησιμοποιηθεί το Tensorflow Object Detection API [33]. Πριν γίνει αυτό, πρέπει πρώτα να δοθεί εντολή για την απεγκατάσταση του πακέτου Cython. Αν δεν γίνει αυτό πρώτα, υπάρχει μεγάλη πιθανότητα, κατά την διάρκεια της δοκιμής της εγκατάστασης, να εμφανιστεί το σφάλμα “no module named object_detection”. Συνεπώς είναι απαραίτητο. Στην συνέχεια, στο ίδιο κελί γίνεται και η αντιγραφή των αρχείων του API από το GitHub και συγκεκριμένα, μόνο η τελευταία έκδοσή του.

Με τα αρχεία πλέον στο εικονικό περιβάλλον, είναι αναγκαία η εφαρμογή του protoc buffer σε όλα τα .proto αρχεία. Το protoc buffer ή Protocol Buffer είναι ένας μηχανισμός που αξιοποιεί η Google και μέσω αυτού δημιουργεί κλάσεις ή κώδικα (π.χ., Python, Java, C++) που μπορεί να χρησιμοποιηθεί σε μία εφαρμογή για σειριοποίηση, αποσειριοποίηση και διαχείριση των δομών δεδομένων που ορίζονται στα αρχεία .proto.

Το επόμενο βήμα κανονικά, είναι να εγκατασταθούν τα πακέτα που ορίζονται στο αρχείο setup.py, που βρίσκεται στο “models/research/”. Ωστόσο η έκδοση των πακέτων Tensorflow, συγκεκριμένα το “tf-models-official", δημιουργούσε πολλά προβλήματα συμβατότητας με άλλα πακέτα και βιβλιοθήκες, με αποτέλεσμα να μην είναι δυνατή η επιτυχής εγκατάσταση του Tensorflow Object Detection. Η λύση ήταν να δημιουργηθεί ένα κελί κώδικα, όπου θα ανοίγει το αρχείο setup.py

και θα αντικαθιστά την γραμμή κώδικα που ορίζει την έκδοση των πακέτων που συσχετίζονται με το Tensorflow, με μία έκδοση που δεν οδηγεί σε προβλήματα συμβατότητας. Μετά από μερικές δοκιμές, βρέθηκε ότι μία αρκετά ικανοποιητική έκδοση ήταν το Tensorflow 2.8. Άρα μέσω του κελιού αυτού και του εργαλείου "re", αντικαθίσταται η επιθυμητή έκδοση του "tf-models-official" από '>=2.5.1' (οποιαδήποτε έκδοση είναι πιο καινούργια από την 2.5.1) σε '==2.8.0' (υποχρεωτικά την 2.8.0).

Έχοντας λοιπόν ετοιμάσει σωστά το αρχείο setup.py, μπορεί να γίνει η εκτέλεσή του και στην συνέχεια η εγκατάσταση του ίδιου του Tensorflow 2.8.0. Ένα άλλο πακέτο που μερικές φορές προκαλούσε προβλήματα συμβατότητας με τα υπόλοιπα κατά την αυτόματη εγκατάστασή του, ήταν το pyyaml, οπότε μπορεί πολύ εύκολα να εγκατασταθεί ξεχωριστά σε μία συμβατή έκδοση (όπως η 5.3).

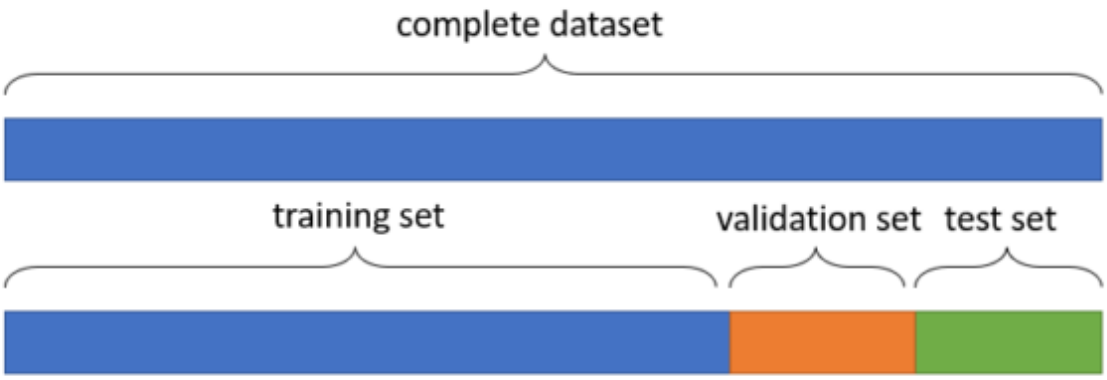
Το Tensorflow Object Detection API θα έπρεπε να έχει εγκατασταθεί με επιτυχία πλέον. Εφόσον πρόκειται να αξιοποιηθεί GPU για την διαδικασία της εκπαίδευσης, πρέπει να γίνει και η εγκατάσταση των απαραίτητων βιβλιοθηκών CUDA. Το στάδιο αυτό δεν είναι υποχρεωτικό αν η διαθέσιμη GPU, δεν έχει την δυνατότητα υποστήριξης της επιτάχυνσης υλικού μέσω CUDA. Η έκδοση που προτείνεται και είναι συμβατή με το Tensorflow 2.8.0 είναι το CUDA 11.0. Μόλις τελειώσει η διαδικασία αυτή, είναι δυνατή η συνέχεια προς την εισαγωγή και προεπεξεργασία των δεδομένων. Ωστόσο, προτείνεται η εκτέλεση ενός ακόμη αρχείου που λέγεται "model_builder_tf2_test.py" , το οποίο επαληθεύει την σωστή λειτουργία του API και είναι ένας πολύ εύκολος και χρήσιμος τρόπος, ώστε να σιγουρευτεί ο χρήστης ότι η διαδικασία εγκατάστασης έχει ολοκληρωθεί με επιτυχία.

4.3 Προεπεξεργασία των δεδομένων

Στο Google Colab υπάρχουν δύο βασικές μέθοδοι για την εισαγωγή δεδομένων. Η πρώτη είναι η απευθείας μεταφόρτωση ενός αρχείου (αν πρόκειται για φάκελο, τότε πρέπει υποχρεωτικά να βρίσκεται σε συμπιεσμένη μορφή, όπως zip/7z/rar) και η δεύτερη είναι η μεταφορά του μέσω της υπηρεσίας Google Drive. Γενικά προτείνεται στις περιπτώσεις μεγάλων αρχείων, η δεύτερη επιλογή. Τα σύνολα δεδομένων για μηχανική μάθηση τυπικά αποτελούνται από αρκετά μεγάλους φακέλους, και η περίπτωση των αρχείων για την διπλωματική εργασία αυτή δεν αποτελεί εξαίρεση. Μάλιστα, οι συμπιεσμένοι φάκελοι των πολλαπλών εκδοχών των dataset που χρησιμοποιήθηκαν, κυμαίνονταν (σε μέγεθος) από 390MB (Megabyte - 10^6 byte) έως και 590 MB. Άλλος ένας καθοριστικός παράγοντας είναι η ταχύτητα μεταφόρτωσης που διαθέτει ο κάθε χρήστης. Η επιλογή ήταν σαφής και τα σύνολα δεδομένων θα μεταφερόντουσαν από το Google Drive στο εικονικό περιβάλλον. Για να γίνει αυτό όμως, πρέπει πρώτα να γίνει μία διαδικασία που ονομάζεται Drive mounting, όπου πραγματοποιείται η μεταφορά όλων των αρχείων που βρίσκονται εκείνη την στιγμή στον αποθηκευτικό χώρο του cloud στο περιβάλλον του Google Colab. Απο εκεί, με μερικές εντολές (cp (αντιγραφή αρχείου), mkdir (δημιουργία ενός directory), unzip (αποσυμπίεση)), δημιουργείται ένας

φάκελος ‘data’ και στην συνέχεια σε αυτόν, αντιγράφεται ο συμπιεσμένος φάκελος με τα δεδομένα και αποσυμπιέζεται, ώστε στο τέλος να υπάρχει ένας φάκελος ‘'data’ με έναν υποφάκελο ‘all_data’ που θα περιέχει όλες τις εικόνες και τα xml αρχεία που θα περάσουν από το στάδιο της προεπεξεργασίας.

Σκοπός του επόμενου σταδίου είναι η οργάνωση των αρχείων και ο διαχωρισμός τους σε τρία υποσύνολα. Το πρώτο είναι το training και θα περιέχει τις εικόνες που θα χρησιμοποιηθούν για να εκπαιδεύσουν το μοντέλο. Το δεύτερο είναι το validation το οποίο θα βοηθήσει στην αποφυγή του overfitting κατά την διάρκεια της εκπαίδευσης και το τρίτο είναι το testing, που θα αποτελείται από ένα σύνολο άγνωστων εικόνων για το μοντέλο και θα χρησιμεύσει για την δοκιμή της ακρίβειάς του. Επιπλέον, είναι καλό να ελέγχονται τα αρχεία xml για τυχών λάθη, όπως οι μη έγκυρες συντεταγμένες των bounding boxes (π.χ. να βρίσκονται εκτός εικόνας).



Εικόνα 4.3: Οπτικοποίηση του διαχωρισμού του συνόλου δεδομένων σε τρία υποσύνολα [34]

Τα παραπάνω καλείται να πραγματοποιήσει το επόμενο κελί κώδικα. Στην αρχή εισάγονται τα απαραίτητα python modules, τα οποία είναι: το os που προσφέρει την δυνατότητα αλληλεπίδρασης με το λειτουργικό σύστημα, το random για την εφαρμογή αλγορίθμων παραγωγής τυχαίων αριθμών, το shutil που επιτρέπει την μετακίνηση αρχείων και το xml.etree.ElementTree για την ανάγνωση των xml. Στην συνέχεια δημιουργούνται τέσσερις μεταβλητές τύπου string που περιέχουν τα directories των φακέλων που θα περιέχουν τις εικόνες και τα xml αρχεία που τους αντιστοιχούν για τα υποσύνολα training (training_folder), validation (validation_folder), testing (testing_folder) και corrupted (corrupted_folder) (για τα μη έγκυρα αρχεία xml). Επίσης ορίζεται και μία ακόμη μεταβλητή που δείχνει τον φάκελο με όλα τα δεδομένα (data_folder). Μέσω των εντολών os.path.exists και os.makedirs, ελέγχεται αν υπάρχουν οι φάκελοι στα directories που δείχνουν οι μεταβλητές που αναφέρθηκαν προηγουμένως και αν δεν υπάρχουν, τότε δημιουργούνται.

Για τον έλεγχο των αρχείων xml, δημιουργήθηκε μία συνάρτηση που λαμβάνει ως είσοδο το directory του φακέλου που πρέπει να ελεγχθεί και επιστρέφει μία λίστα με όλα τα προβληματικά αρχεία xml. Αναλυτικά, μέσω του os.listdir, λαμβάνονται όλα τα directories των αρχείων που βρίσκονται μέσα στον φάκελο και ελέγχεται ποια από αυτά έχουν ως κατάληξη το '.xml' μέσω του endswith. Τα αρχεία αυτά αποδομούνται και οι τιμές των διαστάσεών τους και των συντεταγμένων των bounding boxes παραχωρούνται στις μεταβλητές width, height (για τις διαστάσεις της εικόνας), xmin, ymin, xmax, ymax (για τις συντεταγμένες των bounding boxes). Αν έστω μία από τις τιμές xmin και ymin είναι μικρότερη του 0 ή μία από τις τιμές xmax και ymax είναι μεγαλύτερη από το width και το height αντίστοιχα, τότε το αρχείο θεωρείται προβληματικό και προστίθεται στην λίστα corrupted_xml. Μόλις ολοκληρωθεί ο βρόχος επανάληψης και έχουν ελεγχθεί όλα τα αρχεία του φακέλου, τότε επιστρέφεται η τελική μορφή της λίστας corrupted_xml.

Η επόμενη συνάρτηση αφορά την μεταφορά των xml αρχείων στον σωστό φάκελο σύμφωνα με το όνομά τους. Όπως αναφέρθηκε στην υποενότητα 4.1, όπου περιγράφεται η διαδικασία της δημιουργίας του συνόλου δεδομένων, το όνομα ενός xml και της εικόνας που περιγράφει, πρέπει να έχουν την ίδια βάση. Οπότε όταν γίνει η τυχαία κατανομή των εικόνων στα τρία υποσύνολα, πρέπει να υπάρχει και ένας αλγόριθμος που θα μεταφέρει τα xml στους φακέλους με την αντίστοιχη εικόνα τους. Σαν είσοδο θα λαμβάνει τα αρχεία των εικόνων του φακέλου και το directory του ιδίου του φακέλου που πρόκειται να μεταφερθούν. Αλλάζοντας την κατάληξη του κάθε αρχείου από '.png' σε '.xml' με την εντολή replace και ελέγχοντας αν υπάρχουν στον φάκελο που φιλοξενεί όλα αρχεία xml (το data_folder), μέσω της εντολής shutil.move, ολοκληρώνεται η μεταφορά, κατόπιν θετικού αποτελέσματος του ελέγχου.

Έχοντας πλέον όλες τις απαραίτητες συναρτήσεις, μπορεί να ξεκινήσει ο διαμερισμός των δεδομένων. Ξεκινώντας από τα προβληματικά xml, καλείται η συνάρτηση check_xml_files() που δημιουργήθηκε και κάθε αρχείο που περιέχεται στην λίστα corrupted_xml, μετακινείται στον φάκελο που δείχνει το corrupted_folder μαζί με την εικόνα του. Στην συνέχεια, ορίζεται μία νέα λίστα και μέσω των os.listdir και endswith, καταγράφονται τα directories όλων των υπόλοιπων αρχείων png. Η εντολή random.shuffle επιτρέπει το τυχαίο ανακάτεμα των αρχείων της λίστας και στην συνέχεια μέσω του len καταγράφεται ο συνολικός αριθμός των εικόνων. Οι εικόνες για την εκπαίδευση (training) θα αποτελούν το 80% του συνόλου, για την επαλήθευση (validation) το 10% και για τις δοκιμές (testing) το υπόλοιπο 10%. Με την βοήθεια μεταβλητών που περιέχουν τον αριθμό που αντιστοιχεί στο κάθε υποσύνολο μπορούν μεταφερθούν: το πρώτο 80% των εικόνων στον φάκελο training, το επόμενο 10% στο φάκελο validation και όσες περισσεύουν (περίπου το 10%) στον φάκελο testing. Το μόνο που μένει πλέον, είναι η κλήση της συνάρτησης move_xml_files() που δημιουργήθηκε προηγουμένως, για να μεταφερθούν τα αρχεία xml στους φακέλους που βρίσκονται οι εικόνες που αντιπροσωπεύουν. Προαιρετικά, με το print μπορεί να εκτυπωθούν στην κονσόλα: ο

συνολικός αριθμός εικόνων, το σύνολο τον εικόνων για training, validation και testing καθώς και τα corrupted xml.

Το Tensorflow Object Detection απαιτεί την δημιουργία ενός αρχείου τύπου .txt που ονομάζεται labelmap και περιέχει σε απλή μορφή τα ονόματα των κλάσεων που θα εντοπίζει το μοντέλο. Στο Colab αυτό επιτυγχάνεται μέσω του '%%bash', που επιτρέπει την εκτέλεση εντολών bash όπως το cat (concatenate), για την δημιουργία και επεξεργασία ενός αρχείου που θα τοποθετηθεί στο ψηφιακό περιβάλλον με όνομα labelmap.txt. Η εντολή αυτή συνοδεύεται από το '<<EOF >>', που επισημάνει την έναρξη του κειμένου με περιεχόμενο τα ονόματα των δύο κλάσεων (With Helmet, Without Helmet) σε διαφορετικές γραμμές. Η εντολή ολοκληρώνεται με το 'EOF'.

Στην προηγούμενη υποενότητα αναφέρθηκε ότι τα μοντέλα για object detection δεν χρησιμοποιούν γενικά δεδομένα σε μορφή Pascal VOC, οπότε συνήθως χρειάζεται να μετατραπούν σε μία άλλη μορφή. Το Tensorflow Object Detection API απαιτεί τα δεδομένα να βρίσκονται υπό την μορφή TFRecords που ενώ είναι δύσκολα κατανοητή από τον άνθρωπο, είναι πολύ αποδοτική και διαβάζεται εύκολα από το Tensorflow. Το πρώτο στάδιο για την μετατροπή αυτή είναι να δημιουργηθούν αρχεία csv για το κάθε υποσύνολο (training/validation/testing (προαιρετικό)), με τα δεδομένα που βρίσκονται στα αρχεία xml. Το κάθε αρχείο csv πρέπει να έχει την κάθε εγγραφή (bounding box) σε διαφορετική γραμμή. Αναλυτικά, πρώτα τοποθετείται ο αύξοντας αριθμός της εγγραφής, μετά το όνομα της εικόνας που βρίσκεται, το μήκος και το ύψος της εικόνας αυτής, η κλάση του αντικειμένου και τέλος οι συντεταγμένες του bounding box (xmin, ymin, xmax, ymax). Είναι φανερό λοιπόν, ότι μερικές φωτογραφίες θα παρέχουν περισσότερες εγγραφές από άλλες.

Για την δημιουργία του κώδικα χρειάστηκε η εισαγωγή των ακόλουθων python modules: το os για την αλληλεπίδραση με το λειτουργικό, το glob για την εύρεση αρχείων, το pandas για την επεξεργασία των δεδομένων και την δημιουργία του αρχείου και τέλος το xml.etree.ElementTree για την ανάγνωση και αποδόμηση των αρχείων xml. Εφόσον η μετατροπή πρέπει να γίνει για τουλάχιστον δύο υποσύνολα (training και validation), είναι καλή ιδέα να δημιουργηθεί μία συνάρτηση που θα πραγματοποιεί την μετατροπή. Η συνάρτηση xml_to_csv(), δέχεται σαν είσοδο το directory του φακέλου που περιέχει τα xml αρχεία. Στην συνέχεια, δημιουργείται μία αρχικά άδεια λίστα, όπου θα τοποθετηθούν οι εγγραφές. Μέσω του glob.glob, καταγράφονται όλα τα αρχεία του φακέλου που έχουν ως κατάληξη το '.xml'. Για το καθένα, εφαρμόζονται οι εντολές ET.parse και getroot, ώστε να γίνει η εξαγωγή των δεδομένων του αρχείου. Επιπλέον, το findall(΄object') βρίσκει όλες τις εγγραφές μέσα στο αρχείο, καθώς το στοιχείο 'object' σηματοδοτεί την ύπαρξη αντικειμένου. Από το καθένα, αποθηκεύεται το 'filename' που είναι το όνομα της εικόνας, το 'size [0]' που είναι το πλάτος της εικόνας (width), το 'size [1]' που είναι το ύψος της εικόνας (height) και τέλος η τιμή στο πρώτο 'παιδί' του στοιχείου με την μορφή κειμένου, που είναι το όνομα της κλάσης και οι τέσσερις τιμές στο 5ο 'παιδί' του στοιχείου που είναι οι συντεταγμένες του bounding box. Η διαδικασία αυτή επαναλαμβάνεται για όλα τα αρχεία και η τελική λίστα εισάγεται σε ένα pandas

dataframe με τα ονόματα των στηλών να είναι αυτά που αναφέρθηκαν και παραπάνω ('filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax'), το οποίο είναι και αυτό που επιστρέφεται από την συνάρτηση. Μόλις λοιπόν γίνει η κλήση της για τους φακέλους με τα δεδομένα, χρησιμοποιείται η εντολή to_csv που προσφέρει το pandas, πάνω σε κάθε dataframe που έχει επιστραφεί από την συνάρτηση xml_to_csv(), με την οποία αποθηκεύονται στο εικονικό περιβάλλον με την μορφή '.csv'. Προαιρετικά, ο χρήστης μπορεί να κάνει μία ανασκόπηση του περιεχομένου των αρχείων για την επαλήθευση της ορθής δομής τους, μέσω της εντολής pd.read_csv(το path του αρχείου) του pandas.

Το μόνο που μένει πλέον για την ολοκλήρωση της προεπεξεργασίας των δεδομένων είναι να μετατραπούν τα δεδομένα των αρχείων csv σε TFRecord. Σε αυτό το βήμα, χρησιμοποιήθηκε ο κώδικας 'generate_tfrecord.py' του datitran από το GitHub [35]. Αρχικά εισάγονται τα απαραίτητα modules που είναι το os, το pandas, το io, το tensorflow.compat.v1 αν η έκδοση του Tensorflow είναι μεγαλύτερη του 2.0 για καλύτερη συμβατότητα με παλαιότερα πακέτα (αλλιώς εισάγεται απλά το Tensorflow), το Image από το PIL, το dataset_util από το object_detection.utils και τέλος, τα namedtuple και OrderedDict από το collections.

Στην συνέχεια ορίζονται τα flags που θα ευθύνονται για την αποθήκευση των παραμέτρων που χρειάζονται για την εκτέλεση του κώδικα. Οι παράμετροι αυτοί είναι: η τοποθεσία του αρχείου csv, η τοποθεσία του αρχείου labelmap.txt, η τοποθεσία του φακέλου των εικόνων και τέλος η τοποθεσία που θα αποθηκευτεί το αρχείο tfrecord. Δύο βοηθητικές συναρτήσεις δημιουργούνται παρακάτω. Η πρώτη είναι η split, που λαμβάνει ένα pandas dataframe και ένα group και χωρίζει το dataframe σε πολλαπλά μικρότερα groups βασισμένα στο παρεχόμενο από τον χρήστη group. Η δεύτερη είναι και η πιο σημαντική. Λέγεται create_tf_example και λαμβάνει σαν εισόδους το group και την τοποθεσία των εικόνων. Διαβάζει τις εικόνες που περιέχονται στο group, τις μετατρέπει σε μορφή JPEG και εξάγει πρώτα τις διαστάσεις τους (width, height). Μετά κωδικοποιεί τα 'filenames' σε UTF-8 (για λόγους συμβατότητας) και αρχικοποιεί λίστες για τις συντεταγμένες των bounding boxes, τις κλάσεις σε μορφή κειμένου και τις κλάσεις σε μορφή ακέραιου αριθμού. Αφού γίνει αυτό, ανοίγει το αρχείο labelmap.txt και εξάγει τις κλάσεις που περιέχονται μέσα σε αυτό, αποθηκεύοντάς τες σε μία ακόμη λίστα. Στην συνέχεια κανονικοποιεί όλες τις τιμές των συντεταγμένων των bounding boxes ώστε να κυμαίνονται από το 0 έως το 1. Αυτό γίνεται μέσω της διαίρεσης των τιμών με τις αντίστοιχες τιμές πλάτους (για xmin και xmax) και ύψους (για ymin και ymax). Τα ονόματα των κλάσεων κωδικοποιούνται και αυτά σε UTF-8 και μετά μετατρέπονται σε έναν ακέραιο αριθμό που αντιστοιχεί στην σειρά που βρίσκονται στο αρχείο labelmap.txt. Το τελευταίο κομμάτι κατασκευάζει ένα αντικείμενο από όλα τα δεδομένα, το οποίο και επιστρέφει. Αυτό γίνεται μέσω του tf.train.Example που είναι ένα protocol buffer (protobuf) που σειριοποιεί αυτά τα δεδομένα σε byte-strings.

Με τις βασικές συναρτήσεις έτοιμες, ακολουθεί η βασική συνάρτηση (main). Στην αρχή ενεργοποιείται ο tf.python_io.TFRecordWriter που θα γράψει στην τοποθεσία που έχει ορίσει η παράμετρος εξόδου. Δύο βοηθητικές μεταβλητές, το path και το examples, χρησιμοποιούνται για την

κατασκευή του path προς τις εικόνες, βάσει του path που βρίσκεται εκείνη την στιγμή και την μετατροπή των δεδομένων του αρχείου csv που δείχνει η αντίστοιχη παράμετρος, σε pandas dataframe. Αφού γίνει η κλήση της split με εισόδους το examples και το filename ως το επιλεγμένο group, τα αποτελέσματά της αποθηκεύονται σε άλλη μία βοηθητική λίστα που είναι το grouped. Για κάθε group μέσα στην λίστα, καλείται η συνάρτηση create_tf_example και τα αποτελέσματά της γράφονται στο αρχείο TFRecord από το tf.python_io.TFRecordWriter, σειριοποιόντας τα σε μορφή string. Τέλος, απενεργοποιείται το writer και εμφανίζεται ένα μήνυμα στην κονσόλα αν η διαδικασία έχει πραγματοποιηθεί με επιτυχία.

