



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

**ΑΝΑΓΝΩΡΙΣΗ ΣΥΓΚΡΟΥΣΕΩΝ ΑΥΤΟΚΙΝΗΤΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ
ΜΟΝΤΕΛΟΥ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ**

ΜΠΕΛΕΜΕΜΗΣ ΔΗΜΗΤΡΙΟΣ-ΧΡΗΣΤΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ

ΚΟΛΟΜΒΑΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

ΛΑΜΙΑ 2026



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

**ΑΝΑΓΝΩΡΙΣΗ ΣΥΓΚΡΟΥΣΕΩΝ ΑΥΤΟΚΙΝΗΤΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ
ΜΟΝΤΕΛΟΥ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ**

ΜΠΕΛΕΜΕΜΗΣ ΔΗΜΗΤΡΙΟΣ-ΧΡΗΣΤΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ

ΚΟΛΟΜΒΑΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

ΛΑΜΙΑ 2026



UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND TELECOMMUNICATIONS

CAR COLLISION DETECTION USING DEEP LEARNING MODEL

BELEMEMIS DIMITRIOS CHRISTOS

FINAL THESIS

ADVISOR

KOLOMVATSOS KONSTANTINOS

ASSOCIATE PROFESSOR

LAMIA 2026

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί χωρίς να τα περικλείω σε εισαγωγικά και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.

2. Δέχομαι ότι η αυτολεξεί παράθεση χωρίς εισαγωγικά, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια.

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 06/09/2026

Ο Δηλών.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Στην σημερινή εποχή, η ραγδαία τεχνολογική πρόοδος στους τομείς της τεχνητής νοημοσύνης και των τεχνολογιών βαθιάς μάθησης, έχει επηρεάσει άμεσα την πλειοψηφία των φορέων της κοινωνίας. Οι εξελίξεις σε αυτούς τους τομείς, προσφέρουν ουσιαστικές λύσεις σε ζητήματα που απαιτούν αυτοματοποίηση, για την άμεση αναγνώριση και αντιμετώπιση του κινδύνου. Ένα τέτοιο μείζον ζήτημα, είναι και η αναγνώριση συγκρούσεων αυτοκινήτων στο οδικό δίκτυο μιας κοινωνίας. Στην εργασία αυτήν, υλοποιείται και παρουσιάζεται μία μεθοδολογία ανίχνευσης συγκρούσεων αυτοκινήτων (Car Crash Collision Detection) από CCTV footage, με την χρήση ενός προ-εκπαιδευμένου μοντέλου CNN (Convolutional Neural Network, EfficientNet) και ενός YOLOv8 μοντέλου. Η συγκεκριμένη εφαρμογή, χρησιμοποιεί το YOLOv8 για την ανίχνευση των αυτοκινήτων και το EfficientNet (transfer learning) για την αναγνώριση των συγκρούσεων. Το σύστημα, διαθέτει την ικανότητα να λαμβάνει πραγματικά βίντεο οδικής κυκλοφορίας, εξαγόμενα από συστήματα CCTV. Στην συνέχεια, πραγματοποιείται η λεπτομερής ανάλυση τους, για την εύρεση πιθανών συγκρούσεων. Συγκεκριμένα, με την εφαρμογή του YOLOv8 και ορισμένων τεχνικών ελέγχου επικάλυψης των bounding boxes που εξάγονται από τον αλγόριθμο, πραγματοποιείται η ανατροφοδότηση του EfficientNet, με super boxes που αποκόπτονται από το υπόλοιπο frame και περικλείουν την σκηνή μίας πιθανής σκηνής σύγκρουσης ενός ζευγαριού οχημάτων. Έπειτα, το EfficientNet αναλύει την σκηνή που απεικονίζει το super box και την κατηγοριοποιεί ως σύγκρουση ή μη.

Μια τέτοια υλοποίηση, θα μπορούσε να προσφέρει ακαριαία ενημέρωση προς τις αρχές και τις υπηρεσίες έκτακτης ανάγκης, για την άμεση ανταπόκριση και την επίλυση του συμβάντος. Σκοπός της εργασίας, αποτελεί η ανίχνευση συγκρούσεων αυτοκινήτων μέσα από πραγματικό βιντεοληπτικό υλικό (CCTV footage), με την συνεισφορά τεχνολογιών βαθιάς μάθησης και μεθόδων επεξεργασίας εικόνας. Επιπλέον, η συγκεκριμένη εργασία αναδεικνύει την χρησιμότητα και την αποδοτικότητα των Deep Learning μοντέλων, σε τομείς που σχετίζονται με την επεξεργασία βίντεο και την αυτοματοποίηση των διαδικασιών.

ABSTRACT

Nowadays, the technological evolution that is being experienced in the fields of Artificial Intelligence and Deep Learning technologies, has affected dramatically many domains of the society. The progress that has been made in these fields, offer substantive and various solutions in cases where automation is required, for the immediate identification and solving of the incident. A major issue like this, could very likely be the collision of vehicles in a traffic road. In this project, a methodology will be developed and represented for the detection of car collisions from CCTV footage, using a pretrained EfficientNet and the YOLOv8n algorithm. This specific application, uses the YOLOv8n model for the detection of cars and the EfficientNet for the identification of a collision that is caused. The system, contains the ability to receive an actual video of public road traffic that has been extracted from CCTV systems. Afterwards, the videos are being deeply analyzed for the detection of collisions that maybe occurred. Specifically, with the usage of YOLOv8n and certain checking overlay techniques on the bounding boxes that are being detected from the YOLO algorithm, the system provides the EfficientNet with super boxes that are being cropped from the rest of the frame and contain the scene of a possible car collision. Then, the EfficientNet model analyzes the super box and decides whether there is a collision or not.

An application like this, could provide rapid information to the authorities and the emergency services for the immediate response and dealing of the incident that occurred. The purpose of this project, is to offer an efficient methodology for the detection of car collisions from CCTV footage, using Deep Learning technologies and image processing techniques. Last but not least, this application highlights the usage and the effectiveness of Deep Learning models, in cases where there are closely connected to video processing and automation of procedures.

Περιεχόμενα

ΠΕΡΙΛΗΨΗ	v
ABSTRACT	vi
ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ	1
1.1 Τροχαία ατυχήματα και οδική ασφάλεια	1
1.2 CCTV (Closed-Circuit Television)	2
1.3 Αυτόματα συστήματα αναγνώρισης συγκρούσεων	3
ΚΕΦΑΛΑΙΟ 2: ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΚΑΙ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ.....	5
2.1 Τεχνητή Νοημοσύνη.....	5
2.2 Μηχανική Μάθηση	6
2.2.1 Supervised Learning	6
2.2.2 Unsupervised Learning.....	7
2.2.3 Reinforcement Learning	8
2.3 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΒΑΘΙΑ ΜΑΘΗΣΗ	9
2.3.1 Artificial Neural Networks	9
2.3.2 Deep Learning	10
2.4 CNN (Convolutional Neural Network)	11
2.5 Transfer Learning (Μεταφορά Μάθησης).....	12
2.6 EfficientNet.....	13
2.7 Αλγόριθμοι ανίχνευσης αντικειμένων	15
2.7.1 Αλγόριθμος YOLO (You Only Look Once).....	16
2.7.2 Μοντέλα YOLO (You Only Look Once)	18
2.8 Παρόμοιες προσεγγίσεις ανίχνευσης συγκρούσεων	21
2.8.1 FASTER R-CNN	21
2.8.2 CNN-LSTM	22
ΚΕΦΑΛΑΙΟ 3: ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ.....	23
3.1 Εκπαίδευση δικτύου αναγνώρισης συγκρούσεων	23
3.1.1 Dataset (Σύνολο Δεδομένων) και Pre-Processing (Προ-επεξεργασία).....	23
3.1.2 Αρχιτεκτονική και Εκπαίδευση.....	27
3.1.3 Αποτελέσματα εκπαίδευσης EfficientNet-B3.....	32
3.2 Σύστημα ανίχνευσης αναγνώρισης συγκρούσεων	34
3.3 Αποτελέσματα και δοκιμή συστήματος.....	43
3.4 Συμπεράσματα	48
3.5 Πιθανές μελλοντικές βελτιώσεις.....	48
ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΑΝΑΦΟΡΕΣ	49

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Τροχαία ατυχήματα και οδική ασφάλεια

Τα τελευταία χρόνια, μία από τις πιο συχνές αιτίες θανάτων και τραυματισμών διεθνώς είναι τα τροχαία ατυχήματα. Ένα τροχαίο ατύχημα είναι ένα συμβάν, όπου κατά το οποίο ένα όχημα συγκρούεται με ένα άλλο, με έναν πεζό, με κάποιο ζώο ή με κάποιο στατικό εμπόδιο. Ένα τέτοιο γεγονός, μπορεί να προκληθεί συνήθως, από ανθρώπινο λάθος (όπως αναπτυγμένη ταχύτητα και οδήγηση υπό την επήρεια αλκοόλ) ή από περιβαλλοντικούς παράγοντες (όπως μειωμένη ορατότητα λόγω καιρικών συνθηκών και ελαττωματικού οδοστρώματος) [1].



Εικόνα 1: Σύγκρουση αυτοκινήτων [50]

Σύμφωνα με την τελευταία μελέτη του Παγκόσμιου Οργανισμού Υγείας, υπολογίζεται ότι περίπου 1.1160.000 άνθρωποι παγκοσμίως χάνουν την ζωή τους κάθε χρόνο από τροχαίο ατύχημα, ενώ παράλληλα περίπου 35 εκατομμύρια άνθρωποι υποφέρουν από σοβαρούς τραυματισμούς. Επιπρόσθετα, τα τροχαία ατυχήματα αποτελούν την νούμερο ένα αιτία θανάτου σε ηλικίες από 5 έως 29 χρονών [2].

Γίνεται πλέον εύκολα αντιληπτό, ότι η άμεση ανίχνευση ενός τροχαίου ατυχήματος μπορεί να προβεί καθοριστική, ώστε να ληφθούν τα απαραίτητα μέτρα, για την αντιμετώπιση των συνεπειών ενός τέτοιου γεγονότος. Με τον έγκαιρο εντοπισμό του ατυχήματος, οι υπηρεσίες έκτακτης ανάγκης

ειδοποιούνται την ίδια στιγμή, ώστε να μπορέσουν να προσφέρουν την απαραίτητη ιατρική βοήθεια στους τραυματίες του ατυχήματος.



Εικόνα 2: Σύγκρουση αυτοκινήτων και αντιμετώπιση [51]

Επιπλέον, η αποκατάσταση της κυκλοφορίας θα γίνει με ταχύτερο τρόπο σε περίπτωση που το συμβάν την εμποδίσει και κωλυσιεργήσει το οδικό δίκτυο. Συνοψίζοντας, η δημιουργία ενός συστήματος ανίχνευσης συγκρούσεων οχημάτων θα μειώσει σημαντικά το ποσοστό σοβαρών και μόνιμων τραυματισμών και θα μπορέσει να βοηθήσει τις αρχές να σπεύσουν άμεσα στο σημείο του ατυχήματος για την αντιμετώπιση των συνεπειών [3].

1.2 CCTV (Closed-Circuit Television)

Τα συστήματα CCTV είναι ευρέως γνωστά στον χώρο της τεχνολογίας, ως video surveillance cameras. Η πραγματική τους ορολογία όμως, είναι Closed-Circuit Television και αποτελούν συστήματα καμερών, που εκπέμπουν σε περιορισμένο αριθμό οθονών σε αντίθεση με τις κλασικές κάμερες. Η δημιουργία της τεχνολογίας των CCTV, πραγματοποιήθηκε την δεκαετία του 1940 από Γερμανούς επιστήμονες, για την παρακολούθηση εκτόξευσης πυραύλων και έπειτα υιοθετήθηκε από Αμερικάνους επιστήμονες, για την εκτενή παρακολούθηση πειραμάτων της ατομικής βόμβας.

Πλέον, τα CCTV συστήματα χρησιμοποιούνται κυρίως για την ανίχνευση εγκληματικών ενεργειών και των παραβάσεων του κώδικα οδικής κυκλοφορίας, αλλά και σε πολλές περιπτώσεις αποτελούν την πρώτη γραμμή άμυνας για την ασφάλεια ενός σπιτιού ή μίας επιχείρησης. Με την τοποθέτηση συστημάτων CCTV, ο κάτοχος μπορεί να διαθέσει στις αρχές υλικό από βίντεο για έρευνα, που μπορεί να προσφέρει αποδεικτικά στοιχεία για την επίλυση υποθέσεων και την ενοχή των παραβατών. Σε αυτήν την εργασία, το υλικό από τις CCTV κάμερες χρησιμοποιείται για την συλλογή δεδομένων, για την εκπαίδευση του μοντέλου και έπειτα το σύστημα CCTV σε συνδυασμό με το μοντέλο, καθίσταται ικανό να ανιχνεύσει συγκρούσεις αυτοκινήτων [4].

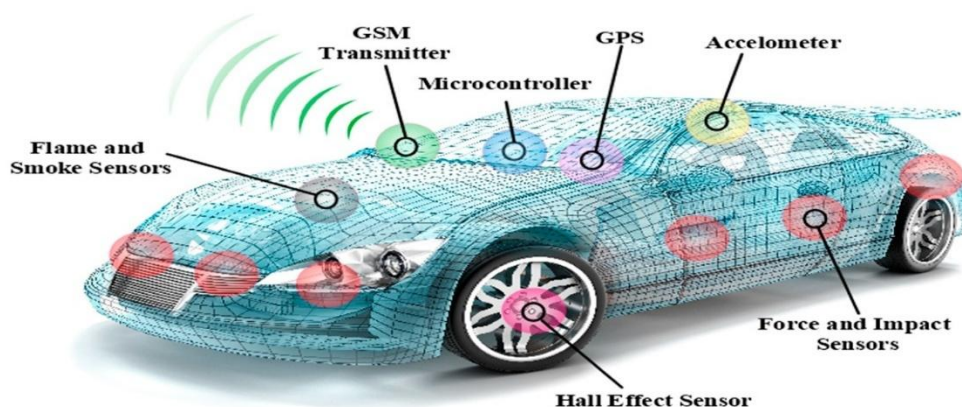


Εικόνα 3: CCTV κάμερες [28]

Παρ' όλα αυτά, το βιντεοληπτικό υλικό που συλλέγεται από αυτά τα συστήματα αποτελεί έναν τεράστιο όγκο δεδομένων. Για αυτόν τον λόγο, η διαρκής παρακολούθηση από ανθρώπινο δυναμικό είναι σχεδόν αδύνατη. Με τα παραπάνω δεδομένα, γίνεται κατανοητό ότι η ανάγκη για δημιουργία αυτοματοποιημένων συστημάτων που θα στηρίζονται σε τεχνικές βαθιάς μηχανικής μάθησης και θα διαθέτουν την δυνατότητα να επεξεργάζονται το υλικό και να λαμβάνουν στιγμιαία απόφαση καθίσταται όλο και πιο σημαντική. Με την κατάλληλη εκπαίδευση, ένα μοντέλο έχει την δυνατότητα να αναγνωρίσει ασυνήθιστες συμπεριφορές και παραβάσεις.

