ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

Πτυχιακή/ Διπλωματική Εργασία

Τίτλος εργασίας

Εφαρμογές στην βαθιά μηχανική μάθηση

Συγγραφέας/είς

Καλαμπόκας Νικόλαος

ΑΜ: 47090

Επιβλέπων/ούσα:

Νικολάου Γρηγόριος

Αθήνα, Ιούνιος 2021

UNIVERSITY OF WEST

ATTICA SCHOOL

DEPARTMENT INDUSTRIAL DESIGN AND PRODUCTION ENGINEERS

Diploma Thesis

Title

Applications on machine deep learning

Student name and

surname:

Kalampokas Nikolaos

Registration Number:

47090

Supervisor name and

surname:

Nikolaou Grigorios

Athens, June 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ

ΑΤΤΙΚΗΣ ΣΧΟΛΗ

ΤΜΗΜΑ

Τίτλος εργασίας

Εφαρμογές στην βαθιά μηχανική μάθηση

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η πτυχιακή/διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Γ. Νικολάου		GRIGORIOS NIKOLAOU Digitally signed by GRIGORIOS NIKOLAOU Date: 2021.11.12 19:16:41 +02'00'
2	Σ. Βασιλειάδου		Soultana Vasileiadou Digitally signed by Soultana Vasileiadou Date: 2021.11.12 08:32:04 +02'00'
3	Χ. Δρόσος		Christos Drosos Digitally signed by Christos Drosos Date: 2021.11.11 09:03:37 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

0/η κάτωθι υπογεγραμμένος/η Καλαμπόκας Νικόλαος του.
με αριθμό μητρώου 47090 φοιτητής/τρια του Πανεπιστημίου Δυτικής Αττικής της
Σχολής Μηχανικών του Τμήματος Βιομηχανικής Σχεδίασης και Παραγωγής δηλώνω υπεύθυνα ότι:

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

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

0/Η Δηλών/ούσα

Ευχαριστίες

Με την ολοκλήρωση της προπτυχιακής διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω τον καθηγητή μου Νικολάου Γρηγόριο που μου εμπιστεύτηκε το συγκεκριμένο θέμα και για την βοήθεια και τις γνώσεις που μου μετέδωσε.

Επίσης θα ήθελα να ευχαριστήσω και να την αφιερώσω στον αδερφό μου, Θεοφάνη Καλαμπόκα, για την καθοδήγησή του, την συμπαράστασή του και τη συνεχή του υποστήριξη. Επίσης τον ευχαριστώ θερμά που μου έδειξε τον κόσμο της βαθιάς μηχανικής μάθησης και για όλες τις γνώσεις που μου έχει προσφέρει και έχει μοιραστεί μαζί μου μέσα από τις εποικοδομητικές και μορφωτικές συζητήσεις μας. Επίσης θα ήθελα να αναφέρω πόσο ευγνώμον και πόσο περήφανος νιώθω που έχω την τύχη να τον ως στήριγμά μου σε όλο αυτό.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου για όλη τη στήριξη, τη συμπαράσταση και την κατανόηση τους, καθ' όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η Μηχανική Μάθηση είναι υποπεδίο της επιστήμης των υπολογιστών, που αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Το 1959, ο Άρθουρ Σάμουελ ορίζει την μηχανική μάθηση ως Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί. Η μηχανική μάθηση διερευνά τη μελέτη αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Η παρούσα διπλωματική εργασία με τίτλο «Εφαρμογές στην Μηχανική Μάθηση» αναφέρεται στη συγκριτική αξιολόγηση της επίδοσης μοντέλων αναγνώρισης αντικειμένων και συγκεκριμένα της επιβλεπόμενης μάθησης για την αναγνώριση οχημάτων, ανθρώπων και σημάτων κατά τη διάρκεια της οδήγησης. Τα μοντέλα που χρησιμοποιήθηκαν είναι τα SSD (Single Shot Detection), YOLO (You Only Look Once) και Mask-RCNN. Γίνεται παρουσίαση των μοντέλων που επιλέχθηκαν και των αποτελεσμάτων εφαρμογής των αλγορίθμων σε ένα σύνολο δεδομένων. Η επεξεργασία πραγματοποιήθηκε με τη χρήση της γλώσσας προγραμματισμού Python.

Λέξεις κλειδιά: βαθιά μηχανική μάθηση, μοντέλα αναγνώρισης αντικειμένων, SSD, YOLO, Mask-RCNN, αναγνώριση αντικειμένων.

ABSTRACT

Machine Learning is a subfield of computer science, developed by the study of pattern recognition and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined Machine Learning as a field of study that gives computers the ability to learn without being explicitly programmed. Machine Learning explores the study of algorithms that can learn from data and make predictions about it. Such algorithms work by constructing models from experimental data in order to make prediction based on data or to make decisions that are expressed as the result.

The present dissertation entitled “Application in Machine Learning” refers to the comparative evaluation of the performance of abject recognition models and in particular of supervised learning for the recognition of vehicles, pedestrians and signs during driving. The models used are SSD (Single Shot Detection), YOLO (You Only Look Once) and Mask-RCN. The selected models and the results of the application of the algorithms in a dataset are presented in the following sections. Editing was done using the Python programming language.

Keywords: Deep Machine Learning, Object recognition models, SSD, YOLO, Mask-RCNN, object detection.

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή

1.1: Εισαγωγή

Στην σύγχρονη κοινωνία όλα περιστρέφονται γύρω από την γνώση και την απόκτησή της. Η γνώση μπορεί να προέλθει από διάφορες πηγές, όμως σε κάθε περίπτωση, ακόμα και στην εμπειρική της διάσταση, αποτελείται από ένα σύνολο επεξεργασμένων δεδομένων. Τα δεδομένα αυτά μπορεί να υπάρχουν γύρω μας έτοιμα προς συλλογή, όμως η σωστή ανάλυση και διερμηνεία αυτών είναι που δημιουργεί την πληροφορία που θα μετατραπεί σε τελική γνώση.

Με την ραγδαία εξέλιξη της τεχνολογίας και την ολοένα και μεγαλύτερη χρήση πληροφοριακών συσκευών στην καθημερινότητα μας, βρισκόμαστε αντιμέτωποι με ένα, τεραστίων διαστάσεων, εν δυνάμει κινητήρα γνώσης. Γνώση δεν είναι μόνο η πληροφορία που υπάρχει ήδη στα ηλεκτρονικά βιβλία, ούτε τα εκατομμύρια διαθέσιμα προς ανάγνωση αποτελέσματα οποιασδήποτε μηχανής αναζήτησης στο διαδίκτυο.

Η εξόρυξη γνώσης μπορεί να γίνει εφικτή με την ανάλυση οποιουδήποτε πακέτου δεδομένων, αν ακολουθηθεί μια σωστή και μελετημένη διαδικασία, προσαρμοσμένη στο εκάστοτε πεδίο έρευνας. Τέτοια δεδομένα μπορεί να προέρχονται από τεχνητά και μη μέσα, και να αφορούν τόσο τεχνολογικά όσο και φυσικά περιβάλλοντα εφαρμογής.

1.2: Πρόβλημα

Λίγο πολύ όλοι μας έχουμε δει και στους περισσότερους που οδηγούν, αν όχι σε όλους, έχει τύχει να το πάθουν. Πολλές φορές για διάφορους λόγους διαφορετικούς για την κάθε περίπτωση παρατηρείται έλλειψη περιφερειακής όρασης και αντίληψης από τα άτομα που οδηγούν με αποτέλεσμα να μην προλαβαίνουν έγκαιρα να αντιληφθούν κάποιο άλλο όχημα, ένα πεζό, ένα ποδηλάτη ή και ένα φωτεινό σηματοδότη που θα υπάρχει στο δρόμο μας με αποτέλεσμα να συμβαίνουν πολλά ατυχήματα και δυστυχήματα καθημερινά.

1.3: Δομή διπλωματικής εργασίας

Στα πλαίσια της διπλωματικής εργασίας μελετήθηκαν διάφορες διεθνείς δημοσιεύσεις που αφορούν την αναγνώριση αντικειμένων στα πλαίσια της οδήγησης με τη χρήση τον αντίστοιχων μοντέλων. Συγκεκριμένα, έγινε η μελέτη για την αναγνώριση οχημάτων ΙΧ, φορτηγών, πεζών, ποδηλατών και για φωτεινούς σηματοδότες, με τη χρήση του SSD (Single Shot Detection), YOLO (You Only Look Once) και Mask-RCNN μοντέλων.

Στο κεφάλαιο 2 γίνεται βιβλιογραφική ανασκόπηση σχετικά με την αναγνώριση αντικειμένων. Αρχικά γίνεται μια επεξήγηση τι εννοούμε χρησιμοποιώντας τον όρο αυτό. Στη συνέχεια γίνεται μια περιγραφή σχετικά με το πως δουλεύουν τα μοντέλα τα οποία χρησιμοποιήσαμε για να πάρουμε το επιθυμητό αποτέλεσμα.

Στο κεφάλαιο 3 παρουσιάζεται η μεθοδολογία, η οποία αναπτύχθηκε στην παρούσα μεταπτυχιακή διπλωματική εργασία. Παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση τη εργασίας καθώς και αποσπάσματα από τους κώδικες που βοήθησαν ώστε να βελτιωθούν τα τελικά αποτελέσματα.

Στο κεφάλαιο 4 αναφέρονται τα αποτελέσματα από την εφαρμογή των αλγορίθμων που αναλύθηκαν, επιλέχτηκαν και εφαρμόστηκαν βάση των χαρακτηριστικών τους όπως παρουσιάστηκαν στο κεφάλαιο 2. Πιο συγκεκριμένα γίνεται παρουσίαση των αποτελεσμάτων που είχαμε με τη χρήση του SSD μοντέλου, του YOLO μοντέλου και του Mask-RCNN. Τέλος γίνεται μια συγκριτική αξιολόγηση των αλγορίθμων στο σύνολο δεδομένων που έχουμε.

Στο κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα. Επίσης, προτείνονται διάφοροι νέοι τρόποι για μελλοντική μελέτη στο πρόβλημα της αναγνώρισης αντικειμένων.

Στο κεφάλαιο 6 παρουσιάζονται πηγές που χρησιμοποιήθηκαν για τη διεκπεραίωση της διπλωματικής εργασίας.

1.4: Συνεισφορά

Τα ατυχήματα που γίνονται στο δρόμο κατά ένα μεγάλο ποσοστό έχουν θανάσιμη κατάληξη ή στην καλύτερη των περιπτώσεων τα άτομα που εμπλέκονται την γλιτώνουν με ένα ποσοστό αναπηρίας. Η συγκεκριμένη προπτυχιακή διπλωματική εργασία, προσπαθεί να δημιουργήσει ένα αποτελεσματικό και γρήγορο μοντέλο μηχανικής μάθησης. Το έργο που θα επιτελεί το συγκεκριμένο μοντέλο είναι στο να συμβάλει στην μείωση τον δυστυχημάτων που συμβαίνουν στο δρόμο κατά την διάρκεια της οδήγησης.

Κεφάλαιο 2: Βιβλιογραφική Επισκόπηση

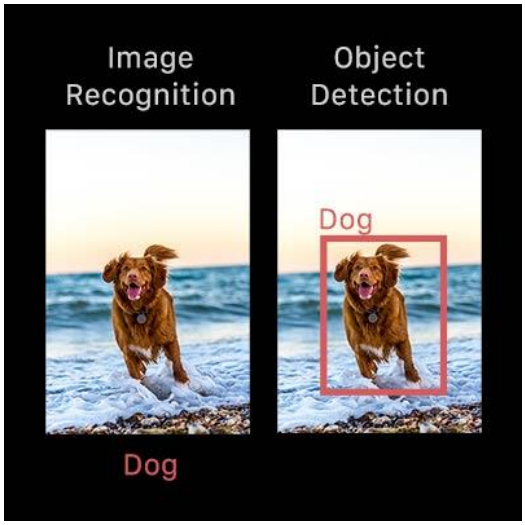
Στο κεφάλαιο που ακολουθεί γίνεται μια βιβλιογραφική ανασκόπηση της διπλωματικής εργασίας. Αρχικά γίνεται μια επεξήγηση για το τι είναι το Object detection και το πως δουλεύει. Στη συνέχεια αναλύεται ο τρόπος με τον οποίο δουλεύουν τα μοντέλα που επιλέχθηκαν.

2.1: Object detection (Αναγνώριση αντικειμένων)

Η αναγνώριση αντικειμένων είναι μια τεχνική μηχανικής όρασης η οποία εντοπίζει και αναγνωρίζει αντικείμενα σε μια εικόνα ή σε κάποιο βίντεο. Συγκεκριμένα, η μέθοδος αυτή σχεδιάζει περιγράμματα γύρω από τα αντικείμενα που του ζητάμε να εντοπίσει σε μία εικόνα ή ακόμα έχουμε την δυνατότητα να δούμε ένα σημείο ενδιαφέροντος το πως κινείται σε κάποιο βίντεο.

Η αναγνώριση αντικειμένων πολλές φορές παρομοιάζετε με την αναγνώριση εικόνας, οπότε πριν προχωρήσουμε παρακάτω, είναι σημαντικό να μπορούμε να ξεχωρίσουμε αυτές τις δύο έννοιες.

Η αναγνώριση εικόνας αποδίδει μια ταμπέλα σε μια εικόνα. Για παράδειγμα η εικόνα ενός σκύλου έχει ταμπέλα «σκύλος». Μια εικόνα που περιέχει δύο σκυλιά πάλι έχει ταμπέλα «σκύλος». Στην αναγνώριση αντικειμένων, απ’ την άλλη, σχεδιάζετε ένα περίγραμμα γύρω από κάθε σκύλο και το περίγραμμα αυτό έχει ταμπέλα «σκύλος» . Το μοντέλο εντοπίζει που βρίσκετε το κάθε αντικείμενο και τι ταμπέλα θα πρέπει να βάλει για το καθένα. Με αυτόν τον τρόπο, η αναγνώριση αντικειμένων παρέχει περισσότερες πληροφορίες για μια εικόνα.



Εικόνα 1: Διαφοράς αναγνώρισης αντικειμένων με αναγνώριση εικόνας

2.1.1: Τρόποι και τύποι αναγνώρισης αντικειμένων

Γενικά, η αναγνώριση αντικειμένων μπορεί να χωριστεί σε δύο προσεγγίσεις αυτή της μηχανικής μάθησης και αυτή της βαθιάς μηχανικής μάθησης.

Στις προσεγγίσεις που είναι βασισμένες στη μηχανική μάθηση, οι τεχνικές μηχανικής όρασης χρησιμοποιούνται για να δούμε διάφορα χαρακτηριστικά μιας εικόνας, όπως είναι το ιστόγραμμα χρωμάτων, που μας δίνει πληροφορίες σχετικά με το ποσοστό υπεροχής των τριών βασικών χρωμάτων RGB (Κόκκινο, Πράσινο, Μπλε), για κάποιο συγκεκριμένο σημείο, για να αναγνωρίσουμε μια ομάδα εικονοστοιχείων (pixels) τα οποία μπορεί να ανήκουν σε κάποιο αντικείμενο ενδιαφέροντος. Αυτά τα χαρακτηριστικά μετά τροφοδοτούνται σε ένα μοντέλο παλινδρόμησης το οποίο προβλέπει την τοποθεσία του αντικειμένου μαζί με την ταμπέλα του.

Από την άλλη, η προσέγγιση της βαθιάς μηχανικής μάθησης χρησιμοποιεί convolutional neural networks (CNNs) (συνελικτικά νευρικά δίκτυα) για να πραγματοποιήσουν από την αρχή μέχρι το τέλος, μη επιβλεπόμενη αναγνώριση αντικειμένων, στην οποία τα χαρακτηριστικά δεν χρειάζεται να οριστούν και να εξαχθούν ξεχωριστά.

2.1.2: Η σημαντικότητα της αναγνώρισης αντικειμένων

Η αναγνώριση αντικειμένων είναι απόλυτα συνδεδεμένη με άλλες παρόμοιες τεχνικές μηχανικής όρασης όπως είναι η αναγνώριση εικόνας και η περικοπή εικόνας, με τρόπο ο οποίος μας βοηθάει να καταλάβουμε και να αναλύσουμε σκηνές σε εικόνες ή στιγμιότυπα.

Αλλά υπάρχουν σημαντικές διαφορές. Η αναγνώριση εικόνας εξάγει μια κατηγοριοποιημένη ταμπέλα για ένα αντικείμενο, και η περικοπή εικόνας δημιουργεί μια εικονοστοιχείων-επιπέδου αντίληψη για τα στοιχεία μιας σκηνής. Αυτό που ξεχωρίζει την αναγνώριση αντικειμένων από αυτές τις δύο μεθόδους είναι η ικανότητα το εντοπίζει αντικείμενα μέσα σε κάποιο στιγμιότυπο ή εικόνα. Αυτό στην πορεία μας επιτρέπει να μετρήσουμε και τα παρακολουθήσουμε αυτά τα αντικείμενα.

Λαμβάνοντας υπόψιν τις ικανότητες της αναγνώρισης αντικειμένων κάποιοι από τους τομείς που θα μπορούσε να χρησιμοποιηθεί είναι για αυτοκινούμενα οχήματα, καταμέτρηση πληθυσμού, παρακολούθηση βίντεο ή και αναγνώριση προσώπων.

2.2: Τρόπος λειτουργείας

Τώρα που γνωρίζουμε τι είναι η αναγνώριση αντικειμένων, τις διαφορές που υπάρχουν μεταξύ των διάφορων ειδών αναγνώρισης, και πως μπορεί να χρησιμοποιηθεί, ας μελετήσουμε πιο αναλυτικά τον τρόπο με τον οποίο δουλεύει.

Στα υποκεφάλαια που ακολουθούν παρουσιάζονται μερικές προσεγγίσεις βασισμένες στην βαθιά μάθηση όπου αξιολογούνται τα πλεονεκτήματα και οι περιορισμοί της αναγνώρισης αντικειμένων. Πιο συγκεκριμένα θα δούμε προσεγγίσεις που χρησιμοποιούν νευρωνικά δίκτυα, τα οποία είναι οι συχνότερη μέθοδος για την εφαρμογή της αναγνώρισης αντικειμένων.

2.2.1: Βασική δομή

Τα μοντέλα αναγνώρισης αντικειμένων που βασίζονται στην βαθιά μηχανική μάθηση συνήθως αποτελούνται από δύο μέρη. Έναν κωδικοποιητή που δέχεται την εικόνα και την περνά μέσα από μια σειρά ελέγχου μέσω της οποίας μαθαίνει να εξάγει στατιστικά χαρακτηριστικά τα οποία στην συνέχεια χρησιμοποιούνται για να εντοπιστούν και να

κατηγοριοποιηθούν τα ζητούμενα αντικείμενα. Ότι εξάγεται από τον κωδικοποιητή στην συνέχεια μεταφέρεται στον αποκωδικοποιητή, ο οποίος προβλέπει τα περιγράμματα και τις ταμπέλες από κάθε αντικείμενο.[1]

Ο πιο απλός αποκωδικοποιητής θεωρείται ένας παλινδρομητής. Ο οποίος είναι συνδεδεμένος στην έξοδο του κωδικοποιητή και προβλέπει την τοποθεσία και το μέγεθος του κάθε περιγράμματος. Το αποτέλεσμα του μοντέλου είναι οι καρτεσιανές συντεταγμένες του αντικειμένου και της έκτασής του πάνω στην εικόνα. Ωστόσο αν και απλό, αυτού του είδος το μοντέλο έχει περιορισμούς. Χρειάζεται να δηλωθεί ο αριθμός τον περιγραμμάτων εκ των προτέρων. Για παράδειγμα, εάν μια φωτογραφία απεικονίζει δύο σκυλιά, αλλά το μοντέλο έχει σχεδιαστεί για να εντοπίζει ένα αντικείμενο, ένα από τα δύο σκυλιά θα παραμείνει χωρίς ταμπέλα. Παρ' όλα αυτά, εάν ο χρήστης γνωρίζει τον αριθμό αντικειμένων που χρειάζεται να προβλέψει σε μια εικόνα, μοντέλα βασισμένα στην απλή παλινδρόμηση είναι μια καλή επιλογή.

Μια εξελιγμένη προσέγγιση του παλινδρομητή είναι ένα RPN (Region Proposal Network). Σε αυτού του τύπου τον αποκωδικοποιητή, το μοντέλο επεξεργάζεται τμήματα της εικόνας όπου πιστεύει ότι υπάρχει κάποιο αντικείμενο. Τα εικονοστοιχεία που ανήκουν σε αυτά τα τμήματα τροφοδοτούνται στην συνέχεια σε ένα υποδίκτυο κατηγοριοποίησης για να τους τοποθετηθεί μια ταμπέλα (ή να απορριφθούν). Τέλος τα εικονοστοιχεία που περιέχουν αυτά τα τμήματα περνούν από ένα δίκτυο κατηγοριοποίησης. Το προνόμιο αυτής της μεθόδου είναι να έχουμε ένα πιο ακριβές και ευέλικτο μοντέλο που μπορεί να προτείνει έναν αυθαίρετο αριθμό τμημάτων που θα μπορούσαν να περιέχουν ένα περίγραμμα. Η προστιθέμενη ακρίβεια, ωστόσο, κοστίζει υπολογιστική αποδοτικότητα.