Υπάρχει βέβαια και ένα τελευταίο στάδιο, που είναι η δημιουργία ενός ακόμη αρχείου που λέγεται labelmap.pbtxt, το οποίο διαφέρει λίγο σε δομή από το labelmap.txt. Ενώ το προηγούμενο είχε μόνο τις κλάσεις σε μορφή κειμένου, το καινούργιο περιέχει το όνομα και μία id για κάθε κλάση. Η id είναι ένας θετικός ακέραιος αριθμός που είναι διάφορος του μηδενός. Πρέπει δηλαδή σε κάθε κλάση να προστίθεται +1, καθώς η αρίθμηση των σειρών ξεκινάει από το 0. Η δημιουργία του συνεπώς είναι αρκετά εύκολη. Πρώτα, κατασκευάζεται ένα path για το απλό αρχείο labelmap.txt. Συνεχίζοντας, διαβάζεται το περιεχόμενο του κειμένου της κάθε σειράς, όπως και το νούμερό της. Έτσι για την κάθε κλάση, στο 'name' τοποθετείται το κείμενο, ενώ στο 'id' το νούμερο της σειράς +1. Η τοποθεσία που θα αποθηκευτεί είναι η ίδια με το αρχικό αρχείο labelmap.

Ο κώδικας αυτός είναι υποχρεωτικό να εκτελεστεί για το υποσύνολο του training και του validation. Για να γίνει εύκολα αυτό, μπορεί να αποθηκευτεί σε ένα αρχείο python ως 'generate_tfrecord.py', τοποθετώντας την γραμμή '%%writefile generate_tfrecord.py' στην αρχή του. Έτσι, στο επόμενο κελί κώδικα του Google Colab, μπορεί να εκτελεστεί δύο φορές με τις αντίστοιχες παραμέτρους των δύο υποσυνόλων.

4.4 Εισαγωγή, ρύθμιση και εκπαίδευση ενός Pre-Trained μοντέλου

Για την εισαγωγή ενός pre-trained μοντέλου από το Tensorflow 2 Detection Model Zoo, πρέπει να οριστούν τρεις μεταβλητές. Η πρώτη περιέχει το επίσημο όνομα του μοντέλου όπως εμφανίζεται στο Model Zoo, η δεύτερη περιέχει το path για το αρχικό αρχείο ρυθμίσεων του μοντέλου και η τρίτη το path του pretrained checkpoint, που είναι ένα συμπιεσμένο αρχείο (tar) με όλα τα συναπτικά βάρη για ένα ορισμένο dataset (όπως το COCO). Επειδή θα εκπαιδευτούν δύο μοντέλα σε αυτή την εργασία, το SSD-MobileNetV2 και το SSD-MobileNetV2-fpnlite 320x320, ο κώδικας του κελιού μπορεί να περιέχει ένα λεξικό, το οποίο θα κατέχει το model_name (το όνομα του μοντέλου), το path για το base_pipeline_file (το αρχείο με τις ρυθμίσεις του) και το path για το pretrained_checkpoint (τα συναπτικά βάρη του) και των δύο μοντέλων. Από εκεί μπορεί μέσω μίας βοηθητικής μεταβλητής να επιλέγεται το μοντέλο που πρόκειται να χρησιμοποιηθεί και στην συνέχεια να τοποθετούνται τα αντίστοιχα στοιχεία στις μεταβλητές model, base_pipeline_file και pretrained_checkpoint, που θα είναι πολύ χρήσιμες κατά την διαδικασία της εισαγωγής και της ρύθμισης.

Το επόμενο κελί κώδικα ευθύνεται για την εισαγωγή του επιλεγμένου μοντέλου στο περιβάλλον του Google Colab. Πρώτα όμως, δημιουργείται ένας φάκελος στον οποίο θα τοποθετηθούν να αρχεία που προαναφέρθηκαν. Αυτό γίνεται μέσω της εντολής mkdir. Ο φάκελος αυτός θα λέγεται 'mymodel' και θα βρίσκεται μέσα στον φάκελο του Tensorflow Object Detection API. Στην συνέχεια, η εντολή cd ακολουθούμενη από την διεύθυνση του δημιουργημένου φακέλου, ορίζει την διεύθυνση αυτή ως εκείνη που πρόκειται να τοποθετηθούν τα αρχεία του μοντέλου. Επειδή το ένα από τα τρία αρχεία είναι συμπιεσμένο, εισάγεται και το tarfile module για να εκτελέσει την αποσυμπίεσή του. Η εισαγωγή των αρχείων, γίνεται τοποθετώντας σε μία βοηθητική μεταβλητή, την βάση του συνδέσμου που οδηγεί στην ιστοσελίδα που βρίσκονται τα αρχεία από όλα τα μοντέλα, με επέκταση την αντίστοιχη μεταβλητή που ορίστηκε στο προηγούμενο βήμα. Η μεταβλητή αυτή με τον τελικό σύνδεσμο χρησιμοποιείται με την εντολή wget, η οποία εκκινεί την διαδικασία της εισαγωγής των αρχείων. Για την περίπτωση του pretrained_checkpoint, το tarfile module δίνει την δυνατότητα της εξαγωγής των συναπτικών βαρών στον φάκελο.

Οι πιο σημαντικές ρυθμίσεις που μπορούν να γίνουν από τον χρήστη, που καθορίζουν τον τρόπο με τον οποίο θα γίνει η εκπαίδευση, λέγονται **υπερπαράμετροι** (hyperparameters). Η πρώτη και πιο βασική υπερπαράμετρος είναι **ο αριθμός των βημάτων** εκπαίδευσης που είναι επιθυμητό να γίνουν. Η τιμή αυτή δεν είναι αυστηρή. Ο χρήστης, παρακολουθώντας την πορεία της εκπαίδευσης, μπορεί οποιαδήποτε στιγμή να σταματήσει την διαδικασία. Στην περίπτωση που δεν υπάρχει παρέμβαση, η εκπαίδευση θα ολοκληρωθεί αυτόματα, μόλις τα βήματα εξισωθούν με αυτά που ορίστηκαν αρχικά. Η τιμή που βρέθηκε ως βέλτιστη για το σύνολο δεδομένων της εργασίας αυτής, ήταν 40000. Η δεύτερη υπερπαράμετρος που είναι επίσης πολύ σημαντική, είναι το **batch size**. Η τιμή αυτή, θέτει την ποσότητα των εικόνων που θα λαμβάνονται σε κάθε επανάληψη της εκπαιδευτικής διαδικασίας. Ο αριθμός αυτός πρέπει να είναι δύναμη του 2, για να είναι βέλτιστη η χρήση της μνήμης της μονάδας επεξεργασίας. Μεγαλύτερες τιμές batch size μπορεί να οδηγήσουν σε μία πιο ομαλή εκπαιδευτική διαδικασία, ωστόσο επιβαρύνουν περισσότερο την μνήμη. Στην τελική, ο καθορισμός αυτής της τιμής, βασίζεται στην αρχιτεκτονική του μοντέλου και την διαθέσιμη μνήμη. Στην εργασία αυτή, για τις δύο αρχιτεκτονικές που χρησιμοποιήθηκαν, η τιμή αυτή τέθηκε ως 16. Οι υπόλοιπες υπερπαράμετροι είναι αυστηρές και αν δεν οριστούν σωστά, η εκπαίδευση δεν θα είναι εφικτή. Αυτές είναι:

- Ο αριθμός των κλάσεων.
- Η διεύθυνση του αρχείου tfrecord για την εκπαίδευση.
- Η διεύθυνση του αρχείου tfrecord για την επαλήθευση.
- Η διεύθυνση του αρχείου labelmap.pbtxt.
- Η διεύθυνση του pretrained_checkpoint με τα συναπτικά βάρη.

Η εφαρμογή των υπερπαραμέτρων γίνεται μέσα στο αρχείο base_pipeline. Αξίζει να σημειωθεί ότι οι παραπάνω υπερπαράμετροι είναι οι απολύτως αναγκαίες. Για κάθε αρχιτεκτονική υπάρχουν και άλλες ρυθμίσεις που μπορεί να αλλάξει ο χρήστης, για να βελτιστοποιήσει τις επιδόσεις του μοντέλου του.

Για να πραγματοποιηθούν αλλαγές στο αρχείο ρυθμίσεων μέσω κώδικα, μπορεί να αξιοποιηθεί το regular expression module, το οποίο ψάχνει για σημεία του κειμένου που ταιριάζουν με αυτά που γράφει ο χρήστης στον κώδικα. Η λογική είναι ότι θα βρίσκεται το επιθυμητό σημείο που πρέπει να αλλάξει (παλιά ρύθμιση) και θα αντικαθίσταται με το καινούργιο κείμενο (καινούργια ρύθμιση). Ξεκινώντας με την εντολή open, με είσοδο την διεύθυνση του base_pipeline που ορίστηκε προηγουμένως, για να ανοίξει το αρχείο ως f και να αποθηκεύσει τα περιεχόμενά του σε μία μεταβλητή s. Στην συνέχεια, μέσω του sub του re γίνονται οι ακόλουθες αλλαγές στην s:

1. στην γραμμή ‘fine_tune_checkpoint’: "PATH_TO_BE_CONFIGURED"‘, προστίθεται η διεύθυνση του pretrained_checkpoint με τα συναπτικά βάρη.
2. στην γραμμή ‘input_path’: “PATH TO BE CONFIGURED” (train)’, προστίθεται η διεύθυνση του αρχείου tfrecord για την εκπαίδευση.
3. στην γραμμή ‘input_path’: "PATH_TO_BE_CONFIGURED” (val)’, προστίθεται η διεύθυνση του αρχείου tfrecord για την επαλήθευση.
4. στην γραμμή ‘label_map_path’: "PATH_TO_BE_CONFIGURED”‘, προστίθεται η διεύθυνση του αρχείου labelmap.pbtxt.
5. στην γραμμή ‘batch_size’: {προεπιλεγμένη τιμή}’, προστίθεται η τιμή της μεταβλητής batch size που έχει οριστεί στο προηγούμενο κελί κώδικα.
6. στην γραμμή ‘num_steps’: {προεπιλεγμένη τιμή}‘, προστίθεται η τιμή της μεταβλητής num_steps (αριθμός βημάτων) που έχει οριστεί στο προηγούμενο κελί κώδικα.
7. στην γραμμή ‘num_classes’: {προεπιλεγμένη τιμή}‘, προστίθεται η τιμή της μεταβλητής num_classes (αριθμός κλάσεων) που έχει οριστεί στο προηγούμενο κελί κώδικα.
8. στην γραμμή ‘fine_tune_checkpoint_type’: "classification"’ αντικαθίσταται το είδος ‘classification’ με ‘detection’.

Αυτές θα γίνουν και για τις δύο αρχιτεκτονικές. Ωστόσο, για την περίπτωση του SSD-MobileNetV2, παρατηρήθηκε ότι οι τιμές του learning rate (base και warmup) ήταν πολύ διαφορετικές από τις συνήθεις τιμές. Θα αντικατασταθούν λοιπόν με τις τιμές που έχει το SSD-MobileNetV2-fpnlite 320x320 ως προεπιλογή, που είναι learning_rate_base = 0.08 (από 0.8) και warmup_learning_rate = 0.26666 (από 0.13333). Τέλος, αυξήθηκε και το μέγεθος της εικόνας εισόδου (input size) από 300x300 σε 510x510 (πρέπει να είναι πολλαπλάσιο του 30 λόγω της αρχιτεκτονικής του συνελικτικού δικτύου), ώστε να αυξηθεί η ακρίβεια του μοντέλου, θυσιάζοντας φυσικά ένα κομμάτι από την ταχύτητά του, που όπως θα αναφερθεί και στις επόμενες ενότητες, δεν είχε ιδιαίτερη σημασία. Αφού λοιπόν έχουν γίνει όλες οι αλλαγές, η εντολή f.write(s), γράφει τα

περιεχόμενα που έχουν αποθηκευτεί στην μεταβλητή s, στο αρχείο. Προαιρετικά ο χρήστης μπορεί να κάνει επισκόπηση όλων των ρυθμίσεων ανοίγοντας το αρχείο config μέσω της εντολής cat ακολουθούμενη από την διεύθυνσή του (/content/models/mymodel/pipeline_file.config).

Το μόνο που μένει πλέον είναι η εκκίνηση της διαδικασίας της εκπαίδευσης. Το Tensorflow Object Detection API διαθέτει ένα έτοιμο αρχείο python που λέγεται 'model_main_tf2.py'. Το αρχείο αυτό απαιτεί ορισμένες παραμέτρους για να λειτουργήσει, που έχουν οριστεί ήδη σε προηγούμενα βήματα. Πρώτο είναι το path για το αρχείο pipeline_config, που περιέχει τις ρυθμίσεις που πραγματοποιήθηκαν παραπάνω, ενώ μετά ακολουθούν, η διεύθυνση που επιθυμεί ο χρήστης να αποθηκεύσει το καινούργιο μοντέλο και τα checkpoints του (δεν έχει οριστεί, αλλά μπορεί για παράδειγμα να είναι ένας φάκελος που λέγεται training) και ο αριθμός των βημάτων (num_steps). Τέλος, τοποθετούνται και οι παράμετροι alsologtostderr, ώστε να εμφανίζονται μηνύματα σχετικά με την πορεία της εκπαίδευσης και sample_1_of_n_eval_examples=1, ώστε κάθε παράδειγμα αξιολόγησης, να χρησιμοποιείται στην φάση της αξιολόγησης. Υψηλότερες τιμές μπορεί να μειώσουν τον χρόνο που απαιτείται για την αξιολόγηση, εις βάρος της ακρίβειας. Μόλις γίνει η εκτέλεση του αρχείου, θα εκκινήσει και η εκπαίδευση η οποία μπορεί να κρατήσει αρκετές ώρες.

Κατά την διάρκειά της, το πρόγραμμα επιστρέφει ανά 100 βήματα, πληροφορίες σχετικά με το συνολικό αριθμό βημάτων που έχουν εκτελεστεί, ο χρόνος που χρειάστηκε ανά βήμα, την τιμή του learning rate, τις απώλειες εντοπισμού (localization loss), τις απώλειες κανονικοποίησης (regularization loss) και τις συνολικές απώλειες (total loss). Όσο το μοντέλο εκπαιδεύεται η συνολικές απώλειες μειώνονται, ωστόσο αν γίνει η χάραξη των τιμών αυτών συναρτήσει των βημάτων, είναι φανερό ότι η μείωση έχει εκθετικό χαρακτήρα και περιέχει πολλές ταλαντώσεις. Η τιμή των συνολικών απωλειών σταθεροποιείται σε ένα παράθυρο τιμών, που είναι και η ένδειξη ότι η εκπαίδευση έχει πρακτικά ολοκληρωθεί. Το learning rate ξεκινάει από μία αρχική τιμή (warmup), αυξάνεται μέχρι την μέγιστη τιμή που έχει δοθεί (base) και μετά σιγά σιγά μειώνεται. Όλα αυτά γίνονται ακόμα πιο εμφανή μέσω του Tensorboard. Το Tensorboard session μπορεί να ξεκινήσει σε ένα κελί κώδικα που προηγείται αυτό της εκπαίδευσης. Εκεί, εμφανίζονται όλα τα απαραίτητα γραφήματα με τα δεδομένα που προειπώθηκαν για την διευκόλυνση του χρήστη (πρέπει να γίνεται ανανέωση μέσω του ειδικού κουμπιού που διαθέτει).

4.5 Μετατροπή του μοντέλου σε μοντέλο TFlite

Για να γίνει η μετατροπή του μοντέλου, σε tflite μοντέλο, καλό είναι να περάσει από δύο ξεχωριστά στάδια. Στο πρώτο πρέπει να εκτελεστεί το αρχείο 'export_tflite_graph_tf2.py', το οποίο δημιουργεί ένα ενδιάμεσο μοντέλο για να διευκολύνει την τελική μετατροπή. Το script χρειάζεται τρεις παραμέτρους. Η πρώτη είναι η διεύθυνση που βρίσκεται το τελευταίο checkpoint του μοντέλου που μόλις εκπαιδεύτηκε. Η δεύτερη είναι η διεύθυνση του αρχείου pipeline config. Η τελευταία είναι η διεύθυνση που είναι επιθυμητό να αποθηκευτεί η έξοδος που θα παράγει. Εκεί καλό είναι να

δημιουργηθεί ένας νέος φάκελος για το tflite μοντέλο. Αφού ολοκληρωθεί η εκτέλεση του προγράμματος, το ενδιάμεσο μοντέλο τροφοδοτείται στο converter που παρέχει το API, το 'tf.lite.TFLiteConverter.from_saved_model(ενδιάμεσο μοντέλο)', όπου τελειώνει η πλήρης μετατροπή σε tflite. Τα περιεχόμενα της μετατροπής τοποθετούνται σε μία βοηθητική μεταβλητή tflite_model. Για την αποθήκευσή του tflite μοντέλου στο περιβάλλον του Google Colab, αρκεί απλά να δημιουργηθεί ένα προσωρινό αρχείο detect.tflite, όπου θα εγγραφούν τα περιεχόμενα της μετατροπής σε αυτό, μέσω του 'with open(διεύθυνση του αρχείου)' ως f και στην συνέχεια 'f.write(βοηθητική μεταβλητή που περιέχει το tflite μοντέλο)'.

4.6 Δοκιμή του μοντέλου στις εικόνες του υποσυνόλου 'testing'

Δεν είναι κακή ιδέα να γίνει μία πρώτη εκτίμηση της ικανότητας εντοπισμού και της ακρίβειάς του νέου μοντέλου μέσω της εκτέλεσής του στο Google Colab, σε φωτογραφίες που βρίσκονται στο υποσύνολο testing. Για να γίνει αυτό χρειάζεται ένας κώδικας που θα εκτελεί **inference**, δηλαδή, θα χρησιμοποιεί το μοντέλο για να παράξει απεικονισμένα αποτελέσματα/συμπεράσματα (προβλέψεις, bounding boxes). Η συγγραφή ενός τέτοιου προγράμματος δεν είναι εύκολη και τα αποτελέσματα που παράγει δεν είναι πάντα έγκυρα, καθώς σε πραγματικές εφαρμογές θα γίνονται 10 με 45 inferences το δευτερόλεπτο (για τα μοντέλα που δημιουργήθηκαν). Αυτό σημαίνει ότι για ένα πολύ μικρό χρονικό διάστημα μπορεί να γίνει κάποιος λάθος εντοπισμός (ή και να μην γίνει καν), αλλά μετά να διορθωθεί (και το αντίστροφο φυσικά). Αυτό δεν είναι εμφανές όταν πραγματοποιείται inference μία φορά ανά εικόνα. Μία λύση είναι να γίνουν 2-4 inferences ανά εικόνα, αλλά αυτό θα είναι ακόμα πιο χρονοβόρο (ειδικά όταν οι εικόνες είναι περίπου 100) και χωρίς να υπάρχει εγγύηση για βελτίωση των αποτελεσμάτων, καθώς μερικές φορές την αλλαγή στις προβλέψεις την προκαλούν άλλοι παράγοντες, όπως η γωνία ή η τοποθεσία του αντικειμένου στην εικόνα, κάτι που δεν αλλάζει όσες φορές και αν γίνει η επανάληψη. Ένας ακόμη λόγος είναι ότι επειδή ο διαχωρισμός εικόνων σε υποσύνολα γίνεται τυχαία, το inference γίνεται σε διαφορετικές εικόνες για κάθε διαφορετικό μοντέλο που εκπαιδεύεται, οπότε η απευθείας σύγκρισή τους είναι λάθος. Άρα οι τελικές δοκιμές, από τις οποίες θα βγει και το πόρισμα της εργασίας, θα γίνουν πάνω στο Google Coral Dev Board με την χρήση κάμερας σε ένα καινούργιο σύνολο εικόνων.

Ο κώδικας για την εκτέλεση inference στο υποσύνολο testing έχει συγγραφεί από τον Evan Juras (EdjeElectronics στο GitHub) [36], με μερικές μικρές αλλαγές στον τρόπο που γίνεται η προβολή των αποτελεσμάτων, ώστε να είναι πιο ορατά. Αρχικά εισάγονται τα ακόλουθα modules: το tensorflow ως tf, το os, το cv2, το numpy ως np, το sys, το glob, το random, το importlib.util, το Interpreter από το tensorflow.lite.python.interpreter και το matplotlib και το matplotlib.pyplot ως plt. Στην συνέχεια δημιουργείται η βασική συνάρτηση tflite_detect_images που θα εκτελεί και το inference. Η συνάρτηση αυτή λαμβάνει σαν εισόδους: την διεύθυνση του μοντέλου tflite, την διεύθυνση των εικόνων για τις δοκιμές, την διεύθυνση του labelmap, το ελάχιστο όριο σιγουριάς

(minimum confidence threshold), που ορίζει πότε πρέπει να εμφανίζεται μία πρόβλεψη με βάση το ποσοστό σιγουριάς της (0%-100%), τον αριθμό εικόνων που θέλει ο χρήστης να χρησιμοποιήσει στις δοκιμές, την διεύθυνση της αποθήκευσης των αποτελεσμάτων και ένα flag που ορίζει αν χρειάζεται να εμφανίσει τις προβλέψεις ή να τις αποθηκεύσει απλά υπό την μορφή κειμένου. Μέσω του glob βρίσκονται όλες οι εικόνες από τον φάκελο του υποσυνόλου και στην συνέχεια φορτώνεται το περιεχόμενο του labelmap στην μνήμη (σε μία λίστα). Μέσω του Interpreter module, εισάγεται και το tflite μοντέλο στην μνήμη, όπου μετά είναι εύκολη η εξαγωγή βασικών πληροφοριών για την είσοδο και την έξοδό του (π.χ. τις διαστάσεις της εικόνας που δέχεται σαν είσοδο). Συνεχίζοντας, επιλέγονται οι εικόνες. Αν ο χρήστης έχει δώσει ένα νούμερο μικρότερο από το ποσό των εικόνων που βρίσκονται στο υποσύνολο (έστω n), τότε επιλέγονται τυχαία (n εικόνες) και για την καθεμία, πραγματοποιείται φόρτωση και μετατροπή των διαστάσεών της ώστε να είναι συμβατή με την είσοδο του μοντέλου. Στην περίπτωση που τα δεδομένα είναι τύπου float32 (float 32 bit), γίνεται και κανονικοποίηση των τιμών των pixel της. Το interpreter module παρέχει εντολές όπως το set_tensor, μέσω του οποίου τίθονται τα δεδομένα εισόδου στον τανυστή εισόδου και το invoke που εκτελεί inference με το μοντέλο. Μετά, με το get_tensor μπορεί να γίνει η εξαγωγή των αποτελεσμάτων σε μεταβλητές που θα περιέχουν τις συντεταγμένες των bounding boxes, τις εντοπισμένες κλάσεις, καθώς και τα scores, δηλαδή το ποσοστό σιγουριάς της κάθε πρόβλεψης. Δημιουργείται μία άδεια λίστα για τα δεδομένα του κάθε εντοπισμού. Σε έναν βρόχο επανάληψης for, ελέγχεται αν η κάθε πρόβλεψη έχει ποσοστό σιγουριάς μεγαλύτερο από το ορισμένο κατώτατο όριο και σε περίπτωση θετικής απάντησης, αποθηκεύονται τα δεδομένα των κλάσεων και των αντίστοιχων bounding boxes και μετά σχεδιάζονται πάνω στην εικόνα τους. Το module cv2 διαθέτει μία πληθώρα από ρυθμίσεις για τον τρόπο που θα εμφανίζονται τα αποτελέσματα, όπως το μέγεθος και το είδος της γραμματοσειράς που θα δείχνει την κλάση, καθώς και το πάχος και το χρώμα του bounding box. Ο κάθε χρήστης λοιπόν μπορεί να κάνει οποιαδήποτε αλλαγή του φαίνεται πιο οπτικά ευχάριστη. Το τελευταίο κομμάτι της συνάρτησης ελέγχει αν το flag για την αποκλειστική αποθήκευση των αποτελεσμάτων των εντοπισμών υπό την μορφή κειμένου είναι ενεργό (True) και αν δεν είναι, τότε παρουσιάζει τις εικόνες και τους εντοπισμούς.

Αν μετά τις δοκιμές, το μοντέλο κριθεί άξιο (εμπειρικά) για την περαιτέρω δοκιμή σε πραγματικό χρόνο, τότε μπορεί να αποθηκευτεί στον υπολογιστή του χρήστη, αφού ο φάκελος που βρίσκεται, συμπιεστεί σε αρχείο zip.