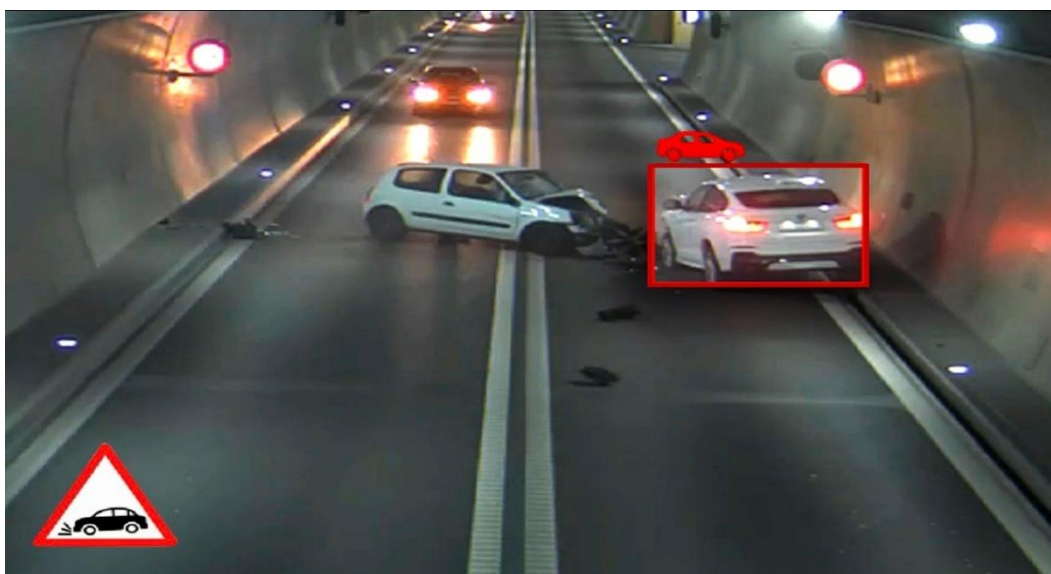
1.3 Αυτόματα συστήματα αναγνώρισης συγκρούσεων

Στις μέρες μας, τα περισσότερα καινούργια μοντέλα αυτοκινήτων διαθέτουν τεχνολογίες αυτόματης ανίχνευσης συγκρούσεων, χρησιμοποιώντας διάφορα είδη αισθητήρων, όπως πρόσκρουσης και κραδασμού. Παράλληλα, αποσπούν πληροφορίες από GPS trackers και επιταχυνσιόμετρα του αυτοκινήτου. Με αυτόν τον τρόπο, τα συστήματα αυτά είναι ικανά να προσδιορίσουν απότομες μεταβολές, σχετικά με την θέση και την ταχύτητα του αυτοκινήτου. Στην συνέχεια, εκπέμπουν μηνύματα άμεσης βοήθειας, προς τις υπηρεσίες έκτακτης ανάγκης. Τα συστήματα αυτά είναι γνωστά στον χώρο της τεχνολογίας ως AID (Automatic Incident Detection) [52].

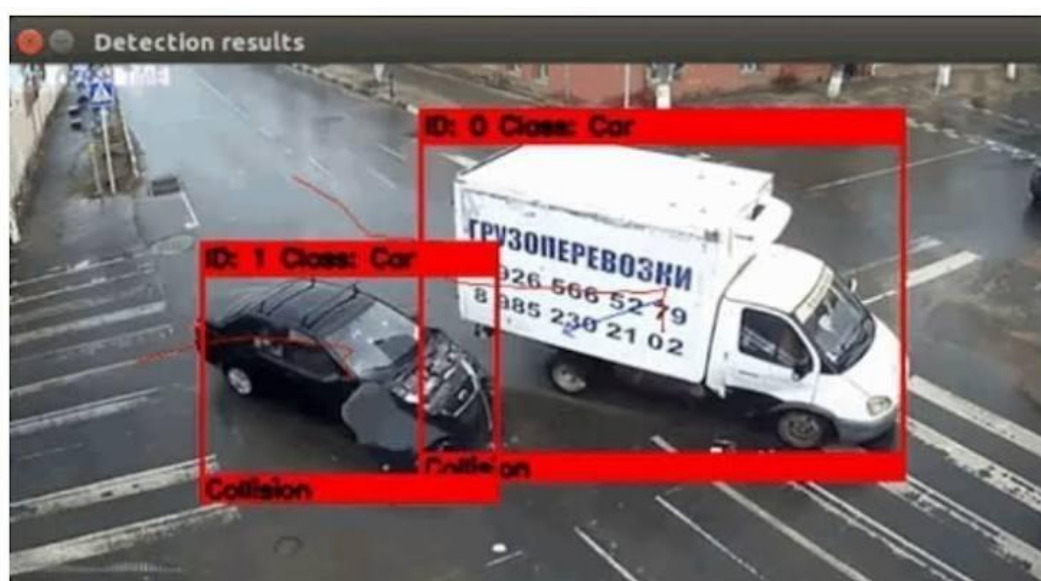


Εικόνα 4: Τεχνολογίες ανίχνευσης συγκρούσεων [30]

Ωστόσο, η χρήση αυτών των συστημάτων καθίσταται δυνατή μόνο σε αυτοκίνητα τελευταίας τεχνολογίας, που διαθέτουν τον απαραίτητο εξοπλισμό για την αποτελεσματική λειτουργία τους. Το γεγονός αυτό μαζί με την απότομη εξέλιξη της τεχνητής νοημοσύνης, έφερε στο προσκήνιο μία νέα οπτική, αυτήν της βαθιάς μάθησης. Πλέον, τα συστήματα ανίχνευσης συγκρούσεων επωφελούνται από το βιντεοληπτικό υλικό που προσφέρουν οι CCTV κάμερες και σε συνδυασμό με την εφαρμογή τεχνικών βαθιάς μηχανικής μάθησης λαμβάνουν το επιθυμητό αποτέλεσμα. Τα κυριότερα πλεονεκτήματα των συστημάτων που υιοθετούν αυτήν την μέθοδο, είναι η έγκαιρη ανίχνευση του συμβάντος χωρίς την ανθρώπινη παρουσία ενός χειριστή και η εύκολη εφαρμογή τους ανεξαρτήτου μοντέλου και τεχνολογίας των εμπλεκόμενων οχημάτων στο συμβάν. Η αποκλειστικότητα της πηγής δεδομένων των συστημάτων αυτών, τα έχει κάνει να χαρακτηρίζονται και ως Video-Based Incident Detection Systems [52].



Εικόνα 5: Σύστημα ανίχνευσης συγκρούσεων [53]

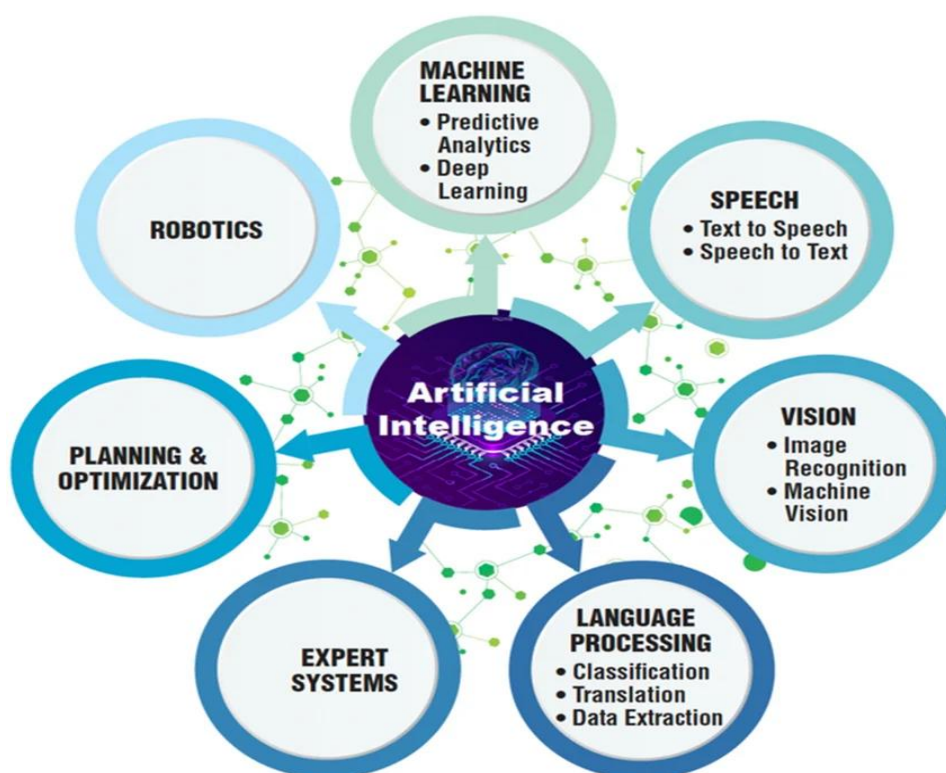


Εικόνα 6: Σύστημα ανίχνευσης συγκρούσεων [29]

ΚΕΦΑΛΑΙΟ 2: ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΚΑΙ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

2.1 Τεχνητή Νοημοσύνη

Τα τελευταία χρόνια ο όρος Τεχνητή Νοημοσύνη (Artificial Intelligence (A.I.)) έχει κάνει δυναμική την παρουσία του τόσο στην καθημερινή ζωή όσο και σε ερευνητικό και βιομηχανικό επίπεδο. Πιο συγκεκριμένα, η τεχνητή νοημοσύνη είναι μία αναδυόμενη τεχνολογία, που έχει την δυνατότητα βελτιώσει και να εκσυγχρονίσει πολλούς τομείς της κοινωνίας όπως η υγεία, η οικονομία και οι μεταφορές. Σύμφωνα με τους Russel S.J και Norvig P. η τεχνητή νοημοσύνη είναι ένας τομέας της επιστήμης της πληροφορικής, που έχει ως σκοπό την δημιουργία ενός προγράμματος ή ενός μηχανήματος που θα επιλύει προβλήματα που υπό άλλες συνθήκες θα χρειαζόταν ανθρώπινη νοημοσύνη. Με λίγα λόγια, η τεχνητή νοημοσύνη προσπαθεί να μιμηθεί όσο πιο καλά γίνεται μερικές από τις ικανότητες του ανθρώπου όπως είναι η λήψη αποφάσεων, η επίλυση προβλημάτων και η κατανόηση.



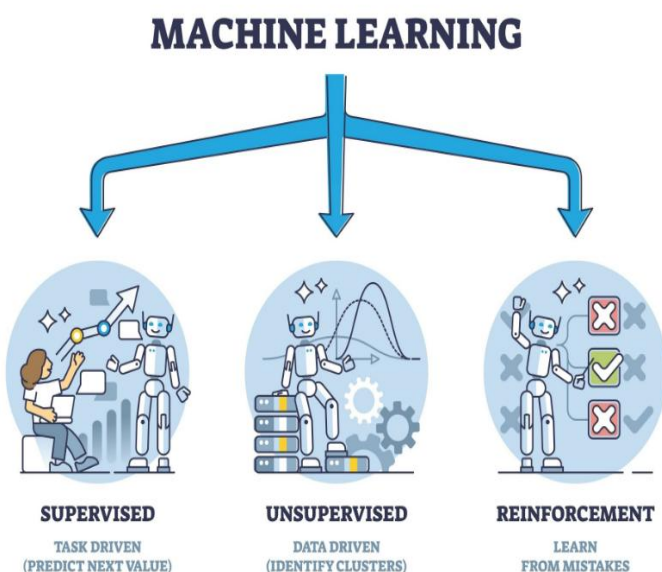
Εικόνα 7: Τεχνολογίες τεχνητής νοημοσύνης [31]

Ιστορικά, η έννοια της τεχνητής νοημοσύνης τέθηκε στα μέσα του 20^{ου} αιώνα από τους Alan Turing, John McCarthy και Marvin Minsky. Συγκεκριμένα, η πρώτη μέθοδος αξιολόγησης νοημοσύνης των μηχανών γνωστό και ως Turing Test προτάθηκε από τον Alan Turing. Τις επόμενες δεκαετίες, έγιναν

πολλές προτάσεις μεθόδων και καινοτομιών σχετικά με την τεχνητή νοημοσύνη και την εξέλιξη της, όμως λόγω της περιορισμένης υπολογιστικής ισχύς και των λιγοστών δεδομένων, το εγχείρημα δεν συνεχίστηκε. Πλέον, με την ραγδαία τεχνολογική πρόοδο των τελευταίων χρόνων, η τεχνητή νοημοσύνη αναπτύσσεται γοργούς ρυθμούς και χρησιμοποιείται στους περισσότερους τομείς τις κοινωνίας μας [6], [7].

2.2 Μηχανική Μάθηση

Τα περισσότερα έξυπνα συστήματα, που διαθέτουν ικανότητες τεχνικής νοημοσύνης σχετίζονται άμεσα με την μηχανική μάθηση. Σκοπός της μηχανικής μάθησης, είναι η αυτόματη μάθηση σημαντικών σχέσεων και μοτίβων από παραδείγματα και παρατηρήσεις. Πιο συγκεκριμένα, ο υπολογιστής τροφοδοτείται με διαφόρων ειδών δεδομένα από πολλαπλές πηγές, τα οποία αναλύονται ώστε να βρεθούν συσχετισμοί μεταξύ τους και να εξαχθούν πολύτιμα συμπεράσματα και προβλέψεις. Με το πέρασ κάθε επανάληψης, η μηχανή γίνεται πιο ισχυρή και πιο βέβαιη για τα αποτελέσματα και τις προβλέψεις που εξάγει [8].