2.2.2: Convolutional Neural Network (CNN)

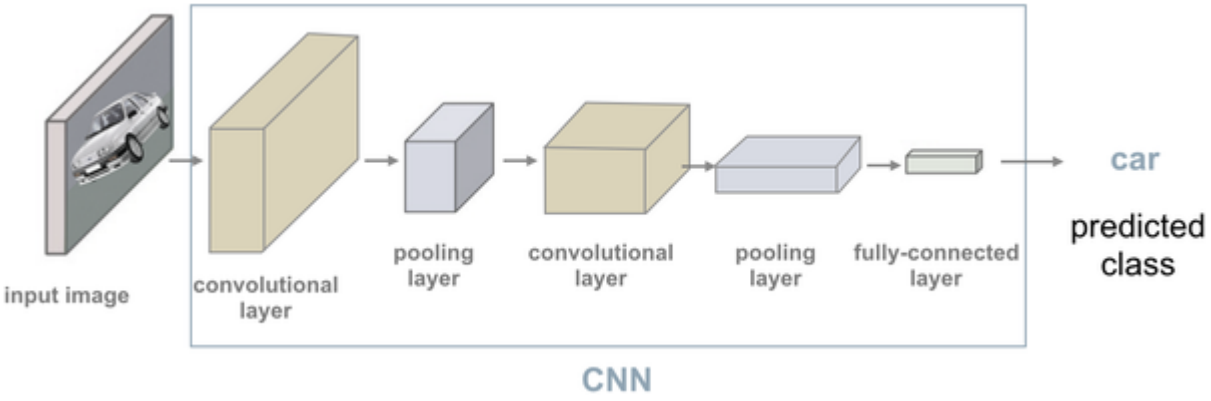
Για να μπορέσουμε να μιλήσουμε για μοντέλα αναγνώρισης αντικειμένων θα πρέπει πρώτα να αναφέρουμε τι είναι τα συνελικτικά νευρωνικά δίκτυα. Ένα συνελικτικό νευρωνικό δίκτυο είναι ένας τύπος τεχνητού δικτύου το οποίο χρησιμοποιείται για την αναγνώριση και επεξεργασία εικόνας και είναι ρυθμισμένο ώστε να επεξεργάζεται δεδομένα εικονοστοιχείων. Επομένως, συνελικτικά νευρωνικά δίκτυα είναι τα θεμέλια της υπολογιστικής όρασης για την τμηματοποίηση εικόνας.

Η αρχιτεκτονική τους αποτελείται από τρία βασικά επίπεδα:

- **Convolutional layer**: Το συγκεκριμένο βοηθάει στο να αποσπαστεί η εικόνα που εισέρχεται ως χάρτης χαρακτηριστικών μέσω φίλτρων και πυρήνων.

- **Pooling layer:** Το οποίο βοηθά στη μείωση της δειγματοληψίας των χαρτών χαρακτηριστικών συνοψίζοντας την παρουσία των χαρακτηριστικών επιδιορθώνοντάς τους.
- **Fully connected layer:** Το επίπεδο αυτό ενώνει κάθε νευρώνα σε ένα επίπεδο με κάθε νευρώνα των άλλων επιπέδων.

Συνδυάζοντας τα επίπεδα ενός δικτύου επιτρέπει στο σχεδιασμένο νευρωνικό δίκτυο να μάθει πώς να εντοπίζει και να αναγνωρίζει το αντικείμενο που μας ενδιαφέρει σε μια εικόνα. Απλά συνελικτικά νευρωνικά δίκτυα σχεδιάζονται για κατηγοριοποίηση εικόνων και αναγνώριση αντικειμένων με ένα μόνο αντικείμενο στην εικόνα.



Εικόνα 2: Παρουσίαση του πως δουλεύει ένα συνελικτικό νευρωνικό δίκτυο.

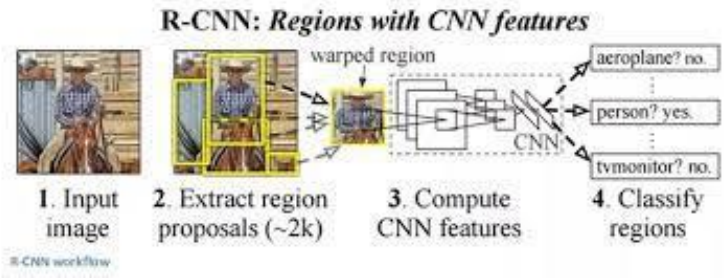
Σε πιο περίπλοκες περιπτώσεις με πολλά αντικείμενα σε μια εικόνα, ένα συνελικτικό νευρωνικό δίκτυο απλής αρχιτεκτονικής δεν είναι αρκετό. Για τέτοιες περιπτώσεις, χρησιμοποιούνται νευρωνικά δίκτυα βασισμένα σε R-CNN.

2.2.3: Τι είναι το R-CNN

R-CNN ή RCNN, βγαίνει από το Region-Based Convolutional Neural Network, είναι ένας τύπος μοντέλου μηχανικής μάθησης το οποίο χρησιμοποιείται για υπολογιστική όραση, με ειδίκευση στην αναγνώριση αντικειμένων.

Στην εικόνα που ακολουθεί απεικονίζεται η λογική του R-CNN. Αυτή η προσέγγιση χρησιμοποιεί πλαίσια οριοθέτησης γύρω από την περιοχή του αντικειμένου, το οποίο στη συνέχεια αξιολογεί τα συνελικτικά δίκτυα ανεξάρτητα σε όλες τις Περιφέρειες Ενδιαφέροντος (ROI) για να ταξινομήσει πολλές περιοχές εικόνας στην προτεινόμενη κλάση.

Η αρχιτεκτονική των RCNN δημιουργήθηκε για να λύνει προβλήματα εντοπισμού αντικειμένων. Επίσης, η αρχιτεκτονική R-CNN αποτελεί την βάση του Mask R-CNN που είναι μια αρκετά πιο βελτιωμένη έκδοση που θα εξηγήσουμε στην συνέχεια.[2]



Εικόνα 3 Προοπτική των R-CNN

2.2.4: One-stage και two-stage detectors

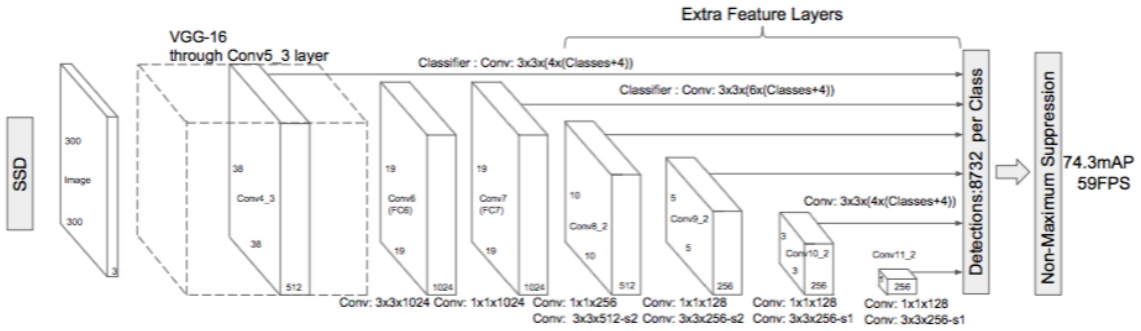
Είναι σημαντικό να αναφέρουμε πως τα μοντέλα αναγνώρισης χωρίζονται σε δύο κατηγορίες. Η πρώτη κατηγορία είναι οι two-stage detectors οι οποίοι σε πρώτη φάση χρησιμοποιούν είναι δίκτυο προτεινόμενης περιοχής (Region Proposal Network – RPN) για να παραχθούν περιοχές ενδιαφέροντος μέσα στις οποίες υπάρχουν τα αντικείμενα που εντοπίζονται και σε δεύτερη φάση στέλνει αυτές τις περιοχές για να γίνει κατηγοριοποίηση και εκτίμηση των πλαισίων οριοθέτησης, Τέτοια μοντέλα έχουν μεγαλύτερη ακρίβεια, αλλά είναι πιο αργά Στην κατηγορία αυτή ανήκει το Mask R-CNN. Η δεύτερη είναι οι one-stage detectors οι οποίοι αντιμετωπίζουν την αναγνώριση αντικειμένων ως ένα πρόβλημα παλινδρόμησης λαμβάνοντας σαν είσοδο την εικόνα και εντοπίζουν τις συντεταγμένες των πλαισίων οριοθέτησης και τις πιθανότητες του ανήκει σε μια κατηγορία το αντικείμενο. Τέτοια μοντέλα συνήθως υστερούν λίγο στην ακρίβεια αλλά είναι πολύ πιο γρήγορα από τα two-stage μοντέλα. Στην κατηγορία αυτή ανήκουν το SSD και το YOLOv5.

2.3: Single shot detectors (SSDs)

Το SSD (Single Shot Multibox Detector) εμφανίστηκε τέλη Νοεμβρίου του 2016 (Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg) και σε σχέση με τα μοντέλα που υπήρχαν μέχρι τότε το συγκεκριμένο κατέληξε να είναι το καλύτερο σε θέματα ακρίβειας και επίδοσης όσον αφορά την αναγνώριση αντικειμένων, με σκορ πάνω από 74% mAP (mean Average Precision) στα 59 fps (frames per second) σε στάνταρ σετ δεδομένων. Το όνομα της συγκεκριμένης αρχιτεκτονικής προέρχεται από:

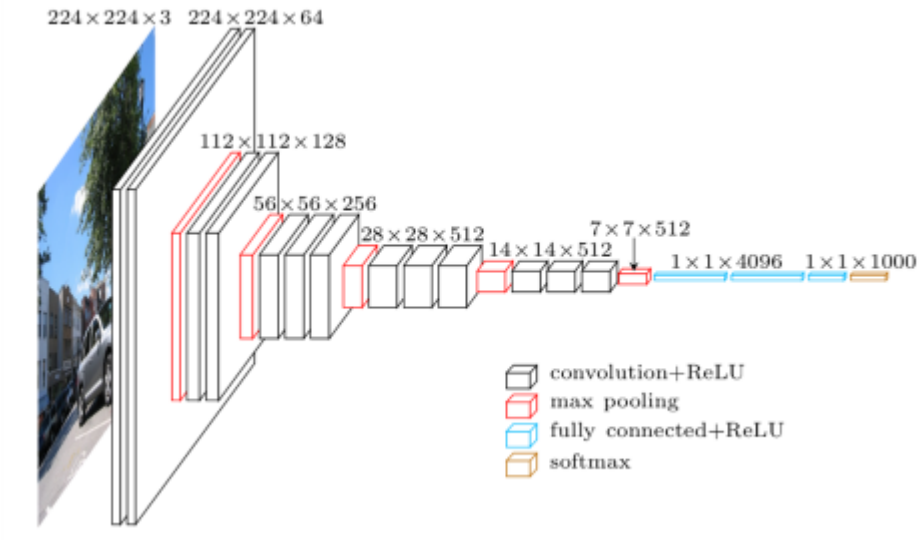
- **Single Shot:** αυτό σημαίνει ότι ο εντοπισμός και η ταξινόμηση του αντικειμένου γίνεται σε ένα μόνο πέρασμα του νευρωνικού δικτύου.
- **MultiBox:** αυτό είναι το όνομα της τεχνικής για την παλινδρόμηση οριοθέτησης περιγράμματος.
- **Detector:** το συγκεκριμένο νευρωνικό δίκτυο είναι ένα μοντέλο αναγνώρισης αντικειμένων το οποίο ταξινομεί τα συγκεκριμένα αντικείμενα.

2.3.1: Αρχιτεκτονική



Εικόνα 4 Αρχιτεκτονική του μοντέλου SSD

Όπως φαίνεται και στο παραπάνω διάγραμμα, η αρχιτεκτονική του SSD έχει στηριχτεί στο VGG-16 μοντέλο το οποίο είναι ένα από τα πρώτα που δημιουργήθηκαν για αναγνώριση και ταξινόμηση. Οι λόγοι που το μοντέλο αυτό χρησιμοποιήθηκε ως τα θεμέλια του SSD είναι η απόδοσή του σε εργασίες ταξινόμησης εικόνων υψηλής ανάλυσης και η δημοτικότητα για προβλήματα όπου το tranfer learning βοηθά στην βελτίωση των αποτελεσμάτων. Αντί για τα αρχικά VGG πλήρως ενωμένα layers, προστέθηκε ένα set από βοηθητικά συνελικτικά στρώματα, επιτρέποντας έτσι την εξαγωγή χαρακτηριστικών σε πολλαπλές κλίμακες και μειώνοντας σταδιακά το μέγεθος της εικόνας σε κάθε επόμενο στρώμα.



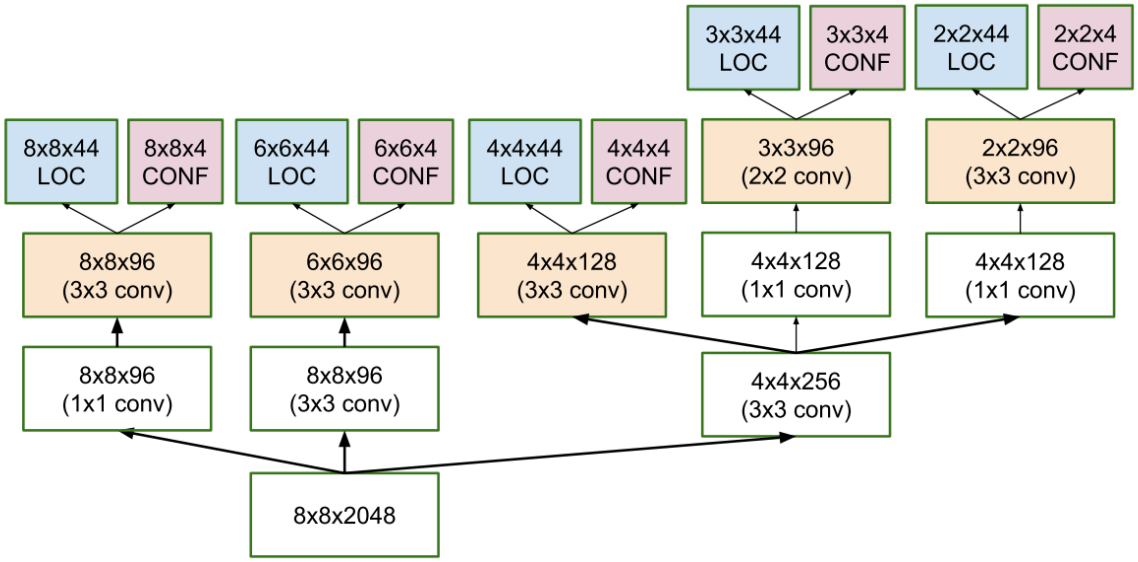
Εικόνα 5 VGG architecture

2.3.2: MultiBox

Η τεχνική παλινδρόμησης οριοθέτησης περιγράμματος του SSD, είναι μια μέθοδος η οποία με μεγάλη ταχύτητα βρίσκει συντεταγμένες από οριοθετημένα περιγράμματα από αντικείμενα που

προς το παρόν η κατηγορία στην οποία ανήκουν είναι άγνωστη. Είναι αρκετά ενδιαφέρον το γεγονός ότι η επεξεργασία που γίνεται στο MultiBox χρησιμοποιείται ένα Inception-style συνελικτικό δίκτυο.

Το κύριο χαρακτηριστικό της αρχιτεκτονικής Inception είναι η βελτιωμένη χρήση των υπολογιστικών πόρων μέσα στο δίκτυο. Αυτό επιτεύχθηκε με έναν σχεδιασμό που επιτρέπει την αύξηση του βάθους και του πλάτους του δικτύου διατηρώντας παράλληλα τον υπολογιστικό προϋπολογισμό σταθερό.



Εικόνα 6 Αρχιτεκτονική μιας πολυεπίπεδης συνελικτικής πρόβλεψης της θέσης και της σιγουριάς του multibox.

Η συνάρτηση απόκλισης του MultiBox συνδυάζει δυο βασικούς παράγοντες που βοήθησαν να ενταχθεί στο SSD μοντέλο:

- **Confidence Loss:** μετράει πόσο σίγουρο είναι το δίκτυο για την αντικειμενικότητα του υπολογισμένου πλαισίου οριοθέτησης. Η Categorical cross-entropy χρησιμοποιείται για τον υπολογισμό αυτής της απώλειας.
- **Location Loss:** μετράει πόση είναι η απόκλιση του πλαισίου οριοθέτησης που πρόβλεψε το μοντέλο από την κανονική θέση του πλαισίου. Για να επιτευχθεί αυτό χρησιμοποιείται L2-Norm. Το οποίο μαθηματικά είναι το συνολικό μέγεθος η μάκρος των διανυσμάτων σε ένα διανυσματικό χώρο ή πίνακες. Με λίγα λόγια, μπορούμε να πούμε ότι όσο υψηλότερο είναι το norm, τόσο μεγαλύτερο είναι σε τιμή το διάνυσμα ή ο πίνακας.

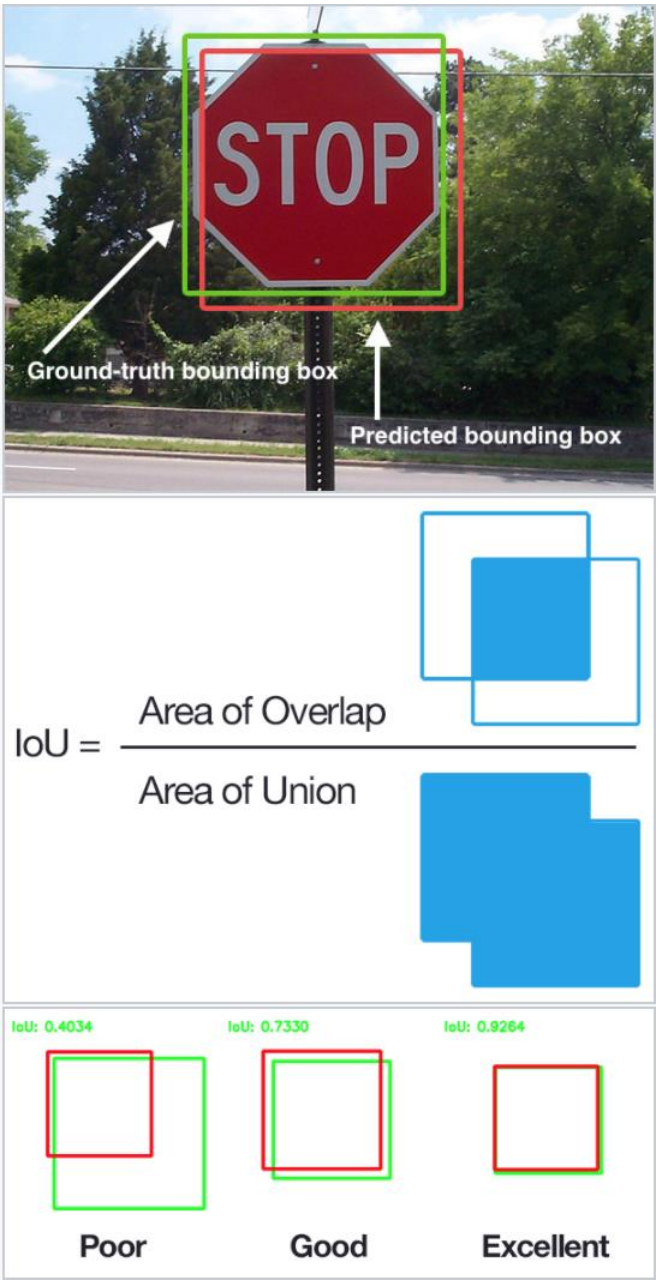
Με λίγα λόγια οι συνάρτηση που υπολογίζει πόσο απέχουν τα πλαίσια είναι η εξής:

multibox_loss = confidence_loss + alpha * location_loss.

Η μεταβλητή alpha βοηθάει στην εξισορρόπηση του location_loss. Ως συνήθως στην βαθιά μηχανική μάθηση, ο στόχος είναι να βρεθούν οι τιμές των παραμέτρων που μειώνουν την απώλεια, δηλαδή να έρθουν οι προβλέψεις πιο κοντά στις πραγματικές τιμές.

2.3.3: MultiBox Priors And IoU

Η λογική γύρω από την σχεδίαση του κουτιού οριοθέτησης στην πραγματικότητα είναι πιο περίπλοκη. Στη μέθοδο MultiBox, έχουν δημιουργηθεί άγκυρες (anchors), οι οποίες είναι προ-υπολογισμένα, στάνταρ μεγέθους πλαίσια οριοθέτησης τα οποία μοιάζουν αρκετά στα πραγματικά πλαίσια. Στην πραγματικότητα οι άγκυρες αυτές επιλέγονται με τέτοιο τρόπο ώστε ο βαθμός Intersection over Union (IoU) να είναι μεγαλύτερος του 0.5. Όπως φαίνεται και στην παρακάτω εικόνα, ένα IoU με 0.5 δεν επαρκεί αλλά παρ' όλα αυτά προσφέρει ένα δυνατό σημείο αναφοράς για την παλινδρόμηση του κουτιού οριοθέτησης. Η τεχνική αυτή είναι πολύ καλύτερη απ' το να ξεκινάει η πρόβλεψη με τυχαίες συντεταγμένες.



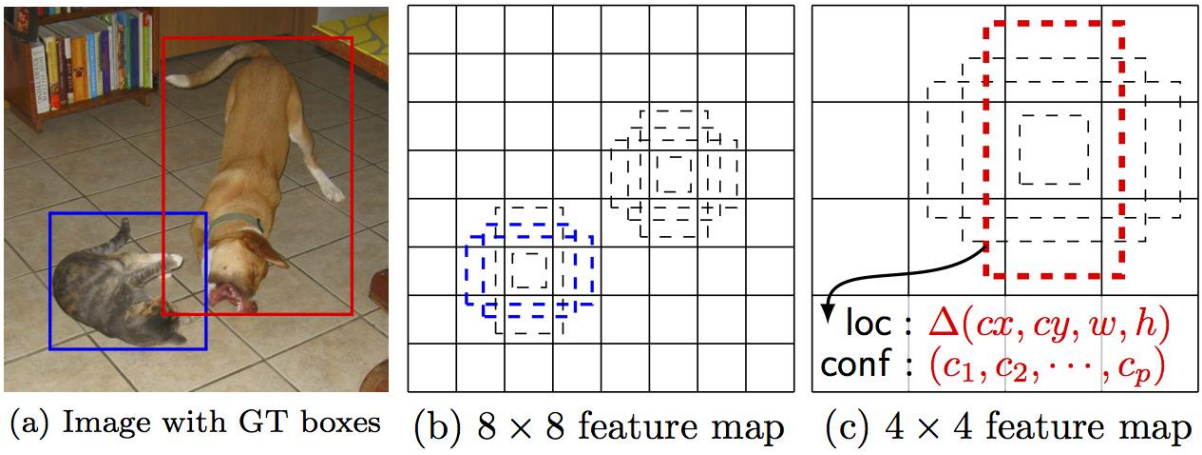
Εικόνα 7 Διάγραμμα επεξήγησης IoU

Η τελική αρχιτεκτονική περιέχει 11 άγκυρες για κάθε feature map cell (8x8, 6x6, 4x4, 3x3, 2x2) και μόνο ένα 1x1 feature map, με αποτέλεσμα να υπάρχουν στο σύνολο 1430 άγκυρες για κάθε εικόνα, επιτρέποντας έτσι ισχυρή κάλυψη για εικόνες σε πολλαπλές κλίμακες, για τον εντοπισμό αντικειμένων διαφόρων μεγεθών.