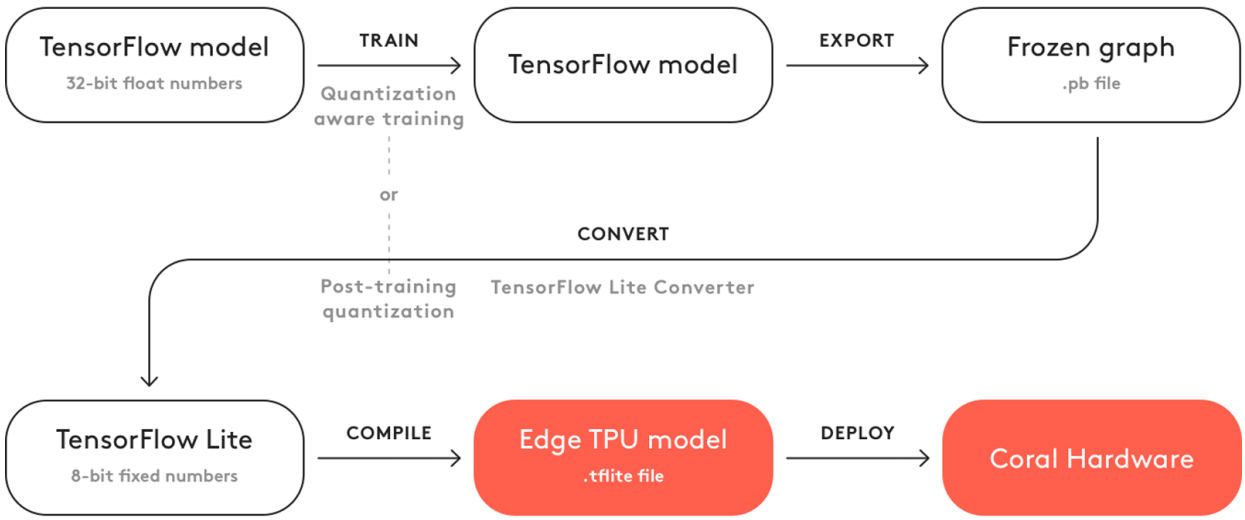
4.7 Quantization του μοντέλου και εφαρμογή του edge compiler

Το quantization είναι μία μέθοδος, κατά την οποία τα συναπτικά βάρη μετατρέπονται από τιμές τύπου float 32 bit σε ακέραιους (integer) 8 bit, αυξάνοντας αρκετά την ταχύτητα εκτέλεσης του detector. Το βήμα αυτό είναι απαραίτητο για οποιοδήποτε μοντέλο πρόκειται να εκτελεστεί σε ένα EdgeTPU. Το Tensorflow έχει όλα τα εργαλεία για την διαδικασία αυτή, ωστόσο για να

αξιοποιηθούν, πρέπει πρώτα να δημιουργηθεί ένα καινούργιο αντιπροσωπευτικό σύνολο δεδομένων που μπορεί πολύ απλά να είναι το ίδιο με το υποσύνολο των εικόνων για την εκπαίδευση. Ο κώδικας είναι γραμμένος από την Google με ορισμένες μικρές αλλαγές για να μπορεί να δουλέψει στο συγκεκριμένο Colab Notebook. Αρχικά, ενεργοποιεί τον interpreter, δεσμεύει μνήμη για τους τανυστές της εισόδου και εξόδου και λαμβάνει τις πληροφορίες σχετικά με τις διαστάσεις τους. Την δημιουργία του αντιπροσωπευτικού συνόλου αναλαμβάνει η συνάρτηση representative_data_gen που δεν απαιτεί κάποια είσοδο από τον χρήστη. Πρώτα, μαζεύονται σε μία λίστα όλα τα αρχεία των εικόνων που έχουν δοθεί (υποσύνολο training). Τριακόσια από αυτά επιλέγονται τυχαία, παραμορφώνονται με τέτοιο τρόπο ώστε να έχουν τις κατάλληλες διαστάσεις για να είναι συμβατά με την είσοδο του νευρωνικού δικτύου και οι τιμές των pixel τους κανονικοποιούνται (οι τιμές λαμβάνουν τιμές από το 0 έως το 1). Τέλος, προσθέτει μία ακόμη διάσταση batch (batch dimension) και επιστρέφει τα αποτελέσματα σε μία λίστα.

Για την διαδικασία του quantization, αρχικά ενεργοποιείται το TFLiteConverter με το tflite μοντέλο. Μετά εφαρμόζονται οι προεπιλεγμένες βελτιστοποιήσεις από το εργαλείο optimizations και ορίζεται το σύνολο που δημιουργήθηκε προηγουμένως ως το αντιπροσωπευτικό σύνολο μέσω του representative_dataset. Αφού τεθεί η είσοδος ως ακέραιος 8 bit (8 bit integer – int8) και η έξοδος ως float 32 bit (float32), εκτελείται το converter και κάνει το quantization. Τα καινούργια δεδομένα αποθηκεύονται με την μορφή ενός νέου μοντέλου που στην βάση του ονόματος του tflite μοντέλου, προστίθεται το ‘quant’ για να δείξει ότι είναι quantized (π.χ. detect_quant.tflite).

Με το μοντέλο κβαντισμένο (quantized), μένει απλά να χρησιμοποιηθούν τα εργαλεία που παρέχει η Google, για να μεταφερθούν όσο το δυνατόν περισσότερες λειτουργίες και πράξεις στην EdgeTPU. Πρώτα, απαιτείται η εγκατάσταση των πακέτων του EdgeTPU, καθώς και του ίδιου του compiler. Έπειτα, εκτελείται ο compiler στο κβαντισμένο μοντέλο. Τέλος, το καινούργιο μοντέλο αποθηκεύεται ξεχωριστά στον ίδιο φάκελο με τα υπόλοιπα, με την προσθήκη του edgetpu στην βάση του ονόματός του (detect_quant_edgetpu.tflite).



Εικόνα 4.4: Η διαδικασία μετατροπής ενός μοντέλου Tensorflow σε μοντέλο που είναι συμβατό με το Google Coral [37]

Στο τέλος του Google Colab Notebook, θα πρέπει να είναι έτοιμος ένας φάκελος με το βασικό tflite μοντέλο, το κβαντισμένο tflite μοντέλο και το συμβατό με την EdgeTPU tflite μοντέλο. Το μόνο που μένει είναι να συμπιεστεί ο φάκελος και να αποθηκευτεί στον υπολογιστή για μελλοντική χρήση. Όλη αυτή η διαδικασία έγινε αρκετές φορές κατά την διάρκεια της πειραματικής διαδικασίας της διπλωματικής εργασίας, για να προκύψουν δύο τελικά μοντέλα με τις αρχιτεκτονικές SSD-MobileNetV2 και SSD-MobileNetV2-fpnlite 320x320 (υπάρχει και με μεγαλύτερες διαστάσεις εισόδου, αλλά δεν προτείνεται για εφαρμογές που απαιτούν μεγάλη ταχύτητα), που εκπαιδεύτηκαν με το ίδιο σύνολο δεδομένων, με το ίδιο batch size και learning rate και για τον ίδιο αριθμό βημάτων. Αυτά τα δύο μοντέλα θα είναι και αυτά που θα δοκιμαστούν στο Google Coral Dev Board. Οι αποτυχημένες προσπάθειες έδειξαν ότι το βασικό κριτήριο για την επίδοση ενός μοντέλου ήταν η ποσότητα και η ποιότητα των δεδομένων.

5 ΚΕΦΑΛΑΙΟ 5ο : Μεταφορά και εφαρμογή του μοντέλου στο Google Coral Dev Board

Ο σκοπός του κεφαλαίου αυτού είναι να γίνει η αναλυτική περιγραφή της διαδικασίας, με την οποία μπορεί να πραγματοποιηθεί η μεταφορά και η εφαρμογή των μοντέλων από τον κεντρικό υπολογιστή στο Google Coral Dev Board. Αρχικά, θα γίνει το στήσιμο του dev board, με την εγκατάσταση των απαραίτητων πακέτων για την ορθή λειτουργία του και την εύκολη και αξιόπιστη επικοινωνία με τον κεντρικό υπολογιστή. Στην συνέχεια, θα γίνει η μεταφορά των αρχείων των μοντέλων και στο τέλος, η εφαρμογή τους σε εικόνες και σε πραγματικό χρόνο μέσω κάμερας.

5.1 Στήσιμο του Google Coral Dev Board

Για να μπορέσει να εκτελεστεί κάποιο μοντέλο μηχανικής μάθησης στο Google Coral, πρέπει πρώτα να γίνει η εγκατάσταση του λειτουργικού Mendel Linux, όπως και του Pycoral API. Ακολουθώντας τα βήματα που παρέχει η Google στον οδηγό της [20], πρώτα επιβεβαιώνεται ότι ο υπολογιστής πληροί όλες τις προϋποθέσεις και ο χρήστης κατέχει τον απαραίτητο εξοπλισμό. Συγκεκριμένα, ο υπολογιστής πρέπει:

- να έχει λειτουργικό σύστημα Linux (προτείνεται), Mac ή Windows 10 και πάνω
- εγκατεστημένη Python3

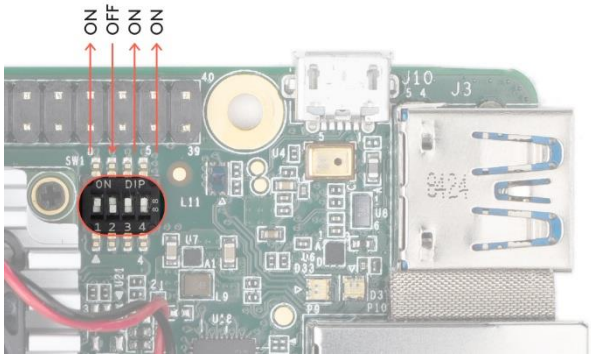
Από την πλευρά του ο χρήστης πρέπει να έχει [20]:

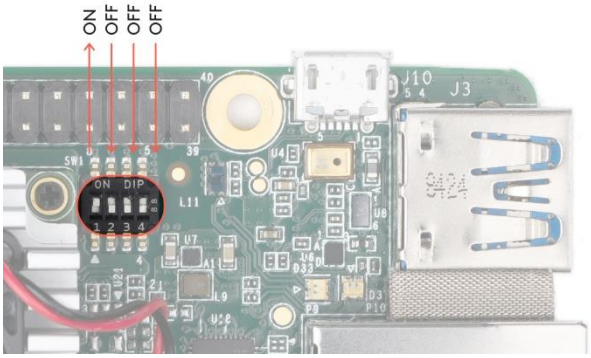
- μία κάρτα microSD με τουλάχιστον 8 GB (Gigabyte – 109 bytes) αποθηκευτικό χώρο
- μία εξωτερική τροφοδοσία 5 V με καλώδιο USB-C και 2-3 A μέγιστη ένταση ρεύματος
- ένα καλώδιο USB-C/Micro-USB σε USB-A για την σύνδεση και την επικοινωνία με τον υπολογιστή
- διαθέσιμη σύνδεση με το Internet (Wifi/Ethernet)

Επιπλέον, η Google προτείνει την εγκατάσταση του Git for Windows και την χρήση της κονσόλας Git Bash, που υποστηρίζει όλες τις εντολές που θα εκτελεστούν. Αυτό δεν χρειάζεται αν η επικοινωνία με τον κεντρικό υπολογιστή γίνεται μέσω σειριακής κονσόλας [20].

Για να κάνει ο χρήστης flash το dev board με το Mendel Linux, αρχικά πρέπει να κατεβάσει το zip αρχείο που διαθέτει μέσω συνδέσμου η Google. Το zip πρέπει να περιέχει ένα αρχείο με όνομα ‘flashcard_arm64.img’. Μετά, πρέπει να χρησιμοποιήσει ένα πρόγραμμα όπως το balenaEtcher για να κάνει flash το αρχείο ‘.img’ στην microSD. Με το board εκτός τροφοδοσίας, πρέπει να τεθούν μερικοί μικροί διακόπτες που βρίσκονται δίπλα από την TPU, σε ορισμένες θέσεις. Οι θέσεις αυτές είναι (από τα αριστερά στα δεξιά) switch1(on), switch2(off), switch3(on) και switch4(on) και σηματοδοτούν ότι η εκκίνηση θα γίνει με την κάρτα microSD. Μόλις τελειώσει η διαδικασία, το board θα κλείσει αυτόματα. Μετά πρέπει να ρυθμιστούν πάλι οι προηγούμενοι διακόπτες στις θέσεις

switch1(on), switch2(off), switch3(off) και switch4(off), ώστε να εκκινεί το λειτουργικό που βρίσκεται πλέον στην μνήμη eMMC (embedded MMC) [20].





Εικόνα 5.1 και Εικόνα 5.2: Ρύθμιση διακόπτων για λειτουργία SD (αριστερά) και eMMC (δεξιά) [20]

Το τελευταίο βήμα είναι η σύνδεση του board με το internet, για να εγκατασταθεί τοπικά το Pycoral API, μέσω των εντολών που παρέχει ο οδηγός της Google [20]. Για να επικοινωνήσει ο κεντρικός υπολογιστής με το board, μπορεί να συνδεθεί με καλώδιο USB-C και να εκτελέσει εντολές μέσω του MDT shell (της κονσόλας του Mendel Development Tool), είτε μέσω σειριακής κονσόλας, από μία εφαρμογή όπως το PuTTY [20]. Μόλις γίνουν και αυτά, το board είναι και επισήμως έτοιμο.

5.2 Μεταφορά των μοντέλων

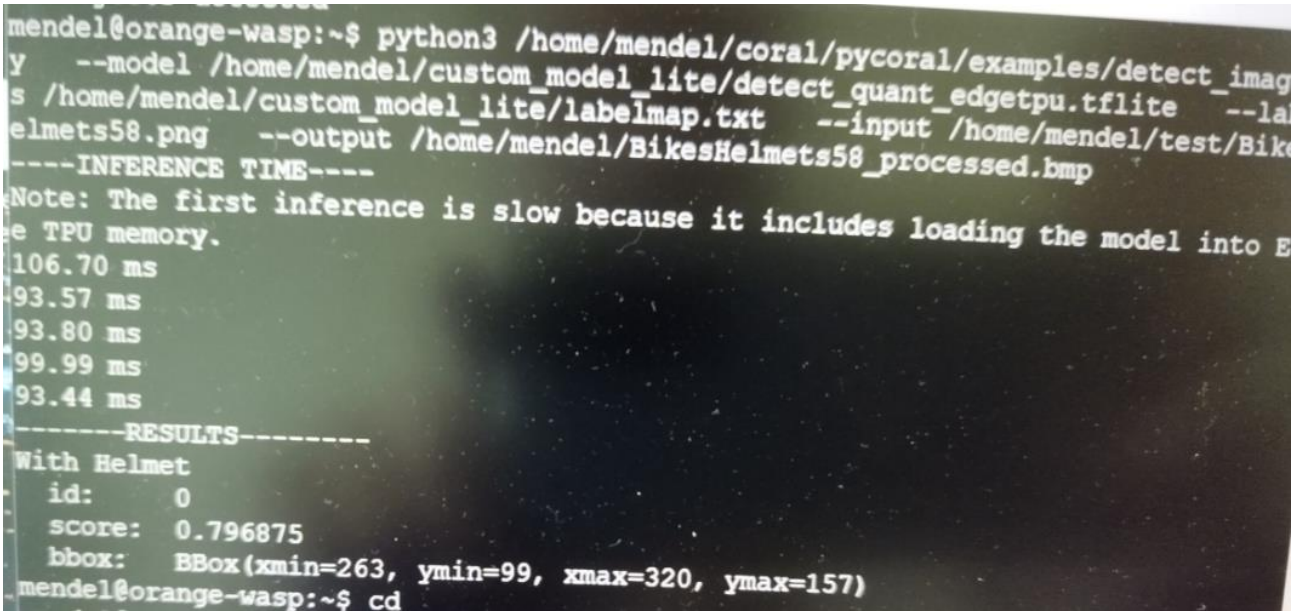
Η μεταφορά αρχείων από τον κεντρικό υπολογιστή στο Google Coral γίνεται με δύο βασικούς τρόπους. Ο πρώτος είναι με τις εντολές push και pull, μέσω του MDT shell (της κονσόλας του Mendel Development Tool), όπου προωθούνται τα αρχεία απευθείας, εύκολα και γρήγορα. Ο δεύτερος τρόπος είναι μέσω κάρτας microSD. Η επιλογή αυτή είναι αρκετά πιο χρονοβόρα, αλλά έχει το πλεονέκτημα ότι τα αρχεία μπορούν να βρίσκονται μέσα στην κάρτα για όσο χρόνο επιθυμεί ο χρήστης. Στην περίπτωση της διπλωματικής εργασίας, δυστυχώς είχε προκύψει ένα πρόβλημα με την επικοινωνία μέσω καλωδίου USB-C και κατά συνέπεια, η χρήση της κονσόλας MDT ήταν αδύνατη. Άρα η μόνη επιλογή ήταν πρακτικά η δεύτερη, οπότε η επικοινωνία με το board θα γίνεται αποκλειστικά μέσω σειριακής σύνδεσης από την εφαρμογή PuTTY. Μόλις γίνει format του αποθηκευτικού χώρου σε μορφή NTFS (καθώς πριν ήταν flashed με την εικόνα του Mendel Linux), γίνεται και η αντιγραφή των φακέλων των μοντέλων σε συμπιεσμένη μορφή (zip). Να σημειωθεί ότι αν ο υπολογιστής δεν έχει θύρα για microSD, μπορεί να χρησιμοποιηθεί αντάπτορας microSD σε SD, αν υπάρχει θύρα για κάρτα SD και microSD σε USB-A, αν ούτε αυτή είναι διαθέσιμη.

Με όλα τα αρχεία που είναι επιθυμητό να μεταφερθούν στην κάρτα microSD, αφαιρείται από τον κεντρικό υπολογιστή και τοποθετείται στην ειδική θύρα που έχει το Google Coral. Μόλις εκκινήσει το board, πρέπει να γίνει μία μικρή διαδικασία για την σωστή μεταφορά και οργάνωση των αρχείων αυτών. Πρώτα πρέπει να βρεθεί το όνομα του αποθηκευτικού χώρου (microSD) μέσω της εντολής

'sudo fdisk –l'. Το όνομα αυτό πρέπει να χρησιμοποιηθεί στην εντολή 'sudo mount (όνομα microSD) /mnt'. Με αυτό στην ουσία τίθεται η διεύθυνση /mnt ως η διεύθυνση της κάρτας. Προτείνεται σε όλες τις εντολές να προηγείται το sudo (superuser do) για να δίνεται πρόσβαση σε εντολές διαχειριστή. Ειδικά όταν πρόκειται για την επεξεργασία αρχείων μέσα στο board, είναι σχεδόν πάντα υποχρεωτικό, ώστε να μηδενιστεί η πιθανότητα να μην ολοκληρωθεί κάποια διαδικασία λόγω έλλειψης άδειας. Δημιουργείται ένας φάκελος για τα μοντέλα που θα τοποθετηθούν. Η εντολή για την δημιουργία του φακέλου είναι το mkdir ακολουθούμενο από την παρούσα διεύθυνση (αν ο χρήστης βρίσκεται ήδη στην διεύθυνση που θέλει να προσθέσει τον φάκελο, τότε δεν χρειάζεται να προσθέσει κάτι) και το όνομα του φακέλου. Έστω λοιπόν, ότι ο φάκελος αυτός έχει όνομα 'models'. Μπορεί τώρα μέσω της εντολής cp (copy) μαζί με την αρχική και την τελική διεύθυνση του αρχείου, να γίνει η αντιγραφή των μοντέλων στον φάκελο. Επειδή όμως είναι συμπιεσμένα, αρκεί απλά να γίνει χρήση της εντολής unzip μαζί με την διεύθυνση του αρχείου zip, για να ολοκληρωθεί η διαδικασία. Τέλος, προαιρετικά, μπορούν να διαγραφούν τα αρχεία zip μέσω της εντολής 'rm -rf' και στην συνέχεια την διεύθυνση του κάθε αρχείου.

5.3 Εφαρμογή των μοντέλων

Για την εκτέλεση inference πάνω στο Google Coral Dev Board, υπάρχουν δύο εύκολοι βασικοί τρόποι, αλλά απαιτούν κβαντισμένα μοντέλα και την ύπαρξη EdgeTPU. Αν δεν υπάρχουν αυτά τα δύο, τότε ο χρήστης μπορεί να προγραμματίσει το δικό του αρχείο για inference μέσω του Pycoral API. Ο πρώτος είναι και ο πιο απλός. Χρησιμοποιεί ένα από τα έτοιμα αρχεία python, ανάλογα με τον τύπο εντοπισμού που πρέπει να γίνει (π.χ classification ή object detection). Για το object detection, χρησιμοποιείται το detect_image.py (το classify_image.py για κατηγοριοποίηση εικόνας) και χρειάζονται τέσσερις παράμετροι, οι οποίες είναι, η διεύθυνση του μοντέλου tflite, η διεύθυνση του labelmap, η διεύθυνση της εικόνας που είναι επιθυμητό να εκτελεστεί το detection και τέλος η διεύθυνση στην οποία θα αποθηκευτεί η επεξεργασμένη εικόνα (εικόνα με bounding box, confidence score και κλάση). Αυτό το αρχείο στην συνέχεια εκτελεί inference με την EdgeTPU (αν υποστηρίζεται και είναι διαθέσιμη) τέσσερις φορές και εμφανίζει τα αποτελέσματα στην κονσόλα, μαζί με τους χρόνους που χρειάστηκε η κάθε εκτέλεση. Αυτή η μέθοδος είναι ένας εύκολος τρόπος για να γίνουν μερικές δοκιμές οι οποίες καταγράφονται με λεπτομέρεια και αποθηκεύονται τοπικά στην συσκευή (η εικόνα ως αρχείο bmp – bitmap), ωστόσο δεν είναι δυνατό να γίνει σε εφαρμογές πραγματικού χρόνου. Το κενό αυτό έρχεται να καλύψει ο δεύτερος τρόπος, που είναι η εκτέλεση inference στην TPU με κάμερα [20].



Εικόνα 5.3: Η κονσόλα μετά από inference σε εικόνα

Εικόνα 5.4: Η εικόνα με τα bounding boxes που εξάγεται

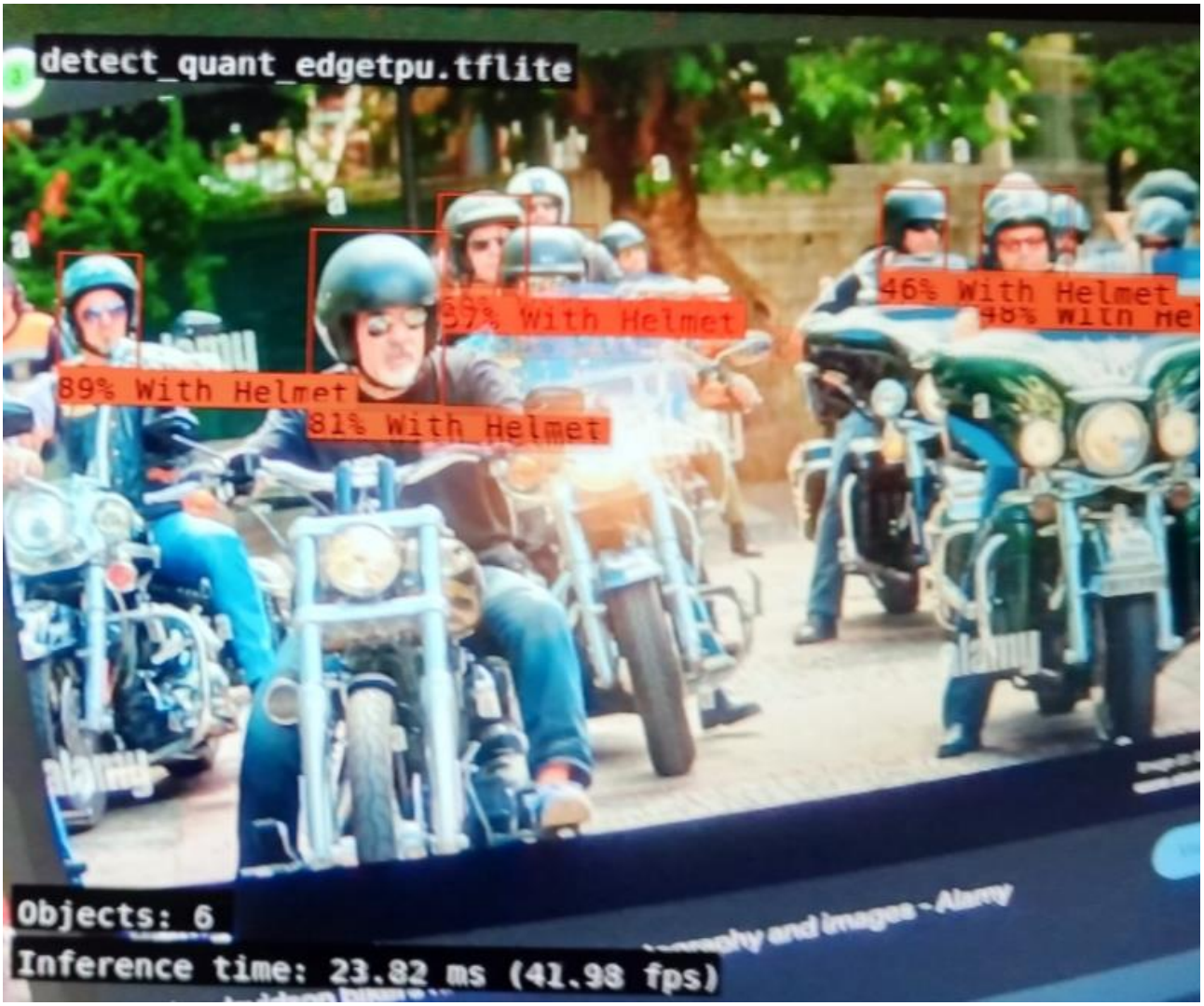
Για την διπλωματική εργασία, χρησιμοποιήθηκε μία USB camera η οποία συνδέθηκε στην διαθέσιμη θύρα USB-A του Google Coral. Έπειτα, μέσω της εντολής edgetpu_detect μαζί με μερικές παραμέτρους μπορεί να εκτελεστεί inference σε πραγματικό χρόνο. Η Google δίνει την δυνατότητα εμφάνισης της εξόδου, είτε σε οθόνη που είναι συνδεδεμένη με το board μέσω καλωδίου HDMI (η ανάλυση της οθόνης πρέπει να είναι 1920x1080, αλλιώς πρέπει να γίνουν αλλαγές στο αρχείο 'weston.ini', που βρίσκεται μέσα στο board), είτε σε έναν τοπικό server, που είναι προσβάσιμος από όλες τις συσκευές που είναι συνδεδεμένες στο ίδιο δίκτυο με αυτό. Επειδή υπάρχει διαθέσιμη οθόνη και είναι καλή ιδέα να φαίνεται η έξοδος του μοντέλου χωρίς να χρειάζεται σύνδεση στο internet, επιλέγεται η πρώτη μέθοδος. Οι παράμετροι που χρειάζονται είναι το source, δηλαδή η πηγή που θα εισάγονται οι εικόνες (μπορεί να είναι και αρχείο με βίντεο, ωστόσο επιβαρύνεται πολύ το board), η διεύθυνση του μοντέλου που θα τρέχει (οπωσδήποτε κβαντισμένο) και η διεύθυνση του labelmap. Επιπλέον, μπορεί να ρυθμιστεί και το κατώτατο όριο σιγουριάς (minimum confidence threshold) για να αφαιρούνται οι προβλέψεις χαμηλής σιγουριάς. Η τιμή που θα δοθεί πρέπει να είναι μία από το 0 έως το 1 [38].