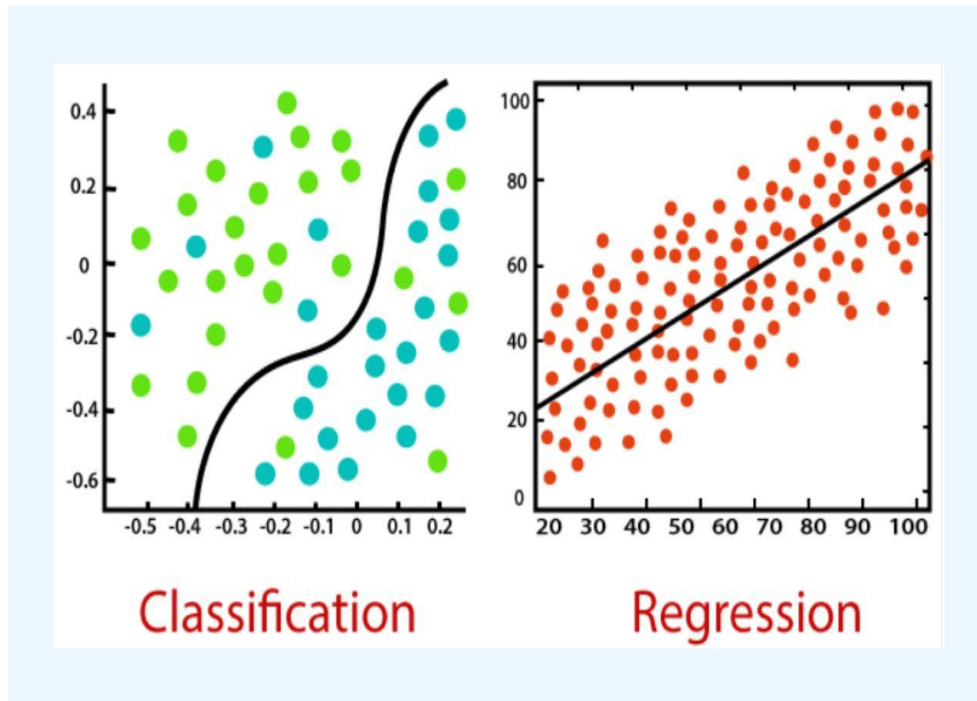


Εικόνα 8: Κατηγορίες μηχανικής μάθησης [32]

2.2.1 Supervised Learning

Η επιβλεπόμενη μάθηση (Supervised Learning), αποτελεί μία ειδική περίπτωση της μηχανικής μάθησης και προϋποθέτει ότι το σύνολο δεδομένων που χρησιμοποιείται για την εκπαίδευση ενός μοντέλου, διαθέτει ετικέτες (labels) στην έξοδο. Στην ουσία, κατά την διάρκεια της εκπαίδευσης το μοντέλο επεξεργάζεται τα δεδομένα εισόδου και προσπαθεί να τα ταξινομήσει στην αντίστοιχη έξοδο

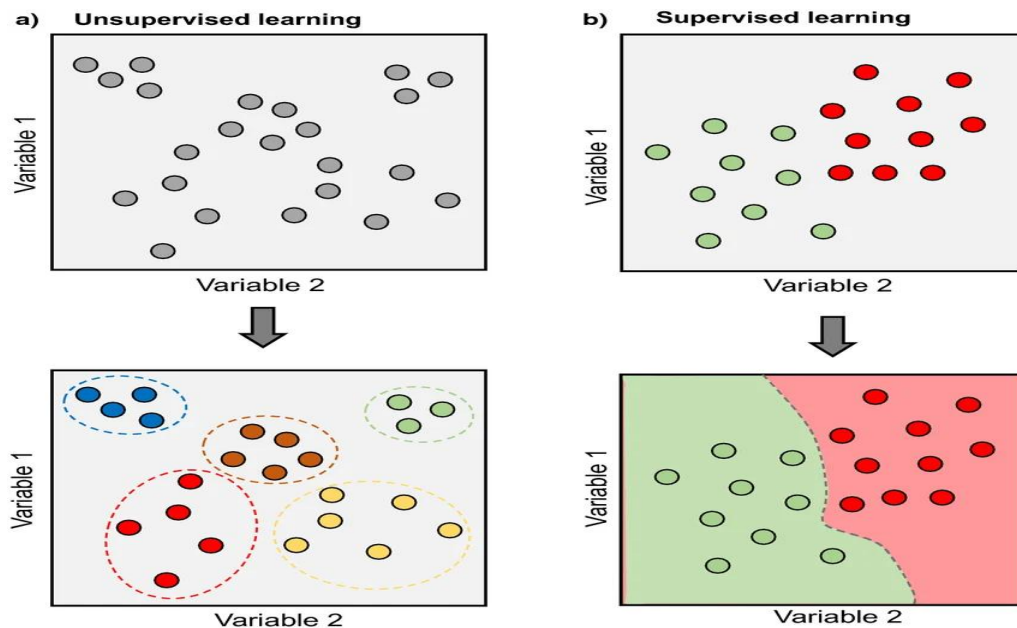
με την σωστή ετικέτα. Όταν τελειώσει η εκπαίδευση, το μοντέλο είναι σε θέση να λάβει άγνωστα δεδομένα που δεν χρησιμοποιήθηκαν κατά την διάρκεια της εκπαίδευσης του και να προβλέψει με μεγάλα ποσοστά ακρίβειας την σωστή έξοδο. Βασικά παραδείγματα επιβλεπόμενης μάθησης αποτελούν η κατηγοριοποίηση (Classification) και η παλινδρόμηση (Regression) [8].



Εικόνα 9: Παραδείγματα κατηγοριοποίησης και παλινδρόμησης [33]

2.2.2 Unsupervised Learning

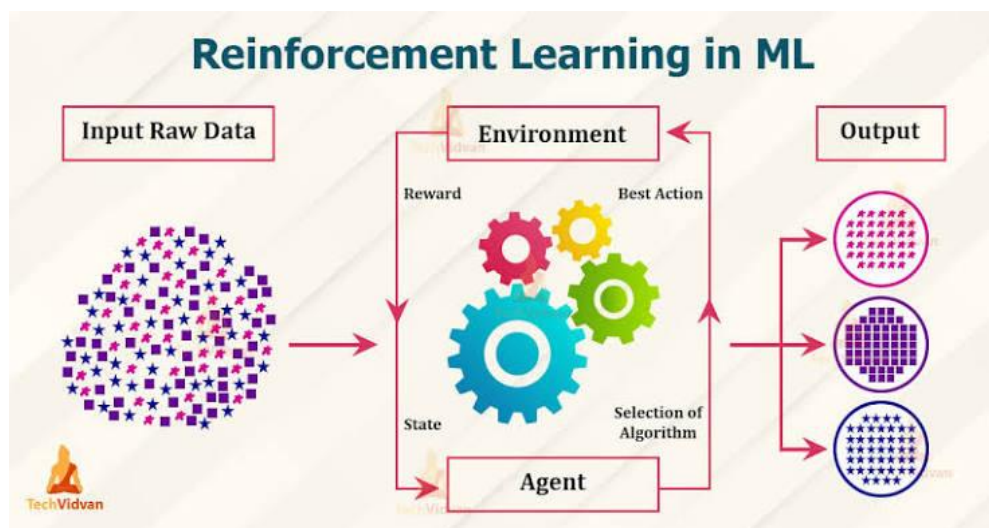
Η μη επιβλεπόμενη μάθηση (Unsupervised Learning), είναι και αυτή μία ξεχωριστή περίπτωση της μηχανικής μάθησης και σε αντίθεση με την επιβλεπόμενη μάθηση, το μοντέλο προσπαθεί να βρει συσχετισμούς και μοτίβα μεταξύ των δεδομένων εισόδου, χωρίς την ύπαρξη ετικετών στα δεδομένα εξόδου. Έτσι λοιπόν, στο σύνολο δεδομένων τα δεδομένα εισόδου x , δεν διαθέτουν την αντίστοιχη σωστή ετικέτα (label) y . Το μοντέλο προσπαθεί χωρίς βοήθεια, να κάνει ομαδοποιήσεις μεταξύ των δεδομένων. Οι ομαδοποιήσεις αυτές είναι στην ουσία συστάδες (clusters) δεδομένων, που το μοντέλο επισήμανε ότι έχουν κοινά χαρακτηριστικά και συσχετίζονται μεταξύ τους. Η τεχνική αυτή, συχνά χρησιμοποιείται σε αδόμητα δεδομένα και σε περιπτώσεις όπου χρειάζεται διαχωρισμός, όπως σε εφαρμογές που σχετίζονται με πελάτες, επιχειρήσεις και φοιτητές [8].



Εικόνα 10: Παράδειγμα συσταδοποίησης [34]

2.2.3 Reinforcement Learning

Στην ενισχυτική μάθηση (Reinforcement Learning), η διαδικασία της εκπαίδευσης δεν γίνεται με την κλασική διαδικασία, όπου δίνονται στο μοντέλο ζευγάρια εισόδων και εξόδων δεδομένων. Συγκεκριμένα, στο μοντέλο παρέχεται μια πλήρης περιγραφή της τρέχουσας κατάστασης όπου βρίσκεται, του στόχου, των επιτρεπτών ενεργειών που είναι σε θέση να αξιοποιήσει στην κατάσταση αυτήν, καθώς και τυχόν περιορισμών που υφίστανται. Το μοντέλο αφομοιώνει μία λογική ανταμοιβής-ποινής, όπου για κάθε σωστή ενέργεια επιβραβεύεται και για κάθε λανθασμένη τιμωρείται. Στόχος του, αποτελεί η μεγιστοποίηση της ανταμοιβής. Η ενισχυτική μάθηση (Reinforcement Learning), έχει υιοθετηθεί από πολλές εταιρίες παραγωγής στον χώρο του gaming καθώς και σε πολλές έξυπνες ηλεκτρονικές συσκευές σπιτιού [8].

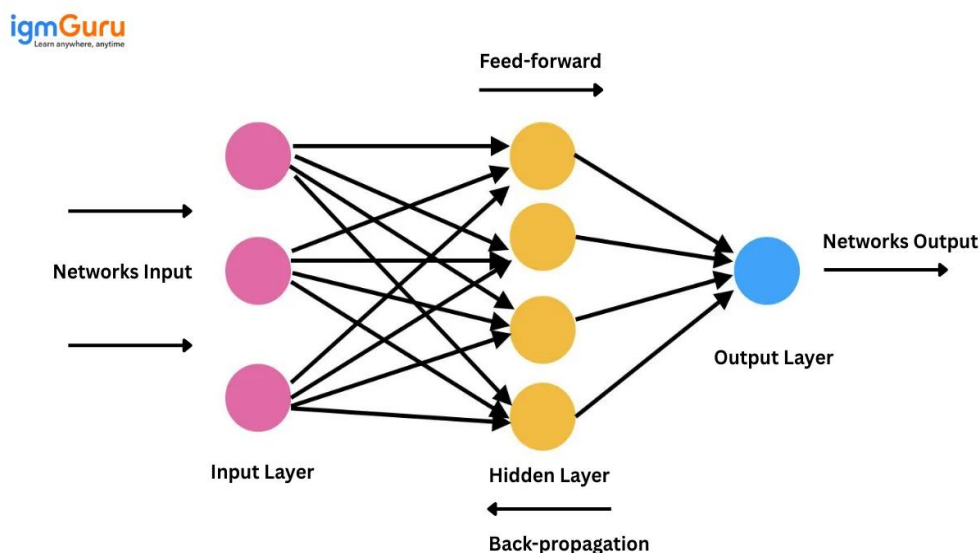


Εικόνα 11: Παράδειγμα ενισχυμένης μάθησης [35]

2.3 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΒΑΘΙΑ ΜΑΘΗΣΗ

2.3.1 Artificial Neural Networks

Τα νευρωνικά δίκτυα (Artificial Neural Networks), αποτελούν μία τεχνική μάθησης μηχανών, που αποσκοπούν στην προσομοίωση του μηχανισμού μάθησης του ανθρώπινου εγκεφάλου. Το θεωρητικό σκέλος των νευρωνικών δικτύων, πρωτοεμφανίστηκε σαν μαθηματικό μοντέλο την δεκαετία του 1940. Από τότε, τα νευρωνικά δίκτυα έχουν αποτελέσει την κινητήρια δύναμη αναβάθμισης, δεκάδων κλάδων ενασχόλησης με την τεχνητή νοημοσύνη και την μηχανική μάθηση. Ένα νευρωνικό δίκτυο αποτελείται από κόμβους που συχνά αποκαλούνται νευρώνες, οι οποίοι διασυνδέονται μεταξύ τους για την υλοποίηση μαθηματικών υπολογισμών. Το παραγόμενο αποτέλεσμα, μεταδίδεται σε νευρώνες επόμενων επιπέδων. Στα νευρωνικά δίκτυα ενός επιπέδου, το σύνολο εισόδων χαρτογραφείται κατευθείαν σε μία έξοδο. Από την άλλη, οι νευρώνες στα νευρωνικά δίκτυα πολλαπλών επιπέδων (εμπροσθοδοτά) διατάσσονται κυρίως σε τρία επίπεδα [6], [7].



Εικόνα 12: Παράδειγμα νευρωνικού δικτύου [36]

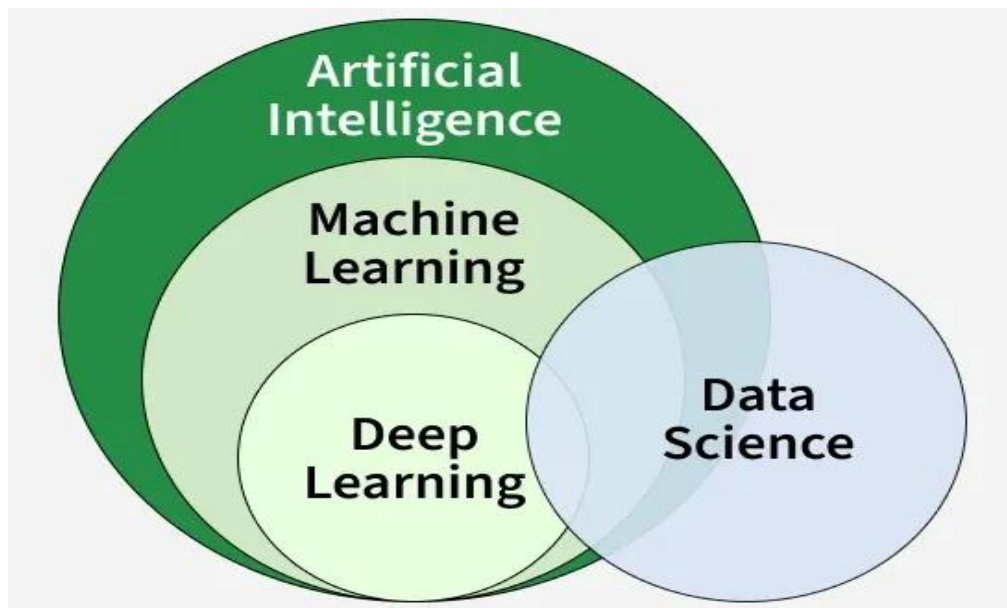
Input Layer: Σε αυτό το επίπεδο, τα δεδομένα εισέρχονται στο νευρωνικό δίκτυο και στην συνέχεια οι νευρώνες επεξεργάζονται τα δεδομένα και τα αποστέλλουν στο επόμενο επίπεδο.

Hidden Layer: Τα επίπεδα αυτά, είναι στην ουσία τα υπολογιστικά επίπεδα και αποτελούν το ενδιάμεσο στρώμα ενός νευρωνικού δικτύου. Σε αυτό σημείο, πραγματοποιείται οι πλειοψηφία των σημαντικότερων και πιο περίπλοκων υπολογισμών του δικτύου.

Output Layer: Αποτελεί το τελευταίο στάδιο του νευρωνικού δικτύου, από όπου λαμβάνεται το επιθυμητό αποτέλεσμα του δικτύου.