2.3.4: Βελτιώσεις SSD

Σταθερές Άγκυρες: σε αντίθεση με τη μέθοδο των MultiBox, κάθε feature map cell συσχετίζεται με ένα σετ από προκαθορισμένα πλαίσια οριοθέτησης με διαφορετικές διαστάσεις και

αναλογίες. Αυτές οι άγκυρες επιλέγονται χειροκίνητα, ενώ στα MultiBox, επιλέγονταν επειδή το IoU τους ως προς τα πραγματικά πλαίσια οριοθέτησης ήταν μεγαλύτερο από 0.5. Αυτό στη θεωρεία επιτρέπει στο SSD να γενικεύει για κάθε είδος εισόδου, δίχως να χρειάζεται μια προ-εκπαιδευμένη φάση για την παραγωγή αγκυρών. Για παράδειγμα, υποθέτοντας ότι έχουμε διαμορφώσει 2 διαγώνια αντίθετα σημεία (x1, y1) και (x2, y2) για κάθε b προκαθορισμένα πλαίσια οριοθέτησης ανά feature map cell, και c κλάσεις για ταξινόμηση, σε έναν feature map μεγέθους f = m * n, το SSD θα υπολόγιζε τιμές ίσες με f * b * (4 + c) για το συγκεκριμένο feature map.



Εικόνα 8 SSD προκαθορισμένα πλαίσια σε 8x8 και 4x4 feature maps

Location Loss: το SSD χρησιμοποιεί L1-Norm για να υπολογίσει την απώλεια τοποθεσίας. Παρ' όλο που δεν είναι τόσο ακριβές όσο το L2-Norm, είναι αρκετά αποτελεσματικό και δίνει στο SSD περισσότερο περιθώριο ελιγμών καθώς δεν προσπαθεί να είναι «pixel perfect» στην πρόβλεψη του πλαισίου οριοθέτησης.

Classification: το MultiBox δεν έχει ταξινόμηση αντικειμένων, σε σχέση με το SSD που έχει. Επομένως, για κάθε προβλεπόμενο πλαίσιο οριοθέτησης, υπολογίζεται ένα σύνολο προβλέψεων από c κλάσεις, για κάθε πιθανή κλάση στο σύνολο δεδομένων.

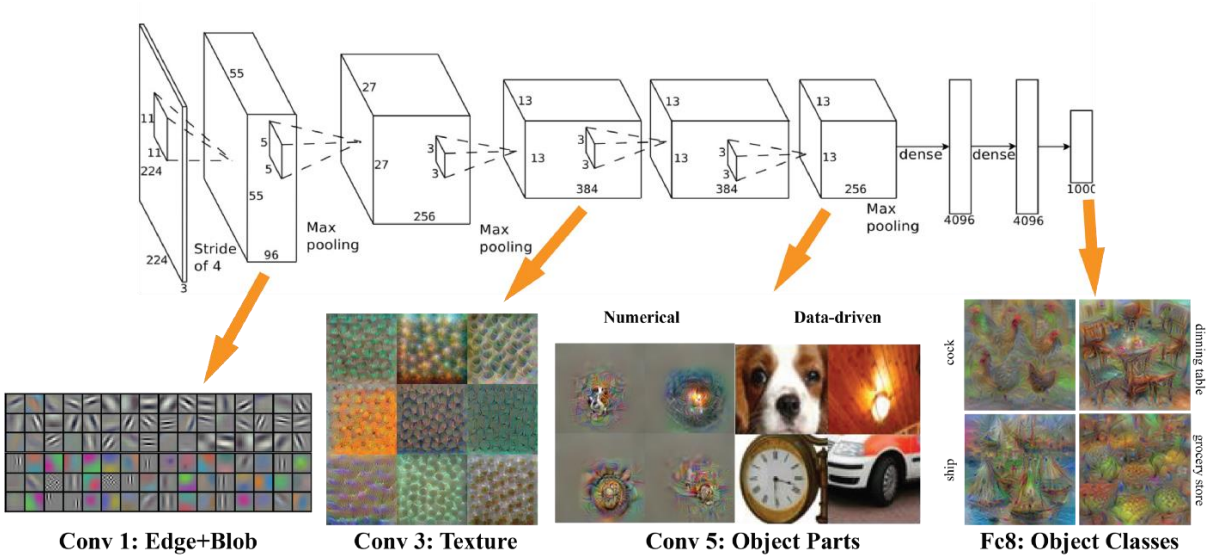
2.3.5: Εκπαίδευση του SSD

Στην υπό-παράγραφο που ακολουθεί γίνεται μια επεξήγηση για το πως εκπαιδεύεται και λειτουργεί ένα μοντέλο SSD.

Αρχικά για την εκπαίδευση του SSD χρειάζεται να υπάρχουν σετ δεδομένων για εκπαίδευση και για δοκιμές με τα πραγματικά πλαίσια οριοθέτησης και με εκχωρημένες ετικέτες κλάσης.

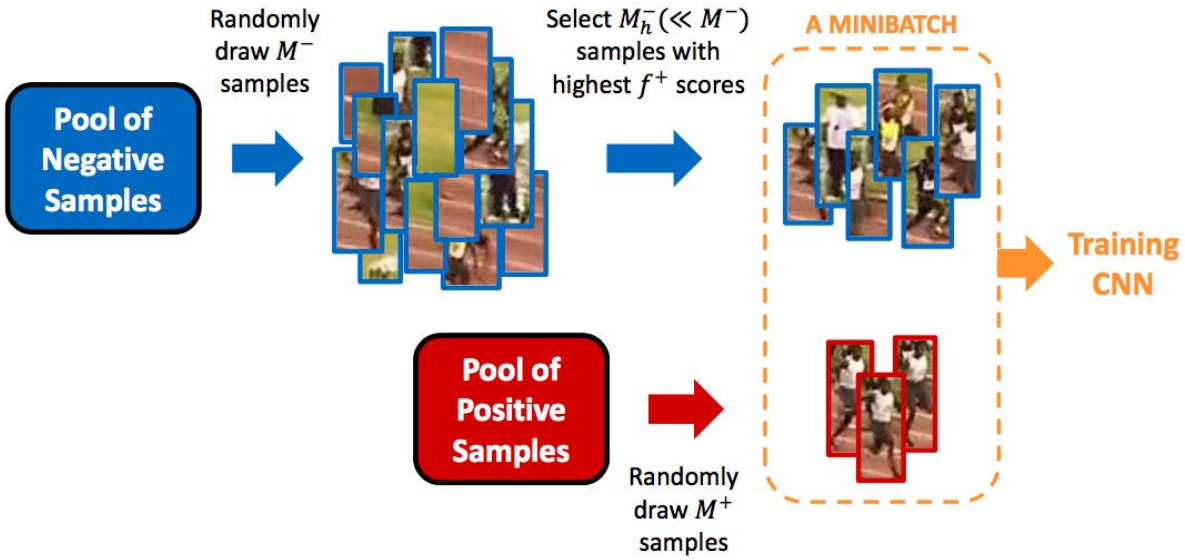
Επίσης προτείνεται να διαμορφώνεται ένα ποικίλο σετ από προκαθορισμένα πλαίσια οριοθέτησης, με διαφορετικές κλίμακες και μεγέθη έτσι ώστε να εξασφαλισθεί η αναγνώριση των περισσότερων αντικειμένων.

Επιπλέον τα feature maps είναι μια αναπαράσταση από τα κυρίαρχα χαρακτηριστικά μιας εικόνας σε διαφορετικά μεγέθη, επομένως εφαρμόζοντας MultiBox σε πολλαπλά feature maps αυξάνει την πιθανότητα κάθε αντικείμενο (μικρό ή μεγάλο) να εντοπιστεί, να αναγνωριστεί και να ταξινομηθεί κατάλληλα. Η παρακάτω εικόνα δείχνει πως το δίκτυο βλέπει μια εικόνα στα feature maps της



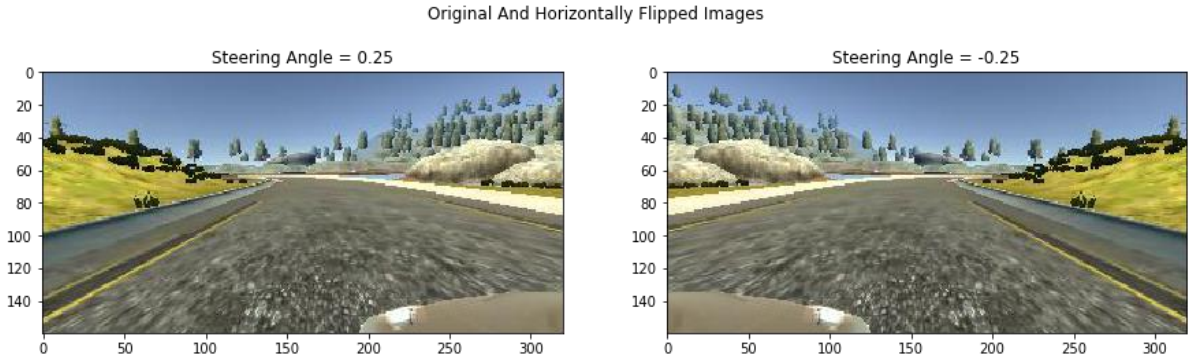
Εικόνα 9 VGG Feature Map Visualization

Επιπρόσθετα υπάρχει και η Βαριά Αρνητική Εξόρυξη (Hard Negative Mining), κατά την διάρκεια της εκπαίδευσης, καθώς τα περισσότερα πλαίσια οριοθέτησης θα έχουν χαμηλό IoU και επομένως θα ερμηνεύονται ως αρνητικά παραδείγματα εκπαίδευσης, υπάρχει περίπτωση να καταλήξουμε με ένα δυσανάλογο αριθμό από αρνητικά παραδείγματα στο σετ μας. Επομένως, αντί να χρησιμοποιούμε όλες τις αρνητικές προβλέψεις, συνιστάται να υπάρχει ένα 3:1 ποσοστό θετικών προς αρνητικών παραδειγμάτων. Ο λόγος που πρέπει να κρατήσουμε τα αρνητικά παραδείγματα είναι επειδή το δίκτυο χρειάζεται να μάθει το πως είναι μια εσφαλμένη πρόβλεψη.



Εικόνα 10 Παράδειγμα από βαριά αρνητική εξόρυξη

Οι δημιουργοί του SSD δήλωσαν ότι η αύξηση δεδομένων, όπως και πολλές άλλες εφαρμογές βαθιάς μάθησης, έπαιξαν σημαντικό ρόλο για να μάθει το δίκτυο να είναι πιο στιβαρό κατά την είσοδο ποικίλων μεγεθών αντικειμένων. Μέχρι σήμερα, έχουν δημιουργήσει επιπλέον παραδείγματα εκπαίδευσης με μπαλώματα της πραγματικής εικόνας σε διαφορετικά μεγέθη IoU αλλά και σε τυχαία μπαλώματα. Επιπλέον, κάθε εικόνα περιστρέφεται τυχαία με πιθανότητα 0.5, διασφαλίζοντας έτσι πιθανά αντικείμενα να εμφανίζονται αριστερά και δεξιά με παρόμοια πιθανότητα.



Εικόνα 11 Παράδειγμα οριζόντιας περιστροφής

Λαμβάνοντας υπόψιν το μεγάλο αριθμό των πλαισίων που παράγονται κατά τη διάρκεια μιας προώθησης SSD κατά τον χρόνο εξαγωγής, είναι σημαντικό να απορριφθούν τα περισσότερα πλαίσια οριοθέτησης εφαρμόζοντας μια τεχνική γνωστή ως Non-Maximum Suppression (NMS). Αυτό που κάνει η συγκεκριμένη τεχνική είναι τα πλαίσια με σιγουριά μικρότερη από ct (π.χ. 0.01) και IoU μικρότερο από lt (π.χ. 0.45) τα απορρίπτει, και μόνο αυτά με το μεγαλύτερο ποσοστό ακρίβειας αποθηκεύονται. Αυτό εξασφαλίζει ότι οι πιθανότερες προβλέψεις διατηρούνται από το δίκτυο, ενώ οι λιγότερο πιθανές απορρίπτονται. [3]

Figure 1: We propose a non-maximum suppression convnet that will re-score all raw detections (top). Our network is trained end-to-end to learn to generate exactly one high scoring detection per object (bottom, example result).

Εικόνα 12 NMS παράδειγμα

2.4: YOLO Model (You Only Look Once)

Το συγκεκριμένο μοντέλο είναι ένα από τα 3 που χρησιμοποιήθηκαν για την υλοποίηση της διπλωματικής εργασίας. Το YOLO (Joseph Redmon, 2016) είναι ένας αλγόριθμος που χρησιμοποιεί νευρωνικά δίκτυα για τα παρέχει αναγνώριση αντικειμένων σε πραγματικό χρόνο. Ο συγκεκριμένος αλγόριθμος είναι γνωστός λόγω της ταχύτητας και της ακρίβειάς του. Έχει χρησιμοποιηθεί σε ποικίλες εφαρμογές για τον εντοπισμό πινακίδων, ανθρώπων, παρκόμετρων και ζώων. Στη συγκεκριμένη παράγραφο γίνεται μια επεξήγηση για το πως δουλεύει το συγκεκριμένο μοντέλο.

2.4.1: Τι είναι το YOLO

YOLO είναι η συντομογραφία του όρου “You Only Look Once”. Είναι ένας αλγόριθμος που εντοπίζει και αναγνωρίζει διάφορα αντικείμενα σε μια εικόνα. Η αναγνώριση αντικειμένων στο YOLO αντιμετωπίζεται σαν ένα πρόβλημα παλινδρόμησης και ως αποτέλεσμα δίνει την πιθανότητα που έχει το αντικείμενο προς αναγνώριση να ανήκει σε μια κατηγορία.

Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί συνελικτικά νευρωνικά δίκτυα για την αναγνώριση αντικειμένων σε πραγματικό χρόνο. Όπως φαίνεται και από το όνομα ο αλγόριθμος απαιτεί ένα μόνο πέρασμα μέσω ενός νευρωνικού δικτύου για τον εντοπισμό αντικειμένων.

Αυτό σημαίνει ότι η πρόβλεψη σε μια ολόκληρη εικόνα γίνεται μέσα σε ένα κύκλο του αλγορίθμου. Το CNN χρησιμοποιείται για την πρόβλεψη ποικίλων πιθανοτήτων μια κατηγορίας και περιμετρικών πλαισίων ταυτοχρόνως. Επίσης ο αλγόριθμος YOLO έχει διάφορες εκδοχές. Στην συγκεκριμένη διπλωματική χρησιμοποιείται η το μοντέλο YOLOv5.

2.4.2: Γιατί ο αλγόριθμος YOLO είναι σημαντικός

Ο συγκεκριμένος αλγόριθμος είναι σημαντικός για τους εξής λόγους:

- **Ταχύτητα**: Ο χρόνος εντοπισμού βελτιώνεται επειδή μπορεί να κάνει προβλέψεις σε πραγματικό χρόνο.
- **Υψηλή ακρίβεια**: Το μοντέλο YOLO είναι μια τεχνική πρόβλεψης που παρέχει ακριβή αποτελέσματα με ελάχιστα λάθη στο παρασκήνιο.
- **Μαθησιακές δυνατότητες**: Ο αλγόριθμος έχει εξαιρετικές μαθησιακές πιθανότητες που του επιτρέπουν να μάθει το παρουσιαστικό των αντικειμένων και να τα εφαρμόσει στην αναγνώριση αντικειμένων.

2.4.3: Πως δουλεύει ο αλγόριθμος YOLO

Ο συγκεκριμένος αλγόριθμος δουλεύει χρησιμοποιώντας τις εξής τρεις τεχνικές:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

Residual Blocks

Αρχικά, η εικόνα διαχωρίζεται σε διάφορα πλέγματα. Κάθε πλέγμα έχει διάσταση S x S. Η παρακάτω εικόνα δείχνει πως μια εικόνα εισόδου χωρίζεται σε πλέγματα.

Εικόνα 13 Διαχωρισμός εικόνας σε πλέγματα

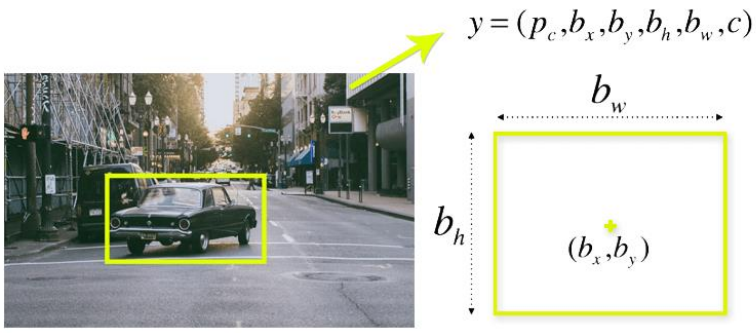
Στην παραπάνω εικόνα, υπάρχουν πολλά πλέγματα με ίσες διαστάσεις. Κάθε πλέγμα εντοπίζει αντικείμενα που εμφανίζονται μέσα σε αυτά. Για παράδειγμα, εάν εμφανιστεί κάποιο αντικείμενο μέσα σε ένα συγκεκριμένο πλέγμα, τότε αυτό το πλέγμα είναι υπεύθυνο για τον εντοπισμό του αντικειμένου.

Bounding box regression

Ένα περιμετρικό πλαίσιο είναι μια γραμμή που τονίζει ένα αντικείμενο στην εικόνα. Κάθε τέτοιο πλαίσιο αποτελείται από τα ακόλουθα χαρακτηρίστηκα:

- Πλάτος
- Ύψος
- Κατηγορία (π.χ. άνθρωπος, αμάξι, φωτεινός σηματοδότης, κ.τ.λ.)
- Το κέντρο του

Στην εικόνα που ακολουθεί παρουσιάζεται ένα παράδειγμα περιμετρικού πλαισίου. Το οποίο παρουσιάζεται με κίτρινο περίγραμμα.



Εικόνα 14 Περιμετρικό κουτί

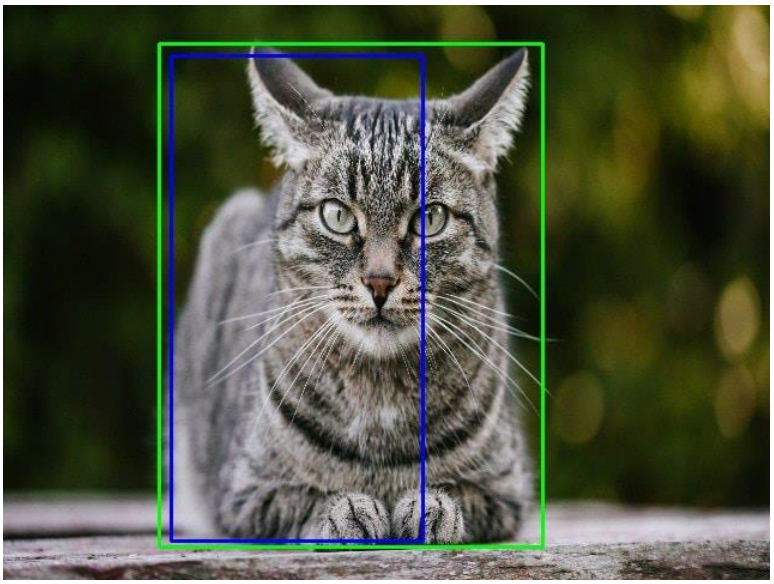
Το YOLO χρησιμοποιεί μια παλινδρόμηση περιμετρικού πλαισίου για να προβλέψει το ύψος, πλάτος, κέντρο και κατηγορία του αντικειμένου. Η παραπάνω εικόνα, παρουσιάζει την πιθανότητα να παρουσιάζεται κάποιο αντικείμενο μέσα στο περιμετρικό πλαίσιο.

Intersection over union (IOU)

Intersection over union είναι μια μετρική στη αναγνώριση αντικειμένων που περιγράφει πως αλληλεπικαλύπτονται το πραγματικό πλαίσιο σε σχέση με το πλαίσιο πρόβλεψης. Το YOLO χρησιμοποιεί IOU για να παρέχει ένα πλαίσιο το οποίο καλύπτει περιμετρικά τέλεια τα αντικείμενα.

Κάθε πλεγματικό κελί είναι υπεύθυνο για να προβλέπει τα περιμετρικά πλαίσια και το ποσοστό σιγουριάς τους. Το IOU είναι ίσο με 1 εάν το περιμετρικό πλαίσιο που βρέθηκε είναι το ίδιο με το πραγματικό. Αυτός ο μηχανισμός απορρίπτει περιμετρικά πλαίσια που δεν είναι ίδια με τα πραγματικά.

Η εικόνα που ακολουθεί παρουσιάζει ένα παράδειγμα για το πως δουλεύει το IOU.

