ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ - ΜΗΧΑΝΙΚΩΝ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

«Τεχνικές Βαθιάς Μάθησης για την ανίχνευση και χαρτογράφηση των διαβάσεων πεζών σε επίπεδο πόλης»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλεξοπούλου Χρυσοβαλάντα

Επιβλέπουσα : Καραθανάση Βασιλεία

Καθηγήτρια ΕΜΠ

Αθήνα, Μάρτιος 2023

NATIONAL TECHNICAL UNIVARSITY OF ATHENS

SCHOOL OF RURAL AND SURVEYING ENGINEERING

- GEOINFORMATICS ENGINEERING

«Deep Learning Techniques for detecting and mapping pedestrian crossings at city level»

Diploma Thesis

Alexopoulou Chrysovalanta

Athens, March 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ - ΜΗΧΑΝΙΚΩΝ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

«Τεχνικές Βαθιάς Μάθησης για την ανίχνευση και χαρτογράφηση των διαβάσεων πεζών σε επίπεδο πόλης»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλεξοπούλου Χρυσοβαλάντα

Επιβλέπουσα : Καραθανάση Βασιλεία

Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή επιτροπή τον Μάρτιο 2023

……	……	……
Καραθανάση Βασιλεία	Δουλάμης Αναστάσιος	Κεπαπτσόγλου Κωνσταντίνος
Καθηγήτρια ΕΜΠ	Αναπλ. Καθηγητής ΕΜΠ	Αναπλ. Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2023

……..

Αλεξοπούλου Χρυσοβαλάντα
Αγρονόμος και Τοπογράφος Μηχανικός – Μηχανικός Γεωπληροφορικής

Ευχαριστίες

Ευχαριστώ πάρα πολύ την καθηγήτριά μου Καραθανάση Βασιλεία για την εμπιστοσύνη της σε μένα, την υποστήριξη, την καθοδήγηση και τις επιστημονικές υποδείξεις της σε όλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας. Μέσα από την ενθάρρυνση , την υπομονή και τη σκληρή δουλειά , φτάνουμε στην ολοκλήρωση των προπτυχιακών σπουδών μου.

Ευχαριστώ θερμά τον Δρ. Μηχανικό Πολυχρόνη Κολοκούση και την υποψήφια Δρ. Μηχανικό Βικτωρία Κριστολλάρη του εργαστηρίου τηλεπισκόπησης ΕΜΠ, για τη συνεργασία, την βοήθεια και τις χρήσιμες συμβουλές.

Ευχαριστώ όλο το διδακτικό προσωπικό που συνέβαλαν στην πορεία μου ως τώρα.

Τέλος ευχαριστώ την οικογένεια μου για την αγάπη και την στήριξη και τους φίλους μου για την κατανόηση τους όλο αυτό το διάστημα.

Περίληψη

Για την καλή λειτουργία μιας πόλης πρέπει οι άνθρωποι που αλληλοεπιδρούν σε αυτήν και με αυτήν να είναι χαρούμενοι. Έτσι δημιουργήθηκαν οι χώροι αναψυχής. Πρέπει η πόλη να είναι βιώσιμη, δηλαδή να παρέχει υψηλή ποιότητα ζωής. Αυτό σημαίνει την κάλυψη των αναγκών των ανθρώπων με τον καλύτερο δυνατό τρόπο. Με τους χώρους αναψυχής καλύπτονται οι ανάγκες της κοινωνικοποίησης, της ξεκούρασης και της χαλάρωσης. Αυτό οδηγεί σε ένα αίσθημα χαράς, που σε συνδυασμό με την κοινωνικοποίηση και αλληλοεπίδραση των ανθρώπων μεταξύ τους μέσω των χώρων αναψυχής, αυξάνει την παραγωγικότητα, την επικοινωνία και την συλλογικότητα σε μία πόλη. Αυτό με τη σειρά του συνεπάγεται σε οικονομική ύφεση και αύξηση της βιωσιμότητας. Είναι προφανής λοιπόν η σημαντικότητα των χώρων αναψυχής. Οι πολίτες πρέπει να έχουν πρόσβαση στους χώρους αυτούς, κάτι το οποίο είναι εφικτό μέσω των διαβάσεων πεζών, ώστε η πρόσβαση να γίνεται με ασφάλεια. Οι διαβάσεις ορίζουν τη νόμιμη δίοδο διέλευσης του δρόμου για τους πεζούς, καθώς και ειδοποιούν τους οδηγούς πως πλησιάζουν περιοχή διέλευσης πεζών και πρέπει να μειώσουν ταχύτητα. Η μελέτη, η ανίχνευση και η γνώση της τοποθεσίας τους είναι πολύ σημαντική για την επίτευξη μεγαλύτερης ασφάλειας στο δρόμο και για τους πεζούς και για τα οχήματα, που θα οδηγήσει σε μια πιο βιώσιμη πόλη με καλύτερο δίκτυο μεταφοράς και μετακίνησης.

Για αυτό η παρούσα Διπλωματική Εργασία αναφέρεται στην ανίχνευση και χαρτογράφηση διαβάσεων πεζών σε επίπεδο πόλης με χρήση τεχνικών βαθιάς μάθησης. Πιο συγκεκριμένα, εφαρμόζονται τρείς τεχνικές βαθιών συνελικτικών νευρωνικών δικτύων: τα ZF - Net και Resnet και ο αλγόριθμος YOLOV3. Πραγματοποιήθηκε συλλογή και επεξεργασία δορυφορικών απεικονίσεων και στη συνέχεια μετά τη αξιολόγηση των δικτύων επιλέχθηκε η καλύτερη τεχνική, με την οποία έπειτα χαρτογραφήθηκαν οι διαβάσεις. Η επιλογή βασίζεται στην ακρίβεια των μοντέλων ενώ τα αποτελέσματα επιβεβαιώνουν την τεχνική που επιλέγεται. Τα πειράματα και η προτεινόμενη μεθοδολογία εφαρμόστηκαν στις πόλεις του Μπρίστολ και του Μάντσεστερ στην Αγγλία.

Λέξεις κλειδιά : βαθιά μάθηση, τεχνικές βαθιάς μάθησης, διαβάσεις πεζών, ZF - Net, Resnet, YOLOV3, εκπαίδευση , ακρίβεια

Abstract

For a city to function well, the people who interact in it and with it must be happy. This is how the recreational areas were created. The city must be sustainable, i.e. provide a high quality of life. This means the city must meet people's needs in the best possible way. Recreational areas meet the needs of socialization, rest and relaxation. This leads to a feeling of joy, which combined with the socialization and interaction of people with each other through recreational spaces, increases productivity, communication and collegiality in a city. This in turn leads to economic recession and increased sustainability. The importance of recreational areas is therefore obvious. Citizens must have access to these spaces, which is possible through pedestrian crossings, so that access is done safely. Crosswalks define the legal way for pedestrians to cross the road, as well as alert drivers that they are approaching a crosswalk and need to slow down. Studying, detecting and knowing a crosswalk's location is very important to achieve greater road safety for both pedestrians and vehicles, which will lead to a more sustainable city with a better transport and mobility network.

This is why this Diploma Thesis refers to the detection and mapping of pedestrian This is why this Diploma Thesis refers to the detection and mapping of pedestrian crossings at city level using deep learning techniques. More specifically, three techniques are applied: ZF-Net, Resnet and the YOLOV3 algorithm. Satellite images were collected and processed and after the evaluation of the three convolutional networks the best technique was selected, with which the crossings were then mapped. The choice is based on the accuracy of the models while the results confirm the chosen technique. The methodology was developed and applied on satellite images depicting the cities of Bristol and Manchester in England.

Keywords : deep learning, deep learning techniques, pedestrian crossings, ZF-Net, Resnet, YOLOV3, training, accuracy

Πίνακας περιεχομένων

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Συντομογραφίες

fc → Fully connected layers

ReLU → Rectified Linear Units

FPN → Feature Pyramid Network

NMS → Non-Maximal Suppression

IoU → Intersection over Union

GPU → Graphics Processing Unit

CPU → Central Processing Unit

API → Application Programming Interface

OSM → Open Street Map

HOG → Histogram of Oriented Gradients

LBPH → Local Binary Pattern Histogram

SVM → Support Vector Machine

GLCM → Gray Level Co-occurrence Matrix

Κλπ. → και λοιπά

CNN → Convolutional Neural Network

Ann → Artificial neural network

RGB → Red Green Blue

Ea → Early stopping

Cp → Check point

YOLO → You Only Look Once

GPS → Global Positioning System

mAP → mean average precision

Λεξικό

Fully connected layers → Πλήρως συνδεδεμένα δίκτυα

Rectified Linear Units → Διορθωμένες Γραμμικές Μονάδες - συνάρτηση

Leaky Rectified Linear Units → Rectified Linear Units που δέχεται και αρνητικές τιμές

Graphics Processing Unit → Μονάδα Επεξεργασίας Γραφικών

Feature Pyramid Network → Χαρακτηριστικό Δίκτυο Πυραμίδας

Non-Maximal Suppression → Μη μέγιστη καταστολή

Intersection over Union → Τομή ανά ένωση

Google Static Maps API → Διεπαφή προγραμματισμού εφαρμογών των στατικών χαρτών της Google

Google Maps Direction API → Διεπαφή προγραμματισμού εφαρμογών των χαρτών κατευθύνσεως της Google

Gray Level Co-occurrence Matrix → Πίνακας συν-εμφάνισης επιπέδου γκρι

Support Vector Machine → Διανυσματική Μηχανή Υποστήριξης

Central Processing Unit → Κεντρική μονάδα επεξεργασίας

Convolutional Neural Network → Συνελικτικό Νευρωνικό Δίκτυο

Artificial neural network → Τεχνητά νευρωνικά δίκτυα

Early stopping → Σταμάτημα νωρίς

Check point → Σημείο ελέγχου

Histogram of Oriented Gradients → Ιστόγραμμα Προσανατολισμένων Διαβαθμίσεων

Local Binary Pattern Histogram → Τοπικό Ιστόγραμμα Δυαδικού Μοτίβου

Continental → Ευρωπαϊκή

Dashed → Διακεκομμένη

Output → Αποτέλεσμα – εξερχόμενο δεδομένο

Input → Εισερχόμενο δεδομένο

Layers → Επίπεδα

Activation function → Συνάρτηση ενεργοποίησης

Binary → Δυαδικός

Hidden layers → Κρυμμένα επίπεδα

Deep learning → Βαθιά Μάθηση

Deep networks → Βαθιά δίκτυα

Error → Σφάλμα

Loss → Απώλεια – εδώ αναφέρεται στη συνάρτηση που υπολογίζει το πόσο κοντά είναι η προβλεπόμενη με την πραγματική τιμή, δηλαδή τι «χάθηκε»

Back / forward propagation → Μέθοδος διαδρομών προς τα πίσω και προς τα εμπρός αντίστοιχα

Gradient descent → Πτώση βαθμίδας

Max pooling → Μέγιστη συγκέντρωση

Flattening → Ισοπέδωση

Features maps → Επίπεδα χαρακτηριστικών

Tuning network → Δίκτυο συντονισμού

Callbacks → Επανακλήσεις

regularizers → Σταθερές για την αποφυγή του θορύβου

stride → Βήμα

summary → Σύνοψη

Bounding Box → Πλαίσιο όπου δημιουργείται στην ανίχνευση για να προσδιορίσει το ζητούμενο αντικείμενο

YOLO → Κοιτάς μόνο μια φορά

Batch → Παρτίδα

Patch → Κομμάτι

Confusion Matrix → Πίνακας σύγχυσης – χρησιμοποιείται στην αξιολόγηση των αποτελεσμάτων

Precision → Ακρίβεια

Specificity → Ειδικότητα

Identity → Ταυτότητα

Global Positioning System → Παγκόσμιο Σύστημα Τοποθεσίας

mean average precision → μέση ακρίβεια

1 Εισαγωγή

1.1 Θέμα

Διάβαση ορίζεται «*το τμήμα του οδοστρώματος που ορίζεται με ειδική σήμανση ή διαγράμμιση ή σηματοδότηση για τη διέλευση των πεζών.*» [1] Η διάβαση αποτελεί σημαντικό στοιχείο υποδομής, μιας και καθημερινά εξυπηρετεί χιλιάδες ανθρώπους ως προς την μετακίνησή τους. Το περπάτημα θεωρείται μέσο μεταφοράς και η ασφάλεια των πεζών μπορεί να επιτευχθεί μόνο μέσω των διαβάσεων, αφού αυτή θεωρείται η νόμιμη δίοδος για να μπορέσει κάποιος πεζός να διασχίσει το δρόμο.

Οι διαβάσεις όμως έχουν διπλό σκοπό. Εκτός από την ασφάλεια των πεζών οι ορισμένες αυτές διαγραμμίσεις στο οδόστρωμα επικοινωνούν με τα οχήματα και τα μηχανάκια, προειδοποιώντας πώς πρέπει να μειώσουν ταχύτητα και να σταματήσουν, διότι πλησιάζουν σε περιοχή διέλευσης πεζών. Με αυτόν τον τρόπο οι διαβάσεις ελέγχουν τη ροή της κίνησης.

Συνεπώς η ανίχνευση διαβάσεων, καθώς και η εύρεση της τοποθεσίας τους είναι σημαντική. Το να είναι γνωστή η τοποθεσία, χρησιμεύει πολύ σε διαδικτυακούς χάρτες, διότι αυξάνεται η ποσότητα και η ποιότητα των γεωχωρικών δεδομένων. Αυτά με τη σειρά τους, εκτός από την χαρτογραφία μπορούν να χρησιμοποιηθούν στην πολεοδομία, στην χωροταξία, στα συγκοινωνιακά και γενικότερα σε μελέτες που αφορούν την ενασχόληση-βελτίωση των υπηρεσιών μετακίνησης και μεταφοράς. Αυτό οδηγεί σε μια πιο βιώσιμη αστική περιοχή, με καλύτερο δίκτυο και μεγαλύτερη ασφάλεια για τους πεζούς.

Επίσης, σημαντική είναι η ανίχνευση και η γνώση της τοποθεσίας των διαβάσεων ως προς θέματα χωροθέτησης. Ειδικότερα, γίνεται λόγος για τη χωροθέτηση διαβάσεων γύρω από χώρους αναψυχής χώρων αναψυχής των πολιτών, όπως πλατείες, πάρκα, παιδικές χαρές και άλλα. Για την καλή λειτουργία μιας πόλης πρέπει οι άνθρωποι που ζουν, εργάζονται και γενικότερα αλληλοεπιδρούν σε αυτήν να είναι χαρούμενοι. Πρέπει η πόλη να είναι βιώσιμη, δηλαδή να παρέχει σε όλους τους πολίτες υψηλή ποιότητα ζωής. Αυτό σημαίνει την κάλυψη των αναγκών των ανθρώπων με τον καλύτερο δυνατό τρόπο. Ανάγκη των ανθρώπων είναι η κοινωνικοποίηση, διότι ο άνθρωπος αποτελεί κοινωνικό όν. Ανάγκη είναι και η ξεκούραση και η χαλάρωση. Με τους χώρους αναψυχής καλύπτονται οι παραπάνω ανάγκες. Μάλιστα το γεγονός πως είναι χαρούμενοι οι άνθρωποι, κοινωνικοποιούνται και αλληλοεπιδρούν μεταξύ τους μέσω των χώρων αναψυχής, συμβάλλει στην αύξηση της παραγωγικότητας, της επικοινωνίας και της συλλογικότητας σε μία πόλη, που συνεπάγεται η βελτίωση της οικονομίας σε αυτήν, όπως και αύξηση της βιωσιμότητας. Είναι προφανής λοιπόν η σημαντικότητα των χώρων αναψυχής. Οι πολίτες πρέπει να έχουν πρόσβαση στους χώρους αυτούς, κάτι το οποίο είναι εφικτό μέσω των διαβάσεων πεζών, ώστε η πρόσβαση να γίνεται με ασφάλεια.

Εκτός από τη σημαντικότητα της ανίχνευσης αλλά και της γνώσης της τοποθεσίας των διαβάσεων για μελέτη, για αύξηση των χωρικών στοιχείων, την ασφαλή μετακίνηση και την πρόσβαση σε χώρους αναψυχής σε όλους, η ανίχνευση των διαβάσεων είναι πολύ σημαντική για μία καλύτερη και ασφαλέστερη μετακίνηση και για τους ανθρώπους με προβλήματα όρασης, ενώ χρησιμοποιείται αρκετά και στα αυτόνομα οχήματα συμβάλλοντας έτσι στην αυτονομία της κίνησης. Στα παραπάνω η εφαρμογή της ανίχνευσης διαβάσεων συναντάται όλο και περισσότερο, διότι όλοι έχουν δικαίωμα στην ασφαλή και γρήγορη μετακίνηση.

[1] ΚΟΚ (Νόμος Ν.2696/23.03.1999 ΦΕΚ.57α, Άρθρο 2)

Για αυτό το λόγο και η διπλωματική αυτή εργασία αποσκοπεί στην ανίχνευση και χαρτογράφηση των διαβάσεων πεζών σε επίπεδο πόλης χρησιμοποιώντας τεχνικές βαθιάς μάθησης. Έτσι ώστε μετέπειτα : είτε μελετητές, είτε μηχανικοί, είτε ερευνητές είτε σχεδιαστές να χρησιμοποιήσουν αυτά τα χωρικά δεδομένα ως προς βελτίωση του αστικού δικτύου και χώρου, κάνοντας τη μετακίνηση των πεζών πιο ασφαλή και πιο γρήγορη καθώς και την πόλη πιο βιώσιμη.

1.2 Σκοπός της εργασίας

Η εργασία σκοπεύει στην ανίχνευση και χαρτογράφηση των διαβάσεων πεζών σε επίπεδο πόλης χρησιμοποιώντας τεχνικές βαθιάς μάθησης. Πιο συγκεκριμένα ανιχνεύονται διαβάσεις με διαγράμμιση ζέβρα και διαβάσεις με επιφάνεια κόκκινου χρωματισμού στο οδόστρωμα για τις πόλεις του Μπρίστολ και του Μάντσεστερ αντίστοιχα με χρήση ποικίλων νευρωνικών δικτύων καθώς και του YOLOV3. Ειδικότερα, το μοντέλο YOLOV3 που αποτελεί τεχνική της βαθιάς μάθησης βασίζεται σε νευρωνικά δίκτυα, είναι γρήγορο και ακριβές και μπορεί να προβλέψει αντικείμενα σε τρείς διαφορετικές κλίμακες. Για αυτό και χρησιμοποιείται ευρέως σε ανίχνευση αντικειμένων. Να τονιστεί πως η ανίχνευση γίνεται σε δορυφορικές εικόνες. Στη συνέχεια, χαρτογραφούνται οι συντεταγμένες των διαβάσεων, οι οποίες μετά μπορούν να χρησιμοποιηθούν από μηχανικούς, επιχειρήσεις, μελετητές και άλλους. Τα στοιχεία αυτά όπως και το προτεινόμενο μοντέλο μπορούν να χρησιμοποιηθούν αργότερα σε μελέτες (χαρτογραφικές, χωροταξικές, χωροθέτησης κλπ.) για την επίτευξη μιας πιο βιώσιμης αστικής περιοχής, με καλύτερο δίκτυο και μεγαλύτερη ασφάλεια για τους πεζούς.

1.3 Βιβλιογραφικές αναφορές

Σε γενικές γραμμές πολλοί ερευνητές έχουν ασχοληθεί με την ανίχνευση διαβάσεων, κυρίως με διαγράμμιση ζέβρα. Ο σκοπός είναι η μεγαλύτερη ασφάλεια στον δρόμο για όλους. Οι περισσότεροι προτείνουν μοντέλα, συστήματα, εφαρμογές για μια πιο ασφαλή και καλύτερη η μετακίνηση των ανθρώπων με προβλήματα όρασης.

Όπως φαίνεται παρακάτω παρατίθενται σχετικές βιβλιογραφικές αναφορές για μοντέλα και συστήματα που έχουν προταθεί σχετικά με την ανίχνευση διαβάσεων χρησιμοποιώντας τεχνικές βαθιάς μάθησης :

- Ειδικότερα οι (Ahmetovic et al., 2015) προτείνουν τεχνική υπολογιστικής όρασης που λαμβάνει δορυφορικές εικόνες για ανίχνευση διαβάσεων διαγράμμισης ζέβρα. Στη συνέχεια τα αποτελέσματα συγκρίνονται με τις εικόνες της Google. Στη μεθοδολογία εφαρμόστηκε ανίχνευση στο Σαν Φρανσίσκο σε διαβάσεις που ορίστηκαν χειροκίνητα. Οι εικόνες είναι 791 σε δρόμους και 406 σε διασταυρώσεις. Η τελική ακρίβεια είναι 97 %. Οι συγγραφείς θεωρούν πως το να είναι γνωστή η θέση των διαβάσεων είναι σημαντικό για έναν τυφλό που θέλει να περάσει το δρόμο με ασφάλεια. Είναι σημαντικό για τους ανθρώπους αυτούς, επειδή μπορεί να τους βοηθήσει να πάρουν καλύτερες αποφάσεις για το δρομολόγιό τους, με μεγαλύτερη ασφάλεια. Συμπεραίνουν λοιπόν πως με τη χρήση των δορυφορικών εικόνων επέρχεται μεγαλύτερη ακρίβεια που συνεπάγεται μεγαλύτερη ασφάλεια κατά τη μετακίνηση των ανθρώπων με προβλήματα όρασης. Να τονιστεί σε αυτό το

σημείο πως διαθέσιμα είναι μόνο τα αποτελέσματα της ανίχνευσης και οι συντεταγμένες της περιοχής μελέτης.

- Οι (Koester D et al., 2016) προτείνουν ένα μοντέλο όπου πραγματοποιείται ανίχνευση σε αεροφωτογραφίες για τις διαβάσεις με διαγράμμιση ζέβρα. Το σύστημα μαθαίνει αυτόματα από γεωχωρικά δεδομένα που διατίθενται για μια περιοχή μελέτης. Χρησιμοποιούντα HOG, LBPH και SVM για την ανίχνευση σε ποικίλα σύνολα δεδομένων. Συμπεραίνουν πως πρακτικά η ανίχνευση των διαβάσεων με διαγράμμιση ζέβρα μπορεί να χρησιμοποιηθεί στην πλοήγηση και στην οδήγηση ή σε εφαρμογές βοήθειας για ανθρώπους με προβλήματα όρασης. Αλλά και γενικότερα χρησιμεύει για την βελτίωση της γενικότερης διαθεσιμότητας και ποιότητας των γεωχωρικών δεδομένων. Με αυτό τον τρόπο, με το να υπάρχει η παραπάνω διαθεσιμότητα δηλαδή, υπάρχουν διαθέσιμα γεωχωρικά δεδομένα με μεγαλύτερη ακρίβεια, τα οποία βοηθούν και στα αυτόνομα αυτοκίνητα, κάτι που αυτομάτως αυξάνει την ασφάλεια στο δρόμο.
- Σύμφωνα με τους (Berriel et al., 2017) : προτείνεται ένα σύστημα που αυτόματα ταξινομεί διαβάσεις με διαγράμμιση ζέβρα σε δορυφορικές εικόνες μεγάλης κλίμακας που χρησιμοποιεί η Google από το OSM. Οι εικόνες ειδικότερα, λαμβάνονται αυτόματα από τα Google Static Maps API, Google Maps Direction API και OSM. Οι εικόνες αφορούν διαβάσεις αλλά και μη. Δεν παρέχεται το σύνολο των εικόνων αυτούσιο για χρήση, όμως παρέχονται κάποια scripts κώδικα για την λήψη των αντίστοιχων εικόνων εφαρμόζοντας μεταβολές. Το προτεινόμενο μοντέλο αναφέρεται σε διαβάσεις με διαγράμμιση ζέβρα και η Διπλωματική Εργασία δεν αναφέρεται μόνο σε αυτές. Επίσης, επειδή πραγματοποιήθηκε μελέτη της περιοχής μελέτης της εργασίας ως προς τους χώρους αναψυχής και τις διαβάσεις ως προς αυτούς στο λογισμικό QGIS, ήταν βολικό συγχρόνως κατά τη μελέτη της περιοχής μελέτης να λαμβάνονται χειροκίνητα και οι εικόνες των διαβάσεων που απασχολούν την εργασία. Επιπλέον το σύστημα χρησιμοποιεί μοντέλα που βασίζονται σε deep learning, τα οποία χρησιμοποιούν αυτές τις εικόνες, που λαμβάνονται, για την εκπαίδευση και την αξιολόγησή τους. Ειδικότερα, ως μεθοδολογία εφάρμοσαν το παραπάνω σύστημα σε 3 Ηπείρους, 9 χώρες και περισσότερο από 20 πόλεις, με συνολικά 245768 εικόνες. Το τελικό αποτέλεσμα έδειξε ακρίβεια 97.11 % στη μεθοδολογία τους. Το σύστημα το πρότειναν για τη διευκόλυνση των ανθρώπων με προβλήματα όρασης αλλά και για τα αυτόνομα αυτοκίνητα.

  Αξιοσημείωτο είναι το γεγονός πως αρχικά για την ανίχνευση διαβάσεων με σκοπό τη διευκόλυνση της μετακίνησης χρησιμοποιήθηκε η κάμερα των κινητών τηλεφώνων μέσω μιας εφαρμογής που βοηθάει τους ανθρώπους με προβλήματα όρασης ώστε να γνωρίζουν που βρίσκονται οι διαβάσεις και να τους κατευθύνουν. Όσον αφορά τα αυτοκίνητα είχε δημιουργηθεί και εγκατασταθεί μια ειδική κάμερα ως υποβοήθημα για τους οδηγούς, που εντοπίζει γραμμές - τμήματα γραμμών (Berriel et al., 2017). Όμως βάσει των συγγραφέων εντοπίζει μόνο διαβάσεις που είναι κάθετες στην πορεία της οδήγησης καθώς και αυτές που είναι μέχρι μίας συγκεκριμένης απόστασης. Για αυτό και στη συνέχεια οι (Ahmetovic et al., 2015) σύμφωνα με τους συγγραφείς προτείνει την ανίχνευση σε δορυφορικές εικόνες.
- Οι (Ghilardi et al., 2018) προτείνουν ένα μοντέλο, το οποίο ανιχνεύει και ταξινομεί σε δορυφορικές εικόνες που έχει λάβει από το Google Maps για την διευκόλυνση των ανθρώπων με προβλήματα όρασης στο να διασχίσουν το δρόμο. Πρακτικά ο χρήστης μέσω του κινητού του στέλνει τις συντεταγμένες

στο σημείο που βρίσκεται (GPS). Αυτές έπειτα συγκρίνονται με τις συντεταγμένες από τη Google και έτσι βρίσκουν την τοποθεσία σε σχέση με το χρήστη π χ αν είναι η διάβαση δεξιά του ή αριστερά του. Η ακρίβεια είναι πολύ καλή με τις πιο πολλές περιπτώσεις να επιτυγχάνουν ακρίβεια 92.7 %. Συμπεραίνουν πως το προτεινόμενο μοντέλο βοηθά τους ανθρώπους με προβλήματα όρασης να κινούνται με ασφάλεια.

- Οι (Silva E et al., 2020) προτείνουν ένα σύστημα που ταξινομεί τις διαβάσεις με διαγράμμιση ζέβρα για την πιο ασφαλή μετακίνηση των ανθρώπων με προβλήματα όρασης. Χρησιμοποιεί έναν υποστηρικτή μηχανής (SVM), με τη βοήθεια του οποίου μειώνεται η μνήμη και μπορεί να τοποθετηθεί σε μικρή συσκευή, σε σχέση με τα υπόλοιπα αντίστοιχα προτεινόμενα ή ήδη υπάρχοντα μοντέλα. Χρησιμοποιούν ένα σύνολο δεδομένων διαβάσεων τεσσάρων τάξεων : 1. Σαν κάποιος να τις κοιτά από την μπροστινή πλευρά, 2. και 3. Από τη δεξιά και την αριστερή πλευρά και 4. Εικόνες που αφορούν τον δρόμο – όχι διαβάσεις. Αυτό το σύνολο δεδομένων όμως δεν είναι συμβατό με τις ανάγκες της παρούσας Διπλωματικής Εργασίας.
- Οι (Dow et al., 2020) προτείνουν ένα μοντέλο, όπου μέσα από κάμερες ανιχνεύονται διαβάσεις και περιοχές στις οποίες αναμένουν οι πεζοί χρησιμοποιώντας συνελικτικά νευρωνικά δίκτυα, με σκοπό τη μελέτη ως προς μείωση των ατυχημάτων. Αντίστοιχα οι (Wang & Jiao, 2021) προτείνουν ένα μοντέλο που χρησιμοποιεί αλγορίθμους αφαίρεσης γραμμών, βελτίωσης της εικόνας και ανίχνευσης των άκρων, με σκοπό την ανίχνευση και την εύρεση της τοποθεσίας διαβάσεων με διαγράμμιση ζέβρα σε βίντεο σε επίπεδο δρόμου.
- Ο (Karatzaferis Odyssefs, 2022) στην διπλωματική του εργασία *«Crosswalk detection for the outdoor navigation of people with visual impairment»* προτείνει ένα μοντέλο για την διέλευση των πεζών με προβλήματα όρασης κάνοντας ανίχνευση διαβάσεων διαγράμμισης ζέβρα χρησιμοποιώντας τον αλγόριθμο YOLOV5 σε αεροφωτογραφίες που λαμβάνονται αυτόματα με σκοπό την είσοδο του μοντέλου σε εφαρμογή για τους ανθρώπους με προβλήματα όρασης.

1.4 Δομή της εργασίας

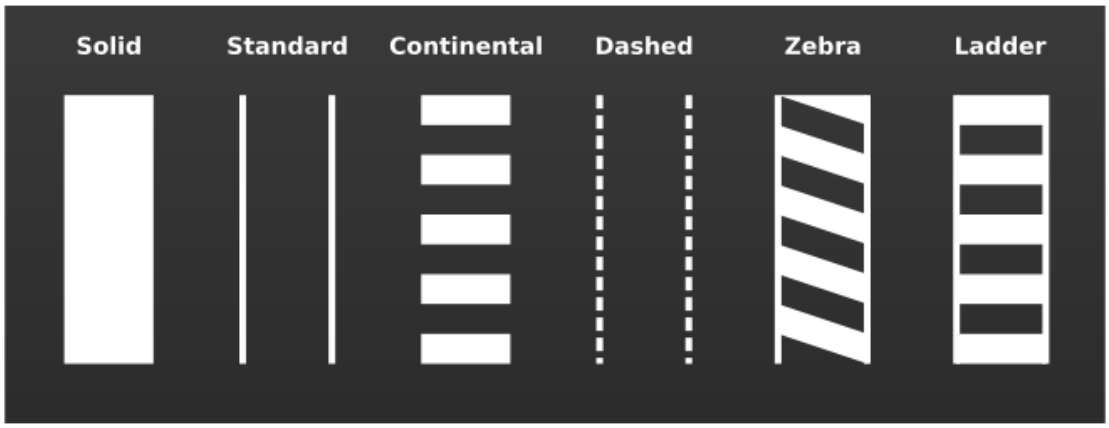
Η παρούσα Διπλωματική εργασία χωρίζεται σε τέσσερα κεφάλαια. Το πρώτο αποτελεί την εισαγωγή, είναι αυτό στο οποίο αναφέρεται το θέμα, ο σκοπός, οι βιβλιογραφικές αναφορές και η δομή της εργασίας. Το δεύτερο κεφάλαιο είναι το θεωρητικό υπόβαθρο και αποτελεί την εκτενή ανάλυση της θεωρίας και των πληροφοριών που χρειάζονται για την κατανόηση του πρακτικού μέρους της εργασίας αλλά και τη γενικότερη εκπόνηση της. Σε αυτό το κεφάλαιο αναλύονται η φιλοσοφία τεχνικών βαθιάς μάθησης, η λειτουργία τους και η αρχιτεκτονική τους, που στη συνέχεια βάσει αυτών χτίζεται ο κώδικας που χρησιμοποιείται στο επόμενο κεφάλαιο, το τρίτο, το κεφάλαιο της μεθοδολογίας και των αξιολογήσεων. Το συγκεκριμένο κεφάλαιο, αναλύει την εφαρμογή του θεωρητικού υποβάθρου ως προς την επίτευξη του σκοπού της εργασίας, καθώς μετέπειτα αξιολογούνται και τα αποτελέσματα της εφαρμογής αυτής. Τέλος, το τέταρτο κεφάλαιο απαρτίζεται από τα τελικά συμπεράσματα και τις προοπτικές.

2. Θεωρητικό Υπόβαθρο

2.1 Διαβάσεις

Όπως αναφέρθηκε παραπάνω μέσω των διαβάσεων οι πεζοί μπορούν να διασχίσουν το δρόμο με ασφάλεια. Υπάρχουν τέσσερα είδη διαβάσεων : οι ελεγχόμενες, οι μη ελεγχόμενες, οι επισημασμένες και η μη επισημασμένες. Οι ελεγχόμενες είναι οι διαβάσεις που συναντώνται σε συνδυασμό με πινακίδες ή και φωτεινή σηματοδότηση. Αντίθετα οι μη ελεγχόμενες είναι εκείνες που συναντώνται χωρίς να συνοδεύονται με σήμανση ή και φωτεινή σηματοδότηση. Οι επισημασμένες από την άλλη πλευρά, αποτελούν τις διαβάσεις οι οποίες σημειώνονται στην επιφάνεια του δρόμου με κάποιο είδος διαγράμμισης, υποδεικνύοντας στους πεζούς τη δίοδο διέλευσης του δρόμου. Ενώ η μη επισημασμένες δεν εμφανίζουν κάποιο είδος διαγράμμισης.

Παρακάτω παρατίθενται τα είδη των επισημασμένων διαβάσεων. Πιο συγκεκριμένα, όπως αναφέρει ένα άρθρο για την ασφάλεια των δρόμων στο Σαν Φρανσίσκο ("*Crosswalks*", (City & County of San Francisco, 2015)), αυτή είναι η κατηγοριοποίηση σύμφωνα με την Αμερική. Οι παρούσες διαβάσεις θεωρούνται υψηλής ορατότητας, κάτι που τις καθιστά προτιμότερες από τα άλλα είδη διαβάσεων. Η τρίτη διάβαση στην εικόνα είναι η ονομαζόμενη στην Ευρώπη διαγράμμιση ζέβρα (continental) και έχει την παρακάτω μορφή : τις λευκές παράλληλες λωρίδες, όπου σε συνδυασμό με το χρώμα του οδοστρώματος μοιάζουν με το χρώμα της ζέβρας. Δεξιά της continental διάβασης φαίνεται το είδος επισημασμένης διάβασης που έχει τη μορφή δύο παράλληλων διακεκομμένων λευκών γραμμών λωρίδων (dashed). Δεξιά της είναι η διάβαση με διαγράμμιση ζέβρα όπως την χρησιμοποιούν στην Αμερική. Το είδος continental και dashed απασχολούν την Διπλωματική Εργασία.



Εικόνα 2.1 -1 Είδη επισημασμένων διαβάσεων (πηγή : (City & County of San Francisco., 2015), Copyright © 2015 City & County of San Francisco.)

Πιο συγκεκριμένα η παρούσα εργασία πραγματεύεται την ανίχνευση επισημασμένων διαβάσεων στις πόλεις Μπρίστολ και Μάντσεστερ της Αγγλίας. Το Μπρίστολ παρουσιάζει κατά κύριο λόγο διαβάσεις με διαγράμμιση ζέβρα, οι οποίες χρησιμοποιούνται περισσότερο και είναι οι πιο δημοφιλείς (continental) παγκοσμίως. Το Μάντσεστερ αντιθέτως περιλαμβάνει επί των πλείστων διαβάσεις ορισμένες από δύο διακεκομμένες παράλληλες λωρίδες (dashed), εντός των οποίων υφίσταται κόκκινος χρωματισμός στο οδόστρωμα, όπως φαίνεται παρακάτω.

Εικόνα 2.1 -2 Διάβαση στο Μπρίστολ

Εικόνα 2.1 -3 Διάβαση στο Μάντσεστερ

2.1.1 Κόκκινη χρωματισμένη επίστρωση

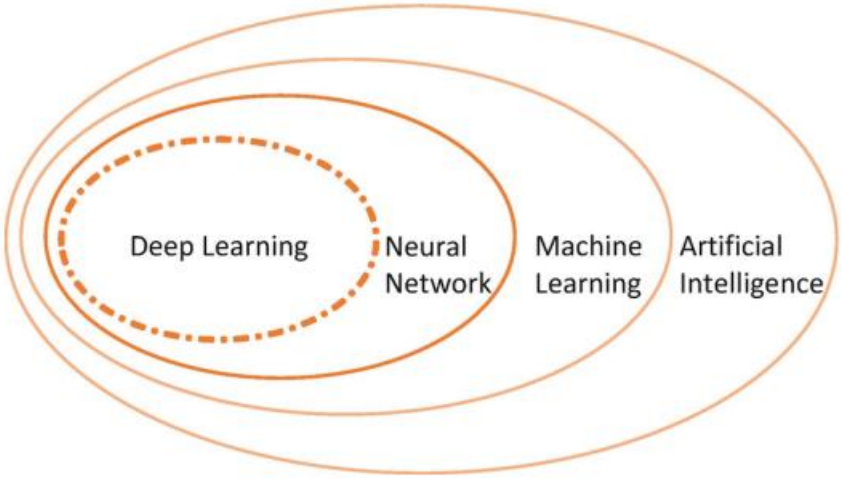
Όπως προαναφέρθηκε το ένα είδος διαβάσεων που ανιχνεύεται περιέχει εντός της διαγραμμίσεως του κόκκινο χρωματισμό στην επιφάνεια του οδοστρώματος. Ο χρωματισμός αυτός αποτελεί μια ειδική διακοσμητική επίστρωση στο οδόστρωμα, η οποία υφίσταται για να προειδοποιεί τα οχήματα και τους μοτοσικλετιστές πως πλησιάζουν σε περιοχή διέλευσης πεζών, ώστε να αποφευχθούν ατυχήματα αυξάνοντας με αυτόν τον τρόπο την ασφάλεια των πεζών. Ειδικότερα, τέτοιου είδους τεχνικές χρησιμοποιούνται σε περιοχές όπου παρατηρείται έντονη κινητικότητα πεζών, καθώς και σε κεντρικές περιοχές, εμπορικές περιοχές, περιοχές που παρουσιάζουν αυξημένο όγκο κυκλοφορίας ή και περιοχές στις οποίες έχουν συμβεί αρκετά ατυχήματα στο παρελθόν σύμφωνα με το άρθρο *«What do colourful surface markings mean on roads?»* της εταιρίας NatraTex, η οποία ειδικεύεται σε επιστρώσεις χρώματος σε επιφάνειες του οδοστρώματος (NatraTex, n.d.)