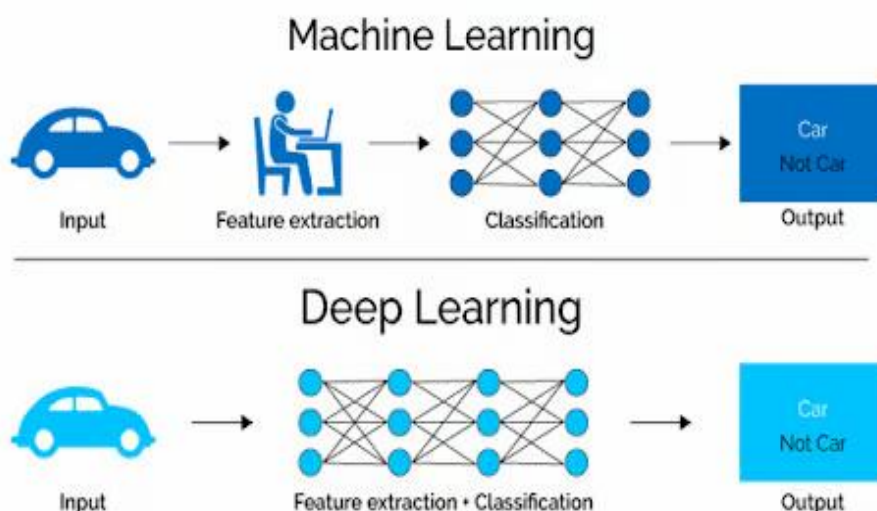
2.3.2 Deep Learning

Στον κόσμο της τεχνολογίας, συχνά ο όρος βαθιά μάθηση συγχέεται με τον όρο της μηχανικής μάθησης. Η βαθιά μάθηση (Deep Learning) θεωρείται μία εξελιγμένη μορφή της μηχανικής μάθησης, που στηρίζεται στα τεχνητά βαθιά νευρωνικά δίκτυα (Artificial Neural Networks). Η βαθιά μάθηση σε αντίθεση με την απλή μηχανική μάθηση, προσφέρει μία αυτοματοποιημένη κατάσταση εξαγωγής χαρακτηριστικών των δεδομένων, καθώς καθορίζει από μόνη της, την βαρύτητα που φέρουν ορισμένα χαρακτηριστικά, για την αξιοποίηση τους μελλοντικά σε επόμενες λειτουργίες του δικτύου.



Εικόνα 13: Εξήγηση μηχανικής μάθησης [37]

Ένας από τους λόγους που η βαθιά μάθηση θεωρείται αποτελεσματικότερη από την μηχανική μάθηση, σχετίζεται με τα δίκτυα που αποτελούνται από περισσότερα από ένα κρυφά επίπεδα (Hidden Layers), που διακρίνονται για την ικανότητα τους να δέχονται ωμά δεδομένα, να προχωρούν στην εξονυχιστική ανάλυσή τους και στην συνέχεια να ανταποκρίνονται με εντυπωσιακά αποτελεσματικό τρόπο στο πρόβλημα. Ακόμη, οι νευρώνες στην βαθιά μάθηση είναι πιο εξελιγμένοι, καθώς χρησιμοποιούν περισσότερες και πιο περίπλοκες συναρτήσεις ενεργοποίησης, σε σχέση με τους νευρώνες στην απλή μηχανική μάθηση. Λόγω της πολυπλοκότητας και των εκατομμυρίων παραμέτρων που διαθέτουν τα δίκτυα στην βαθιά μάθηση, συνήθως απαιτείται μεγάλη επάρκεια δεδομένων, καθώς και πληθώρα υπολογιστικών πόρων, για την αποτελεσματική εκπαίδευση ενός τέτοιου δικτύου [8], [9], [10].



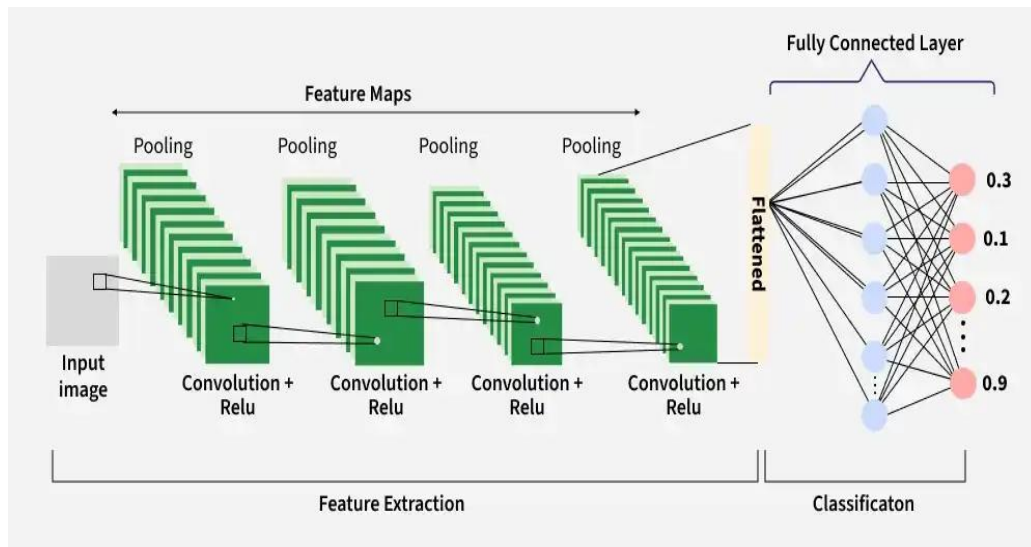
Εικόνα 14: Διαφορές ML και Deep Learning [38]

Λόγω της προηγμένης αρχιτεκτονικής των δικτύων, της βαθιάς μάθησης και τον τεράστιο όγκο δεδομένων που πρέπει να διαχειριστούν, η υπολογιστική ισχύς που χρειάζεται για την εκπαίδευση τους αποτελεί ένα ζήτημα στον χώρο της τεχνολογίας. Σύμφωνα με σύγχρονες έρευνες, αλγόριθμοι της μηχανικής μάθησης μπορούν να παρουσιάσουν ισάξια αποτελέσματα με αυτά των αλγορίθμων της βαθιάς μάθησης, σε δεδομένα εισόδου μικρών διαστάσεων και όγκου, ξοδεύοντας παράλληλα πολύ μικρότερη υπολογιστική ισχύ. Για αυτόν τον λόγο, τα τελευταία χρόνια υπάρχει μία τάση αξιοποίησης της βαθιάς μάθησης σε περιπτώσεις όπου τα δεδομένα είναι περίπλοκα, υπέρογκα και χρήζουν βοήθειας των τεχνικών της, που αναπτύσσονται με ραγδαίους ρυθμούς [8], [9], [10], [11].

2.4 CNN (Convolutional Neural Network)

Ένα από τα πιο γνωστά και σημαντικά δίκτυα βαθιάς μάθησης, είναι τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks). Τα CNN, πρωτοεμφανίστηκαν σαν ιδέα το 1959 και ανήκουν στην οικογένεια των νευρωνικών δικτύων, υιοθετώντας την ίδια λογική με τα κλασικά ANN. Η διαφορά τους ενδείκνυται στο γεγονός, ότι τα CNN είναι φτιαγμένα για την εξαγωγή χαρακτηριστικών από εικόνες. Πιο συγκεκριμένα, τα κλασικά ANN παρουσιάζουν δυσκολία στην υπολογιστική πολυπλοκότητα που χρειάζεται για τον υπολογισμό δεδομένων εικόνας. Για αυτόν τον λόγο, τα CNN αποτελούν την πιο αποδοτική επιλογή για εφαρμογές υπολογιστικής όρασης [12], [13].

Όπως και στα ANN ένα στρώμα εισόδου αποθηκεύει τις τιμές όλων των pixel της εικόνας. Η συνολική αρχιτεκτονική των CNN συγκροτείται από τριών ειδών στρώματα:



Εικόνα 15: Παράδειγμα CNN [39]

Convolutional Layer: Ένα τέτοιου είδους στρώμα καθορίζει την έξοδο των νευρώνων, οι οποίοι με την σειρά τους είναι συνδεδεμένοι με περιοχές της εικόνας. Αυτό πραγματοποιείται με την χρήση φίλτρων (kernels) όπου σκανάρουν ολόκληρη την εικόνα και πραγματοποιούν μαθηματικές πράξεις συνέλιξης. Έπειτα, εξάγεται ένας χάρτης χαρακτηριστικών, ο οποίος μέσω των τιμών του, φανερώνει τα χαρακτηριστικά όπου εξάχθηκαν. Σε ένα CNN, μπορεί να υπάρχουν περισσότερα από ένα τέτοια επίπεδα, με το καθένα να εμβαθύνει ολοένα και περισσότερο στην εικόνα, ανακαλύπτοντας πιο δύσκολα μοτίβα [12], [13].

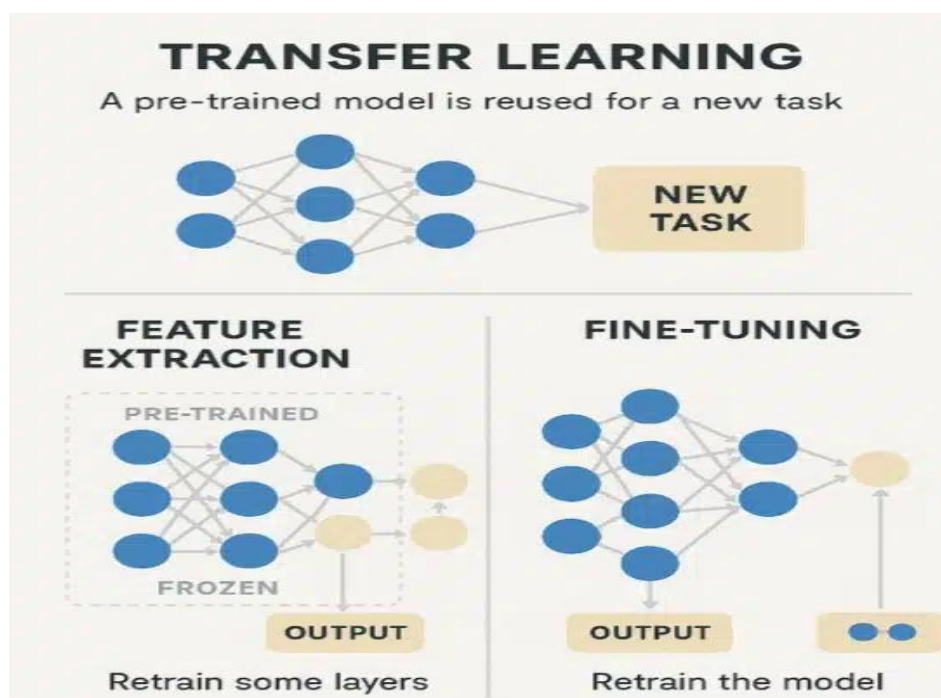
Pooling Layer: Αποτελεί το επόμενο στάδιο ενός CNN, καθώς είναι ένα στρώμα υποδειγματοληψίας. Στην ουσία το Pooling Layer, μειώνει τις διαστάσεις της δοσμένης εξόδου (χάρτης χαρακτηριστικών), όπου έλαβε σαν είσοδο, με στόχο την ελαχιστοποίηση των παραμέτρων και την αποφυγή αχρείαστων μοτίβων, κρατώντας μόνο τα σημαντικά χαρακτηριστικά της εικόνας.

Fully Connected Layer: Συνήθως ένα CNN, αποτελείται από περισσότερα από ένα τέτοιου είδους στρώματα. Ακολουθούν την ίδια λογική με τα κλασικά ANN, διότι αποσκοπούν στην παραγωγή βαθμολογιών κλάσης, για την κατηγοριοποίηση της εικόνας στην σωστή κλάση.

2.5 Transfer Learning (Μεταφορά Μάθησης)

Ο τομέας της μηχανικής μάθησης έχει βιώσει ραγδαία τεχνολογική πρόοδο σε αρκετές μεθοδολογίες πρόβλεψης δεδομένων, όπως κατηγοριοποίηση, συσταδοποίηση και παλινδρόμηση. Ένα από τα προβλήματα που εμφανίστηκαν τα τελευταία χρόνια στο πεδίο της βαθιάς μάθησης είναι ότι, συγκεκριμένες τεχνικές και μοντέλα αποδίδουν αποτελεσματικά μόνο σε τομείς, όπου έχουν εκπαιδευτεί με ίδιου ή παρόμοιου είδους δεδομένα. Επιπρόσθετα, το υπολογιστικό κόστος και η σπατάλη χρόνου επανεκπαίδευσης ενός νέου μοντέλου εντείνουν το πρόβλημα που υπάρχει.

To Transfer Learning, είναι μία τεχνική που δίνει μία άλλη διάσταση στην κατάσταση αυτήν. Στην ουσία το Transfer Learning, προσφέρει την δυνατότητα σε έναν software engineer να αξιοποιήσει την λειτουργία ενός pretrained μοντέλου, όπου έχει χρησιμοποιηθεί αποτελεσματικά, σε ενός παρόμοιου είδους προβλήματος. Τα πρώτα στρώματα του προεπαιδευμένου μοντέλου, συνήθως διατηρούνται πλήρως ανέπαφα, για την αναγνώριση και την εξαγωγή χαρακτηριστικών, ενώ τα τελευταία στρώματα τις περισσότερες φορές ρυθμίζονται και προσαρμόζονται από τον αρχιτέκτονα, με βάση το πρόβλημα που επιλύεται και την έξοδο όπου επιθυμείτε (Fine-Tuning) [15].



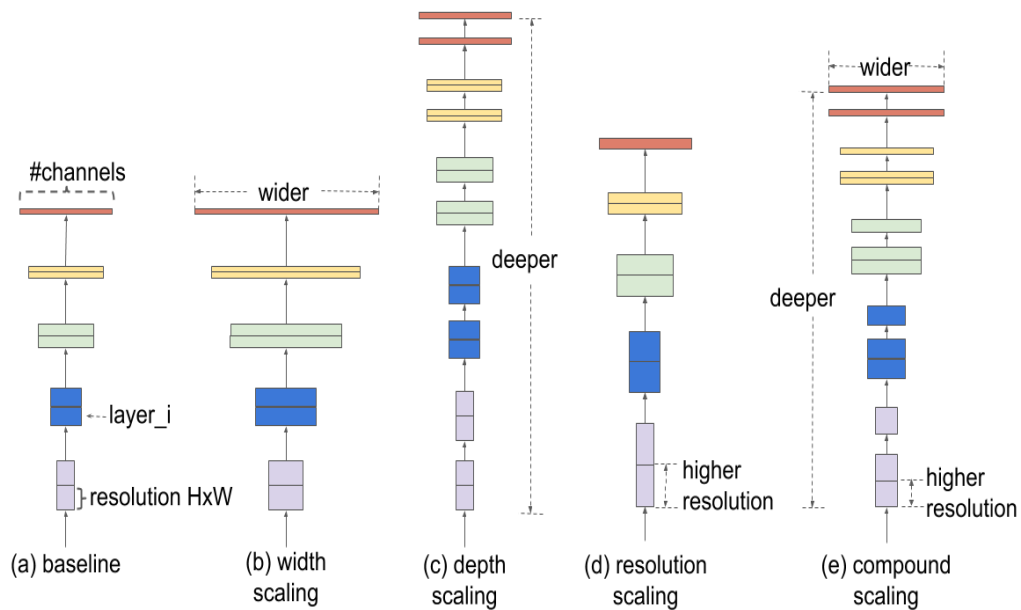
Εικόνα 16: Transfer Learning παράδειγμα [40]

2.6 EfficientNet

Η αναβάθμιση των Convolutional Neural Networks αποτελεί μία κοινή αποδεκτή μέθοδο, για την βελτιστοποίηση της ακρίβειας των CNN και την παραγωγή καλύτερων αποτελεσμάτων στον τομέα της υπολογιστικής όρασης. Το EfficientNet είναι ένα προηγμένο CNN, που υιοθετήθηκε από τους Mingxing Tan και Quoc V. Le το 2019, με σκοπό την δημιουργία ενός δικτύου, που θα διακρίνεται για την αποτελεσματικότητά του, την αξιοπιστία καθώς και την οικονομικότητά του [14].