Η παράμετρος της πηγής είναι άκρως σημαντικό να έχει τις σωστές ρυθμίσεις. Μόλις συνδεθεί η κάμερα και εκκινήσει το board, πρέπει ο χρήστης να βεβαιωθεί ότι έχει αναγνωριστεί σωστά. Με την εντολή 'v4l2-ctl --list-devices', εμφανίζονται όλες οι συσκευές εγγραφής εικόνας που εντοπίζονται από το board. Στην λίστα που εξάγεται, πρέπει να εμφανίζεται η κάμερα ως '/dev/video1'. Μια γρήγορη δοκιμή μπορεί να γίνει μέσω της εντολής snapshot, ώστε να εμφανιστεί η ζωντανή μετάδοσή της στην οθόνη. Οι κάμερες πολλές φορές διαθέτουν διάφορες επιλογές ως προς την ανάλυση και την ταχύτητα ανανέωσής τους. Μία λίστα με όλες αυτές τις επιλογές μπορεί να εκτυπωθεί μέσω της εντολής 'v4l2-ctl --list-formats-ext --device (όνομα συσκευής)'. Ποιο format όμως είναι το καλύτερο; Αυτό που είναι πιο κοντά στις διαστάσεις εισόδου του μοντέλου που θα εκτελείται και με ρυθμό ανανέωσης που δεν περιορίζει την ταχύτητά του. Για παράδειγμα, αν ένα object detector πετυχαίνει 20 inferences το δευτερόλεπτο, η επιλογή ρυθμού ανανέωσης 10 fps (frames per second - καρέ ανά δευτερόλεπτο) δεν είναι και η καλύτερη, καθώς το μοντέλο χρειάζεται μόλις 50 ms για να κάνει τον εντοπισμό, ενώ η κάμερα παρέχει μία καινούργια εικόνα κάθε 100 ms, άρα στο 50% του χρόνου λειτουργίας, δεν γίνεται κάποιος εντοπισμός. Γενικά, είναι καλύτερο να δουλεύει η κάμερα σε λίγο μεγαλύτερους ρυθμούς, αν δεν είναι δυνατό να δουλεύουν συγχρονισμένα, που είναι και το ιδανικό σενάριο. Εφόσον οι είσοδοι των μοντέλων που δημιουργήθηκαν είναι 510x510 (SSD-MobileNetV2) και 320x320 (SSD-MobileNetV2-fpnlite), πρέπει και η ανάλυση της κάμερας να είναι όσο το δυνατόν πιο κοντά σε αυτές τις διαστάσεις, ώστε να ελαχιστοποιούνται οι παραμορφώσεις κατά την μετατροπή τους από το image resizer. Η κάθε κάμερα μπορεί να εμφανίσει διάφορα format, όπως YUYV/YUY2 και MJPEG (Motion JPEG – compressed), όμως το μόνο που υποστηρίζεται από το board είναι το YUYV/YUY2 (στην εντολή μπορεί να δίνεται και με τα δύο ονόματα, καθώς είναι το ίδιο πράγμα) [38]. Η κάμερα που

χρησιμοποιήθηκε για την διπλωματική εργασία, μπορούσε να παρέχει σε αυτό το format εικόνα από 5 fps έως και 30 fps, σε διάφορες αναλύσεις που φαίνονται στην παρακάτω λίστα με λεπτομέρεια:

- 640x480 – 30 fps
- 1920x1080 – 5 fps
- 1280x720 – 10 fps
- 1280x720 – 5 fps
- 800x600 – 20 fps
- 800x600 – 15 fps
- 800x600 – 10 fps
- 800x600 – 5 fps
- 960x540 – 20 fps
- 960x540 – 15 fps
- 960x540 – 10 fps
- 960x540 – 5 fps

Η επιλογή της ανάλυσης και του ρυθμού ανανέωσης απλοποιείται αρκετά αφαιρώντας όλες αυτές με τα 5 fps, καθώς είναι εντελώς ακατάλληλες για τις εφαρμογές σε πραγματικό χρόνο, με αντικείμενα που μπορεί να κινούνται με ταχύτητα μεγαλύτερη από 30 χιλιόμετρα την ώρα. Τα 10 fps επίσης είναι στο όριο, ωστόσο επειδή υπάρχουν και επιλογές με μεγαλύτερο framerate στην ίδια ανάλυση (εκτός από τα 1280x720), θα αποκλειστούν και αυτά. Η ανάλυση 960x540 έχει το θετικό ότι ο αριθμός των pixel στον οριζόντιο άξονα είναι πολλαπλάσιο των pixel εισόδου του SSD-MobileNetV2-fpnlite (3x320 = 960) και έχει και επιλογές που παράγουν εικόνες σε χρόνους που πλησιάζουν τον χρόνο του inference του. Το μόνο αρνητικό είναι ότι ο αριθμός των pixel στον κάθετο άξονα είναι πολύ μικρότερος από αυτός του οριζόντιου και κατά συνέπεια οι εικόνες στην είσοδο θα έχουν μεγαλύτερη παραμόρφωση, άρα ούτε αυτές οι επιλογές είναι οι βέλτιστες. Οι αναλύσεις 800x600 και 640x480, έχουν την ίδια αναλογία διαστάσεων (aspect ratio = 4:3), όμως η δεύτερη είναι η μόνη που μπορεί να προσφέρει 30 fps, ενώ είναι και πιο κοντά στις διαστάσεις εισόδου των δύο μοντέλων. Ειδικά για το SSD-MobileNetV2, όπως θα αναφερθεί και στην ενότητα των αποτελεσμάτων, οποιαδήποτε άλλη ρύθμιση θα το περιόριζε πολύ περισσότερο.



Εικόνα 5.5: Inference μέσω κάμερας σε εικόνα

6 ΚΕΦΑΛΑΙΟ 6ο : Δημιουργία οπτικού και ηχητικού συστήματος ανατροφοδότησης

Στην ενότητα αυτή, θα γίνει η πλήρης περιγραφή της δημιουργίας του συστήματος που θα παρέχει θετική ή αρνητική ανατροφοδότηση. Οι δύο μορφές ανατροφοδότησης που θα χρησιμοποιηθούν είναι: α) η οπτική, μέσω οθόνης **LCD** (liquid crystal display) β) και η ηχητική μέσω **piezo buzzer**. Έτσι, το helmet detector αποκτά "ζωή", χρησιμοποιώντας τους εντοπισμούς του για την συμμόρφωση των μοτοσικλετιστών και ποδηλατών με σαφή μηνύματα, προκειμένου να τους παροτρύνει να φορέσουν κράνος ή την επιβράβευση αυτών που ήδη φοράνε. Η δημιουργία του συστήματος μπορεί να χωριστεί σε 4 κύρια στάδια, που θα αναλυθούν στις επόμενες υποενότητες.

6.1 Εξαγωγή σημάτων για τις εντοπισμένες κλάσεις από το Google Coral Dev Board

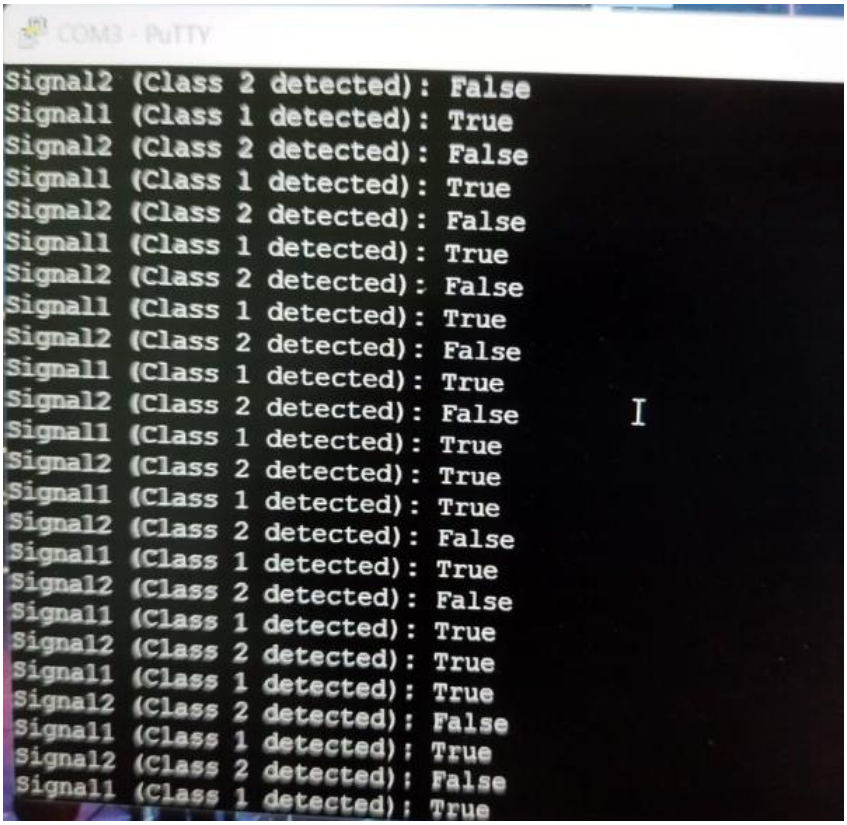
Με το Google Coral να είναι ήδη απασχολημένο με την εκτέλεση του object detector, είναι σαφές ότι θα ήταν καλό να χρησιμοποιηθεί ένας εξωτερικός μικροελεγκτής, που θα αναλάβει την οδήγηση της οθόνης LCD και του piezo buzzer. Πρέπει λοιπόν, να μεταφερθούν λογικά σήματα σχετικά με τις κλάσεις που εντοπίζονται από το μοντέλο. Το coral περιέχει αρκετά GPIO pins για αυτή την δουλειά, από τα οποία χρειάζονται μόλις δύο και θα είναι:

- το GPIO pin 36 για την κλάση With Helmet
- το GPIO pin 37 για την κλάση Without Helmet

Προφανώς, θα χρησιμοποιηθεί και ένα ground pin που θα συνδεθεί με ένα ground pin του μικροελεγκτή για να υπάρχει κοινό δυναμικό. Μέσω του python module που λέγεται python-periphery, μπορούν να προγραμματιστούν τα παραπάνω pins ώστε να μεταδίδουν λογικά σήματα ανάλογα με την κατάσταση εντοπισμού της κάθε κλάσης, δηλαδή να δίνουν τάση 3,3 V αν υπάρχει εντοπισμός και 0 V αν δεν υπάρχει. Ο κώδικας αυτός όμως, πρέπει να ενσωματωθεί στον κώδικα που εκτελεί το inference του μοντέλου. Ο χρήστης πρέπει να βεβαιωθεί ότι το python-periphery είναι εγκατεστημένο στο Google Coral, κάτι που γίνεται εύκολα με το 'pip'. Το αρχείο python που ευθύνεται για την εκτέλεση του inference είναι το 'detect.py' και βρίσκεται στην διεύθυνση '/usr/lib/python3/dist-packages/edgetpuvision/'. Πραγματοποιώντας την διαδικασία μεταφοράς αρχείων, απλά αυτή την φορά από το board στην κάρτα microSD, μπορεί να γίνει η επεξεργασία του script στον host υπολογιστή.

Το πρώτο πράγμα που πρέπει γίνει, είναι η εισαγωγή του python-periphery. Στην συνέχεια, πρέπει να οριστούν τα pin που θα μεταφέρουν τα σήματα. Σύμφωνα με τον οδηγό που παρέχει η Google [39], για τα GPIO pins 36 και 37 πρέπει να δοθεί το chip τους, που είναι το 4 και το 2 αντίστοιχα, η γραμμή τους, που είναι η 4 και για τα δύο και την επιθυμητή λειτουργία τους (είσοδος ή έξοδος), που είναι η έξοδος. Συνεχίζοντας, πρέπει να γίνει και μία μικρή αλλαγή στα χρώματα των bounding boxes, ώστε να ταιριάζουν περισσότερο με τις κλάσεις τους. Η προεπιλεγμένη παλέτα έδινε

το κυανό χρώμα στην κλάση ‘Without Helmet’ και το κόκκινο χρώμα για την κλάση ‘With Helmet’. Θα ήταν λοιπόν καλύτερο, αν γινόταν το αντίστροφο. Πηγαίνοντας στην συνάρτηση make_get_color, πρέπει να αντικατασταθεί το προηγούμενο μπλοκ κώδικα με ένα καινούργιο, όπου τα labels θα λαμβάνουν ένα συγκεκριμένο χρώμα από έναν χάρτη χρωμάτων τοποθετημένο σε ένα λεξικό, όπου το 0, δηλαδή η πρώτη κλάση (με κράνος), θα έχει το κυανό και το 1 (η δεύτερη κλάση/χωρίς κράνος), θα παίρνει το κόκκινο. Το επόμενο βήμα είναι να προστεθεί ένα κομμάτι κώδικα στην συνάρτηση render_gen, που είναι και αυτή που πραγματοποιεί το inference. Στο βρόχο επανάληψης while, είναι δυνατόν να εξάγονται οι καταστάσεις των κλάσεων και να τοποθετούνται σε μία βοηθητική boolean μεταβλητή, η οποία θα ορίζει και τα σήματα των pin 36 και 37. Η συνάρτηση any της Python επιστρέφει ‘True’, αν έστω ένα από τα στοιχεία μίας ακολουθίας ικανοποιεί μία συνθήκη. Άρα μέσω αυτής, ελέγχεται για κάθε κλάση, αν υπάρχει έστω και ένα εντοπισμένο αντικείμενο. Για λόγους ασφαλείας, ο βρόχος while τοποθετείται μέσα σε ένα try μπλοκ. Αυτό το μπλοκ συνοδεύεται πάντα από ένα αντίστοιχο finally μπλοκ και μέσω αυτών, μόλις για οποιοδήποτε λόγο σταματήσει η εκτέλεση του κώδικα στο try, εκτελείται πάντα και ο κώδικας στο finally. Αυτό επιτρέπει τον μηδενισμό των σημάτων και τον τερματισμό της λειτουργίας των pin, αφού τελειώσει το inference, είτε με εντολή του χρήστη, είτε από κάποιο σφάλμα. Για τον έλεγχο της ορθής λειτουργίας του κώδικα, μπορεί προσωρινά να γίνεται και η εκτύπωση κάποιου μηνύματος που θα περιέχει τις boolean τιμές των μεταβλητών που ορίζουν τα σήματα. Με το νέο και επεξεργασμένο script έτοιμο, μένει απλά να αντικατασταθεί το προηγούμενο, που βρίσκεται ακόμα στο Google Coral. Η διαδικασία μεταφοράς είναι η ίδια, ωστόσο αυτή την φορά επειδή τα αρχεία έχουν το ίδιο όνομα και μπορεί να προκύψουν διάφορα λάθη, καλό είναι το καινούργιο να έχει μία μικρή διαφορά. Με αυτά τα δύο να βρίσκονται στην ίδια διεύθυνση, πρέπει πρώτα να διαγραφεί το προηγούμενο και μετά να μετονομαστεί το νέο ως ‘detect.py’. Η εντολή που το επιτρέπει αυτό στο Mendel Linux είναι η mv (move), που ουσιαστικά μεταφέρει ένα αρχείο από μία διεύθυνση σε μία άλλη. Ωστόσο, ο χρήστης μπορεί απλά να το τοποθετήσει στην ίδια διεύθυνση, αλλά με διαφορετικό όνομα (μετονομασία).



Εικόνα 6.1: Τα λογικά σήματα των κλάσεων σε πραγματικό χρόνο

Ένα σημαντικό πλεονέκτημα του board είναι ότι η τάση που δίνεται όταν υπάρχει λογικό ‘1’, είναι 3,3 V και δεν υπάρχει περίπτωση να προκαλέσει ζημιά στον μικροελεγκτή (που θα χρησιμοποιηθεί), αν γίνει απευθείας σύνδεση μεταξύ τους. Επίσης, οι μικροελεγκτές που λειτουργούν με λογική 5 V, αντιλαμβάνονται όλα τα σήματα με τάση πάνω από 3 V ως λογικό ‘1’, άρα μπορεί να συνδεθούν, χωρίς κάποια παρέμβαση, με ένα απλό καλώδιο.

6.2 Επεξεργασία των σημάτων από μικροελεγκτή

Ο μικροελεγκτής που χρησιμοποιήθηκε είναι ο **ATmega328P**, ως μέρος της δημοφιλούς πλακέτας **Arduino Uno R3**, ωστόσο υπάρχουν πολύ καλύτερες επιλογές, όπως η οικογένεια μικροελεγκτών **stm32** της STMicroelectronics. Δυστυχώς όμως δεν υπήρχε κάποιος διαθέσιμος, οπότε η δουλειά ξεκίνησε και ολοκληρώθηκε στο Arduino. Η λογική πίσω από τα επόμενα στάδια είναι πρακτικά η ίδια, είτε χρησιμοποιείται κάποιο Arduino, είτε κάποιο stm32 board όπως ένα **ST Nucleo**.

Το Arduino Uno R3 θα προγραμματιστεί στο περιβάλλον προγραμματισμού **Arduino IDE** [40], που εκτός από ενσωματωμένο compiler, που κάνει την μεταφόρτωση του κώδικα γρήγορη και εύκολη (ο compiler δεν παράγει και τον πιο βελτιστοποιημένο κώδικα βέβαια), περιέχει μία τεράστια ποικιλία από βιβλιοθήκες για να γίνει την δουλειά ακόμη πιο εύκολη. Λόγω της δημοτικότητάς του,

υπάρχουν και πάρα πολλά εξαρτήματα στην αγορά, που κατασκευάζονται με σκοπό να τοποθετηθούν στην πλακέτα αυτή, γνωστά και ως shields, καθώς εφαρμόζονται πάνω στο Arduino σαν ασπίδα. Το Arduino Uno R3 παρέχει 13 pins για εισαγωγή ή εξαγωγή ψηφιακών σημάτων. Δύο από αυτά θα χρειαστούν για τα σήματα του Google Coral.

Ξεκινώντας λοιπόν με το κομμάτι των σημάτων των κλάσεων, πρέπει να οριστούν δύο ψηφιακά pins ως ψηφιακές είσοδοι. Ο ATmega328P λειτουργεί με λογική 5 V, όμως όπως αναφέρθηκε και στην προηγούμενη υποενότητα, αυτό δεν είναι πρόβλημα, καθώς οι τάσεις πάνω από 3 V αναγνωρίζονται κανονικά ως λογικό '1'. Άρα, ένα καλώδιο female-male ενώνει το GPIO pin 36 του Google Coral με το ψηφιακό pin 2 του Arduino, άλλο ένα ενώνει το GPIO pin 37 με το ψηφιακό pin 3 και άλλο ένα ενώνει δύο τυχαία ground pins ώστε να έχουν οι δύο πλακέτες κοινή γείωση (0 V).

Στο Arduino IDE, τα pins τοποθετούνται σε δύο (global) int (integer) μεταβλητές (π.χ. signal1pin και signal2pin). Στην συνέχεια, στην συνάρτηση setup, ορίζονται ως ψηφιακές είσοδοι με την χρήση της εντολής pinMode, με παραμέτρους τον αριθμό του pin και την επιθυμητή λειτουργία του (input).

Ο αλγόριθμος για την παραγωγή θετικών και αρνητικών μηνυμάτων είναι αρκετά απλός. Προφανώς υπάρχει περίπτωση να έχουν εντοπιστεί και οι δύο κλάσεις μέσα σε μία εικόνα, όμως δεν είναι δυνατό να εκπέμπονται και τα δύο μηνύματα ταυτόχρονα. Επειδή δίνεται περισσότερο έμφαση στην μη χρήση κράνους, δίνεται και η αντίστοιχη προτεραιότητα στην κλάση 'Without Helmet'. Συνεπώς, άμα η μεταβλητή της κλάσης αυτής έχει λογικό '1' (HIGH), τότε ενεργοποιούνται τα αρνητικά μηνύματα, ανεξαρτήτου κατάστασης της κλάσης 'With Helmet'. Άλλος ένας παράγοντας που πρέπει να ληφθεί υπόψη, είναι ότι τα μηνύματα πρέπει να δίνονται μία φορά για κάθε ξεχωριστή περίπτωση εντοπισμού, δηλαδή να μην επαναλαμβάνονται για το ίδιο αντικείμενο. Η λύση είναι η δημιουργία δύο επιπλέον boolean μεταβλητών, που θα δείχνουν την κατάσταση των μηνυμάτων της κάθε κλάσης (αρχικοποιούνται ως 'False'). Επιστρέφοντας στον αλγόριθμο και έχοντας τα παραπάνω υπόψη, αρχικά διαβάζεται μέσω του digitalRead, η τάση του pin 3 (Without Helmet). Το αρνητικό μήνυμα, που είναι και το πρώτο που ελέγχεται, θα παίξει αν η τιμή της τάσης δώσει λογικό '0' και ταυτόχρονα η κατάστασή του είναι False. Αφού περάσουν 100ms, μέσω της συνάρτησης delay, γίνεται πάλι ο έλεγχος της τιμής της τάσης του pin για να βεβαιωθεί το σύστημα ότι πρόκειται για σταθερό εντοπισμό και όχι στιγμιαίο. Αν και μόνο αν, ικανοποιηθούν αυτές οι συνθήκες, τότε ενεργοποιείται το 'αρνητικό' μπλοκ κώδικα. Αν δεν γίνει αυτό, τότε διαβάζεται η τιμή της τάσης στο pin2 (With Helmet) και ακολουθεί η ίδια διαδικασία με πριν, για τον έλεγχό της. Αν δεν υπάρχει ούτε αυτή η κλάση, τότε μετά από 100 ms γίνεται πάλι σύγκριση των σημάτων των δύο ψηφιακών pin και αν δώσουν λογικό '0', απενεργοποιούνται όλα τα μηνύματα και τίθονται οι μεταβλητές κατάστασης boolean σε 'False'. Το 'θετικό' και το 'αρνητικό' μπλοκ κώδικα θα ολοκληρωθούν στα επόμενα βήματα.