Το χρώμα κόκκινο λοιπόν και γενικότερα τα έντονα χρώματα στις επιφάνειες των δρόμων βοηθάνε τους οδηγούς να καταλάβουν πώς να κινηθούν στο δρόμο με ασφάλεια, διασφαλίζοντας με αυτόν τον τρόπο την ομαλή ροή κυκλοφορίας και τη μείωση της αποσυμφόρησης όπου δύναται. Από μόνη της μια τέτοια επίστρωση χρωματισμού δεν είναι δυνατό να παρέχει τόση ασφάλεια όση μια υψηλής ορατότητας διάβαση. Επομένως, χρήζει συνδυασμού του χρωματισμού με διαγράμμιση διάβασης υψηλής ορατότητας, κάτι το οποίο παρατηρείται σήμερα σε παγκόσμιο επίπεδο, εκτός από το Μάντσεστερ.

Συνοψίζοντας, χρησιμοποιείται το κόκκινο για την αύξηση της προσοχής του οδηγού την επαγρύπνηση του και την επισήμανση της προσοχής για την περιοχή στην οποία περιλαμβάνεται ο χρωματισμός.

2.2 Βαθιά Μάθηση

Τα μοντέλα που χρησιμοποιούνται στην εργασία αποτελούν τεχνικές της βαθιάς μάθησης. Η βαθιά μάθηση βασίζεται στα νευρωνικά δίκτυα και είναι τμήμα της μηχανικής μάθησης, η οποία με τη σειρά της ανήκει στην τεχνητή νοημοσύνη. Η σχέση των προαναφερθέντων αποτυπώνεται στο παρακάτω σχήμα και εξηγείται εκτενέστερα στα υποκεφάλαια που ακολουθούν.



Εικόνα 2.2 -2 Σχέση τεχνητής νοημοσύνης, μηχανικής και βαθιάς μάθησης και νευρωνικών δικτύων (πηγή : (Long & Zeng, 2022b))

2.2.1 Τεχνητή Νοημοσύνη

Ο σκοπός της τεχνητής νοημοσύνης είναι οι μηχανές να μιμηθούν την ανθρώπινη νοημοσύνη και την ανθρώπινη συμπεριφορά ως προς την εξαγωγή συμπερασμάτων. Ο άνθρωπος όμως, είναι ένα πολύπλοκο ον, που συνεπάγεται πως πολλά από τα οποία κάνει δεν δύναται να εκλογικευτούν. Με συνέπεια, ο σκοπός της τεχνητής νοημοσύνης να αυξάνεται σε επίπεδο δυσκολίας καθώς δεν μπορεί να φτάσει η μίμηση των μηχανών το επίπεδο του ανθρώπινου εγκεφάλου αυτό κάθε αυτό. Για παράδειγμα, το πώς ένας άνθρωπος μαθαίνει ξένες γλώσσες ή το πως αναγνωρίζει αντικείμενα κοιτώντας εικόνες δεν μπορεί να μιμηθεί από μηχανές ακολουθώντας μία σειρά κανόνων (Long & Zeng, 2022a). Συνεπώς, δημιουργήθηκε η μηχανική μάθηση, η οποία παρέχει τη δυνατότητα σε «*μηχανές να μαθαίνουν αυτόματα κανόνες από δεδομένα»* (Long & Zeng, 2022a).

2.2.2 Μηχανική Μάθηση

Η μηχανική μάθηση αποτελεί τμήμα της τεχνητής νοημοσύνης και παρέχει τη δυνατότητα σε μηχανές να μαθαίνουν αυτόματα κανόνες. Μάλιστα, μονίμως βελτιώνει τη διαδικασία μάθησης. Αυτό υφίσταται, βάσει του άρθρου *«Computer Vision .vs Machine Learning .vs Deep Learning» (2022)* (byteant, 2022)της Software Development εταιρείας byteant επειδή χρησιμοποιεί μία «συμπεριφορά» του ανθρώπου: το γεγονός δηλαδή πως μαθαίνει από τα λάθη του και στη συνέχεια δρα αναλόγως, κάνοντας τη μηχανή να συμπεραίνει και να βελτιώνεται συνεχώς. Πρακτικά χρησιμοποιεί έναν «κύκλο» δράσεων που βασίζεται στην εμπειρία. Πολλές περιπτώσεις όμως απαιτούν την *«εκμάθηση περίπλοκης, αφηρημένης λογικής»* (Long & Zeng, 2022a) Η εκμάθηση αυτή πραγματοποιείται μέσω των νευρωνικών δικτύων.

2.2.3 Υπολογιστική Όραση

2.2.3.1 Υπολογιστική όραση – Ταξινόμηση - Ανίχνευση αντικειμένων

Σύμφωνα με τον ((Verdhan, 2021b)) η ανίχνευση αντικειμένων αποτελεί «*τεχνική της υπολογιστικής όρασης».* Ειδικότερα, η υπολογιστική όραση είναι το τμήμα της τεχνητής νοημοσύνης, το οποίο προσπαθεί να μιμηθεί την «όραση» του ανθρώπου και να επωφεληθεί από τις δυνατότητες που αυτή προσφέρει. Για παράδειγμα το γεγονός πως ο άνθρωπος μπορεί να διακρίνει πολλά αντικείμενα σε μία εικόνα και να τα αναγνωρίσει σε οποιοδήποτε μέγεθος και αν αυτά βρίσκονται. Μάλιστα, η υπολογιστική όραση μπορεί να ξεπεράσει και τον άνθρωπο, σε επίπεδο μνήμης αλλά και αναγνώρισης περισσότερων λεπτομερειών.

Η διαφορά λοιπόν, της υπολογιστικής όρασης με την μηχανική μάθηση έγκειται στο γεγονός πως η πρώτη λειτουργεί όπως η δεύτερη, απλώς η μηχανική μάθηση εκπαιδεύει μηχανές να μαθαίνουν αυτόματα από δεδομένα (υποκεφάλαιο 2.2), ενώ η υπολογιστική όραση εκπαιδεύει μηχανές όσων αφορά τον ορατό κόσμο, δηλαδή την κατανόηση της ανθρώπινης συμπεριφοράς σαν μέσα από μια κάμερα ή μια εικόνα.

Η βαθιά μάθηση και η υπολογιστική όραση συνήθως συνδυάζονται ως προς βέλτιστα αποτελέσματα λόγω των θετικών των νευρωνικών δικτύων (υποκεφάλαια 2.2.2, 2.2.3). Πιο συγκεκριμένα, η υπολογιστική όραση αποτελείται από συνελικτικά δίκτυα, τα οποία συμβάλλουν στην αναγνώριση εικόνων (σε επίπεδο εικονοστοιχείου). Στο επόμενο στάδιο με τη βοήθεια των μοντέλων / τεχνικών της βαθιάς μάθησης αυτοματοποιείται η διαδικασία. Ουσιαστικά, χρησιμοποιούνται νευρωνικά με inputs εικόνα / εικόνες και εν τέλει το αντικείμενο αναγνωρίζεται ή ταξινομείται.

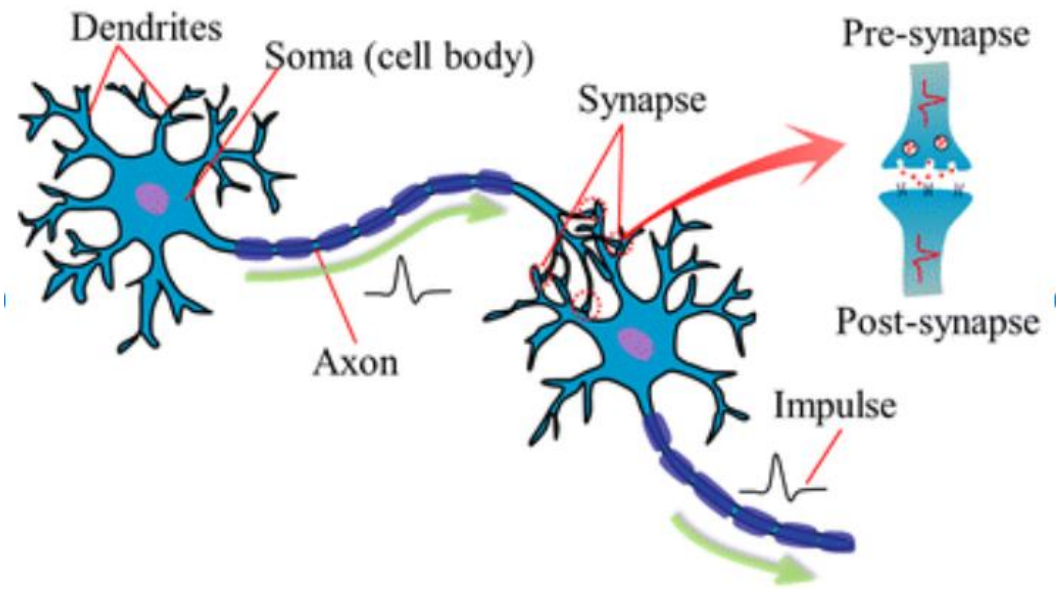
Ένας από τους αλγόριθμους της υπολογιστικής όρασης είναι η ταξινόμηση εικόνων. Αποτελεί τον αλγόριθμο που καθορίζει σε ποια τάξη ανήκει ένα αντικείμενο ή περισσότερα σε μια εικόνα. Πρακτικά το μοντέλο μαθαίνει να ταξινομεί οπτικά δεδομένα χρησιμοποιώντας κατηγορίες εικόνων. Το πιο γνωστό παράδειγμα είναι η ταξινόμηση εικόνων σε γάτα ή σκύλο, όπου χρησιμοποιώντας κατά την ταξινόμηση πολλές εικόνες με γάτες και σκύλους με αποτέλεσμα το μοντέλο να μαθαίνει τι είναι γάτα και τι σκύλος.

Ένας ακόμη γνωστός αλγόριθμος της υπολογιστικής όρασης είναι η ανίχνευση αντικειμένων, η οποία αναγνωρίζει αντικείμενα σε εικόνα / βίντεο και ανιχνεύει την τοποθεσία τους πάνω στην εικόνα / βίντεο. Μία τεχνική βαθιάς μάθησης, η οποία χρησιμοποιείται αρκετά για ανίχνευση αντικειμένων είναι το YOLO.

2.2.4 Νευρωνικά δίκτυα – Βαθιά Μάθηση

2.2.4.1 Λειτουργία νευρωνικού δικτύου

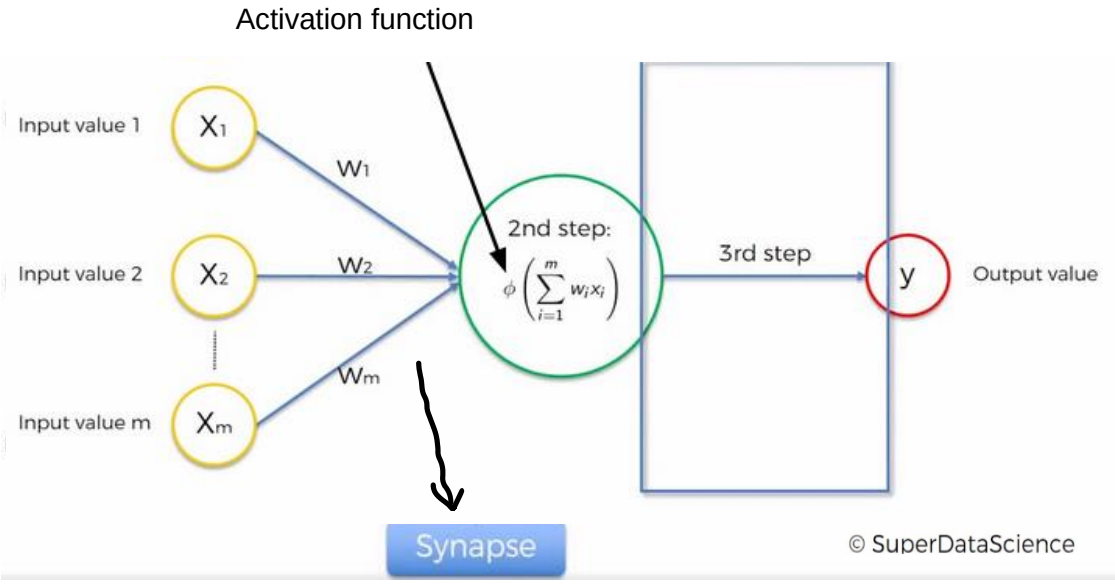
Ο εγκέφαλος είναι το πιο δυνατό εργαλείο που διαθέτει ο άνθρωπος, για να μαθαίνει γρήγορα και να υιοθετεί δεξιότητες και μετά να μπορεί να τις εφαρμόσει. Αυτός είναι και ο σκοπός της μηχανικής μάθησης, να μιμηθεί τη λειτουργία του εγκεφάλου. Για αυτό το λόγο χρησιμοποιεί πολύ και τα νευρωνικά δίκτυα, διότι λειτουργούν πρακτικά όπως οι βιολογικοί νευρώνες. Παρακάτω παρατίθεται μια εικόνα από έναν βιολογικό νευρώνα.



Εικόνα 2.2.4.1 -2 Βιολογικός νευρώνας (πηγή : (Yin et al., 2017)

Ειδικότερα, από την εικόνα παρατηρείται η λειτουργία ενός βιολογικού νευρώνα. Ένας νευρώνας λοιπόν, αποτελείται από τους δενδρίτες (πάνω αριστερά), οι οποίοι δέχονται το σήμα, από τον άξονα που το μεταφέρει και από τη σύναψη, μέσα από την οποία το σήμα μεταφέρεται σε άλλο νευρώνα.

Ομοίως λειτουργεί και ένα νευρωνικό δίκτυο. Δηλαδή, εισάγει δεδομένα (inputs) και παράγει νέα δεδομένα (outputs). Οι συνάψεις δέχονται βάρη. Παρακάτω παρατίθεται εικόνα που αναπαριστά τη λειτουργία ενός νευρωνικού δικτύου, καθώς στη συνέχεια πραγματοποιείται η ανάλυση της σύμφωνα με τους (Long & Zeng, 2022a), (Wichert A & Sa-Couto L, 2021)



Εικόνα 2.2.4.1 -2 Λειτουργία νευρωνικού δικτύου

Παραπάνω απεικονίζεται το νεύρο σε ένα νευρωνικό δίκτυο. Τα inputs με τα βάρη τους δημιουργούν ένα σταθμισμένο άθροισμα το οποίο μεταφέρεται στο νεύρο και εκεί

εφαρμόζεται μια συνάρτηση, η λεγόμενη activation function, που κρίνει αν εν τέλει θα περάσει το σήμα ή όχι. Αν περάσει παράγεται το output, όπου και εδώ εφαρμόζεται μια αντίστοιχη συνάρτηση έτσι ώστε να βγει το τελικό output, το οποίο θα είναι είτε τιμή είτε σε binary μορφή (0,1). Πρακτικά, με μαθηματικούς όρους παράγονται δεδομένα εφαρμόζοντας μια συνάρτηση στα εισαγόμενα δεδομένα.

Στα επίπεδα δεν είναι απαραίτητη η χρήση της ίδιας συνάρτησης (activation function). Σημαντικές συναρτήσεις είναι οι :

- Threshold (Κατώφλι) : Έχει αποτέλεσμα 0 ή 1. Όσο το input είναι < 0 το αποτέλεσμα είναι 0 ενώ για input > = 0 το αποτέλεσμα είναι 1. Σύμφωνα με τη καμπύλη δεν διαφοροποιείται η συνάρτηση στο μηδέν.

  $$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

- Σιγμοειδής : Έχει αποτέλεσμα 0 ή 1 και αυτή. Όμως συγκριτικά με την παραπάνω συνάρτηση το αποτέλεσμα εκφράζει πιθανότητα. Ειδικότερα, για input < 0 το αποτέλεσμα τείνει στο 0, ενώ για input > 0 το αποτέλεσμα τείνει στο 1.

  $$\phi(x) = \frac{1}{1 + e^{-x}}$$

- Tanh : Είναι βέλτιστη σιγμοειδής, με διαφορά ότι το εύρος είναι μεγαλύτερο, από -1 έως 1.

  $$\phi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

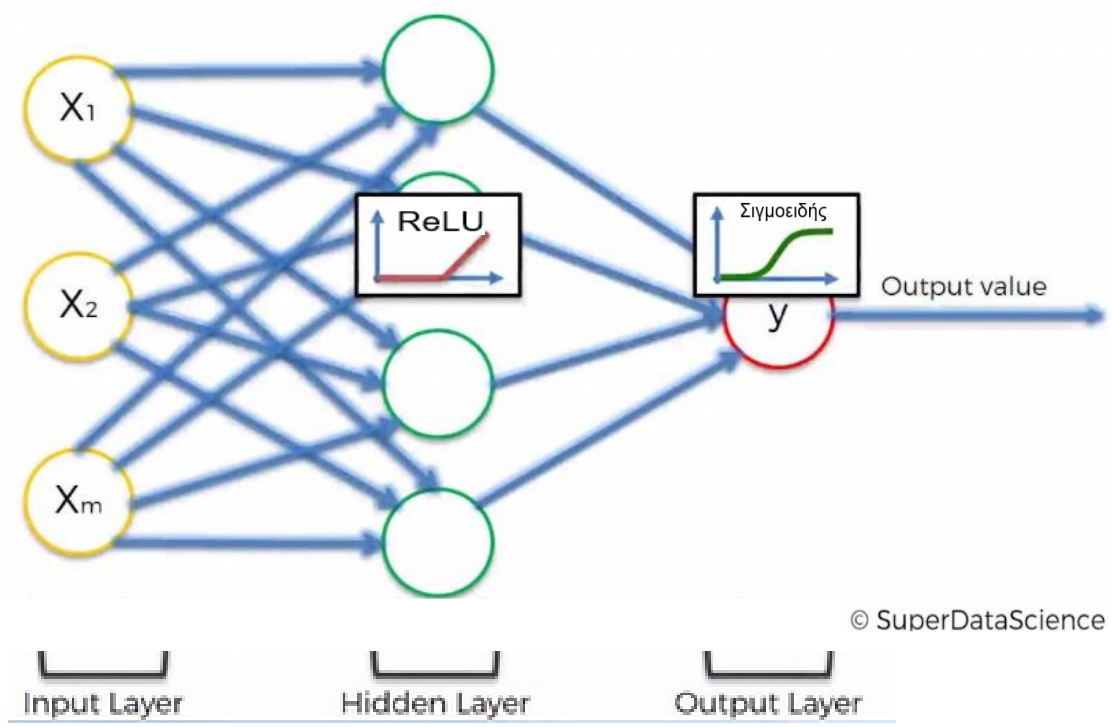
- ReLU : Είναι η συνάρτηση που χρησιμοποιείται περισσότερο. Για input < 0 επιστρέφει 0 και για > 0 επιστρέφει την τιμή του input.

  $$\phi(x) = \max(x, 0)$$

- LeakyReLU : Πρακτικά είναι ReLU με τη διαφορά πως για αρνητικό input υπάρχει συνεχής κλίση, η οποία είναι μια παράμετρος α που βάζει ο χρήστης. LeakyReLU : ReLU Φ(x) = (αx, x)

Προφανώς η δεύτερη συνάρτηση στο τρίτο βήμα, όπως προκύπτει από την Εικόνα 2.2.4.1-2, είναι είτε threshold, είτε σιγμοειδής, αφού και οι δύο παρουσιάζουν αποτέλεσμα 0 ή 1 (αποτέλεσμα binary μορφής).

Παρακάτω φαίνεται στην εικόνα η διατύπωση, σε ένα τεχνητό νευρωνικό δίκτυο, όλης της διαδικασίας μετάδοσης του σήματος που γίνεται στον εγκέφαλο. Όπως το σήμα μεταφέρεται πρώτα στους νευρώνες πριν φτάσει στον εγκέφαλο, έτσι και εδώ υπάρχουν τα κρυμμένα επίπεδα (hidden layers) που ενώνουν το input και output layer και από εκεί μεταφέρεται το σήμα.



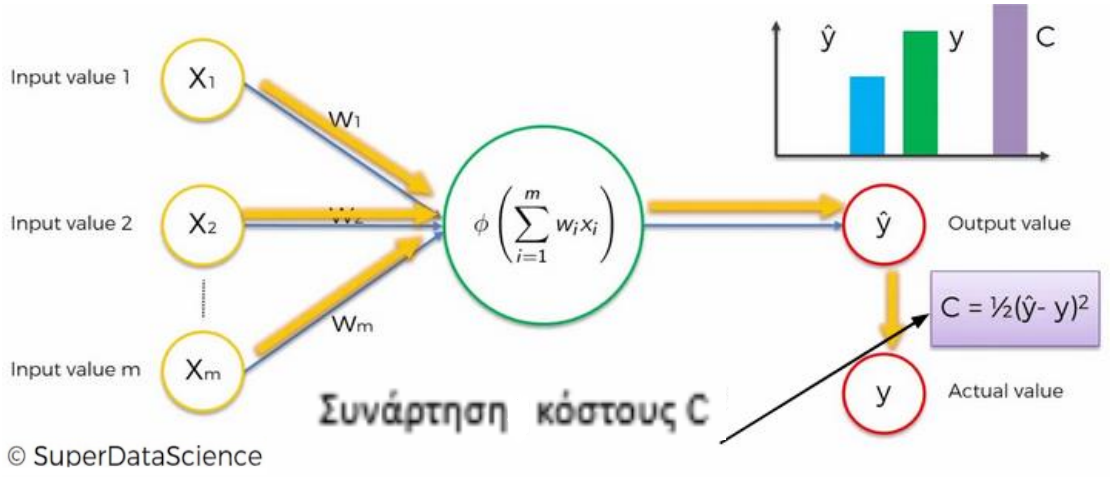
Εικόνα 2.2.4.1 -3 τεχνητό νευρωνικό δίκτυο

Παρατηρείται στην εικόνα ακόμα, πως τα νευρωνικά δίκτυα έχουν ένα επίπεδο εισαγωγής και ένα παραγωγής δεδομένων αλλά δύναται να έχουν πολλά “κρυμμένα” (hidden) ενδιάμεσα. Επίσης, δεν πάνε όλες οι τιμές στα ίδια νευρώνια. Κάθε ένα από αυτά αντιλαμβάνεται διαφορετικές σχέσεις που υπάρχουν σε εκείνα. Αυτό έχει ως συνέπεια την ύπαρξη περισσότερης πληροφορίας ως προς συγκεκριμένα χαρακτηριστικά που βελτιώνουν το δίκτυο και του προσδίδουν ευλυγισία.

Όταν υπάρχουν περισσότερο από ένα κρυμμένα (hidden), τότε το δίκτυο ονομάζεται βαθύ (deep). Επομένως η βαθιά μάθηση (deep learning) ονομάζεται έτσι διότι, χρησιμοποιεί βαθιά (deep) δίκτυα.

2.2.4.2 «Μάθηση» – Εκπαίδευση νευρωνικού δικτύου

Στο 2.2.4.1 περιεγράφηκε η λειτουργία ενός νευρωνικού δικτύου. Στο παρόν στάδιο αναλύεται η διαδικασία της μάθησης του δικτύου σύμφωνα με τους (Wichert A & Sa-Couto L, 2021) Όπως αναφέρθηκε προηγουμένως στο 2.2.2 γίνεται λόγος για έναν «κύκλο» δράσεων που βασίζεται στην εμπειρία. Αυτός ο «κύκλος» παρατηρείται και στα νευρωνικά (ως τμήμα της μηχανικής μάθησης). Το πρώτο κομμάτι του «κύκλου» ονομάζεται forward propagation και παρατίθεται παρακάτω :

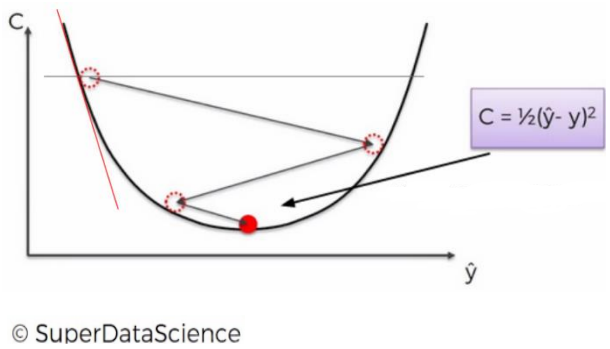


Εικόνα 2.2.4.2 -1 forward propagation

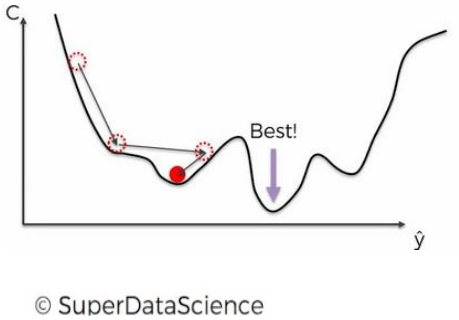
Η διαδικασία έχει πορεία από τα αριστερά προς τα δεξιά. Στο 2.2.3.1 αναλύθηκε η λειτουργία του δικτύου μέχρι την παραγωγή του output. Αυτό όμως δεν αποτελεί το πραγματικό output αλλά προκύπτει από την πρόβλεψη, οπότε συμβολίζεται με ένα «καπέλο» (^) από πάνω του. Για να μπορέσει να μάθει το δίκτυο από το «λάθος» του (2.2.2) είναι σημαντικό να βρεθεί αρχικά αυτό το λάθος (error). Για την εύρεση του λάθους πραγματοποιείται σύγκριση της προβλέψιμης τιμής με της πραγματικής. Η σύγκριση αυτή λαμβάνει χώρα μέσω της συνάρτησης κόστους C ή loss. Στόχος είναι η ελαχιστοποίηση της συνάρτησης αυτής. Αυτό σημαίνει πως η διαφορά της πραγματικής και της προβλεπόμενης τιμής να είναι η ελάχιστη και συνεπώς το σφάλμα να είναι το ελάχιστο. Στην εικόνα παραπάνω παρατηρείται η συνάρτηση κόστους σε ένα μοβ πλαίσιο.

Το δεύτερο τμήμα της διαδικασίας μάθησης ονομάζεται back propagation και έχει πορεία από δεξιά προς τα αριστερά. Ουσιαστικά η πληροφορία από την συνάρτηση κόστους πηγαίνει προς τα πίσω, φτάνει στα βάρη των input νευρώνων και ενημερώνονται ή διορθώνονται. Αυτή είναι η διαδικασία μάθησης, η οποία αναφέρεται σε μια σειρά δεδομένων.

Το επόμενο στάδιο της μάθησης είναι η εκπαίδευση, όπου το νευρωνικό «εκπαιδεύεται» με αυτά που έμαθε στη διαδικασία της μάθησης. Η back propagation πραγματοποιείται με τη μέθοδο gradient descent. Στην συνάρτηση C που αναφέρθηκε (2ου βαθμού→ καμπύλη με κοίλα προς τα πάνω) μελετάται η κλίση σε διάφορα σημεία από πάνω αριστερά συνεχίζοντας με κίνηση προς τα κάτω (αρνητική κλίση) ώστε να καταλήξει στο ολικό ελάχιστο της συνάρτησης, μιας και ο στόχος είναι η ελαχιστοποίησή της όπως αναφέρθηκε παραπάνω. Η μέθοδος καλείται batch gradient descent, η οποία ενημερώνει τα βάρη ταυτόχρονα με το πέρας της εκπαίδευσης όλων των σειρών. Όμως η C δεν είναι πάντα τέτοιας απλής μορφής οπότε ελλοχεύει ο κίνδυνος εύρεσης τοπικού ελαχίστου με αυτόν τον τρόπο. Συνεπώς, χρησιμοποιείται η μέθοδος stochastic gradient descent, που ενημερώνει τα βάρη μετά την εκπαίδευση κάθε σειράς ή ενός αριθμού σειρών. Παρατίθενται αντίστοιχες εικόνες παρακάτω :



Εικόνα 2.2.4.2 -2 batch gradient descent



Εικόνα 2.2.4.2 -3 stochastic gradient descent

Λαμβάνοντας υπόψη τα παραπάνω, η διαδικασία της εκπαίδευσης (training) ενός τεχνητού νευρωνικού είναι η εξής :

- Αρχικοποιούνται τυχαία τα βάρη με τιμές κοντά στο μηδέν
- Εισάγεται η πρώτη παρατήρηση του dataset για την πρώτη γραμμή
- Γίνεται η forward propagation από αριστερά προς τα δεξιά μέχρι να βρεθεί το τελικό output.
- Αυτό όμως δεν είναι το αληθινό αλλά μια πρόβλεψη, οπότε γίνεται μια σύγκριση ανάμεσά τους και εκτιμάται το σφάλμα
- Γίνεται η back propagation
- Όλα τα παραπάνω επαναλαμβάνονται για κάθε σειρά ή αριθμό σειρών
- Όταν ολοκληρωθεί η διαδικασία για όλο το dataset έχει ολοκληρωθεί μία εποχή. Η εκπαίδευση τελειώνει με το πέρας όλων των ζητούμενων εποχών.

2.2.4.3 CNN

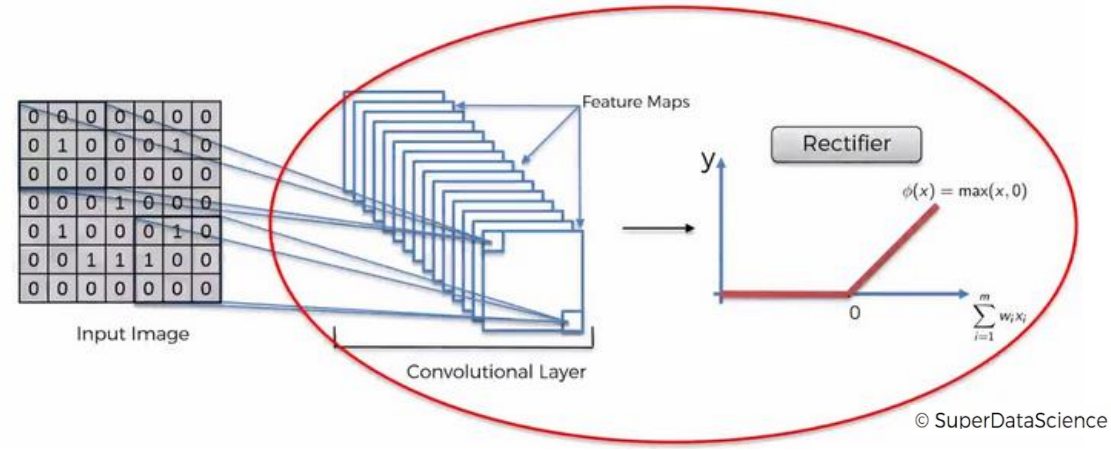
CNN ((Verdhan, 2021b)) ονομάζονται τα συνελικτικά νευρωνικά δίκτυα, τα οποία είναι βαθιά δίκτυα και χρησιμεύουν πολύ στην υπολογιστική όραση (υποκεφάλαιο 2.3). Απαρτίζονται από επίπεδα convolutional (συνελικτικά), pooling (max, min), flattening και fc. Ειδικότερα, βάση ορισμού :

- convolution : *παίρνει δύο εικόνες ως input και παράγει μια ως output. Η μια input είναι η δεδομένη εικόνα και η δεύτερη αποτελεί ένα φίλτρο (kernel) που εφαρμόζεται στην εικόνα και παράγεται μια νέα τροποποιημένη και επειδή τα δεδομένα είναι σε μορφή πινάκων, ουσιαστικά γίνεται πολλαπλασιασμός του kernel στην εικόνα.*
- pooling : *πρακτικά μειώνει τη διάσταση του input δεδομένου του, ώστε να ληφθούν πληροφορίες για τις δεσμευμένες περιοχές.*
- *flatten : Τα στοιχεία μπαίνουν σε μία στήλη για να χρησιμοποιηθούν ως input στα fc*
- *fc : τεχνητά (ANN) δίκτυα (dense layers) ώστε τελικά να βρεθεί το output*

Τα βήματα για την δημιουργία ενός συνελικτικού νευρωνικού δικτύου είναι τα εξής : πρώτα το convolutional επίπεδο, έπειτα συμβαίνει max pooling, flattening και τέλος το fc επίπεδο.

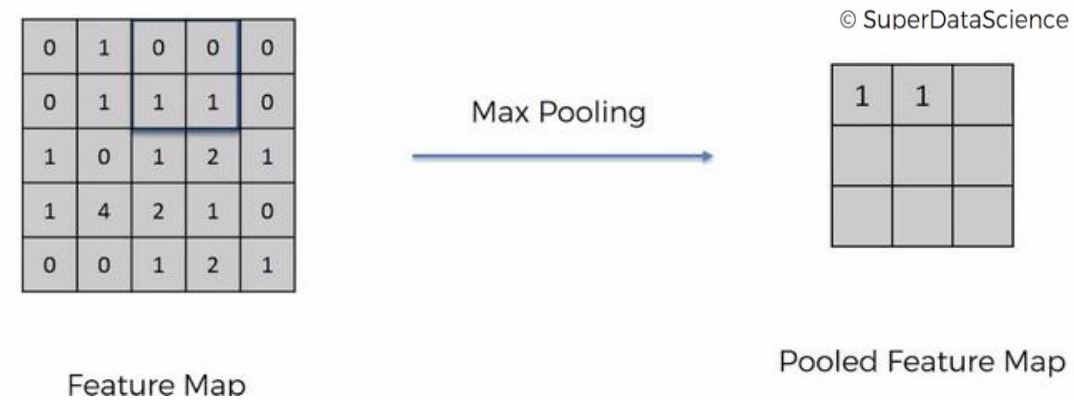
Ειδικότερα, η δημιουργία του convolutional επιπέδου αποτελεί το πρώτο στάδιο. Ένα φίλτρο kernel (σταθερό τετράγωνο που ονομάζεται τοπικό δεκτικό πεδίο) εφαρμόζεται στην εικόνα και προκύπτουν feature maps (το αποτέλεσμα της εφαρμογής ενός φίλτρου στο προηγούμενο επίπεδο), στα οποία εφαρμόζεται μια activation function, με

συνηθέστερη την ReLu. Η συνάρτηση εφαρμόζεται για να αφαιρεθεί η γραμμικότητα, γιατί οι εικόνες δεν είναι γραμμικές. Η διαδικασία φαίνεται παρακάτω.



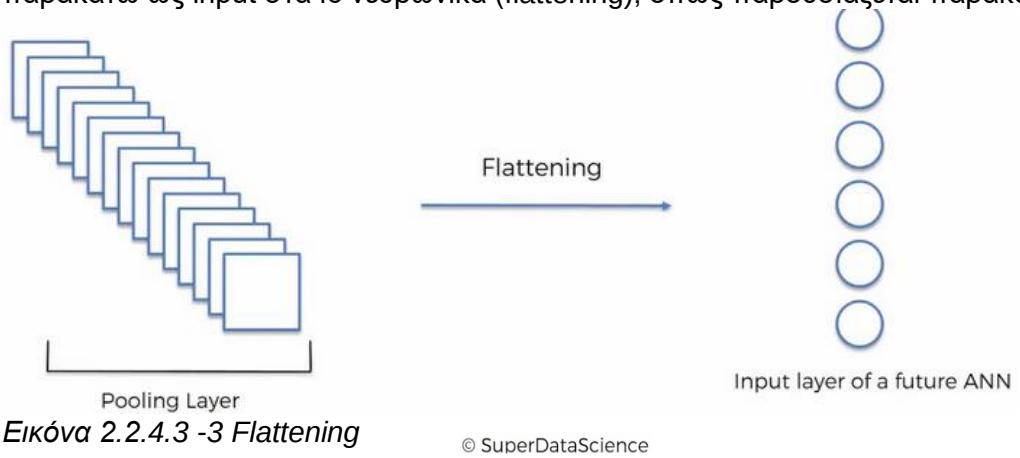
Εικόνα 2.2.4.3 -1 Συνελικτικό επίπεδο

Στη συνέχεια, συμβαίνει max pooling. Το pooling μειώνει τη διάσταση κρατώντας τη σημαντική πληροφορία με αποτέλεσμα να συντελεί κατά του overfitting (θόρυβος στα δεδομένα εκπαίδευσης που γίνεται δεκτός από το μοντέλο). Στην περίπτωση του max, που είναι η πιο συνήθης επιλογή, κρατούνται οι μέγιστες τιμές, ενώ στην περίπτωση του min οι ελάχιστες, όπως φαίνεται παρακάτω.