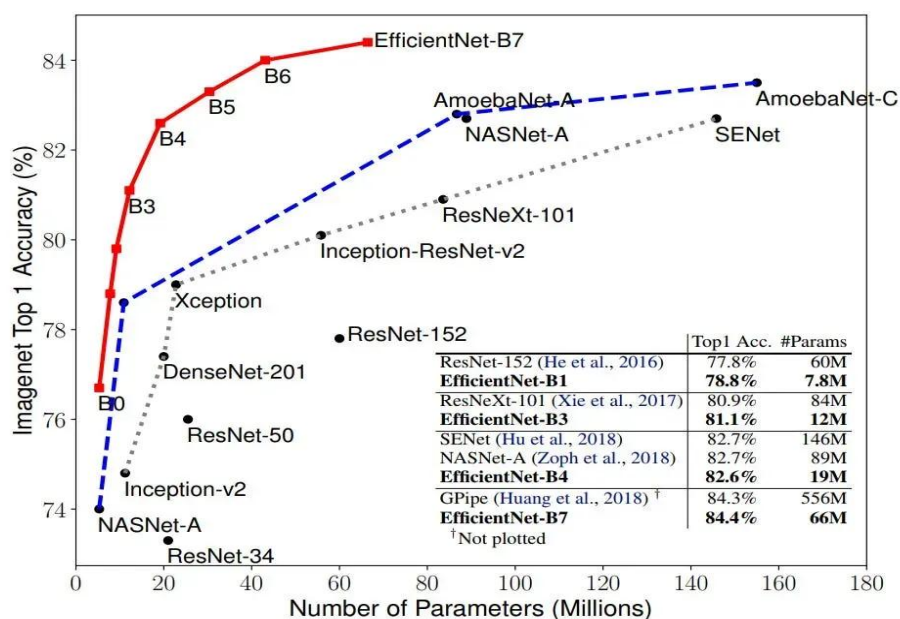
Η ιδιαιτερότητα της αρχιτεκτονικής του EfficientNet, εμφανίζεται στον τρόπο με τον οποίο κλιμακώνονται το βάθος, το πλάτος και η ανάλυση της εισόδου του δικτύου. Πιο αναλυτικά, οι δημιουργοί του επινοήσαν το Compound Scaling, μία τεχνική που αψηφά την μέθοδο με την οποία κλιμακώνονταν τα μέχρι τότε εξελιγμένα CNN. Το Compound Scaling, στην ουσία αυξάνει και τις τρεις διαστάσεις του δικτύου με fixed παραμέτρους ταυτόχρονα και με τον ίδιο τρόπο. Οι Mingxing Tan και Quoc V. Le υποστηρίζουν ότι η ομοιόμορφη κλιμάκωση και των τριών διαστάσεων την ίδια στιγμή, προσφέρει καλύτερη εκμετάλλευση της υπολογιστικής ισχύς καθώς και υψηλότερους δείκτες

ακρίβειας. Σύμφωνα με την μελέτη τους, η δικιά τους πρόταση φαίνεται να υπερισχύει της λογικής των δικτύων που κλιμακώνουν μόνο μία διάσταση, με αυθαίρετο τρόπο κάθε φορά [14].



Εικόνα 17: Διάγραμμα *compound scaling*

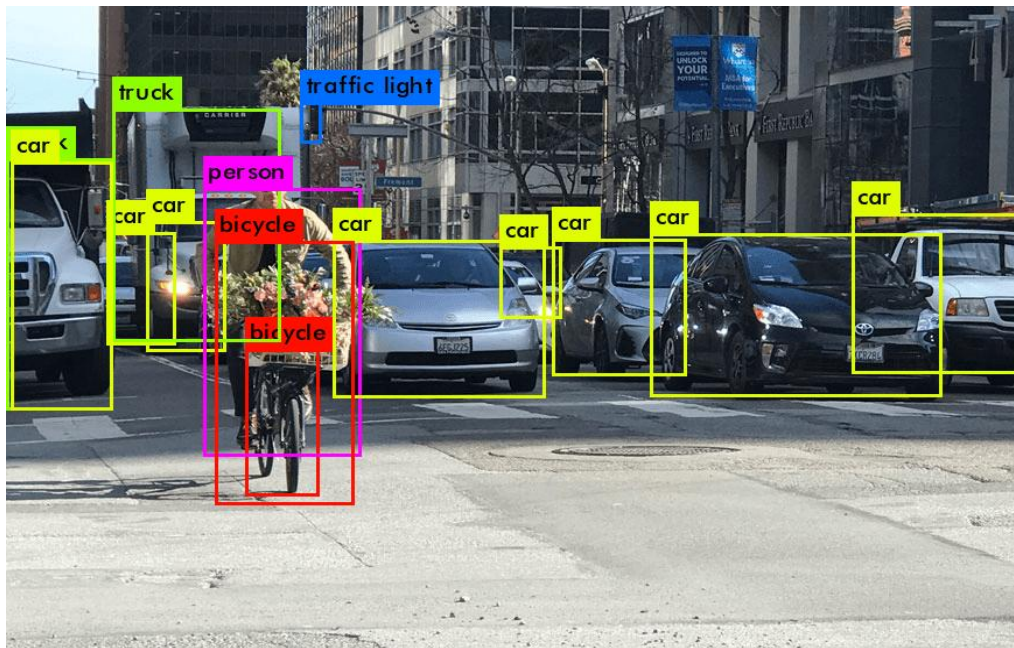
Οι οικογένεια των EfficientNet διαθέτει επτά μοντέλα, οι πιο εξελιγμένες εκδόσεις έχουν μεγαλύτερη και πιο περίπλοκη αρχιτεκτονική, προσφέρουν καλύτερα αποτελέσματα, αλλά δαπανούν πολύ περισσότερη ενέργεια. Τα μικρότερα μοντέλα διακρίνονται για τα άμεσα, αξιόπιστα αποτελέσματα και την μικρή υπολογιστική ισχύ που ξοδεύουν [14].



Εικόνα 18: Σύγκριση μοντέλων [41]

2.7 Αλγόριθμοι ανίχνευσης αντικειμένων

Η ζήτηση για συστήματα και τεχνολογίες που θα εντοπίζουν και θα ανιχνεύουν στιγμιαία αντικείμενα, σε πραγματικό χρόνο από βιντεοληπτικό υλικό έχει αποκορυφωθεί. Η ανίχνευση αντικείμενων έχει κοινά θεωρηθεί, ως ένας συνδυασμός ταξινόμησης εικόνας και εντοπισμού αντικειμένου. Πιο συγκεκριμένα, οι αλγόριθμοι ανίχνευσης αντικειμένων έχουν την δυνατότητα να εντοπίζουν αντικείμενα σε μία εικόνα, να τα οριοθετούν σε πλαίσια και στην συνέχεια να κατηγοριοποιούν το κάθε ένα σε μία κλάση. Κάθε πλαίσιο που εξάγεται, διαθέτει μία τιμή βεβαιότητας, που αναφέρει κατά πόσο [16], [17], [18].



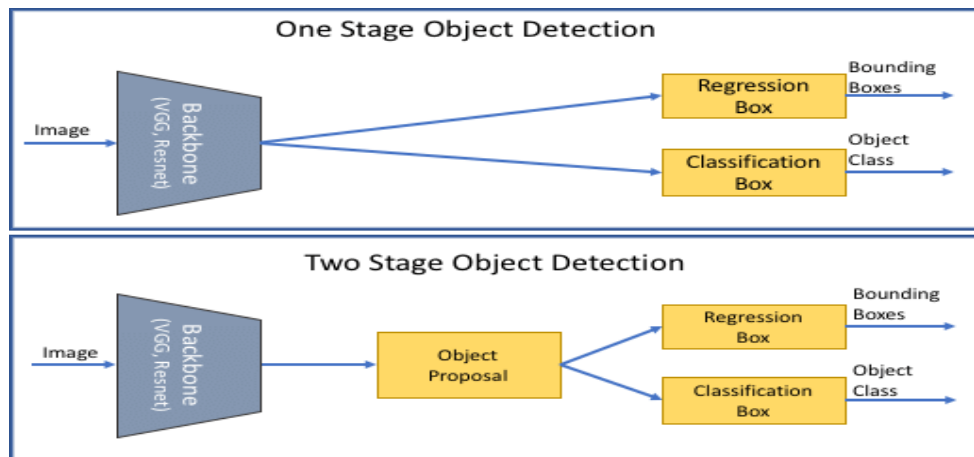
Εικόνα 19: Παράδειγμα ανίχνευσης αντικειμένων [42]

Παρόλα αυτά, υπάρχουν ορισμένα ζητήματα που οφείλουν να αντιμετωπίσουν τα συστήματα αυτά, για την παραγωγή αξιόπιστων και βέλτιστων αποτελεσμάτων. Συγκεκριμένα:

- Πολλές σκηνές από τον πραγματικό κόσμο, μπορεί να είναι απρόβλεπτες σε σχέση με τον φωτισμό τους, την κλιμάκωση καθώς και την απόσταση του ενδιαφερόμενου αντικειμένου από τον φακό της κάμερας.
- Οι ρυθμίσεις και οι ποικιλομορφία των καμερών διαφέρουν αισθητά, στην πλειοψηφία των περιπτώσεων.
- Αρκετά αντικείμενα παρεμβαίνουν και καλύπτουν μέρος ενός άλλου αντικειμένου, καθιστώντας την ταξινόμηση αβέβαιη.

- Οι περισσότερες εφαρμογές πλέον, απαιτούν στιγμιαία και συγχρόνως αξιόπιστα αποτελέσματα, για την λήψη καθοριστικών αποφάσεων, όπως αυτόνομη οδήγηση, συγκρούσεις οχημάτων, εστίες πυρκαγιών, υγεία κ.α.

Στην υπολογιστική όραση κυριαρχούν δύο θεωρίες αλγορίθμων ανίχνευσης αντικειμένων, οι One-Stage Detection και Two-Stage Detection.



Εικόνα 20: Διαφορές αλγορίθμων ανίχνευσης αντικειμένων [43]

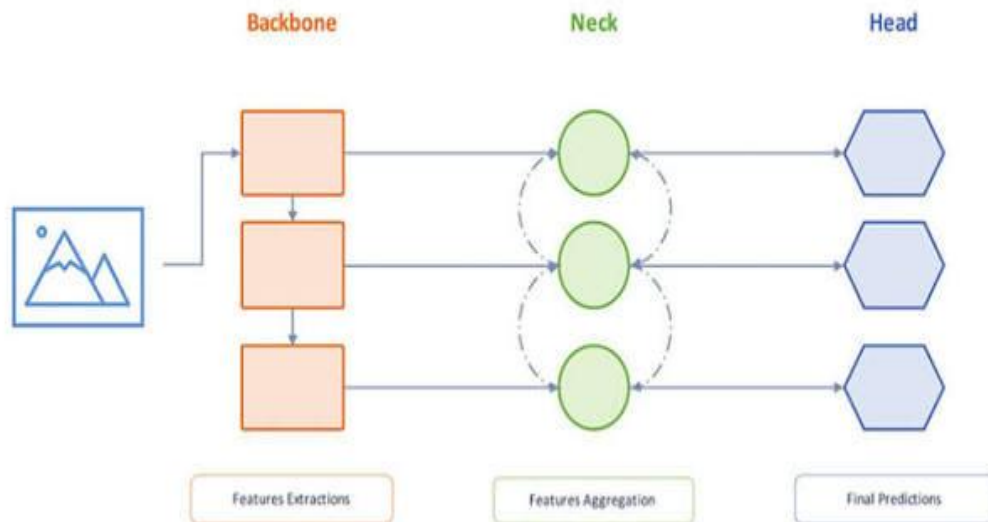
Οι Two-Stage Detection αλγόριθμοι, εστιάζουν την προσοχή τους σε περιοχές ενδιαφέροντος, πριν πραγματοποιηθεί η ανίχνευση. Έπειτα, η κάθε εξαγόμενη περιοχή ενδιαφέροντος αναλύεται ξεχωριστά και παράγεται το επιθυμητό αποτέλεσμα, από την εύρεση και την κατηγοριοποίηση του αντικειμένου. Η υψηλή ακρίβεια που παρουσιάζουν, αποτελεί το πλεονέκτημά τους, ενώ η αχίλλειος πτέρνα τους, καθίσταται η καθυστερημένη ανταπόκριση τους σε πραγματικό χρόνο, καθώς και οι αυξημένες απαιτήσεις που διαθέτουν, όσον αφορά την υπολογιστική ισχύ [16], [17], [18].

Οι One-Stage Detection αλγόριθμοι από την άλλη, πραγματοποιούν όλη την διαδικασία της ανίχνευσης σε ένα μόλις βήμα. Πιο αναλυτικά, τόσο ο εντοπισμός των συντεταγμένων και η δημιουργία πλαισίων όσο και η ανίχνευση ενός πραγματικού αντικειμένου και της ταξινόμησής του, λαμβάνουν χώρα σε ένα βήμα. Η λειτουργία αυτή, καθιστά τους αλγορίθμους αυτούς (YOLO, SSD, RetinaNet, κ.α) εξαιρετικά γρήγορους και κατάλληλους για εφαρμογές σε πραγματικό χρόνο [16], [17], [18].

2.7.1 Αλγόριθμος YOLO (You Only Look Once)

Ο YOLO αλγόριθμος αποτελεί ένα από τα πιο γνωστά και συχνά χρησιμοποιούμενα εργαλεία, για την ανίχνευση αντικειμένων σε πραγματικό χρόνο από βιντεοληπτικό υλικό. Οι οικογένειες των YOLO, συγκροτείται από μία μεγάλη ποικιλομορφία μοντέλων και διαφορετικών αρχιτεκτονικών, που εξελίσσονται με ραγδαίους ρυθμούς τα τελευταία χρόνια. Ανήκει στην κατηγορία των One-Stage Detectors αλγορίθμων και διακρίνεται για την εξαιρετική του ικανότητα, της ισορρόπησης μεταξύ της στιγμιαίας ανταπόκρισης και της πολύ ανταγωνιστικής ακρίβειας που προσφέρει.