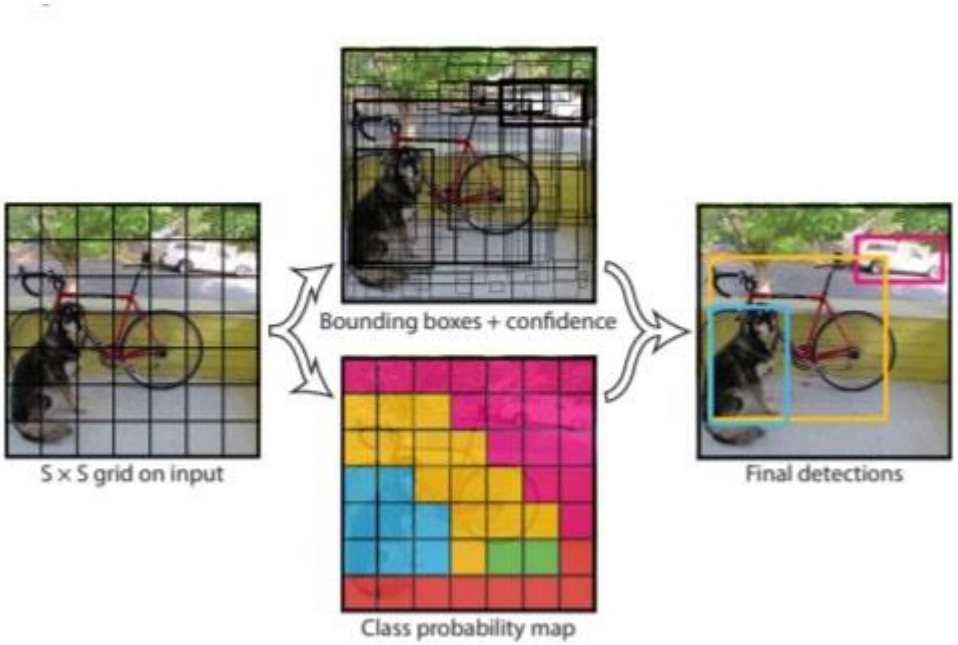


Εικόνα 15 Παράδειγμα λειτουργίας IOU

Στην παραπάνω εικόνα, υπάρχουν δύο περιμετρικά πλαίσια, ένα πράσινο και ένα μπλε. Το μπλε είναι το προβλεπόμενο πλαίσιο και το πράσινο το πραγματικό. Το YOLO διασφαλίζει ότι τα δύο περιμετρικά πλαίσια είναι ίσα.

Συνδυασμός των τριών τεχνικών

Στην παρακάτω εικόνα παρουσιάζεται πως εφαρμόζονται και οι τρεις τεχνικές για να παραχθεί το τελικό αποτέλεσμα.



Εικόνα 16 Πως δουλεύει ο αλγόριθμος YOLO

Πρώτων, η εικόνα διαιρείται σε πλέγματα. Κάθε πλέγμα προβλέπει Β περιμετρικά πλαίσια και παρέχει τα ποσοστά σιγουριάς τους. Τα κελιά προβλέπουν τις πιθανότητες σχετικά με το σε ποια κατηγορία ανήκει το αντικείμενο.

Για παράδειγμα, μπορούμε να παρατηρήσουμε τουλάχιστον τρεις κατηγορίες αντικειμένων: ένα σκυλί, ένα αμάξι και ένα ποδήλατο. Όλες οι προβλέψεις γίνονται ταυτόχρονα χρησιμοποιώντας ένα συνελικτικό νευρωνικό δίκτυο.

Το Intersection over union επιβεβαιώνει πως τα προβλεπόμενα περιμετρικά πλαίσια είναι ίδια με τα πραγματικά των αντικειμένων. Το φαινόμενο αυτό απορρίπτει περιττά περιμετρικά πλαίσια τα οποία δεν πληρούν τις προϋποθέσεις των αντικειμένων (όπως ύψος και πλάτος). Η τελική πρόβλεψη αποτελείται από ξεχωριστά περιμετρικά πλαίσια τα οποία περιέχουν τα αντικείμενα τέλεια.

Για παράδειγμα, το αμάξι περιέχεται μέσα στο ροζ πλαίσιο ενώ το ποδήλατό σε πορτοκαλί. Το σκυλί τονίζεται με ένα μπλε περιμετρικό πλαίσιο.

2.4.4: Εφαρμογές που μπορεί να χρησιμοποιηθεί

Το YOLO μπορεί αν εφαρμοστεί στους ακόλουθους τομείς:

- **Αυτόνομη οδήγηση**: Ο αλγόριθμος μπορεί να χρησιμοποιηθεί σε αυτοκινούμενα οχήματα για την αναγνώριση αντικειμένων γύρω από το αμάξι όπως είναι άλλα οχήματα, άνθρωποι και σήματα παρκαρίσματος. Η αναγνώριση αντικειμένων στα αυτοκινούμενα οχήματα γίνεται για την αποφυγή τρακαρισμάτων εφόσον το αμάξι δεν ελέγχεται από κάποιο άτομο.

- **Πανίδα:** Ο αλγόριθμος χρησιμοποιείτε για τον εντοπισμό διαφόρων ειδών ζώων στα δάση. Αυτού του τύπου η αναγνώριση βοηθάει δασοφύλακες και δημοσιογράφους να αναγνωρίσουν ζώα σε βίντεο και εικόνες. Μερικά από τα ζώα που μπορούν εντοπιστούν είναι καμηλοπαρδάλεις, ελέφαντες και αρκούδες.
- **Φύλαξη**: Το YOLO επίσης χρησιμοποιείται σε συστήματα ασφαλείας για την ενίσχυση της ασφάλειας σε μια περιοχή. Ας υποθέσουμε ότι έχει απαγορευτεί να διέρχεται κόσμος από μια συγκεκριμένη περιοχή για λόγους ασφάλειας. Αν κάποιος περάσει μέσα από απαγορευμένη περιοχή, ο αλγόριθμος θα τον/την εντοπίσει, και θα ενημερώσει το προσωπικό να λάβει δράση. [4]

2.4.5: Βελτιώσεις του YOLOv5

Όπως προαναφέρθηκε για την συγκεκριμένη διπλωματική χρησιμοποιήθηκε το μοντέλο YOLOv5 οι βελτιώσεις που έχουν γίνει στο συγκεκριμένο μοντέλο σε σχέση με τους προγόνους έχουν ως εξής.

Αρχικά, είναι το πρώτο μοντέλο της οικογένειας YOLO που γράφτηκε σε PyTorch αντί για Darknet. Το Darknet είναι μια αρκετά ευέλικτη πλατφόρμα, αλλά δεν έχει χτιστεί με περιβάλλον παραγωγής κατά νου. Επίσης έχει μικρότερο αριθμό χρηστών. Λαμβάνοντας όλα αυτά υπόψιν, συμπεραίνουμε πως το Darknet είναι πιο δύσκολο στην διαμόρφωση και λιγότερο παραγωγικά έτοιμο.

Επειδή το YOLOv5 εφαρμόζεται αρχικά στο PyTorch, επωφελείται από το καθιερωμένο οικοσύστημά του. Δηλαδή η υποστήριξή του είναι απλούστερη και η ανάπτυξη ευκολότερη. Επιπλέον, ως ευρύτερα γνωστό ερευνητικό πλαίσιο, η χρήση του YOLOv5 μπορεί να είναι ευκολότερη λόγω τις ευρύτερης ερευνητικής κοινότητας.

Επιπλέον, το YOLOv5 είναι εκπληκτικά πιο γρήγορο. Σε ένα πείραμα που χρησιμοποιήθηκε διαπιστώθηκαν χρόνοι συμπερασμάτων έως 0.007 δευτερόλεπτα ανά εικόνα, δηλαδή 140 frames το δευτερόλεπτο. Αντίθετα με το YOLOv4 που πέτυχε 50 frames το δευτερόλεπτο αφού μετατράπηκε στην ίδια βιβλιοθήκη PyTorch.

Ακόμη, το YOLOv5 είναι ακριβές. Στα πειράματα που έγιναν, πέτυχαν ακρίβεια περίπου 0.895 μετά από 100 εποχές εκπαίδευσης.

Τέλος, είναι μικρό σε μέγεθος. Συγκριτικά με το YOLOv4 το YOLOv5 είναι σχεδόν ενενήντα τις εκατό πιο ελαφρύ. Αυτό σημαίνει ότι μπορεί να αναπτυχθεί σε ενσωματωμένες συσκευές πολύ πιο εύκολα.[5]

2.5: Mask R-CNN

Το συγκεκριμένο μοντέλο είναι συνελικτικό νευρωνικό δίκτυο ένα από τα καλύτερα για τμηματοποίηση εικόνας και περιστατικών. Το Mask R-CNN (Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, 2017) αναπτύχθηκε πάνω στο Faster R-CNN, ένα Region-Based Convolutional Neural Network

Αρχικά για να κατανοήσουμε το πώς δουλεύει το Mask R-CNN θα πρέπει πρώτα να καταλάβουμε την λογική της τμηματοποίησης της εικόνας.

Το συγκεκριμένο έργο είναι η διαδικασία το να χωριστεί μια ψηφιακή εικόνα σε πολλαπλά τμήματα. Αυτή η τμηματοποίηση χρησιμοποιείται για να εντοπιστούν αντικείμενα και όρια.

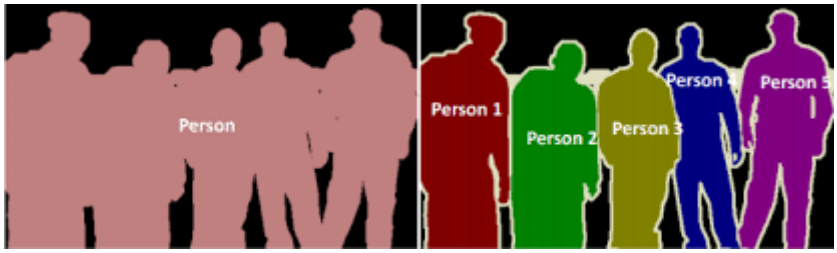
Υπάρχουν δύο τύποι τμηματοποίησης εικόνας που περιέχονται στο Mask R-CNN:

1. Semantic Segmentation
2. Instance Segmentation

2.5.1 Semantic Segmentation

Η συγκεκριμένη τμηματοποίηση κατηγοριοποιεί κάθε εικονοστοιχείο σε συγκεκριμένο σετ κατηγοριών χωρίς διαφορετικά αντικείμενα. Με λίγα λόγια, η semantic segmentation είναι υπεύθυνη για την αναγνώριση/ κατηγοριοποίηση ίδιων αντικειμένων σαν μια κλάση.

Όπως φαίνεται και στην παρακάτω εικόνα όλα τα αντικείμενα κατηγοριοποιούνται ως μια οντότητα (άτομο). Είναι γνωστή αλλιώς και ως τμηματοποίηση φόντου επειδή διαχωρίζει τα αντικείμενα της εικόνας από το φόντο.



Εικόνα 17 Διαφορά μεταξύ Semantic και Instance Segmentation

2.5.2: Instance Segmentation

Instance Segmentation, ή Instance Recognition, είναι η τμηματοποίηση η οποία είναι υπεύθυνη για την ορθή αναγνώριση όλων των αντικειμένων στην εικόνα καθώς επίσης τμηματοποιεί κάθε παρουσία. Είναι επομένως, ένας συνδυασμός αναγνώρισης αντικειμένων, εντοπισμός αντικειμένων και κατηγοριοποίηση αντικειμένων. Με λίγα λόγια, αυτός ο τύπος τμηματοποίησης

προχωρεί περαιτέρω για να δώσει μια σαφή διάκριση μεταξύ κάθε αντικειμένου που ταξινομείται ως παρόμοιες παρουσίες.

Όπως φαίνεται και στο παραπάνω παράδειγμα, για την Instance Segmentation, όλα τα αντικείμενα είναι άτομα, αλλά αυτή η διαδικασία ξεχωρίζει κάθε άτομο σαν ξεχωριστή οντότητα. Τέλος είναι γνωστή και ως τμηματοποίηση προσκήνιου επειδή τονίζει τα αντικείμενα της εικόνας αντί για το φόντο.

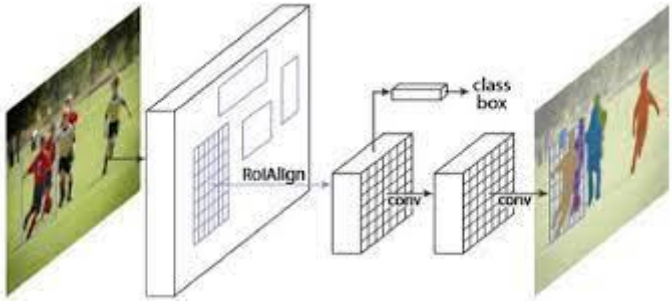
2.5.3: Πως δουλεύει το Mask R-CNN

Το Mask R-CNN φτιάχτηκε πάνω στο Faster R-CNN. Ενώ το Faster R-CNN είχε 2 εξόδους για κάθε υποψήφιο αντικείμενο, μια label και μια απόκλιση του κουτιού οριοθέτησης, το Mask R-CNN είναι η προσθήκη ενός τρίτου κλάδου που εξάγει τη μάσκα του αντικειμένου. Η πρόσθετη έξοδος διαφέρει από τις εξόδους κλάσης και κουτιού, απαιτώντας την εξαγωγή μιας πολύ λεπτότερης χωρικής διάταξης ενός αντικειμένου.

Το Mask R-CNN είναι μια προέκταση του Faster R-CNN και δουλεύει προσθέτοντας ένα κλάδο για να προβλέπει την μάσκα ενός αντικειμένου παράλληλα με τον ήδη υπάρχοντα κλάδο για την αναγνώριση του κουτιού οριοθέτησης.

2.5.4: Πλεονεκτήματα του Mask-RCNN

- **Απλότητα**: Είναι απλό στην εκπαίδευσή του.
- **Επίδοση**: Το συγκεκριμένο μοντέλο ξεπερνά σε επίδοση όλα τα υπάρχοντα μοντέλα μονής εισόδου σε κάθε τομέα.
- **Αποτελεσματικότητα**: Η μέθοδος αυτή είναι αρκετά αποτελεσματική με ένα μικρό επιπλέον κόστος στον πρόγονό του Faster R-CNN
- **Ευκαμψία**: Το Mask R-CNN είναι εύκολο να γενικευτεί σε άλλες εργασίες. Για παράδειγμα, είναι δυνατό να χρησιμοποιηθεί Mask R-CNN για την εκτίμηση ανθρώπινης πόζας στο ίδιο πλαίσιο.



Εικόνα 18 Το πλαίσιο για τμηματοποίηση παρουσίας του Mask R-CNN

Το κύριο χαρακτηριστικό του Mask R-CNN είναι η ευθυγράμμιση εικονοστοιχείου προς εικονοστοιχείου, η οποία είναι το βασικό κομμάτι που λείπει από το Faster R-CNN. Το Mask R-CNN υιοθετεί την ίδια διαδικασία δύο σταδίων με ένα χαρακτηριστικό πρώτο στάδιο (που είναι το RPN). Στο δεύτερο στάδιο, παράλληλα με την πρόβλεψη της κλάσης και του κουτιού οριοθέτησης, το Mask R-CNN εξάγει μια μάσκα για κάθε RoI (Region of Interest). Αυτό έρχεται σε αντίθεση με τα περισσότερα πρόσφατα συστήματα, όπου η κατηγοριοποίηση βασίζεται στις προβλέψεις μασκών.

Περεταίρω, το Mask R-CNN είναι απλό στην εφαρμογή και στην εκπαίδευσή του που διευκολύνει ένα ευρύ φάσμα ευέλικτων σχεδίων αρχιτεκτονικής. Τέλος, ο κλάδος της μάσκας προσθέτει μόνο ένα μικρό υπολογιστικό κόστος, επιτρέποντας ένα γρήγορο σύστημα και γρήγορο πειραματισμό.[6]

Κεφάλαιο 3: Ανάλυση κώδικα μοντέλων

Στο κεφάλαιο που ακολουθεί γίνεται ανάλυση και επεξήγηση του κώδικα των μοντέλων που χρησιμοποιήθηκαν για την υλοποίηση της πτυχιακής εργασίας. Πριν την ανάλυση του κώδικα μια επεξήγηση σχετικά με αγγλικές ορολογίες που χρησιμοποιούνται παρακάτω.

Ορολογίες που αφορούν το μοντέλο:

- Grid: Είναι ένα πλέγμα το οποίο εφαρμόζει το μοντέλο πάνω στην εικόνα για να ξεκινήσει την διαδικασία εντοπισμού επιθυμητών αντικειμένων.
- Overfitting: Με τον όρο αυτό εννοούμε όταν το μοντέλο μας είναι πάρα πολύ καλά εκπαιδευμένο με το σετ δεδομένων που χρησιμοποιούμε για την εκπαίδευσή του με αποτέλεσμα να μην μπορεί να έχει καλή επίδοση με διαφορετικά δεδομένα. Με λίγα λόγια να μην πετυχαίνουμε την γενίκευση που θα πρέπει να έχει το μοντέλο.
- Layers: Ονομάζονται τα επίπεδα από τα οποία περνάει η πληροφορία κατά την εκπαίδευση του μοντέλου.
- Matrix: Είναι ένας δισδιάστατος πίνακας κλιμάκων με μία ή περισσότερες γραμμές και στήλες.

- Kernel: Στην επεξεργασία εικόνων είναι ένα matrix που μπορεί να χρησιμοποιηθεί για την θόλωση, το ακόνισμα, την ανάγλυφη εκτύπωση, την ανίχνευση άκρων και πολλά άλλα, κάνοντας μια σύμπλεξη μεταξύ ενός matrix και μιας εικόνας.
- Activation functions: Είναι μια λειτουργία που αποφασίζει εάν ένας νευρώνας πρέπει να ενεργοποιηθεί ή όχι. Αυτό σημαίνει ότι θα αποφασίσει εάν η είσοδος του νευρώνα στο δίκτυο είναι σημαντική ή όχι κατά τη διαδικασία πρόβλεψης χρησιμοποιώντας απλούστερες μαθηματικές πράξεις. Ο ρόλος τους είναι να αντλούν έξοδο από ένα σύνολο τιμών εισόδου που τροφοδοτούνται σε ένα νευρωνικό δίκτυο.
- Weights: Μαθησιακές παράμετροι που ορισμένων μοντέλων μηχανικής μάθησης.
- Gradients: Μέτρηση αλλαγής τιμών των βαρών σε σχέση με το σφάλμα.

Ορολογίες που αφορούν το δεδομένα:

- Data_augmentation: Ονομάζεται η μέθοδος με την οποία αυξάνουμε τον αριθμό των δεδομένων που έχουμε. Αυτό γίνεται παίρνοντας μια εικόνα και αλλάζοντας κάποια στοιχεία της όπως είναι το μέγεθος, περιστροφή, κλίμακα ή κάποιες φορές και αλλαγή χρώματος.
- TruePossitive: Έχουμε όταν η πρόβλεψη που κάνει το μοντέλο για κάποιο αντικείμενο είναι ορθή δηλαδή το καταχωρεί σε μια συγκεκριμένη κατηγορία και όντως ανήκει σε αυτή την κατηγορία.
- FalsePossitive: Έχουμε όταν η πρόβλεψη που κάνει το μοντέλο για κάποιο αντικείμενο είναι λανθασμένη.
- Mask: Εντοπίζει τις πιο σημαντικές πτυχές κάθε εισόδου (εικόνας στην συγκεκριμένη πτυχιακή) για πρόβλεψη του δικτύου. Οι μάσκες δημιουργούνται από ένα δευτερεύον δίκτυο, στόχος του οποίου είναι να δημιουργήσει όσο το δυνατόν μικρότερη εξήγηση, διατηρώντας παράλληλα την προγνωστική ακρίβεια του αρχικού δικτύου.
- Dataloader: Οι dataloader βοηθούν στη διαχείριση των δεδομένων διευκολύνοντας έτσι την εκπαίδευση. Όταν εισάγουμε δεδομένα οι dataloader διαβάζουν, εξάγουν και φορτώνουν πληροφορίες για τα αρχεία που επεξεργαζόμαστε κατά την εκπαίδευση.

Ορολογίες που αφορούν την εκπαίδευση:

- Adam: Όνομα αλγορίθμου βελτιστοποίησης μοντέλων.
- Callbacks: Αντικείμενο που ενεργεί σε διάφορες φάσης κατά την εκπαίδευση του μοντέλου.
- Epochs: Αντιπροσωπεύει τον αριθμό του πόσες φορές θα διαβάσει ο αλγόριθμος το σετ δεδομένων.
- Batch_size: Αναφέρεται στον αριθμό των στοιχείων εκπαίδευσης που χρησιμοποιούνται σε μια επανάληψη.

- Learning rate: Είναι μια παράμετρος συντονισμού σε έναν αλγόριθμο βελτιστοποίησης που καθορίζει το βήμα σε κάθε επανάληψη ενώ μειώνεται η συνάρτηση απώλειας.

3.1: Μοντέλο SSD

```
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, TerminateOnNaN, CSVLogger
from keras import backend as K
from keras.models import load_model
from math import ceil
import numpy as np
from matplotlib import pyplot as plt
from models.keras_ssd7 import build_model
from keras_loss_function.keras_ssd_loss import SSDLoss
from keras_layers.keras_layer_AnchorBoxes import AnchorBoxes
from keras_layers.keras_layer_DecodeDetections import DecodeDetections
from keras_layers.keras_layer_DecodeDetectionsFast import DecodeDetectionsFast
from ssd_encoder_decoder.ssd_input_encoder import SSDInputEncoder
from ssd_encoder_decoder.ssd_output_decoder import decode_detections, decode_detections_fast
from data_generator.object_detection_2d_data_generator import DataGenerator
from data_generator.object_detection_2d_misc_utils import apply_inverse_transforms
from data_generator.data_augmentation_chain_variable_input_size import DataAugmentationVariableInputSize
from data_generator.data_augmentation_chain_constant_input_size import DataAugmentationConstantInputSize
from data_generator.data_augmentation_chain_original_ssd import SSDDataAugmentation
from data_generator.object_detection_2d_geometric_ops import Resize
```

Εικόνα 19 Εισαγωγή βιβλιοθηκών

Στην αρχή κάνουμε import τις απαραίτητες βιβλιοθήκες όπως φαίνεται και στην παραπάνω εικόνα. Παρατηρώντας την εικόνα βλέπουμε ότι μια από τις βιβλιοθήκες που χρησιμοποιούμε είναι η keras.