Εικόνα 2.2.4.3 -2 Max Pooling

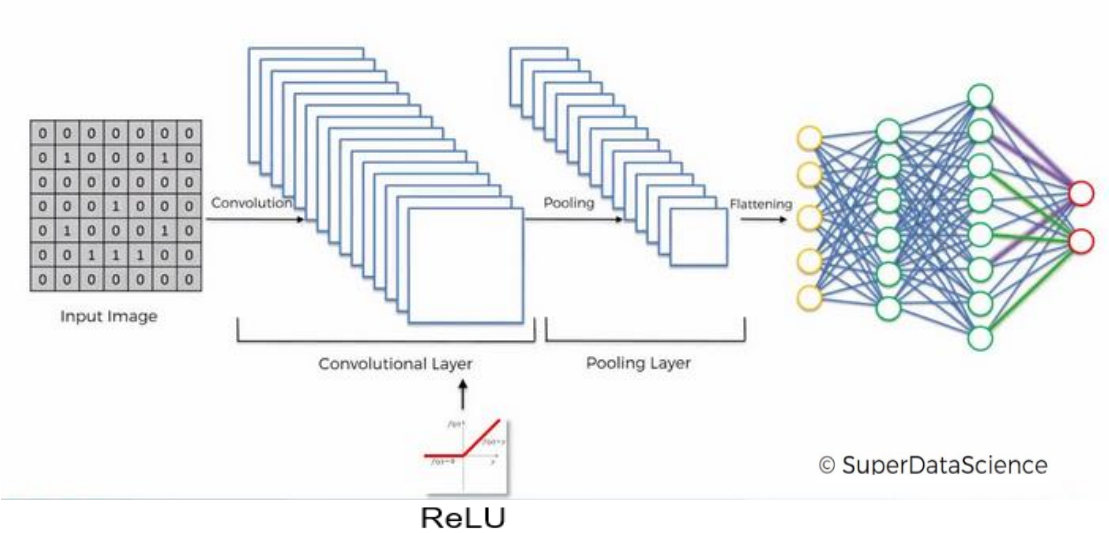
Ύστερα, όλα τα στοιχεία του πίνακα μπαίνουν σε μία στήλη για να χρησιμοποιηθούν παρακάτω ως input στα fc νευρωνικά (flattening), όπως παρουσιάζεται παρακάτω.



Εικόνα 2.2.4.3 -3 Flattening

Τελευταίο στάδιο τα fc, τα οποία ουσιαστικά είναι τεχνητά νευρωνικά δίκτυα, τα οποία θα δώσουν τελικά το output και τα οποία αναλύθηκαν σε προηγούμενο υποκεφάλαιο.

Με μία εικόνα, η αρχιτεκτονική ενός συνελικτικού νευρωνικού δικτύου, βάσει των προαναφερθέντων παρουσιάζεται παρακάτω :



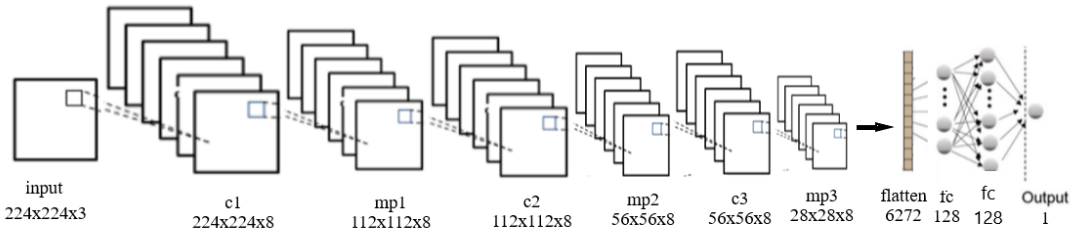
Εικόνα 2.2.4.3 -4 Αρχιτεκτονική CNN

2.2.4.4 ZF – Net και Resnet

Υφίστανται πολλά είδη συνελικτικών νευρωνικών δικτύων. Δύο από αυτά είναι το ZF – Net και το ResNet.

Αρχικά για το ZF – Net (Zeiler & Fergus, 2013):

Το δίκτυο αυτό περιέχει μια μίξη συνελικτικών και pooling επιπέδων με fc επίπεδα. Πετυχαίνει καλό αποτέλεσμα με εκπαίδευση χιλίων εικόνων ανά δείγμα. Η λειτουργία του αναλύεται στο προηγούμενο υποκεφάλαιο σε σχέση με τη σημασία κάθε του επιπέδου, καθώς και τη διαδικασία της εκπαίδευσης. Ειδικότερα, εισέρχεται η εικόνα, η οποία έχει μέγεθος 224 x 224 και τρία κανάλια RGB. Τη συνέχεια, δημιουργείται το δίκτυο μέσω των επιπέδων που αναφέρονται παραπάνω, έπειτα γίνεται flattening, τα αποτελέσματα του οποίου εισέρχονται ως δεδομένα σε fc δίκτυα, από τα οποία εν τέλει απορρέει το αποτέλεσμα. Η διπλωματική εργασία εφαρμόζει το παρόν νευρωνικό για ταξινόμηση διαβάσεων σε επίπεδο πόλης. Επομένως το τελικό αποτέλεσμα είναι μία τιμή μηδέν ή ένα (0 ή 1) που δηλώνει αν το αντικείμενο είναι διάβαση (1) ή όχι (0). Παρακάτω φαίνεται η αρχιτεκτονική του, όπου παρουσιάζεται η μίξη των επιπέδων που προαναφέρθηκε :



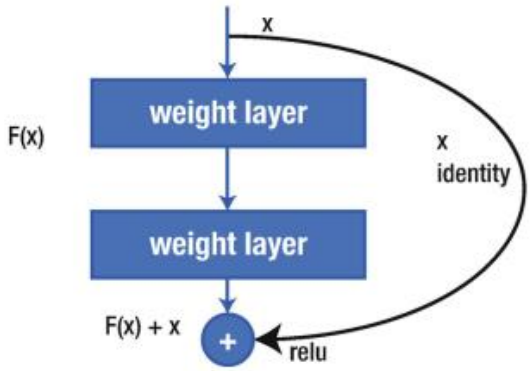
Εικόνα 2.2.4.3 -5 Αρχιτεκτονική ZF – Net

Όπου :

- c →CNN
- mp → max pooling

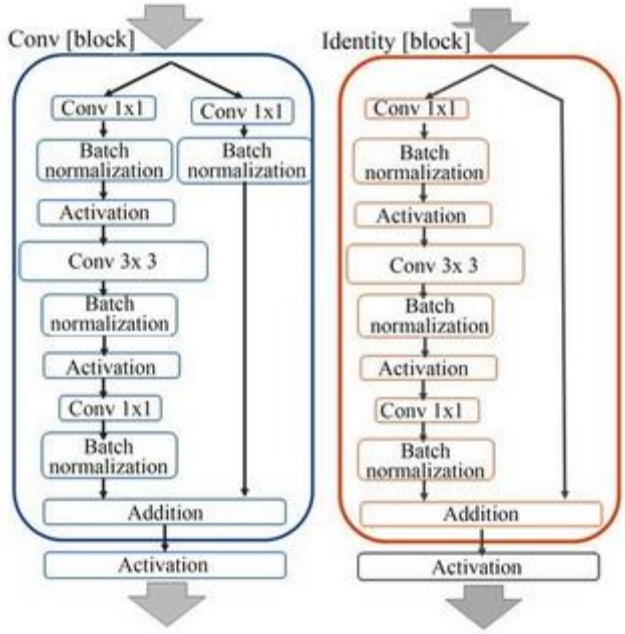
Για το ResNet (Verdhan, 2021b) :

Το δίκτυο αυτό περιέχει 152 επίπεδα και εκατομμύρια εκπαιδευόμενες παραμέτρους που για να υπολογιστούν χρειάζονται γύρω στα σαράντα χρόνια με ένα απλό συνελικτικό δίκτυο. Θεωρείται καλό για ταξινόμηση, καθώς λύνει και το πρόβλημα των vanishing gradients (κανόνας της αλυσίδας). Όσο πιο βαθύ είναι ένα δίκτυο τόσο περισσότερη πληροφορία ως προς συγκεκριμένα χαρακτηριστικά υπάρχει και επομένως θεωρείται καλύτερο το δίκτυο. Όμως, όσο πιο βαθύ τόσο πιο πολλά νευρώνια υπάρχουν. Αυτό σημαίνει πως κατά την back propagation πρέπει να γίνει περισσότερες φορές η gradient descent, άρα είναι περισσότερες και οι πράξεις που πρέπει να γίνουν βάση του κανόνα της αλυσίδας (υπολογίζει τις παραγώγους σύνθετων συναρτήσεων). Αυτό έχει ως αποτέλεσμα η πληροφορία από τη loss function που θα φτάσει στα αρχικά βάρη να είναι ελάχιστη ή και μηδαμινή, που συνεπάγεται τη μη ενημέρωσή τους και επομένως το δίκτυο δε θα “μάθει” τι πρέπει να γίνει. Το Resnet λύνει αυτό το πρόβλημα διότι η πληροφορία πηγαίνει κατευθείαν πίσω μέσω των συνδέσεων παράλειψης. Οι συνδέσεις αυτές διαφοροποιούν το δίκτυο στην αρχιτεκτονική από τα άλλα συνελικτικά δίκτυα. Γίνεται λόγος για το residual block του Resnet. Σύμφωνα με αυτό το Resnet αποτελείται από δύο “κλαδιά”, το κύριο που είναι σειρά επιπέδων νευρωνικών δικτύων και το shortcut που η πληροφορία από το input πηγαίνει κατευθείαν στο output, ενώ η τελική activation function μπαίνει μετά το shortcut κλαδί, όπως φαίνεται παρακάτω :



Εικόνα 2.2.4.3 -6 Residual block του Resnet (πηγή : (Verdhan, 2021a)*)*

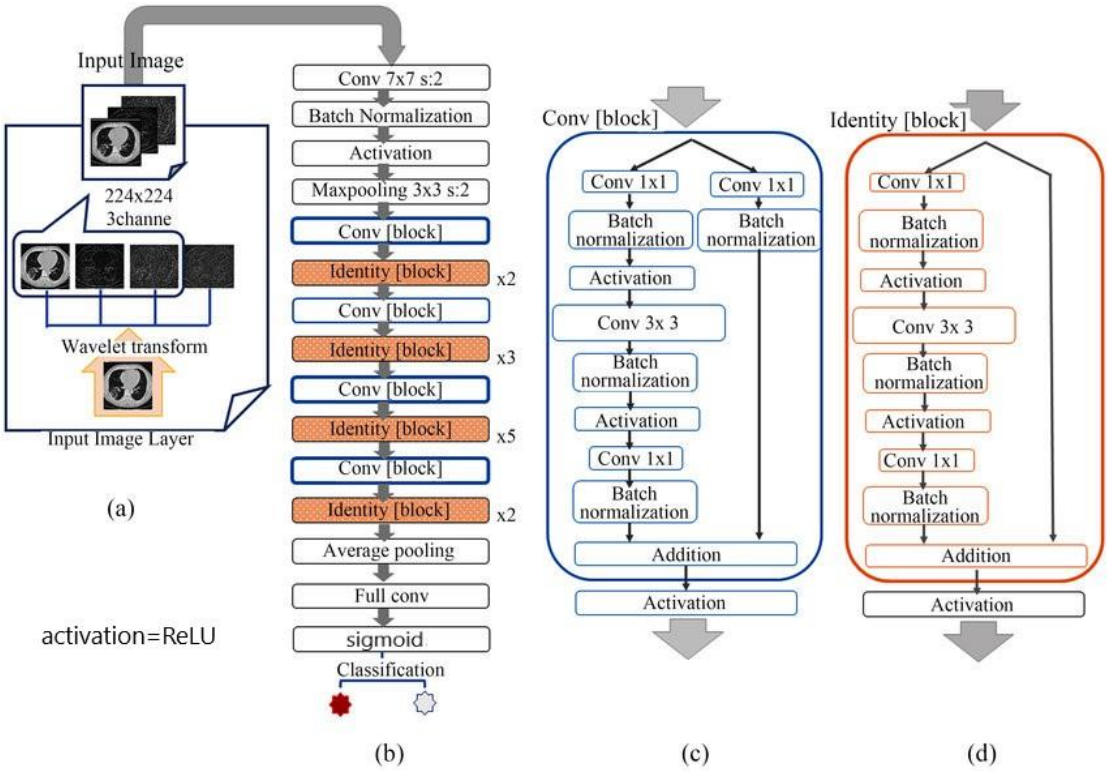
Περιέχει δύο residual blocks, το identity και το convolutional. Παρακάτω παρατίθεται η αρχιτεκτονική τους :



Εικόνα 2.2.4.3 -7 Αρχιτεκτονική Resnet -50 (πηγή :(Matsuyama, 2020)*)*

Και τα δύο είναι residuals δηλαδή έχουν shortcut για τον κανόνα της αλυσίδας, καθώς επίσης και τα δύο «τρέχουν» κατά την εφαρμογή του μοντέλου. Απλώς το κάθε ένα σχετίζεται με το μέγεθος των διαστάσεων των inputs / outputs του μοντέλου. Ειδικότερα, το καθένα χρησιμοποιείται στο μοντέλο ανάλογα τις διαστάσεις για κάθε πληροφορία. Δηλαδή αν οι διαστάσεις των inputs είναι διαφορετικές των outputs χρησιμοποιείται το convolutional block, διαφορετικά αν οι διαστάσεις είναι ίδιες χρησιμοποιείται το identity. Έτσι, η πληροφορία περνά από το αντίστοιχο κάθε φορά. Να σημειωθεί πως μπορούν να αλλάξουν οι διαστάσεις κάνοντας χρήση φίλτρου kernel 1 x 1 και με padding, το πόσα pixels δηλαδή θα προστεθούν στην εικόνα κατά τη χρήση φίλτρου.

Υπάρχουν πολλά είδη Resnet, το -18, -34, -50 και άλλα (He et al., 2015). Η εργασία αυτή ενδιαφέρεται για το -50. Όπως φαίνεται στην παραπάνω εικόνα το identity block έχει στο κύριο κλαδί μια σειρά από convolutional επίπεδα και στο shortcut επίπεδο η πληροφορία πηγαίνει κατευθείαν στο output. Αντίστοιχα, στο convolutional block Η διαφορά με το άλλο block είναι πώς στο shortcut υπάρχει και ένα convolutional επίπεδο. Η activation function παρατηρείτε πως εφαρμόζεται στο τέλος μετά το shortcut κλαδί. Το block αυτό περιέχει και batch normalization, το οποίο βελτιώνει την ακρίβεια, σταθεροποιεί το μοντέλο και αυξάνει το ρυθμό μάθησης. Επίσης παρατηρούνται τρία συνελικτικά επίπεδα στο κύριο κλαδί για κάθε block. Η αρχιτεκτονική του Resnet -50 παρατίθεται παρακάτω :



Εικόνα 2.2.4.3 -8 Αρχιτεκτονική Resnet -50 (αναλυτική)
(πηγή :(Matsuyama, 2020)*)*

Αντίστοιχα με το προηγούμενο δίκτυο, και αυτό χρησιμοποιείται στην εργασία για ταξινόμηση διαβάσεων σε επίπεδο πόλης. Παρατηρείται στην εικόνα αναλυτικά η είσοδος της εικόνας, η δημιουργία του δικτύου με τα αντίστοιχα επίπεδα, περνάει στα fc και καταλήγει στο αποτέλεσμα της ταξινόμησης, που όπως και στο ZF – Net το αποτέλεσμα είναι μία τιμή (0 ή 1).

Έπειτα από την πρόβλεψη (αποτελέσματα ταξινόμησης) σειρά έχει η αξιολόγηση των αποτελεσμάτων. Στα νευρωνικά δίκτυα όπως αναφέρθηκε σε προηγούμενο υποκεφάλαιο γίνεται η πρόβλεψη του output y^ και στη συνέχεια συγκρίνεται με το πραγματικό υπολογίζοντας τον confusion matrix, εκτιμώντας με αυτόν τον τρόπο και τα σφάλματα: recall = tp / (tp + fn), precision = tp / (tp + fp), specificity = tn / (tn + fp) βάση του ορισμού του πίνακα:

		Predicted	
		Negative (**N**) **-**	Positive (**P**) **+**
Actual	Negative **-**	True Negatives (T**N**)	False Positives (F**P**) **Type I error**
	Positive **+**	False Negatives (F**N**) **Type II error**	True Positives (T**P**)

Εικόνα 2.2.4.3 -9 Confusion matrix (πηγή : (NBSHARE NOTEBOOKS, n.d.))

Όπου :

- Τιμή αληθής και θετική (TP).
- Τιμή ψευδής και θετική (FP).
- Εάν η πρόβλεψη είναι ψευδής ενώ το αντικείμενο υφίσταται πραγματικά, τότε η τιμή θεωρείται ψευδής αρνητική (FN)
- Εάν η πρόβλεψη είναι ψευδής και το αντικείμενο δεν υφίσταται πραγματικά, τότε η τιμή θεωρείται αληθής αρνητική (TN)

Ενώ η ολική ακρίβεια του μοντέλου: (tp + tn) / (fp + fn + tn + tp)

2.2.5 Ανίχνευση αντικειμένων με YOLOV3

2.2.5.1 YOLO

Ο αλγόριθμος YOLO δημιουργήθηκε από τους (Redmon et al., 2016b) Σύμφωνα με το αντίστοιχο paper τους, το YOLO (You Only Look Once) αποτελεί μια τεχνική βαθιάς μάθησης, ένα μοντέλο / αλγόριθμος, ο οποίος χρησιμοποιείται συχνά στην ανίχνευση εικόνων και το όνομά του «προδίδει» την αιτία. Πιο συγκεκριμένα, το μοντέλο με μία μόνο επανάληψη, με μία μόνο «ματιά», σε μία εικόνα / βίντεο ή πλήθος εικόνων / βίντεο σε πραγματικό χρόνο, έχει την ικανότητα να ανιχνεύει το ζητούμενο αντικείμενο, προβλέποντας ένα Bounding Box. Ταυτόχρονα προβλέπει και την πιθανότητα (σε μορφή ποσοστού %) κάθε τάξεως / κατηγορίας εφαρμόζοντας ένα μόνο συνελικτικό νευρωνικό δίκτυο σε όλη την εικόνα.

Ειδικότερα, η ανίχνευση γίνεται χωρίζοντας την εικόνα σε πολλές περιοχές προβλέποντας Bounding Boxes και πιθανότητες για κάθε περιοχή. Όμως κάθε περιοχή έχει την ικανότητα να προβλέψει αντικείμενο/α μόνο μίας κατηγορίας, που συνεπάγεται περιορισμό σε ανιχνεύσεις όπου υπάρχουν αντικείμενα παραπάνω από μια κατηγορία σε μια περιοχή. Επιπροσθέτως, μπορεί να προβλεφθεί μικρός αριθμός Bounding Boxes, που σημαίνει πως μικρά όμοια αντικείμενα στην ίδια περιοχή είναι δύσκολο να ανιχνευθούν.

2.2.5.2 YOLOV3

Υφίστανται αρκετές εκδόσεις YOLO. Η YOLOV3 απασχολεί την διπλωματική εργασία. Είναι η τρίτη επίσημη έκδοση και διαφέρει από τις προηγούμενες YOLO, YOLOV2 στο γεγονός πως είναι ταχύτερη και ακριβέστερη (Redmon & Farhadi, 2018). Προτάθηκε το μοντέλο αυτό από τους Joseph Redmon, Ali Farhadi, οι οποίοι αποτελούν τους δημιουργούς και των προηγούμενων εκδόσεων. Αξιοσημείωτο είναι το γεγονός πως ο Joseph Redmon έπειτα από την 3η έκδοση αποσύρθηκε από την κοινότητα της τεχνητής νοημοσύνης για ηθικούς λόγους, για αυτό και όλες οι υπόλοιπες μετέπειτα εκδόσεις δε θεωρούνται «επίσημα» δουλειά του.

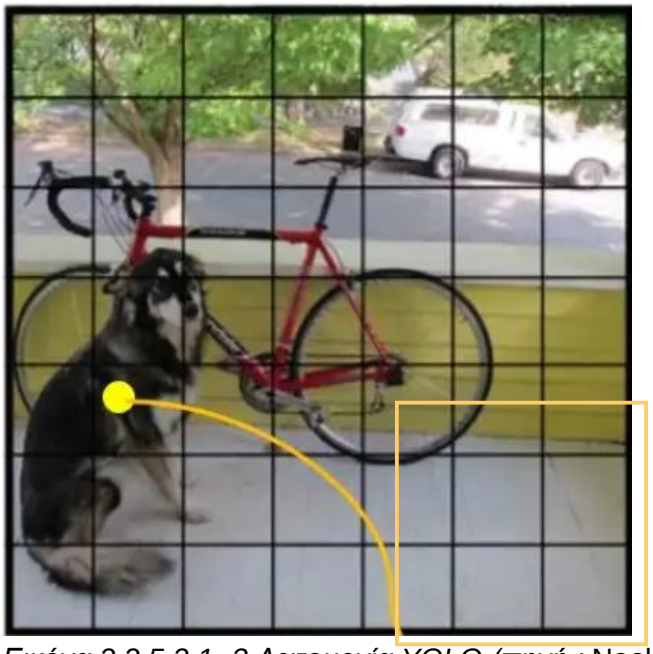
2.2.5.2.1 Λειτουργία - Αρχιτεκτονική

Ο αλγόριθμος αυτός με μία πρώτη ματιά λειτουργεί ως εξής (Redmon & Farhadi, 2018):

- Όπως αναφέρθηκε στο 2.2.5.1 εφαρμόζεται ένα μόνο νευρωνικό δίκτυο. Αυτό το δίκτυο χωρίζει την εικόνα σε περιοχές. Ειδικότερα σε S x S κελιά ενός καννάβου.

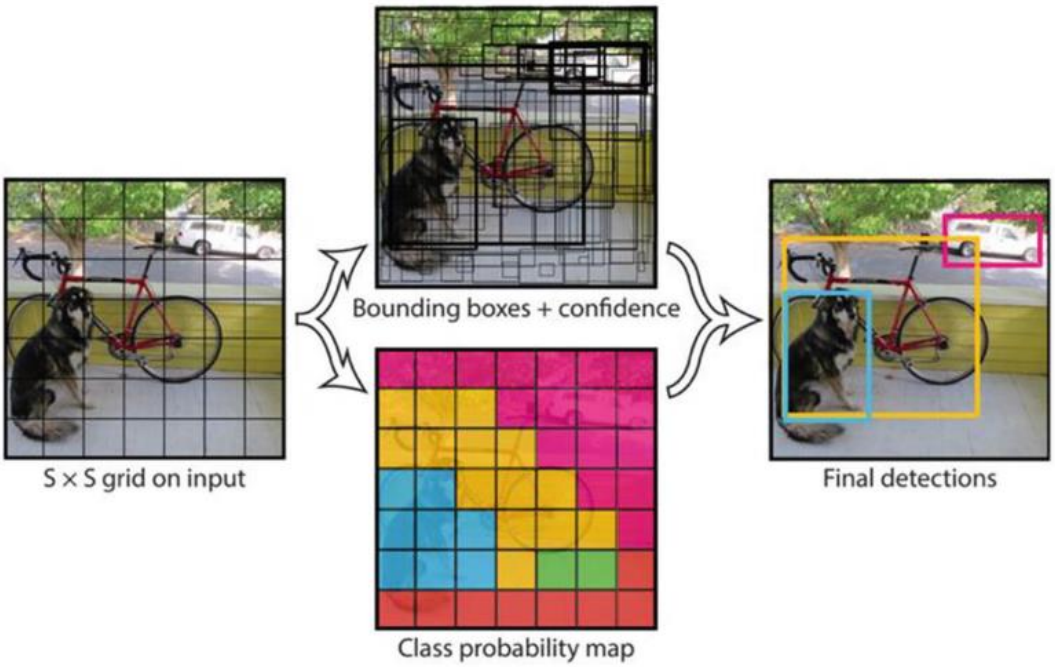
- Κάθε κελί προβλέπει έναν αριθμό B Bounding Boxes, τις πιθανότητες των αντικειμένων να ανήκουν σε μια τάξη, αλλά και τις προβλέψεις των τάξεων που ανήκει το αντικείμενο.
- Μάλιστα, αν το κεντρικό σημείο ενός αντικειμένου υφίσταται σε ένα κελί, τότε αυτό το κελί είναι υπεύθυνο για την πρόβλεψη του, καθώς και μόνο από τα αντίστοιχα κελιά απορρέουν τα output των αντικειμένων, δηλαδή τα Bounding Boxes και όλα όσα περιλαμβάνουν.
- Το Bounding Box περιλαμβάνει τέσσερα στοιχεία θέσης (x , y , w , h), ένα πέμπτο με την πιθανότητα Pc ύπαρξης του αντικειμένου σε ένα κελί και τις προβλέψεις για τις τάξεις Ci όπου :
    - (x , y) → Οι συντεταγμένες του κεντρικού pixel του Bounding Box. Οι τιμές κυμαίνονται από 0 έως και 1, διότι για κάθε κελί θεωρείται σημείο έναρξης πάνω αριστερά (0 , 0) και πέρατος κάτω δεξιά (1 , 1). Πρακτικά χρησιμοποιείται μία σιγμοειδής.
    - (w , h) → Το πλάτος και το ύψος του Bounding Box αντίστοιχα. Οι τιμές κυμαίνονται είτε > 1 είτε < 1.
        - Ύψος : στην περίπτωση που το αντικείμενο είναι ψηλότερο από το κελί, τότε οι τιμές κυμαίνονται > 1. Αντίθετα < 1 αν το αντικείμενο δεν είναι ψηλότερο.
        - Πλάτος : με όμοιο τρόπο αν το αντικείμενο είναι πλατύτερο από το κελί, τότε οι τιμές κυμαίνονται > 1. Αντίθετα < 1 αν το αντικείμενο δεν είναι πλατύτερο.
    - Πρόβλεψη τάξης Ci → Φανερώνει την ύπαρξη ή μη ενός αντικειμένου σε ένα Bounding Box.
    - Pc → Πιθανότητα ύπαρξης του αντικειμένου σε ένα κελί

Παρακάτω παρατίθεται ένα παράδειγμα που υποδεικνύει ένα κελί όπου περιέχει το κεντρικό σημείο ενός αντικειμένου :



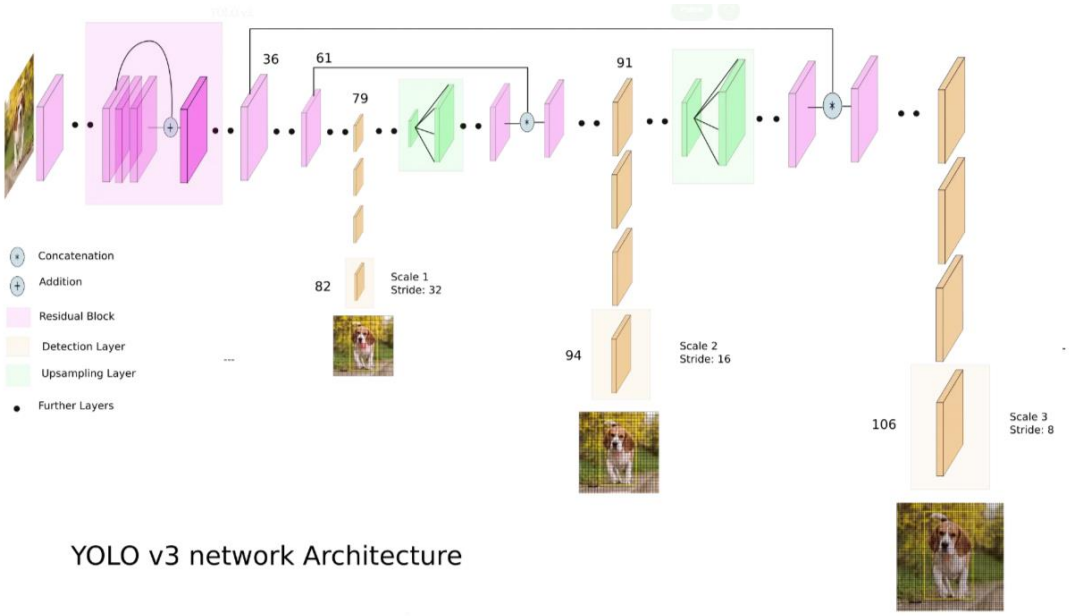
Εικόνα 2.2.5.2.1 -3 Λειτουργία YOLO (πηγή : Neelam, 2021*)*

Σε οπτικοποίηση της αναφερόμενης λειτουργίας προκύπτουν όλα τα αντικείμενα και τα Bounding Boxes ταξινομημένα βάσει της πιθανότητας Pc όπως γίνεται αντιληπτό στο παρακάτω σχήμα :

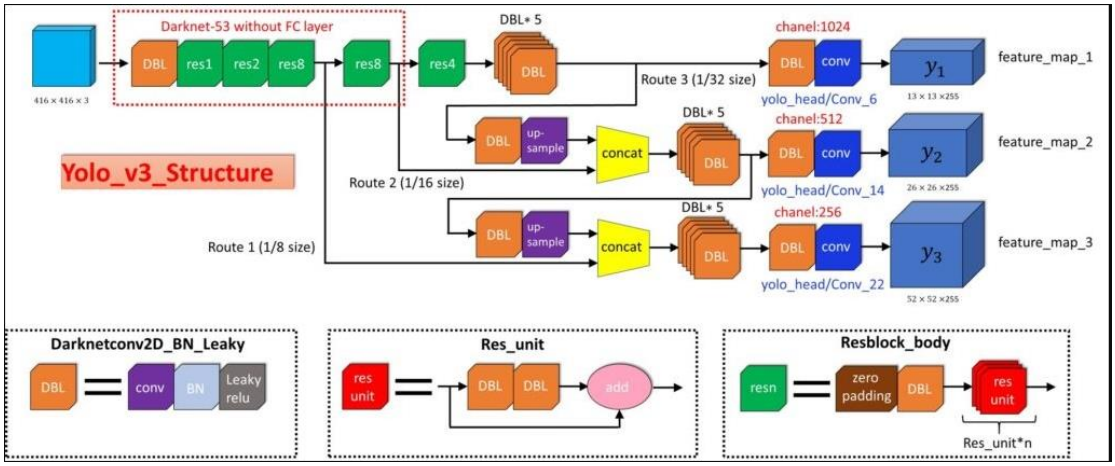


Εικόνα 2.2.5.2.1 -4 Οπτικοποίηση λειτουργίας αλγορίθμου (πηγή : (Verdhan, 2021a)*)*

Παρατίθεται η αρχιτεκτονική του αλγορίθμου :



Εικόνα 2.2.5.2.1 -3 Αρχιτεκτονική YOLOV3 (πηγή : (Neelam, 2021)*)*



Εικόνα 2.2.5.2.1 -4 Αρχιτεκτονική YOLOV3 πιο αναλυτικά (πηγή : (datahacker.rs, 2019)*)*

Ο αλγόριθμος χρησιμοποιεί μια Darknet53 δομή (Redmon & Farhadi, 2018). Το δίκτυο αποτελείται από 53 επίπεδα αλλά για την επίτευξη της διαδικασίας ανίχνευσης προστέθηκαν άλλα 53, με αποτέλεσμα να γίνουν 106 όπως φαίνεται και πάνω δεξιά (Εικόνα 2.2.5.2.1 -3).

Η δομή Darknet53 αποτελεί τα κουτιά στο κόκκινο πλαίσιο πάνω αριστερά (Εικόνα 2.2.5.2.1 -4). Η δομή φαίνεται παρακάτω :

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

Εικόνα 2.2.5.2.1 -5 Darknet53 δομή (πηγή : (Redmon & Farhadi, 2018))

Χρησιμοποιείται η δομή χωρίς τη σύνδεση στα fc δίκτυα.

Το Residual Block συναντάται στο συνελικτικό δίκτυο Resnet και συνεπώς έχει παρόμοια δομή. Αυτό σημαίνει πως το δίκτυο αποτελείται από δύο “κλαδιά”, το κύριο που είναι σειρά επιπέδων νευρωνικών δικτύων και το shortcut που η πληροφορία από το input πηγαίνει κατευθείαν στο output, ενώ η τελική activation function μπαίνει μετά το shortcut κλαδί = branch. Στην αναλυτικότερη αρχιτεκτονική το παραπάνω αποτελεί το κόκκινο κουτί (Res_unit). Φαίνεται πως περιέχει ένα shortcut κλαδί, όπου η πληροφορία πηγαίνει «πίσω» στην διαδικασία back propagation ώστε να ενημερωθούν τα αρχικά βάρη του δικτύου και στη συνέχεια αυτό να μπορέσει να

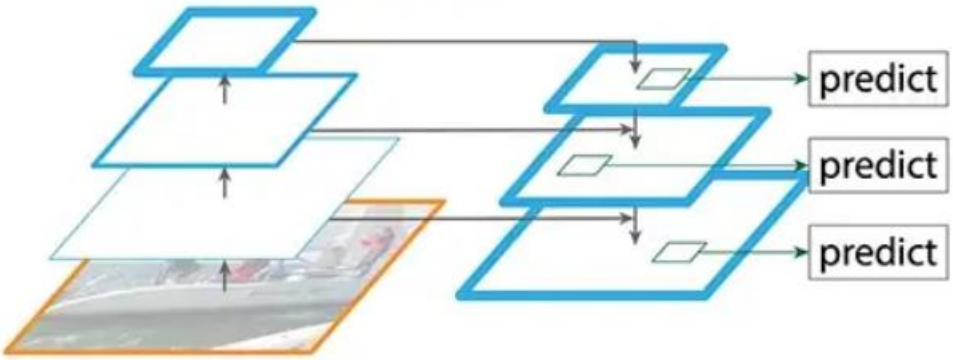
«μάθει». Επιπλέον, φαίνεται και το άλλο κλαδί που αποτελείται από δύο DBL κουτιά, τα οποία με τη σειρά τους αποτελούν σειρά νευρωνικών δικτύων. Πιο συγκεκριμένα, ένα συνελικτικό επίπεδο, ακολουθούμενο από batch normalization και στη συνέχεια η συνάρτηση ReLU για τη μη-γραμμικότητα.

Επιπλέον, στη Darknet53 δομή στο κόκκινο πλαίσιο περιέχονται και πράσινα κουτιά, τα οποία αποτελούνται από ένα DBL με μηδενικό padding (προστέθηκαν μηδενικά γύρω από την εικόνα κατά τη διαδικασία του kernel), καθώς και ένα Res_unit.

Πρακτικά στην πιο αναλυτική αρχιτεκτονική φαίνεται η αντιστοίχιση της δομής (Εικόνα 2.2.5.2.1 -3) με την Εικόνα 2.2.5.2.1 -4 πιο αναλυτικά σε σχέση με την Εικόνα 2.2.5.2.1 -3. Έπειτα από τις επεξηγήσεις της δομής στις εικόνες, επέρχεται η ανάλυση της αρχιτεκτονικής γενικότερα. Συνεχίζοντας στην Εικόνα 2.2.5.2.1 -3 παρατηρείται πως ο αλγόριθμος προβλέπει σε 3 διαφορετικές κλίμακες. Αυτό είναι πολύ σημαντικό, διότι το ίδιο αντικείμενο σε μια εικόνα μπορεί να έχει διαφορετικό μέγεθος και συνεπώς με το μοντέλο αυτό μπορεί να προβλεφθεί. Πιο συγκεκριμένα στα 106 συνελικτικά επίπεδα που αναφέρθηκαν παραπάνω οι ανιχνεύσεις διεξάγονται στο 82° , στο 94° και στο 106° επίπεδο (Εικόνα 2.2.5.2.1 -3). Με την πρώτη κλίμακα να έχει μέγεθος 13 x 13 και να χρησιμοποιείται για την ανίχνευση μεγάλου μεγέθους αντικειμένων, η δεύτερη με μέγεθος 26 x 26 να χρησιμοποιείται για την ανίχνευση μεσαίου μεγέθους αντικειμένων και η τρίτη κλίμακα με μέγεθος 52 x 52 για την ανίχνευση μικρού μεγέθους αντικειμένων (Εικόνα 2.2.5.2.1 -3). Σύμφωνα με την προαναφερόμενη εικόνα στην παρούσα παράγραφο τα αποτελέσματα αυτά προκύπτουν από input εικόνα 416 x 416.

Επομένως, βάσει αρχιτεκτονικής αρχικά εφαρμόζεται η δομή Darknet53 στην input εικόνα ή πλήθος εικόνων, στη συνέχεια γίνεται η ανίχνευση στις τρείς κλίμακες, έπειτα από τις οποίες φαίνονται στην Εικόνα 2.2.5.2.1 -3 κάποια πράσινα τετράγωνα. Αυτά αποτελούν ένα upsample επίπεδο, μέσω του οποίου αυξάνεται το μέγεθος του output από το input.