Η εικόνα που εισέρχεται ως είσοδος σε οποιοδήποτε από τα σύγχρονα και επίσημα YOLO μοντέλα, διαπερνάτε από τρία διαδοχικά επίπεδα:



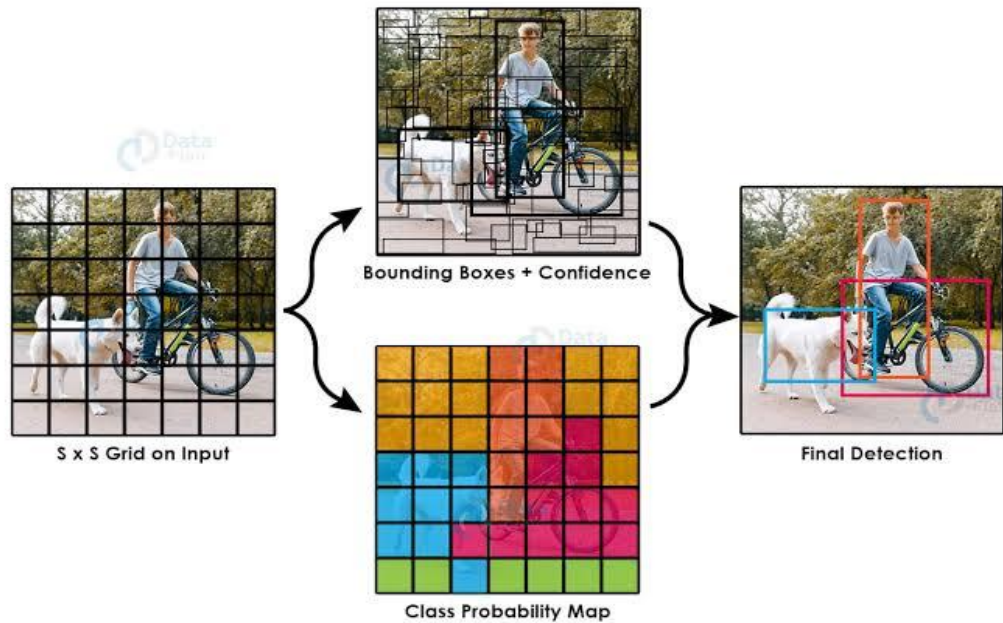
Εικόνα 21: Αρχιτεκτονική YOLO [44]

Backbone: Είναι υπεύθυνο για την εξαγωγή χαρακτηριστικών, από την δοσμένη είσοδο εικόνα, σε πολλαπλές διαφορετικές αναλύσεις της.

Neck: Κάνει μία μίξη των χαρακτηριστικών, που προέρχονται από διαφορετικές διαστάσεις, με σκοπό την ανίχνευση αντικειμένων μικρής και μεγάλης κλίμακας ταυτόχρονα.

Head: Το τελικό στάδιο, όπου το μοντέλο οριοθετεί τα πλαίσια που περικλείουν τα αντικείμενα και προβλέπει την κατηγορία τους δίνοντας μία βαθμολογία, που εξάγεται με βάση τα αναμειγμένα χαρακτηριστικά που αναλύθηκαν.

Από το πρώτο μοντέλο, υιοθετήθηκε μία λογική σχεδιασμού, όπου η εικόνα που εισάγεται ως είσοδος στο μοντέλο και διαιρείται σε πλέγματα, προκαθορισμένου και σταθερού μεγέθους, ως προς τις δύο διαστάσεις ($S \times S$). Το πλέγμα που καλύπτει το κέντρο ενός αντικείμενου, είναι υπεύθυνο για την ανίχνευσή του. Στην ουσία, κάθε πλέγμα εξάγει πέντε κύριες πληροφορίες, οι οποίες συγκροτούν το όριο πλαισίου που περικλείει το αντικείμενο. Οι πληροφορίες αυτές, είναι οι δύο συντεταγμένες του κέντρου πλαισίου σε σχέση με το πλέγμα, οι δύο διαστάσεις ύψος και πλάτος σε σχέση με τις διαστάσεις της συνολικής εικόνας και η βαθμολογία βεβαιότητας, που υποδηλώνει κατά πόσο πιθανή είναι ύπαρξη αντικείμενου και κατά πόσο έχει τοποθετηθεί σωστά το πλαίσιο οριοθέτησης. Η λογική αυτή κάνει το YOLO ένα δίκτυο που διαθέτει την ικανότητα, να εντοπίζει και να ταξινομεί τα αντικείμενα της δοσμένης εικόνας εισόδου στιγμιαία [16], [17], [18].

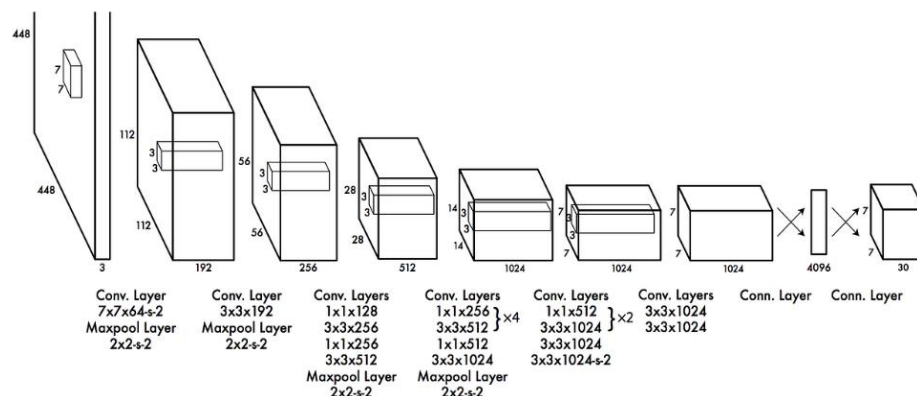


Εικόνα 22: Παράδειγμα YOLO [45]

2.7.2 Μοντέλα YOLO (You Only Look Once)

YOLOv1

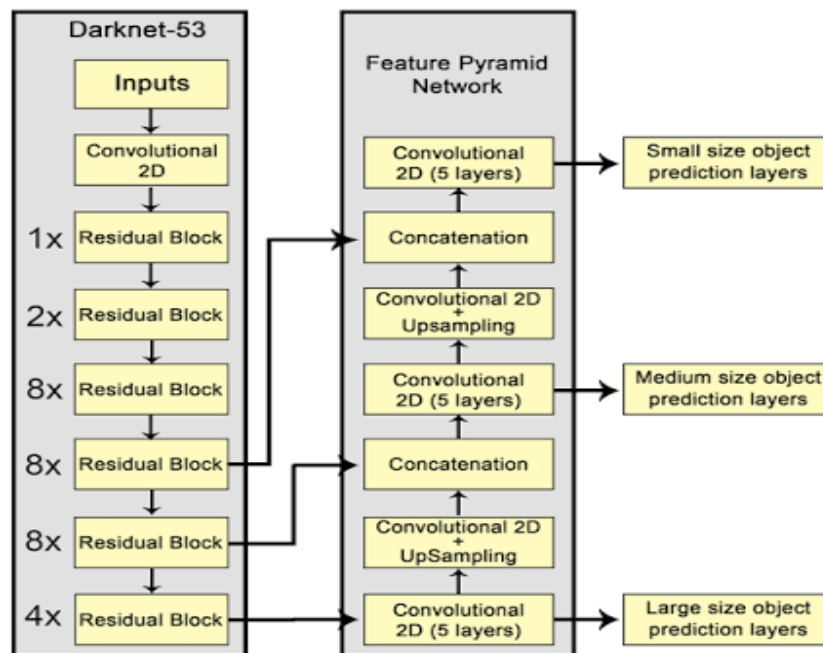
Το YOLOv1 δημιουργήθηκε το 2015 και αποτελεί το μοντέλο που έθεσε τις βάσεις στην οικογένεια του αλγορίθμου. Αποτελείται από 24 συνεκτικά νευρωνικά δίκτυα, τα οποία εξάγουν χαρακτηριστικά από την εικόνα εισόδου και δημιουργούν feature maps. Επιπλέον, ο αλγόριθμος χρησιμοποιεί 4 Max Pooling layers για την κλιμάκωση των διαστάσεων, την γενίκευση του μοντέλου καθώς και για την ενθάρρυνση του μοντέλου να εμβαθύνει σε σημαντικά χαρακτηριστικά. Ακόμη, για την εξαγωγή προβλέψεων υιοθετούνται 2 πλήρως συνδεδεμένα επίπεδα. Στην πλειοψηφία των στρωμάτων του δικτύου, εφαρμόζονται συναρτήσεις ενεργοποίησης Leaky ReLU, για την εισροή μη γραμμικότητας στο δίκτυο αλλά και για την αποφυγή απενεργοποίησης των νευρώνων [16], [17], [18].



Εικόνα 23: Αρχιτεκτονική YOLO1 [49]

YOLOv3

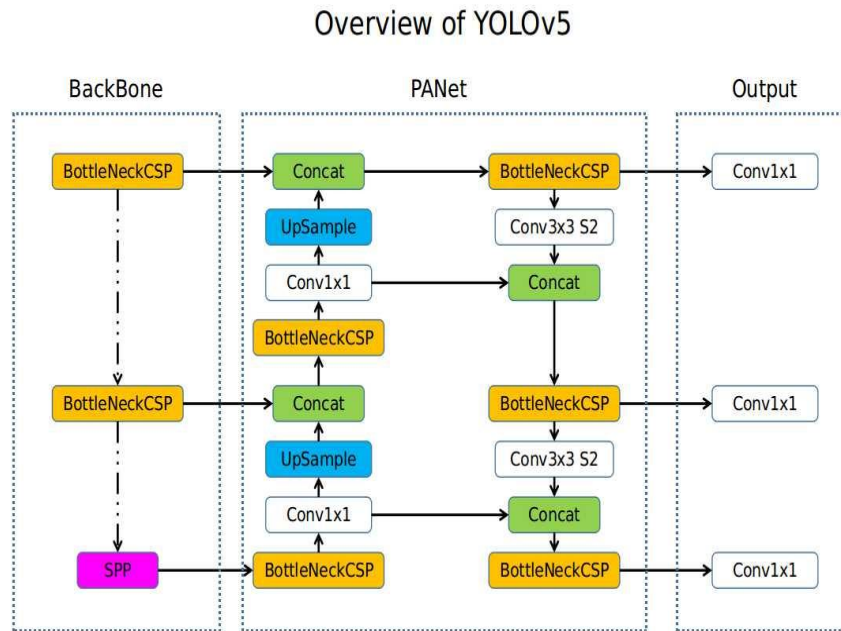
Το YOLOv3 σχεδιάστηκε, ώστε να διορθωθούν τα προβλήματα εντοπισμού και ανίχνευσης σε μικρότερα αντικείμενα, που εμφανίζονταν στα προηγούμενα μοντέλα. Η αρχιτεκτονική του αποτελείται από την χρήση 53 συνελκτικών στρωμάτων, γνωστό και ως Darknet-53. Ακόμη, υιοθετεί τεχνικές όπως residual blocks, skip connections, batch normalization, Leaky ReLU και up-sampling, οι οποίες το καθιστούν ικανό να λειτουργεί αποτελεσματικά σε διαφορετικές κλίμακες. Σε αντίθεση με το YOLOv1, δεν εφαρμόζει Max Pooling layers, καθώς επιτυγχάνει την κλιμάκωση των διαστάσεων με την χρήση stride=2 στα συνελκτικά επίπεδα [16], [17], [18].



Εικόνα 24: Αρχιτεκτονική YOLO3 [48]

YOLOv5

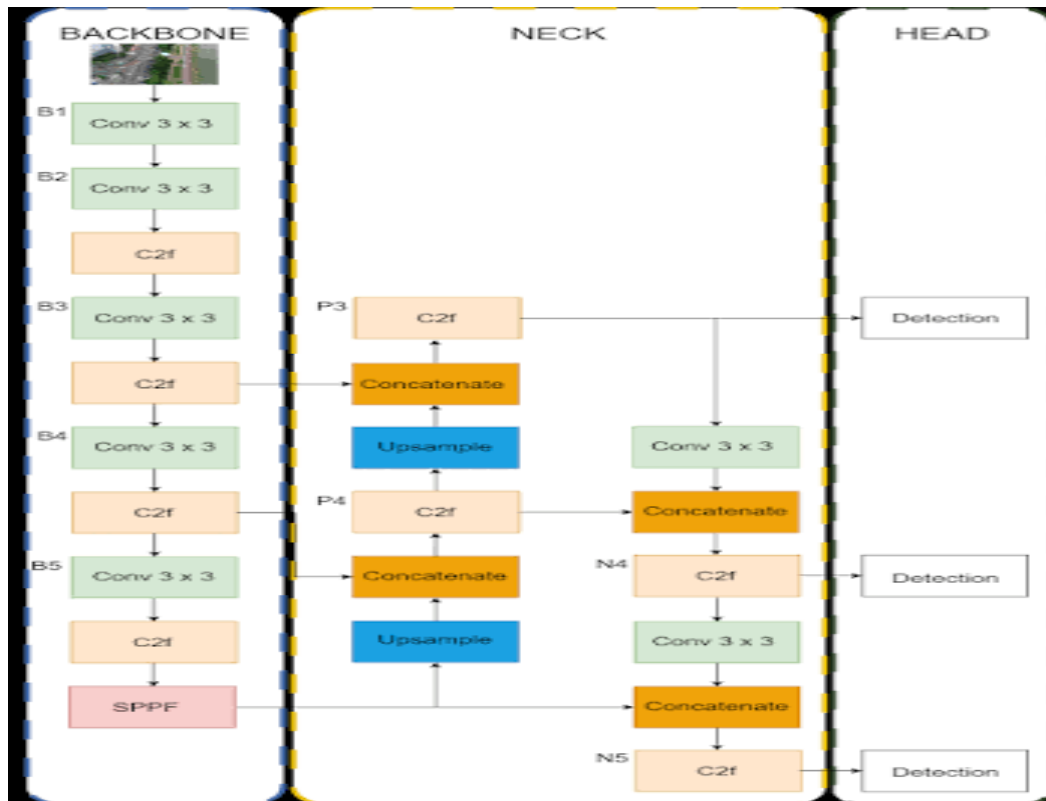
Το YOLOv5 αποτελεί δημιούργημα του Glenn Jocker, ο οποίος υιοθέτησε μια παραλλαγή του Darknet, το CSPDarknet. Στο δίκτυο αυτό, ενσωματώνεται μία τεχνική που διαφοροποιεί τους χάρτες χαρακτηριστικών που εξάγονται σε δύο ξεχωριστά μονοπάτια, καθιστώντας το μοντέλο υπολογιστικά οικονομικότερο, χωρίς απώλειες ακρίβειας. Για την μεταφορά των πληροφοριών σε όλο το δίκτυο, υιοθετείται η τεχνική PANet (Path Aggregation Network) στο μεσαίο επίπεδο του δικτύου neck. Ακόμη, λόγω του SPPF (Spatial Pyramid Pooling Fast) και του Data Augmentation είναι σε θέση να ανιχνεύει αντικείμενα πολλαπλών κλιμάκων. Η συγκεκριμένη υλοποίηση, αναβάθμισε και εκσυγχρόνισε σημαντικά την οικογένεια μοντέλων ανίχνευσης αντικειμένων YOLO [16], [17], [18].