Η συγκεκριμένη είναι μια open-source βιβλιοθήκη που παρέχει στην Python επαφή με τεχνητά νευρωνικά δίκτυα. Επίσης η Keras λειτουργεί ως διεπαφή για την βιβλιοθήκη TensorFlow, η οποία χρησιμοποιείται για εφαρμογές που αφορούν την μηχανική μάθηση. Η keras εμπεριέχει μια πολυάριθμες υλοποιήσεις συνηθισμένων δομικών στοιχείων νευρωνικών δικτύων όπως, layers, objectives, activation functions, optimizers, και μια σειρά από εργαλεία για να διευκολυνθεί η εργασία με δεδομένα εικόνας και κειμένου για να απλοποιηθεί η κωδικοποίηση που είναι απαραίτητη για τη σύνταξη κώδικα νευρωνικού δικτύου.

Από τα δομικά στοιχεία που προαναφέραμε στην συγκεκριμένη εργασία χρησιμοποιήσαμε τα optimizers και callbacks. Αρχικά optimizers είναι κλάσεις που περιέχουν την μέθοδο με την οποία εκπαιδεύεται το μοντέλο μας. Είναι σημαντικό να επιλεχθεί ο κατάλληλος optimizer για το κάθε μοντέλο διότι είναι υπεύθυνοι για την ταχύτητα εκπαίδευσης και απόδοση του μοντέλου.

Στην συγκεκριμένη περίπτωση χρησιμοποιήσαμε τον Adam optimizer. Το πλήρες όνομα είναι Adaptive Moment Estimation Algorithm, ο optimizer παρουσιάστηκε το 2015 από τους

Diederik P. Kingma και Jimmy Lei Ba. Ο συγκεκριμένος αλγόριθμος εκτιμά στιγμές και τις χρησιμοποιεί για την βελτίωση μια συνάρτησης.

Ο αλγόριθμος Adam υπολογίζει έναν εκθετικό weighted μέσο όρο του gradient και στη συνέχεια τετραγωνίζει το υπολογισμένο gradient. Αυτός ο αλγόριθμος έχει δύο παραμέτρους διάσπασης που ελέγχουν τους ρυθμούς αποσύνθεσης αυτών των υπολογισμένων μέσων όρων. Κάποια από τα πλεονεκτήματα του Adam είναι:

- Εύκολος στην εφαρμογή του
- Αρκετά υπολογιστικά αποδοτικός
- Χρησιμοποιεί λίγη μνήμη
- Καλός για μη στατικούς στόχους
- Δουλεύει καλά σε προβλήματα με gradients που έχουν ήχο ή είναι αραιά
- Τέλος είναι αποδοτικός σε μεγάλα σετ δεδομένων και μεγάλες παραμέτρους.

Τα callbacks που χρησιμοποιούμε έχουν ως εξής. Αρχικά μια κλάση που μας επιτρέπει να ορίσουμε που να ελέγξουμε τα weights του μοντέλου, του πως πρέπει να ονομάζεται το αρχείο και υπό ποιες συνθήκες να γίνεται ένα σημείο ελέγχου (checkpoint) του μοντέλου η κλάση αυτή ονομάζεται ModelCheckpoint. Επίσης μας επιτρέπει να καθορίσουμε ποια μέτρηση θα παρακολουθείτε, όπως η απώλεια ή ακρίβεια στο σύνολο δεδομένων εκπαίδευσης ή επικύρωσης.

Επίσης ένα από τα callbacks μας επιτρέπει να ορίσουμε έναν αριθμό epochs που εάν η επίδοση του μοντέλου δεν βελτιώνεται σταματάει την εκπαίδευση (EarlyStopping).

Ακόμη όταν μια συγκεκριμένη παράμετρος παύει να βελτιώνεται για έναν συνεχόμενο αριθμό epochs μειώνουμε το learning_rate του μοντέλου δίχως να σταματάμε την εκπαίδευση του. Αυτό γίνεται μέσω του callback ReduceLROnPlateou.

Επιπλέον ελέγχουμε το loss σε κάθε batch και αν είναι NaN τότε σταματάει η εκπαίδευση και τυπώνεται ο αριθμός του batch που προκάλεσε την συγκεκριμένη τιμή αυτό γίνεται με το callback TerminateOnNaN.

Τέλος έχουμε CSVLogger που όπως φαίνεται και από το όνομα εξάγει όλες τις τιμές που μπορούν να παρουσιαστούν ως σειρά χαρακτήρων σε ένα .csv αρχείο.

Τα backend μηχανικής μάθησης επεξεργάζονται σύνολα δεδομένων και χρησιμοποιούνται για εκπαίδευση, πρόβλεψη και αξιολόγηση μοντέλων.

Τα AnchorBoxes είναι το σύνολο προκαθορισμένων πλαισίων οριοθέτησης ορισμένου ύψους και πλάτους. Αυτά τα πλαίσια έχουν οριστεί για να καταγράφουν την κλίμακα και τον λόγο διαστάσεων συγκεκριμένων κατηγοριών αντικειμένων που θέλουμε να εντοπίσουμε και συνήθως επιλέγονται με βάση τα μεγέθη αντικειμένων στα σύνολα δεδομένων εκπαίδευσης.

Με τον όρο DecodeDetections σημαίνει πως γίνεται αποκωδικοποίηση της αναγνώρισης σε κατανοητή γλώσσα.

Κάνοντας εισαγωγή του SSDInputEncoder το μοντέλο μας μπορεί να μετατρέπει να δεδομένα που εισάγονται σε κώδικα μηχανής που διαβάζει το μοντέλο SSD.

Χάρης του data_generator μας δίνεται η δυνατότητα να γίνει επεξεργασία των δεδομένων όπως είναι data_augmentation και αλλαγή των διαστάσεων τους.

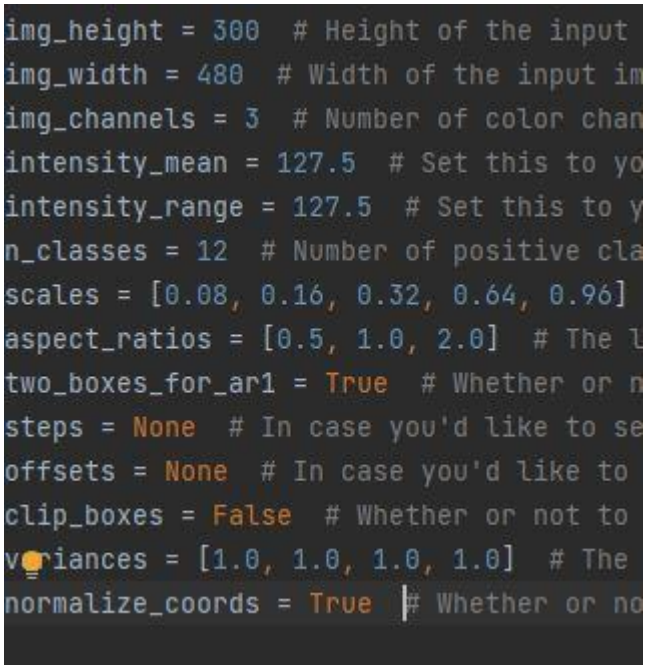
```
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

Εικόνα 20 συναρτήσεις που αποθηκεύουν ορισμένες μετρήσεις

Στην παραπάνω εικόνα δημιουργούμε συναρτήσεις οι οποίες μας βοηθούν να μελετήσουμε την ακρίβεια του μοντέλου. Πιο συγκεκριμένα η recall είναι μια μέτρηση που ποσοτικοποιεί τον αριθμό των TruePositive προβλέψεων από όλες τις positive προβλέψεις που θα μπορούσαν να είχαν γίνει. Η precision επιστρέφει τον αριθμό για το πόσες από τις προβλέψεις που γίνανε είναι σωστές. Τέλος επειδή καμία από τις δύο δεν μπορεί να χρησιμοποιηθεί μόνη της χρειαζόμαστε την f1_m η οποία όπως φαίνεται και στην εικόνα συνδυάζει τις δυο προηγούμενες συναρτήσεις και εξάγει ένα αποτέλεσμα.



Εικόνα 21 Στοιχεία που αφορούν τα δεδομένα που εισάγονται

Στην παραπάνω εικόνα ορίζουμε τα χαρακτηριστικά που θα έχουν τα δεδομένα που εισάγονται. Αρχικά ορίζουμε το ύψος, το πλάτος και τα κανάλια χρωμάτων τις εικόνας. Στη συνέχεια μετατρέπουμε τις τιμές των εικονοστοιχείων στο διάστημα από [-1,1]. Επίσης ορίζουμε τον αριθμό των κατηγοριών που περιέχει το σετ δεδομένων μας. Τα scales και aspect_ratios αφορούν τις κλιμακώσεις των AnchorBoxes. Με την παράμετρο two_boxes_for_ar1 ορίζουμε εάν θέλουμε σε ένα anchor box να υπάρχουν δύο πλαίσια οριοθέτησης (bounding boxes). Το step και offsets αφορούν το grid που εφαρμόζεται πάνω στην εικόνα δηλαδή το step αφορά ανά πόσα κουτάκια του grid θα γίνεται αναζήτηση αντικειμένου και το offsets αφορά το ποσοστό ελέγχου από το μέγεθος του κουτιού του grid. Ορίζοντας τις τιμές σε None αυτή η διαδικασία γίνεται δυναμικά και προσαρμόζονται ανάλογα με το αντικείμενο που εντοπίζεται. Επιπλέον το clip_boxes υπάρχει για το αν θέλουμε να ψαλιδίσουμε τα AnchorBoxes ώστε να υπάρχουν ολοκληρωτικά μέσα στην εικόνα. Τα variances αφορούν την κλιμάκωση των συντεταγμένων από τα στοιχεία που εισάγονται. Με την παράμετρο normalize_coords δηλώνουμε την επιθυμία εάν το μοντέλο είναι να χρησιμοποιήσει συντεταγμένες ανάλογες στο μέγεθος της εικόνας. Τέλος διαγράφουν από την μνήμη προ-υπάρχοντα μοντέλα.

```
model = build_model(image_size=(img_height, img_width, img_channels),
                    n_classes=n_classes,
                    mode='training',
                    l2_regularization=0.0005,
                    scales=scales,
                    aspect_ratios_global=aspect_ratios,
                    aspect_ratios_per_layer=None,
                    two_boxes_for_ar1=two_boxes_for_ar1,
                    steps=steps,
                    offsets=offsets,
                    clip_boxes=clip_boxes,
                    variances=variances,
                    normalize_coords=normalize_coords,
                    subtract_mean=intensity_mean,
                    divide_by_stddev=intensity_range,
                    coords='centroids')
```

Εικόνα 22 Ορισμός παραμέτρων μοντέλου

Στην παραπάνω εικόνα ορίζουμε τις παραμέτρους του μοντέλου όπου χρησιμοποιούμε τις μεταβλητές που ορίσαμε προηγουμένως. Η μεταβλητή l2_regularization είναι μια τεχνική που χρησιμεύει στο να μειωθεί η πιθανότητα να συμβεί overfitting κατά την εκπαίδευση του μοντέλου.

```
from keras.optimizers import Adam, SGD, Adadelta

adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=1e-6)
adade = Adadelta()

ssd_loss = SSDLoss(neg_pos_ratio=3, alpha=1.0)

model.compile(optimizer=adam, loss=ssd_loss.compute_loss, metrics=[precision_m, recall_m, f1_m])

train_dataset = DataGenerator(load_images_into_memory=False, hdf5_dataset_path=None)
val_dataset = DataGenerator(load_images_into_memory=False, hdf5_dataset_path=None)
```

Εικόνα 23 Δημιουργία μοντέλου

Στη συνέχεια ορίζουμε τις παραμέτρους του αλγόριθμου βελτιστοποίησης Adam και δημιουργούμε το μοντέλο μέσω της εντολής compile. Στο τέλος τις εικόνας τα DataGenerator είναι συμπληρωμένα με False και None διότι εισάγουμε εμείς δικές μας εικόνες και επειδή δημιουργούμε δικά μας weights κατά την εκπαίδευση του μοντέλου.

```
# Images
images_dir = r'C:\Users\Nickos_Kal\Desktop\SSD_FLAGS_STABLE\Nick\images'

# Ground truth
train_labels_filename = r'C:\Users\Nickos_Kal\Desktop\SSD_FLAGS_STABLE\Nick\labels_train.csv'
val_labels_filename = r'C:\Users\Nickos_Kal\Desktop\SSD_FLAGS_STABLE\Nick\labels_val.csv'
```

```
classes = ['background', 'car', 'pedestrian', 'trafficLight-Red', 'trafficLight-Green',
           'truck', 'trafficLight', 'biker', 'trafficLight-RedLeft', 'trafficLight-GreenLeft',
           'trafficLight-Yellow', 'trafficLight-YellowLeft']  # Just so we can print class nam
```

Εικόνα 24 Εισαγωγή δεδομένων εκπαίδευσης

Στην συνέχεια βάζουμε την τοποθεσία όπου βρίσκονται να δεδομένα μας και τα αρχεία που εμπεριέχουν τις πληροφορίες τους και τα χαρακτηριστικά τους (xml) και δείχνουμε στο μοντέλο για το ποιες κατηγορίες είναι αυτές που μας ενδιαφέρουν. Την κατηγορία background θα πρέπει να την προσθέσουμε για να μπορεί να ξεχωρίσει το μοντέλο το φόντο από τα αντικείμενα ενδιαφέροντος στην εικόνα.

```
train_dataset.parse_xml(images_dirs=[Dataset_train_images_dir],
                        image_set_filenames=[Dataset_train_image_set_filename],
                        annotations_dirs=[Dataset_train_labels_dir],
                        classes=classes,
                        include_classes='all',
                        exclude_truncated=True,
                        exclude_difficult=True,
                        ret=False)

val_dataset.parse_xml(images_dirs=[Dataset_test_images_dir],
                      image_set_filenames=[Dataset_val_image_set_filename],
                      annotations_dirs=[Dataset_test_labels_dir],
                      classes=classes,
                      include_classes='all',
                      exclude_truncated=True,
                      exclude_difficult=True,
                      ret=False)
```

Εικόνα 25 Μέθοδος parse_xml

Με τη μέθοδο που βλέπουμε την parse_xml το μοντέλο μας αντλεί τις επιθυμητές πληροφορίες από τα αντίστοιχα σετ δεδομένων που έχουμε ορίσει για εκπαίδευση και επαληθέψει.

```
batch_size = 1
# 4: Define the image processing chain.
resize = Resize(height=img_height, width=img_width)
...
predictor_sizes = [model.get_layer('classes4').output_shape[1:3],
                   model.get_layer('classes5').output_shape[1:3],
                   model.get_layer('classes6').output_shape[1:3],
                   model.get_layer('classes7').output_shape[1:3]]
ssd_input_encoder = SSDInputEncoder(img_height=img_height,
                                    img_width=img_width,
                                    n_classes=n_classes,
                                    predictor_sizes=predictor_sizes,
                                    scales=scales,
                                    aspect_ratios_global=aspect_ratios,
                                    two_boxes_for_ar1=two_boxes_for_ar1,
                                    steps=steps,
                                    offsets=offsets,
                                    clip_boxes=clip_boxes,
                                    variances=variances,
                                    matching_type='multi',
                                    pos_iou_threshold=0.5,
                                    neg_iou_limit=0.3,
                                    normalize_coords=normalize_coords)
```

Εικόνα 26 Batch_size και ssd encoder

Στη συνέχεια ορίζουμε τον αριθμό του batch_size που στην προκειμένη περίπτωση βάζουμε 1 επειδή δεν έχουμε αρκετή υπολογιστική ισχύ για παραπάνω. Ορίζουμε πόσο θέλουμε να είναι το resize. Επιλέγουμε τα επίπεδα εξόδων του μοντέλου. Τέλος ορίζουμε τις παραμέτρους του κωδικοποιητή του SSD μοντέλου.

```
# 6: Create the generator handles that will be passed to Keras' `fit_generator()` function.
train_generator = train_dataset.generate(batch_size=batch_size,
                                         shuffle=True,
                                         transformations=[resize],
                                         label_encoder=ssd_input_encoder,
                                         returns={'processed_images',
                                                  'encoded_labels'},
                                         keep_images_without_gt=False)
val_generator = val_dataset.generate(batch_size=batch_size,
                                     shuffle=False,
                                     transformations=[resize],
                                     label_encoder=ssd_input_encoder,
                                     returns={'processed_images',
                                              'encoded_labels'},
                                     keep_images_without_gt=False)
model_checkpoint = ModelCheckpoint(filepath='ssd7_epoch-{epoch:02d}_loss-{loss:.4f}_val_loss-{val_loss:.4f}.h5',
                                   monitor='loss',
                                   verbose=1,
                                   save_best_only=True,
                                   save_weights_only=False,
                                   mode='auto',
                                   period=1)
```

Εικόνα 27 train και val generators

```
csv_logger = CSVLogger(filename='ssd7_training_log.csv',
                       separator=',',
                       append=True)
early_stopping = EarlyStopping(monitor='val_loss',
                               min_delta=0.0,
                               patience=15,
                               verbose=1)
reduce_learning_rate = ReduceLROnPlateau(monitor='val_loss',
                                         factor=0.2,
                                         patience=2,
                                         verbose=1,
                                         epsilon=0.001,
                                         cooldown=0,
                                         min_lr=0.00001)

callbacks = [model_checkpoint,
             csv_logger,
             early_stopping,
             reduce_learning_rate,
             ]

initial_epoch = 0
final_epoch = 50
steps_per_epoch = int(2 * train_dataset_size / batch_size)
```

Εικόνα 28 Δημιουργία των callbacks

Στις παραπάνω εικόνες τα train και val generators δημιουργούν batches δειγμάτων και αντίστοιχες ετικέτες. Υπάρχει η δυνατότητα να ανακατέψει τα δείγματα μετά από κάθε πέρασμα. Προαιρετικά παίρνει μια λίστα με αυθαίρετους μετασχηματισμούς εικόνας για να εφαρμόσει στα δείγματα. Στην προκειμένη περίπτωση έχουμε βάλει μόνο το resize στους μετασχηματισμούς που αλλάζει το τις διαστάσεις της εικόνας. Στη συνέχεια δημιουργούμε τα callbacks που εξηγήσαμε την λειτουργία τους στην αρχή του κεφαλαίου. Τέλος ορίζουμε πόσα epochs θέλουμε και τα βήματα που έχει το καθένα πιο συγκεκριμένα τα βήματα που έχουμε βάλει είναι το διπλάσιο του πηλίκου από το σύνολο των δεδομένων εκπαίδευσης προς το batch_size.

```
history = model.fit_generator(generator=train_generator,
                              steps_per_epoch=steps_per_epoch,
                              epochs=final_epoch,
                              callbacks=callbacks,
                              validation_data=val_generator,
                              validation_steps=ceil(val_dataset_size/batch_size),
                              initial_epoch=initial_epoch)

numpy_loss_history = np.array(history)
model.save("loss_history.txt", numpy_loss_history)

plt.figure(figsize=(20, 12))
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.legend(loc='upper right', prop={'size': 24});

plt.figure(figsize=(20,12))
plt.plot(history.history['acc'], label='acc')
plt.plot(history.history['val_acc'], label='val_acc')
plt.legend(loc='upper right', prop={'size': 24});

dataset_val = DataGenerator()
```

Εικόνα 29 Έναρξη εκπαίδευσης μοντέλου

Με την εντολή fit_generator γίνεται η έναρξη της εκπαίδευσης του μοντέλου μας. Στη συνέχεια φτιάχνουμε μια γραφική παράσταση όπου αναπαριστάτε το loss σε σχέση με το val_loss κατά τη διάρκεια της εκπαίδευσης.

```
Dataset_test_images_dir = r'C:\Users\Nickos_Kal\Desktop\SSD_FLAGS_STABLE\Nick\image_test'
# The directories that contain the annotations.
Dataset_test_labels_dir = r'C:\Users\Nickos_Kal\Desktop\SSD_FLAGS_STABLE\Nick\xml_test'
...
dataset_val.parse_xml(images_dirs=[Dataset_test_images_dir],
                      image_set_filenames=[Dataset_val_image_set_filename],
                      annotations_dirs=[],
                      classes=classes,
                      include_classes='all',
                      exclude_truncated=False,
                      exclude_difficult=True,
                      ret=False)
pred_generator = dataset_val.generate(batch_size=1,
                                      shuffle=True,
                                      transformations=[],
                                      returns={'processed_images',
                                               'filenames',
                                               'inverse_transform',
                                               'original_images',
                                               'original_labels'},
                                      keep_images_without_gt=False)
```

Εικόνα 30 Εισαγωγή των test δεδομένων και δημιουργία του generator

Στη συνέχεια ορίζουμε την τοποθεσία όπου βρίσκονται τα test δεδομένα μας. Με την parse_xml εξάγουμε τις πληροφορίες που μας ενδιαφέρουν και δημιουργούμε τον generator για τις προβλέψεις του μοντέλου.