Παρατηρείται πως για την δεύτερη και τρίτη κλίμακα υφίστανται δύο «πορείες», εισέρχονται δηλαδή πληροφορίες από δύο πλευρές, για αυτό και χρησιμοποιείται η λεγόμενη concatenation, δηλαδή μία δίοδος ώστε η πληροφορία από τις δύο πλευρές να συνεχίζει την πορεία της σε μία κατεύθυνση. Ειδικότερα, η πληροφορία έρχεται μετά από την εφαρμογή κάθε κλίμακας (σειρά συνελικτικών επιπέδων) καθώς και από skip συνδέσεις, οι οποίες έχουν τοποθετηθεί για να βοηθήσουν στην καλύτερη πρόβλεψη της τοποθεσίας των πλαισίων. Επίσης ως skip συνδέσεις διευκολύνουν την εκπαίδευση (όπως και στο Resnet, δηλαδή είναι ένα shortcut κλαδί). Αυτή η μορφή με τις αντίστοιχες δύο «πορείες» αποτελεί μια δομή FPN, η οποία χρησιμοποιείται στην ανίχνευση αντικειμένων για να αυξηθεί όσο γίνεται η ακρίβεια σύμφωνα με το άρθρο «Understanding Feature Pyramid Networks for object detection (FPN)» του (Hui, 2018b). Πρακτικά η δομή αυτή διαμορφώνει feature maps πολλαπλής κλίμακας με πληροφορίες καλύτερης ποιότητας. Για αυτό χρησιμοποιείται και στο YOLOV3. Παρακάτω φαίνεται μία δομή FPN :



Εικόνα 2.2.5.2.1 -6 Δομή FPN (πηγή : (Hui, 2018a)*)*

Στη συνέχεια, έπειτα από την εφαρμογή της δομής FPN, μέσα από συνελικτικά επίπεδα γίνεται η ανίχνευση σε κάθε κλίμακα. Με λίγα λόγια : **εικόνα → Darknet53 (δομή παρόμοια με το Resnet) → δομή FPN → ανίχνευση αντικειμένων σε τρείς κλίμακες.**

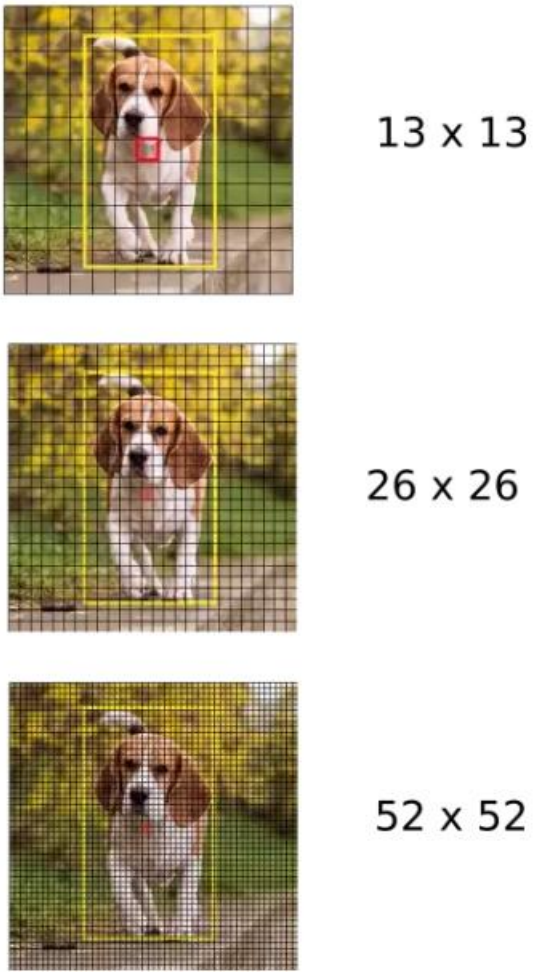
Περισσότερες λεπτομέρειες για την αρχιτεκτονική (Redmon & Farhadi, 2018) :

- Περιέχονται 53 συνελικτικά επίπεδα, ακολουθούμενα με batch normalization και ReLU, όπως προαναφέρθηκε.
- Όπως αποφαίνεται από την Εικόνα 2.2.5.2.1 -3 κατά την εφαρμογή κάθε συνελικτικού επιπέδου (convolutional) χρησιμοποιούνται πολλά φίλτρα στις εικόνες και παράγονται πολλά feature maps (outputs).
- Δεν εφαρμόζεται καθόλου pooling επίπεδο και συνεπώς για να μειωθεί το μέγεθος των feature maps εφαρμόζονται συνελικτικά επίπεδα με stride δύο (2) (το φίλτρο κινείται με βήμα (2) στην εικόνα). Φαίνεται στις τρείς κλίμακες, όπου χρησιμοποιείται βήμα 32, 16, 8 αντίστοιχα (Εικόνα 2.2.5.2.1 -3). Αυτό συνεπάγεται το output μέγεθος σε τρία σημεία να είναι μικρότερο του input.
- Αποτρέπει την απώλεια στοιχείων χαμηλού επιπέδου, όπου ένα pooling επίπεδο απορρίπτει. Με αποτέλεσμα τη βελτίωση της πρόβλεψης μικρών αντικειμένων.

Περισσότερες λεπτομέρειες ως προς τη λειτουργία γνωρίζοντας πλέον την αρχιτεκτονική : ξεκινώντας από την αρχή, δηλαδή από την input εικόνα ή πλήθος εικόνων. Έστω πλήθος εικόνων (n , 416 , 416, 3), όπου n = αριθμός εικόνων, (416 , 416) = (width = πλάτος, height = ύψος εικόνων), 3 = 3 κανάλια RGB.

Ειδικότερα, το ύψος και το πλάτος μπορούν να πάρουν τιμές 416 x 416, 608 x 608 1024 x 1024 και γενικότερα τιμές που διαιρούνται με το 32. Αυτό είναι απαραίτητο, διότι αν διαιρούνται με το 32 θα διαιρούνται και με το 16 αλλά και το 8, κάτι σημαντικό αφού αυτά είναι τα βήματα στις κλίμακες (Redmon & Farhadi, 2018) (Neelam, 2021).

Παρατίθενται σε επόμενο στάδιο η οπτικοποίηση των αποτελεσμάτων των τριών κλιμάκων, όπου γίνεται αντιληπτή η ανίχνευση των μεγάλων, μεσαίων, μικρών αντικειμένων από τις αντίστοιχες κλίμακες :



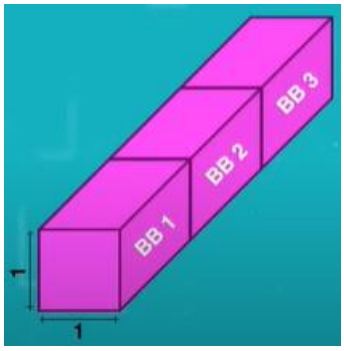
Εικόνα 2.2.5.2.1 -7 Ανίχνευση μεγάλων, μεσαίων, μικρών αντικειμένων από τις αντίστοιχες κλίμακες 13 x 13, 26 x 26, 52 x 52 (πηγή : (Neelam, 2021)*)*

Στη συνέχεια κατά την διαδικασία της ανίχνευσης (detector επίπεδο στην Εικόνα 2.4.1.2 -1) ο αλγόριθμος YOLOV3 εφαρμόζει 1 x 1 φίλτρα kernels στα τρία output επίπεδα. Με αυτόν τον τρόπο μειώνεται το μέγεθος στις συγκεκριμένες χωρικές διαστάσεις : 13 x 13, 26 x 26, 52 x 52. Εκτός από το μέγεθος των φίλτρων υπάρχει και το βάθος, το οποίο υπολογίζεται με τον εξής τρόπο : (b * (5 + c)), όπου : b = αριθμός Bounding Boxes, c = αριθμός τάξεων. Στον αλγόριθμο υπάρχει ένα σύνολο δεδομένων το COCO, το οποίο περιέχει 80 τάξεις ήδη εκπαιδευμένες. Μπορεί να χρησιμοποιηθεί το σύνολο δεδομένων αυτό κατευθείαν για ανίχνευση μίας ή και περισσότερων τάξεων (Redmon & Farhadi, 2018) (Neelam, 2021).

Όσων αφορά το τμήμα (5 + c), αυτό εκρέει από το γεγονός πως κάθε Bounding Box περιέχει 5 + c ιδιότητες. Ειδικότερα, όπως αναφέρεται παραπάνω τέσσερα στοιχεία : (x , y , w , h), πέμπτο : την πιθανότητα Pc και τις προβλέψεις των τάξεων Ci = c.

Ο YOLOV3 προβλέπει 3 Bounding Boxes για κάθε κελί σε κάθε μία από τις 3 κλίμακες. Συνεπώς, στην περίπτωση λοιπόν του COCO το βάθος των φίλτρων είναι : b = 3, c = 80 και η συνάρτηση υπολογίζεται ως : (3 * (5 + 80)) = 255 ιδιότητες. Οπότε τα feature maps των 3 κλιμάκων έχουν μέγεθος (13 x 13 x 255), (26 x 26 x 255), (52 x 52 x 255). Από την Εικόνα 2.4.1.2 -5 φαίνεται πως τα outputs των 3 κλιμάκων αποτελούν καννάβους, που τα κελιά τους προφανώς είναι κελιά ανίχνευσης. Επομένως, ο αλγόριθμος σκοπεύει να αναγνωρίσει σε ποιο κελί βρίσκεται το κεντρικό σημείο του

αντικειμένου (Redmon & Farhadi, 2018) (Neelam, 2021). Παρακάτω φαίνεται το σχήμα ενός φίλτρου:



Εικόνα 2.2.5.2.1 -8 (1 x 1) φίλτρα kernels – σχήμα
(πηγή : (Valentyn Sichkar, 2021)*, youtube.com)*

Όπως αναφέρθηκε παραπάνω, το κελί στο οποίο βρίσκεται το κεντρικό σημείο του αντικειμένου είναι υπεύθυνο για την ανίχνευσή του και αυτό συμβαίνει διότι, κατά την εκπαίδευση υφίσταται ένα «αληθινό» Bounding Box (ground truth Bounding Box) που ανιχνεύει ένα αντικείμενο. Συνεπώς, το YOLOV3 πρέπει να ανακαλύψει σε ποια κελιά βρίσκεται αυτό το Bounding Box και για αυτό προβαίνει σε προβλέψεις σε 3 κλίμακες. Αυτός λοιπόν είναι και ο λόγος που το κελί με το κεντρικό σημείο ευθύνεται για την ανίχνευση. Επειδή ήδη το αντικείμενο πρακτικά βρίσκεται στο κέντρο του «αληθινού» Bounding Box, επομένως κάτι αντίστοιχο πρέπει να συμβαίνει και στο προβλεπόμενο Bounding Box (Redmon & Farhadi, 2018) (Neelam, 2021).

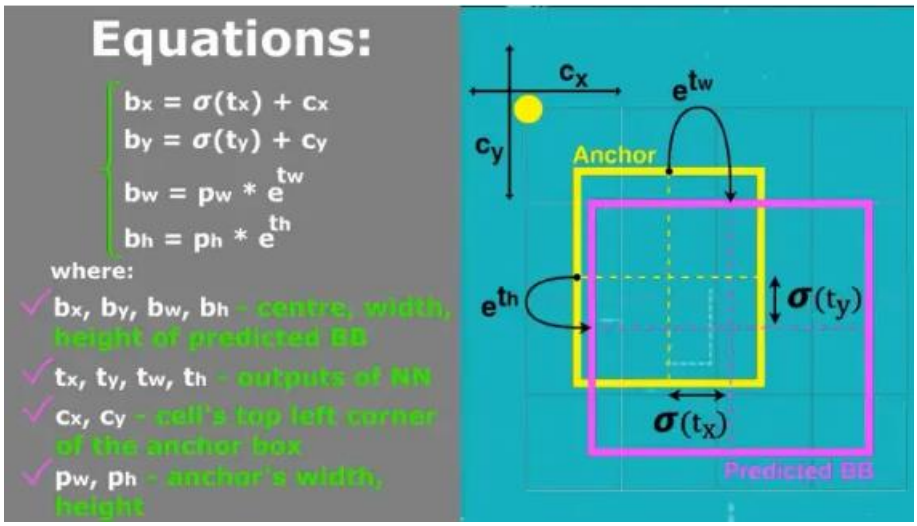
Γνωρίζοντας πλέον τι προβλέπει και γιατί ένα Bounding Box, επόμενο στάδιο είναι η πρόβλεψη του τελικού Bounding Box. Για τις προβλέψεις των Bounding Boxes ο αλγόριθμος χρησιμοποιεί άλλα προκαθορισμένα Bounding Boxes που ονομάζονται anchor boxes. Αυτά χρησιμοποιούνται και για την εκτίμηση του «αληθινού» πλάτους και ύψους για το προβλεπόμενο Bounding Box. Χρησιμοποιούνται 3 anchor boxes για κάθε κλίμακα, που συνεπάγεται 9 στο σύνολο. Συμπερασματικά, ο κάθε κάνναβος των κλιμάκων προβλέπει 3 Bounding Boxes κάνοντας χρήση 3 anchor boxes. Για τον υπολογισμό των Bounding Boxes αυτών εφαρμόζεται K-Means Clustering (ένας αλγόριθμος που χωρίζει σε K τμήματα μία παρατήρηση), με αποτέλεσμα τα w,h των anchor boxes να είναι τα παρακάτω (Redmon & Farhadi, 2018) :

- Κλίμακα 1 : (116 x 90), (156 x 198), (373 x 326)
- Κλίμακα 2 : (30 x 61), (62 x 45), (59 x 119)
- Κλίμακα 3 : (10 x 13), (16 x 30), (33 x 23)

Συνεπώς τα προβλεπόμενα Bounding Boxes :

- Για την πρώτη κλίμακα είναι 13 x 13 x 3 = 507
- Για την δεύτερη 26 x 26 x 3 = 2028
- Για την τρίτη 52 x 52 x 3 = 8112
- Ενώ συνολικά 10847 Bounding Boxes προβλέπονται από τον αλγόριθμο.

Επόμενο βήμα η πρόβλεψη του «αληθινού» w και h των Bounding Boxes. Το YOLOV3 εκτιμά μετατοπίσεις σε anchor boxes και όχι ακριβείς συντεταγμένες του κέντρου των Bounding Boxes. Επίσης, το πλάτος και το ύψος τους έχουν κανονικοποιηθεί με το «αληθινό» ύψος και πλάτος. Για αυτό το λόγο εφαρμόζεται σιγμοειδής στις παραγόμενες συντεταγμένες του κέντρου tx, ty, ώστε το κελί με το κεντρικό σημείο να είναι υπεύθυνο για την ανίχνευση του (Redmon & Farhadi, 2018) Οι αντίστοιχες εξισώσεις φαίνονται παρακάτω :



Εικόνα 2.2.5.2.1 -9 Εξισώσεις εύρεσης «αληθινού» ύψους και πλάτους των Bounding Boxes (πηγή (Neelam, 2021)*)*

Όπως φαίνεται παραπάνω :

- ✓ bx, by, bw, bh → κεντρικό σημείο, πλάτος και ύψος των προβλεπόμενων Bounding Boxes
- ✓ tx, ty, tw, th → αποτελέσματα από το νευρωνικό δίκτυο
- ✓ cx, cy → συντεταγμένες πάνω αριστερά του κελιού του anchor box
- ✓ pw, ph → πλάτος και ύψος του anchor box

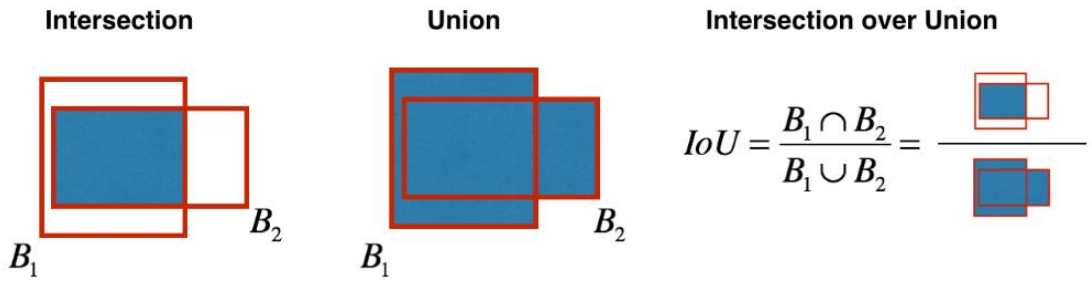
Επομένως, για την εύρεση των συντεταγμένων του κεντρικού σημείου του Bounding Box χρησιμοποιούνται οι πρώτες δύο εξισώσεις, που συνεπάγεται πως ο YOLOV3 εφαρμόζει σιγμοειδούς συνάρτηση, μέσα από την οποία προκύπτουν τα tx, ty. Με αυτά στη συνέχεια, βάζοντάς τα στις παραπάνω εξισώσεις, καταλήγει στις συντεταγμένες του κεντρικού σημείου των Bounding Boxes.

Συμπληρωματικά, με τις υπόλοιπες εξισώσεις, έχοντας γνωστές τις αντίστοιχες μεταβλητές λόγω του υπολογισμού των anchor boxes προηγουμένως, εκτιμώνται και τα ύψος και πλάτος των προβλεπόμενων Bounding Boxes.

Την παρούσα στιγμή είναι γνωστές οι συντεταγμένες του κεντρικού σημείου των Bounding Boxes, το πλάτος και το ύψος τους. Όμως, υπάρχουν ακόμα 10847 Bounding Boxes συνολικά. Σκοπός είναι η πρόβλεψη ενός τελικού μοναδικού Bounding Box για κάθε αντικείμενο. Οπότε, πρέπει με κάποιον τρόπο να απορριφθούν Bounding Boxes. Ένας τρόπος είναι υιοθέτηση κατωφλίου στις προβλέψεις βάσει της πιθανότητας Pc. Πρακτικά, όσα πλαίσια βρίσκονται κάτω από το κατώφλι θα απορρίπτονται. Στη συνέχεια, γίνεται χρήση του φίλτρου NMS – Non Maximal Suppression. Ο YOLOV3 χρησιμοποιεί το Non - Maximal Suppression φίλτρο για να κρατήσει μόνο το καλύτερο Bounding Box. Πρακτικά, απορρίπτει τα «αδύναμα» πλαίσια, ενώ ευθύνεται για την ανίχνευση ενός αντικειμένου μόνο μια φορά, δηλαδή απορρίπτει τα διπλά Bounding Boxes. Η παραπάνω διαδικασία γίνεται επίσης με τη χρήση ενός κατωφλίου, σύμφωνα με το οποίο όσα Bounding Boxes παρουσιάζουν πιθανότητα ανίχνευσης κάτω από το κατώφλι απορρίπτονται. Το παραπάνω φίλτρο χρησιμοποιεί μια σημαντική συνάρτηση, την Intersection over Union ή IoU (τομή / ένωση). Η συγκεκριμένη συνάρτηση φανερώνει την επικάλυψη του προβλεπόμενου Bounding Box με το «αληθινό», που σημαίνει πως όσο πιο κοντινή είναι η απόσταση τους, τόσο καλύτερη ακρίβεια της πρόβλεψης. Σε σχέση με το προηγούμενο φίλτρο, έπειτα από την απόρριψη των πλαισίων με χαμηλή πιθανότητα ανίχνευσης, η επόμενη φάση στο φίλτρο αυτό αποτελεί η επιλογή των Bounding Boxes με τη μεγαλύτερη

πιθανότητα ανίχνευσης και η απόρριψη των Bounding Boxes όπου η τιμή της συνάρτησης είναι μεγαλύτερη από ένα κατώφλι της συνάρτησης. Αυτό σημαίνει πώς όλα τα προβλεπόμενα Bounding Boxes που η τιμή της συνάρτησης είναι μεγαλύτερη από το κατώφλι σε σχέση με τα καλύτερα Bounding Boxes απορρίπτονται. Έτσι προκύπτει ένα τελικό μοναδικό Bounding Box για κάθε αντικείμενο (Redmon & Farhadi, 2018).

Παρατίθεται παρακάτω η μορφή της συνάρτησης Intersection over Union :



Εικόνα 2.2.5.2.1 -10 Η συνάρτηση Intersection over Union (πηγή : (PyLessons, 2020)*)*

Επιπλέον σημαντικό κομμάτι του αλγορίθμου αποτελεί η συνάρτηση loss. Η συνάρτηση βελτιστοποιείται κατά την εκπαίδευση (Redmon et al., 2016b) όπως αναλύεται παραπάνω στην λειτουργία των νευρωνικών δικτύων και παρατίθεται παρακάτω :

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Εικόνα 2.2.5.2.1 *-11 Loss function yolov3 (πηγή :* (Redmon et al., 2016a)*)*

Όπου σύμφωνα με τους (Redmon et al., 2016b):

- 1^{obj}_{i} αφορά στο αν υφίσταται το αντικείμενο στο κελί i
- 1^{obj}_{ij} αναφέρεται στο κελί του j Bounding Box που ευθύνεται για την ανίχνευση
- 1^{noobj}_{ij} είναι συμπληρωματικό του 1^{obj}_{ij}
- λ_{coord} ευθύνεται για την αύξηση βάρους στις συντεταγμένες του Bounding Box
- λ_{noobj} μειώνει το βάρος στη συνάρτηση όταν ανιχνεύεται background
- Cˆi αποτελεί την πιθανότητα δημιουργίας του j Bounding Box στο κελί i

2.2.5.2.2 Αξιολόγηση μοντέλου

Με το πέρας της εκπαίδευσης του μοντέλου υπολογίζεται η μέση ακρίβεια mean average precision (mAP). Χρησιμοποιείται για να αξιολογήσει το μοντέλο. Ειδικότερα, συγκρίνει το αληθινό Bounding Box με το προβλεπόμενο και επιστρέφει μια τιμή. Όσο μεγαλύτερη είναι αυτή η τιμή τόσο πιο ακριβές είναι το μοντέλο στις ανιχνεύσεις . Είναι προφανές λοιπόν, σύμφωνα με το προηγούμενο υποκεφάλαιο 2.2.4.2.1, πως η

ακρίβεια ισούται θα μπορούσε να πει κανείς με την τιμή της συνάρτηση Intersection over Union (PyLessons, 2020). Συνεπώς και η συνάρτηση αυτή φανερώνει ακρίβεια. Πιο συγκεκριμένα, αν η τιμή είναι πάνω ή ίση του 0.5 τότε η ανίχνευση είναι καλή.

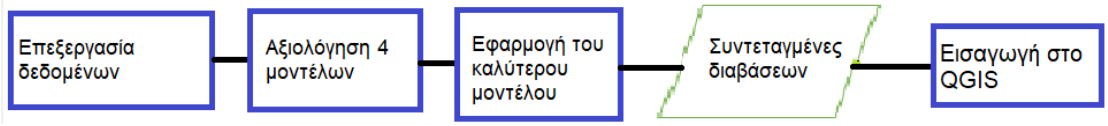
Το mAP δημιουργείται με τη βοήθεια της Intersection over Union. Στην ουσία οριοθετείται ένα κατώφλι για παράδειγμα > = 0.5. Εάν η τιμή της συνάρτησης είναι μεγαλύτερη από το κατώφλι τότε η ανίχνευση θεωρείται αληθής και θετική (TP). Αν είναι λιγότερη από το κατώφλι τότε θεωρείται ψευδής και θετική (FP). Εάν η πρόβλεψη είναι ψευδής ενώ το αντικείμενο υφίσταται πραγματικά, τότε η ανίχνευση θεωρείται ψευδής αρνητική (FN). Για τον υπολογισμό της ακρίβειας χρειάζονται οι υπολογισμοί του precision και recall για όλα τα αντικείμενα στις εικόνες (PyLessons, 2020) όπου :

- precision = TP / (TP + FP)
- recall = TP / (TP + FN)

Ως προς το κομμάτι της μεθοδολογίας και αξιολόγησης (κεφάλαιο 3) η εύρεση του mAP γίνεται πλέον με μία εντολή κατά τη διάρκεια της εκπαίδευσης του μοντέλου (PyLessons, 2020).

3. Μεθοδολογία και Αξιολόγηση

Η διαδικασία της μεθοδολογίας και αξιολόγησης που περιγράφεται στο παρών κεφάλαιο περιγράφεται από το παρακάτω διάγραμμα ροής :



Εικόνα 3 -1 Διάγραμμα ροής εργασιών για την υλοποίηση της μεθοδολογίας αυτής της Διπλωματικής Εργασίας

Ειδικότερα, η διαδικασία έχει ως εξής :

- Αφού συλλεχθούν τα δεδομένα (δορυφορικές εικόνες στην παρούσα εργασία) επεξεργάζονται ως προς το μέγεθος αλλά και ως προς άλλες απαιτήσεις των μοντέλων που διερευνώνται στη συνέχεια.
- Έπειτα, πραγματοποιείται αξιολόγηση των 4 μοντέλων (ZF Net, Resnet 50, Resnet απλοποιημένο και το YOLOV3) μέσω σύγκρισης της ακρίβειας της εκπαίδευσης των μοντέλων.
- Ύστερα, εφαρμόζεται το καλύτερο μοντέλο ως προς ανίχνευση διαβάσεων.
- Μετά υπολογίζονται οι συντεταγμένες των διαβάσεων που ανιχνεύονται.
- Τέλος, οι παραπάνω συντεταγμένες εισάγονται στο λογισμικό QGIS.

Παρακάτω παρουσιάζεται εκτενώς η πάνω διαδικασία.

3.1 Ψηφιακό Περιβάλλον

Κατά την εφαρμογή χρησιμοποιήθηκε η γλώσσα προγραμματισμού python. Η γλώσσα αυτή είναι η καλύτερη για την τεχνητή νοημοσύνη, διότι χρησιμοποιείται ευρέως επειδή είναι εύκολη στο να μαθευτεί, με λίγο συντακτικό και συνοδεύεται από ένα μεγάλο πλήθος ενσωματωμένων βιβλιοθηκών (Turing, 2023). Οι περισσότερες από τις βιβλιοθήκες αυτές είναι για χρήση σε εφαρμογές τεχνητής νοημοσύνης και μηχανικής μάθησης. Χρησιμοποιήθηκε python 3.7, η οποία εγκαταστάθηκε μέσω της πλατφόρμας Anaconda. Ως ολοκληρωμένο περιβάλλον ανάπτυξης ορίστηκε το Spyder, το οποίο λήφθηκε από το Anaconda με εγκατεστημένες ήδη πολλές βιβλιοθήκες. Η κύρια βιβλιοθήκη που χρησιμοποιήθηκε και η οποία είναι σημαντική για την υπολογιστική όραση είναι η Open - Cv. Άλλες βιβλιοθήκες που χρησίμευσαν είναι η TensorFlow, NumPy (για πράξεις πινάκων), καθώς και η blob (για την μετατροπή μιας εικόνας σε σύνολο δεδομένων).

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο η δομή που αναπτύχθηκε το YOLOV3 είναι η Darknet53. Ο Alexey Bochkovskiy δημιούργησε έναν αναλυτικό οδηγό για χρήση μοντέλων YOLO σε εφαρμογές, χρησιμοποιώντας ο καθένας τα δικά του δεδομένα. Ο οδηγός αυτός βρίσκεται στο GitHub (Αμερικάνικη εταιρία, θυγατρική της Microsoft, στην οποία παρέχεται «φιλοξενία» για έκδοση λογισμικού ανάπτυξης καθώς και ο έλεγχός του). Επίσης, από το GitHub λήφθηκε και ένας «κλώνος» της δομής Darknet, με αποτέλεσμα τη δημιουργία ενός φακέλου με όνομα darknet και όλα τα αρχεία που περιέχονται στο GitHub (Bochkovskiy Alexey, n.d.). Σύμφωνα λοιπόν με τον σχετικό οδηγό, πραγματοποιήθηκαν οι απαραίτητες αλλαγές στον κώδικα πριν την εκπαίδευση και στη συνέχεια ανίχνευση των διαβάσεων στις δύο πόλεις. Οι αλλαγές αφορούσαν την συμπερίληψη του GPU (αναλύεται παρακάτω), της βιβλιοθήκης βαθιών νευρωνικών δικτύων CUDNN και της βιβλιοθήκης OPENCV.

Ειδικότερα, η εκπαίδευση πραγματοποιήθηκε με ήδη εκπαιδευμένα βάρη (darknet53.conv.74), τα οποία λήφθηκαν από το GitHub, διότι έχουν ήδη εκπαιδευτεί σε μεγάλα σύνολα δεδομένων. Επίσης, λήφθηκαν με αντίστοιχο τρόπο και από την ίδια πλατφόρμα βάρη για τον έλεγχο (testing) (yolov3.weights). Με όμοιο τρόπο ακόμη, αποκτείται ένας cfg φάκελος, ο φάκελος δηλαδή που περιέχει τη σύνθεση της Darknet53 δομής. Αξιοσημείωτες είναι οι αλλαγές στον φάκελο αυτό που αφορούν το πλήθος των φίλτρων, τις τάξεις και το max_batches. Βάσει οδηγού αλλά και της παραπάνω αναλυμένης εξίσωσης : (b * (5 + c)), τα φίλτρα είναι 18. Πιο συγκεκριμένα τα πλαίσια είναι 3 στον αλγόριθμο αυτόν, ζητείται ανίχνευση ενός μόνο αντικειμένου, οπότε c = 1, που οδηγεί στο (3 * (5 + 1)) = 18 φίλτρα. Το max_batches σύμφωνα με τον οδηγό του Alexey Bochkovskiy υπολογίζεται από την εξίσωση : αριθμός τάξεων * 2000, αλλά να μην είναι λιγότερο από τον αριθμό των εικόνων που χρησιμοποιούνται στην εκπαίδευση, καθώς και ούτε μικρότερο από 6000. Στην παρούσα περίπτωση η τάξη είναι μία, που σημαίνει 1 * 2000 = 2000. Όμως, πρέπει η τιμή να είναι τουλάχιστον 6000, και μιας που οι εικόνες για την εκπαίδευση όπως θα αναλυθεί παρακάτω είναι λιγότερες από 2000, το max_batches εντέλει τέθηκε 6000. Επιπλέον να τονιστεί πως ο φάκελος της δομής αφορά 416 x 416 εικόνες.

Η εκπαίδευση γίνεται στο Google colab, διότι παρέχει άδεια για χρήση GPU, το οποίο είναι πολύ σημαντικό, διότι αυξάνει την ταχύτητα της εκπαίδευσης. Το αποτέλεσμα είναι κάποια βάρη, τα οποία χρησιμοποιούνται μετέπειτα στην ανίχνευση.

Η ανίχνευση γίνεται στο Spyder, αφού δεν χρειάζεται πλέον GPU. Κατά την ανίχνευση ο κώδικας βασίζεται στο θεωρητικό υπόβαθρο του προηγούμενου κεφαλαίου.

Για τα ZF – Net και Resnet -50 δίκτυα χρησιμοποιούνται αντίστοιχες βιβλιοθήκες με την εκπαίδευση και την ταξινόμηση να λαμβάνουν χώρα επίσης στο Google colab χωρίς χρήση του GPU.

Η εκπαίδευση, η ανίχνευση και τα αποτελέσματα αναλύονται παρακάτω εκτενέστερα.

3.2 Συλλογή και Επεξεργασία Δεδομένων

Η διπλωματική εργασία ασχολείται με την ανίχνευση διαβάσεων σε δύο πόλεις της Αγγλίας. Πιο συγκεκριμένα οι περιοχές μελέτης είναι οι πόλεις Μπρίστολ και Μάντσεστερ.

Στο Μπρίστολ πραγματοποιείται ανίχνευση διαβάσεων με διαγράμμιση ζέβρα. Η πόλη κατά κύριο λόγο είναι γεμάτη με τέτοιου είδους διαβάσεων. Το Μάντσεστερ αντιθέτως, παρατηρείται πως περιέχει πολύ λίγες διαβάσεις με διαγράμμιση ζέβρα, ενώ σε μεγάλο βαθμό κυριαρχούν διαβάσεις που διαγράφονται με δύο διακεκομμένες παράλληλες λευκές γραμμές καθώς παρουσιάζουν έναν κόκκινο χρωματισμό στην επιφάνεια του οδοστρώματος. Μάλιστα, οι διαβάσεις αυτές συναντώνται κατά κύριο λόγο σε σημεία πόλους έλξης πεζών, όπως πάρκα και πλατείες (χώροι αναψυχής). Επομένως σύμφωνα με τον σκοπό της εργασίας, αυτά τα είδη διαβάσεων στις δύο αυτές πόλεις, επιλέχθηκαν να τεθούν ως προς ανίχνευση, και ενδεχομένως στη συνέχεια να χαρτογραφηθούν οι συντεταγμένες τους. Ενδεχομένως, διότι η χαρτογράφηση μπορεί να πραγματοποιηθεί έπειτα από την ανίχνευση στα δίκτυα ZF – Net και Resnet -50. Πρακτικά αρχικά γίνεται μια σύγκριση στην ακρίβεια των αποτελεσμάτων για να επέλθει ύστερα η χαρτογράφηση.

Για την ανίχνευση στα δίκτυα ZF – Net και Resnet -50 βρέθηκαν και λήφθηκαν χειροκίνητα δορυφορικές εικόνες διαβάσεων με διαγράμμιση ζέβρα στην πόλη του Μπρίστολ καθώς και μη – διαβάσεων εικόνες από το υπόβαθρο της Google earth στο λογισμικό QGIS, ανάλυσης 10 εκατοστών. Χρησιμοποιήθηκαν αυτές οι εικόνες που απεικονίζουν διαβάσεις που περιτριγυρίζουν τους χώρους αναψυχής λόγω της σημασίας τους. Στη συνέχεια, οι εικόνες κόπηκαν με τέτοιο τρόπο ώστε να προκύψουν εικόνες διαβάσεων και μη με μέγεθος 224 x 224, όπως απαιτείται από τα δίκτυα δίκτυα ZF – Net και Resnet -50. Οι εικόνες με διαβάσεις γύρω από τους χώρους αναψυχής ήταν 80 και οι μη 300. Η σύγκριση με το YOLOV3 έγινε για την πόλη του Μπρίστολ. Έπειτα από τη σύγκριση, που τα αποτελέσματά της φαίνονται παρακάτω, ο τελικός αλγόριθμος εφαρμόζεται και στις δύο πόλεις. Σύμφωνα με το θεωρητικό υπόβαθρο καλή ακρίβεια επέρχεται με δείγμα 1000 εικόνων. Επομένως με την διαδικασία augmentation αυξάνονται οι εικόνες σε 1000 για τις διαβάσεις και 1000 οι μη. Το σύνολο δεδομένων αυτό λοιπόν παρουσιάζει ίδιες εικόνες σε στροφή όπως φαίνεται παρακάτω :

Εικόνα 3.2 -1 Διάβαση στο Μπρίστολ 224 x 224 σε στροφή

Στη συνέχεια το μεγάλο αυτό σύνολο δεδομένων χωρίστηκε σε δύο : train (80 % των εικόνων) και test (20 % των εικόνων).

Για την ανίχνευση με YOLOV3 από την άλλη πλευρά λήφθηκαν εικόνες και για τις δύο πόλεις. Πιο συγκεκριμένα, οι εικόνες που λήφθηκαν από το Μπρίστολ κόπηκαν 224 x 224 για τις τεχνικές που αναφέρονται στην προηγούμενη παράγραφο, όμως για το YOLOV3 υπάρχει απαίτηση μεγέθους που διαιρείται με το 32 (όπως αναφέρεται παραπάνω), οπότε επιλέγεται το 416 x 416 και επομένως κόπηκαν οι εικόνες σε αυτό το μέγεθος. Σύμφωνα με τον οδηγό για τη χρήση του μοντέλου προτείνεται η χρήση 2000 με 3000 εικόνων διαβάσεων. Στην εφαρμογή αυτή γίνεται εκπαίδευση με μόνο 200 εικόνες για κάθε πόλη σε μορφή jpg. Να τονιστεί πως οι εικόνες διαβάσεων που χρησιμοποιούνται στην ανίχνευση για την εκπαίδευση δεν αφορούν μόνο χώρους αναψυχής. Να σημειωθεί πως οι 200 εικόνες για κάθε περιοχή είναι μοναδικές και απεικονίζουν διαβάσεις με ποικίλες κλίσεις και στροφές. Ακόμα υφίστανται εικόνες που εμφανίζουν διαβάσεις υπό σκιά, υπό ήλιο καθώς και κάποιες δεν ήταν καλά συντηρημένες. Δηλαδή ναι μεν οι εικόνες είναι 200, αρκετά λιγότερες από την πρόταση των 2000, αλλά συμπεριλαμβάνονται όλες οι «καταστάσεις» μίας διάβασης. Η διαφορά είναι πως για τον αλγόριθμο αυτόν δεν χρησιμοποιούνται μόνο εικόνες διαβάσεων γύρω από χώρους αναψυχής. Με αυτόν τον τρόπο συγκρίνεται και η ανίχνευση με εκπαίδευση ίδιων εικόνων (augmentation) στα προηγούμενα δίκτυα αλλά και ξεχωριστών εικόνων στο συγκεκριμένο αλγόριθμο στην εκπαίδευση. Και οι τρείς τεχνικές θεωρούνται καλές για ανίχνευση επομένως αυτός είναι άλλος ένας τρόπος σύγκρισης τους, διότι η χρήση διαφορετικών ή μη εικόνων δε θα μεταβάλει την

ακρίβεια εκπαίδευσης αλλά ενδέχεται να μεταβάλει το βαθμό ανίχνευσης. Παρακάτω παρατίθενται δύο από τις 200 εικόνες :

Εικόνα 3.2 -2 Διάβαση στο Μπρίστολ 416 x 416

Εικόνα 3.2 -3 Διάβαση στο Μάντσεστερ 416 x 416

Στη συνέχεια, για την ανίχνευση ενός αντικειμένου με τη χρήση του YOLOV3 (Redmon & Farhadi, 2018), διεξάγεται η διαδικασία του labeling ή αλλιώς annotation. Βάση της θεωρίας λοιπόν, επέρχεται σύγκριση μεταξύ του προβλεπόμενου Bounding Box με το αληθινό. Όμως για να λάβει χώρα αυτή η σύγκριση, το σύστημα με κάποιον τρόπο οφείλει να γνωρίζει το «αληθινό» Bounding Box. Συνεπώς, μέσω της διαδικασίας του labeling εισέρχεται πληροφορία στο σύστημα για το πού ακριβώς βρίσκονται οι διαβάσεις στις εικόνες ώστε το σύστημα να μάθει. Για τη διαδικασία αυτή χρησιμοποιείτε το λογισμικό labelimg, στο οποίο ορίζονται Bounding Boxes γύρω από τις διαβάσεις και η ετικέτα (label), η οποία στην παρούσα μελέτη είναι η λέξη διάβαση στα αγγλικά (crosswalk). Συγχρόνως με το πέρας της αποθήκευσης των Bounding Boxes παράγονται txt αρχεία, τα οποία περιέχουν τις συντεταγμένες του «αληθινού» Bounding Box, καθώς και τον αριθμό μηδέν (0) που δηλώνει την ύπαρξη μίας τάξης. Σε περίπτωση δηλαδή που οι τάξεις ήταν δύο και το αντικείμενο οριστεί ότι ανήκει στη δεύτερη τάξη τότε ο αριθμός θα ήταν ένα (1). Πρακτικά, η αρίθμηση ξεκινά από το μηδέν. Το κάθε αρχείο περιέχει τις συντεταγμένες για κάθε διάβαση που υφίσταται στην εικόνα, είτε είναι μόνο μία, είτε παραπάνω. Παρακάτω φαίνεται η διαδικασία του labelimg και το αποτέλεσμα σε txt αρχείο :

Εικόνα 3.2 -4 Labeling με labelimg στην πόλη του Μπρίστολ

```
0 0.552885 0.417067 0.254808 0.262019
0 0.479567 0.641827 0.300481 0.206731
```

Εικόνα 3.2 -5 Αποτέλεσμα του labelimg (txt αρχείο)

Από την εικόνα Εικόνα 3.2 -4 γίνεται αντιληπτό το γεγονός πως οι συντεταγμένες (x , y) πάνω αριστερά και (x , y) κάτω δεξιά παρουσιάζουν εύρος [0 , 1]. Αυτό συμβαίνει διότι η εικόνα λαμβάνεται υπόψη με αρχή πάνω αριστερά (0 , 0) και τέλος κάτω δεξιά (1 , 1). Το παραπάνω γεγονός είναι σημαντικό, διότι κατά την εφαρμογή του αλγορίθμου το προβλεπόμενο Bounding Box παρουσιάζει συντεταγμένες του κεντρικού σημείου, στις οποίες έχει εφαρμοστεί σιγμοειδής συνάρτηση που συνεπάγεται αποτέλεσμα [0 , 1]. Λαμβάνοντας υπόψη όλα τα παραπάνω, είναι εφικτή η σύγκριση του προβλεπόμενου Bounding Box με το «αληθινό» κατά την ανίχνευση των διαβάσεων με χρήση του YOLOV3.