Εικόνα 25: Αρχιτεκτονική YOLO5 [47]

YOLOv8

Ίσως το πιο διαδεδομένο μοντέλο της οικογενείας, λόγω της ευχρηστίας και της εφαρμογής του σε πολλαπλά πεδία πέρα της ανίχνευσης αντικειμένων. Συνεχίζει να διατηρεί τον κορμό του YOLOv5, που αποτελείται από τα backbone, neck και head, με μικρές παραλλαγές και βελτιώσεις. Σε εμβάθυνση, στο backbone έχουν προστεθεί CSP connections, τα οποία συμβάλλουν στην βέλτιστη ροή των gradients και στην εξοικονόμηση ενέργειας. Στο μεσαίο στάδιο, στο neck διατηρείται η τεχνική του SPPF και συνδυάζεται με το PANet για βελτιωμένη ανάμειξη χαρακτηριστικών διαφορετικών κλιμάκων, με πληροφορίες διαφορετικών αναλύσεων. Μία επιπλέον σπουδαία αναβάθμιση που υπάρχει στην αρχιτεκτονική του YOLOv8, είναι η αντικατάσταση της C3 δομής, με αυτήν του C2f. Το C2f μεταδίδει ως έξοδο του bottleneck, έναν συνδυασμό όλων των block του, σε αντίθεση με το C3 όπου η έξοδος του bottleneck, αποτελούνταν μόνο από το τελευταίο block του. Αυτή η καινοτομία, προσφέρει στο επόμενο επίπεδο του δικτύου περισσότερα και πλουσιότερα χαρακτηριστικά. Οι νευρώνες του δικτύου, διαθέτουν μία πιο ολοκληρωμένη εικόνα για την σκηνή που αναπαράγεται γύρω από το αντικείμενο, ενώ ταυτόχρονα η πολυπλοκότητα δεν αυξάνεται σημαντικά. Επιπρόσθετα, όπως και στο YOLOv5, κατά την εκπαίδευση του δικτύου, χρησιμοποιούνται προηγμένες τεχνικές αύξησης δεδομένων, ενισχύοντας με αυτόν τον τρόπο το δίκτυο, με ποικιλομορφία δεδομένων και ανθεκτικότητα. Για την βελτιστοποίηση της οριοθέτησης των αντικειμένων, υιοθετούνται οι τεχνικές CloU (Complete Intersection over Union) και DFL (Distribution Focal Loss), οι οποίες ενισχύουν τον τρόπο με τον οποίο τοποθετούνται τα bounding boxes. Τέλος, η συνάρτηση απώλειας που χρησιμοποιείται είναι η BCE (Binary Cross Entropy), η οποία παρουσιάζει στο μοντέλο το σφάλμα της πρόβλεψής του, σε σχέση με την σωστή ταξινόμηση του αντικειμένου [16], [17], [18].



Εικόνα 26: Αρχιτεκτονική YOLO8 [46]

2.8 Παρόμοιες προσεγγίσεις ανίχνευσης συγκρούσεων

2.8.1 FASTER R-CNN

Το 2019 οι επιστήμονες Kyu Beom Lee και Hyu Soung Shin του KICT (Korea Institute of Civil Engineering and Building Technology) στην προσπάθειά τους να δημιουργήσουν έναν αλγόριθμο όπου θα ανιχνεύει αυτόματα ατυχήματα σε τούνελ μέσα από βιντεοληπτικό υλικό προερχόμενο από συστήματα CCTV προσέγγισαν το θέμα με μία διαφορετική οπτική. Πιο συγκεκριμένα, παρήγαγαν ένα σύστημα όπου χρησιμοποιεί faster R-CNN για την ανίχνευση αντικειμένων και τον αλγόριθμο SORT για την ανάθεση μοναδικής ταυτότητας σε κάθε αντικείμενο που ανιχνεύεται και επιπλέον για την ιχνηλάτησή του. Η λειτουργία του tracking αλγόριθμου SORT, πραγματοποιείται με βάση την σταθερή τιμή της IoU (Intersection over Union) που έχει τεθεί, όπου όταν σε μία δεδομένη χρονική στιγμή t ανιχνευθεί ένα πλαίσιο αντικειμένου, στον επόμενο αμέσως χρόνο $t+r$ το ίδιο αντικείμενο πλαισίου θα θεωρηθεί αυτό, που έχει την μεγαλύτερη επικάλυψη IoU με το προηγούμενο και για αυτόν τον λόγο, θα του ανατεθεί η ίδια ταυτότητα. Για την πρόβλεψη της θέσης του επόμενου πλαισίου, συμβάλλουν τεχνικές όπως, Hungarian αλγόριθμος και Kalman φίλτρο. Η πρόβλεψη του ατυχήματος γίνεται με βάση την επικάλυψη δύο ή περισσότερων πλαισίων, με την απότομη ελάττωση της ταχύτητας, καθώς και με το IoL (Intersection over Line) όπου χρησιμοποιείται για την αναγνώριση απρόβλεπτης αλλαγής κατεύθυνσης του οχήματος. Η εκπαίδευση χρειάστηκε 60 ώρες και ένα σύνολο δεδομένων περίπου 70000 χιλιάδων εικόνων, εξαγόμενες από βίντεο. Τα αποτελέσματα του συστήματος είναι πάρα πολύ ικανοποιητικά, καθώς φαίνεται πως αναγνωρίζει ένα ατύχημα μέσα σε 2 δευτερόλεπτα, ενώ παράλληλα

αναγνωρίζει σε περίπου 8 δευτερόλεπτα οχήματα με λανθασμένη κατεύθυνση κίνησης. Σίγουρα, αποτελεί ένα αποδοτικό σύστημα με οριακά αποδεκτές καθυστερήσεις λόγω της φύσης του (Two-Stage Detector), που χρειάζεται όμως μεγάλη επάρκεια υπολογιστικών πόρων και δεδομένων [20].

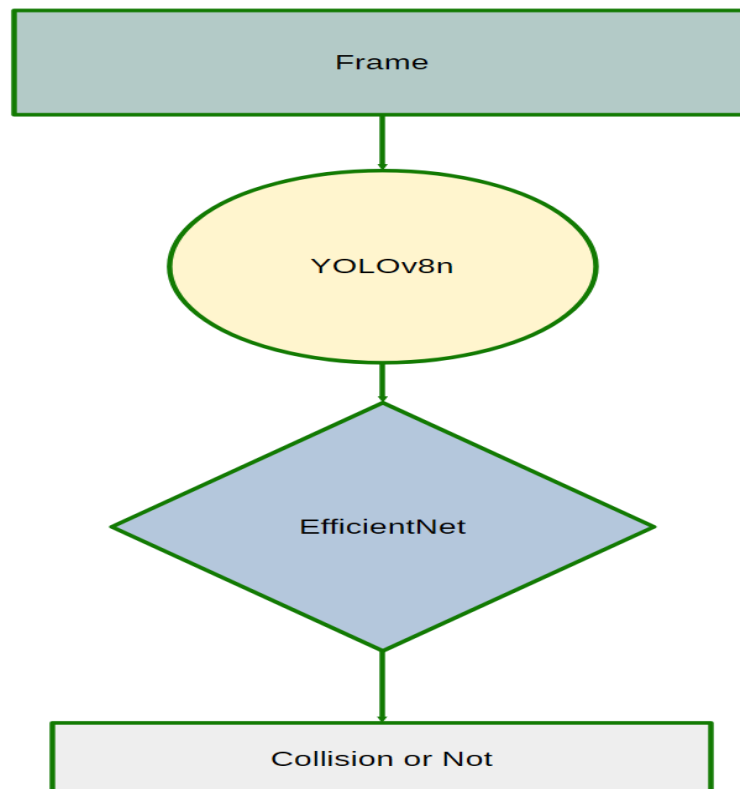
2.8.2 CNN-LSTM

Μία μεθοδολογία που προτάθηκε το 2018 και αναλύεται από τους , Charith Chitraranjan, Vipooshan Vipulanathan, Thunarakan Sritharan, για την πρόβλεψη συγκρούσεων οχημάτων είναι η χρήση ενός CNN, σε συνδυασμό με ένα LSTM. Αναλυτικά, η είσοδος του δικτύου λαμβάνει frames εξαγόμενα από βίντεο, όπου στην συνέχεια διαπερνάνε από τρία στρώματα ConvLSTM, στα οποία πραγματοποιείται η επεξεργασία τους, τόσο στην διάσταση του χώρου, όσο και στο χρόνο. Έπειτα, η πληροφορία αυτή εισρέει σε δύο πλήρως συνδεδεμένα στρώματα για την παραγωγή της πρόβλεψης. Το CNN, είναι υπεύθυνο για την εξαγωγή χωρικών χαρακτηριστικών κάθε frame, ενώ το LSTM λαμβάνει αυτά τα χαρακτηριστικά και μαθαίνει το μοντέλο να αντιλαμβάνεται πως αλλάζει η σκηνή και η κίνηση των οχημάτων σε κάθε frame. Ακόμη, στην αρχιτεκτονική του δικτύου χρησιμοποιούνται πολλαπλά forward passes, καθώς και dropouts για την αποφυγή του overfitting. Το δίκτυο, εκπαιδεύτηκε με την χρήση τεραστίου όγκου συνθετικών εικόνων, από διαφορετικές οπτικές γωνίες και πληροφορίες όπως αισθητήρες αυτοκινήτου. Η ακρίβεια του μοντέλου είναι πάρα πολύ υψηλή, ωστόσο η χρήση συνθετικών δεδομένων, δεν αναδεικνύει το μοντέλο για ουσιαστική χρήση στην καθημερινότητα. Παρόλα αυτά, το πείραμα αποδεικνύει ότι με τα κατάλληλα πραγματικά δεδομένα, την υπολογιστική ισχύ και την υπάρχουσα αρχιτεκτονική που υιοθετήθηκε για την δημιουργία του μοντέλου, μπορούν να καταστήσουν το μοντέλο ικανό, να αναπαράγει βέλτιστα αποτελέσματα [21].

ΚΕΦΑΛΑΙΟ 3: ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

3.1 Εκπαίδευση δικτύου αναγνώρισης συγκρούσεων

Για την υλοποίηση του συστήματος ανίχνευσης συγκρούσεων, χρησιμοποιήθηκε μία μεθοδολογία, όπου συνδυάζει το EfficientNet για την αναγνώριση των συγκρούσεων και το YOLOv8n για τον εντοπισμό των αυτοκινήτων. Στην πράξη, όταν δύο bounding boxes που αναπαριστούν το καθένα από ένα αυτοκίνητο, συγκλίνουν αρκετά με τέτοιο τρόπο, ώστε να υπάρχει σημαντική επικάλυψη μεταξύ τους, δημιουργείται ένα super crop που περιλαμβάνει και περικλείει και τα δύο bounding boxes. Στην συνέχεια, το super crop μεταδίδεται στο εκπαιδευμένο EfficientNet, για την αναγνώριση της συγκρούσεως. Επιπλέον, η υλοποίηση του συστήματος πραγματοποιήθηκε με την χρήση του framework PyTorch.



Εικόνα 27: Διάγραμμα ροής συστήματος

3.1.1 Dataset (Σύνολο Δεδομένων) και Pre-Processing (Προ-επεξεργασία)

Για την εκπαίδευση του EfficientNet, χρησιμοποιήθηκε ένα σύνολο δεδομένων που αποτελείται από δύο κλάσεις Accident και Non Accident. Η κλάση Accident, διαθέτει 1460 cropped frames την στιγμή

της σύγκρουσης των δύο αυτοκινήτων, ενώ η κλάση Non Accident 3657, cropped frames από αυτοκίνητα που κινούνται αρκετά κοντά μεταξύ τους, είτε παράλληλα, είτε σε στροφές ή είναι στοιβαγμένα το ένα πίσω από το άλλο. Τα frames και των δύο κλάσεων έχουν εξαχθεί από ελεύθερα διαθέσιμα dataset στο Kaggle και στο Roboflow και έχουν τροποποιηθεί κατάλληλα για τις ανάγκες του project. Η ανάλυση της εικόνας εισόδου ενός EfficientNet είναι 300x300, οπότε τα cropped frames του dataset διαθέτουν ανάλυση τουλάχιστον από 100x100 και άνω, ώστε κατά την είσοδό τους στο δίκτυο να μην πραγματοποιείται τεράστια παραμόρφωση των εικόνων όπου έπειτα θα δυσχεραίνουν την εκμάθηση και την αποτελεσματικότητα του μοντέλου. Το dataset χωρίζεται σε 80% τις εκατό για την εκπαίδευση του EfficientNet και 20% για την αποτίμηση του. Παράλληλα, υιοθετούνται ορισμένα transforms κατά την εκπαίδευση του δικτύου, για την αύξηση των δεδομένων (Data Augmentation) και την ενίσχυση της ποικιλομορφίας των εικόνων, ώστε να προσεγγίσουν καλύτερα τις συνθήκες λήψης των συστημάτων CCTV.

Ενδεικτικές εικόνες από το σύνολο δεδομένων:



Εικόνα 28: Δείγμα κλάσης Accident



Εικόνα 29: Δείγμα κλάσης Non Accident

Παρακάτω, παρατίθενται οι μετασχηματισμοί εκπαίδευσης του μοντέλου: [21]

```
- transforms.Resize((300,300)),
- transforms.RandomHorizontalFlip(),
- transforms.RandomRotation(15),
- transforms.ColorJitter(brightness=0.3,contrast=0.3,saturation=0.2)
,
- transforms.GaussianBlur(kernel_size=(3,3), sigma=(0.1,2.0)),
- transforms.ToTensor(),
- transforms.RandomErasing(p=0.5,scale=(0.02,0.2),
- ratio=(0.3,3.3),value=0.5),
```

- **Resize((300,300))**: Χρησιμοποιείται για την προσαρμογή των frames του dataset στην ανάλυση εισόδου που απαιτεί το **EfficientNet (300x300)**.

- **RandomHorizontalFlip()**: Πραγματοποιείται τυχαία οριζόντια περιστροφή της εικόνας από αριστερά προς τα δεξιά. Η προκαθορισμένη πιθανότητα περιστροφής μίας εικόνας είναι **50%**. Η συγκεκριμένη λειτουργία βοηθάει το μοντέλο να γενικεύσει μία σκηνή και προς τις δύο κατευθύνσεις ενός ατυχήματος ή μίας κανονικής οδήγησης.

- **RandomRotation(15)**: Μία εικόνα περιστρέφεται τυχαία μεταξύ ενός εύρους μοιρών από **-15** έως **+15**. Όπως και στο προηγούμενο **transform**, προσφέρονται διαφορετικές οπτικές και κατευθύνσεις μίας σκηνής.

- **ColorJitter(brightness=0.3,contrast=0.3,saturation=0.2)**: Η φωτεινότητα και η αντίθεση της εικόνας αλλάζουν τυχαία σε ένα εύρος τιμών από **0,7** έως **1,3**, ενώ ο κορεσμός της αλλάζει από **0,8** έως **1,2**. Αυτές οι παραλλαγές στις εικόνες βοηθάνε το μοντέλο να προσαρμοστεί σε συνθήκες φωτισμού και σε ρυθμίσεις συστημάτων CCTV.

- **GaussianBlur(kernel_size=(3,3), sigma=(0.1,2.0))**: Το **Gaussian Blur** βοηθάει το μοντέλο να αγνοήσει τον θόρυβο του background και να εστιάσει σε σχήματα με μεγάλο όγκο όπως τα αυτοκίνητα. Το μοντέλο γίνεται πιο ανθεκτικό και αποσπά περισσότερες πληροφορίες από τα αντικείμενα ενδιαφέροντος. Στην ουσία, ομαλοποιεί την εικόνα μέσω της θόλωσης της χρησιμοποιώντας ένα **kernel 3x3** και το **sigma** σε εύρος από **0.1** έως **2.0**(όσο πιο μεγάλη η τιμή τόσο αυξάνεται και η θόλωση της εικόνας).

- **ToTensor()**: Μετατρέπει κάθε **PIL** εικόνα σε **tensor** για τις ανάγκες του **PyTorch**. Το **PyTorch** λειτουργεί με **tensors** και απαιτεί τις εικόνες σε μορφή κανάλι, ύψος, πλάτος (**C,H,W**) από **H,W,C**, ενώ παράλληλα γίνεται κανονικοποίηση κάθε **pixel** της εικόνας σε **0** έως **1**.