6.3 Δημιουργία γραφικών για το TFT Display

Ο οθόνη που θα χρησιμοποιηθεί στην διπλωματική εργασία είναι ένα Thin-Film-Transistor Liquid-Crystal Display (TFT LCD). Για την μεταφορά δεδομένων ανάμεσα σε αυτό και τον μικροελεγκτή, απαιτείται ένα πρωτόκολλο που λέγεται **SPI** (Serial Periphery Interface). Μέσω του SPI επιτυγχάνεται σύγχρονη σειριακή επικοινωνία μεταξύ μίας συσκευής master (π.χ. ένας μικροελεγκτής) και πολλαπλών περιφερειακών συσκευών που λέγονται slaves (όπως η οθόνη). Χρησιμοποιεί 4 λογικά σήματα για την σωστή λειτουργία του [41]:

- ΤΟ CS (Chip Select) ή SS (Slave Select): Λαμβάνει λογικό ‘1’ από την συσκευή που πρόκειται να επικοινωνήσει
- ΤΟ SCLK (To Serial Clock): Το ρολόι του master που δημιουργεί τετραγωνικούς παλμούς τάσης σε μία συχνότητα (clock frequency).
- ΤΟ MOSI (Master-Out Slave-In): Σειριακά δεδομένα από τον master στον slave με πρώτο bit το most-significant-bit (MSB)
- ΤΟ MISO (Master-In Slave-Out): Σειριακά δεδομένα από τον slave στον master με πρώτο bit το most-significant-bit (MSB)

Η βιβλιοθήκη SPI.h αναλαμβάνει την μεταφορά των δεδομένων μέσω του SPI και παρέχει και συναρτήσεις για την περαιτέρω ρύθμισή του, όπως την αλλαγή της συχνότητας του ρολογιού SCLK. Επειδή η οθόνη είναι ένα shield 3,5 ιντσών (320x480), κατασκευασμένο από την Waveshare και έχει pins που συνδέονται με συγκεκριμένες εισόδους της πλακέτας Arduino Uno R3, υπάρχει μία βιβλιοθήκη αποκλειστικά για αυτή που απλώς θέτει όλα τα pins του Arduino που χρειάζονται στα αντίστοιχα του shield, ρυθμίζει το SPI για τα δεδομένα του οδηγού της οθόνης (ILI9486) μέσω εντολών του SPI.h και παρέχει επιπλέον συναρτήσεις που ελέγχουν άλλες λειτουργίες, όπως την αναστροφή της εικόνας ή το επίπεδο φωτεινότητάς της. Αυτό το βήμα δεν είναι υποχρεωτικό και μπορεί ο χρήστης να το παραλείψει, κάνοντας μόνος του όλες αυτές τις ρυθμίσεις, ωστόσο επειδή αυτές είναι συγκεκριμένες για αυτή την οθόνη, προτείνεται απλά να χρησιμοποιήσει τις συναρτήσεις που παρέχονται για να κάνει οποιαδήποτε αλλαγή επιθυμεί. Η οθόνη αυτή μπορεί να λειτουργήσει και ως οθόνη αφής (touchscreen), όμως αυτό δεν είναι αναγκαίο για την εργασία, οπότε τα pins που συσχετίζονται με αυτή την λειτουργία δεν θα συνδεθούν.

Εικόνα 6.2 και Εικόνα 6.3: Το tft lcd από πίσω (αριστερά) και από μπροστά (δεξιά)

Για την σχεδίαση των γραφικών μπορεί να αξιοποιηθεί μία πολύ δημοφιλής βιβλιοθήκη, γραμμένη από την Adafruit, την **Adafruit GFX**. Αυτή δίνει την δυνατότητα σχεδίασης απλών σχημάτων, όπως ένας κύκλος ή ένα ορθογώνιο παραλληλόγραμμο ή και πολύπλοκων σχεδίων bitmap μέσω της συνάρτησης drawBitmap. Σκοπός είναι να δημιουργηθεί ένα απλό σχέδιο, που θα απεικονίζει ένα κράνος, το χρώμα του οποίου θα είναι είτε κόκκινο είτε πράσινο. Ορίζεται λοιπόν μία συνάρτηση drawHelmet, που θα λαμβάνει ως είσοδο το χρώμα. Μέσα σε αυτή θα περιέχονται οι συναρτήσεις της Adafruit GFX με συγκεκριμένα δεδομένα, για να χαράζεται πάντα ένα συγκεκριμένο σχέδιο. Οι συναρτήσεις που χρειάστηκαν ήταν:

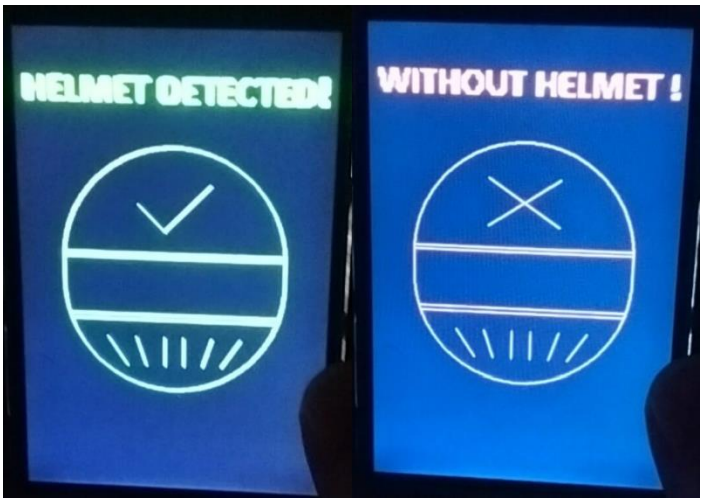
1. το drawRoundRect: Σχεδιάζει ένα στρογγυλεμένο ορθογώνιο. Λαμβάνει τις συντεταγμένες x,y της αρχής (πάνω αριστερά) και του τέλους (κάτω δεξιά) του παραλληλογράμμου, τον αριθμό των pixel που θα έχει η κάθε γωνία και τέλος, το χρώμα των pixel σε μορφή αριθμού 16 bit.
2. το drawRect: Σχεδιάζει ένα ορθογώνιο παραλληλόγραμμο. Λαμβάνει τις συντεταγμένες x,y της αρχής (πάνω αριστερά), του τέλους (κάτω δεξιά) του παραλληλόγραμμου και το χρώμα των pixel σε μορφή αριθμού 16 bit.
3. το drawLine: Σχεδιάζει μία ευθεία γραμμή. Λαμβάνει τις συντεταγμένες x,y της αρχής, του τέλους της και το χρώμα των pixel σε μορφή αριθμού 16 bit.

Αρχικά, σχεδιάζεται η βάση του κράνους ως ένα στρογγυλεμένο παραλληλόγραμμο, τοποθετημένο στο κέντρο της οθόνης και με 125 pixel ανά γωνία. Συνέχεια έχει το γείσο του κράνους. Για αυτό σχεδιάζονται γραμμές περίπου στην μέση του στρογγυλεμένου παραλληλογράμμου, με την μεταξύ τους απόσταση να είναι 38 pixel. Για περισσότερη λεπτομέρεια, σχεδιάζονται άλλες δύο γραμμές με απόσταση 5 pixel από την κάθε γραμμή του γείσου. Παρατηρείται από τις πρώτες

δοκιμές, ότι το πάνω κομμάτι του μπορεί απλά να σχεδιαστεί ως ένα ορθογώνιο παραλληλόγραμμο. Οι τελευταίες 6 γραμμές είναι για την σχεδίαση των καναλιών εξαερισμού, που βρίσκονται μερικές φορές στο κάτω μέρος ενός κράνους.

Στο πάνω κομμάτι του κράνους που είναι κενό και πάνω από το ίδιο το σχέδιο, θα εμφανίζονται σύμβολα και γραπτά μηνύματα, ανάλογα με την ανατροφοδότηση που θα δοθεί από το σύστημα. Αυτά θα είναι σε μορφή ενός μονοχρωματικού πίνακα, όπου το κάθε bit αντιπροσωπεύει ένα pixel, που είναι '0' για μαύρο και '1' για λευκό (byte array – 8 bit bitmap). Για την εύκολη δημιουργία ενός τέτοιου **bitmap**, μπορεί να δημιουργηθεί ένα σχέδιο στο **Microsoft Paint** ή στο **Gimp** και στην συνέχεια να μετατραπεί σε μονοχρωματικό bitmap. Μετά, προτείνεται η χρήση κάποιου εργαλείου, όπως το image2cpp [42], για την μετατροπή του σε πίνακα. Οι πίνακες αυτοί πρέπει να αποθηκευτούν στην μνήμη flash (μέσω του PROGMEM) και να δοθούν στην συνάρτηση drawBitmap, μαζί με τις αρχικές συντεταγμένες x,y, τις διαστάσεις του πίνακα και τέλος, το χρώμα των pixel σε μορφή αριθμού 16 bit.

- Για την θετική ανατροφοδότηση: ένα γραπτό μήνυμα 'Helmet Detected!' και το σύμβολο checkmark στο πάνω μέρος του κράνους
- Για την αρνητική ανατροφοδότηση: ένα γραπτό μήνυμα 'Without Helmet!' και το σύμβολο 'X' στο πάνω μέρος του κράνους.
-



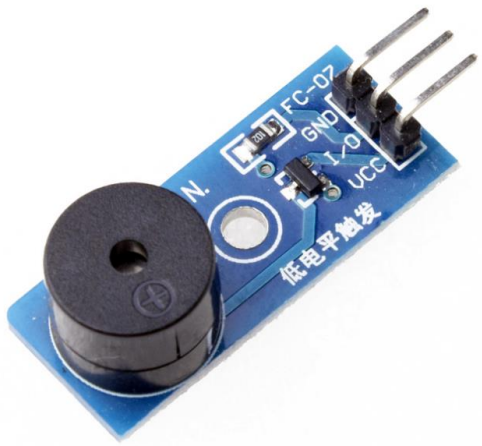
Εικόνα 6.4: Τα γραφικά για την θετική (αριστερά) και την αρνητική (δεξιά) ανατροφοδότηση

Οι παραπάνω εντολές drawBitmap δεν χρειάζεται να βρίσκονται μέσα στην συνάρτηση drawHelmet. Μετά από μερικές πρόχειρες δοκιμές, παρατηρείται ότι η σχεδίαση του κράνους είναι πολύ πιο γρήγορη από αυτή των bitmap. Ο κύριος λόγος είναι ότι η συνάρτηση drawHelmet δεν χρειάζεται να διαβάσει δεδομένα από την μνήμη flash (που είναι πιο αργή από την sram) και επειδή

έχει συγκεκριμένα pixels που θα ενεργοποιήσει (ορισμένα από τον προγραμματιστή), δεν χρειάζεται να εκτελεί ελέγχους για το κάθε bit των πινάκων.

6.4 Δημιουργία ηχητικών σημάτων

Για τα ηχητικά σήματα, χρησιμοποιείται ένα piezo buzzer. Το piezo buzzer ενισχύεται μέσω ενός **MOSFET** (Metal Oxide Semiconductor Field Effect Transistor). Το gate του συνδέεται με ένα ψηφιακό pin του Arduino, το pin 4. Το drain συνδέεται με σταθερή τροφοδοσία 3,3 V από το αντίστοιχο pin του Arduino και το source γειώνεται με σύνδεση σε ένα ground pin του Google Coral, καθώς δεν υπήρχε διαθέσιμο στο Arduino. Αυτό το buzzer δουλεύει με βάση το πιεζοηλεκτρικό φαινόμενο (συλλογή ηλεκτρικού φορτίου σε κάποιο στερεό), για την παραγωγή ήχου. Η αρχική μηχανική κίνηση γίνεται με την εφαρμογή τάσης στο υλικό. Αυτή μετά μετατρέπεται σε ακουστό ήχο μέσω διαφραγμάτων και συντονιστών [43]. Υπάρχουν δύο είδη piezo buzzer στην αγορά, το **παθητικό** (passive) και το **ενεργό** (active). Το ενεργό παράγει έναν συγκεκριμένο ήχο (συγκεκριμένο τόνο) για όσο χρόνο του εφαρμόζεται τάση. Το παθητικό χρειάζεται ένα κυματοειδές σήμα για την παραγωγή ήχου. Αυτό σημαίνει ότι με την χρήση σημάτων διαφορετικών συχνοτήτων, ένα παθητικό buzzer μπορεί να παράγει ήχο με διαφορετικές νότες. Επειδή θα παίζει ήχος και για τις δύο περιπτώσεις ανατροφοδότησης, θα χρησιμοποιηθεί ένα παθητικό buzzer, το οποίο θα παράγει γρήγορες μελωδίες τριών νοτών.



Εικόνα 6.5: Το παθητικό piezo buzzer σε module [44]

Στο Arduino IDE υπάρχουν συναρτήσεις αποκλειστικά για αυτό. Συγκεκριμένα για την παραγωγή ήχου, υπάρχει η tone, που λαμβάνει το ψηφιακό pin που είναι συνδεδεμένο με το buzzer,

την συχνότητα του κυματοειδούς σήματος και προαιρετικά την διάρκεια του ήχου. Αξίζει να σημειωθεί ότι η τρίτη είσοδος μερικές φορές δημιουργεί προβλήματα, άρα καλύτερα να συνοδεύονται οι εντολές tone με κάποιο delay και όταν πρέπει να σταματήσει ο ήχος, να χρησιμοποιείται η συνάρτηση noTone μαζί με το ψηφιακό pin του buzzer. Έχοντας υπόψη τα παραπάνω, ήρθε η ώρα για την δημιουργία των μελωδιών.

Για την μελωδία της θετικής ανατροφοδότησης: Η μελωδία αυτή θα έχει μία ακολουθία από ήχους με όλο και υψηλότερη συχνότητα, ώστε να θυμίζει την επίτευξη κάποιου στόχου (την οδική ασφάλεια). Οι νότες που χρησιμοποιήθηκαν είχαν συχνότητες 523 Hz, 659 Hz και 784 Hz αντίστοιχα, ενώ η διάρκεια της καθεμίας ήταν 100 ms.

Για την μελωδία της αρνητικής ανατροφοδότησης: Η μελωδία αυτή πρέπει να έχει προειδοποιητικό χαρακτήρα, άρα είναι καλή ιδέα η επανάληψη ενός ήχου υψηλής συχνότητας τρεις φορές. Η συχνότητα της νότας θα είναι 1000 Hz και η διάρκεια της καθεμίας θα είναι και πάλι 100 ms.

Τέλος, για να είναι πιο καλογραμμένος ο κώδικας, οι εντολές που χρησιμοποιούνται για την κάθε μελωδία, μπορούν να εισαχθούν σε μία δική τους συνάρτηση, που θα καλείται μέσα στο αντίστοιχο μπλοκ κώδικα της κάθε ανατροφοδότησης.

6.5 Μπλοκ κώδικα θετικής και αρνητικής ανατροφοδότησης

Έχοντας έτοιμες όλες τις συναρτήσεις, όπως και τον βασικό αλγόριθμο, μένει απλά να τοποθετηθούν με την σωστή σειρά μέσα στα μπλοκ κώδικα για την θετική και αρνητική ανατροφοδότηση, στον συνεχώς επαναλαμβανόμενο κώδικα. Η σειρά με την οποία εκτελούνται οι εντολές αυτές είναι άκρως σημαντική, καθώς εκτελούνται σειριακά, δηλαδή δεν γίνεται να προχωρήσει το πρόγραμμα στην επόμενη εντολή, αν δεν έχει ολοκληρωθεί η προηγούμενη. Οι συναρτήσεις positiveSound και negativeSound, που είναι αυτές που αναλαμβάνουν την εκτέλεση του κώδικα των μελωδιών, δεν πρέπει να είναι πρώτες στα μπλοκ, γιατί η σχεδίαση των γραφικών θα ξεκινήσει αφού σταματήσει ο ήχος. Αντιθέτως, με τα γραφικά να ολοκληρώνονται σε μόλις λίγα millisecond, η παραγωγή του ήχου δεν καθυστερεί ιδιαίτερα. Για την εκτύπωση των γραφικών στην οθόνη, πρέπει πρώτα να αφαιρεθούν όλα τα περιεχόμενά της, μέσω της συνάρτησης fillScreen μαζί με το επιθυμητό χρώμα που είναι το μαύρο. Στην συνέχεια, πρέπει να τοποθετηθούν οι συναρτήσεις, drawBitmap με το σύμβολο, drawHelmet (σχεδίασης του κράνους) και η drawBitmap με το κείμενο, με μοναδικό κριτήριο τον χρόνο εκτέλεσής τους. Πρώτη λοιπόν είναι η σχεδίαση του συμβόλου, μετά του κράνους και τέλος του κειμένου. Το χρώμα των pixel θα είναι πράσινο για τα θετικά μηνύματα (0x07E0) και κόκκινο για τα αρνητικά (0xF800). Επίσης είναι σημαντικό να αλλάξουν και οι τιμές των boolean μεταβλητών που δείχνουν τις καταστάσεις των μηνυμάτων, σε ‘True’ για αυτή που αντιστοιχεί στην κλάση που εντοπίστηκε και ‘False’ για την άλλη. Αν δεν υπάρχει κάποια

εντοπισμένη κλάση, τότε στο αντίστοιχο μπλοκ κώδικα, μέσω του fillScreen όλα τα pixels γίνονται μαύρα, τερματίζεται οποιαδήποτε νότα παράγεται (αν παράγεται) με το noTone και οι δύο boolean μεταβλητές κατάστασης τίθονται ως 'False'.

7 ΚΕΦΑΛΑΙΟ 7ο : Δοκιμές και Αποτελέσματα

Στην ενότητα αυτή θα αναλυθεί η διαδικασία της **τελικής δοκιμής** του συστήματος και θα συζητηθούν τα αποτελέσματα. Συγκεκριμένα, θα χρησιμοποιηθούν τα μοντέλα των δύο διαφορετικών αρχιτεκτονικών που εκπαιδεύτηκαν (SSD-MobileNetV2 + SSD-MobileNetV2-FPNlite) πάνω σε **καινούργιες εικόνες**, που περιέχουν ανθρώπους που είτε φοράνε, είτε δεν φοράνε κράνος. Σε αυτό το σύνολο εικόνων, τα αντικείμενα που πρέπει να εντοπιστούν έχουν διαφορετικό βαθμό δυσκολίας, είτε λόγω απόστασης, είτε λόγω φωτεινότητας, είτε λόγω κάποιου εμποδίου. Το μεγάλο πλεονέκτημα με αυτή την μέθοδο είναι ότι και τα δύο θα εφαρμοστούν στις ίδιες συνθήκες, οπότε αποκλείονται οι περιπτώσεις που οι καιρικές συνθήκες ή τα ίδια τα δείγματα έχουν μικρές διαφορές. Παράλληλα με τις δοκιμές, θα γίνεται και μέτρηση της τάσης και της έντασης του ρεύματος για κάθε μοντέλο, ώστε να αναλυθεί αργότερα η κατανάλωση ενέργειας που έχει το καθένα. Για να γίνουν τα αποτελέσματα πιο καθαρά, θα δημιουργηθούν οι αντίστοιχοι **πίνακες σύγχυσης** (confussion matrix) και θα υπολογιστεί και το **mAP** (mean average precision). Τέλος, θα γίνουν και μερικές γρήγορες δοκιμές για το πρώτο μοντέλο, που εκπαιδεύτηκε με το αρχικό σύνολο δεδομένων (πριν δηλαδή εφαρμοστούν οι μέθοδοι βελτιστοποίησης).

7.1 Δοκιμές σε εικόνες

Μαζεύτηκαν 11 εικόνες, στις οποίες τα αντικείμενα έχουν διαφορετικά επίπεδα ορατότητας, φωτεινότητας και απόστασης. Επίσης υπάρχουν ορισμένα αντικείμενα που έχουν ως σκοπό να δυσκολέψουν τον εντοπισμό, όπως σκουφιά, κουκούλες κ.λπ. Το σύνολο των αντικειμένων ήταν 46, με 23 περιπτώσεις χρήσης κράνους και 23 περιπτώσεις απουσίας κράνους. Παρακάτω, θα περιγραφούν αναλυτικά τα αποτελέσματα των δοκιμών για καθεμία εικόνα ξεχωριστά. Αξίζει να σημειωθεί, ότι οι περιπτώσεις μετρούνται για την κάθε εικόνα, όπως εμφανίζονται από τα αριστερά έως τα δεξιά. Επίσης, η **ελάχιστη τιμή σιγουριάς** (minimum confindence threshold) και για τα δύο μοντέλα κατά την διάρκεια των δοκιμών, ήταν ορισμένη ως 0,3 (30%). Εικόνες που περιέχουν αντικείμενα σε νυχτερινές ώρες δεν χρησιμοποιήθηκαν, καθώς και τα δύο μοντέλα δεν μπορούν να πραγματοποιήσουν εντοπισμούς σε τόσο χαμηλά επίπεδα φωτός.

Εικόνα ‘test1’:

Στην εικόνα αυτή υπάρχουν δύο άνθρωποι, ένας με κράνος και ένας άλλος χωρίς. Η ορατότητα είναι πολύ καλή και η απόσταση αρκετά μικρή. Έτσι, και τα δύο μοντέλα δεν είχαν κάποια δυσκολία στο να εντοπίσουν και τα δύο αντικείμενα. Οι εντοπισμοί επίσης ήταν σταθεροί, καθώς δεν παρουσιάστηκαν μεταβολές στις προβλέψεις των κλάσεων και τα bounding boxes παρέμειναν σταθερά και με μεγάλη ακρίβεια. Ως πρώτη εικόνα, έπρεπε να αντιπροσωπεύονται και οι δύο κλάσεις με καθαρά χαρακτηριστικά, ώστε να υπάρχει βεβαίωση ότι στις ευκολότερες δυνατές συνθήκες, μπορεί ένα μοντέλο να δουλέψει σωστά. Οι υπόλοιπες φυσικά θα παρουσιάζουν κάποιες δυσκολίες, ώστε να δοκιμαστούν και στα όριά τους.

Εικόνα 7.1: Η εικόνα ‘test1’ με αριθμημένες τις περιπτώσεις [45]

Εικόνα ‘test2’:

Στην δεύτερη εικόνα υπάρχουν πολλοί μηχανόβιοι χωρίς κράνος σε μία σειρά, που από τα αριστερά έως τα δεξιά βρίσκονται ολοένα και πιο μακριά. Το άτομο που βρίσκεται τρίτο από αριστερά φοράει καπέλο, ενώ το πέμπτο μία μπαντάνα. Η δοκιμή σε αυτή την εικόνα θα αναδείξει την ικανότητα εντοπισμού της κλάσης ‘Without Helmet’ σε συνθήκες ικανοποιητικής φωτεινότητας, καλής ορατότητας, αλλά σε ολοένα και μικρότερα αντικείμενα (λόγω απόστασης). Το SSD-MobileNetV2 κατάφερε να εντοπίσει ορθά την πρώτη, την τρίτη και την τέταρτη περίπτωση, ωστόσο κατηγοριοποίησε την δεύτερη ως ‘κράνος’ που είναι φυσικά λάθος. Ο εντοπισμός αυτός όμως, δεν ήταν σταθερός. Το μοντέλο με μικρές μεταβολές στην γωνία ή την απόσταση, άλλαζε την κλάση που εντόπιζε. Οι τελευταίες δύο περιπτώσεις, που ήταν και οι πιο απομακρυσμένες, δεν εντοπίστηκαν. Το καπέλο της περίπτωσης νούμερο τρία, δεν φάνηκε να δυσκόλεψε το μοντέλο. Το SSD-MobileNetV2-FPNlite κατάφερε να εντοπίσει σωστά τις περιπτώσεις ένα και τέσσερα, ενώ δεν κατάφερε να εντοπίσει τις υπόλοιπες. Όλες οι προβλέψεις επίσης ήταν σταθερές. Αυτό είναι ένα καθαρό σημάδι, ότι το μοντέλο αυτό θα είναι πιο συντηρητικό από το πρώτο, ωστόσο αναμένεται να έχει καλύτερη επίδοση στο κομμάτι της κατηγοριοποίησης.