Επόμενο βήμα ο διαχωρισμός των εικόνων σε δύο txt αρχεία, ένα για την εκπαίδευση και ένα για τον έλεγχο (testing). Υφίστανται 200 εικόνες για κάθε πόλη. Το 80 % των εικόνων, δηλαδή 160 από τις 200 χρησιμοποιούνται για την εκπαίδευση, ενώ οι υπόλοιπες 40 (20 %) για τον έλεγχο. Ο διαχωρισμός υλοποιείται και για τις δύο πόλεις. Στα αρχεία είναι γραμμένο το path κάθε εικόνας που αντιστοιχεί στο αρχείο, σύμφωνα με την κατηγοριοποίηση που συνέβη προηγουμένως.

Στη συνέχεια, γίνεται μία προετοιμασία των δεδομένων ώστε μετέπειτα να χρησιμοποιηθούν στο Google colab για την εκπαίδευση. Ειδικότερα, μέχρι στιγμής για κάθε πόλη υπάρχουν : ένας φάκελος που περιέχει τις 200 εικόνες, ένας που περιέχει όλα τα παραγόμενα txt αρχεία από τη διαδικασία του labeling και τα δύο αρχεία με τα paths των εικόνων αντίστοιχα. Δημιουργούνται ακόμα ένας φάκελος με το όνομα της ετικέτας, δηλαδή crosswalk, καθώς και ένας ακόμα που κάνει μία σύνοψη με τα υπόλοιπα αρχεία δημιουργώντας και ένα backup αρχείο. Ο συγκεκριμένος φάκελος είναι αυτός που χρησιμοποιείται στη συνέχεια στην εκπαίδευση.

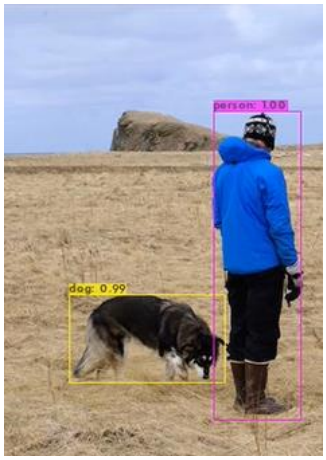
3.3 Εκπαίδευση (Training)

3.3.1 Εκπαίδευση YOLOV3

Στο προηγούμενο στάδιο συλλέχθηκαν τα δεδομένα και επεξεργάστηκαν. Τέλος προετοιμάστηκαν για ένταξη στο Google colab για την εκπαίδευση. Πιο συγκεκριμένα, τα παραπάνω αρχεία που «προετοιμάστηκαν» ανεβαίνουν στο Google Drive μέσα σε έναν φάκελο ονομαζόμενο darknet σε μορφή zip. Έπειτα, ξεκινάει η διαδικασία της εκπαίδευσης.

- Αρχικά, λαμβάνονται οι λεπτομέρειες του συστήματος και γίνεται μια ενημέρωση σε αυτές.
- Έπειτα, «βγαίνει» από τη μορφή zip ο φάκελος με τα δεδομένα που ανέβηκε στο Google Drive.
- Στον φάκελο αυτόν αφαιρείται η μορφοποίηση που έχουν τα windows και στη συνέχεια μετατρέπονται τα αρχεία σε unix, ώστε να μην παραληφθούν binary αρχεία.
- Δίνεται άδεια σε όλα τα αρχεία και διεξάγεται το compilation.
- Ελέγχεται η darknet δομή με ανίχνευση αντικειμένων σε μια εικόνα «άσχετη» κάνοντας χρήση το COCO σύνολο δεδομένων. Επίσης χρησιμοποιείται και το cfg αρχείο που προϋπάρχει στο darknet, διότι με το COCO δεν χρειάζεται εκπαίδευση (υποκεφάλαιο 2.4.2.1) και καμία άλλη μεταποίηση. Παρακάτω παρατίθεται η εικόνα και το αποτέλεσμα της ανίχνευσης :

Εικόνα 3.3 -4 Δοκιμαστική εικόνα πηγή : (zhangphil, 2019)



Εικόνα 3.3 -5 Αποτέλεσμα ανίχνευσης πηγή αρχικής εικόνας : (zhangphil, 2019)

Φαίνεται από τις εικόνες πολύ καλή ακρίβεια, μάλιστα για την ανίχνευση ανθρώπου 100 % και για του σκύλου 99 %. Είναι εμφανές λοιπόν πως η διαδικασία βαίνει καλώς. Συνεπώς, τώρα προχωρά η διαδικασία της εκπαίδευσης για τις εικόνες των δύο πόλεων.

- Σε συνέχεια με τα προηγούμενα στάδια, αφαιρείται ο backup φάκελος που υπάρχει στο darknet φάκελο και δημιουργείται ένας σύνδεσμος με έναν εξωτερικό backup φάκελο στο Google Drive, όπου τα παραγόμενα βάρη από την εκπαίδευση θα αποθηκεύονται εκεί.
- Εκτελείται σε αυτή τη φάση η εκπαίδευση με εντολή, σύμφωνα με τον οδηγό του Alexey Bochkovskiy στο GitHub, όπου χρησιμοποιούνται ο φάκελος με την σύμπτυξη των αρχείων (προετοιμασία δεδομένων), τα ήδη εκπαιδευμένα βάρη που λήφθηκαν από το GitHub καθώς και το cfg αρχείο με τις μεταβολές που έγιναν (αριθμός φίλτρων, τάξεων κ.λπ.), όπως αναφέρονται στα υποκεφάλαια 3.1 και 3.2. Σε αυτό το σημείο εντάσσεται και η εντολή για την εύρεση του mAP, σύμφωνα με το οποίο αξιολογείται το μοντέλο.
- Η εκπαίδευση παράγει βάρη, τα οποία χρησιμοποιούνται στην ανίχνευση αργότερα.

Παρακάτω παρατίθεται η εντολή της εκπαίδευσης από το Google colab:

```
!./darknet detector train cov_data/cov.data cov_yolov3.cfg darknet53.conv.74 -dont_show -map
```

Εικόνα 3.3 -6 Εκπαίδευση μοντέλου YOLOV3 και εύρεση ακρίβειας mAP

3.3.2 Εκπαίδευση των ZF – Net και Resnet

Για το ZF – Net : η εκπαίδευση έγινε για 30 εποχές, με activation function ReLu και στη συνέχεια δημιουργήθηκε το μοντέλο βάση της αρχιτεκτονικής του δικτύου με τα αντίστοιχα επίπεδα καθώς και με 8 feature maps για να είναι πιο ελαφρύ. Στη συνέχεια, πραγματοποιήθηκε η εκπαίδευση με την εντολή : model.fit(x_train, y_train,

```
batch_size=batch_size, #πλήθος feature maps
epochs=epochs,
verbose=1, #δείχνει την εποχή που σταμάτησε
validation_data=(x_test, y_test))
```

Για το Resnet (Mohan, 2020) : η εκπαίδευση έγινε για 30 εποχές, με activation function ReLu και στη συνέχεια δημιουργήθηκε το μοντέλο βάση της αρχιτεκτονικής του δικτύου με τα αντίστοιχα επίπεδα καθώς και με 8 feature maps. Ειδικότερα, αρχικά δημιουργήθηκαν τα δύο blocks, στη συνέχεια όλο το μοντέλο μέχρι το πέρας των blocks και τέλος μπήκαν τα υπόλοιπα επίπεδα. Στο δίκτυο αυτό χρησιμοποιείται το λεγόμενο tunic network. Σύμφωνα με αυτό τα αρχικά επίπεδα μαθαίνουν γενικά χαρακτηριστικά και όσο πιο βαθιά είναι τα δίκτυα μαθαίνουν όλο και πιο συγκεκριμένα χαρακτηριστικά. Για αυτό τα πρώτα αφήνονται όπως είναι και εκπαιδεύονται τα υπόλοιπα. Έτσι μειώνεται ο χρόνος και οι παράμετροι. Συνεπώς "παγώνει" ένα μέρος του μοντέλου. Επίσης χρησιμοποιούνται ήδη εκπαιδευμένα βάρη όπως και στην ανίχνευση παραπάνω, καθώς και τα λεγόμενα callbacks. Πιο συγκεκριμένα το ea και το cp. Το πρώτο χρησιμοποιείται για όταν το μοντέλο δεν βελτιώνεται κατά την εκτέλεσή του. Χρειάζεται ένα monitor = συγκεκριμενοποιεί το μέτρο της εκτέλεσης, συνήθως η ακρίβεια, ένα mode = αν η ακρίβεια μεγαλώνει, verbose = 1 δείχνει την εποχή που σταμάτησε, patience = περιθώριο παύσης κάποιων εποχών αφού δεν μεταβάλλεται η ακρίβεια. Εδώ το patience έχει τιμή 10. Το δεύτερο είναι το check point με το οποίο αποθηκεύεται το μοντέλο με την καλύτερη εκτέλεση. Γίνεται λοιπόν η εκπαίδευση με αντίστοιχη εντολή :

```
H=model.fit_generator(train_generator,validation_data=test_generator,epochs
=100,verbose=1,callbacks=[mc,es])
```

Έτσι μετά το πέρας της εκπαίδευσης επιλέγονται τα καλύτερα βάρη για την ανίχνευση.

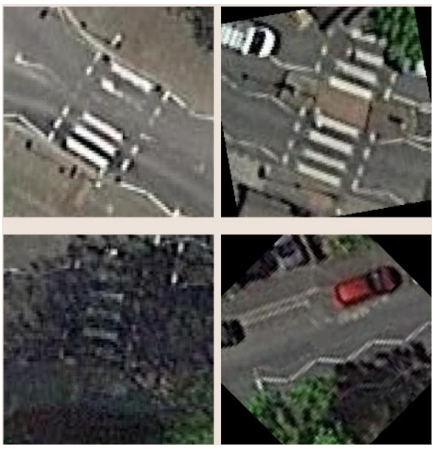
3.4 Επιλογή τεχνικής βαθιάς μάθησης

3.4.1 Ανίχνευση – Αξιολόγηση – ZF - Net - Resnet

Ύστερα της εκπαίδευσης το επόμενο στάδιο είναι η ταξινόμηση και έπειτα η αξιολόγηση των αποτελεσμάτων.

Για το ZF - Net : για 30 εποχές, ReLu, μία τάξη (διάβαση) και 8 feature maps η συνολική ακρίβεια του μοντέλου είναι 89 %, μια σχετικά καλή ακρίβεια αλλά με τιμή > 90 % θεωρείται ικανοποιητικό αποτέλεσμα συνεπώς έγιναν και άλλες δοκιμές αλλάζοντας τους παράγοντες (εποχές κλπ.) ή την activation function καθώς προστέθηκαν και regularizers για το overfitting με σκοπό την εύρεση του βέλτιστου μοντέλου. Έτσι, το βέλτιστο σενάριο είναι με τα δεδομένα : 15 εποχές, μία τάξη και συνάρτηση LeakyReLu. Σε αυτήν την περίπτωση άλλαξε η activation function διότι η Leaky ReLU σε σχέση με την ReLU δέχεται και αρνητικές τιμές. Επιπλέον προστέθηκαν και οι regularizers l1 και l2 για την αποφυγή overfitting. Ειδικότερα, ο l1: αφαιρεί μικρή ποσότητα βάρους από τα βάρη που δεν έχουν πληροφορία σε κάθε επανάληψη μηδενίζοντάς τα εν τέλει. Ο l2 : αφαιρεί μικρό ποσοστό βαρών σε κάθε επανάληψη. Το σενάριο αυτό έχει ακρίβεια 94 % καλύτερη από την προηγούμενη. Ως προς την ανίχνευση, το δίκτυο δεν μπόρεσε να προβλέψει εικόνες με διαβάσεις που παρουσιάζουν έλλειψη ορατότητας λόγω σκίασης ή παρουσίας φυλλωμάτων δέντρων, σβησίματα στο μοτίβο της διαγράμμισης, αλλά και διαβάσεις με ύπαρξη νησίδας. Μάλιστα τα παραπάνω σφάλματα παρατηρούνται και στις αντίστοιχες εικόνες σε στροφή. Επίσης κάποιες εικόνες με μόνο οχήματα ταξινομούνται ως διαβάσεις. Οι εικόνες με διαβάσεις και οι εικόνες με οχήματα μπορεί να ταξινομούνται με αυτόν τον τρόπο, λόγω της μη ύπαρξης ή ύπαρξης αντίστοιχα αρκετών εικόνων με τα αντίστοιχα αντικείμενα στην εκπαίδευση. Συνεπώς χρειάζονται περισσότερες διαφορετικές εικόνες καθώς και ένα πιο βαθύ δίκτυο για να μαθαίνει περισσότερα χαρακτηριστικά.

Επομένως το augmentation στην παρούσα φάση δεν βοήθησε και πολύ. Παρακάτω παρατίθενται παραδείγματα σφαλμάτων, δηλαδή εικόνες όπου οι διαβάσεις δεν ανιχνεύθηκαν και μια εικόνα όπου ανιχνεύθηκε χωρίς να είναι διάβαση :



Εικόνα 3.4.1 -1 Σφάλματα στην ανίχνευση ZF – Net – Μπρίστολ (γύρω από χώρους αναψυχής)

Για το Resnet : για 100 εποχές, τα καλύτερα βάρη από την εκπαίδευση, 8 feature maps, patience 10 και μία τάξη η ακρίβεια εκπαίδευσης είναι γύρω στο 98.2 %. Πολύ καλή ακρίβεια. Το Resnet -50 όμως λόγω της πολυπλοκότητάς του εξαιτίας των πολλών επιπέδων και παραμέτρων καθυστερεί. Συγκεκριμένα χρειάστηκαν περίπου 7 λεπτά για κάθε εποχή. Συνεπώς, σε επόμενο στάδιο απλοποιείται το δίκτυο ώστε να λυθεί το παραπάνω ζήτημα, δηλαδή απλοποιείται για μεγαλύτερη ταχύτητα και μείωση πολυπλοκότητας.

Ειδικότερα, σε σχέση με την πλήρη αρχιτεκτονική ενός Resnet -50 που δημιουργήθηκε αρχικά, τα identity και convolutional blocks έμειναν αυτούσια, ενώ στη δημιουργία του ολικού Resnet μετέπειτα κρατήθηκε μόνο ένα identity και το προτελευταίο «ζευγάρι» identity και convolutional blocks απορρίφθηκε εντελώς. Συνεπώς για να είναι στο τέλος ίδιες οι διαστάσεις με το “ολόκληρο” Resnet -50 στο πρώτο στάδιο ορίζεται stride = s = 2 αντί για 1 έτσι ώστε να “καλυφθεί το κενό” του σταδίου που απορρίφθηκε. Έτσι μειώθηκαν κατά πολύ οι παράμετροι όπως φαίνεται παρακάτω από το summary:

```
Total params: 1,166,337
```
→ Απλοποιημένο

```
Total params: 32,009,601
```
→Αρχικό

Εικόνα 3.4.1 -2 Παράμετροι

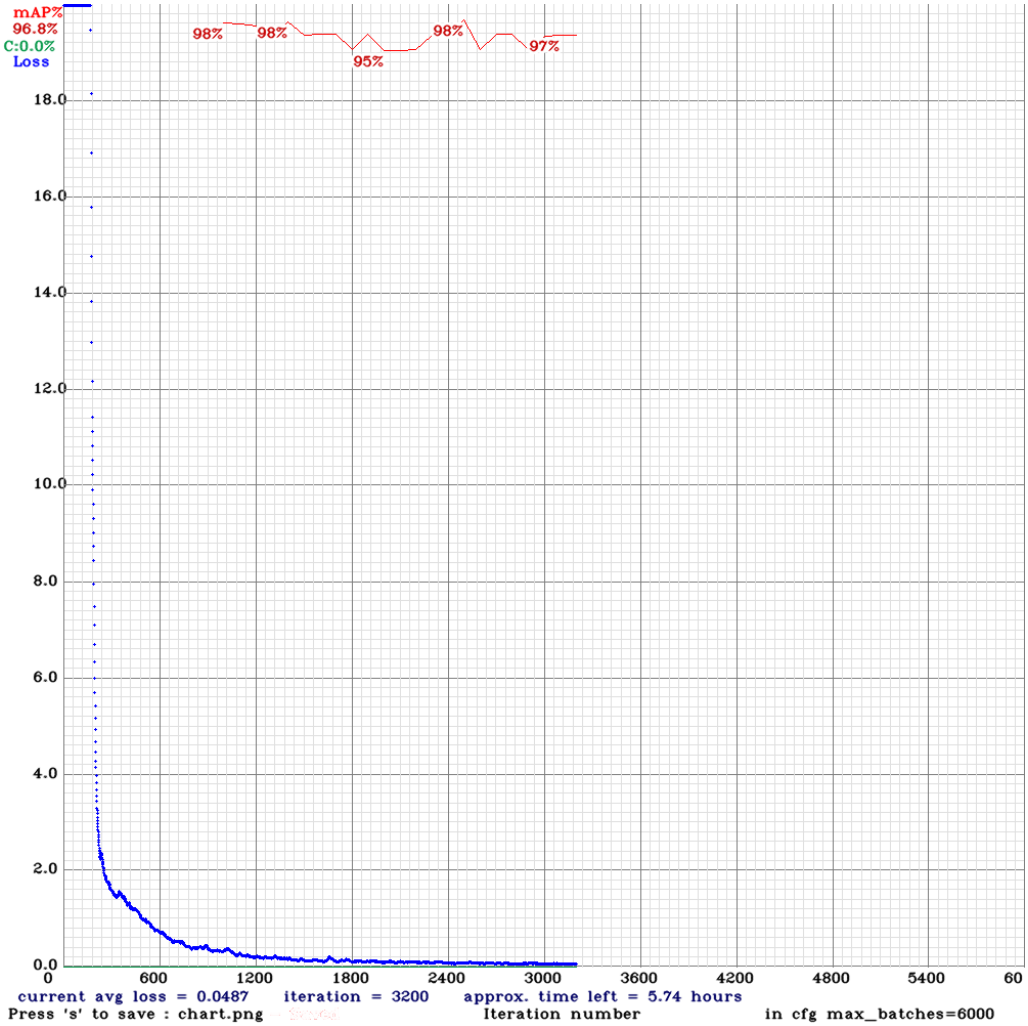
Η ακρίβεια σε αυτήν την περίπτωση είναι περίπου 98.3 %, σχεδόν ίδια αλλά λίγο καλύτερη από το περίπλοκο δίκτυο. Τα σφάλματα στην ανίχνευση που παρουσιάζει το δίκτυο αυτό αποτελούν τα ίδια με το προηγούμενο δίκτυο ως προς τις διαβάσεις, με τη διαφορά πως δεν ταξινομεί το δίκτυο ως διάβαση καμία εικόνα που δεν περιέχει διαβάσεις. Προφανώς είναι καλύτερο το απλούστερο δίκτυο και από το ZF – Net αλλά και από το Resnet -50. Τα παραπάνω φαίνονται στον παρακάτω πίνακα :

Πίνακας 3.4.1 -1 Ακρίβεια εκπαίδευσης των δύο δικτύων για το Μπρίστολ

μοντέλο	ZF - Net	Resnet -50	**Resnet απλ/νο**
Ακρίβεια (%)	94	98.2	**98.3**

3.4.2 Αξιολόγηση εκπαίδευσης – YOLOV3

Καταρχάς σύμφωνα με το προηγούμενο υποκεφάλαιο (3.1) το max_batches = 6000, με συνέπεια την επίτευξη 6000 επαναλήψεων κατά την εκπαίδευση. Κατά τη διάρκεια αυτή σε διάγραμμα αποτυπώνεται και η ακρίβεια mAP καθώς και το μέσο σφάλμα (average loss) της εκπαίδευσης για κάθε επανάληψη. Το σφάλμα αυτό πρέπει να είναι όσο το δυνατόν μικρότερο. Στον φάκελο των βαρών αποθηκεύονται τα βάρη των 1000, 2000, 3000 κ.λπ. επαναλήψεων, καθώς και τα καλύτερα βάρη μαζί με εκείνα που αντιστοιχούν στη συγκεκριμένη επανάληψη που διεξάγεται όταν σταματήσει η εκπαίδευση. Η εκπαίδευση χρήζει πολλών ωρών, όμως γύρω στις 12 ώρες περίπου διαρκεί η άδεια που παραχωρεί το Google colab για τη GPU. Για την εξοικονόμηση χρόνου ακολουθήθηκε ένας εμπειρικός κανόνας[2] σύμφωνα με τον οποίο αν το average loss είναι μικρότερο (<) του 0.060730 = 0.06 τότε η εκπαίδευση μπορεί να σταματήσει. Παρακάτω παρατίθενται τα τελικά διαγράμματα που παρουσιάζουν την ακρίβεια και το μέσο σφάλμα της εκπαίδευσης για τις πόλεις του Μάντσεστερ και του Μπρίστολ αντίστοιχα :

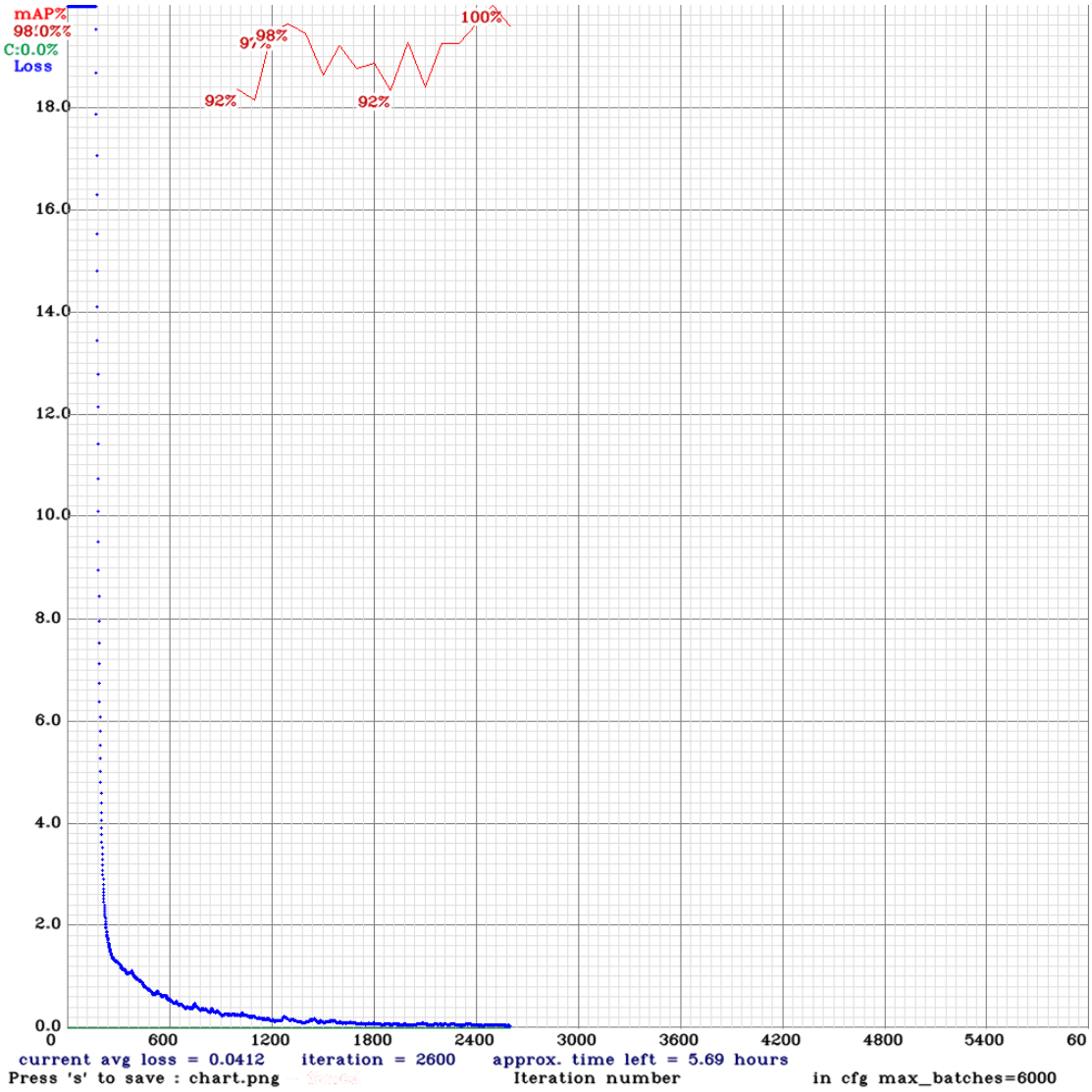


Εικόνα 3.4.2 -4 Ακρίβεια mAP και μέσο σφάλμα (average loss) της εκπαίδευσης για την πόλη του Μάντσεστερ

Παρατηρείται πως η πρώτη τιμή ακρίβειας βγαίνει στην χιλιοστή επανάληψη και έχει τιμή 98 %. Η εκπαίδευση σταμάτησε με σφάλμα 0.0487 < 0.06. Η ακρίβεια στο σημείο

[2] https://github.com/AlexeyAB/darknet

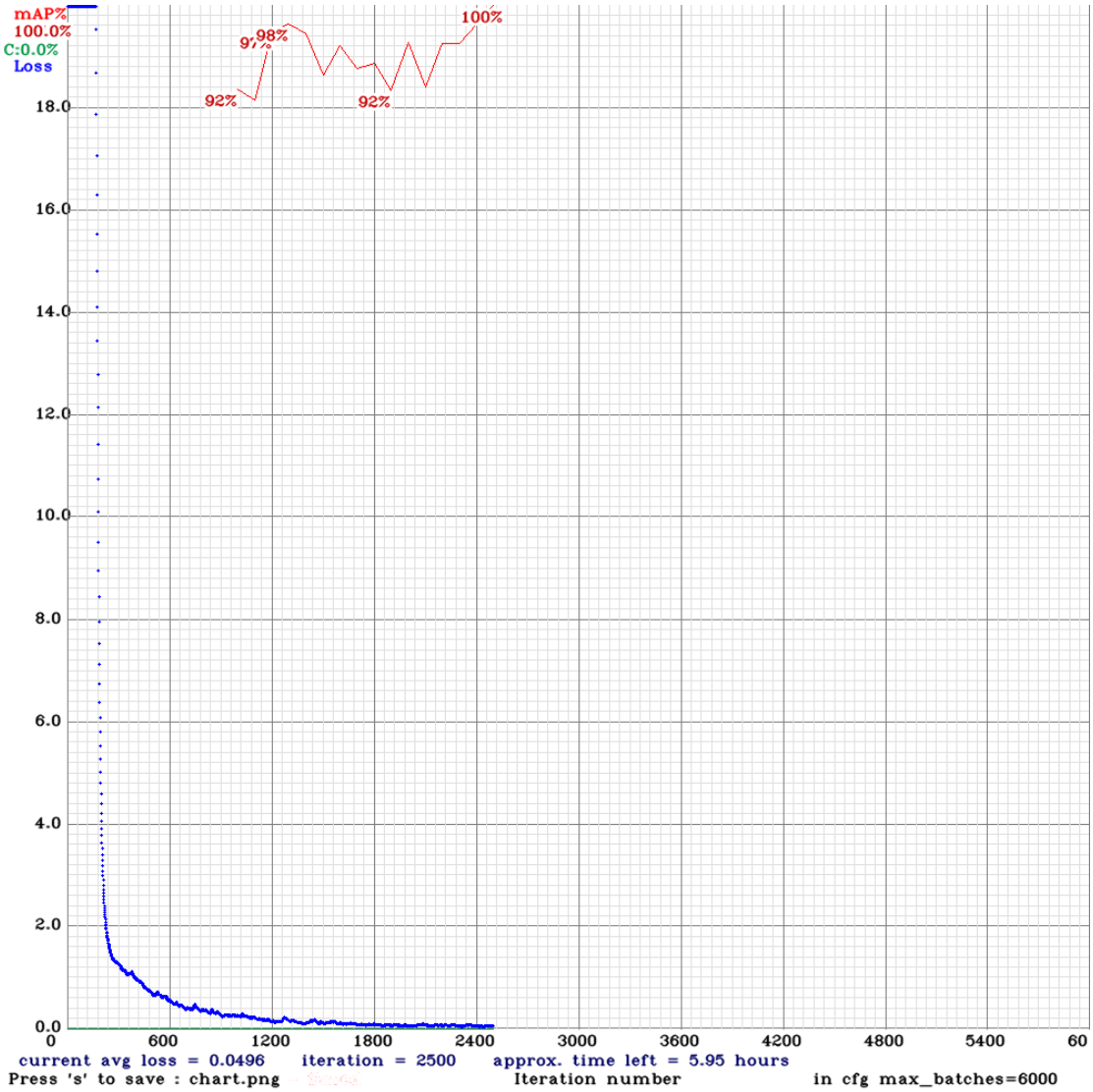
αυτό είναι 96.8 % περίπου 97 % δηλαδή και φαίνεται πως έπειτα από τη μεγαλύτερη ακρίβεια, που είναι 98,3 % περίπου, οι τιμές του mAP βρίσκονται γύρω στο 97 %, οπότε είναι περίπου σταθερή η ακρίβεια. Τα βάρη που στη συνέχεια θα χρησιμοποιηθούν στην ανίχνευση είναι τα καλύτερα από τα υπάρχοντα, δηλαδή τα βάρη που αντιστοιχούν στην καλύτερη ακρίβεια (98,3 %). Η ελάχιστη ακρίβεια στην εκπαίδευση αποτελεί η τιμή 95 %. Παρόλα αυτά όμως είναι προφανές πως η ακρίβεια είναι πολύ καλή (πάνω από 90 %), ακόμα και η «χειρότερη». Αυτά είναι τα αποτελέσματα της εκπαίδευσης της πόλης του Μάντσεστερ. Στη συνέχεια ακολουθούν τα αποτελέσματα για την πόλη του Μπρίστολ :



Εικόνα 3.4.2 -5 Ακρίβεια mAP και μέσο σφάλμα (average loss) της εκπαίδευσης για την πόλη του Μπρίστολ

Παρατηρείται πως η πρώτη τιμή ακρίβειας έχει τιμή 92 %. Η εκπαίδευση σταμάτησε με σφάλμα 0.0412 < 0.06. Η ακρίβεια στο σημείο αυτό είναι 98% και φαίνεται πως έπειτα από τη μεγαλύτερη ακρίβεια, που είναι 100 %, η τιμή αυτή του mAP αποτελεί την αμέσως καλύτερη ακρίβεια. Μάλιστα έφτασε σε αυτήν την ακρίβεια και προηγουμένως η εκπαίδευση συνεχίζοντας αργότερα στην 100 %. Σε εκείνο το σημείο, όπως φαίνεται παρακάτω στην εικόνα που ακολουθεί, θα σταματούσε η εκπαίδευση διότι δεν θα μπορούσε να υπάρξει καλύτερη ακρίβεια από 100 %. Επίσης, το σφάλμα ήταν 0.0496

< 0.06. Όμως η εκπαίδευση δεν διακόπηκε ως προς μελέτη της ακρίβειας στη συνέχεια έπειτα από την «μέγιστη». Παρατηρήθηκε λοιπόν, πως η ακρίβεια «έπεσε» ξανά στο 98 %, που όπως προαναφέρθηκε ήταν η μέγιστη ακρίβεια μέχρι την επίτευξη του 100 %. Για αυτό το λόγο και η εκπαίδευση διακόπηκε στις 2600 επαναλήψεις όπως φαίνεται στην εικόνα παραπάνω. Τα βάρη που στη συνέχεια θα χρησιμοποιηθούν στην ανίχνευση είναι τα καλύτερα από τα υπάρχοντα, δηλαδή τα βάρη που αντιστοιχούν στην καλύτερη ακρίβεια (100 %). Στην συγκεκριμένη περίπτωση όμως θα ληφθούν και αυτά της τελευταίας επανάληψης πριν το πέρας της εκπαίδευσης, για σύγκριση της «τέλειας» ακρίβειας με την πολύ καλή. Να τονιστεί πως επιλέγεται η δεύτερη ακρίβεια με τιμή 98 % και όχι η πρώτη, επειδή το σφάλμα είναι μικρότερο επανάληψη με την επανάληψη, όπως μπορεί να αποφανθεί στην μπλε καμπύλη στην παραπάνω εικόνα. Η ελάχιστη ακρίβεια στην εκπαίδευση αποτελεί η τιμή 92 %. Παρόλα αυτά όμως είναι προφανές πως η ακρίβεια είναι πολύ καλή (πάνω από 90 %), ακόμα και η «χειρότερη». Παρακάτω παρουσιάζονται τα αποτελέσματα της εκπαίδευσης για την πόλη του Μπρίστολ πετυχαίνοντας την μέγιστη ακρίβεια :



Εικόνα 3.4.2 -6 Ακρίβεια mAP και μέσο σφάλμα (average loss) της εκπαίδευσης για την πόλη του Μπρίστολ (επίτευξη μέγιστης ακρίβειας 100 %)

Όπως φαίνεται για το Μάντσεστερ η ακρίβεια είναι 98.3 % και για το Μπρίστολ 100 %. Θα γίνει όμως παρακάτω και μια σύγκριση με τη δεύτερη καλύτερη ακρίβεια 98 %. Από το προηγούμενο υποκεφάλαιο έχει επιλεχθεί καλύτερο το απλοποιημένο δίκτυο Resnet με ακρίβεια 98.3 %. Παρατηρείται πως οι ακρίβειες για το Μπρίστολ είναι πρακτικά ίδιες στην χειρότερη περίπτωση που δεν ληφθεί υπόψη η 100 %. Διαφορετικά το 100 % είναι πολύ καλύτερη ακρίβεια. Επίσης το YOLOV3 εκπαιδεύεται να ανιχνεύσει το αντικείμενο μέσα στο πλαίσιο που δημιουργήθηκε στο labeling (αληθινό πλαίσιο). Επομένως, δεν υπάρχει το πρόβλημα να ανιχνευθεί κάτι ως διάβαση χωρίς να είναι. Επίσης βάσει του θεωρητικού υποβάθρου ανιχνεύει σε τρείς κλίμακες που σημαίνει πως μπορούν να ανιχνευθούν αντικείμενα σε διαφορετικό μέγεθος. Επιπλέον το μοντέλο αυτό έχει τόσο καλή ακρίβεια με εκπαίδευση μόνο 200 εικόνων σε αντίθεση με τις 1000 των άλλων μοντέλων. Βέβαια είναι πιο αργό. Παρόλα αυτά όμως όλα οδηγούν στην επιλογή του YOLOV3. Η τελική όμως απόφαση θα παρθεί λαμβάνοντας υπόψη και τα αποτελέσματα της ανίχνευσης παρακάτω. Τα παραπάνω παρουσιάζονται και στους παρακάτω πίνακες :

Πίνακας 3.4.2 -1 Ακρίβεια εκπαίδευσης Μπρίστολ

μοντέλο	ZF - Net	Resnet -50	Resnet απλ/νο	**YOLOV3**	
Ακρίβεια (%)	94	98.2	98.3	**100**	98

Πίνακας 3.4.2 -2 Ακρίβεια εκπαίδευσης Μάντσεστερ

μοντέλο	YOLOV3
Ακρίβεια (%)	98.3

3.5 Ανίχνευση – Αξιολόγηση - YOLOV3

Η διαδικασία της ανίχνευσης στο πρακτικό κομμάτι (κώδικας) ακολουθεί το θεωρητικό υπόβαθρο. Δηλαδή :

- Εισάγονται ως input οι εικόνες 416 x 416, λαμβάνονται το πλάτος και το ύψος και μετατρέπονται οι εικόνες σε σύνολο δεδομένων με τη βοήθεια της βιβλιοθήκης blob. Με αυτόν τον τρόπο θα μπορέσουν να εισαχθούν στο μοντέλο και να προκύψουν προβλέψεις.
- Οι ετικέτες ορίζονται ως μία (crosswalk), αφού μόνο μία τάξη υφίσταται και ορίζεται στη συνέχεια το χρώμα των πλαισίων.
- Φορτώνεται έπειτα το ήδη εκπαιδευμένο μοντέλο. Σε αυτό το σημείο χρησιμοποιούνται τα βάρη από την εκπαίδευση και το cfg αρχείο που μεταποιήθηκε πριν την εκπαίδευση και χρησιμοποιήθηκε αργότερα από αυτή. Η αντίστοιχη εντολή φαίνεται παρακάτω :
    - cv2.dnn.readNetFromDarknet()

Έπειτα, με μία λούπα ο κώδικας «πηγαίνει» σε όλα τα επίπεδα layers από το δίκτυο του YOLOV3 και «κρατάει» το τελευταίο επίπεδο το output.