```
for ii in range(1, 10):
    batch_images, batch_filenames, batch_inverse_transforms, batch_original_images, _ = next(pred_generator)
    i = 0  # Which batch item to look at
    print("Image:", batch_filenames[i])
    print()
    print("Ground truth boxes:\n")
    y_pred = model.predict(batch_images)
    y_pred_decoded = decode_detections(y_pred,
                                       confidence_thresh=0.05,
                                       iou_threshold=0.45,
                                       top_k=200,
                                       normalize_coords=normalize_coords,
                                       img_height=img_height,
                                       img_width=img_width)

    np.set_printoptions(precision=2, suppress=True, linewidth=90)
    print("Predicted boxes:\n")
    print('   class   conf xmin   ymin   xmax   ymax')
    print(y_pred_decoded[i])
    plt.figure(figsize=(20,12))
    plt.imshow(batch_images[i])
    current_axis = plt.gca()
    colors = plt.cm.hsv(np.linspace(0, 1, n_classes+1)).tolist() # Set the colors for the bounding boxes
```

Εικόνα 31 Τύπωση των bounding boxes που βρήκε

```
# Draw the ground truth boxes in green (omit the label for more clarity)
for box in y_pred_decoded[i]:
    xmin = box[1]
    ymin = box[2]
    xmax = box[3]
    ymax = box[4]
    label = '{}'.format(classes[int(box[0])])
    current_axis.add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin, color='green', fill=False, linewidth=
    current_axis.text(xmin, ymin, label, size='x-large', color='white', bbox={'facecolor': 'green', 'alpha': 1.0})

# Draw the predicted boxes in blue
for box in y_pred_decoded[i]:
    xmin = box[-4]
    ymin = box[-3]
    xmax = box[-2]
    ymax = box[-1]
    color = colors[int(box[0])]
    label = '{}: {:.2f}'.format(classes[int(box[0])], box[1])
    current_axis.add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin, color=color, fill=False, linewidth=2)
    current_axis.text(xmin, ymin, label, size='x-large', color='white', bbox={'facecolor': color, 'alpha': 1.0})
plt.show()
```

Εικόνα 32 Αναπαράσταση αυτών πάνω στις αντίστοιχες εικόνες

Τέλος τυπώνουμε τα bounding boxes των εικόνων στην κονσόλα του προγράμματος και κάνουμε παρουσίαση των εικόνων με ζωγραφισμένα τα boxes γύρω από το αντίστοιχο αντικείμενο.

3.2 Μοντέλο YOLOv5

```
def train(hyp,  # path/to/hyp.yaml or hyp dictionary
        opt,
        device,
        callbacks=Callbacks()
        ):
    save_dir, epochs, batch_size, weights, single_cls, evolve, data, cfg, resume, noval, nosave, workers, freeze, = \
        Path(opt.save_dir), opt.epochs, opt.batch_size, opt.weights, opt.single_cls, opt.evolve, opt.data, opt.cfg, \
        opt.resume, opt.noval, opt.nosave, opt.workers, opt.freeze

    # Directories
    w = save_dir / 'weights'  # weights dir
    w.mkdir(parents=True, exist_ok=True)  # make dir
    last, best = w / 'last.pt', w / 'best.pt'

    # Hyperparameters
    if isinstance(hyp, str):
        with open(hyp) as f:
            hyp = yaml.safe_load(f)  # load hyps dict
    LOGGER.info(colorstr('hyperparameters: ') + ', '.join(f'{k}={v}' for k, v in hyp.items()))

    # Save run settings
    with open(save_dir / 'hyp.yaml', 'w') as f:
        yaml.safe_dump(hyp, f, sort_keys=False)
    with open(save_dir / 'opt.yaml', 'w') as f:
        yaml.safe_dump(vars(opt), f, sort_keys=False)
    data_dict = None
```

Εικόνα 33 Συνάρτηση train

Αρχικά δημιουργούμε μια συνάρτηση train η οποία δημιουργεί ένα φάκελο με το όνομα weights όπου αποθηκεύονται δύο αρχεία με τις υπερ-παραμέτρους. Στη συνέχεια φορτώνει τις παραμέτρους αυτές για να της χρησιμοποιήσει κατά την διάρκεια της εκπαίδευσης. Κατόπιν αποθηκεύει αυτά τα αρχεία με το όνομα hyp.yaml και opt.yaml.

```
    # Loggers
    if RANK in [-1, 0]:
        loggers = Loggers(save_dir, weights, opt, hyp, LOGGER)  # loggers instance
        if loggers.wandb:
            data_dict = loggers.wandb.data_dict
            if resume:
                weights, epochs, hyp = opt.weights, opt.epochs, opt.hyp

        # Register actions
        for k in methods(loggers):
            callbacks.register_action(k, callback=getattr(loggers, k))

    # Config
    plots = not evolve  # create plots
    cuda = device.type != 'cpu'
    init_seeds(1 + RANK)
    with torch_distributed_zero_first(RANK):
        data_dict = data_dict or check_dataset(data)  # check if None
    train_path, val_path = data_dict['train'], data_dict['val']
    nc = 1 if single_cls else int(data_dict['nc'])  # number of classes
    names = ['item'] if single_cls and len(data_dict['names']) != 1 else data_dict['names']  # class names
    assert len(names) == nc, f'{len(names)} names found for nc={nc} dataset in {data}'  # check
    is_coco = data.endswith('coco.yaml') and nc == 80  # COCO dataset
```

Εικόνα 34 Loggers και Config

Τα Loggers αφορούν το τι προβάλετε στην κονσόλα όταν τρέχει το πρόγραμμα και τα Config δημιουργούν γραφικές παραστάσεις που χρησιμεύουν στην μετέπειτα ανάλυση των αποτελεσμάτων.

```
# Model
pretrained = weights.endswith('.pt')
if pretrained:
    with torch_distributed_zero_first(RANK):
        weights = attempt_download(weights)  # download if not found locally
    ckpt = torch.load(weights, map_location=device)  # load checkpoint
    model = Model(cfg or ckpt['model'].yaml, ch=3, nc=nc, anchors=hyp.get('anchors')).to(device)  # create
    exclude = ['anchor'] if (cfg or hyp.get('anchors')) and not resume else []  # exclude keys
    csd = ckpt['model'].float().state_dict()  # checkpoint state_dict as FP32
    csd = intersect_dicts(csd, model.state_dict(), exclude=exclude)  # intersect
    model.load_state_dict(csd, strict=False)  # load
    LOGGER.info(f'Transferred {len(csd)}/{len(model.state_dict())} items from {weights}')  # report
else:
    model = Model(cfg, ch=3, nc=nc, anchors=hyp.get('anchors')).to(device)  # create
```

Εικόνα 35 Model

Στην παραπάνω εικόνα δημιουργείται το μοντέλο που θα χρησιμοποιήσουμε. Πιο συγκεκριμένα εάν δεν υπάρχει στον φάκελο που βρίσκεται το script που τρέχουμε τότε κατεβαίνει αυτόματα και δημιουργείται διαφορετικά εάν το μοντέλο υπάρχει ήδη κατεβασμένο τότε απλά παίρνουμε το αρχείο και δημιουργούμε το μοντέλο.

```
# Model parameters
hyp['box'] *= 3. / nl  # scale to layers
hyp['cls'] *= nc / 80. * 3. / nl  # scale to classes and layers
hyp['obj'] *= (imgsz / 640) ** 2 * 3. / nl  # scale to image size and layers
hyp['label_smoothing'] = opt.label_smoothing
model.nc = nc  # attach number of classes to model
model.hyp = hyp  # attach hyperparameters to model
model.class_weights = labels_to_class_weights(dataset.labels, nc).to(device) * nc  # attach class weights
model.names = names

# Start training
t0 = time.time()
nw = max(round(hyp['warmup_epochs'] * nb), 1000)  # number of warmup iterations, max(3 epochs, 1k iterations)
# nw = min(nw, (epochs - start_epoch) / 2 * nb)  # limit warmup to < 1/2 of training
last_opt_step = -1
maps = np.zeros(nc)  # mAP per class
results = (0, 0, 0, 0, 0, 0, 0)  # P, R, mAP@.5, mAP@.5-.95, val_loss(box, obj, cls)
scheduler.last_epoch = start_epoch - 1  # do not move
scaler = amp.GradScaler(enabled=cuda)
stopper = EarlyStopping(patience=opt.patience)
compute_loss = ComputeLoss(model)  # init loss class
LOGGER.info(f'Image sizes {imgsz} train, {imgsz} val\n'
            f'Using {train_loader.num_workers} dataloader workers\n'
            f"Logging results to {colorstr('bold', save_dir)}\n"
            f'Starting training for {epochs} epochs...')
```

Εικόνα 36 Model parameters

Κατόπιν αφού δημιουργηθεί το μοντέλο ορίζονται οι παράμετροί του. Επιπλέον όταν ξεκινάμε την εκπαίδευση αρχίζει να μετράει ένας timer έτσι ώστε να ξέρουμε τον συνολικό χρόνο που έκανε το μοντέλο μας να τελειώσει την εκπαίδευση, ορίζουμε τις παραμέτρους που θέλουμε να μετρήσουμε, εφαρμόζουμε EarlyStopping και τέλος εμφανίζουμε στην κονσόλα μας τα περιεχόμενα του LOGGER τα οποία είναι το μέγεθος των εικόνων, το πόσους dataloader χρησιμοποιούμε, το που φαίνονται τα αποτελέσματα και για το πόσα epochs θα γίνεται η εκπαίδευση.

```
for epoch in range(start_epoch, epochs):  # epoch ------------------------------------------------------------------
    model.train()
    # Update image weights (optional, single-GPU only)
    if opt.image_weights:
        cw = model.class_weights.cpu().numpy() * (1 - maps) ** 2 / nc  # class weights
        iw = labels_to_image_weights(dataset.labels, nc=nc, class_weights=cw)  # image weights
        dataset.indices = random.choices(range(dataset.n), weights=iw, k=dataset.n)  # rand weighted idx
    ...
    mloss = torch.zeros(3, device=device)  # mean losses
    if RANK != -1:
        train_loader.sampler.set_epoch(epoch)
    pbar = enumerate(train_loader)
    LOGGER.info(('\n' + '%10s' * 7) % ('Epoch', 'gpu_mem', 'box', 'obj', 'cls', 'labels', 'img_size'))
    if RANK in [-1, 0]:
        pbar = tqdm(pbar, total=nb)  # progress bar
    optimizer.zero_grad()
    for i, (imgs, targets, paths, _) in pbar:  # batch -------------------------------------------------------------
        ni = i + nb * epoch  # number integrated batches (since train start)
        imgs = imgs.to(device, non_blocking=True).float() / 255.0  # uint8 to float32, 0-255 to 0.0-1.0

        # Warmup
        if ni <= nw:
            xi = [0, nw]  # x interp
            # compute_loss.gr = np.interp(ni, xi, [0.0, 1.0])  # iou loss ratio (obj_loss = 1.0 or iou)
            accumulate = max(1, np.interp(ni, xi, [1, nbs / batch_size]).round())
            for j, x in enumerate(optimizer.param_groups):
                # bias lr falls from 0.1 to lr0, all other lrs rise from 0.0 to lr0
                x['lr'] = np.interp(ni, xi, [hyp['warmup_bias_lr'] if j == 2 else 0.0, x['initial_lr'] * lf(epoch)])
                if 'momentum' in x:
                    x['momentum'] = np.interp(ni, xi, [hyp['warmup_momentum'], hyp['momentum']])
```

Εικόνα 37 Epochs, Logger

Στη συνέχεια η πρώτη if που φαίνεται δεν χρησιμοποιείται για τον λόγω ότι δεν χρησιμοποιούμε κάρτα γραφικών. Επίσης οι πληροφορίες που φαίνονται στην συγκεκριμένη εικόνα είναι ότι φαίνεται στην κονσόλα. Το νούμερο της εποχής που βρίσκεται η εκπαίδευση, οι μετρήσεις που θέλουμε και μια μπάρα που δείχνει την πρόοδο της εποχής.

```
    # Multi-scale
    if opt.multi_scale:
        sz = random.randrange(imgsz * 0.5, imgsz * 1.5 + gs) // gs * gs  # size
        sf = sz / max(imgs.shape[2:])  # scale factor
        if sf != 1:
            ns = [math.ceil(x * sf / gs) * gs for x in imgs.shape[2:]]  # new shape (stretched to gs-multiple)
            imgs = nn.functional.interpolate(imgs, size=ns, mode='bilinear', align_corners=False)
    # Forward
    with amp.autocast(enabled=cuda):
        pred = model(imgs)  # forward
        loss, loss_items = compute_loss(pred, targets.to(device))  # loss scaled by batch_size
        if RANK != -1:
            loss *= WORLD_SIZE  # gradient averaged between devices in DDP mode
        if opt.quad:
            loss *= 4.
    # Backward
    scaler.scale(loss).backward()
    # Optimize
    if ni - last_opt_step >= accumulate:
        scaler.step(optimizer)  # optimizer.step
        scaler.update()
        optimizer.zero_grad()
        if ema:
            ema.update(model)
        last_opt_step = ni

    # Log
    if RANK in [-1, 0]:
        mloss = (mloss * i + loss_items) / (i + 1)  # update mean losses
        mem = f'{torch.cuda.memory_reserved() / 1E9 if torch.cuda.is_available() else 0:.3g}G'  # (GB)
        pbar.set_description(('%10s' * 2 + '%10.4g' * 5) % (
            f'{epoch}/{epochs - 1}', mem, *mloss, targets.shape[0], imgs.shape[-1]))
        callbacks.on_train_batch_end(ni, model, imgs, targets, paths, plots, opt.sync_bn)
    # end batch ------------------------------------------------------------------------------------------------
```

Εικόνα 38 Multi-scale

Στην συνέχεια φαίνεται το πως γίνονται scale οι εικόνες με βάση το μέγεθος που ορίζουμε εμείς. Το forward και backward δεν χρησιμοποιούνται επίσης επειδή δεν χρησιμοποιούμε κάρτα γραφικών.

```
            # Save model
            if (not nosave) or (final_epoch and not evolve):  # if save
                ckpt = {'epoch': epoch,
                        'best_fitness': best_fitness,
                        'model': deepcopy(de_parallel(model)).half(),
                        'ema': deepcopy(ema.ema).half(),
                        'updates': ema.updates,
                        'optimizer': optimizer.state_dict(),
                        'wandb_id': loggers.wandb.wandb_run.id if loggers.wandb else None}

                # Save last, best and delete
                torch.save(ckpt, last)
                if best_fitness == fi:
                    torch.save(ckpt, best)
                del ckpt
                callbacks.on_model_save(last, epoch, final_epoch, best_fitness, fi)

            # Stop Single-GPU
            if stopper(epoch=epoch, fitness=fi):
                break
            ...
        ...
    # end training -----------------------------------------------------------------------------------------------------
```

Εικόνα 39 Save model

Αφού τελειώσει η εκπαίδευση σώζουμε το μοντέλο και κρατάμε το epoch στο οποίο το μοντέλο μας είχε την καλύτερη επίδοση κατά την διάρκεια της εκπαίδευσης.

```
    if RANK in [-1, 0]:
        LOGGER.info(f'\n{epoch - start_epoch + 1} epochs completed in {(time.time() - t0) / 3600:.3f} hours.')
        if not evolve:
            if is_coco:  # COCO dataset
                for m in [last, best] if best.exists() else [last]:  # speed, mAP tests
                    results, _, _ = val.run(data_dict,
                                            batch_size=batch_size // WORLD_SIZE * 2,
                                            imgsz=imgsz,
                                            model=attempt_load(m, device).half(),
                                            iou_thres=0.7,  # NMS IoU threshold for best pycocotools results
                                            single_cls=single_cls,
                                            dataloader=val_loader,
                                            save_dir=save_dir,
                                            save_json=True,
                                            plots=False)
            # Strip optimizers
            for f in last, best:
                if f.exists():
                    strip_optimizer(f)  # strip optimizers
        callbacks.on_train_end(last, best, plots, epoch)
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}")

    torch.cuda.empty_cache()
    return results
```

Εικόνα 40 Τέλος εκπαίδευσης

Τέλος αφού τελειώσει η εκπαίδευση εμφανίζουμε τον συνολικό χρόνο που χρειάστηκε όλη η διαδικασία και τον φάκελο που αποθηκεύτηκαν τα αποτελέσματα.

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default='yolov5s.pt', help='initial weights path')
    parser.add_argument('--data', type=str, default='data/coco128.yaml', help='dataset.yaml path')
    parser.add_argument('--epochs', type=int, default=300)
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
    parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')
    parser.add_argument('--upload_dataset', action='store_true', help='Upload dataset as W&B artifact table')
    parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box image logging interval for W&B')
    parser.add_argument('--save_period', type=int, default=-1, help='Log model after every "save_period" epoch')
    parser.add_argument('--artifact_alias', type=str, default="latest", help='version of dataset artifact to be used')
    parser.add_argument('--local_rank', type=int, default=-1, help='DDP parameter, do not modify')
    parser.add_argument('--freeze', type=int, default=0, help='Number of layers to freeze. backbone=10, all=24')
    parser.add_argument('--patience', type=int, default=30, help='EarlyStopping patience (epochs)')
    parser.add_argument('--workers', type=int, default=8, help='maximum number of dataloader workers')
```

Εικόνα 41 Δήλωση παραμέτρων

Για να εφαρμοστούν όλα τα παραπάνω θα πρέπει να δώσουμε εμείς ορισμένα στοιχεία πριν ξεκινήσει η εκπαίδευση έτσι καλώντας την έναρξη της δίνουμε παράλληλα και το μέγεθος που θέλουμε να έχουν οι εικόνες, το όνομα ενός αρχείο μέσα στο οποίο περιέχονται οι διαδρομές για τους φακέλους με τα στοιχεία εκπαίδευσης και τις κατηγορίες που θέλουμε, το batch_size, τα epochs και τα weights του μοντέλου. Για τις υπόλοιπες παραμέτρους χρησιμοποιούμε τις προκαθορισμένες τιμές.

3.3: Μοντέλο Mask R-CNN

```
from os import listdir
from xml.etree import ElementTree
from numpy import zeros
from numpy import asarray
from numpy import expand_dims
from numpy import mean
from mrcnn.utils import Dataset
from mrcnn.utils import compute_ap
from mrcnn.config import Config
from mrcnn.model import MaskRCNN
from mrcnn.model import mold_image
from mrcnn.model import load_image_gt
from matplotlib import pyplot
from matplotlib.patches import Rectangle
```

Εικόνα 42 Εισαγωγή βιβλιοθηκών

Αρχικά όπως και σε κάθε μοντέλο εισάγουμε τις απαραίτητες βιβλιοθήκες για να μπορέσουμε να χρησιμοποιήσουμε το μοντέλο. Την xml την χρειαζόμαστε επειδή για την εκπαίδευση του μοντέλου είναι απαραίτητο να έχουμε τα annotations των εικόνων εκπαίδευσης δηλαδή τα αρχεία τα οποία περιέχουν τις πληροφορίες που διαβάζει το μοντέλο έτσι ώστε να μπορέσει να μάθει και να κάνει προβλέψεις. Τα αρχεία αυτά έχουν κατάληξη .xml.

```python
# class that defines and loads the udacity dataset
class self_driving_car_mask_r_cnnDataset(Dataset):
    # load the dataset definitions
    def load_dataset(self, dataset_dir, is_train=True):
        # define the clasees
        self.add_class("self_driving_car_mask_r_cnn", 1, "car")
        self.add_class("self_driving_car_mask_r_cnn", 2, "pedestrian")
        self.add_class("self_driving_car_mask_r_cnn", 3, "trafficLight-Red")
        self.add_class("self_driving_car_mask_r_cnn", 4, "trafficLight-Green")
        self.add_class("self_driving_car_mask_r_cnn", 5, "truck")
        self.add_class("self_driving_car_mask_r_cnn", 6, "trafficLight")
        self.add_class("self_driving_car_mask_r_cnn", 7, "biker")
        self.add_class("self_driving_car_mask_r_cnn", 8, "trafficLight-RedLeft")
        self.add_class("self_driving_car_mask_r_cnn", 9, "trafficLight-GreenLeft")
        self.add_class("self_driving_car_mask_r_cnn", 10, "trafficLight-Yellow")
        self.add_class("self_driving_car_mask_r_cnn", 11, "trafficLight-YellowLeft")
        # define data locations
        images_dir = dataset_dir + '/train_img/'
        annotations_dir = dataset_dir + '/train_xml/'
        # find all images
        for filename in listdir(images_dir):
            # extract image id
            image_id = filename.split('_')[0]
            # skip all images after 80% if we are building the train set
            if is_train and int(image_id) >= 1478899823662333255:
                continue
            # skipp all images before 80% if we are building the test/val set
            if not is_train and int(image_id) < 1478899823662333255:
                continue
            img_path = images_dir + filename
            image_id = filename[:-4]
            ann_path = annotations_dir + image_id + '.xml'
            # add to dataset
            self.add_image('self_driving_car_mask_r_cnn', image_id=image_id, path=img_path, annotation=ann_path)
```

Εικόνα 43 Δημιουργία κλάσης για την εκπαίδευση

Στη συνέχεια αρχίζουμε να κατασκευάζουμε μια κλάση η οποία εισάγει το σετ δεδομένων που του δίνουμε. Αντλεί από αυτό τα bounding boxes για την κάθε εικόνα και τέλος εφαρμόζει στις εικόνες που έχουμε ορίσει εμείς για να γίνει αξιολόγηση του μοντέλου το mask που έχει δημιουργηθεί μέσω τις εκπαίδευσης.