Εικόνα 7.2: Η εικόνα ‘test2’ με αριθμημένες τις περιπτώσεις [46]

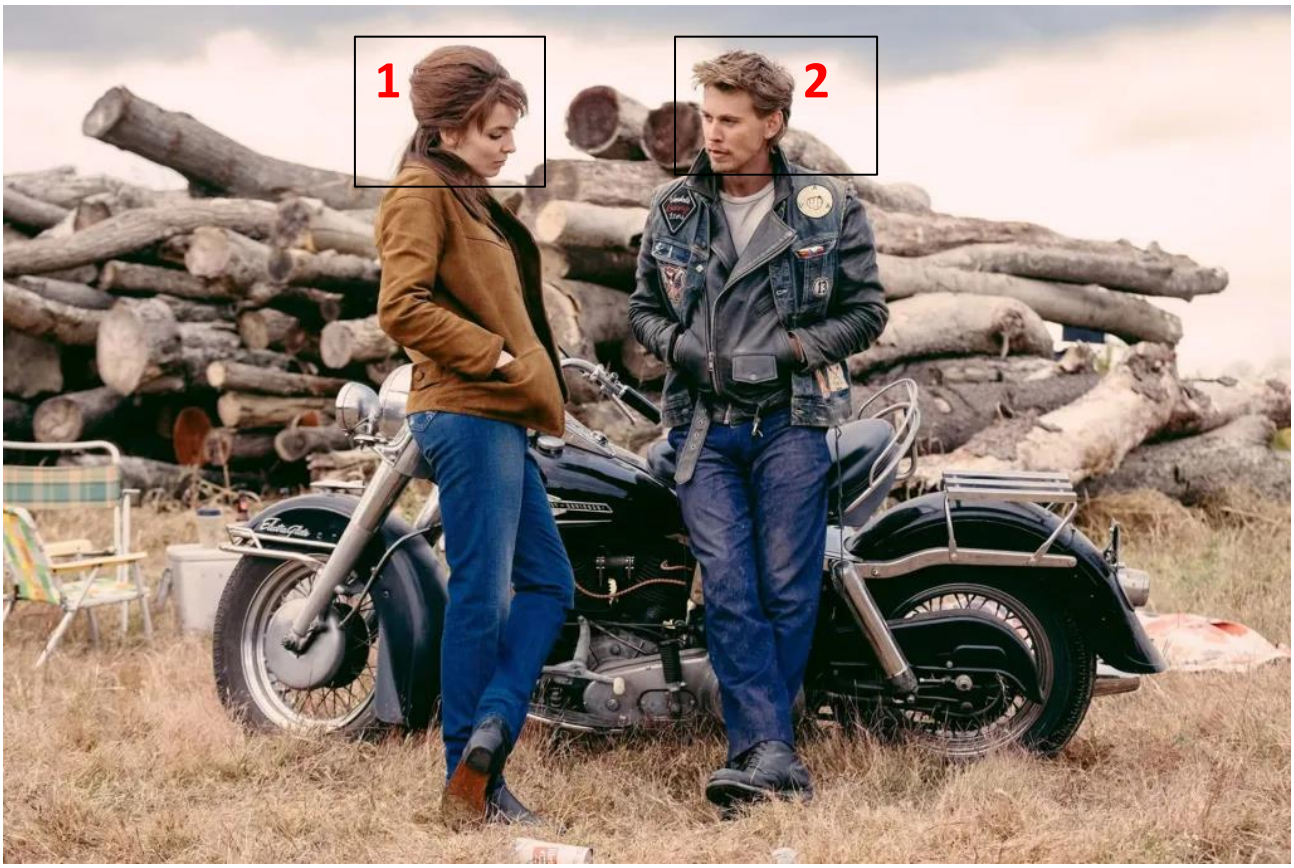
Εικόνα ‘test3’:

Η τρίτη εικόνα έχει την ίδια λογική με την δεύτερη, αλλά αυτή την φορά δοκιμάζει την ικανότητα εντοπισμού της κλάσης ‘With Helmet’. Περιέχει πέντε άτομα που φοράνε κράνος μηχανής, με καλή ορατότητα, καλή φωτεινότητα και αύξουσα από τα αριστερά έως τα δεξιά απόσταση. Το SSD-MobileNetV2 βρήκε σωστά τις πρώτες τρεις περιπτώσεις, αλλά απέτυχε στον εντοπισμό των τελευταίων δύο, που βρισκόντουσαν και πιο μακριά. Σημειώνεται ότι οι προβλέψεις για την πρώτη περίπτωση, δεν ήταν πάντα σταθερές. Αρκετό ενδιαφέρον αποτελούν τα αποτελέσματα της δοκιμής με το αντίστοιχο FPNlite μοντέλο. Αυτό, είχε σωστές προβλέψεις στις περιπτώσεις δύο και τρία και κατηγοριοποίησε λάθος την πρώτη περίπτωση ως ‘Without Helmet’. Το πόρισμα που βρέθηκε από την προηγούμενη εικόνα, ότι δηλαδή το FPNlite μοντέλο θα είναι πιο συντηρητικό με τις προβλέψεις του, δεν φαίνεται να ισχύει, καθώς εντόπισε και την τέταρτη περίπτωση, ωστόσο την κατηγοριοποίησε και αυτή ως ‘Without Helmet’ που είναι λάθος. Για άλλη μία φορά, οι προβλέψεις δεν είχαν κάποια μεταβολή ως προς την κλάση τους. Η πέμπτη περίπτωση δεν εντοπίστηκε. Είναι εμφανές ότι το μοντέλο αυτό τείνει πιο εύκολα στην δεύτερη κλάση (Without Helmet).

Εικόνα 7.3: Η εικόνα ‘test3’ με αριθμημένες τις περιπτώσεις [47]

Εικόνα ‘test4’:

Στην τέταρτη εικόνα εμφανίζονται δύο άτομα χωρίς κράνος στην εξοχή. Η φωτεινότητα είναι πολύ καλή και η ορατότητα είναι επίσης αρκετά ικανοποιητική. Πρόκειται για άλλο ένα σχετικά εύκολο ζεύγος από αντικείμενα της κλάσης ‘Without Helmet’, που και τα δύο μοντέλα βρήκαν πολύ εύκολα.



Εικόνα 7.4: Η εικόνα ‘test4’ με αριθμημένες τις περιπτώσεις [48]

Εικόνα ‘test5’:

Στην πέμπτη εικόνα παρουσιάζονται τέσσερα άτομα με κράνος, που βρίσκονται σε μεγάλες αποστάσεις. Η φωτεινότητα είναι αρκετά υψηλή. Η ορατότητα για την περίπτωση νούμερο ένα είναι καλή, ωστόσο οι περιπτώσεις δύο και τρία μπορεί να δημιουργήσουν δυσκολίες, καθώς τα κράνη εφάπτονται. Η τέταρτη περίπτωση βρίσκεται πολύ μακριά και είναι θολή. Το SSD-MobileNetV2 κατάφερε να εντοπίσει σωστά την πρώτη περίπτωση, όμως κατηγοριοποίησε λάθος τις επόμενες δύο ως ‘Without Helmet’, σε αντίθεση με το FPNlite, που οι προβλέψεις του για αυτές ήταν όλες σωστές. Και τα δύο απέτυχαν στον εντοπισμό του τέταρτου και πιο δύσκολου αντικειμένου. Ενώ στην τρίτη εικόνα το FPNlite έδειξε μία τάση προς την κατηγοριοποίηση των κράνων ως ‘Without Helmet’, η επίδοσή του σε αυτή την εικόνα ήταν αρκετά καλύτερη, ξεπερνώντας εύκολα το SSD-MobileNetV2. Οι προβλέψεις και για τα δύο μοντέλα ήταν σταθερές.

Εικόνα 7.5: Η εικόνα ‘test5’ με αριθμημένες τις περιπτώσεις [49]

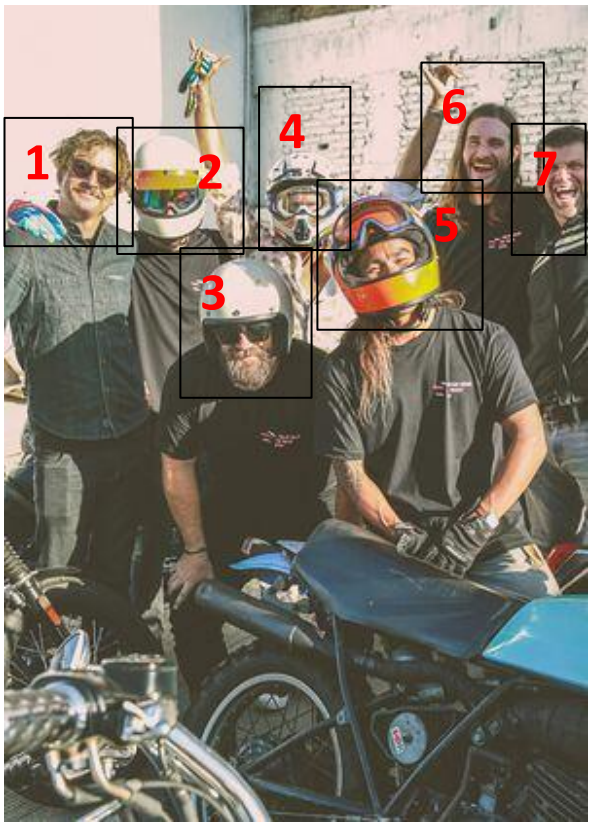
Εικόνα ‘test6’:

Η έκτη εικόνα είναι άλλη μία σχετικά εύκολη. Σε αυτή, εμφανίζονται δύο άτομα σε μία μηχανή που φοράνε κράνος. Ενώ και τα δύο μοντέλα μέχρι στιγμής είχαν κάποια προβλήματα με τα κράνη σε ορισμένες περιπτώσεις, σε αυτή δεν υπήρχε κάποια δυσκολία, ούτε στον εντοπισμό, ούτε στην κατηγοριοποίηση. Οι προβλέψεις ήταν σταθερές και τα bounding boxes ακριβή.

Εικόνα 7.6: Η εικόνα ‘test6’ με αριθμημένες τις περιπτώσεις [50]

Εικόνα ‘test7’:

Η έβδομη εικόνα περιέχει άτομα με και χωρίς κράνος, όπου η απόσταση είναι πρακτικά η ίδια. Η φωτεινότητα είναι ίσως υπερβολικά υψηλή, κάνοντας την ορατότητα αρκετά κακή για ορισμένες περιπτώσεις. Συνολικά είναι επτά άτομα, τέσσερα από τα οποία φοράνε κράνος και τρία δεν φοράνε. Από τα τέσσερα κράνη, τα τρία είναι λευκά και με τον τοίχο από πίσω να είναι και αυτός λευκός, αναμένονται ενδιαφέροντα αποτελέσματα. Οι τρεις περιπτώσεις χωρίς κράνος έχουν ικανοποιητική ορατότητα, άρα θεωρητικά δεν θα έπρεπε να υπάρχει κάποιο πρόβλημα. Και αυτό ακριβώς έγινε. Και τα δύο μοντέλα κατηγοριοποίησαν σωστά και σταθερά τις περιπτώσεις ένα, έξι και επτά ως ‘Without Helmet’. Το ίδιο δεν μπορεί να ειπωθεί για τις περιπτώσεις δύο, τρία, τέσσερα και πέντε, που ήταν με κράνος. Συγκεκριμένα, το SSD-MobileNetV2 εντόπισε μόνο την τρίτη και την πέμπτη, αλλά οι προβλέψεις ήταν σωστές και σταθερές. Το SSD-MobileNetV2-FPNlite απ’ την άλλη, εντόπισε τα ίδια αντικείμενα, ωστόσο έκανε λάθη στην κατηγοριοποίησή τους. Οι περιπτώσεις δύο και τέσσερα δεν εντοπίστηκαν και από τα δύο μοντέλα, με τα λευκά κράνη, σε συνδυασμό με τον λευκό τοίχο, να τα καθιστούν πρακτικά αόρατα.



Εικόνα 7.7: Η εικόνα ‘test7’ με αριθμημένες τις περιπτώσεις [51]

Εικόνα ‘test8’:

Στην όγδοη εικόνα βρίσκονται τρεις ποδηλάτες χωρίς κράνος. Η φωτεινότητα είναι ικανοποιητική, η απόσταση είναι μέτρια προς μεγάλη και η ορατότητα είναι επίσης σχετικά καλή. Και τα δύο μοντέλα είχαν τα ίδια αποτελέσματα, εντοπίζοντας τις πρώτες δύο περιπτώσεις σωστά, χωρίς να μπορούν να εντοπίσουν με σταθερότητα την περίπτωση νούμερο τρία. Το SSD-MobileNetV2 ήταν το μόνο που με μερικές μικρές μεταβολές στην γωνία της κάμερας, εντόπιζε έστω και για λίγο, την μη ύπαρξη κράνους σε αυτή. Επειδή όμως ήταν για μικρό χρονικό διάστημα (δεν ήταν σταθερή), δεν μέτρησε ως εύστοχη πρόβλεψη.

Εικόνα 7.8: Η εικόνα ‘test8’ με αριθμημένες τις περιπτώσεις [52]

Εικόνα ‘test9’:

Η ένατη εικόνα είναι μακράν η πιο δύσκολη. Περιέχει συνολικά δέκα περιπτώσεις, σε διαφορετικές αποστάσεις, με μέτρια φωτεινότητα και μέτρια προς κακή ορατότητα. Η πρώτη έχει ένα άτομο με κράνος ποδηλάτου, είναι σχετικά μακριά και η ορατότητά του είναι χαμηλή, με τα χρώματα του φόντου να μην αναδεικνύουν τα όριά του. Η δεύτερη περίπτωση είναι ένα άτομο με κουκούλα, αρκετά μακριά και το πρόσωπό του φαίνεται ελάχιστα. Η τρίτη, τέταρτη, έβδομη και δέκατη είναι οι πιο εύκολες, καθώς έχουν την μικρότερη απόσταση. Η πέμπτη, έκτη και ένατη είναι οι χειρότερες, με τα αντικείμενα να είναι τα μικρότερα από όλα, ενώ η όγδοη επικαλύπτεται σχεδόν ολικά από το έβδομο αντικείμενο. Το SSD-MobileNetV2 κατάφερε να εντοπίσει το πρώτο, το τρίτο, το πέμπτο, το έκτο, το έβδομο και το δέκατο αντικείμενο ορθά και με αρκετά μεγάλη σταθερότητα. Την περίπτωση νούμερο τέσσερα την εντόπισε λάθος ως ‘With Helmet’, ενώ το άτομο φοράει σκουφάκι. Για τις περιπτώσεις δύο, οκτώ και εννιά δεν εμφάνισε κάποια πρόβλεψη. Το SSD-MobileNetV2-FPNlite από την μεριά του, εντόπισε σωστά την πρώτη, την τρίτη, την τέταρτη, την έκτη και την έβδομη περίπτωση, λάθος την δέκατη περίπτωση, που την κατηγοριοποίησε ως ‘Without Helmet’, ενώ δεν εμφάνισε προβλέψεις για τα αντικείμενα δύο, πέντε, οκτώ και εννιά. Γενικά, ενώ το αναμενόμενο ήταν να δυσκολευτούν πολύ και τα δύο μοντέλα σε αυτή την εικόνα, φάνηκε ότι τα πήγαν αρκετά καλά, κάνοντας μονάχα ένα λάθος το καθένα και εντοπίζοντας ακόμα και μερικά αρκετά μακρινά αντικείμενα. Και τα δύο βέβαια δεν τα κατάφεραν για την περίπτωση του ατόμου με την κουκούλα (περίπτωση νούμερο δύο), για το άτομο με κράνος που βρίσκεται πίσω από ένα άλλο (περίπτωση νούμερο οκτώ) και για το άτομο που βρίσκεται πιο μακριά από όλα (περίπτωση νούμερο εννιά). Ενδιαφέρον έχει βέβαια, το γεγονός ότι το πρώτο μοντέλο εντόπισε σωστά την περίπτωση νούμερο δέκα, που ήταν και εύκολη, ενώ το FPNlite την κατηγοριοποίησε λάθος. Το πρώτο όμως έκανε λάθος στην περίπτωση νούμερο τέσσερα, ενώ το δεύτερο δεν δυσκολεύτηκε ιδιαίτερα με την ύπαρξη σκούφου.

Εικόνα 7.9: Η εικόνα 'test9' με αριθμημένες τις περιπτώσεις [53]

Εικόνα ‘test10’:

Η δέκατη εικόνα ήταν αρκετά ενδιαφέρουσα. Σε αυτή εμφανίζονται τρία άτομα με κράνος ποδηλάτου. Η πρώτη είναι πολύ μακριά και είναι και θολή, η δεύτερη είναι θολή αλλά σχετικά κοντά, ενώ η τρίτη έχει πολύ καλή ορατότητα και είναι και αρκετά κοντά. Η επιδόσεις των δύο μοντέλων σε αυτή την εικόνα ήταν προβληματικές. Ο μη εντοπισμός της πρώτης περίπτωσης δεν εκπλήσσει, αλλά οι άλλες δύο ήταν αρκετά πιο εύκολες και κατηγοριοποιήθηκαν ως ‘Without Helmet’ και από τα δύο μοντέλα. Αξίζει βέβαια να αναφερθεί, ότι το SSD-MobileNetV2 παρουσίαζε αρκετά μεγάλη αστάθεια ως προς την πρόβλεψη της κλάσης της τρίτης περίπτωσης, ενώ το FPNlite ήταν σχεδόν βέβαιο (80-90% σιγουριά).

Εικόνα 7.10: Η εικόνα ‘test10’ με αριθμημένες τις περιπτώσεις [54]

Εικόνα ‘test11’:

Η εικόνα νούμερο έντεκα έχει δύο άτομα πάνω σε ποδήλατα, χωρίς κράνος. Το περιβάλλον είναι μίας πόλης, η απόσταση είναι μέτρια, η φωτεινότητα ικανοποιητική, ωστόσο το φόντο μπορεί να δυσκολέψει λίγο τα μοντέλα, καθώς τα όρια των αντικειμένων δεν είναι πολύ καθαρά. Και τα δύο μοντέλα όμως δεν είχαν πρόβλημα, δεν δυσκολεύτηκαν και εντόπισαν ορθά και σταθερά και τις δύο περιπτώσεις.

Εικόνα 7.11: Η εικόνα ‘test11’ με αριθμημένες τις περιπτώσεις [55]

7.2 Οπτικοποίηση Αποτελεσμάτων

Πίνακας αποτελεσμάτων

Τα αποτελέσματα που αναλύθηκαν παραπάνω μπορούν να τοποθετηθούν σε έναν ενιαίο πίνακα, ώστε να γίνει πιο εύκολη και γρήγορη η ανάγνωσή τους. Στον πίνακα αυτόν, θα υπάρχει μία στήλη που θα αναγράφει τον συνολικό αύξοντα αριθμό της κάθε περίπτωσης, μία για τον αύξοντα αριθμό της κάθε περίπτωσης ανά εικόνα, μία που περιέχει το όνομα της εικόνας που βρίσκεται, μία με το όνομα της πραγματικής κλάσης και μία για τον εντοπισμό ή μη του κάθε μοντέλου. Αν το μοντέλο δεν εντόπισε κάποιο αντικείμενο, τότε τοποθετείται μία παύλα (-) στο αντίστοιχο κελί. Για την περαιτέρω διευκόλυνση στην ανάγνωσή του, η κλάση 'With Helmet' έχει μπλε χρώμα, η κλάση 'Without Helmet' έχει πορτοκαλί χρώμα, ενώ για τους εντοπισμούς, οι σωστοί εντοπισμοί έχουν πράσινο χρώμα (σκούρο για 'Without Helmet' και ανοιχτό για 'With Helmet) και οι λάθος εντοπισμοί έχουν κόκκινο χρώμα.

Πίνακας 7.1: Πίνακας με τα αποτελέσματα των δοκιμών των δύο μοντέλων

α/α	**α/α ανά εικόνα**	**Όνομα εικόνας**	**Πραγματική κλάση**	**SSD-MobileNetV2**	**SSD-MobileNetV2-FPNlite**
1	1	test1	With Helmet	With Helmet	With Helmet
2	2	test1	Without Helmet	Without Helmet	Without Helmet
3	1	test2	Without Helmet	Without Helmet	Without Helmet
4	2	test2	Without Helmet	With Helmet	-
5	3	test2	Without Helmet	Without Helmet	-
6	4	test2	Without Helmet	Without Helmet	Without Helmet
7	5	test2	Without Helmet	-	-
8	6	test2	Without Helmet	-	-
9	1	test3	With Helmet	With Helmet	Without Helmet
10	2	test3	With Helmet	With Helmet	With Helmet
11	3	test3	With Helmet	With Helmet	With Helmet
12	4	test3	With Helmet	-	Without Helmet
13	5	test3	With Helmet	-	-
14	1	test4	Without Helmet	Without Helmet	Without Helmet
15	2	test4	Without Helmet	Without Helmet	Without Helmet
16	1	test5	With Helmet	With Helmet	With Helmet

17	2	test5	With Helmet	Without Helmet	With Helmet
18	3	test5	With Helmet	Without Helmet	With Helmet
19	4	test5	With Helmet	-	-
20	1	test6	With Helmet	With Helmet	With Helmet
21	2	test6	With Helmet	With Helmet	With Helmet
22	1	test7	Without Helmet	Without Helmet	Without Helmet
23	2	test7	With Helmet	-	-
24	3	test7	With Helmet	With Helmet	Without Helmet
25	4	test7	With Helmet	-	-
26	5	test7	With Helmet	With Helmet	Without Helmet
27	6	test7	Without Helmet	Without Helmet	Without Helmet
28	7	test7	Without Helmet	Without Helmet	Without Helmet
29	1	test8	Without Helmet	Without Helmet	Without Helmet
30	2	test8	Without Helmet	Without Helmet	Without Helmet
31	3	test8	Without Helmet	-	-
32	1	test9	With Helmet	With Helmet	With Helmet
33	2	test9	Without Helmet	-	-
34	3	test9	With Helmet	With Helmet	With Helmet
35	4	test9	Without Helmet	With Helmet	Without Helmet
36	5	test9	Without Helmet	Without Helmet	-
37	6	test9	Without Helmet	Without Helmet	Without Helmet
38	7	test9	Without Helmet	Without Helmet	Without Helmet
39	8	test9	With Helmet	-	-
40	9	test9	Without Helmet	-	-
41	10	test9	With Helmet	With Helmet	Without Helmet
42	1	test10	With Helmet	-	-
43	2	test10	With Helmet	Without Helmet	Without Helmet
44	3	test10	With Helmet	Without Helmet	Without Helmet
45	1	test11	Without Helmet	Without Helmet	Without Helmet
46	2	test11	Without Helmet	Without Helmet	Without Helmet

Πίνακες Σύγχυσης

Η δημιουργία πινάκων σύγχυσης είναι ένας πολύ καλός τρόπος για να οπτικοποιηθούν οι επιδόσεις των δύο μοντέλων. Στον πίνακα αυτό, καταγράφονται οι φορές που βρέθηκαν η κλάσεις σωστά και εκείνες που βρέθηκαν λάθος. Αναλυτικά, έχοντας δύο συνολικά κλάσεις, την κλάση 1 ή 'With Helmet' και την κλάση 2 ή 'Without Helmet', δημιουργείται ένας πίνακας με διαστάσεις 3x3, όπου:

- Στο κελί που βρίσκεται στην πρώτη γραμμή και στην πρώτη στήλη, τοποθετείται ο αριθμός των περιπτώσεων που το μοντέλο εντόπισε την κλάση 1 σωστά.
- Στο κελί που βρίσκεται στην δεύτερη γραμμή και στην δεύτερη στήλη, τοποθετείται ο αριθμός των περιπτώσεων που το μοντέλο εντόπισε την κλάση 2 σωστά.
- Στο κελί που βρίσκεται στην πρώτη γραμμή και στην δεύτερη στήλη, τοποθετείται ο αριθμός των περιπτώσεων που το μοντέλο εντόπισε την κλάση 1, ενώ η σωστή ήταν η κλάση 2.
- Στο κελί που βρίσκεται στην δεύτερη γραμμή και στην πρώτη στήλη, τοποθετείται ο αριθμός των περιπτώσεων που το μοντέλο εντόπισε την κλάση 2, ενώ η σωστή ήταν η κλάση 1.
- Στο κελί που βρίσκεται στην τρίτη γραμμή και στην τρίτη στήλη, τοποθετείται το σύνολο των σωστών προβλέψεων διά το σύνολο των προβλέψεων (accuracy).
- Στο κελί που βρίσκεται στην πρώτη γραμμή και στην τρίτη στήλη, τοποθετείται το σύνολο των περιπτώσεων που το μοντέλο προέβλεψε την κλάση 1, μαζί με το ποσοστό επιτυχίας της.
- Στο κελί που βρίσκεται στην δεύτερη γραμμή και στην τρίτη στήλη, τοποθετείται το σύνολο των περιπτώσεων που το μοντέλο προέβλεψε την κλάση 2, μαζί με το ποσοστό επιτυχίας της.
- Στο κελί που βρίσκεται στην τρίτη γραμμή και στην πρώτη στήλη, τοποθετείται το σύνολο των περιπτώσεων που είχε η κλάση 1, μαζί με το ποσοστό σωστής πρόβλεψής της.
- Στο κελί που βρίσκεται στην τρίτη γραμμή και στην δεύτερη στήλη, τοποθετείται το σύνολο των περιπτώσεων που είχε η κλάση 2, μαζί με το ποσοστό σωστής πρόβλεψής της.

Με αυτούς τους κανόνες υπόψη, δημιουργήθηκαν οι δύο πίνακες σύγχυσης, οι οποίοι εμφανίζονται παρακάτω. Για τον σχεδιασμό τους, χρησιμοποιήθηκε το Confusion Matrix Generator [56].