- Στη συνέχεια, το αποτέλεσμα της χρήσης του blob (οι εικόνες ως σύνολο δεδομένων) εισάγεται στο μοντέλο και παράγονται τα επίπεδα προβλέψεων μέσω της forward μεθόδου (κεφάλαιο 2 Θεωρητικό υπόβαθρο).

- Αφού έχουν παραχθεί τα επίπεδα προβλέψεων τώρα ορίζονται λίστες για τα στοιχεία των προβλέψεων, για παράδειγμα λίστα για τα προβλεπόμενα Bounding Boxes με τα στοιχεία τους σύμφωνα με την θεωρία.
- Ύστερα, με λούπα σε κάθε επίπεδο προβλέψεων λαμβάνονται οι προβλέψεις με τα στοιχεία τους.
- Όμως όπως αναλύθηκε στο θεωρητικό υπόβαθρο, σκοπός είναι ένα μοναδικό τελικό προβλεπόμενο Bounding Box, οπότε αρχικά με τη βοήθεια κατωφλιού 20 % απορρίπτονται όλα τα πλαίσια με πιθανότητα ύπαρξης διάβασης κάτω (<) 20 %. Τα Bounding Boxes που «παραμένουν», τα συνοδεύουν οι αντίστοιχες ετικέτες καθώς και οι συντεταγμένες του αρχικού σημείου πάνω αριστερά των προβλεπόμενων Bounding Boxes, οι οποίες υπολογίζονται από αυτές του κεντρικού σημείου των Bounding Boxes και τα πλάτη και ύψη μετασχηματισμένα στην κλίμακα του αντικειμένου (upscaling).
- Σε αυτό το στάδιο αποθηκεύονται οι πάνω συντεταγμένες, με τα υπόλοιπα στοιχεία των προβλεπόμενων Bounding Boxes σε λίστες (χωρίς το κεντρικό σημείο), ώστε στη συνέχεια να πραγματωθεί η NMS για την περαιτέρω μείωση των Bounding Boxes.
- Στο βήμα αυτό εφαρμόζεται η NMS με κατώφλι 0.4 και κατώφλι της IoU 0.5. Πρακτικά η εφαρμογή αυτή επιφέρει μόνο τις επιλεγμένες τιμές (id) που αντιστοιχούν στα αντίστοιχα στοιχεία ενός προβλεπόμενου Bounding Box.
- Από τις παραπάνω τιμές λαμβάνονται τα στοιχεία του Bounding Box και λαμβάνονται οι αντίστοιχες ετικέτες και πιθανότητες.
- Έπειτα υπολογίζονται και οι συντεταγμένες κάτω δεξιά του Bounding Box, και στη συνέχεια προγραμματίζεται να εμφανίζονται οι πιθανότητες σε μορφή ποσοστού στην οθόνη.
- Αργότερα, σχεδιάζεται το Bounding Box και προγραμματίζεται να εμφανίζονται οι πιθανότητες σε μορφή ποσοστού εκτός από την οθόνη και στις εικόνες.

3.5.1 Ανίχνευση σε δοκιμαστικές εικόνες – YOLOV3

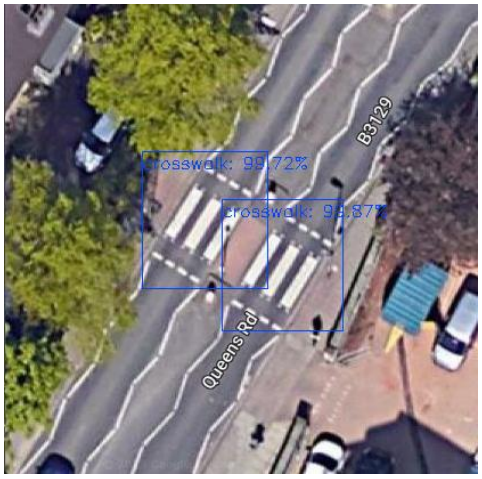
Η παραπάνω διαδικασία της ανίχνευσης χρησιμοποιείται για την επιλογή των βαρών για την πόλη του Μπρίστολ, όπως αναλύεται παρακάτω.

Σε αυτό το στάδιο γίνεται ανίχνευση σε δοκιμαστικές εικόνες 416 x 416 για την επιλογή των καλύτερων βαρών για την περιοχή του Μπρίστολ. Ειδικότερα, η ανίχνευση γίνεται με χρήση των βαρών που αντιστοιχούν στις δύο καλύτερες ακρίβειες 100 % και 98 %. Στη συνέχεια πραγματοποιείται σύγκριση στα δύο αποτελέσματα και εντέλει επιλέγονται τα καλύτερα βάρη. Θεωρητικά αναμενόμενο είναι τα καλύτερα βάρη να είναι αυτά που αντιστοιχούν στην ακρίβεια 100 %, όμως με αυτήν τη σύγκριση μπορεί να αποφανθεί αν με λίγο χειρότερη ακρίβεια αλλά με καλύτερο σφάλμα επιτυγχάνονται καλύτερα αποτελέσματα ή αν όντως η τιμή 100 % δηλώνει αυτομάτως πως δεν χρήζει αμφιβολιών. Παρακάτω παρατίθεται μία από τις δοκιμαστικές εικόνες και τα αποτελέσματά της :

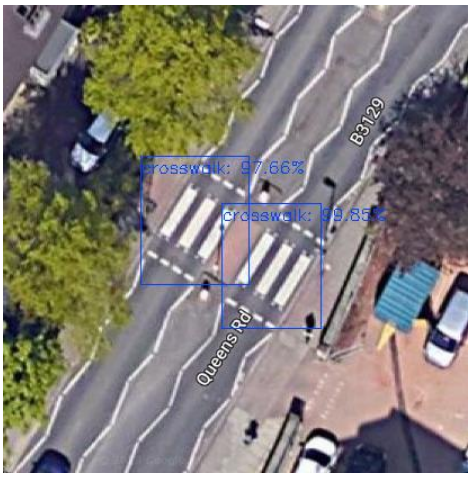
Εικόνα 3.5.1 -7 Δοκιμαστική ανίχνευση (1) - Μπρίστολ με ακρίβεια 100 %

Εικόνα 3.5.1 -8 Δοκιμαστική ανίχνευση (1) - Μπρίστολ με ακρίβεια 98 %

Οι παραπάνω εικόνες αποτελούν εικόνες, οι οποίες χρησιμοποιήθηκαν στην εκπαίδευση. Φαίνεται πως με ακρίβεια 100 % ανιχνεύθηκε η διάβαση με πιθανότητα περίπου 88 %, μια καλή πιθανότητα δηλαδή, ενώ με ακρίβεια 98 % δεν μπόρεσε να ανιχνευθεί η διάβαση. Επομένως, σε αυτήν την περίπτωση η 100 % ακρίβεια είναι καλύτερη. Σε όλες τις άλλες δοκιμές τα αποτελέσματα ήταν σχεδόν ίδια και μάλλιστα με πιθανότητες άνω του 90 %. Συνεπώς η ακρίβεια 100 % μπορεί να ανιχνεύσει μια διάβαση υπό σκιά και με όχι την καλύτερη ορατότητα. Οι συγκεκριμένες εικόνες όμως όπως αναφέρθηκε έλαβαν μέρος στην εκπαίδευση, οπότε το επόμενο βήμα είναι δοκιμές σε εικόνες της πόλης που δεν συμμετείχαν στην εκπαίδευση ως προς μελέτη της ακρίβειας αλλά και της πιθανότητας. Δηλαδή, μελέτη του αν η πιθανότητα είναι όντως άνω του 90 % και δεν υφίσταται μόνο σε εικόνες που χρησιμοποιήθηκαν στην εκπαίδευση. Παρακάτω παρατίθεται μια δοκιμαστική ανίχνευση σε εικόνα που δεν χρησιμοποιήθηκε στην εκπαίδευση με χρήση βαρών που αντιστοιχούν σε ακρίβεια 100 % και 98 % αντίστοιχα :



Εικόνα 3.5.1 -9 Δοκιμαστική ανίχνευση (2) - Μπρίστολ με ακρίβεια 100 %



Εικόνα 3.5.1 -10 Δοκιμαστική ανίχνευση (2) - Μπρίστολ με ακρίβεια 98 %

Φαίνεται όπως στην πρώτη εικόνα η πιθανότητα είναι άνω του 90 % και μάλιστα πολύ καλή με ποσοστά 99,7 % περίπου και 99,87 %. Αντίθετα, στην ανίχνευση με ακρίβεια

98 % τα αποτελέσματα ναι μεν είναι πολύ κοντά με αυτά της πρώτης εικόνας, δηλαδή 97,7 % περίπου και 99,85 %, αλλά ειναι ελάχιστα μικρότερα. Η πιθανότητα γενικά είναι πάρα πολύ καλή και φαίνεται πως δεν σχετίζεται μόνο με τις εικόνες που χρησιμοποιήθηκαν στην εκπαίδευση. Επίσης, προκύπτει το συμπέρασμα πως οι διαφορές στην ανίχνευση με χρήση βαρών ακριβείας 98 % και 100 % είναι πολύ λίγες, με την 100 % να είναι καλύτερη, καθώς όπως προκύπτει από την πρώτη δοκιμαστική ανίχνευση που παρατίθεται παραπάνω έχει την ικανότητα να ανιχνεύει διαβάσεις που βρίσκονται σε πιο «δυσμενείς» συνθήκες. Επειδή όμως, η λήψη των εικόνων έγινε χειροκίνητα, λαμβάνοντας υπόψη το ανθρώπινο λάθος (την περίπτωση μία ή παραπάνω από τις εικόνες που θεωρούνται πως δεν συμμετείχαν στην εκπαίδευση να έχουν συμμετάσχει) γίνεται έλεγχος σε μια διαφοερετική πόλη. Πιο συγκεκριμένα, έγιναν μερικές δοκιμές στην διπλανή πόλη του Μπρίστολ, την πόλη του Μπαθ και τα αποτελέσματα για μία από αυτές πρατίθενται παρακάτω :

Εικόνα 3.5.1 -11 Δοκιμαστική ανίχνευση (3) - Μπαθ με ακρίβεια 100 %

Εικόνα 3.5.1 -12 Δοκιμαστική ανίχνευση (3) - Μπαθ με ακρίβεια 98 %

Όπως φαίνεται η ανίχνευση με χρήση βαρών ακριβείας 100 % είναι καλύτερη ξανά με ελάχιστη διαφορά, ενώ η πιθανότητα είναι πολύ καλή, πάνω από 90 %. Πρακτικά, οι διαφορές των βαρών είναι ελάχιστες, βέβαια και οι ακρίβειες 98 % και 100 % δεν διαφέρουν πολύ. Φαίνεται λοιπόν πως τα βάρη ακριβείας 100 % επιλέγονται για την ανίχνευση μετέπειτα στην πόλη του Μπρίστολ. Επίσης, λαμβάνοντας υπόψη όλα τα παραπάνω, αποφαίνεται πως η εκπαίδευση μπορούσε να σταματήσει και όταν επιτεύχθηκε η ακρίβεια 100 % σε συνδυασμό με το σφάλμα, το οποίο ήταν λιγότερο από 0.06. Επιπλέον αξιοσημείωτο «εύρημα» αποτελεί το γεγονός πως οι πιθανότητες στην ανίχνευση δεν επηρεάζονται από τη συμμετοχή ή όχι των εικόνων στην εκπαίδευση.

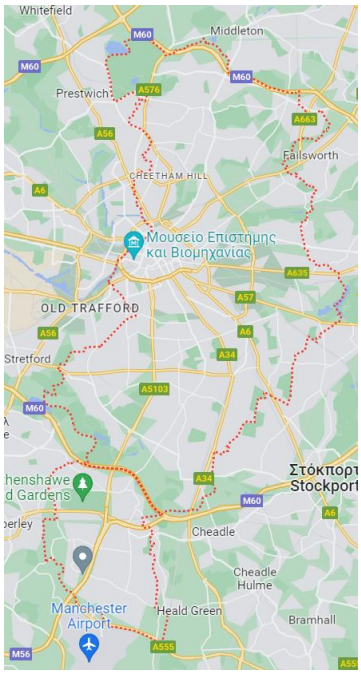
Τελικά επιλέχθηκε ο αλγόριθμος YOLOV3, διότι σύμφωνα με τα παραπάνω αποτελέσματα η ακρίβεια 100 % είναι η καλύτερη, αλλά επίσης φαίνεται να μην παρουσιάζει τα σφάλματα των άλλων τεχνικών βαθιάς μάθησης, δηλαδή μπορεί να ανιχνεύει διαβάσεις υπό σκιά αλλά και με έλλειψη ορατότητας εξαιτίας των δέντρων. Εν κατακλείδι η προτεινόμενη μεθοδολογία χρησιμοποιεί το YOLOV3 και για τις δύο πόλεις. Παρακάτω παρουσιάζεται και συγκεντρωτικός πίνακας με την ακρίβεια εκπαίδευσης του αλγορίθμου :

Πίνακας 3.5.1 -1 Ακρίβεια εκπαίδευσης YOLOV3 για τις δύο πόλεις

Πόλεις	Μπρίστολ	Μάντσεστερ
Ακρίβεια (%)	100	98.3

3.5.2 Ανίχνευση διαβάσεων σε τμήμα χάρτη

Στο σημείο αυτό γίνεται ανίχνευση των διαβάσεων για κάθε πόλη σε ένα τμήμα χάρτη για κάθε πόλη. Ειδικότερα ένα τμήμα που αποτελεί το 0.05 % (53 στρέμματα) της πόλης του Μπρίστολ (110 τ. χλμ.)[3] και ένα που αποτελεί το 0.04 % (43 στρέμματα) της πόλης του Μάντσεστερ (115.6 τ. χλμ.)[4], ώστε ύστερα να υπολογιστούν οι συντεταγμένες των διαβάσεων. Ειδικότερα, σύμφωνα με τη Google Maps Τα όρια των πόλεων παρατίθενται παρακάτω σύμφωνα με τη Google Maps :



Εικόνα 3.5.2 -1 Πόλεις Μάντσεστερ και Μπρίστολ αντίστοιχα (πηγή : Google Maps[5])

3.5.2.1 Ανίχνευση διαβάσεων σε τμήμα χάρτη για την πόλη του Μπρίστολ

Πιο συγκεκριμένα, λήφθηκε μία εικόνα 3014 x 1760 από το υπόβαθρο της Google earth στο λογισμικό QGIS, που απεικονίζει ένα μέρος του κέντρου της πόλης. Η εικόνα αυτή αντιστοιχεί στο 0.05 % της πόλης και συνεπώς για την κάλυψη όλης της πόλης χρειάζονται 100 / 0.05 = 2000 εικόνες. Για την ανίχνευση χρησιμοποιείται η διαδικασία που αναλύεται παραπάνω με κάποιες διαφοροποιήσεις. Ειδικότερα, αφού ο αλγόριθμος χρήζει εικόνων 416 x 416 η «μεγάλη» αυτή εικόνα κόβεται σε κομμάτια (patches) μεγέθους 416 x 416 και στη συνέχεια αυτά υποβάλλονται σε ανίχνευση σύμφωνα με την παραπάνω διαδικασία. Έπειτα της ανίχνευσης τα patches ξανά ενώνονται σε μία νέα εικόνα μεγέθους ίσου με την αρχική αλλά συμπεριλαμβανομένων και των αποτελεσμάτων της ανίχνευσης. Να σημειωθεί πως αποθηκεύονται σε ξεχωριστό φάκελο όλα τα patches για έλεγχο λαθών, καθώς και μόνο αυτά που ανιχνεύθηκαν διαβάσεις. Με αυτόν τον τρόπο μπορεί να εισαχθεί ένα τμήμα χάρτη με αποτέλεσμα την ανίχνευση των διαβάσεων με διαγράμμιση ζέβρα στον χάρτη αυτόν.

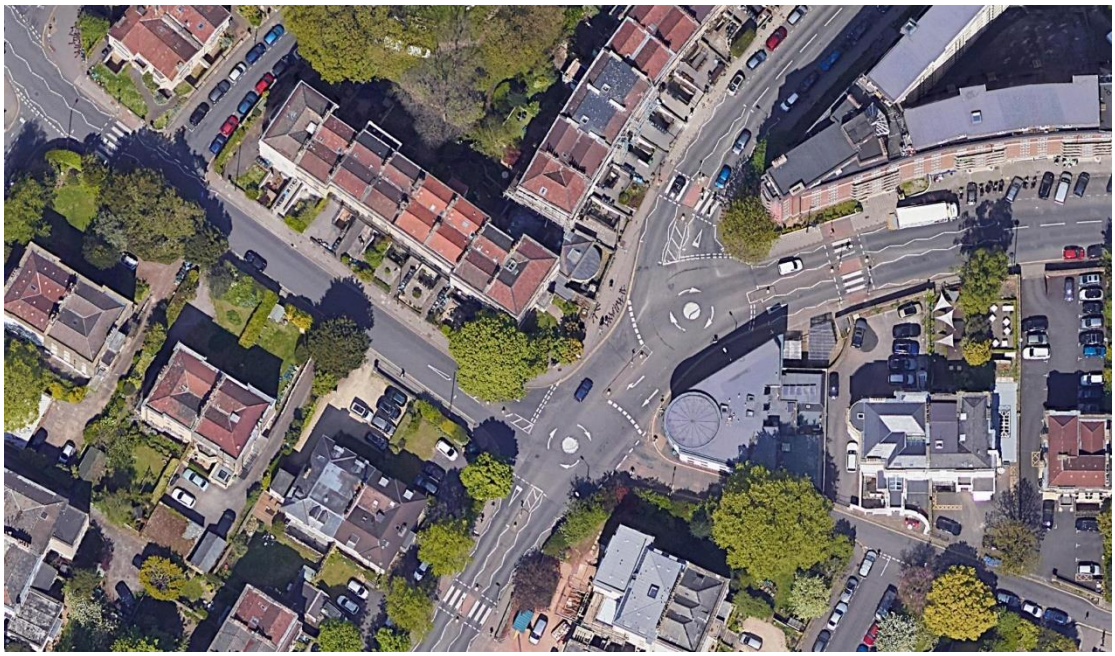
---

[3] Wikipaidia.org https://el.wikipedia.org/wiki/%CE%9C%CF%80%CF%81%CE%AF%CF%83%CF%84%CE%BF%CE%BB

[4] Wikipaidia.org https://el.wikipedia.org/wiki/%CE%9C%CE%AC%CE%BD%CF%84%CF%83%CE%B5%CF%83%CF%84%CE%B5%CF%81

[5] Google maps https://www.google.gr/maps/@53.5160245,-2.3601363,6.65z?hl=el

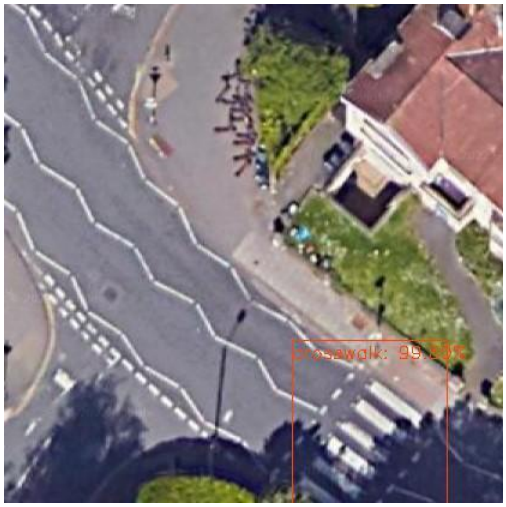
Παρακάτω παρατίθεται η αρχική εικόνα και το αποτέλεσμα της ανίχνευσης για την πόλη του Μπρίστολ :



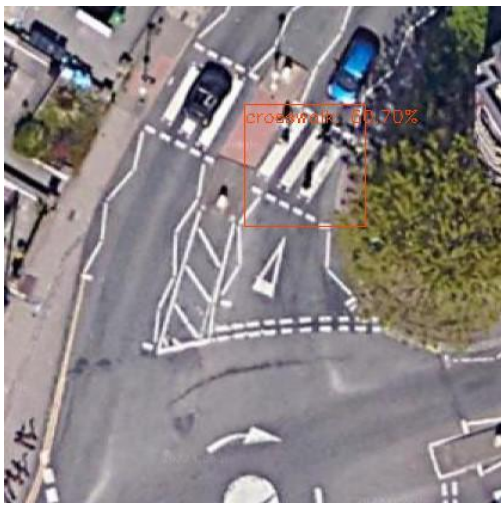
Εικόνα 3.5.2.1 -3 Δορυφορική εικόνα στο κέντρο του Μπρίστολ πριν την ανίχνευση

Εικόνα 3.5.2.1 -4 Δορυφορική εικόνα του κέντρου του Μπρίστολ μετά την ανίχνευση

Παρακάτω παρατίθενται τα ανιχνεύσιμα patches της παραπάνω εικόνας για καλύτερη οπτικοποίηση :



Εικόνα 3.5.2.1 -3 Patch 99.80 % - Μπρίστολ



Εικόνα 3.5.2.1 -4 Patch 60.70 % - Μπρίστολ

Εικόνα 3.5.2.1 -5 Patch 98.13 % - Μπρίστολ

Εικόνα 3.5.2.1 -6 Patch 88.35 % - Μπρίστολ

Εικόνα 3.5.2.1 -7 Patch 98.37 % - Μπρίστολ

```
runfile('C:/Users/valen/Desktop/bri
(4, 7, 1, 416, 416, 3)
predicted object crosswalk: 99.80%
predicted object crosswalk: 60.70%
predicted object crosswalk: 98.13%
predicted object crosswalk: 88.35%
predicted object crosswalk: 98.37%
```

Εικόνα 3.5.2.1 -8 Πιθανότητες από την οθόνη

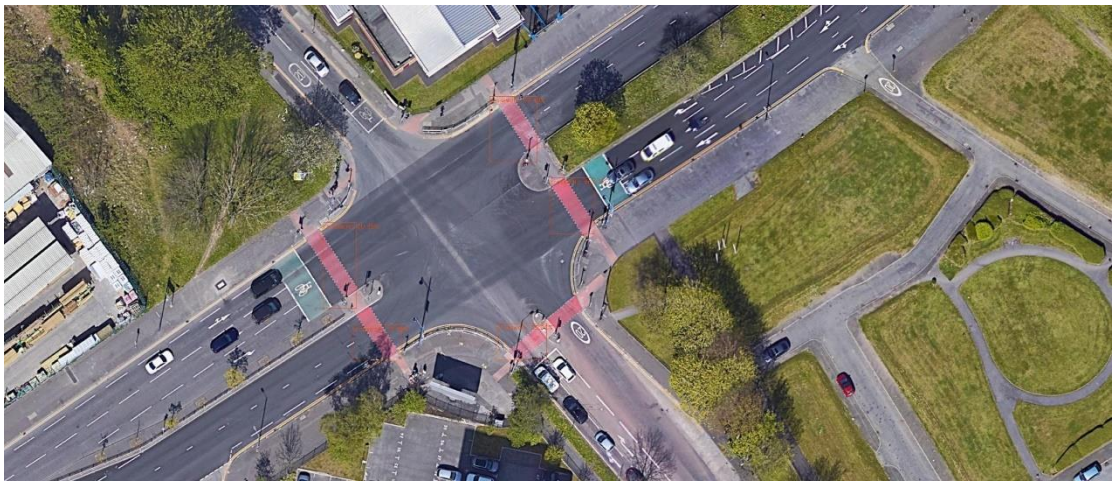
Στην οθόνη παρουσιάζονται τα αποτελέσματα της ανίχνευσης τα οποία αντιστοιχούν στις διαβάσεις στην τελική εικόνα από αριστερά προς δεξιά και από πάνω προς τα κάτω. Στο δεύτερο patch με ακρίβεια 60.70 % φαίνεται πως ανιχνεύθηκε μόνο η μία διάβαση. Λογικά δεν ανιχνεύθηκε η δεύτερη, λόγω του ότι το μεγαλύτερο μέρος της καλύπτεται από το αυτοκίνητο, διακόπτοντας έτσι το μοτίβο (διαγράμμιση παράλληλων λευκών γραμμών) κάνοντας την ανίχνευση ανέφικτη. Στο τέταρο patch με ακρίβεια 88.35 % επειδή η διάβαση κατά την κοπή του patch έτυχε να βρίσκεται κάτω δεξιά, ναι μεν υπάρχει ανίχνευση αλλά το πλαίσιο δεν φαίνεται ολόκληρο, για αυτόν τον λόγο και μετέπειτα στην επανένωση φαίνεται έτσι στην τελική εικόνα. Συνεπώς, σε ποιο σημείο του patch βρίσκονται οι διαβάσεις κάθε φορά επηρεάζει το βαθμό ανίχνευσης. Δυστυχώς, ο τρόπος που χωρίζεται η εικόνα είναι αυτοματοποιημένος και δεν μπορούν να ελεγχθούν τα όρια του κάθε τμήματος ως προς το τι απεικονίζουν. Γενικά όμως παρατηρείται καλή ακρίβεια αφού οι πιθανότητες είναι μεγάλες σε κάθε patch. Ενώ, μόνο μία διάβαση δεν ανιχνεύθηκε και αυτή λόγω έλλειψης ορατότητας.

Σε σχέση με το σφάλμα «κοπής», για την απαλλαγή από αυτό μπορεί να δημιουργηθεί ένα δεύτερος κάνναβος όμοιος με τον πρώτο στις διαστάσεις (416 x 416), ο οποίος θα ξεκινάει από το κεντρικό σημείο του πρώτου patch του πρώτου καννάβου, έτσι ώστε όποια διάβαση στον πρώτο κάνναβο δεν ανιχνεύθηκε λόγω της τοποθεσίας που βρισκόταν στην εικόνα, αυτή τη φορά θα βρίσκεται στο κέντρο του patch και όχι στην άκρη και συνεπώς θα ανιχνευθεί. Στη συνέχεια, πραγματοποιείται η διαδικασία εύρεσης συντεταγμένων των διαβάσεων που ανιχνεύθηκαν και αποθηκεύονται σε ένα αρχείο CSV. Αυτά τα αρχεία έπειτα συγκρίνονται ως προς το ποιες συντεταγμένες είναι κοινές βάσει ενός κατωφλιού. Ενδιαφέρουν αυτές που δεν είναι κοινές, διότι αποτελούν τις νέες διαβάσεις που δεν ανιχνεύθηκαν την πρώτη φορά. Αυτός είναι ένας τρόπος να αντιμετοπιστεί το σφάλμα «κοπής», όμως ο χρόνος προφανώς θα είναι ο διπλάσιος που θα χρειαστεί το πρόγραμμα να τρέξει. (Η διαδικασία εύρεσης των συντεταγμένων αναλύεται παρακάτω).

3.5.2.2 Ανίχνευση διαβάσεων σε τμήμα χάρτη για την πόλη του Μάντσεστερ

Όμοια διαδικασία ακολουθείται και για την πόλη του Μάντσεστερ. Ειδικότερα, λήφθηκε εικόνα 2933 x 1478 λίγο μικρότερη από την προηγούμενη. Η εικόνα αυτή αντιστοιχεί στο 0.04 % της πόλης και συνεπώς για την κάλυψη όλης της πόλης χρειάζονται 100 / 0.04 = 2500 εικόνες. Λόγω του ότι η αρχική εικόνα χωρίζεται σε κομμάτια ίσου μεγέθους 416 x 416 το μέγεθος της αρχικής εικόνας δεν πρέπει να παίζει ρόλο. Επομένως για να επιβεβαιωθεί αυτός ο ισχυρισμός για το Μάντσεστερ λήφθηκαν δύο εικόνες, η παραπάνω και άλλη μία πιο μεγάλη εικόνα 5063 x 2956 (0.13 % της περιοχής). Στην πόλη αυτή γίνεται ανίχνευση διαβάσεων με χρωματισμένη κόκκινη επιφάνεια. Παρακάτω παρατίθεται η αρχική εικόνα 2933 x 1478 και το αποτέλεσμα της ανίχνευσης για την πόλη του Μάντσεστερ :

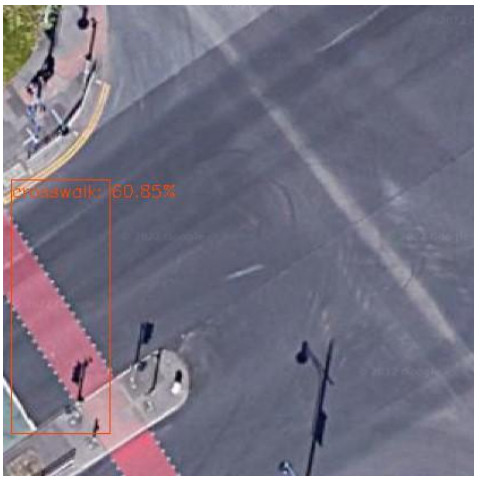
Εικόνα 3.5.2.2 -1 Δορυφορική εικόνα στο κέντρο του Μάντσεστερ πριν την ανίχνευση



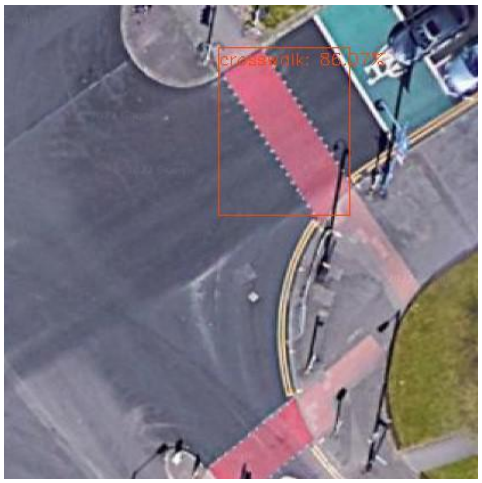
Εικόνα 3.5.2.2 -2 Δορυφορική εικόνα στο κέντρο του Μάντσεστερ μετά την ανίχνευση

Παρακάτω παρατίθενται τα ανιχνεύσιμα patches της παραπάνω εικόνας για καλύτερη οπτικοποίηση :

Εικόνα 3.5.2.2 -3 Patch 95.49 % - Μάντσεστερ

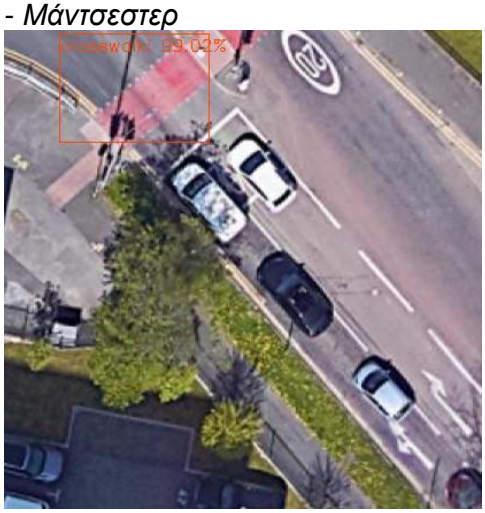


Εικόνα 3.5.2.2 -4 Patch 60.85 % - Μάντσεστερ



Εικόνα 3.5.2.2 -5 Patch 86.07 % - Μάντσεστερ

Εικόνα 3.5.2.2 -6 Patch 70.58 % - Μάντσεστερ



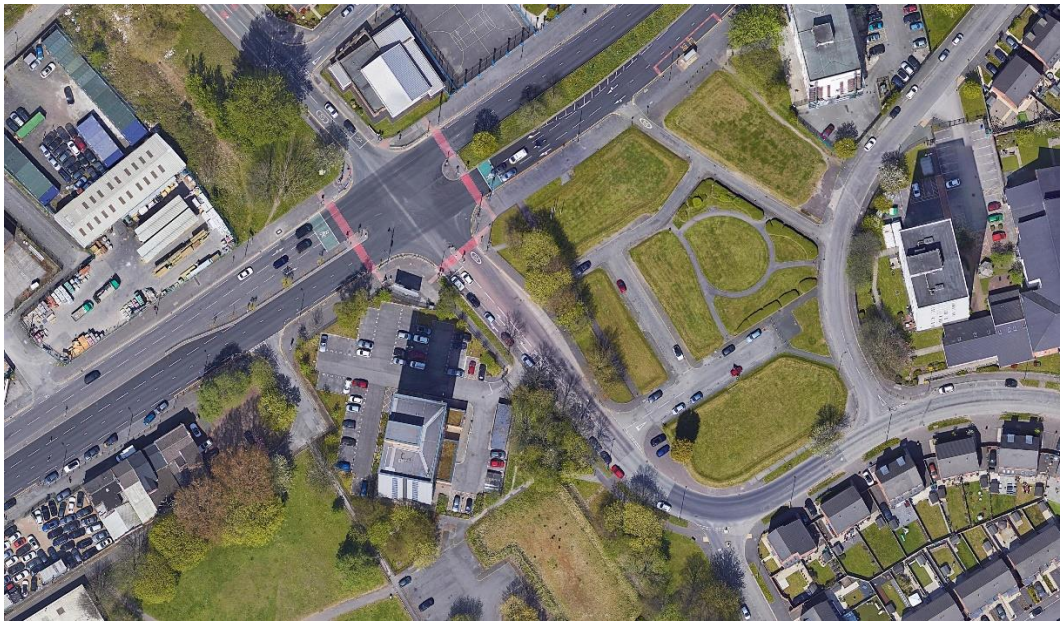
Εικόνα 3.5.2.2 -7 Patch 99.02 % - Μάντσεστερ

```
runfile('D:/backup e-byte/Desktop/b
(3, 7, 1, 416, 416, 3)
predicted object crosswalk: 95.49%
predicted object crosswalk: 60.85%
predicted object crosswalk: 86.07%
predicted object crosswalk: 70.58%
predicted object crosswalk: 99.02%
```

Εικόνα 3.5.2.2 -8 Πιθανότητες από την οθόνη

Στο τρίτο patch με ακρίβεια 86.07 % παρατηρείται πως ανιχνεύθηκε μόνο μία διάβαση, η δεύτερη δεν ανιχνεύθηκε πιθανών λόγω θέσης της διάβασης κάτω αριστερά. Ένα μέρος της είναι αποκομμένο ανομοιόμορφα, οπότε πιθανόν αυτός να είναι ο λόγος. Επίσης, φαίνεται πως είναι καλή η ακρίβεια αφού οι πιθανότητες είναι μεγάλες σε κάθε patch. Μόνο μία διάβαση δεν ανιχνεύθηκε, κάτι το οποίο δεν αποτελεί πρόβλημα στην συγκεκριμένη περίπτωση ή σε κάθε περίπτωση που υφίσταται δύο διαβάσεις με νησίδα στη μέση, διότι η μία από τις δύο ανιχνεύθηκε, που υποδεικνύει την ύπαρξη διάβασης στο χώρο γενικά.

Στη συνέχεια έγινε λήψη της ίδιας εικόνας σε μεγαλύτερο μέγεθος 5063 x 2956. Η αρχική εικόνα και τα αποτελέσματα της ανίχνευσης παρατίθενται παρακάτω :



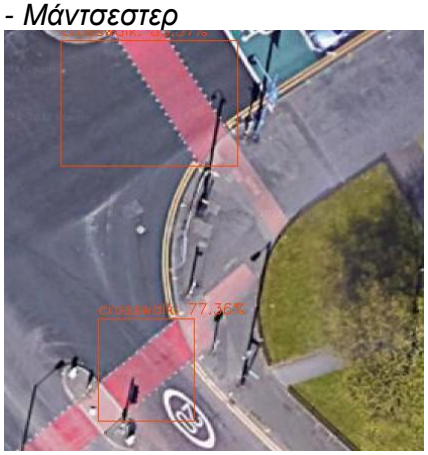
Εικόνα 3.5.2.2 -9 Δορυφορική εικόνα στο κέντρο του Μάντσεστερ πριν την ανίχνευση (μεγάλη εικόνα)

Εικόνα 3.5.2.2 -10 Δορυφορική εικόνα στο κέντρο του Μάντσεστερ μετά την ανίχνευση (μεγάλη εικόνα)

Παρακάτω παρατίθενται τα ανιχνεύσιμα patches της παραπάνω εικόνας για καλύτερη οπτικοποίηση :

Εικόνα 3.5.2.2 -11 Patch 97.59 % - Μάντσεστερ

Εικόνα 3.5.2.2 -12 Patch 91.15 % - Μάντσεστερ



Εικόνα 3.5.2.2 -13 Patch 85.55 % Και 77.36 % και 55.68 % - Μάντσεστερ

```
runfile('C:/Users/valen/Desktop/bri
(7, 12, 1, 416, 416, 3)
predicted object crosswalk: 97.59%
predicted object crosswalk: 91.15%
predicted object crosswalk: 85.57%
predicted object crosswalk: 77.36%
predicted object crosswalk: 55.68%
```

Εικόνα 3.5.2.2 -14 Πιθανότητες από την οθόνη

Σε αυτήν την περίπτωση παρατηρούνται ανίχνευση διάβασης που στην μικρότερη εικόνα δεν είχε ανιχνευθεί Εικόνα 3.5.2.1 -21 με ακρίβεια 77.36%. Πάλι δεν ανιχνεύθηκε μία διάβαση και άλλη μία δεν σχεδιάστηκε το Bounding Box και αυτό σχετίζεται με τη θέση της κάθε διάβασης κατά την κοπή της εικόνας σε κομμάτια. Επίσης, παρατηρείται καλή η ακρίβεια αφού οι πιθανότητες είναι μεγάλες σε κάθε patch. Συνεπώς, το μέγεθος του τμήματος του χάρτη δεν έχει σημασία στο αποτέλεσμα. Μόνο η θέση της κάθε διάβασης κατά τη διάρκεια της κοπής της εικόνας σε κομμάτια επηρεάζει το αποτέλεσμα και τον βαθμό ανίχνευσης, κάτι το οποίο σχετίζεται με τον τρόπο κοπής και δεν μπορεί να επέλθει μεταβολή. Παρόλα αυτά όμως, ακόμα και με αυτό το σφάλμα της «κοπής» το αποτέλεσμα σε όλες τις περιπτώσεις που μελετήθηκαν είναι καλό.