Στην παραπάνω εικόνα εμφανίζεται η συνάρτηση load_dataset η οποία είναι αυτή που εισάγει το μοντέλο. Οι παράμετροι που παίρνει είναι το σετ δεδομένων και αν το is_Train είναι αληθής ή όχι. Την παράμετρο is_Train πρέπει να την έχουμε ως αληθή έτσι ώστε να μπορεί το μοντέλο να αναγνωρίσει ποιες εικόνες είναι για εκπαίδευση και ποιες για αξιολόγηση. Επίσης αυτό που κάνουμε μέσω αυτής της συνάρτησης είναι να δηλώνουμε ποιες είναι οι κατηγορίες των αντικειμένων που μας ενδιαφέρουν στις εικόνες. Τέλος του δηλώνουμε μέχρι ποια εικόνα μπορεί να πάρει για εκπαίδευση και οι υπόλοιπες είναι για αξιολόγηση.

```python
# load all bounding boxes for an image:
def extract_boxes(self, filename):
    # load and parse the file
    tree = ElementTree.parse(filename)
    root = tree.getroot()
    boxes = list()
    # extract each bounding box
    for box in root.findall('.//object'):
        name = box.find('name').text
        xmin = int(box.find('./bndbox/xmin').text)
        ymin = int(box.find('./bndbox/ymin').text)
        xmax = int(box.find('./bndbox/xmax').text)
        ymax = int(box.find('./bndbox/ymax').text)
        coors = [xmin, ymin, xmax, ymax, name]
        if name == 'car' or name == 'pedestrian' or name == 'trafficLight-Red' \
                or name == 'trafficLight-Green' or name == 'truck' or name == 'trafficLight' \
                or name == 'biker' or name == 'trafficLight-RedLeft' or name == 'trafficLight-GreenLeft' \
                or name == 'trafficLight-Yellow' or name == 'trafficLight-YelloLeft':
            boxes.append(coors)
    # extract image dimensions
    width = int(root.find('.//size/width').text)
    height = int(root.find('.//size/height').text)
    return boxes, width, height
```

Εικόνα 44 Συνάρτηση extract_boxes

Ο ρόλος της συνάρτησης που φαίνεται στην παραπάνω εικόνα είναι να αντλεί από το annotation της κάθε εικόνας το bounding box. Τέλος ότι αποτελέσματα βγάλει τα αποθηκεύει μέσα σε μια λίστα και στο τέλος επιστρέφει την λίστα αυτή μαζί με τις διαστάσεις της εικόνας για να χρησιμοποιηθούν μετά στην εκπαίδευση και στην πρόβλεψη.

```python
def load_mask(self, image_id):
    # get details of image
    info = self.image_info[image_id]
    # define box file location
    path = info['annotation']
    # load XML
    boxes, w, h = self.extract_boxes(path)
    # create one array for all masks, each on a different channel
    masks = zeros([h, w, len(boxes)], dtype='uint8')
    # create masks
    class_ids = list()
    for i in range(len(boxes)):
        box = boxes[i]
        row_s, row_e = box[1], box[3]
        col_s, col_e = box[0], box[2]
        if (box[4] == 'car'):
            masks[row_s:row_e, col_s:col_e, i] = 1
            class_ids.append(self.class_names.index('car'))

        elif (box[4] == 'pedestrian'):
            masks[row_s:row_e, col_s:col_e, i] = 2
            class_ids.append(self.class_names.index('pedestrian'))

        elif (box[4] == 'trafficLight-Red'):
            masks[row_s:row_e, col_s:col_e, i] = 3
            class_ids.append(self.class_names.index('trafficLight-Red'))

        elif (box[4] == 'trafficLight-Green'):
            masks[row_s:row_e, col_s:col_e, i] = 4
            class_ids.append(self.class_names.index('trafficLight-Green'))

        elif (box[4] == 'truck'):
            masks[row_s:row_e, col_s:col_e, i] = 5
            class_ids.append(self.class_names.index('truck'))
```

Εικόνα 45 Συνάρτηση load_mask

```
            elif (box[4] == 'trafficLight'):
                masks[row_s:row_e, col_s:col_e, i] = 6
                class_ids.append(self.class_names.index('trafficLight'))

            elif (box[4] == 'biker'):
                masks[row_s:row_e, col_s:col_e, i] = 7
                class_ids.append(self.class_names.index('biker'))

            elif (box[4] == 'trafficLight-RedLeft'):
                masks[row_s:row_e, col_s:col_e, i] = 8
                class_ids.append(self.class_names.index('trafficLight-RedLeft'))

            elif (box[4] == 'trafficLight-GreenLeft'):
                masks[row_s:row_e, col_s:col_e, i] = 9
                class_ids.append(self.class_names.index('trafficLight-GreenLeft'))

            elif (box[4] == 'trafficLight-Yellow'):
                masks[row_s:row_e, col_s:col_e, i] = 10
                class_ids.append(self.class_names.index('trafficLight-Yellow'))

            elif (box[4] == 'trafficLight-YellowLeft'):
                masks[row_s:row_e, col_s:col_e, i] = 11
                class_ids.append(self.class_names.index('trafficLight-YellowLeft'))
        return masks, asarray(class_ids, dtype='int32')
```

Εικόνα 46 Συνάρτηση load_mask

Στις παραπάνω εικόνες φαίνεται η διαδικασία που ακολουθείτε για να εφαρμόζει το μοντέλο το αντίστοιχο mask σε κάθε αντικείμενο που αναγνωρίζει ανάλογα με την κατηγορία που ανήκει.

```python
# define a configuration for the model
class self_driving_car_mask_r_cnnConfig(Config):
    # define the name of the configuration
    NAME = "self_driving_car_mask_r_cnn_cfg"
    # number of classes (background + objects)
    NUM_CLASSES = 1 + 11
    # number of training steps per epoch
    STEPS_PER_EPOCH = 160


# prepare train set
train_set = self_driving_car_mask_r_cnnDataset()
train_set.load_dataset('self_driving_car_mask_r_cnn', is_train=True)
train_set.prepare()
print('Train: %d' % len(train_set.image_ids))
# prepare test/val set
test_set = self_driving_car_mask_r_cnnDataset()
test_set.load_dataset('self_driving_car_mask_r_cnn', is_train=False)
test_set.prepare()
print('Test: %d' % len(test_set.image_ids))
# prepare config
config = self_driving_car_mask_r_cnnConfig()
config.display()
# define the model
model = MaskRCNN(mode='training', model_dir='./', config=config)
# load weights (mscoco) and exclude the output layers
model.load_weights(r'C:\Users\Nickos_Kal\PycharmProjects\Mask_R_CNN\Mask_RCNN\mask_rcnn_coco.h5', by_name=True,
                   exclude=["mrcnn_class_logits", "mrcnn_bbox_fc", "mrcnn_bbox", "mrcnn_mask"])
# train weights (output layers or 'heads')
model.train(train_set, test_set, learning_rate=config.LEARNING_RATE, epochs=150, layers='heads')
```

Εικόνα 47 Κλάση Config και έναρξη εκπαίδευσης

Τα αρχεία config είναι αρχεία τα οποία εξηγούν πως να διαβάζονται τα δεδομένα, τι μοντέλο να χρησιμοποιηθεί, πως να εκπαιδευτεί το μοντέλο, και πως να αξιολογηθεί η απόδοσή του δίχως να υπάρχει αλληλεπίδραση με τον χρήστη. Στην παραπάνω εικόνα χρησιμοποιούμε την κλάση config για να δημιουργήσουμε ένα αρχείο το οποίο ενημερώνει το μοντέλο για το πόσες κατηγορίες έχουμε και επίσης από πόσα βήματα να αποτελείται το κάθε epoch. Κοιτώντας των αριθμό των κατηγοριών παρατηρούμε ότι προσθέτουμε μια επιπλέον κατηγορία πέρα από αυτές που έχουμε. Αυτό γίνεται για να λαμβάνετε υπόψιν και το φόντο της εικόνας. Στη συνέχεια εφαρμόζοντας τις κλάσεις που δημιουργήσαμε ξεκινάμε την εκπαίδευση του μοντέλου μας.

```python
# define the prediction configuration
class PredictionConfig(Config):
    # define the name of the configuration
    NAME = 'predict_self_driving_car_cfg'
    # number of classes (background + object)
    NUM_CLASSES = 1 + 11
    # simplify GPU config
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
```

Εικόνα 48 Predict Config

Κατόπιν αφού τελειώσει η εκπαίδευση δημιουργούμε ένα ακόμη αρχείο config αλλά αυτή την φορά το αρχείο αυτό είναι υπεύθυνο για τις προβλέψεις που θα κάνει το μοντέλο δηλαδή του δίνουμε ξανά τις κατηγορίες, το πόση γραφική μνήμη από την κάρτα γραφικών του υπολογιστή μας να χρησιμοποιήσει και το πόσες εικόνας να επεξεργάζεται κάθε φορά. Βάζοντας την μεταβλητή GPU_COUNT ίση με ένα χρησιμοποιούμε τη μέγιστη δυνατόν μνήμη της κάρτας μας.

```
# calculate the mAP for a model on a given dataset
def evaluate_model(dataset, model, cfg):
    APs = list()
    for image_id in dataset.image_ids:
        # load image, bounding boxes and masks for the image id
        image, image_meta, gt_class_id, gt_bbox, gt_mask = load_image_gt(dataset, cfg, image_id, use_mini_mask=False)
        # convert pixel values (e.g. center)
        scaled_image = mold_image(image, cfg)
        # convert image into one sample
        sample = expand_dims(scaled_image, 0)
        # make prediction
        yhat = model.detect(sample, verbose=0)
        # extract results for first sample
        r = yhat[0]
        # calculate statistics, including AP
        AP, _, _, _ = compute_ap(gt_bbox, gt_class_id, gt_mask, r["rois"], r["class_ids"], r["scores"], r['masks'])
        # store
        APs.append(AP)
    # calculate the mean AP across all images
    mAP = mean(APs)
    return mAP
```

Εικόνα 49 Συνάρτηση αξιολόγησης μοντέλου

Στην παραπάνω εικόνα δημιουργούμε μια συνάρτηση η οποία υπολογίζει το mean Absolute Precision (mAP). Η μεταβλητή αυτή είναι ένας αριθμός ο οποίος μας δείχνει σε κλίματα τις εκατό τον μέσο όρο για το πόσο ακριβής είναι το μοντέλο στις προβλέψεις του.

```python
# plot a number of photos with ground truth and predictions
def plot_actual_vs_predicted(dataset, model, cfg, n_images=5):
    # load image and mask
    for i in range(n_images):
        # load the image and mask
        image = dataset.load_image(i)
        mask, _ = dataset.load_mask(i)
        # convert pixel values (e.g. center)
        scaled_image = mold_image(image, cfg)
        # convert image into one sample
        sample = expand_dims(scaled_image, 0)
        # make prediction
        yhat = model.detect(sample, verbose=0)[0]
        # define subplot
        pyplot.subplot(n_images, 2, i * 2 + 1)
        # plot masks
        for j in range(mask.shape[2]):
            pyplot.imshow(mask[:, :, j], cmap='gray', alpha=0.3)
        # get the context for drawing boxes
        pyplot.subplot(n_images, 2, i * 2 + 2)
        # plot raw pixel data
        pyplot.imshow(image)
        pyplot.title('Predicted')
        ax = pyplot.gca()
        # plot each box
        for box in yhat['rois']:
            # get coordinates
            y1, x1, y2, x2 = box
            # calculate width and height of the box
            width, height = x2 - x1, y2 - y1
            # create the shape
            rect = Rectangle((x1, y1), width, height, fill=False, color='red')
            # draw the box
            ax.add_patch(rect)
    # show the figure
    pyplot.show()
```

Εικόνα 50 Συνάρτηση παρουσίασης προβλέψεων

Στην παραπάνω εικόνα δημιουργούμε μια συνάρτηση η οποία καλείται στον τέλος του κώδικα και μας παρουσιάζει τις προβλέψεις που έκανε το μοντέλο ζωγραφίζοντας γύρω από τα αντικείμενα που εντόπισε ένα σχήμα με το όνομα της κατηγορίας που το έχει εντάξει.

```python
# prepare train set
train_set = self_driving_car_mask_r_cnnDataset()
train_set.load_dataset('self_driving_car_mask_r_cnn', is_train=True)
train_set.prepare()
print('Train: %d' % len(train_set.image_ids))
# prepare test/val set
test_set = self_driving_car_mask_r_cnnDataset()
test_set.load_dataset('self_driving_car_mask_r_cnn', is_train=False)
test_set.prepare()
print('Test: %d' % len(test_set.image_ids))
# create config
cfg = PredictionConfig()
# define the model
model = MaskRCNN(mode='inference', model_dir='./', config=cfg)
# load model weights
model_path = '/content/Mask_RCNN/udacity_cfg20210806T1649/mask_rcnn_udacity_cfg_0150.h5'
model.load_weights(model_path, by_name=True)
# evaluate model on training dataset
train_mAP = evaluate_model(train_set, model, cfg)
print("Train mAP: %.3f" % train_mAP)
# evaluate model on test dataset
test_mAP = evaluate_model(test_set, model, cfg)
print("Test mAP: %.3f" % test_mAP)
# plot predictions for train dataset
plot_actual_vs_predicted(train_set, model, cfg)
# plot predictions for test dataset
plot_actual_vs_predicted(test_set, model, cfg)
```

Εικόνα 51 Τελικό στάδιο του κώδικα

Τέλος καλούμε όλες τις συναρτήσεις, που αναφέραμε ότι χρησιμοποιούνται για την αξιολόγηση και τις προβλέψεις του μοντέλου, βάζοντας τις ανάλογες παραμέτρους για την κάθε μια.

Κεφάλαιο 4: Αποτελέσματα

Κατά την εκπαίδευση των μοντέλων και από το σετ δεδομένων το ογδόντα τις εκατό (80%) των εικόνων που αντιστοιχεί σε δεκαοχτώ χιλιάδες (18000) χρησιμοποιήθηκε για να εκπαιδευτεί το μοντέλο και το υπόλοιπο είκοσι τις εκατό (20%) που αντιστοιχεί σε τέσσερεις χιλιάδες διακόσες σαράντα μια (4241) εικόνες.

Επίσης πριν ξεκινήσουμε την ανάλυση των αποτελεσμάτων θα πρέπει να αναφέρουμε τι σημαίνουν ορισμένες ορολογίες:

- Recall: Είναι ο αριθμός των True Positives προς το σύνολο των True Positives με τα False Negatives. Με άλλα λόγια είναι ο αριθμός των σωστών προβλέψεων που έκανε το μοντέλο.
- Precision: Είναι ο αριθμός των True Positives προς το σύνολο των True Positives με False Positives. Με άλλα λόγια είναι ο αριθμός που αντιπροσωπεύει πόσες από τις συνολικές προβλέψεις ήταν σωστές.
- Confidence: Είναι ένας τρόπος για να υπολογίσουμε την αβεβαιότητα του μοντέλου για μια πρόβλεψη που κάνει.
- F1_score: Είναι μια μέτρηση που αφορά την ακρίβεια του μοντέλου και υπολογίζεται από τον τύπο 2*(precision * recall) / (precision + recall).
- Loss: Η μεταβλητή αυτή αντιπροσωπεύει το πόσο καλές είναι οι προβλέψεις του μοντέλου κατά την διάρκεια της εκπαίδευσης όσο πιο κοντά στο μηδέν (0) βρίσκεται αυτή η τιμή τόσο πιο ακριβές είναι.
- Val_loss: Όπως και με το loss έτσι και το val_loss αντιπροσωπεύει την ακρίβεια του μοντέλου μας αλλά αυτή την φόρα στο dataset που του έχουμε εισάγει για επαλήθευση.

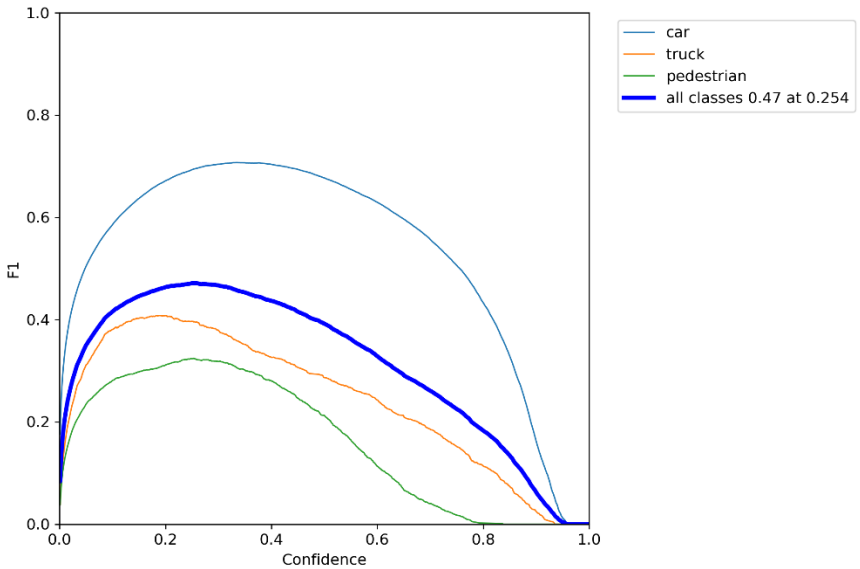
4.1: Αποτελέσματα YOLOv5

```
     99/99        0G   0.03313   0.01672  0.003223       118       256: 100%|
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%
 Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|
]
                 all       4239      36049      0.642      0.396      0.427      0.225
                 car       4239      29748      0.722      0.669      0.709      0.414
               truck       4239       1834      0.747      0.269      0.329      0.181
          pedestrian       4239       4467      0.458       0.25      0.242     0.0803

100 epochs completed in 95.828 hours.
Optimizer stripped from runs\train\exp5\weights\last.pt, 14.4MB
Optimizer stripped from runs\train\exp5\weights\best.pt, 14.4MB
```

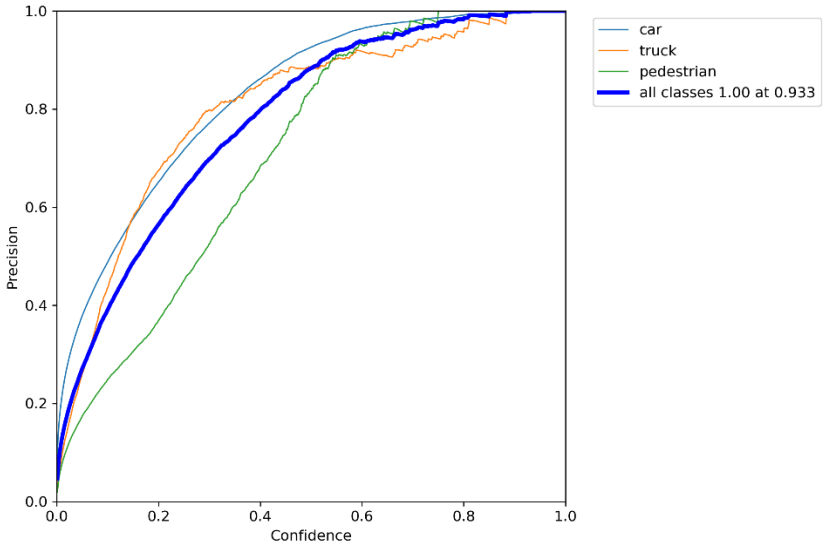
Εικόνα 52 Τέλος εκπαίδευσης

Αρχικά όταν τελειώνει η εκπαίδευση βλέπουμε μια ανάλυση σχετικά με τα αποτελέσματα τις εκπαίδευσης. Παρατηρούμε πως επειδή δεν χρησιμοποιήσαμε κάρτα γραφικών το μοντέλο χρειάστηκε ενενήντα πέντε ώρες (95) για να ολοκληρώσει την όλη διαδικασία. Η λίστα με όνομα class περιέχει τα ονόματα των κατηγοριών που δώσαμε στο μοντέλο και ακριβώς δίπλα βλέπουμε τον αριθμό των εικόνων που χρησιμοποίησε για επικύρωση έτσι ώστε να μπορέσουμε να δούμε μετά την εκπαίδευση τις προβλέψεις του μοντέλου. Στην λίστα Labels βλέπουμε πόσα αντικείμενα που βρήκε στην κάθε εικόνα από την κάθε κατηγορία.



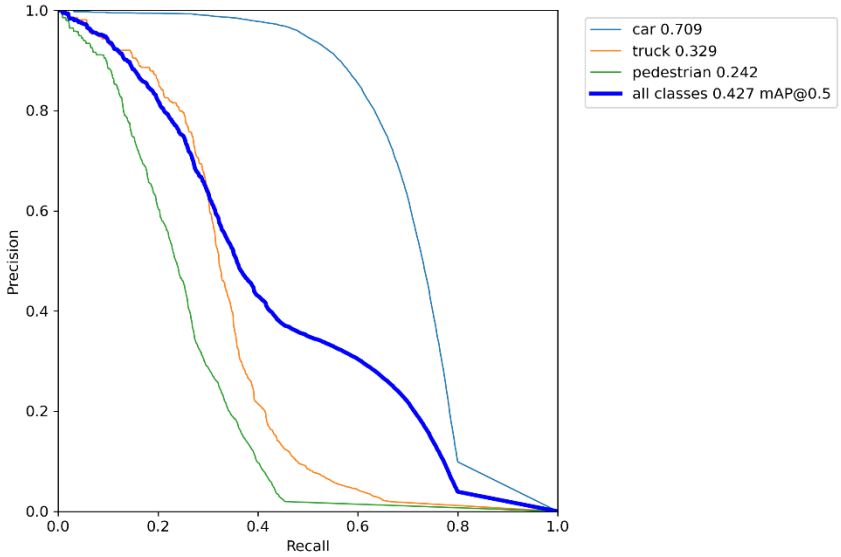
Εικόνα 53 Γραφική F1 με Confidence

Στην παραπάνω γραφική παρουσιάζεται η εξέλιξη του F1_score σε σχέση με το Confidence του μοντέλου για την κάθε κατηγορία. Όπως φαίνεται όσο το confidence αυξανόταν το F1_score μειωνότανε.



Εικόνα 54 Confidence – Precision

Στην παραπάνω γραφική παρατηρούμε πως όσο το confidence του μοντέλο αυξανόταν τόσο ανέβαινε και το precision που σημαίνει πως το σύνολο των σωστών προβλέψεων αυξάνονταν.