SSD-MobileNetV2 (45 FPS)			
TARGET / OUTPUT	With Helmet	Without Helmet	SUM
With Helmet	12 35.29%	2 5.88%	14 85.71% 14.29%
Without Helmet	4 11.76%	16 47.06%	20 80.00% 20.00%
SUM	16 75.00% 25.00%	18 88.89% 11.11%	28 / 34 82.35% 17.65%

Πίνακας 7.2: Ο πίνακας σύγχυσης του SSD-MobileNetV2

SSD-MobileNetV2-FPNlite (10 FPS)			
TARGET / OUTPUT	With Helmet	Without Helmet	SUM
With Helmet	10 31.25%	0 0.00%	10 100.00% 0.00%
Without Helmet	7 21.88%	15 46.88%	22 68.18% 31.82%
SUM	17 58.82% 41.18%	15 100.00% 0.00%	25 / 32 78.13% 21.88%

Πίνακας 7.3: Ο πίνακας σύγχυσης του SSD-MobileNetV2-FPNlite

Με τα αποτελέσματα των δοκιμών σε μορφή πινάκων σύγχυσης, γίνεται πολύ εύκολος ο υπολογισμός του mAP (mean average precision) για τα δύο μοντέλα. Συγκεκριμένα, το mAP για το καθένα, προκύπτει μέσω της εξίσωσης $mAP = \frac{1}{n}\sum_1^n \frac{TP}{TP+FP}$ (11) όπου, n ο αριθμός των κλάσεων, TP οι σωστοί εντοπισμοί της κάθε κλάσης (true positive) και FP οι λάθος εντοπισμοί της κλάσης (false positive). Όπως αναφέρθηκε παραπάνω, αυτά τα δεδομένα υπάρχουν ήδη στους πίνακες που δείχνουν οι εικόνες 7.12 και 7.13, οπότε τοποθετώντας τα νούμερα αυτά στην (11):

SSD-MobileNetV2: $mAP = \frac{1}{n}\sum_1^n \frac{TP}{TP+FP} = \frac{1}{2}\cdot\left(\frac{12}{12+2}+\frac{16}{16+4}\right) = 0{,}8286$

SSD-MobileNetV2-FPNlite: $mAP = \frac{1}{n}\sum_1^n \frac{TP}{TP+FP} = \frac{1}{2}\cdot\left(\frac{10}{10+0}+\frac{15}{15+7}\right) = 0{,}8409$

Πλέον, υπάρχει μία αρκετά καθαρή εικόνα για τα σημεία που πλεονεκτούν και μειονεκτούν τα δύο μοντέλα. Πώς θα ήταν όμως η επίδοσή τους σε αυτές τις δοκιμές, αν είχαν εκπαιδευτεί με το αρχικό σύνολο δεδομένων, που δεν είχε υποστεί ακόμα κάποια από τις αλλαγές που συζητήθηκαν στην υποενότητα 4.1; Μετά από τις δοκιμές στις ίδιες εικόνες, προκύπτει ότι το πρώτο μοντέλο που εκπαιδεύτηκε (SSD-MobileNetV2-FPNlite με το πρώτο dataset), ενώ δεν είναι κακό στην κατηγοριοποίηση των αντικειμένων, είναι πολύ συντηρητικό στις προβλέψεις του. Όπως φαίνεται και στον παρακάτω πίνακα σύγχυσης, εντόπισε μόλις 20 από τις 46 περιπτώσεις, αλλά με αρκετά μεγάλη ακρίβεια. Αυτό ήταν προφανές κυρίως στις εικόνες ‘test2’, ‘test7’, ‘test8’ και ‘test9’, όπου εντόπισε μόλις τα 5 από τα 23 αντικείμενα. Ο λόγος που συμβαίνει αυτό είναι, ότι στο αρχικό σύνολο δεδομένων, υπήρχε αρκετά μεγάλος αριθμός εικόνων, στις οποίες όμως είχαν σχεδιαστεί κουτιά μόνο στα πολύ κοντινά αντικείμενα. Όπως αναγράφεται και στην υποενότητα 4.1, εκτός από τον τεράστιο αριθμό πολλαπλών όμοιων εγγραφών, υπήρχαν και πολλά λάθη στον σχεδιασμό των bounding boxes. Η μεγαλύτερη ακρίβεια δεν μπορεί να θεωρηθεί και καλύτερη, καθώς οι προβλέψεις που έκανε, ήταν μόνο στις εύκολες περιπτώσεις που και τα καινούργια μοντέλα εντόπισαν εύκολα. Επίσης, σε αντίθεση με τα καινούργια, παρουσιάζει μία τάση να κατηγοριοποιεί αντικείμενα ως κράνη, κάτι που είναι ανεπιθύμητο, καθώς πολλά άτομα που δεν θα χρησιμοποιούν κράνος δεν θα λαμβάνουν το αντίστοιχο μήνυμα ανατροφοδότησης. Συνεπώς, οι αλλαγές που παρουσιάστηκαν στην υποενότητα αυτή, φαίνεται να βελτίωσαν σημαντικά την απόδοση των τελικών μοντέλων, πετυχαίνοντας περισσότερες προβλέψεις, χωρίς να υπάρχει ουσιαστική μείωση στην ακρίβειά τους.

1st Model			
TARGET / OUTPUT	With Helmet	Without Helmet	SUM
With Helmet	10 50.00%	2 10.00%	12 83.33% 16.67%
Without Helmet	1 5.00%	7 35.00%	8 87.50% 12.50%
SUM	11 90.91% 9.09%	9 77.78% 22.22%	17 / 20 85.00% 15.00%

Πίνακας 7.4: Ο πίνακας σύγχυσης του πρώτου μοντέλου

Βάσει των τιμών του παραπάνω πίνακα και μέσω της (11), υπολογίζεται το αντίστοιχο mAP του πρώτου μοντέλου: $mAP = \frac{1}{n}\sum_1^n \frac{TP}{TP+FP} = \frac{1}{2} \cdot \left(\frac{10}{10+2} + \frac{7}{7+1}\right) = 0{,}8542$. Σε γενικές γραμμές το mAP δεν μπορεί να χρησιμοποιηθεί από μόνο του για την σύγκριση των μοντέλων, καθώς δεν λαμβάνει υπόψη του τις περιπτώσεις που δεν εντοπίστηκαν τα αντικείμενα. Οι πίνακες σύγχυσης αναδεικνύουν την απόδοση του κάθε μοντέλου για κάθε κλάση ξεχωριστά, δίνοντας έτσι μία πιο ολοκληρωμένη εικόνα.

8 ΚΕΦΑΛΑΙΟ 8ο : Ενεργειακή Αναφορά

Ένα μεγάλο ζήτημα για όλα τα ενσωματωμένα συστήματα, είναι η κατανάλωση ενέργειας και ο συνολικός χρόνος αυτονομίας τους. Σε γενικές γραμμές, το σύστημα αυτό είναι δυνατόν να λειτουργεί με σταθερή τροφοδοσία από οικιακές πρίζες (230 V/ 50 Hz), ωστόσο υπάρχει και η δυνατότητα να γίνει φορητό, με την τροφοδοσία από τις κατάλληλες μπαταρίες. Σε αυτό το κεφάλαιο, θα αναλυθεί η κατανάλωση ενέργειας του συστήματος σε όλες τις πιθανές καταστάσεις λειτουργίας του, δηλαδή σε αδράνεια (idle) και σε κατάσταση λειτουργίας μοντέλου object detection. Έπειτα, θα συζητηθούν οι επιλογές τροφοδοσίας, είτε μέσω οικιακής πρίζας, είτε μέσω μπαταρίας.

8.1 Κατανάλωση Ενέργειας

Υπάρχουν τρεις βασικές συσκευές που καταναλώνουν ενέργεια. Η πρώτη και πιο σημαντική, είναι το Google Coral Dev Board, το οποίο εκτελεί το object detector, εκπέμπει τα λογικά σήματα των κλάσεων που εντοπίζονται και παρέχει και τα δεδομένα για την ζωντανή μετάδοση της εικόνας από την κάμερα στην οθόνη, μέσω του HDMI. Η δεύτερη συσκευή, είναι η οθόνη που εμφανίζει τα αποτελέσματα του μοντέλου σε πραγματικό χρόνο και η τρίτη είναι το σύστημα ανατροφοδότησης (μικροελεγκτής, TFT LCD και piezo buzzer). Οι μετρήσεις έγιναν μέσω μίας συσκευής μέτρησης τάσης και έντασης ρεύματος, που χρησιμοποιεί θύρες USB-A, για λόγους διευκολυνσης.

Εικόνα 8.1: Η συσκευή USB-A για μέτρηση της τάσης και της έντασης ρεύματος [57]

Οι πρώτες μετρήσεις έγιναν για την κεντρική οθόνη (monitor) και είναι οι ακόλουθες:

- Τάση (Voltage): 4,96 V (Volt)
- Ένταση ρεύματος (current) (μέση τιμή): 0,73 A (Ampere)
- Ισχύς DC: 3,62 W (Watt)

Το Google Coral σε αδρανή λειτουργία έχει τις ακόλουθες μετρήσεις:

- Τάση (Voltage): 5,10 V (Volt)
- Ένταση ρεύματος (current) (μέση τιμή): 0,49 A (Ampere)
- Ένταση ρεύματος με τον ανεμιστήρα ψύξης (current) (μέση): 0,55 A (Ampere)
- Ισχύς DC: 2,499 W (Watt)
- Ισχύς DC με ανεμιστήρα ψύξης: 2,805 W (Watt)

Για τις μετρήσεις στο Google Coral, όταν λειτουργούν τα μοντέλα για object detection (χωρίς να λειτουργεί ο ανεμιστήρας), θα δημιουργηθεί ένας μικρός πίνακας για να γίνει πιο εύκολη η οργάνωση και η κατανόησή τους. Σε γενικές γραμμές, οι μικρές διαφορές που έχουν τα δύο μοντέλα μπορεί να οφείλονται στο εσωτερικό σφάλμα του οργάνου μέτρησης και στις μη ιδανικές συνθήκες μέτρησης.

Πίνακας 8.1: Πίνακας με την τάση, ένταση του ρεύματος και την ισχύς του κάθε μοντέλου

Μοντέλο	Τάση (V)	Ένταση ρεύματος (μέση τιμή) (A)	Ισχύς DC (W)
SSD-MobileNetV2-FPNlite	5,09	0,79	4,021
SSD-MobileNetV2	5,10	0,76	3,876

8.2 Επιλογή Τροφοδοσίας

Λαμβάνοντας πληροφορίες σχετικά με την απαιτούμενη τροφοδοσία της κάθε συσκευής, είτε από την ιστοσελίδα της καθεμίας, είτε από το αντίστοιχο documentation (αν παρέχεται), παρατηρείται ότι δεν είναι και πολύ ακριβείς. Συγκεκριμένα, όπως είχε αναφερθεί και στην υποενότητα 5.1, η Google προτείνει για την τροφοδοσία του τάση 5 V και μέγιστη ένταση ρεύματος 2-3 A [20], ωστόσο όπως μετρήθηκε, η συνολική ένταση δεν ξεπερνάει το 1 A. Επιπλέον, η οθόνη που συνδέεται μέσω καλωδίου HDMI έχει στο παρεχόμενο τροφοδοτικό τάση 5 V και μέγιστη ένταση ρεύματος 3 A, ενώ η ενεργειακή κατανάλωσή του είναι 7 kW/1000h (kilowatt/1000 ώρες)

[58], όμως σύμφωνα με τις μετρήσεις, η μέγιστη ένταση του ρεύματος (στις παρούσες ρυθμίσεις του) δεν ξεπερνάει τα 0,8 A. Με τις δύο αυτές συσκευές να απαιτούν τάση 5 V, υπάρχουν δύο μέθοδοι για την επαρκή και ασφαλή τροφοδοσία τους.

Η πρώτη είναι, μέσω τροφοδοτικών πρίζας από 220-230 V AC σε 5 V DC, με μέγιστη ένταση ρεύματος 2-3 A (μπορεί να χρησιμοποιηθεί και φορτιστής κινητού, αρκεί να πληροί τα παραπάνω χαρακτηριστικά). Αυτή η μέθοδος δεν μπορεί να χρησιμοποιηθεί αν το σύστημα πρέπει να είναι φορητό. Η δεύτερη είναι μέσω μπαταρίας. Μία εύκολη λύση είναι ένα powerbank με 2 θύρες USB-A, που λειτουργούν ως έξοδοι (outputs). Η τάση φυσικά πρέπει να είναι 5 V, ενώ για την ένταση του ρεύματος, μία καλή μέγιστη τιμή είναι τα 3 A, καθώς σύμφωνα με τις παραπάνω μετρήσεις, οι δύο συσκευές δεν ξεπερνάνε ποτέ αθροιστικά αυτή την τιμή. Άρα οποιαδήποτε μπαταρία πληροί τις παραπάνω συνθήκες, μπορεί να λειτουργήσει ως φορητό τροφοδοτικό για την οθόνη και το Google Coral. Η χωρητικότητα της μπαταρίας παίζει μεγάλο ρόλο για τον συνολικό χρόνο αυτονομίας του. Για την διπλωματική εργασία, χρησιμοποιήθηκε powerbank με χωρητικότητα μπαταρίας 30000 mAh (μιλιαμπερώρια).

Όσον αφορά την τροφοδοσία του Arduino Uno R3 (μαζί με το TFT LCD και το piezo buzzer), υπάρχουν διάφορες λύσεις. Αρχικά, μπορεί να χρησιμοποιηθεί ένα καλώδιο USB-A σε USB-B για την σύνδεση με υπολογιστή. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη όταν προγραμματίζεται η πλακέτα στο προγραμματιστικό περιβάλλον Arduino IDE. Μπορεί επίσης να συνδεθεί και αυτό σε ένα powerbank με τάση 5 V, ωστόσο η απόδοσή του μπορεί να μην είναι και η καλύτερη όταν είναι υπερφορτωμένο. Η δεύτερη είναι η σύνδεση ενός εξωτερικού τροφοδοτικού, μέσω του barrel jack. Για το Uno R3, προτείνεται τάση 7-12 V και μέγιστη ένταση ρεύματος 1 A. Οι υπόλοιπες δύο επιλογές είναι το Vin pin με τάση 7-12 V και το 3V3/5V pin, ωστόσο δεν προτείνεται η χρήση αυτών των μεθόδων, αν η δεύτερη επιλογή είναι διαθέσιμη [59]. Για την υλοποίηση της διπλωματικής εργασίας, χρησιμοποιήθηκε το barrel jack με μία μπαταρία 9 V, μέσα σε θήκη με ενσωματωμένο διακόπτη, για να γίνεται εύκολα η ενεργοποίηση και απενεργοποίηση της τροφοδοσίας του Arduino.

9 ΣΥΜΠΕΡΑΣΜΑΤΑ

Γενικά σχόλια

Ο στόχος της διπλωματικής ήταν να δημιουργηθεί ένα ενσωματωμένο σύστημα για τον εντοπισμό της χρήσης ή μη χρήση κράνους, το οποίο θα μπορεί να τοποθετηθεί σε διάφορους χώρους στις πόλεις της Ελλάδος (μικροί δρόμοι, χώροι στάθμευσης κ.λπ.), για να προωθηθεί η οδική ασφάλεια και να μειωθούν οι θάνατοι λόγω τροχαίων ατυχημάτων. Εκπαιδεύτηκαν δύο μοντέλα με διαφορετικές αρχιτεκτονικές που προσφέρει το Tensorflow Object Detection API, το SSD-MobileNetV2 και το SSD-MobileNetV2-FPNlite, μέσω της μεθόδου Transfer Learning, στο προγραμματιστικό περιβάλλον Google Colab. Στην συνέχεια, έγινε η εφαρμογή τους πάνω στο Google Coral Dev Board, με την βοήθεια του επιταχυντή EdgeTPU, για να μπορέσει να γίνει ο εντοπισμός σε πραγματικό χρόνο. Τα λογικά σήματα των εντοπισμένων κλάσεων, μεταφέρονται σε ένα Arduino Uno R3, το οποίο τα επεξεργάζεται και οδηγεί μία TFT LCD και ένα παθητικό piezo buzzer, ώστε να παράγονται ηχητικά και οπτικά μηνύματα ανατροφοδότησης.

Κατά την διάρκεια της εκπαίδευσης των μοντέλων παρουσιάστηκαν διάφορα προβλήματα. Συγκεκριμένα, πολλές φορές υπήρχαν αποσυνδέσεις από την υπηρεσία του Google Colab, συνήθως μετά από 2 ή 3 ώρες εκπαίδευσης. Μερικές φορές, η επανασύνδεση γινόταν αρκετά γρήγορα ώστε να μπορέσει να ολοκληρωθεί η διαδικασία. Επίσης, η αποθήκευση ενός checkpoint με τα αντίστοιχα συναπτικά βάρη του, γινόταν κάθε 1000 βήματα, με το τελευταίο να δημιουργείται μόλις ολοκληρωθούν τα 40000 βήματα που είχαν οριστεί. Με τις τιμές των απωλειών να εμφανίζονται ανά 100 βήματα, παρατηρήθηκε ότι σε πολλές περιπτώσεις, τα checkpoints δεν είχαν τις χαμηλότερες δυνατές τιμές απωλειών. Αυτό λύθηκε με την επιλογή του βέλτιστου checkpoint από τα 38000 έως και τα 40000 βήματα, καθώς σε αυτές τις τιμές γινόταν η σταθεροποίηση του παραθύρου τιμών που κυμαίνονταν οι απώλειες.

Με την δοκιμή του κάθε μοντέλου στο Google Colab να γίνεται σε ένα υποσύνολο που δημιουργείται τυχαία κάθε φορά, ήταν φανερό ότι οι επίσημες δοκιμές έπρεπε να γίνουν σε ένα καινούργιο σύνολο εικόνων. Ωστόσο, δεν ήταν κακή η ιδέα να γίνονται και κάποιες πρώτες εκτιμήσεις. Αρκετές φορές μάλιστα, ήταν εύκολα φανερό (εμπειρικά) ποια μοντέλα είχαν καλές επιδόσεις και ποια όχι, κάτι που βοήθησε στην επιλογή των τελευταίων δύο. Η διαδικασία του quantization, όπως και η μετατροπή τους σε μοντέλα συμβατά με την EdgeTPU, δεν παρουσιάσανε κάποια δυσκολία. Περαιτέρω σχόλια για τα δύο μοντέλα γίνονται αναλυτικά παρακάτω.

Η επικοινωνία με το Google Coral αρχικά ήταν πολύ δύσκολη. Ενώ η εγκατάσταση του λειτουργικού έγινε με επιτυχία και χωρίς προβλήματα, η συσκευή δεν μπορούσε να αναγνωριστεί από το Mendell Development Tool (σύνδεση με καλώδιο USB-C). Ευτυχώς, υπήρχε η δυνατότητα σειριακής επικοινωνίας, ωστόσο η μεταφορά αρχείων έγινε αρκετά πιο χρονοβόρα (μεταφορά μέσω microSD). Η εκτέλεση του inference ήταν αρκετά εύκολη και ο κώδικας που παρείχε το Pycoral API,

ήταν ικανοποιητικός. Οι αλλαγές που έγιναν σε αυτόν ήταν επίσης εύκολες, καθώς ο κώδικας ήταν πολύ καλογραμμένος.

Η μεταφορά των σημάτων έγινε και αυτή εύκολα και η διαφορά στην τάση που χρησιμοποιούν οι δύο πλακέτες (Google Coral – 3,3 V, Arduino Uno R3 - 5V) για τις λογικές πράξεις τους, δεν επηρέασε αρνητικά το σύστημα. Το μόνο πρόβλημα που προέκυψε κατά την δημιουργία του συστήματος ανατροφοδότησης, ήταν η αργή φόρτωση των γραφικών στο LCD. Αρχικά, γινόταν αποκλειστική χρήση bitmap (συνάρτηση drawBitmap) και αυτό ακριβώς ήταν το πρόβλημα. Παρατηρήθηκε ότι η διαδικασία που ακολουθείται προκειμένου να χαραχθεί ένα bitmap στην οθόνη, ήταν αρκετά χρονοβόρα λόγω της ανάγνωσης των δεδομένων από την μνήμη flash και την πραγματοποίηση πολλαπλών ελέγχων για το κάθε pixel ξεχωριστά. Έτσι, μέσω της χρήσης συναρτήσεων που χαράζουν σχήματα απευθείας και με τον περιορισμό των bitmaps ωστέ να είναι αρκετά μικρά, μπόρεσε να ελαττωθεί ο συνολικός χρόνος φόρτωσης από τα ~3 s, στα ~200 ms. Φυσικά, με την χρήση ενός πιο γρήγορου μικροελεγκτή, ο χρόνος αυτός γίνεται εύκολα πολύ μικρότερος. Τέλος, τα ηχητικά σήματα έγιναν και αυτά πολύ εύκολα, όμως λόγω έλλειψης διαθέσιμου pin με 5 V στο Arduino, η τροφοδοσία έμεινε στα 3,3 V. Έγινε μία δοκιμή να χρησιμοποιηθεί ένα 5 V pin του Google Coral, όμως κατά την εκτέλεση οποιουδήποτε μοντέλου, εμφανίζονταν διάφορα προβλήματα στους ήχους που δημιουργούσε το buzzer.

Συνολικά, το σύστημα αυτό μπορεί να βελτιωθεί πολύ. Αρχικά μπορούν να τοποθετηθούν φωτοβολταϊκά πάνελ, ώστε να γίνει ενεργειακά αυτόνομο. Στην συνέχεια, μπορεί να αξιοποιηθεί ένας πιο γρήγορος και πιο αποτελεσματικός μικροελεγκτής, όπως ένας της οικογενείας stm32. Έτσι, θα δίνεται η δυνατότητα για πιο γρήγορη επεξεργασία των σημάτων των κλάσεων, καθώς και πιο γρήγορης φόρτωσης πολύπλοκων γραφικών στην LCD. Επιπλέον, θα ήταν καλό να βρεθεί μία λύση για την μικρή απόσταση εντοπισμού των μοντέλων, όπως και για την αδυναμία τους σε χαμηλά επίπεδα φωτεινότητας. Το τελευταίο είναι ιδιαίτερα σημαντικό, γιατί θα μπορεί να λειτουργεί και τις βραδινές ώρες. Άλλη μία ιδέα είναι να γίνει η εφαρμογή κάποιου μοντέλου (π.χ. κάποιο μοντέλο YOLO) στο NVIDIA Jetson Nano, το οποίο είναι και αυτό μία εξαιρετική πλατφόρμα. Παρακάτω θα αναλυθούν τα συμπεράσματα για τα δύο μοντέλα που δοκιμάστηκαν.

Συμπεράσματα για το SSD-MobileNetV2-FPNlite

Το μοντέλο αυτό είχε μέτρια προς καλά αποτελέσματα από τις δοκιμές (πρόχειρες και επίσημες). Φαίνεται να λειτουργεί αρκετά καλά σε συνθήκες με καλό φωτισμό, σε σχετικά κοντινές αποστάσεις, δηλαδή μέχρι τα 6 μέτρα περίπου και σε περιβάλλον που κάνει εύκολο τον διαχωρισμό των αντικειμένων. Είναι αρκετά συντηρητικό ως προς τις προβλέψεις του και παρουσιάζει μία τάση προς την κατηγοριοποίηση των εντοπισμένων αντικειμένων ως ‘Without Helmet’, όπως είναι και εμφανές από τον πίνακα σύγχυσης. Λόγω της πιο πολύπλοκης αρχιτεκτονικής του, που ενσωματώνει ένα FPN στο ήδη υπάρχων SSD-MobileNetV2, είναι πιο αργό κατά την εκτέλεσή του. Συγκεκριμένα, το framerate που πετυχαίνει το Google Coral μαζί με την EdgeTPU, κυμαίνεται ανάμεσα στα 9 και

τα 11 fps. Σύμφωνα με την αναφορά του edgetpu compiler, από τις 160 διαδικασίες του μοντέλου, οι 112 μπορούν να εκτελεστούν από την TPU, ενώ οι υπόλοιπες παραμένουν στην CPU. Με την ταχύτητα αυτή, ο εντοπισμός των αντικειμένων θα είναι πολύ δύσκολος, αν αυτά κινούνται με ταχύτητα μεγαλύτερη των 30 χιλιομέτρων/ώρα. Η αρχιτεκτονική αυτή είναι διαθέσιμη και για διαστάσεις εισόδου 640x640, όμως η ταχύτητα θα μειωθεί ακόμα περισσότερο.