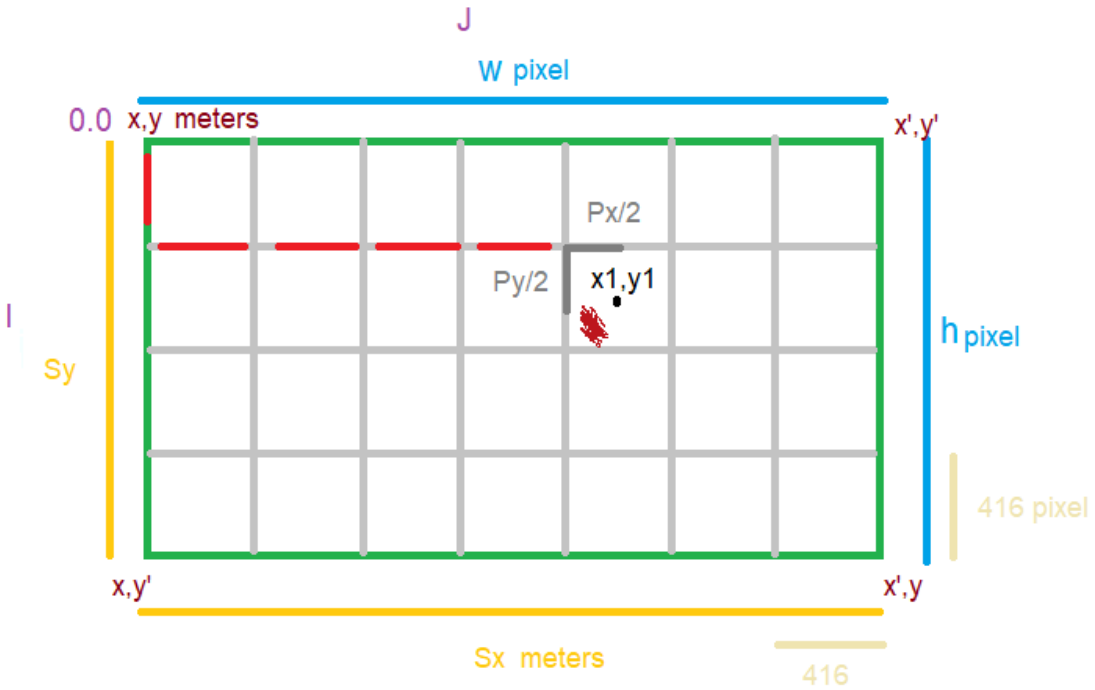
Το γεγονός λοιπόν, ότι το αποτέλεσμα είναι ανεξάρτητο του μεγέθους της αρχικής εικόνας συμβάλλει στη χρήση του μοντέλου για οποιοδήποτε τμήμα χάρτη, είτε σε μέγεθος, είτε όσον αφορά την περιοχή μελέτης. Αυτό συμβαίνει, διότι όπως φάνηκε παραπάνω στον έλεγχο των βαρών στην πόλη του Μπαθ, δηλαδή σε άλλη πόλη, το αποτέλεσμα της ανίχνευσης διαβάσεων με ζέμπρα διαγράμμιση όπως και στο Μπρίστολ, είναι καλό. Συνεπώς, η ανίχνευση διαβάσεων με διαγράμμιση ζέβρα, ή με χρωματισμό κόκκινης επιφάνειας, με χρήση του YOLOV3 μπορεί να επέλθει για οποιοδήποτε τμήμα χάρτη και για οποιαδήποτε τοποθεσία στον κόσμο.

Για το επόμενο βήμα, την προσεγγιστική εύρεση των γεωγραφικών συντεταγμένων των διαβάσεων που ανιχνεύτηκαν για την πόλη του Μπρίστολ και την πόλη του Μάντσεστερ θα χρησιμοποιηθεί η ίδια εικόνα που χρησιμοποιήθηκε για την ανίχνευση

για το Μπρίστολ ενώ για το Μάντσεστερ επιλέγεται η μικρή εικόνα, μιας και δεν παίζει ρόλο το μέγεθος. Η προσεγγιστική εύρεση των συντεταγμένων αναλύεται εκτενέστερα παρακάτω.

3.6 Εύρεση γεωγραφικών συντεταγμένων διαβάσεων

Η διπλωματική εργασία εκτός από την ανίχνευση των διαβάσεων με διαγράμμιση ζέβρα και διαβάσεων με χρωματισμένη κόκκινη επιφάνεια στο οδόστρωμα, σκοπεύει και στην εύρεση των γεωγραφικών συντεταγμένων των διαβάσεων που ανιχνεύονται στις πόλεις Μπρίστολ και Μάντσεστερ και την μετέπειτα χαρτογράφησή τους. Το αποτέλεσμα από το προηγούμενο στάδιο (3.5) αποτελεί μία εικόνα – τμήμα χάρτη που απεικονίζει τις διαβάζεις που ανιχνεύθηκαν σε αυτήν. Όμως, για την εύρεση των διαβάσεων χρειάζεται μια πιο προσεκτική ματιά στις μικρές εικόνες πριν την ένωση, κατά τη διάρκεια της ανίχνευσης. Παρακάτω παρατίθεται ένα σχήμα για την υπάρχουσα κατάσταση :



Εικόνα 3.6 -3 Οπτικοποίηση σχέσης μεταξύ αρχικής εικόνας – τμήμα χάρτη και των χωρισμένων patches κατά τη διάρκεια της ανίχνευσης

Σύμφωνα με την παραπάνω εικόνα :

- Το πράσινο πλαίσιο αντιπροσωπεύει την αρχική εικόνα – τμήμα χάρτη. Οι συντεταγμένες της είναι (x , y) το σημείο πάνω αριστερά, (x' ,y) το σημείο κάτω δεξιά και (x' , y'), (x , y') τα σημεία πάνω δεξιά και κάτω αριστερά αντίστοιχα. Οι συντεταγμένες αυτές είναι γνωστές από το λογισμικό QGIS, από το οποίο λήφθηκαν οι εικόνες, όπως φαίνεται παρακάτω στην εικόνα για τις δύο πόλεις.
- Τα γκρί – ανοιχτό αποτελούν τις πλευρές των patches σε μέτρα, όπου οι πλευρές : Px = Py (m), οι οποίες αντιστοιχούν σε 416 pixels.

- Με το κόκκινο χρώμα απεικονίζονται το πλήθος των παραπάνω πλευρών Px , Py πριν το ζητούμενο patch.
- Οι σκούρο γκρι γραμμές αντιστοιχούν τις μισές πλευρές ενός patch, ενώ ταυτόχρονα αποτελούν και το κέντρο του ξεκινώντας από το σημείο πάνω αριστερά του ίδιου patch. Μεταφράζονται σε Px / 2, Py / 2.
- Το καφέ σημείο στο (i , j) = (1 , 4) patch αποτελεί μια διάβαση. Η διάβαση αυτή λαμβάνεται ως παράδειγμα στο συγκεκριμένο εικονίδιο.
- h , w : ύψος και πλάτος των αρχικών εικόνων. Αυτά είναι επίσης γνωστά από την ίδια την εικόνα καθώς και από το λογισμικό QGIS.

Παρατίθεται παρακάτω για την πόλη του Μπρίστολ και του Μάντσεστερ τα γνωστά «δεδομένα» από το QGIS λογισμικό :

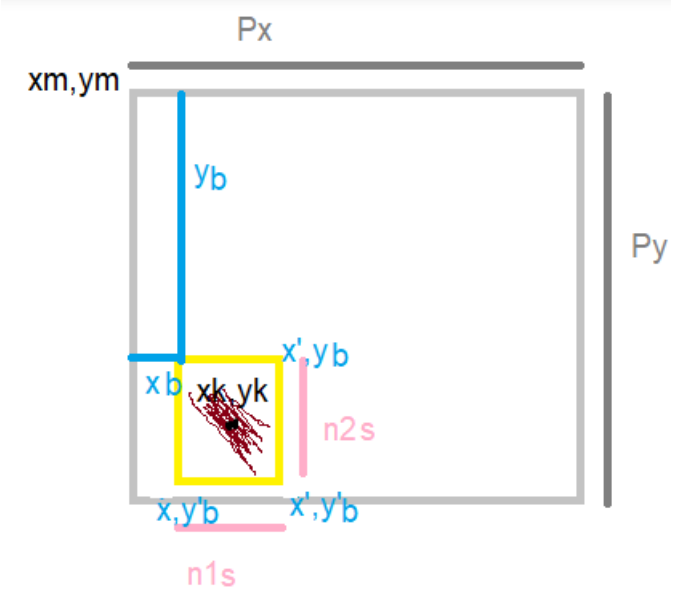
CRS	EPSG:3857 - WGS 84 / Pseudo-Mercator - Projected
Extent	-290993.8219999999855645,6702599.0838000001385808 : -290692.3745999999810010,6702775.1185999996960163
Unit	meters
Width	3014
Height	1760

Εικόνα 3.6 -4 Δεδομένα που αποκτούνται από το λογισμικό QGIS – Μπρίστολ

CRS	EPSG:3857 - WGS 84 / Pseudo-Mercator - Projected
Extent	-247671.6735000000044238,7073751.0492000002413988 : -247378.3985999999858905,7073898.8656000001356006
Unit	meters
Width	2933
Height	1478

Εικόνα 3.6 -3 Δεδομένα που αποκτούνται από το λογισμικό QGIS – Μάντσεστερ

Παρακάτω παρατίθεται σε μεγέθυνση το ζητούμενο (i , j) = (1 , 4) patch :



Εικόνα 3.6 -4 Δεδομένα ζητούμενου patch – Μάντσεστερ

Σύμφωνα με την παραπάνω εικόνα :

- Το γκρι πλαίσιο αντιπροσωπεύει ένα patch όπου έχει ανιχνευθεί διάβαση.

- Το κίτρινο πλαίσιο αποτελεί ένα Bounding box, το οποίο και περιέχει το ζητούμενο αντικείμενο.
- Τα (xb , yb) και (x'b, y'b) είναι οι εικονοσυντεταγμένες του Bounding box από την ανίχνευση με το πρώτο να είναι το αρχικό σημείο πάνω αριστερά και το δεύτερο το τελικό σημείο κάτω δεξιά. Η μέτρηση τους σχετίζεται με το patch θεωρώντας το (xm, ym) = (0, 0) και τελικό (416, 416) pixels. Βάση θεωρίας θεωρείται αρχικό (0, 0) και τελικό (1, 1) όμως κατά την ανίχνευση όπως αναλύθηκε παραπάνω πολλαπλασιάζοντας τις συντεταγμένες του Bounding box με τα πλάτος και ύψος του patch το Bounding box εισέρχεται στο επίπεδο του αντικειμένου.
- n1s, n2s : αποτελούν τη διαφορά των συντεταγμένων του Bounding box σε μέτρα.
- xm, ym : είναι οι συντεταγμένες πάνω αριστερά του patch
- xk, yk : αποτελούν τις ζητούμενες συντεταγμένες. Είναι το κεντρικό σημείο του Bounding box.

Συνεπώς, βάσει όλων των παραπάνω είναι εμφανές πως πολλά δεδομένα είναι σε μορφή pixel, όπως οι πλευρές των εικονιδίων, το ύψος και το πλάτος της αρχικής εικόνας. Το ζητούμενο όμως είναι η εύρεση των συντεταγμένων των διαβάσεων, που σημαίνει πως χρειάζεται μια μετατροπή σε μέτρα. Αρχικά, γίνεται μετατροπή του ύψους και του πλάτους σε μέτρα και υπολογίζεται η αντίστοιχη τιμή των πλευρών των εικονιδίων με τη βοήθεια των μαθηματικών και συγκεκριμένα της μεθόδου των τριών, λαμβάνοντας υπόψη ότι η διαφορά των συντεταγμένων x και y μεταξύ τους αντίστοιχα σε απόλυτη τιμή είναι ισότιμες με τις αντίστοιχες πλευρές του πλάτους και του ύψους σε μέτρα όπως φαίνεται και στην εικόνα παραπάνω. Δηλαδή :

- $|y - y'| = Sy$ (m), $|x - x'| = Sx$ (m).

Επομένως η μέθοδος των τριών έχει ως εξής :

- $Py = Sy * 416 / h$ (m)
- $Px = Sx * 416 / w$ (m).
- Μάλιστα, λόγω μεγέθους του patch (416 x 416) πρέπει Py = Px (m) όπως αναφέρθηκε παραπάνω.

Επομένως, πλέον το ύψος και το πλάτος έχουν μετατραπεί σε μέτρα καθώς αντίστοιχα και η πλευρά του patch.

Στη συνέχεια ως προς την εύρεση των διαβάσεων, έστω όπως φαίνεται και στην πρώτη εικόνα η ζητούμενη διάβαση να βρίσκεται στο patch (i , j) = (1 , 4) με τη μέτρηση να ξεκινάει από το (0 , 0). Ζητείται το xk , yk κεντρικό σημείο του Bounding box του εικονιδίου όπου είναι η διάβαση. Πρακτικά επιλέχθηκε η εύρεση του κεντρικού σημείου του Bounding box του εικονιδίου, διότι οι διαβάσεις ή η διάβαση περιέχονται ή περιέχεται σε αυτό, οπότε τώρα υπολογίζεται και η τοποθεσία των διαβάσεων στην πραγματικότητα.

Βάσει της πρώτης εικόνας παραπάνω, παρατηρείται πως είναι σημειωμένες κάποιες κόκκινες και σκούρο γκρι γραμμές. Οι γραμμές αυτές δηλώνουν τον τρόπο εύρεσης του πάνω αριστερά σημείου του εικονιδίου. Ειδικότερα, για το xm από το αρχικό x προστίθενται όσες στήλες J υφίστανται πριν το ζητούμενο patch. Αντίθετα, για το ym από το αρχικό y αφαιρούνται όσες γραμμές I υφίστανται πριν το ζητούμενο patch. Όμως, το ζητούμενο εικονίδιο βρίσκεται στη στήλη νούμερο τέσσερα (4) ξεκινώντας την αρίθμηση από το μηδέν (0) που συνεπάγεται πως κατά τη μέτρηση των στηλών κάθε αυτών βρίσκεται στην πέμπτη στήλη. Αυτό είναι σημαντικό, διότι για την εύρεση

του κεντρικού σημείου του εικονιδίου δεν λαμβάνεται υπόψη η αρίθμηση μηδέν κατά τη μέτρηση των στηλών και αντίστοιχα γραμμών για το ym. Οπότε, η ζητούμενη διάβαση ή οι ζητούμενες διαβάσεις βρίσκονται στην πέμπτη στήλη ουσιαστικά και ζητούνται όλες οι προηγούμενες για την εύρεση του xm και αντίστοιχα του ym ως προς τις γραμμές. Επομένως, χρειάζεται το J = 4, ώστε βρισκόμενο το εικονίδιο στην πέμπτη στήλη το J = 4 είναι το πλήθος των προηγουμένων. Παρατηρείται πως αυτή είναι και η τιμή του εικονιδίου (i, j) = (1 , 4), ενώ ομοίως συμβαίνει και ως προς τις γραμμές για την εύρεση του ym. Συνεπώς, αντίστοιχα λαμβάνονται υπόψη, για την εύρεση των συντεταγμένων, τα I, J αυτούσια των ζητούμενων εικονιδίων patches.

Εν κατακλείδι σύμφωνα με όλα τα παραπάνω προκύπτουν οι εξής συναρτήσεις για την εύρεση των xm και ym :

- xm = x + (Px * J) (μέτρα)
- ym = y – (Py * i) (μέτρα)

Συνεχίζοντας στη δεύτερη εικόνα παρατηρείται πως για την εύρεση του κεντρικού σημείου του Bounding box χρειάζεται οι εικονοσυντεταγμένες να μετατραπούν σε μέτρα. Παρατηρείται πως στο σημείο πάνω αριστερά (xb, yb) του Bounding box, τα xb, yb αποτελούν σε απόλυτη τιμή τις μπλε πλευρές στην δεύτερη εικόνα όπου η μέτρηση ξεκινά από το (0, 0) και τελειώνει στο (416, 416). Ουσιαστικά αυτή είναι και η σχέση του Bounding box με το patch. Συνεπώς χρειάζονται οι τιμές αυτές σε μέτρα ώστε έπειτα από το επίπεδο του Bounding box να βρεθούν οι συντεταγμένες του κεντρικού σημείου.

Πρακτικά γίνεται η παρακάτω μεταβίβαση επιπέδων : αρχική εικόνα → patch → Bounding box → κεντρικό σημείο του Bounding box.

Συνεπώς, με τη χρήση της μεθόδου των τριών προκύπτουν οι εξής εξισώσεις για τα xb, yb σε σχέση με το patch :

- x_s = |(xb * Px) / 416| (μέτρα) (απόλυτη τιμή, διότι ζητούνται οι μπλε πλευρές όπως φαίνεται στην δεύτερη εικόνα)
- y_s = |(yb * Py) / 416| (μέτρα)

Ύστερα για την εκτίμηση του κεντρικού σημείου είναι απαραίτητες οι n1s, n2s πλευρές και συγκεκριμένα τα μισά τους. Συνεπώς πρώτα υπολογίζονται οι πλευρές των εικονοσυντεταγμένων n1, n2 και στη συνέχεια με μέθοδο των τριών σε σχέση με το patch εκτιμιούνται οι πλευρές σε μέτρα. Προκύπτουν οι παρακάτω εξισώσεις :

- n1 = |x’b – xb|
- n2 = |y’b – yb|
- n1s = (n1* Px) / 416 (μέτρα)
- n2s = (n2* Py) / 416 (μέτρα)

Λαμβάνοντας όλα τα παραπάνω υπόψη οι τελικές εξισώσεις για το κεντρικό σημείο του Bounding box, για τη συγκεκριμένη διάβαση του παραδείγματος αλλά και για κάθε διάβαση που ανιχνεύεται και για τις δύο πόλεις, είναι οι εξής :

- xk = xm + x_s + n1s / 2 (μέτρα)
- yk = ym - y_s – n2s / 2 (μέτρα)

Να σημειωθεί πως οι συντεταγμένες παράγονται σε όποιο γεωδαιτικό σύστημα αντιστοιχεί η αρχική εικόνα, αφού χρησιμοποιούνται οι δικές της συντεταγμένες ως σημείο αναφοράς (σημείο πάνω αριστερά) για την εύρεση τους. Στην εφαρμογή της

διπλωματικής εργασίας το γεωδαιτικό σύστημα της αρχικής εικόνας είναι το WGS 84 , είναι αυτό που χρησιμοποιεί η Google και συνεπώς και οι προσεγγιστικές συντεταγμένες αντιστοιχούν σε αυτό το γεωδαιτικό σύστημα.

Χρησιμοποιώντας τις παραπάνω συναρτήσεις σε κάθε patch, στο οποίο ανιχνεύεται κάποια διάβαση παράγονται οι γεωγραφικές συντεταγμένες των διαβάσεων αυτών. Οι συντεταγμένες αυτές με τη σειρά τους αποθηκεύονται αυτόματα σε ένα CSV αρχείο, το οποίο στη συνέχεια εισέρχεται στο λογισμικό QGIS και παρουσιάζει τις συντεταγμένες με μορφή στίγματος. Έτσι, επιτυγχάνεται η χαρτογράφηση των διαβάσεων που ανιχνεύθηκαν. Παρακάτω παρατίθενται οι διαβάσεις για τα δύο τμήματα χαρτών για το Μπρίστολ και το Μάντσεστερ αντίστοιχα, όπως αυτές προβάλλονται μέσω του αρχείου CVS που δημιουργήθηκε :

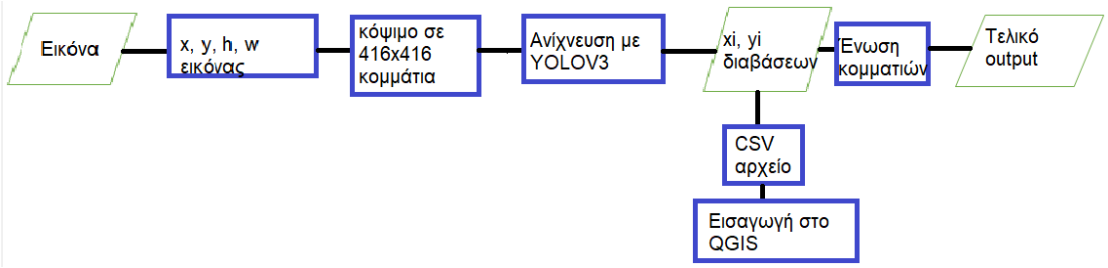
Εικόνα 3.6 -5 Χαρτογράφηση των ανιχνευμένων διαβάσεων με διαγράμμιση ζέβρα στην πόλη του Μπρίστολ

Εικόνα 3.6 -6 Χαρτογράφηση των ανιχνευμένων διαβάσεων με χρωματισμένη κόκκινη επιφάνεια στην πόλη του Μάντσεστερ

Όπως παρατηρείται παραπάνω οι συντεταγμένες είναι αντιπροσωπευτικές της ακριβής τοποθεσίας των διαβάσεων και συνεπώς μπορεί να χρησιμοποιηθεί το μοντέλο σε επίπεδο πόλης. Όπως όμως προκύπτει από την μεθοδολογία στο υποκεφάλαιο 3.5 το μέγεθος της αρχικής εικόνας δεν επηρεάζει την ανίχνευση, επομένως το μοντέλο μπορεί να χρησιμοποιηθεί και για μεγαλύτερες περιοχές.

3.7 Διάγραμμα ροής των εργασιών

Παρακάτω παρατίθεται ένα διάγραμμα ροής, το οποίο περιγράφει την διαδικασία ως προς την λήψη των συντεταγμένων των διαβάσεων σε τμήμα χάρτη, με τη χρήση της μεθοδολογίας που αναπτύχθηκε :



Εικόνα 3.7 -1 Διάγραμμα ροής για την χαρτογράφηση διαβάσεων σε επίπεδο πόλης

Ουσιαστικά συλλέγεται η εικόνα με το ζητούμενο αντικείμενο ή αντικείμενα και έχοντας λάβει υπόψη τις αρχικές συντεταγμένες και το ύψος και το πλάτος της εικόνας αυτής από το QGIS, με χρήση YOLOV3 πραγματοποιείται η ανίχνευση και παράγονται

προσεγγιστικά οι συντεταγμένες των διαβάσεων. Αυτές μέσω ενός αρχείου CSV δύνανται να εισαχθούν στο λογισμικό QGIS. Ο κώδικας για την ανίχνευση – παραγωγή συντεταγμένων παρατίθεται στο παράρτημα. Ο χρήστης χρειάζεται μόνο να γνωρίζει τα x, y, h, w της αρχικής εικόνας, να ενημερώσει τον κώδικα με αυτά τα στοιχεία, όπως και το path στο οποίο επιθυμεί ο χρήστης να αποθηκευτούν οι φάκελοι με τα patches, τις ανιχνεύσεις και το CSV αρχείο. Ο κώδικας παρατίθεται στο παράρτημα.

Η εικόνα ανεξαρτήτου μεγέθους χωρίζεται σε 416 x 416 κομμάτια και σε αυτά εφαρμόζεται η ανίχνευση. Έπειτα για τα ανιχνεύσιμα patches παράγονται οι συντεταγμένες των διαβάσεων, οι οποίες αποθηκεύονται αυτόματα σε ένα αρχείο CSV. Το αρχείο αυτό έπειτα εισέρχεται στο λογισμικό QGIS για τη χαρτογράφηση των ανιχνευμένων διαβάσεων. Στο λογισμικό οι συντεταγμένες παρουσιάζονται με τη μορφή στιγμάτων.

3.8 Χρόνος επεξεργασίας

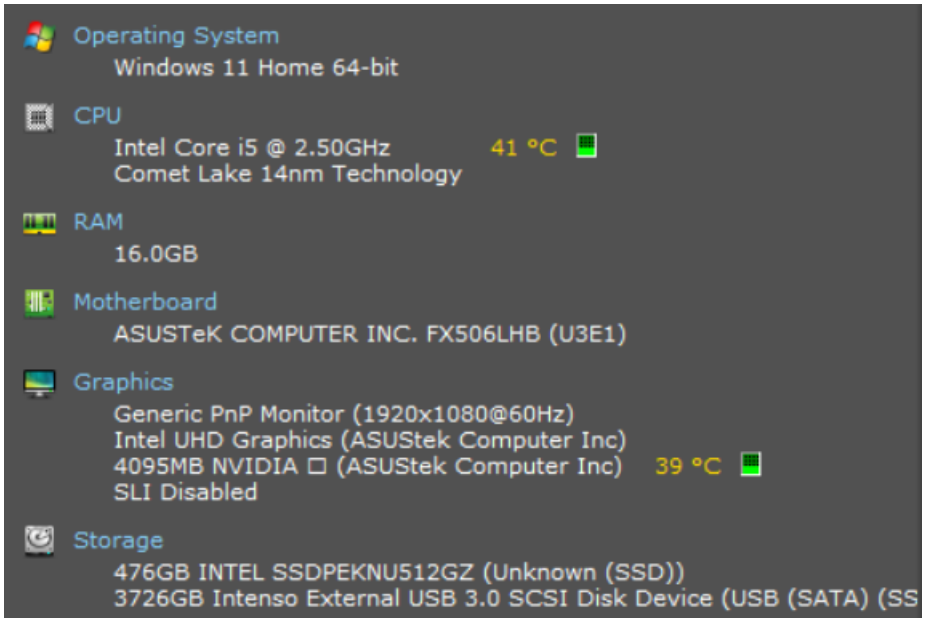
Όσον αφορά την εκπαίδευση του μοντέλου ο χρόνος επεξεργασίας ήταν αρκετός. Πιο συγκεκριμένα, όπως αναφέρθηκε και στο αντίστοιχο υποκεφάλαιο χρειάζονται περίπου 12 ώρες εκπαίδευσης για την κάθε πόλη. Αυτό μάλιστα φαίνεται και στα διαγράμματα με την ακρίβεια της εκπαίδευσης, όπου παρουσιάζεται ο υπολειπόμενος χρόνος για το πέρας των 6000 επαναλήψεων. Εκεί όπως προαναφέρθηκε βάση ενός εμπειρικού κανόνα η εκπαίδευση σταμάτησε όταν το σφάλμα ήταν λιγότερο του 0.06 περίπου. Για να φτάσει σε αυτό το επίπεδο η εκπαίδευση και να μπορέσει να σταματήσει χρειάστηκαν 6 ώρες για κάθε πόλη. Η ακρίβεια όμως του μοντέλου ήταν πολύ καλή, παρά το γεγονός ότι στην εκπαίδευση συμμετείχαν μόνο 200 εικόνες για κάθε πόλη. Θετικό είναι το γεγονός πώς το προτεινόμενο μοντέλο δεν χρειάζεται εκπαίδευση ξανά από το χρήστη, οπότε ο χρόνος αυτός δεν αποτελεί πρόβλημα για το χρήστη.

Όσον αφορά την ανίχνευση των διαβάσεων για τον υπολογισμό των συντεταγμένων τους ο χρόνος επεξεργασίας έχει ως εξής :

- Για την πόλη του Μπρίστολ από την είσοδο της εικόνας στο μοντέλο, τη χρήση του μοντέλου και την παραγωγή των συντεταγμένων αποθηκευμένες σε ένα CSV αρχείο, το πρόγραμμα έτρεξε για περίπου 25.5 δευτερόλεπτα για μια εικόνα μεγέθους 3014 x 1760 pixels και ανάλυση resolution 10 cm = 0.1 m. Συνεπώς γίνεται λόγος για 3014 * 1760 * 0.1^2 = 53046.4 τ.μ. = 53 στρέμματα περίπου. Οπότε για μια περιοχή 53 στρεμμάτων χρειάστηκαν 25.5 δευτερόλεπτα. Για 1000 στρέμματα = 1 τ. χλμ. χρειάζονται λοιπόν 25.5 * 1000 / 53 = 481.13 δευτερόλεπτα = 8.01 = 8 λεπτά. Για την ανίχνευση και υπολογισμό διαβάσεων με διαγράμμιση ζέβρα για περιοχή 1 τ. χλμ. χρειάζεται περίπου ένα οχτάλεπτο. Επίσης για 1 στρέμμα χρειάζεται περίπου 25.5 / 53 = 0.5 δευτερόλεπτα.
- Για την πόλη του Μάντσεστερ έτρεξε το πρόγραμμα για μια εικόνα μεγέθους 2933 x 1478 pixels, δηλαδή 2933 * 1478 * 0.1^2 = 43349.74 τ.μ. = 43 στρέμματα. Το πρόγραμμα έτρεξε για 20.25 δεύτερα. Δηλαδή για 1 στρέμμα χρειάζεται περίπου 20.25 / 43 = 0.47 = 0.5 δευτερόλεπτα. Παρατηρείται λοιπόν πως και στις δύο πόλεις τα αποτελέσματα ταιριάζουν. Η διαφορά στην τιμή του χρόνου που έτρεξε το πρόγραμμα οφείλεται στην διαφορετική έκταση της πόλης.

Συνεπώς αποδεικνύεται και από τις δύο πόλεις πως για κάθε στρέμμα χρειάζονται 0.5 δεύτερα και για 1 τ. χλμ. γύρω στα 8 λεπτά. Επίσης, αποδεικνύεται πως ο χρόνος εξαρτάται μόνο από την έκταση της περιοχής μελέτης.

Να σημειωθεί πως ο χρόνος CPU είναι 46 δεύτερα περίπου και δηλώνει τον χρόνο που ο υπολογιστής έκανε υπολογισμούς για το πρόγραμμα. Τα τεχνικά χαρακτηριστικά του υπολογιστή που έτρεξε το πρόγραμμα φαίνονται παρακάτω χρησιμοποιώντας το Speccy λογισμικό :



Εικόνα 3.8 -1 Τεχνικά χαρακτηριστικά Η/Υ

4. Συμπεράσματα και Προοπτικές

4.1 Παρατηρήσεις – Συμπεράσματα

Ο σκοπός της εργασίας είναι η ανίχνευση και χαρτογράφηση των διαβάσεων πεζών σε επίπεδο πόλης χρησιμοποιώντας τεχνικές βαθιάς μάθησης. Πιο συγκεκριμένα ανιχνεύονται διαβάσεις με διαγράμμιση ζέβρα και διαβάσεις με επιφάνεια κόκκινου χρωματισμού στο οδόστρωμα για τις πόλεις του Μπρίστολ και του Μάντσεστερ αντίστοιχα με χρήση ποικίλων νευρωνικών δικτύων καθώς και του YOLOV3. Έπειτα από σύγκριση των τεχνικών βαθιάς μάθησης η χαρτογράφηση έγινε, με την καλύτερη τεχνική για ανίχνευση, το YOLOV3. Εκπαιδεύτηκε το μοντέλο, έγινε η ανίχνευση των διαβάσεων, υπολογίστηκαν οι συντεταγμένες των διαβάσεων που ανιχνευθήκαν και στη συνέχεια χαρτογραφήθηκαν στο λογισμικό QGIS. Λαμβάνοντας λοιπόν το κεφάλαιο της Μεθοδολογίας και Αξιολόγησης υπόψη προκύπτουν οι παρακάτω παρατηρήσεις - συμπεράσματα :

- Ο αλγόριθμος YOLOV3 όντως αποτελεί εξαίρετο εργαλείο ως προς την ανίχνευση αντικειμένων, όπως αναφέρει και το θεωρητικό υπόβαθρο. Ειδικότερα, στην παρούσα διπλωματική εργασία η εκπαίδευση κατείχε πάρα πολύ καλή ακρίβεια (98 % για το Μάντσεστερ και 100 % για το Μπρίστολ), καθώς και η ανίχνευση στη συνέχεια με χρήση των αντίστοιχων βαρών από την εκπαίδευση, πέτυχε υψηλές πιθανότητες > 90 %, είτε στις δοκιμαστικές μεμονωμένες εικόνες με διαβάσεις είτε στα τμήματα χαρτών για τις δύο πόλεις.
- Αναδείχθηκε καλύτερο από τις άλλες τεχνικές βαθιάς μάθησης.
- Μάλιστα ως προς τις δοκιμαστικές εικόνες παρατηρείται πως το μοντέλο μπορεί να χρησιμοποιηθεί για οποιοδήποτε τοποθεσία στον κόσμο που χρήζει ανίχνευσης διαβάσεων με διαγράμμιση ζέβρα και διαβάσεων με κόκκινη χρωματισμένη επιφάνεια στο οδόστρωμα. Καθώς επίσης, δεν παίζει ρόλο αν η εικόνα συμμετέχει η όχι στην εκπαίδευση.
- Στα τμήματα χάρτη οι πιθανότητες ξεπερνούν τις 50 %, που τις κάνει αποδεκτές και κυρίως αγγίζουν το 98 και 99 %. Μόνο η θέση της κάθε διάβασης κατά τη διάρκεια της κοπής της εικόνας σε κομμάτια επηρεάζει το αποτέλεσμα και τον βαθμό ανίχνευσης, κάτι το οποίο σχετίζεται με τον τρόπο κοπής και δεν μπορεί να επέλθει μεταβολή. Παρόλα αυτά όμως, ακόμα και με αυτό το σφάλμα της «κοπής» το αποτέλεσμα σε όλες τις περιπτώσεις που μελετήθηκαν είναι καλό.
- Όπως αναφέρθηκε μόνο η θέση του ζητούμενου αντικειμένου επηρεάζει το αποτέλεσμα και τον βαθμό ανίχνευσης. Το αποτέλεσμα λοιπόν, όπως προέκυψε στο προηγούμενο κεφάλαιο της Μεθοδολογίας και Αξιολόγησης, είναι ανεξάρτητο από το μέγεθος της αρχικής εικόνας αφού στη συνέχεια επέρχεται διαχωρισμός σε «κομμάτια» - εικονίδια patches ίσου μεγέθους 416 x 416. Η ανίχνευση εφαρμόζεται στα εικονίδια αυτά, επομένως αυτό δεν μεταβάλλεται όποιο και αν είναι το μέγεθος της αρχικής εικόνας.
- Επίσης παρατηρείται πως με τον υπολογισμό του κεντρικού σημείου του Bounding Box του εικονιδίου patch, η κάθε διάβαση στη συνέχεια χαρτογραφείται ως ένα σημείο.
- Από την εύρεση των συντεταγμένων των ανιχνεύσιμων διαβάσεων στις δύο πόλεις παρατηρείται πως οι συντεταγμένες αυτές είναι αντιπροσωπευτικές της ακριβής τοποθεσίας των διαβάσεων και συνεπώς μπορεί να χρησιμοποιηθεί το μοντέλο σε επίπεδο πόλης.

- Αποφαίνεται πως για κάθε στρέμμα χρειάζονται 0.5 δεύτερα και για 1 τ. χλμ. 8 λεπτά. Μάλιστα ο χρόνος εξαρτάται από την απόσταση της περιοχής μελέτης.

4.2 Προοπτικές

- Μπορεί να πραγματοποιηθεί εφαρμογή του προτεινόμενου μοντέλου και σε άλλες πόλεις, όπως στην Αθήνα. Κατά το σχεδιασμό και μελέτη έργων όπως ένα γήπεδο ή ένας χώρος αναψυχής γενικότερα, είναι σημαντική η ύπαρξη διαβάσεων και με το προτεινόμενο μοντέλο δύναται για να μελετηθεί η αντίστοιχη περιοχή ως προς την ύπαρξή τους, ενώ στην περίπτωση που δεν υφίστανται να χωροθετηθούν.
- Όπως αναφέρθηκε στην εισαγωγή οι διαβάσεις ειδοποιούν τους οδηγούς να μειώσουν ταχύτητα, διότι πλησιάζουν σε περιοχή διέλευσης πεζών, με σκοπό την αποφυγή ατυχημάτων. Με αυτόν τον σκοπό μπορεί το μοντέλο να εισέλθει σε εφαρμογή πλοήγησης ώστε κατά την οδήγηση οι οδηγοί να ειδοποιούνται πως πλησιάζουν σε διάβαση και πρέπει να μειώσουν ταχύτητα για αποφυγή ατυχήματος. Αυτό οδηγεί σε επαγρύπνηση τους οδηγούς αλλά και σε μεγαλύτερη ασφάλεια στο δρόμο.
- Το προτεινόμενο μοντέλο ενδιαφέρον θα είχε να εφαρμοστεί με το ήδη υπάρχον σύνολο δεδομένων διαβάσεων ως προς μελέτης της πιθανότητας ανίχνευσης σε εικόνες με το αντικείμενο να φαίνεται πλαγίως. Αν τα αποτελέσματα είναι θετικά, τότε μπορεί το μοντέλο να χρησιμοποιηθεί σε εφαρμογές για αυτόνομα αυτοκίνητα και για ανθρώπους με προβλήματα όρασης.