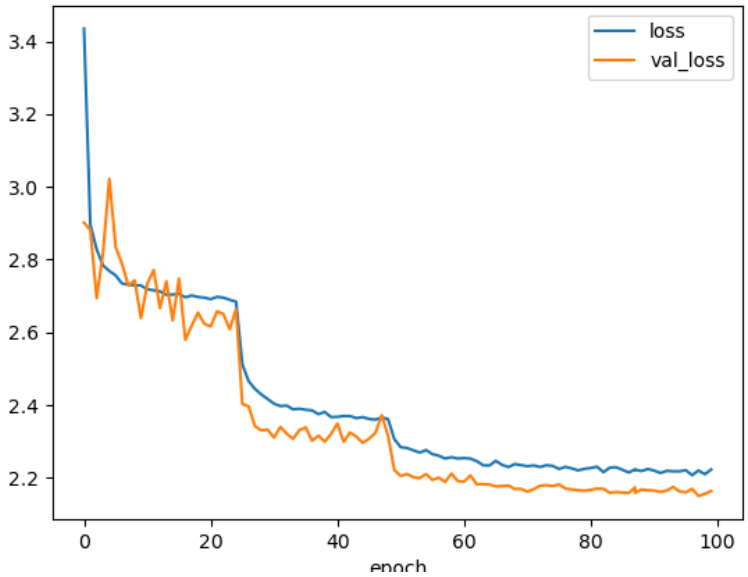
Εικόνα 55 Recall-Precision

Η παραπάνω γραφική αντιπροσωπεύει την σχέση μεταξύ του Recall και του Precision του μοντέλου μετά την εκπαίδευση. Σύμφωνα με αυτήν βλέπουμε ότι με την πάροδο της εκπαίδευσης το Recall αυξανόταν πράγμα το οποίο είναι θετικό. Διότι όσο μεγαλύτερο Recall έχουμε τόσο μικρότερο είναι το False negative.

Εικόνα 56 Yolo predictions

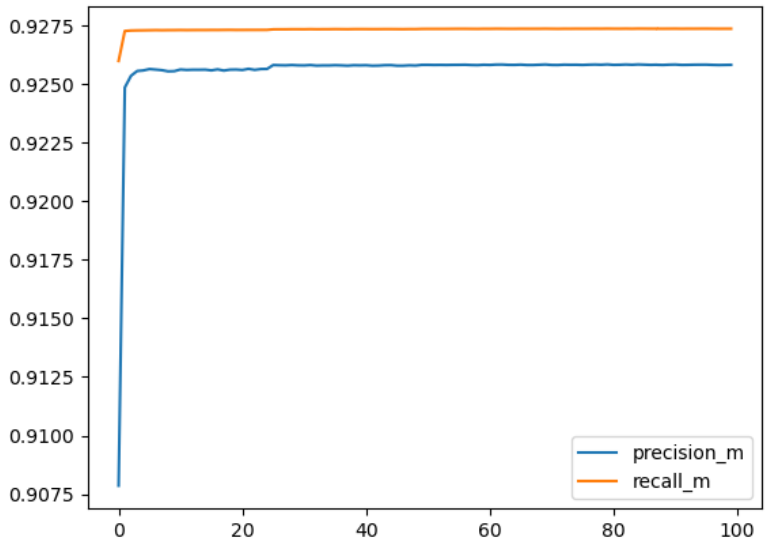
Στην παραπάνω εικόνα βλέπουμε τις προβλέψεις που έκανε το μοντέλο μας μετά την εκπαίδευση. Όπως παρατηρείται οι προβλέψεις που έχουν γίνει είναι αρκετά ακριβείς ακόμα και την κατηγορία των πεζών για τους οποίους έχουμε ένα πολύ μικρό ποσοστό δειγμάτων σε σχέση με τα αυτοκίνητα στο συγκεκριμένο dataset.

4.2: Αποτελέσματα SSD



Εικόνα 57 loss_val_loss

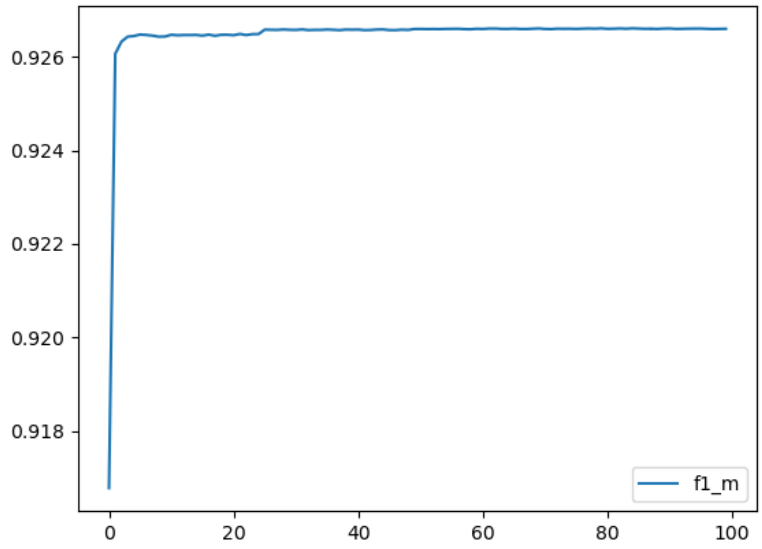
Στην παραπάνω εικόνα παρατηρούμε την γραφική που αντιπροσωπεύει τις τιμές και την βελτίωση των loss και val_loss. Μέσω της γραφικής συμπεραίνουμε πως το μοντέλο μας έχει περιθώρια βελτίωσης αλλά λόγω της έλλειψης υλικού εξοπλισμού είναι το βέλτιστο αποτέλεσμα με που μπορούμε να έχουμε. Όπως φαίνεται και οι δύο τιμές μετά από εκατό (100) epochs καταλήγουν να έχουν τιμή περίπου δύο κόμμα ένα (2.1) με δύο κόμμα τρια (2.3).



Εικόνα 58 Precision_recall

Στην συνέχεια μελετάμε το precision και το recall του μοντέλου μας όπως φαίνεται από την γραφική και τα δύο έχουν αρκετά καλές τιμές και αυτό συμβαίνει επειδή το σετ δεδομένων που έχουμε η κατηγορία των αμαξιών υπερτερεί σε μεγάλο ποσοστό σε σχέση με τις υπόλοιπες

κατηγορίες με αποτέλεσμα η παραπάνω γραφική να επηρεάζεται αρκετά από την συγκεκριμένη κατηγορία και για αυτό τον λόγο να παίρνουμε τόσο καλές τιμές.



Εικόνα 59 f1

Στην παραπάνω γραφική φαίνεται η εξέλιξη της τιμής του f1 και εφόσον είδαμε πως το precision και το recall είναι αρκετά κοντά στο ένα (1) έτσι και αυτό είναι αρκετά κοντά στην βέλτιστη τιμή που μπορεί να έχει.

Εικόνα 60 Αποτελέσματα SSD

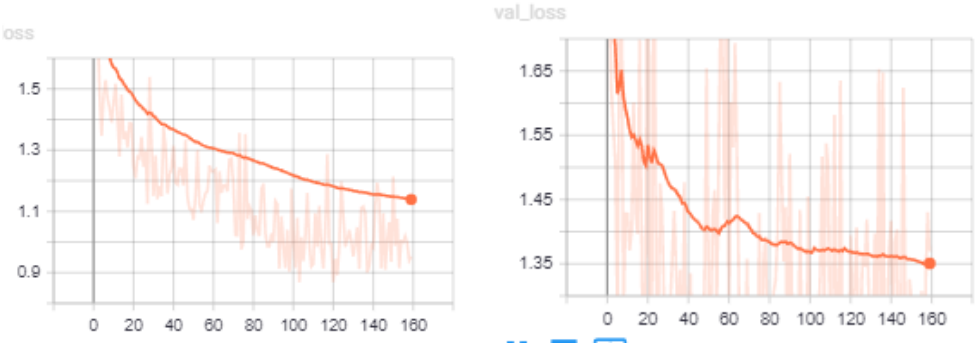
Στην παραπάνω εικόνα μπορούμε να παρατηρήσουμε πως το μοντέλο μας δεν έχει καταφέρει να αναγνωρίσει τους πεζούς που υπάρχουν μέσα σε αυτήν. Αυτό συμβαίνει για τους εξής λόγους. Αρχικά επειδή τα loss και val_loss δεν είναι κοντά στο μηδέν (0) αυτό σημαίνει πως το μοντέλο μας δεν είναι απόλυτα ακριβές με αποτέλεσμα να μην μπορεί να εντοπίσει όλες τις κατηγορίες το ίδιο αποτελεσματικά. Επιπρόσθετα όπως έχουμε εξηγήσει και προηγουμένως τα αμάξια υπερτερούν σε σχέση με τις υπόλοιπες κατηγορίες και αυτό έχει ως αποτέλεσμα το μοντέλο μας να εντοπίζει τα αμάξια με μεγαλύτερη ευκολία απ' ότι τις υπόλοιπες κατηγορίες γι' αυτό τον λόγο βλέπουμε πως είναι το confidence (σιγουριά) πως το αντικείμενο που έχει εντοπίσει είναι ίση με ένα (1) που είναι το μέγιστο ποσοστό που μπορεί να έχει.

Εικόνα 61 Αποτελέσματα SSD

Η παραπάνω εικόνα αποτελεί το καλύτερο παράδειγμα που θα μπορούσαμε να δώσουμε σχετικά με το γεγονός ότι το μοντέλο έχει μάθει να διακρίνει τα αμάξια αρκετά καλύτερα από τις υπόλοιπες κατηγορίες. Όπως παρατηρείται το confidence για το ότι υπάρχει φανάρι είναι μηδέν κόμμα έξι (0.6). Ενώ για το ότι υπάρχουν αμάξια στην εικόνα μπορεί να μας το δείξει με μεγαλύτερη σιγουριά.

4.3 Αποτελέσματα Mask R-CNN

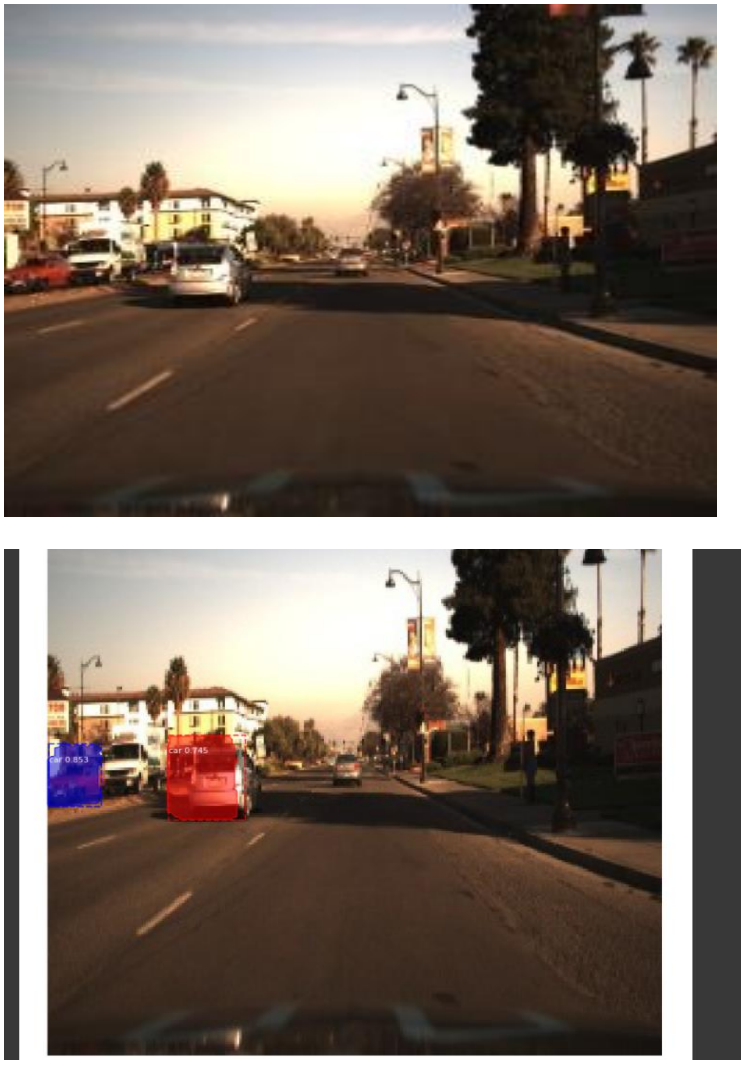
Σε αυτό το σημείο θα ήθελα να αναφέρω πως το συγκεκριμένο μοντέλο δεν γινόταν να εκπαιδευτεί στον υπολογιστή μου ,λόγω έλλειψης κάρτας γραφικών της μάρκας Nvidia, η εκπαίδευσή του έγινε στην πλατφόρμα της Google το Google Colab με αποτέλεσμα ο χρόνος που είχα στην διάθεσή μου για να κάνω την εκπαίδευση ήταν περιορισμένος λόγω του χρονικού ορίου χρήσης ψηφιακής κάρτας γραφικών που έχει επιβάλει η Google στην συγκεκριμένη πλατφόρμα.



Εικόνα 62 Γραφήματα loss και val_loss

Στις παραπάνω γραφικές παραστάσεις απεικονίζονται τα δύο γραφήματα που αφορούν το loss και το val_loss του μοντέλου κοιτάζοντάς τα παρατηρούμε ότι το loss κατέληξε να είναι λίγο κάτω από ένα σε ορισμένα epochs κάτι το οποίο θα σήμαινε πως η εκπαίδευση εξελισσόταν αρκετά καλά. Ομοίως και για το val_loss παρατηρούμε πως η τιμή του κατά το τέλος της εκπαίδευσης κυμαίνονταν στο ένα κόμμα τριάντα πέντε (1.35). Από της δύο γραφικές μπορούμε να καταλάβουμε

πως σε σχέση με τα άλλα δύο μοντέλα οι τιμές για αυτό τα δύο είναι πολύ καλύτερες και αυτό μας προδιαθέτει στο να περιμένουμε αρκετά καλύτερα αποτελέσματα.



Εικόνα 63 Αποτελέσματα Mask_r-cnn

Όπως παρατηρείται στην παραπάνω εικόνα τα αποτελέσματα δεν είναι τόσο ικανοποιητικά όσο θα περίμενε κανείς το μοντέλο μπορεί να έχει καταφέρει να εντοπίσει δύο αντικείμενα στην συγκεκριμένη εικόνα όμως βλέπουμε ότι υπάρχουν τέσσερα αντικείμενα της ίδιας κατηγορίας. Αυτό μας δείχνει ότι το μοντέλο χρειάζεται παραπάνω εκπαίδευση ή ότι θα πρέπει να ξανά εκπαιδεύσουμε το μοντέλο από την αρχή αλλά με εικόνες καλύτερης ανάλυσης διότι όπως είναι φανερό η ανάλυση των εικόνων έχει χαλάσει αρκετά με αποτέλεσμα τα αντικείμενα που υπάρχουν μέσα σε αυτές να αλλοιώνονται. Δυστυχώς κάτι τέτοιο δεν ήταν εφικτό να γίνει διότι με εικόνες μεγαλύτερου μεγέθους το μοντέλο μας χρειαζόταν περισσότερο χρόνο εκπαίδευσης κάτι για το οποίο δεν είχαμε την πολυτέλεια λόγω του περιορισμένου χρόνου που είχαμε στην κατοχή μας.

Κεφάλαιο 5: Συμπεράσματα και Βελτιώσεις

Από τα αποτελέσματα που έχουμε πάρει συμπεραίνουμε ότι και τα τρία μοντέλα Single Shot Detection, Yolo και Mask R-CNN για την υλοποίηση της διπλωματικής άσκησης είναι αρκετά ικανοποιητικά. Παρόλα αυτά είναι φανερό πως υπάρχει χώρος για βελτίωση εκπαιδεύοντας παραπάνω τα μοντέλα και με περισσότερα δεδομένα. Ωστόσο αυτό προϋποθέτει μεγαλύτερο και καλύτερο εξοπλισμό. Ακόμη επειδή τα αντικείμενα ενδιαφέροντος είναι διασκορπισμένα στις εικόνες και επειδή έχουν διαφορετικά μεγέθη, για παράδειγμα το αμάξι σε σχέση με το φανάρι, έχει ως αποτέλεσμα όταν μικραίνουμε τις εικόνες για να μειώσουμε τον χρόνο εκπαίδευσης που απαιτείται αυτή η διαφορά μεγέθους να μεγιστοποιείται και αυτό έχει ως αποτέλεσμα ορισμένα αντικείμενα ενδιαφέροντος να είναι δύσκολο να αναγνωριστούν από τα μοντέλα. Αυτό μας οδηγεί στο συμπέρασμα πως χρειαζόμαστε εικόνες μεγαλύτερου μεγέθους έτσι ώστε τα αντικείμενα προς αναγνώριση να είναι ευδιάκριτα. Επιπλέον υπάρχουν ορισμένες μελλοντικές αναβαθμίσεις που θα μπορούσαν να γίνουν έτσι ώστε να γίνουν πιο προσιτά για να μπορεί να τα χρησιμοποιήσει και κάποιος ο οποίος δεν έχει τόσο καλό εξοπλισμό όσο απαιτούν αυτή την στιγμή τα μοντέλα αναγνώρισης αντικειμένων.

Αρχικά όλα τα μοντέλα δεν έχουν μεγάλη ανοχή (invariance) και αυτό συμβαίνει, επειδή το βάρη τους δημιουργούνται τυχαία, οπότε χρειάζεται να κάνουμε augmentation τα δεδομένα έτσι ώστε να αποκτήσουν τα μοντέλα ανοχή. Αυτό προκαλεί περιττότητα (redundancy), δηλαδή έχουμε τον αρχικό αριθμό των εικόνων και θα πρέπει να δώσουμε στο μοντέλο έναν επιπλέον αριθμό augmented εικόνων για να μάθει το μοντέλο να αναγνωρίζει αντικείμενα σε rotated, scaled ή noised εικόνες. Αυτό έχει ως αποτέλεσμα να γίνεται πιο βαρύ το μοντέλο και να επεκτείνεται ο χρόνος εκπαίδευσης. Ένας τρόπος αντιμετώπισης είναι να δοθούν custom kernels, κάτι το οποίο θα βελτίωνε την εκπαίδευση σε χρόνο και επίσης θα είχαμε μοντέλα με μικρότερη αρχιτεκτονική, άρα και μικρότερες απαιτήσεις σε προβλήματα μνήμης.

Επιπλέον ένα άλλο μεγάλο πρόβλημα είναι η ευαισθησία (covariance) των μοντέλων. Για παράδειγμα τα κάποια χαρακτηριστικά του αμαξιού υπάρχει περίπτωση να ταιριάζουν με κάποια από ένα άλλο αντικείμενο όπως είναι το φορτηγό. Το μοντέλο έχει να λύσει την κατηγοριοποίηση και τον εντοπισμό των αντικειμένων. Το covariance βοηθάει στην εντόπιση με το αρνητικό όμως ότι κάνει το μοντέλο πολύ ευαίσθητο ως προς την κατηγοριοποίηση μιας περιοχής σε μια κλάση, γι’ αυτό τον λόγο τα πλαίσια οριοθέτησης που προβλέπει το μοντέλο δεν συμπίπτουν απόλυτα με τα πραγματικά πλαίσια.

Τέλος μια ακόμη βελτίωση έχει ως εξής. Στόχος του κλάδου είναι να χρησιμοποιηθούν τα μοντέλα αυτά ενσωματωμένα (embedded) σε μηχανές οχήματα ή οποιαδήποτε συσκευή μπορεί να χρησιμοποιηθεί για αναγνώριση αντικειμένων. Αυτό όμως ακόμη δεν μπορεί να γίνει διότι τα μοντέλα απαιτούν η συσκευή να είναι εξοπλισμένη με κάρτα γραφικών η οποία να ανταποκρίνεται στις απαιτήσεις των μοντέλων αυτών έτσι ώστε να μπορούν να λειτουργούν αξιόπιστα και γρήγορα. Αυτό συμβαίνει λόγω της μεγάλης τους αρχιτεκτονικής. Έτσι καταλήγουμε στο γεγονός να μην

έχουμε την δυνατότητα να επεξεργαστούμε τις προβλέψεις των μοντέλων επειδή δεν μας μένουν υπολογιστικοί πόροι. Οπότε μια ακόμη βελτίωση θα ήταν να φτιάξουμε τα μοντέλα με μικρότερη αρχιτεκτονική, έτσι ώστε να μην καταναλώνουν τόσο μεγάλη υπολογιστική ισχύ.

Κεφάλαιο 6: Βιβλιογραφία

[1] Fritz.ai. 2021. *Object Detection Guide | Fritz AI*. [online] Available at: https://www.fritz.ai/object-detection/

[2] viso.ai. 2021. *Mask R-CNN: A Beginner's Guide | viso.ai*. [online] Available at: https://viso.ai/deep-learning/mask-r-cnn

[3] Medium. 2021. *Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning*. [online] Available at: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

[4] Engineering Education (EngEd) Program | Section. 2021. *Introduction to YOLO Algorithm for Object Detection*. [online] Available at: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection

[5] Roboflow Blog. 2021. *YOLOv5 is Here*. [online] Available at: https://blog.roboflow.com/yolov5-is-here

[6] viso.ai. 2021. *Mask R-CNN: A Beginner's Guide | viso.ai*. [online] Available at: https://viso.ai/deep-learning/mask-r-cnn

6.1: Βιβλιογραφία εικόνων

- Εικόνα 1 https://www.fritz.ai/object-detection/#part-how
- Εικόνα 2 https://viso.ai/deep-learning/mask-r-cnn/
- Εικόνα 3 https://viso.ai/deep-learning/mask-r-cnn/
- Εικόνα 4 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 5 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 6 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 7 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 8 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 9 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 10 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 11 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

- Εικόνα 12 https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
- Εικόνα 13 https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/
- Εικόνα 14 https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/
- Εικόνα 15 https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/
- Εικόνα 16 https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/
- Εικόνα 17 https://viso.ai/deep-learning/mask-r-cnn/
- Εικόνα 18 https://viso.ai/deep-learning/mask-r-cnn/