Συμπεράσματα για το SSD-MobileNetV2

Το SSD-MobileNetV2 είχε λίγο καλύτερες επιδόσεις στις δοκιμές. Σίγουρα όχι τέλειες, αλλά αρκετά καλές υπό ορισμένες συνθήκες. Η πιο βασική από αυτές, είναι ο καλός φωτισμός, όπως και η σχετικά μικρή απόσταση των αντικειμένων που πρέπει να εντοπιστούν (μέχρι 5 μέτρα). Εμφανίζει και αυτό μία μικρή τάση προς την κατηγοριοποίηση των αντικειμένων ως 'Without Helmet'. Η τάση αυτή είναι πολύ πιθανό να οφείλεται στο σύνολο δεδομένων που εκπαιδεύτηκαν τα μοντέλα, ωστόσο η ύπαρξή της δεν είναι υποχρεωτικά κακή. Ο βασικός σκοπός του συστήματος είναι να εντοπίζονται τα άτομα που δεν χρησιμοποιούν κράνος, οπότε είναι προτιμότερο να γίνεται αυτό, παρά το αντίστροφο (να κατηγοριοποιούνται τα αντικείμενα ως 'With Helmet', ενώ είναι 'Without Helmet'). Λόγω των μεγαλύτερων διαστάσεων εισόδου (510x510), είναι λίγο καλύτερο στον εντοπισμό αντικειμένων από το FPNlite (320x320), ωστόσο δυσκολεύεται λίγο παραπάνω όταν αυξάνονται οι αποστάσεις. Η βέλτιστη απόσταση είναι μέχρι τα 5 μέτρα περίπου. Στην βασική του μορφή, το μοντέλο έχει διαστάσεις εισόδου 300x300 και πετυχαίνει πάνω από 110 fps στο Google Coral με την EdgeTPU. Σύμφωνα με το αρχείο του edgetpu compiler, από τις 102 διαδικασίες, οι 99 αποδίδονται στην TPU, αφήνοντας μόλις τρεις για την CPU. Αυτός είναι και ο κύριος λόγος, που η διαφορά της ταχύτητας μεταξύ των δύο μοντέλων είναι τόσο μεγάλη. Επειδή η κάμερα δεν μπορεί να ξεπεράσει τα 30 fps, δεν υπάρχει λόγος να λειτουργεί τόσο γρήγορα το object detector. Άρα υπάρχει ένα περιθώριο αύξησης των διαστάσεων εισόδου, ώστε ακόμη και με την μείωση της ταχύτητας, να μην επηρεάζεται αρνητικά το σύστημα. Με τις καινούργιες διαστάσεις (510x510), το SSD-MobileNetV2 έτρεχε στα 45 fps, που σημαίνει ότι μπορεί θεωρητικά να αυξηθεί και λίγο παραπάνω. Το βασικό συμπέρασμα σε αυτό το πείραμα είναι ότι, ενώ το μοντέλο με το δίκτυο FPNlite θα έπρεπε να είναι πιο αργό αλλά και πιο ακριβές, επειδή το απλό είχε σχεδόν όλες τις διαδικασίες του να εκτελούνται στην TPU, δημιούργησε ένα μεγάλο περιθώριο για να αυξηθούν οι διαστάσεις της εισόδου του, αυξάνοντας την ακρίβειά του, σε σημείο που έφτανε ή ξεπερνούσε αυτή του πιο πολύπλοκου, κρατώντας παράλληλα και το προβάδισμά του στην ταχύτητα εκτέλεσης του inference.

Σχόλια για την βελτίωση και των δύο μοντέλων

Μία βασική ρύθμιση που θα μπορούσε να αλλάξει για την βελτίωση της απόδοσης, είναι τα anchor boxes. Τα προεπιλεγμένα κουτιά δεν έχουν τα ιδανικά σχήματα για τα αντικείμενα ενδιαφέροντος. Το πρόβλημα με αυτή την προσέγγιση είναι ότι με την παραμόρφωση των εικόνων κατά την μετατροπή τους στις διαστάσεις εισόδου του μοντέλου, είναι πολύ δύσκολο να προσεγγιστούν τα τελικά σχήματα των αντικειμένων. Μάλιστα, σε μία προσπάθεια να εφαρμοστούν

anchor boxes που ήταν πιο κοντά στα πραγματικά (από [1, 0.5, 2.0] σε [1, 0.67 , 1.33]), το μοντέλο κατά την διάρκεια των πρόχειρων δοκιμών είχε πολύ χειρότερες επιδόσεις, πραγματοποιώντας προβλέψεις με μεγάλη τιμή σιγουριάς, σε αντικείμενα που δεν είχαν κάποια σχέση με την εμφάνιση ενός κράνους ή ενός ανθρώπινου προσώπου. Για να μπορέσει λοιπόν να εφαρμοστεί σωστά η μέθοδος αυτή, πρέπει όλες οι εικόνες του συνόλου για την εκπαίδευση, επαλήθευση και δοκιμή, να έχουν τις ίδιες διαστάσεις με τις ζητούμενες από την αρχιτεκτονική, κάτι που δυστυχώς δεν ήταν δυνατόν στην διπλωματική εργασία. Η εκπαίδευσή του με περισσότερα βήματα επίσης δεν το βελτίωσε. Οι συνολικές απώλειες μετά τα 38000 βήματα περίπου, κυμαίνονταν σε ένα παράθυρο τιμών που είχε ως ελάχιστη τιμή το 0,08 και μέγιστη το 0,11. Αυτό δεν άλλαζε μετά τα 40000 βήματα, που ήταν και ο αριθμός βημάτων που βρέθηκε ως ο βέλτιστος. Ένα άλλο πρόβλημα ήταν και το φαινόμενο του overfitting που γινόταν εμφανές μετά τα 45000 βήματα. Στις δοκιμές παρατηρήθηκε μεγάλη διαφορά στην επίδοση που είχε σε νέες εικόνες (μέτρια προς κακή) και στις εικόνες που είχε ήδη δει μέσω της εκπαίδευσης (σχεδόν άριστη). Το batch size ήταν επίσης πολύ δύσκολο να αυξηθεί σε τιμή μεγαλύτερη του 16, καθώς η απαιτούμενη μνήμη θα ήταν μεγαλύτερη από αυτή που ήταν διαθέσιμη και η διαδικασία της εκπαίδευσης, είτε δεν θα ήταν εφικτή, είτε θα είχε πολύ μεγαλύτερη διάρκεια, καταναλώνοντας περισσότερους υπολογιστικούς πόρους από τους διαθέσιμους (Google Colab Computing Units). Είναι φανερό μετά από αυτούς τους συλλογισμούς, ότι οι μόνη βελτίωση που θα μπορούσε να γίνει (στην διπλωματική εργασία), ήταν με την παροχή περισσότερων και ποιοτικότερων εικόνων/δεδομένων.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] Β. Ανδριανόπουλος. “Το κράνος θα είχε σώσει 105 μοτοσικλετιστές” kathimerini.gr. Accessed: Jul. 25, 2024. [Online]. Available: *https://www.kathimerini.gr/society/563089000/to-kranos-tha-eiche-sosei-105-motosikletistes/*

[2] Κ. Διαμαντάρας and Δ. Μπότσης, *ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ*, 1st ed. Αθήνα, Ελλάδα: Εκδόσεις Κλειδάριθμος, 2019.

[3] V. L. Bandara. “What is Machine Learning?” medium.com. Accessed: Aug. 31, 2024. [Online]. Available: https://vitiya99.medium.com/what-is-machine-learning-d747248ab95e

[4] “Artificial Neural Networks and its Applications” geeksforgeeks.org. Accessed: Aug. 31, 2024. [Online]. Available: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

[5] L. Dormehl. “What is an artificial neural network? Here’s everything you need to know” digitaltrends.com. Accessed: Aug. 31, 2024. [Online]. Available: https://www.digitaltrends.com/computing/what-is-an-artificial-neural-network/

[6] J. Cardete. “Convolutional Neural Networks: A Comprehensive Guide” medium.com. Accessed: Aug. 31, 2024. [Online]. Available: https://medium.com/thedeephub/convolutional-neural-networks-a-comprehensive-guide-5cc0b5eae175

[7] M. Yani, S. Irawan, and C. Setianingsih, "Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail," *Journal of Physics: Conference Series*, vol. 1201, p. 012052, May 2019, doi: 10.1088/1742-6596/1201/1/012052.

[8] S. Park. “A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN" medium.com. Accessed: Jul. 30, 2024. [Online]. Available: https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c

[9] P. Potrimba. “What is R-CNN?” roboflow.com. Accessed: Aug. 31, 2024. [Online]. Available: https://blog.roboflow.com/what-is-r-cnn/

[10] BuildYourAI. *Beginner's Guide to Object Detection With Convolutional Neural Networks*. (Apr. 6, 2022). Accessed: Aug. 1, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=FDTXFlrCeuk

[11] S. Lee. “Implementing Single Shot Detector (SSD) in Keras: Part I — Network Structure” medium.com. Accessed: Aug. 31, 2024. [Online]. Available: https://towardsdatascience.com/implementing-ssd-in-keras-part-i-network-structure-da3323f11cff

[12] K. Sambasivarao. “Non-maximum Suppression (NMS)” towardsdatascience.com. Accessed: Aug. 1, 2024. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

[13] T. Sivamani. “NMS: Non Maximum Suppression” medium.com. Accessed: Aug. 31, 2024. [Online]. Available: https://blog.cubed.run/nms-non-maximum-suppression-157be5bc61ca

[14] Arya K M and Ajith K K. A review on deep learning based helmet detection. Proceedings of the International Conference on Systems, Energy Environment (ICSEE) 2021, 2020.

[15] Hao Wu and Jinsong Zhao. An intelligent vision-based approach for helmet identification for work safety. Safety Science, 109:123–130, 2018.

[16] Wei Zhang et al. Worker’s helmet recognition and identity recognition based on computer vision. Scientific Research Publishing, 12:1–10, 2024.

[17] D. Ren, T. Sun, C. Yu and C. Zhou, "Research on Safety Helmet Detection for Construction Site," *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*, Kunming, China, 2021, pp. 186-189, doi: 10.1109/CISAI54367.2021.00042.

[18] M. I. Hossain, R. B. Muhib and A. Chakrabarty, "Identifying Bikers Without Helmets Using Deep Learning Models," *2021 Digital Image Computing: Techniques and Applications (DICTA)*, Gold Coast, Australia, 2021, pp. 01-08, doi: 10.1109/DICTA52665.2021.9647170.

[19] “Dev Board datasheet” coral.ai. Accessed: Mar. 14, 2024. [Online]. Available: https://coral.ai/docs/dev-board/datasheet/

[20] “Get started with the Dev Board” coral.ai. Accessed: Apr. 4, 2024. [Online]. Available: https://coral.ai/docs/dev-board/get-started#requirements

[21] “Τανυστής” wikipedia.org. Accessed: Aug. 2, 2024. [Online]. Available: https://el.wikipedia.org/wiki/%CE%A4%CE%B1%CE%BD%CF%85%CF%83%CF%84%CE%AE%CF%82

[22] “Tensorflow” wikipedia.org. Accessed: Aug. 2, 2024. [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow

[23] “Why Tensorflow” tensorflow.org. Accessed: Aug. 2, 2024 [Online]. Available: https://www.tensorflow.org/about

[24] “Tensorflow Lite” ai.google.dev. Accessed: Aug. 2, 2024 [Online]. Available: https://ai.google.dev/edge/litert

[25] C. Wang and Z. Xiao, "Lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation," *Agronomy*, vol. 11, p. 1500, Jul. 2021. doi: 10.3390/agronomy11081500.

[26] C. W. Chang, S. Santra, J. W. Hsieh, H. Pirdiansyah, and C. F. Lin, "Multi-fusion feature pyramid for real-time hand detection," *Multimedia Tools and Applications*, vol. 81, Apr. 2022. doi: 10.1007/s11042-021-11897-7.

[27] "Python" wikipedia.org. Accessed: Aug. 8, 2024 [Online]. Available: https://el.wikipedia.org/wiki/Python

[28] Google. *Google Colab*. (2024). Accessed: Mar. 15, 2024. [Online]. Available: https://colab.research.google.com/

[29] F. Oh. "What is CUDA?" blogs.nvidia.com. Accessed: Aug. 3, 2024. [Online]. Available: https://blogs.nvidia.com/blog/what-is-cuda-2/

[30] M. E. Ogbemi. "What is Overfitting in Machine Learning?" freecodecamp.org. Accessed: Aug. 31, 2024. [Online]. Available: https://www.freecodecamp.org/news/what-is-overfitting-machine-learning/

[31] Make ML, "Bikes Helmets Dataset" Make ML. [Online]. Available: https://makeml.app/datasets/helmets

[32] T. Lin. *LabelImg*. GitHub Repository. (2018). tzutalin. Accessed: Apr. 14, 2024. [Online]. Available: https://github.com/tzutalin/labelImg

[33] H. Yu, C. Chen, X. Du, Y. Li, A. Rashwan, L. Hou, P. Jin, F. F. Liu, J. Kim, and J. Li. *TensorFlow Model Garden*. GitHub Repository. (2020). Tensorflow. Accessed: Mar. 19, 2024. [Online]. Available: https://github.com/tensorflow/models

[34] "Splitting datasets and cross-validation" futurelearn.com. Accessed: Aug. 31, 2024. [Online]. Available: https://www.futurelearn.com/info/courses/machine-learning-for-image-data/0/steps/360400

[35] D. Tran. *generate_tfrecord.py*. GitHub Repository. (2017). datitran. Accessed: Mar. 20, 2024. [Online]. Available: https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py

[36] E. Juras. *TFLite_detection_image.py*. GitHub Repository. (2022). EdjeElectronics. Accessed: Mar. 20, 2024. [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/TFLite_detection_image.py

[37] "TensorFlow models on the Edge TPU" coral.ai. Accessed: Aug. 31, 2024. [Online]. Available: https://coral.ai/docs/edgetpu/models-intro/#compatibility-overview

[38] "Connect a camera to the Dev Board" coral.ai. Accessed: Apr. 6, 2024. [Online]. Available: https://coral.ai/docs/dev-board/camera/#connect-the-coral-camera

[39] "Connect to the Dev Board I/O pins" coral.ai. Accessed: Jul. 6, 2024. [Online]. Available: https://coral.ai/docs/dev-board/gpio/

[40] Arduino. *Arduino IDE*. (2024). Arduino LLC. Accessed: Jul. 8, 2024. [Online]. Available: https://www.arduino.cc/en/software

[41] "Serial Peripheral Interface" wikipedia.org. Accessed: Aug. 10, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

[42] J. van Loenen. *img2cpp*. GitHub Repository. (2024). javl. Accessed: Jul. 24, 2024. [Online]. Available: https://javl.github.io/image2cpp/

[43] "Piezoelectric speaker" wikipedia.org. Accessed: Aug. 14, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Piezoelectric_speaker

[44] "Passive Buzzer Module" grobotics.com. Accessed: Aug. 31, 2024. [Online]. Available: https://grobotronics.com/passive-buzzer-module.html?sl=en

[45] Β. Χαντζής. "Πώς βλέπεις τον δρόμο εσύ και πώς ένας μηχανόβιος (Pics)" menshouse.gr. Accessed: Aug. 15, 2024. [Online]. Available: https://menshouse.gr/extras/62536/pos-vlepis-ton-dromo-esy-ke-pos-enas-michanovios-pics

[46] Χ. Κάβουρας. "Επτά πράγματα που ζηλεύουμε στους μηχανόβιους" ratpack.gr. Accessed: Aug. 15, 2024. [Online]. Available: https://www.ratpack.gr/manual/auto-moto/story/93/epta-pragmata-poy-emeis-oi-fovismenoi-zileyoyme-stoys-mixanovioys

[47] "10 πράγματα που πρέπει να κάνεις ως μηχανόβιος, πριν πεθάνεις" mototriti.gr. Accessed: Aug. 15, 2024. [Online]. Available: https://www.mototriti.gr/data/news/preview_news/10-pragmata-poy-epiballetai-na-kaneis-prin-kremaseis-ta-dermata_129302.asp

[48] "The Bikeriders (2023) -IMDb" imdb.com. Accessed: Aug. 15, 2024. [Online]. Available: https://www.imdb.com/title/tt21454134/

[49] "Σαντορίνη: Μηχανόβιοι με κοστούμια και vintage μηχανές έκαναν πορεία! [εικόνες & βίντεο]" iefimerida.gr. Accessed: Aug. 15, 2024. [Online]. Available: https://www.iefimerida.gr/news/448235/santorini-mihanovioi-me-kostoymia-kai-vintage-mihanes-ekanan-poreia-eikones-vinteo

[50] "File:Motorcycle.riders.arp.jpg" wikimedia.org. Accessed: Aug. 15, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:Motorcycle.riders.arp.jpg

[51] "The Most Epic Taco Run 2018" yamahait.com.au. Accessed: Aug. 15, 2024. [Online]. Available: https://yamahait.com.au/the-most-epic-taco-run-2018/

[52] M. Burrows. "Younger Workers in Cities More Likely to Bike to Work" census.gov. Accessed: Aug. 15, 2024. [Online]. Available: https://www.census.gov/library/stories/2019/05/younger-workers-in-cities-more-likely-to-bike-to-work.html

[53] "Bikes lead the way through crisis" bicyclenetwork.com.au. Accessed: Aug. 15, 2024. [Online]. Available: https://bicyclenetwork.com.au/newsroom/2020/03/26/bikes-lead-the-way-through-crisis/

[54] BSI. "ISO Net Zero Guidelines case study report" BSI, London. Accessed: Aug. 15, 2024. [Online]. Available: https://www.bsigroup.com/siteassets/pdf/en/capabilities/bsi-iso-net-zero-report.pdf

[55] A. L. Ashkenaz. "Pop-up bike lanes: How Europe's pandemic cycling schemes paid off" thenewfederalist.eu. Accessed: Aug. 15, 2024. [Online]. Available: https://www.thenewfederalist.eu/pop-up-bike-lanes-how-europe-s-pandemic-cycling-schemes-paid-off

[56] D. Perri. *Confusion Matrix Generator*. GitHub Repository. (2024). DamianoP. Accessed: Aug. 16, 2024. [Online]. Available: https://www.damianoperri.it/public/confusionMatrix/

[57] "Sunshine USB Voltage & Current Meter SS-302A Tester για Service Κινητών" skroutz.gr. Accessed: Aug. 31, 2024. [Online]. Available: https://www.skroutz.gr/s/22001813/Sunshine-USB-Voltage-Current-Meter-SS-302A-Tester-gia-Service-Kiniton.html

[58] "Portable Monitor 14" Full HD 1080p – PM14" verbatim-europe.com. Accessed: Apr. 12, 2024. [Online]. Available: https://www.verbatim-europe.com/en/portable-monitors/products/portable-monitor-14-full-hd-1080p-pm14-49590

[59] J. Bagur, T. Chung and K. Söderby. "Powering Alternatives for Arduino Boards" docs.arduino.cc. Accessed: Aug. 20, 2024. [Online]. Available: https://docs.arduino.cc/learn/electronics/power-pins/

Παράρτημα Α - Κώδικας Google Colab

- Κώδικας για τον διαχωρισμό των δεδομένων και την αφαίρεση μη-έγκυρων εγγραφών

```
import os
import random
import shutil
import xml.etree.ElementTree as ET

data_folder = '/content/data/all_data/data'
training_folder = '/content/data/train'
validation_folder = '/content/data/validation'
testing_folder = '/content/data/test'
corrupted_folder = '/content/data/corrupted'

# Create train, validation, and test folders if they don't already exist
for folder in [training_folder, validation_folder, testing_folder, corrupted_folder]:
    if not os.path.exists(folder):
        os.makedirs(folder)

#Function to check xml files for incorrect bounding boxes
def check_xml_files(target_folder):
    corrupted_xml = []
    data_paths = os.listdir(target_folder)

    for file in data_paths:
        if file.endswith('.xml'):
            xml_file_path = os.path.join(target_folder, file)

            tree = ET.parse(xml_file_path)
            root = tree.getroot()

            width = int(root.find('size/width').text)
            height = int(root.find('size/height').text)

            for obj in root.findall('object'):
                xmin = int(obj.find('bndbox/xmin').text)
                ymin = int(obj.find('bndbox/ymin').text)
                xmax = int(obj.find('bndbox/xmax').text)
                ymax = int(obj.find('bndbox/ymax').text)

                if (xmin < 0 or ymin < 0 or xmax > width or ymax > height):
                    print(f"Bounding box coordinates out of image bounds in file
'{file}'.")
                    corrupted_xml.append(file)
                    break

    return corrupted_xml


#Function to move xml files to folders where their image file resides
def move_xml_files(image_files, folder):
```

```
    for image_file in image_files:
        xml_file = image_file.replace('.png', '.xml') #swap .png with the corresponding
image file type
        if os.path.exists(os.path.join(data_folder, xml_file)):
            shutil.move(os.path.join(data_folder, xml_file), folder)

corrupted_xml = check_xml_files(data_folder)

for file in corrupted_xml:
    image_file = file.replace('.xml', '.png')
    xml_file_path = os.path.join(data_folder, file)
    image_file_path = os.path.join(data_folder, image_file)
    shutil.move(xml_file_path, os.path.join(corrupted_folder, file))
    shutil.move(image_file_path, os.path.join(corrupted_folder, image_file))

# List all remaining files in the data folder
image_files = []
files = os.listdir(data_folder)
for file in files:
    if file.endswith('.png'):
        image_files.append(os.path.join(data_folder, file))

# Shuffle the images randomly
random.shuffle(image_files)

# Split the data into train, validation, and test sets (80-10-10 split)
num_files = len(image_files)
num_train = int(0.8 * num_files)
num_val = int(0.1 * num_files)

#Training, Validation and Testing folder creation
train_files = image_files[:num_train]
val_files = image_files[num_train:num_train + num_val]
test_files = image_files[num_train + num_val:]


for file in train_files:
    shutil.move(os.path.join(data_folder, file), training_folder)

for file in val_files:
    shutil.move(os.path.join(data_folder, file), validation_folder)

for file in test_files:
    shutil.move(os.path.join(data_folder, file), testing_folder)

move_xml_files(train_files, training_folder)
move_xml_files(val_files, validation_folder)
move_xml_files(test_files, testing_folder)

print("The dataset has been split")
print(f"Total images: {num_files}")
print(f"Training images: {len(train_files)}")
```

```
print(f"Validation images: {len(val_files)}")
print(f"Testing images: {len(test_files)}")
print(f"Corrupted: {len(corrupted_xml)}")
```

- Κώδικας για την μετατροπή των αρχείων xml σε csv (συμβατό με tensorflow).

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET

def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for child in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     child[0].text,
                     int(child[4][0].text),
                     int(child[4][1].text),
                     int(child[4][2].text),
                     int(child[4][3].text)
                     )
            xml_list.append(value)
    column_names = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax',
'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_names)
    return xml_df

def main():
    for folder in ['train','validation', 'test']:
        image_path = os.path.join(os.getcwd(), ('data/' + folder))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('data/' + folder + '_labels.csv'), index=None)
        print('CSV file has been successfully created!')
main()
```

Παράρτημα Β - Κώδικας Arduino

- Οι συναρτήσεις drawHelmet, positiveSound και negativeSound.

```
void drawHelmet(int color)
{
  //color -> hex color value
  tft.drawRoundRect(45, 112, 230, 256, 125, color); //the helmet base

  tft.drawRect(45, 220, 230, 5, color); //the two top lines from the helmet visor
```

```
  tft.drawLine(50, 285, 268, 285, color); //the top line from the bottom of the helmet
visor
  tft.drawLine(53, 290, 266, 290, color); //the bottom line from the bottom of the
helmet visor

  tft.drawLine(90, 310, 110, 340, color); //first under-visor texture line
  tft.drawLine(120, 310, 130, 340, color); //second under-visor texture line
  tft.drawLine(150, 310, 150, 340, color); //third under-visor texture line
  tft.drawLine(170, 310, 170, 340, color); //fourth under-visor texture line
  tft.drawLine(200, 310, 190, 340, color); //fifth under-visor texture line
  tft.drawLine(230, 310, 210, 340, color); //sixth under-visor texture line
}

void positiveSound()
{
  tone(4, 523);
  delay(100);
  tone(4, 659);
  delay(100);
  tone(4, 784);
  delay(100);
  noTone(4);
}

void negativeSound()
{
  tone(4, 1000);
  delay(100);
  noTone(4);
  delay(100);
  tone(4, 1000);
  delay(100);
  noTone(4);
  delay(100);
  tone(4, 1000);
  delay(100);
  noTone(4);
}
```

- Ο κώδικας που επαναλαμβάνεται (void loop).

```
/void loop()
{

  int class2_signal = digitalRead(CLASS2_PIN); //without helmet

  if ((class2_signal == HIGH) && (!class2_state))
  {
    delay(100); //around 4 inferences later
    if (class2_signal == HIGH)
    {
      tft.fillScreen(0x0000);
      tft.drawBitmap(123, 148, neg_sign, 75, 50, 0xF800); //negative feedback
```

```
      drawHelmet(0xF800);
      tft.drawBitmap(10, 45, negative_text, 300, 22, 0xF800);
      class2_state = true;
      class1_state = false;
      negativeSound();
      return;
    }
  }
  int class1_signal = digitalRead(CLASS1_PIN); //with helmet
  if ((class1_signal == HIGH) && (!class1_state))
  {
    delay(100); //around 4 inferences later
    if (class1_signal == HIGH)
    {
      tft.fillScreen(0x0000);
      tft.drawBitmap(123, 148, pos_sign, 75, 50, 0x07E0);
      drawHelmet(0x07E0);
      tft.drawBitmap(10, 44, positive_text, 300, 25, 0x07E0);
      class1_state = true;
      class2_state = false;
      positiveSound();
      return;
    }
  }
  else
  {
    delay(100); //around 4 inferences later
    if ((class1_signal == LOW) && (!class2_signal))
    {
      tft.fillScreen(0x0000);
      noTone(4);
      class1_state = false;
      class2_state = false;
      return;
    }
  }
}
```