Αναφορές

Ahmetovic, D., Manduchi, R., Coughlan, J. M., & Mascetti, S. (2015). *Zebra Crossing Spotter: Automatic Population of Spatial Databases for Increased Safety of Blind Travelers*. http://wheelmap.org

Berriel, R. F., Lopes, A. T., de Souza, A. F., & Oliveira-Santos, T. (2017). Deep Learning-Based Large-Scale Automatic Satellite Crosswalk Classification. *IEEE Geoscience and Remote Sensing Letters*, *14*(9), 1513–1517. https://doi.org/10.1109/LGRS.2017.2719863

Bochkovskiy Alexey. (n.d.). *Windows and Linux version of Darknet Yolo v3 & v2 Neural Networks for object detection (Tensor Cores are used)*. Retrieved February 6, 2023, from https://github.com/AlexeyAB/darknet

byteant. (2022, July 25). *Computer Vision .vs Machine Learning .vs Deep Learning | Guide to AI applications*. Byteant.Com. https://www.byteant.com/blog/computer-vision-vs-machine-learning-vs-deep-learning-guide-to-ai-applications/

City & County of San Francisco. (2015). *Crosswalks*. Sfbetterstreets.Org. https://www.sfbetterstreets.org/find-project-types/pedestrian-safety-and-traffic-calming/crosswalks/

City & County of San Francisco. (2015). *High Visibility Crosswalk Treatments*. Sfbetterstreets.Org. https://www.sfbetterstreets.org/find-project-types/pedestrian-safety-and-traffic-calming/crosswalks/

datahacker.rs. (2019, November 14). *YOLOv3 Architecture*. https://datahacker.rs/tensorflow2-0-yolov3/

Dow, C., Ngo, H., Lee, L., Lai, P., Wang, K., & Bui, V. (2020). A crosswalk pedestrian recognition system by using deep learning and zebra-crossing recognition techniques. *Software: Practice and Experience*, *50*(5), 630–644. https://doi.org/10.1002/spe.2742

Ghilardi, M. C., Jacques Junior, J., & Manssour, I. (2018). Crosswalk localization from low resolution satellite images to assist visually impaired people. *IEEE Computer Graphics and Applications*, *38*(1), 30–46. https://doi.org/10.1109/MCG.2016.50

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. https://arxiv.org/pdf/1512.03385.pdf

Hui, J. (2018a, March 27). *FPN*. Jonathan-Hui.Medium.Com. https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

Hui, J. (2018b, March 27). *Understanding Feature Pyramid Networks for object detection (FPN)*. Jonathan-Hui.Medium.Com. https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

Karatzaferis Odyssefs. (2022). *Crosswalk detection for the outdoor navigation of people with visual impairment*. http://www.gdmc.nl/publications/2022/MScThessiOdyssefsKaratzaferis.pdf

Koester D, Lunt B, & Stiefelhagen R. (2016). *Zebra Crossing Detection from Aerial Imagery Across Countries*. https://cvhci.anthropomatik.kit.edu/~dkoester/publications/koesterICCHP2016zebrax.pdf

Long, L., & Zeng, X. (2022a). *Beginning Deep Learning with TensorFlow* (Vol. 1). Apress. https://doi.org/10.1007/978-1-4842-7915-1

Long, L., & Zeng, X. (2022b). *Relationship of artiicial intelligence, machine learning, neural networks, and deep learning* (Vol. 1). Apress. https://doi.org/10.1007/978-1-4842-7915-1

Matsuyama, E. (2020). A Deep Learning Interpretable Model for Novel Coronavirus Disease (COVID-19) Screening with Chest CT Images. *Journal of Biomedical Science and Engineering*, *13*(07), 140–152. https://doi.org/10.4236/jbise.2020.137014

Mohan, S. (2020, December 26). *Keras Implementation of ResNet-50 (Residual Networks) Architecture from Scratch*. Machinelearningknowledge.Ai. https://machinelearningknowledge.ai/keras-implementation-of-resnet-50-architecture-from-scratch/#Quick_Concept_about_Transfer_Learning

NatraTex. (n.d.). *What do colourful surface markings mean on roads?* Natratex.Co.Uk. Retrieved February 6, 2023, from https://www.natratex.co.uk/knowledgehub/what-do-colourful-roads-mean/#:~:text=What%20does%20red%20road%20surfacing,certain%20area%20of%20the%20road

NBSHARE NOTEBOOKS. (n.d.). *Learn And Code Confusion Matrix With Python*. Nbshare.Io. Retrieved February 6, 2023, from https://www.nbshare.io/notebook/626706996/Learn-And-Code-Confusion-Matrix-With-Python/

Neelam, S. (2021, August 29). *YOLO for Object Detection, Architecture Explained! - Image*. Analytics Vidhya. https://medium.com/analytics-vidhya/understanding-yolo-and-implementing-yolov3-for-object-detection-5f1f748cc63a

PyLessons. (2020, July 15). *Understanding the mAP (mean Average Precision) Evaluation Metric for Object Detection*. PyLessons.Com.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016a). *Loss Function - Yolo*. http://pjreddie.com/yolo/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016b). *You Only Look Once: Unified, Real-Time Object Detection*. http://pjreddie.com/yolo/

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. https://pjreddie.com/yolo/.

Silva E, Sampaio F, da Silva L, Medeiros D, & Correia G. (2020). A method for embedding a computer vision application into a wearable device. *Microprocessors and Microsystems*, *76*, 103086. https://doi.org/10.1016/j.micpro.2020.103086

Turing. (2023). *Why Is Python Best Adapted to AI and Machine Learning?* Turing.Com. https://www.turing.com/kb/python-best-adapted-to-ai-and-machine-learning

Valentyn Sichkar. (2021). *Introduction into YOLO v3*. https://www.youtube.com/watch?v=vRqSO6RsptU

Verdhan, V. (2021a). *images: Vol. 5,7*. Apress. https://doi.org/10.1007/978-1-4842-6616-8

Verdhan, V. (2021b). Computer Vision Using Deep Learning. In *Computer Vision Using Deep Learning* (Vols. 1, 2, 7). Apress. https://doi.org/10.1007/978-1-4842-6616-8

Wang, H., & Jiao, K. (2021). Blind guidance system based on image recognition and convolutional neural network. *IOP Conference Series: Earth and Environmental Science*, *769*(4), 042043. https://doi.org/10.1088/1755-1315/769/4/042043

Wichert A, & Sa-Couto L. (2021). *Machine Learning - A Journey to Deep Learning - With Exercises and Answers* (Vol. 1). World Scientific Pub Co Inc. https://www.worldscientific.com/worldscibooks/10.1142/12201#t=aboutBook

Yin, X.-B., Tan, Z.-H., Yang, R., & Guo, X. (2017). Single crystalline SrTiO3 as memristive model system: From materials science to neurological and psychological functions. *Journal of Electroceramics*, *39*(1–4), 210–222. https://doi.org/10.1007/s10832-017-0083-0

Zeiler, M. D., & Fergus, R. (2013). *Visualizing and Understanding Convolutional Networks*. http://arxiv.org/abs/1311.2901

zhangphil. (2019, November 7). *pic.jpgOriginal image*. Blog.Csdn.Net. https://blog.csdn.net/zhangphil/article/details/102960822

Παράρτημα

A. Κώδικας εκπαίδευσης

Ο κώδικας που χρησιμοποιείται στην εκπαίδευση δημιουργήθηκε σύμφωνα με τον οδηγό του Alexey Bochkovskiy στο GitHub. Και για τις δύο πόλεις λαμβάνονται οι ίδιες εντολές με μόνη διαφορά τα δεδομένα. Παρακάτω παρατίθεται ο κώδικας :

```
# Commented out IPython magic to ensure Python compatibility.
%cat /etc/lsb-release

!apt-get update

!unzip '/content/drive/MyDrive/br_m/darknet.zip'

# Commented out IPython magic to ensure Python compatibility.
%cd /content/darknet

!sudo apt install dos2unix

!find . -type f -print0 | xargs -0 dos2unix

!chmod +x /content/darknet

# Commented out IPython magic to ensure Python compatibility.
%pwd

# Commented out IPython magic to ensure Python compatibility.
%cd /content/darknet

!make

!rm /content/darknet/backup -r

!ln -s /content/drive/MyDrive/br_weights/backup /content/darknet

# Commented out IPython magic to ensure Python compatibility.
%cd /content/darknet

!./darknet detector train cov_data/cov.data cov_yolov3.cfg darknet53.conv.74 -dont_show -map
```

Εικόνα π.1 Κώδικας εκπαίδευσης

B. Κώδικας ανίχνευσης – παραγωγής συντεταγμένων

Παρακάτω παρατίθενται οι κώδικες ανίχνευσης – παραγωγής συντεταγμένων για τις δύο πόλεις αλλά και για τις δοκιμαστικές εικόνες :

B1. Τμήμα χάρτη για τις πόλεις Μπρίστολ και Μάντσεστερ

```
import time

# get the start time
st = time.time()

import numpy as np
from patchify import patchify, unpatchify
from PIL import Image
import cv2
k=0
num=0
import os
#formal image's coordinations (meters)
#top left
xl=-290993.8219999999855645
yl=6702775.1185999996960163
#bottom right
xr=-290692.3745999999810010
yr=6702599.0838000001385808


import csv
from csv import DictWriter

# field names
fields = ['x', 'y']
# name of csv file
filename = "coord_man_2.csv"
# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(fields)

#path που είναι ο κώδικας και δημιουργούνται οι φάκελοι
p='D:/backup e-byte/Desktop/bristol_image_try1/split_merge/'
#φάκελος για κομμένη εικόνα: patches_m2
path='D:/backup e-byte/Desktop/bristol_image_try1/split_merge/patches_b' #save patches_b to p folder
os.mkdir(path)
path1='D:/backup e-byte/Desktop/bristol_image_try1/split_merge/only_det'
os.mkdir(path1)
path2='D:/backup e-byte/Desktop/bristol_image_try1/split_merge/detections'
os.mkdir(path2)
```

Παραπάνω φαίνεται η είσοδος των δεδομένων και των βιβλιοθηκών, καθώς και η δημιουργία των φακέλων και του αρχείου CSV που θα αποθηκευτούν στη συνέχεια τα δεδομένα.

Στη συνέχεια παρουσιάζεται ο χωρισμός της αρχικής εικόνας σε κομμάτια 416 x 416. Παρακάτω γίνεται η ανίχνευση σε κάθε κομμάτι της αρχικής εικόνας :

```
os.chdir('D:/backup e-byte/Desktop/bristol_image_try1/split_merge/patches_b/')
# input image
image = Image.open("D:/backup e-byte/Desktop/bristol_image_try1/images_bristolbig.jpg")
image = np.asarray(image)

# splitting the image into patches
image_height, image_width, channel_count = image.shape
h=image_height
w=image_width
patch_height, patch_width, step = 416, 416, 416
patch_shape = (patch_height, patch_width, channel_count)
patches = patchify(image, patch_shape, step=step)
print(patches.shape)

for i in range(patches.shape[0]):
    for j in range(patches.shape[1]):
        patch = patches[i, j, 0]
        patch = Image.fromarray(patch)
        num = i * patches.shape[1] + j
        patch.save(f"patch_{num}.jpg")
```

```
# processing each patch
output_patches = np.empty(patches.shape).astype(np.uint8)
for i in range(patches.shape[0]):
    for j in range(patches.shape[1]):
        patch = patches[i, j, 0]
        img_to_detect=patch
        img_height = img_to_detect.shape[0]
        img_width = img_to_detect.shape[1]

        # convert to blob to pass into model
        img_blob = cv2.dnn.blobFromImage(img_to_detect, 0.003922, (416, 416), swapRB=True, crop=False)
        #recommended by yolo authors, scale factor is 0.003922=1/255, width,height of blob is 320,320
        #accepted sizes are 320×320,416×416,608×608. More size means more accuracy but less speed

        # only single label
        class_labels = ["crosswalk"]

        #Declare only a single color
        class_colors = ["255,69,0"]
        class_colors = [np.array(every_color.split(",")).astype("int") for every_color in class_colors]
        class_colors = np.array(class_colors)
        class_colors = np.tile(class_colors,(1,1))

        # Loading the crosswalk custom model
        # input preprocessed blob into model and pass through the model
        # obtain the detection predictions by the model using forward() method
        yolo_model = cv2.dnn.readNetFromDarknet('D:/backup e-byte/Desktop/costum_yolo_m/model/cov_yolov3.cfg','D:/b

        # Get all layers from the yolo network
        # Loop and find the last layer (output layer) of the yolo network
        yolo_layers = yolo_model.getLayerNames()
        yolo_output_layer = [yolo_layers[yolo_layer - 1] for yolo_layer in yolo_model.getUnconnectedOutLayers()]

        # input preprocessed blob into model and pass through the model
        yolo_model.setInput(img_blob)
        # obtain the detection layers by forwarding through till the output layer
        obj_detection_layers = yolo_model.forward(yolo_output_layer)
```

Η εντολή που δεν φαίνεται ολόκληρη παρατίθεται εδώ και αφορά την φόρτωση του μοντέλου με τα βάρη που προέκυψαν από την εκπαίδευση :

yolo_model = cv2.dnn.readNetFromDarknet('D:/backup e-byte/Desktop/costum_yolo_m/model/cov_yolov3.cfg','D:/backup e-byte/Desktop/costum_yolo_m/model/cov_yolov3_best_b100.weights')

Στη συνέχεια προκύπτει ένα μοναδικό Bounding Box μέσα από την εφαρμογή κατωφλιών και της NMS του οποίου η πιθανότητα μαζί με την τάξη θα φαίνεται στην κονσόλα :

```
############## NMS Change 1 ###############
# initialization for non-max suppression (NMS)
# declare list for [class id], [box center, width & height[], [confidences]
class_ids_list = []
boxes_list = []
confidences_list = []
############## NMS Change 1 END ###########


# loop over each of the layer outputs
for object_detection_layer in obj_detection_layers:
    # loop over the detections
    for object_detection in object_detection_layer:

        # obj_detections[1 to 4] => will have the two center points, box width and box height
        # obj_detections[5] => will have scores for all objects within bounding box
        all_scores = object_detection[5:]
        predicted_class_id = np.argmax(all_scores)
        prediction_confidence = all_scores[predicted_class_id]

        # take only predictions with confidence more than 20%
        if prediction_confidence > 0.20:
            #get the predicted label
            predicted_class_label = class_labels[predicted_class_id]
            #obtain the bounding box co-oridnates for actual image from resized image size
            bounding_box = object_detection[0:4] * np.array([img_width, img_height, img_width, img_height])
            (box_center_x_pt, box_center_y_pt, box_width, box_height) = bounding_box.astype("int")
            start_x_pt = int(box_center_x_pt - (box_width / 2))
            start_y_pt = int(box_center_y_pt - (box_height / 2))


            ############## NMS Change 2 ###############
            #save class id, start x, y, width & height, confidences in a list for nms processing
            #make sure to pass confidence as float and width and height as integers
            class_ids_list.append(predicted_class_id)
            confidences_list.append(float(prediction_confidence))
            boxes_list.append([start_x_pt, start_y_pt, int(box_width), int(box_height)])
            ############## NMS Change 2 END ###########
```

```
max value ids = cv2.dnn.NMSBoxes(boxes list, confidences list, 0.5, 0.4)
```

```
# loop through the final set of detections remaining after NMS and draw bounding box and write text
for max_valueid in max_value_ids:
    max_class_id = max_valueid
    box = boxes_list[max_class_id]
    start_x_pt = box[0]
    start_y_pt = box[1]
    box_width = box[2]
    box_height = box[3]

    #get the predicted class id and label
    predicted_class_id = class_ids_list[max_class_id]
    predicted_class_label = class_labels[predicted_class_id]
    prediction_confidence = confidences_list[max_class_id]
```

```
end_x_pt = start_x_pt + box_width
end_y_pt = start_y_pt + box_height

#get a random mask color from the numpy array of colors
box_color = class_colors[predicted_class_id]

#convert the color numpy array as a list and apply to text and box
box_color = [int(c) for c in box_color]

# print the prediction in console
predicted_class_label = "{}: {:.2f}%".format(predicted_class_label, prediction_confidence * 100)
print("predicted object {}".format(predicted_class_label))

# draw rectangle and text in the image
cv2.rectangle(img_to_detect, (start_x_pt, start_y_pt), (end_x_pt, end_y_pt), box_color, 1)
cv2.putText(img_to_detect, predicted_class_label, (start_x_pt, start_y_pt+15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, box_color, 1)


im=Image.fromarray(img_to_detect)
os.chdir('D:/backup e-byte/Desktop/bristol_image_try1/split_merge/only_det/')
#im=Image.fromarray(img_to_detect)
im.save(f'{k}.jpg')
print(k ,'(i,j)= ',(i,j))
```

Στη συνέχεια υπολογίζονται οι συντεταγμένες για κάθε ανιχνεύσιμη εικόνα, αποθηκεύονται οι ανιχνεύσιμες και στη συνέχεια ενώνονται σε μία τελική εικόνα :

```
#sides (meters)
sx=abs(xl-xr)
#print(sx)
sy=abs(yl-yr)
#print(sy)

#step 416 pixels --> according to each side
px=416.00*sx/w
py=416.00*sy/h
#print(px)
#print(py)

#x,y of detected pictures. x,y of the center of the image. (px*i,j)
#xk= xl+(px*j)+px/2
#yk=yl-(py*i)-py/2

xm=xl+(px*j) #start_p of patch
ym=yl-(py*i)

x_s=abs((px*start_x_pt/416)) #start_p BB meters
y_s=abs((py*start_y_pt/416))

n1=abs(end_x_pt-start_x_pt)
n2=abs(end_y_pt-start_y_pt)

n1s=(px*n1)/416
n2s=(py*n2)/416

#wbb=px*box_width #w,h BB meters
#hbb=py*box_height

xk=xm+x_s+n1s/2 #coordinates of crosswalks
yk=ym-y_s-n2s/2

print(k, 'x=', xk, 'y=' , yk)

# data rows of csv file
fields = ['x', 'y']
dict = {'x': xk, 'y': yk}
```

```
            # writing to csv file

            with open('D:/backup e-byte/Desktop/bristol_image_try1/split_merge/coord_man_2.csv', 'a') as f_object:

                # Pass this file object to csv.writer()
                # and get a writer object
                dictwriter_object = DictWriter(f_object, fieldnames=fields)

                # Pass the list as an argument into
                # the writerow()
                dictwriter_object.writerow(dict)

                # Close the file object
                f_object.close()



            k=k+1


        im=Image.fromarray(img_to_detect)
        os.chdir('D:/backup e-byte/Desktop/bristol_image_try1/split_merge/detections/')
        num=num+1
        im.save(f'{num}.jpg')
        os.chdir('D:/backup e-byte/Desktop/bristol_image_try1/split_merge/')
        #cv2.imshow("Detection Output", img_to_detect)

        output_patch = img_to_detect #process(patch)  # process the patch
        output_patches[i, j, 0] = output_patch

# merging back patches
output_height = image_height - (image_height - patch_height) % step
output_width = image_width - (image_width - patch_width) % step
output_shape = (output_height, output_width, channel_count)
output_image = unpatchify(output_patches, output_shape)
output_image = Image.fromarray(output_image)
output_image.save("output.jpg")

# get the end time
et = time.time()

# get the execution time
elapsed_time = et - st
print('Execution time:', elapsed_time, 'seconds')
```

Αντίστοιχα ο κώδικας για το Μάντσεστερ είναι ο ίδιος με το μόνο που αλλάζει είναι τα δεδομένα.

B2. Δοκιμαστικές εικόνες

Ο κώδικας για την ανίχνευση σε μία δοκιμαστική εικόνα παρατίθεται παρακάτω. Η διαφορά με τον πάνω κώδικα έγκειται στο ότι μετά την εισαγωγή των δεδομένων γίνεται κατευθείαν η ανίχνευση και μετά εμφανίζεται στην κονσόλα με την πιθανότητα και την τάξη :

```
import numpy as  np
import cv2
import glob
from PIL import Image
import os
num=0
```

Να συμπληρωθεί πως η εντολή που δεν φαίνεται παρακάτω είναι η ίδια όπως και παραπάνω :

```python
for img in glob.glob('D:/backup e-byte/compare_weights.jpg'):
    img_to_detect = cv2.imread(img)
    #img_to_detect = cv2.imread('C:/Users/valen/Desktop/the_rest/costum Yolo model2/code/images/testingb/park2.jpg')
    img_height = img_to_detect.shape[0]
    img_width = img_to_detect.shape[1]

    # convert to blob to pass into model
    img_blob = cv2.dnn.blobFromImage(img_to_detect, 0.003922, (416, 416), swapRB=True, crop=False)
    #recommended by yolo authors, scale factor is 0.003922=1/255, width,height of blob is 320,320
    #accepted sizes are 320×320,416×416,608×608. More size means more accuracy but less speed

    # only single label
    class_labels = ["crosswalk"]

    #Declare only a single color
    class_colors = ["255,69,0"]
    class_colors = [np.array(every_color.split(",")).astype("int") for every_color in class_colors]
    class_colors = np.array(class_colors)
    class_colors = np.tile(class_colors,(1,1))

    # Loading the crosswalk custom model
    # input preprocessed blob into model and pass through the model
    # obtain the detection predictions by the model using forward() method
    yolo_model = cv2.dnn.readNetFromDarknet('D:/backup e-byte/Desktop/costum_yolo_m/model/cov_yolov3.cfg','D:/backup

    # Get all layers from the yolo network
    # Loop and find the last layer (output layer) of the yolo network
    yolo_layers = yolo_model.getLayerNames()
    yolo_output_layer = [yolo_layers[yolo_layer - 1] for yolo_layer in yolo_model.getUnconnectedOutLayers()]

    # input preprocessed blob into model and pass through the model
    yolo_model.setInput(img_blob)
    # obtain the detection layers by forwarding through till the output layer
    obj_detection_layers = yolo_model.forward(yolo_output_layer)


    ############## NMS Change 1 ###############
    # initialization for non-max suppression (NMS)
    # declare list for [class id], [box center, width & height[], [confidences]
    class_ids_list = []
    boxes_list = []
    confidences_list = []
    ############## NMS Change 1 END ###########

    # loop over each of the layer outputs
    for object_detection_layer in obj_detection_layers:
        # loop over the detections
        for object_detection in object_detection_layer:

            # obj_detections[1 to 4] => will have the two center points, box width and box height
            # obj_detections[5] => will have scores for all objects within bounding box
            all_scores = object_detection[5:]
            predicted_class_id = np.argmax(all_scores)
            prediction_confidence = all_scores[predicted_class_id]

            # take only predictions with confidence more than 20%
            if prediction_confidence > 0.20:
                #get the predicted label
                predicted_class_label = class_labels[predicted_class_id]
                #obtain the bounding box co-oridnates for actual image from resized image size
                bounding_box = object_detection[0:4] * np.array([img_width, img_height, img_width, img_height])
                (box_center_x_pt, box_center_y_pt, box_width, box_height) = bounding_box.astype("int")
                start_x_pt = int(box_center_x_pt - (box_width / 2))
                start_y_pt = int(box_center_y_pt - (box_height / 2))


                ############## NMS Change 2 ###############
                #save class id, start x, y, width & height, confidences in a list for nms processing
                #make sure to pass confidence as float and width and height as integers
                class_ids_list.append(predicted_class_id)
                confidences_list.append(float(prediction_confidence))
                boxes_list.append([start_x_pt, start_y_pt, int(box_width), int(box_height)])
                ############## NMS Change 2 END ###########
```

```
max_value_ids = cv2.dnn.NMSBoxes(boxes_list, confidences_list, 0.5, 0.4)

# loop through the final set of detections remaining after NMS and draw bounding box and write text
for max_valueid in max_value_ids:
    max_class_id = max_valueid
    box = boxes_list[max_class_id]
    start_x_pt = box[0]
    start_y_pt = box[1]
    box_width = box[2]
    box_height = box[3]

    #get the predicted class id and label
    predicted_class_id = class_ids_list[max_class_id]
    predicted_class_label = class_labels[predicted_class_id]
    prediction_confidence = confidences_list[max_class_id]
    ############## NMS Change 3 END ###########

    end_x_pt = start_x_pt + box_width
    end_y_pt = start_y_pt + box_height

    #get a random mask color from the numpy array of colors
    box_color = class_colors[predicted_class_id]

    #convert the color numpy array as a list and apply to text and box
    box_color = [int(c) for c in box_color]

    # print the prediction in console
    predicted_class_label = "{}: {:.2f}%".format(predicted_class_label, prediction_confidence * 100)
    print("predicted object {}".format(predicted_class_label))

    # draw rectangle and text in the image
    cv2.rectangle(img_to_detect, (start_x_pt, start_y_pt), (end_x_pt, end_y_pt), box_color, 1)
    cv2.putText(img_to_detect, predicted_class_label, (start_x_pt, start_y_pt+15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, box_color, 1)


im=Image.fromarray(img_to_detect)


cv2.imshow("Detection Output", img_to_detect)
```

Γ. Ταξινόμηση ZF – Net και Resnet

Γ1. ZF – Net

Παρακάτω παρατίθεται ο κώδικας ταξινόμησης διαβάσεων διαγράμμισης ζέβρας του ZF – Net :

```
#run 6: epochs=15 batch_size=8 LeakyRelu regularizations for overfitting
import os
import numpy as np
#from PIL import Image
import cv2
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils.vis_utils import plot_model

os.chdir('/content/drive/MyDrive/bristol_1')

#define model  #run2 batch_size=12
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils.vis_utils import plot_model
from keras.regularizers import l2
from keras.regularizers import l1

num_classes=1
batch_size=8
epochs=15

from keras.layers import LeakyReLU
model = Sequential()
model.add(keras.layers.LeakyReLU(alpha=0.05))
# Έβαλα 8 feature maps για να γίνει το δίκτυο πιο ελαφρύ και όρισα να μην αλλάζει το φίλτρο τη διάσταση της εικόνας
model.add(Conv2D(8, kernel_size=(3, 3), activation=keras.layers.LeakyReLU(alpha=0.05), input_shape=(224,224,3), padding='same')) #output 224*224*8
model.add(MaxPooling2D(pool_size=(2, 2))) #output 112*112*8
model.add(Conv2D(8, (3, 3), activation=keras.layers.LeakyReLU(alpha=0.05), padding='same'))#output 112*112*8
model.add(MaxPooling2D(pool_size=(2, 2))) #output 64*64*8
model.add(Conv2D(8, (3, 3), activation=keras.layers.LeakyReLU(alpha=0.05), padding='same'))#output 64x64*8
model.add(MaxPooling2D(pool_size=(2, 2))) #output 32*32*8
model.add(Flatten()) #output 32*32*8 = 8192
model.add(Dense(128, kernel_regularizer=l2(0.01), bias_regularizer=l1(0.01))) #activation=keras.layers.LeakyReLU(alpha=0.05)
model.add(Dropout(0.5))# Το dropout συστήνεται να χρησιμοποιείται κυρίως στα fully connected και όχι στα convolutional layers
model.add(Dense(num_classes, activation = 'sigmoid'))

optimizer=tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(loss='binary_crossentropy',optimizer=optimizer,metrics=['accuracy']) # επειδή έχεις μόνο δύο κλάσεις χρησιμοποιείς το binary crossentropy.
```

```
x_train = np.load("x_train.npy")
y_train = np.load("y_train.npy")
x_test = np.load("x_test.npy")
y_test = np.load("y_test.npy")

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))

model.summary()

model.save_weights("weights_run6_3.h5")
y_pred_run6_3=model.predict(x_test)
y_pred_run6_3=np.rint(y_pred_run6_3)   δίνει τιμή 0 ή 1
#y_pred_run
np.save('y_pred_run6_3.npy',y_pred_run6_3)
from sklearn.metrics import confusion_matrix
cm_run6_3 = confusion_matrix(y_test, y_pred_run6_3)
cm_run6_3

np.save('cm_run6_3.npy',cm_run6_3)
accuracy2=(182+197)/(182+197+8+20)#######
accuracy2

count=0
count2=0
count3=0
count4=0
for i in range(len(y_pred_run6_3)):
  if y_pred_run6_3[i]==1 and y_test[i]==1:
    count=count+1
  if y_pred_run6_3[i]==1 and y_test[i]==0:
    count2=count2+1
  if y_pred_run6_3[i]==0 and y_test[i]==0:
    count3=count3+1
  if y_pred_run6_3[i]==0 and y_test[i]==1:
    count4=count4+1
count #count tp

count2 #count2 fp

count3 #count3 tn

count4 #count4 fn
```

```
matrix1=np.zeros((count,1))
matrix2=np.zeros((count2,1))
matrix3=np.zeros((count3,1))
countnew=-1
countnew1=-1
countnew2=-1
countnew3=-1
for i in range(len(y_pred_run6_3)):
  if y_pred_run6_3[i]==0 and y_test[i]==1:
    countnew=countnew+1
    matrix[countnew]=i #count4 fn
  if y_pred_run6_3[i]==1 and y_test[i]==1:
    countnew1=countnew1+1
    matrix1[countnew1]=i #count tp
  if y_pred_run6_3[i]==1 and y_test[i]==0:
    countnew2=countnew2+1
    matrix2[countnew2]=i #count2 fp
  if y_pred_run6_3[i]==0 and y_test[i]==0:
    countnew3=countnew3+1
    matrix3[countnew3]=i  #count3 tn
matrix

matrix2

recall=count/(count+count4)
specif=count3/(count3+count2)  #% other are classified as other
precition=count/(count+count2) #%samples that are classified as cross are indeed cross
recall

specif

precition

np.save('matrix_c4_run6_3.npy',matrix)
np.save('matrix_c2.npy_run6_3',matrix2)

# evaluate model
    _, acc = model.evaluate(x_test, y_test, verbose=0)
    print('> %.3f' % (acc * 100.0))
```

Γ2. Resnet -50 - Resnet απλοποιημένο

Παρακάτω παρατίθεται ο κώδικας ταξινόμησης διαβάσεων διαγράμμισης ζέβρας του Resnet -50 και Resnet απλοποιημένο :

```
import cv2
import numpy as np
import os
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import tensorflow.keras.backend as K
import keras
from keras.models import Sequential, Model,load_model
from tensorflow.keras.optimizers import SGD
from keras.callbacks import EarlyStopping,ModelCheckpoint
from google.colab.patches import cv2_imshow
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D,MaxPool2D
from keras.preprocessing import image
from keras.initializers import glorot_uniform
from tensorflow.python.keras.backend import get_session

import tensorflow as tf
print(tf.__version__)
print(tf.keras.__version__)

os.chdir('/content/drive/MyDrive/bristol_1')

train_path="/content/drive/MyDrive/bristol_1/br2/train"
test_path="/content/drive/MyDrive/bristol_1/br2/test"
class_names=os.listdir(train_path)
class_names_test=os.listdir(test_path)

print(class_names)
print(class_names_test)

#Sample datasets images
image_cross=cv2.imread('/content/drive/MyDrive/bristol_1/br2/test/cross/r14_26.tif')
cv2_imshow(image_cross)
image_other=cv2.imread('/content/drive/MyDrive/bristol_1/br2/test/other/r14_A3.tif')
cv2_imshow(image_other)

x_train = np.load("x_train.npy")
y_train = np.load("y_train.npy")
x_test = np.load("x_test.npy")
y_test = np.load("y_test.npy")
```

```python
#identity block
def identity_block(X, f, filters, stage, block):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'
    F1, F2, F3 = filters

    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)
    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X = Add()([X, X_shortcut])# SKIP Connection
    X = Activation('relu')(X)

    return X
#convonutional block
def convolutional_block(X, f, filters, stage, block, s=2):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    F1, F2, F3 = filters

    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X_shortcut = Conv2D(filters=F3, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '1', kernel_initializer=glorot_uniform(seed=0))(X_shortcut)
    X_shortcut = BatchNormalization(axis=3, name=bn_name_base + '1')(X_shortcut)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
```

Παρακάτω φαίνεται η δημιουργία του μοντέλου λαμβάνοντας υπόψη τα identity και convolutional blocks που δημιουργήθηκαν παραπάνω. Μάλιστα τα σχόλια (#) αποτελούν το τμήμα που αφαιρείται από το μοντέλο για τη δημιουργία του απλοποιημένου μαζί με τις αλλαγές στο stride :

```python
#resnet50
def ResNet50(input_shape=(224, 224, 3)):

    X_input = Input(input_shape)

    X = ZeroPadding2D((3, 3))(X_input)

    X = Conv2D(16, (7, 7), strides=(2, 2), name='conv1', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name='bn_conv1')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X)

    X = convolutional_block(X, f=3, filters=[16, 16, 64], stage=2, block='a', s=2)
    X = identity_block(X, 3, [16, 16, 64], stage=2, block='b')
    #X = identity_block(X, 3, [16, 16, 256], stage=2, block='c')


    X = convolutional_block(X, f=3, filters=[32, 32, 128], stage=3, block='a', s=2)
    X = identity_block(X, 3, [32, 32, 128], stage=3, block='b')
   # X = identity_block(X, 3, [128, 128, 512], stage=3, block='c')
    #X = identity_block(X, 3, [128, 128, 512], stage=3, block='d')

   # X = convolutional_block(X, f=3, filters=[256, 256, 1024], stage=4, block='a', s=2)
   # X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')
    #X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')
    #X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')
    #X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')
   # X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')#

    X = X = convolutional_block(X, f=3, filters=[128, 128, 512], stage=5, block='a', s=2)
    X = identity_block(X, 3, [128, 128, 512], stage=5, block='b')
   # X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')

    X = AveragePooling2D(pool_size=(2, 2), padding='same')(X)

    model = Model(inputs=X_input, outputs=X, name='ResNet50')

    return model
base_model = ResNet50(input_shape=(224, 224, 3))

#add 3 dense layers
headModel = base_model.output
headModel = Flatten()(headModel)
headModel=Dense(64, activation='relu', name='fc1',kernel_initializer=glorot_uniform(seed=0))(headModel)
headModel=Dense(32, activation='relu', name='fc2',kernel_initializer=glorot_uniform(seed=0))(headModel)
headModel = Dense( 1,activation='sigmoid', name='fc3',kernel_initializer=glorot_uniform(seed=0))(headModel)

#model which takes input from the last layer of the input layer and outputs from the last layer from the head model
model = Model(inputs=base_model.input, outputs=headModel)
model.summary()


for layer in model.layers:
    print(layer, layer.trainable)

from tensorflow.keras.optimizers import Adam
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])

es=EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=10)

#check_point
mc = ModelCheckpoint('/content/drive/MyDrive/bristol_1/best_model34.h5', monitor='val_accuracy', mode='max', save_best_only=True, verbose=1)

#training
epochs=100
batch_size=8
H=model.fit(x_train, y_train,
            batch_size=batch_size,
            epochs=epochs,
            verbose=1,
            validation_data=(x_test, y_test),
            callbacks=[mc,es])
model.evaluate(x_test, y_test, verbose=0)
```

```
from keras.models import model_from_json

#save model for future use
model34_json = model.to_json()
with open("/content/drive/MyDrive/bristol_1/model34.json","w") as json_file:
  json_file.write(model34_json)

#def predict_(image_path):
    #Load the Model from Json File
json_file = open('/content/drive/MyDrive/bristol_1/model34.json', 'r')
model34_json_c = json_file.read()
json_file.close()
model_c = model_from_json(model34_json_c)
# Load the weights
model_c.load_weights("/content/drive/MyDrive/bristol_1/best_model34.h5")
# Compile the model
opt = SGD(lr=1e-4, momentum=0.9)
model_c.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])

y_pred_s4=model_c.predict(x_test)
y_pred_s4=np.rint(y_pred_s4)    #δίνει τιμή 0 ή 1
#y_pred_run
np.save('y_pred_s4.npy',y_pred_s4)
from sklearn.metrics import confusion_matrix
cm_s4 = confusion_matrix(y_test, y_pred_s4)
cm_s4

np.save('cm_s4.npy',cm_s4)
accuracy2=(205+201)/(205+0+1+201)#######
accuracy2

count=0
count2=0
count3=0
count4=0
for i in range(len(y_pred_s4)):
  if y_pred_s4[i]==1 and y_test[i]==1:
    count=count+1
  if y_pred_s4[i]==1 and y_test[i]==0:
    count2=count2+1
  if y_pred_s4[i]==0 and y_test[i]==0:
    count3=count3+1
  if y_pred_s4[i]==0 and y_test[i]==1:
    count4=count4+1
count #count tp
```

```
count2 #count2 fp

count3 #count3 tn

count4 #count4 fn

matrix=np.zeros((count4,1))
matrix1=np.zeros((count,1))
matrix2=np.zeros((count2,1))
matrix3=np.zeros((count3,1))
countnew=-1
countnew1=-1
countnew2=-1
countnew3=-1
for i in range(len(y_pred_s4)):
  if y_pred_s4[i]==0 and y_test[i]==1:
    countnew=countnew+1
    matrix[countnew]=i #count4 fn
  if y_pred_s4[i]==1 and y_test[i]==1:
    countnew1=countnew1+1
    matrix1[countnew1]=i #count tp
  if y_pred_s4[i]==1 and y_test[i]==0:
    countnew2=countnew2+1
    matrix2[countnew2]=i #count2 fp
  if y_pred_s4[i]==0 and y_test[i]==0:
    countnew3=countnew3+1
    matrix3[countnew3]=i  #count3 tn
matrix

matrix2

recall=count/(count+count4)
specif=count3/(count3+count2)  #% other are classified as other
precition=count/(count+count2) #%samples that are classified as cross are indeed cross
recall

specif

precition

np.save('matrix_c4_s4.npy',matrix)
np.save('matrix_c2.npy_s4',matrix2)
```