- **RandomErasing(p=0.5,scale=(0.02,0.2), ratio=(0.3,3.3),value=0.5)**: Αυτή η τεχνική αύξησης δεδομένων χρησιμοποιείται για να βοηθηθεί το μοντέλο να εστιάσει σε ολόκληρη την

πληροφορία της εικόνας και όχι σε μεμονωμένα χαρακτηριστικά της. Συγκεκριμένα, διαλέγονται τυχαία ορθογώνια σε μία εικόνα που καταλαμβάνουν από **2** έως **20%** της περιοχής της, τα οποία διαθέτουν κάθε φορά διαφορετικό μήκος και πλάτος (από **0.3** έως **3.3**, όσο πιο μικρό τόσο πιο μακροστενές είναι το ορθογώνιο). Το **value=0.5** χρωματίζει το πλαίσιο με γκρι, ένα χρώμα όπου το δίκτυο αγνοεί και δεν συμπεριλαμβάνεται στην εκπαίδευση. Με την διαδικασία αυτή, το μοντέλο προσπαθεί να παραλείψει αντικείμενα που παρεμβαίνουν στην σκηνή της σύγκρουσης ή της ομαλής οδήγησης δύο οχημάτων, με σκοπό την κατανόηση της μορφής του σχήματος μίας πιθανής σύγκρουσης.

- **Normalize(mean=[0.485,0.456,0.406], std=[0.229,0.224,0.225]):** Η κανονικοποίηση αυτή πραγματοποιείται καθώς το **EfficientNet** που χρησιμοποιείται στην εργασία όπως και πολλά άλλα γνωστά προεκπαιδευμένα δίκτυα, έχουν εκπαιδευθεί στο **ImageNet**. Στο **ImageNet**, οι εικόνες έχουν κανονικοποιηθεί σύμφωνα με τον παραπάνω μέσο όρο και **standard deviation** για το **RGB(Red, Green, Blue)**. Για τον λόγο αυτόν, η είσοδος του dataset στο δίκτυο πρέπει να πραγματοποιηθεί υπό αυτές τις συνθήκες.

Στο στάδιο αξιολόγησης και αποτίμησης του μοντέλου δεν πραγματοποιείται **Data Augmentation**, διότι αποσκοπούμε στην αντικειμενική αξιολόγηση της εκμάθησης του μοντέλου σε εικόνες και frames που απεικονίζουν την πραγματικότητα.

Παρακάτω, παρατίθενται οι μετασχηματισμοί για την αξιολόγηση του μοντελού:

```
- transforms.Resize((300,300)),
- transforms.ToTensor(),
- transforms.Normalize(
-     [0.485,0.456,0.406],
-     [0.229,0.224,0.225]
- )
```

Τα **Resize**, **ToTensor()** και **Normalize**, παραμένουν απαραίτητοι μετασχηματισμοί για την ομαλή και βέλτιστη εισροή εικόνων του validation set στο δίκτυο.

Έπειτα, τα δεδομένα εκπαίδευσης και αξιολόγησης ενθυλακώνονται σε **DataLoaders** για την τροφοδότηση του δικτύου. Οι **DataLoaders** αποτελούν μία καινοτομία της **PyTorch**, όπου χωρίζουν τα δεδομένα σε **batches** και τα μεταδίδουν στο δίκτυο, με σκοπό την καλύτερη διαχείριση της μνήμης και των πόρων. Ακόμη, θέτουν σε λειτουργία τα transforms και αναμειγνύουν τα δεδομένα (shuffle) σε κάθε εποχή (epoch), για αντικειμενικότερη και λιγότερο **biased** εκπαίδευση [5].

```
- train_dataloader=DataLoader(
-     data_for_training,
-     batch_size=32,
-     shuffle=True
- )
- test_dataloader=DataLoader(
-     data_for_testing,
-     batch_size=32,
-     shuffle=False
- )
```

3.1.2 Αρχιτεκτονική και Εκπαίδευση

Αρχικά, η επιλογή του **EfficientNet** για την αναγνώριση των συγκρούσεων προηγήθηκε εκ των υπολοίπων pre-trained μοντέλων, καθώς όπως αποδείχθηκε και στην έρευνα των Tan και Le, προσφέρει την ιδανική ισορροπία μεταξύ ακρίβειας και υπολογιστικού κόστους που είναι το παν για ένα σύστημα αναγνώρισης σε πραγματικό χρόνο. Ακριβώς με την ίδια λογική πραγματοποιήθηκε και η επιλογή του μεσαίου μοντέλου της οικογένειας των **EfficientNet**, **B3**.

Η εκπαίδευση του δικτύου χωρίστηκε σε δύο φάσεις. Στην πρώτη φάση, τα επίπεδα εξαγωγής χαρακτηριστικών του **Efficient-B3** παρέμειναν αναλλοίωτα και δεν συμμετείχαν στην εκπαίδευση (backbone freeze), ενώ παραμετροποιήσεις πραγματοποιήθηκαν μόνο στον κατηγοριοποιητή (classifier) του δικτύου για τις ανάγκες του προβλήματος. Στην δεύτερη φάση, όλα τα επίπεδα του δικτύου εκπαιδεύονται μαζί. Η μεθοδολογία αυτή είναι ευρέως γνωστή και ως **Fine-Tuning**. Με την μεθοδολογία αυτή, στην αρχή εκπαιδεύεται μόνο ο **classifier** και σταθεροποιούνται τα **gradients**, ενώ στην συνέχεια εκπαιδεύεται όλο το δίκτυο με σκοπό την βέλτιστη προσαρμογή του στα υπάρχοντα δεδομένα. Το **EfficientNet** έχει προ-εκπαιδευτεί στο **ImageNet** και έχει μάθει να αναγνωρίζει σχήματα και μοτίβα, αλλά δεν έχει ειδικευτεί στην αναγνώριση συγκρούσεων. Για αυτόν τον λόγο, εφαρμόστηκε το **Fine-Tuning**, για την προσαρμογή του **EfficientNet** στην αναγνώριση συγκρούσεων αυτοκινήτων.

```
- effnet_accident_det.classifier=nn.Sequential(  
-  
-     nn.BatchNorm1d(input_f),  
-  
-     nn.Dropout(0.4),  
-     nn.Linear(input_f,256),  
-  
-     nn.ReLU(),  
-     nn.Dropout(0.3),  
-     nn.Linear(256,1)  
- )
```

BatchNorm1d(input_f): Συγκεκριμένα, ο **classifier** λαμβάνει τα εξαγόμενα χαρακτηριστικά από το **backbone** και το **neck** του δικτύου και τα κανονικοποιεί, για την αντικειμενική ταξινόμηση τους, καθώς διαθέτουν ακανόνιστη κατανομή.

Dropout(0.4): Με την εντολή αυτή, απενεργοποιείτε με ακαθόριστο τρόπο το **40%** των νευρώνων. Η λειτουργία αυτή χρησιμοποιείται για την γενίκευση του δικτύου, μέσω της αποφυγής της υπερπροσαρμογής (**overfitting**). Το μοντέλο αναγκάζεται να αντιληφθεί και να κατανοήσει πιο περίπλοκα χαρακτηριστικά και μοτίβα όπως μία σύγκρουση αυτοκινήτων.

Linear(input_f,256): Πραγματοποιείται μείωση των διαστάσεων από **input_f=1536** σε **256**. Το νούμερο των διαστάσεων **256**, αποτελεί μία μέση τιμή μεταξύ των **512** και του **128** που χρησιμοποιείται για την ισορροπία μεταξύ συγκρότησης και απώλειας χρήσιμων πληροφοριών.

ReLU(): Η συνάρτηση ενεργοποίησης **ReLU (Rectified Linear Unit)** αποτελεί μία από τις πιο διαδεδομένες και αποδοτικές μεθόδους για την επίλυση του προβλήματος μικρών **gradients** και της

εισαγωγής μη-γραμμικότητας στο μοντέλο. Η **ReLU** διατηρεί τις θετικές τιμές της εισόδου και μετατρέπει τις αρνητικές σε μηδέν. Με αυτόν τον τρόπο καθίσταται πιο εύκολη η εκμάθηση σύνθετων μοτίβων. [22]

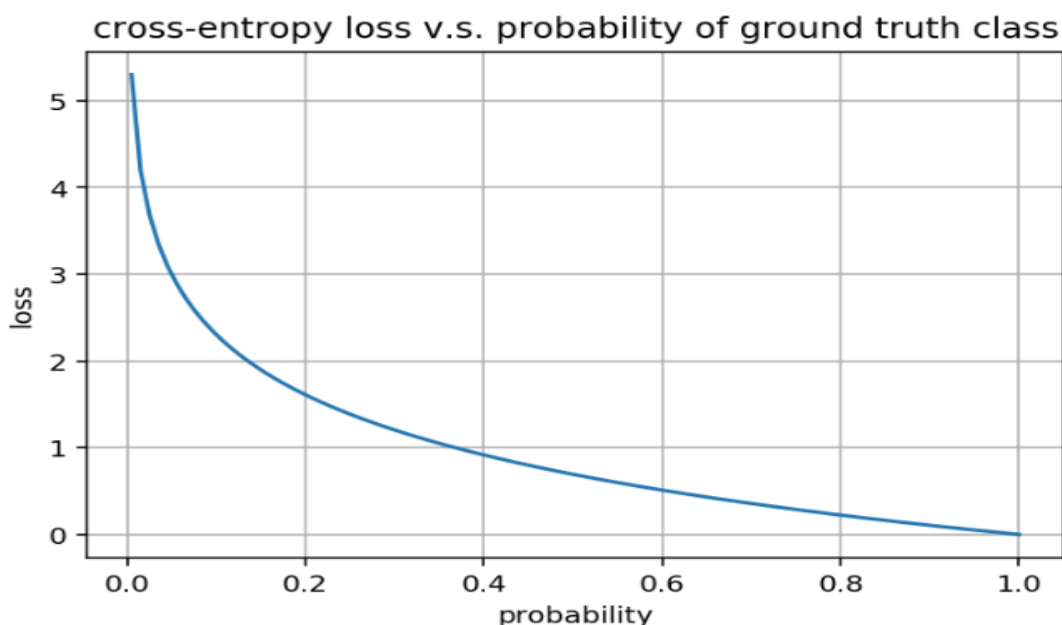
Dropout(0.3): Σε αυτό το σημείο γίνεται απενεργοποίηση των **30%** των νευρώνων καθώς τα πολύτιμα χαρακτηριστικά έχουν ελαχιστοποιηθεί σε **256**.

Linear(256,1): Τέλος, το δίκτυο αποδίδει την έξοδό του σε ένα τελικό μοναδικό αποτέλεσμα, **Accident ή Non Accident**.

Loss Function – Label Smoothing

Στην συνέχεια, στο δίκτυο ως συνάρτησης απώλειας (Loss Function) υιοθετείται η BCE (Binary Cross Entropy Loss) σε συνδυασμό με μία τεχνική, που ονομάζεται Label Smoothing.

Η CE (Cross Entropy Loss) είναι μία συνάρτηση απώλειας, που υπολογίζει την διαφορά μεταξύ της κατανομής της πρόβλεψης του μοντέλου και της κατανομής της πραγματικής κατηγορίας (Ground Truth). Η BCE (Binary Cross Entropy Loss) αποτελεί μία ειδική εφαρμογή της CE, για προβλήματα όπου το μοντέλο είναι αναγκασμένο να προβλέψει μεταξύ δύο κατηγοριών (0 ή 1). Επιπλέον, η BCE προσφέρει λύσεις στο γνωστό πρόβλημα της εξαφάνισης των gradients (τιμωρεί με μεγάλο gradient σε μία λανθασμένη πρόβλεψη) στα Deep Learning μοντέλα, ενώ παράλληλα βοηθάει το μοντέλο να συγκλίνει. Έτσι, η BCE αποτελεί την ιδανική συνάρτηση απώλειας τόσο για την δυαδική πρόβλεψη (Accident/Non Accident) του μοντέλου της εργασίας όσο και για την εύκολη και διαχειρίσιμη υλοποίησή της [26].



Εικόνα 30: Συνάρτηση απώλειας CE [54]

Το Label Smoothing αποτελεί μία ευρέως χρησιμοποιούμενη τεχνική σε πολυάριθμους τομείς του Deep Learning, για την αύξηση της ακρίβειας του μοντέλου. Σε μία υλοποίησή του, ο ερευνητής Szegedy

πρότεινε το Label Smoothing ως μία μεθοδολογία, που βελτιώνει την ακρίβεια του μοντέλου. Στην ουσία, πραγματοποιήθηκε ο υπολογισμός της BCE, χρησιμοποιώντας ένα βεβαρυμμένο συνδυασμό των hard labels με μία ομοιόμορφη κατανομή, ενώ απορρίφθηκε ο υπολογισμός της μόνο με τα hard labels. Ακόμη, η χρήση του Label Smoothing, ενισχύει τους υπολογισμούς του προτελευταίου επιπέδου ενός δικτύου, με σκοπό την παρουσίαση μίας βελτιωμένης εκτίμησης της ορθής κλάσης, ενώ παράλληλα δεν παρουσιάζεται σημαντική απόκλιση από τις υπόλοιπες κλάσεις. Συνοπτικά, η συγκεκριμένη τεχνική αναβαθμίζει την ακρίβεια του EfficientNet και ταυτόχρονα καταρρίπτει την έννοια της απόλυτης σιγουριάς (overconfidence) για μία πρόβλεψη του μοντέλου [24].

```
- bce_ls_function_loss=LF(smooth=0.1,pos_weight=p_w)
```

```
- class LF(nn.Module):
-
-     def __init__(self,smooth=0.1,pos_weight=None):
-         super().__init__()
-         self.sm=smooth
-         self.lsbce=nn.BCEWithLogitsLoss(pos_weight=pos_weight)
-
-     def forward(self,predicts,targ):
-
-         t=targ*(1 -self.sm) + 0.5*self.sm
-         return self.lsbce(predicts,t)
```

Η κλάση **LF** υλοποιεί την συνάρτηση απώλειας BCE, μαζί με την τεχνική Label Smoothing. Με αυτήν την εφαρμογή, το μοντέλο κατά την εκπαίδευση, διατηρεί τα gradients πιο σταθερά, ενώ παράλληλα οι προβλέψεις του γενικεύονται με περισσότερη επιτυχία, για την αποφυγή του overfitting. Το σημείο **smooth=0.1** (10%), αποτελεί την χρυσή τομή για δεδομένα εικόνων που εξάγονται από CCTV με θόρυβο. Το **predicts** αναφέρεται στις προβλέψεις του μοντέλου και το **targ** στις ετικέτες των κλάσεων (0 και 1). Η συνάρτηση επιστρέφει την συνάρτηση απώλειας BCE, μαζί με τις προβλέψεις του μοντέλου και τα soft labels.

Για την επίλυση της ανισορροπίας των δεδομένων μεταξύ των δύο κατηγοριών (Accident=1460, Non Accident=3657), εφαρμόστηκε η τεχνική **pos weight**. Σύμφωνα με τους Ling και Sheng, ανήκει στην κατηγορία των Cost-Sensitive μεθοδολογιών, που αποσκοπούν στην επίλυση αυτού του προβλήματος. Το pos weight, αναθέτει ποινές σε κάθε κατηγορία μέσω ενός πίνακα κόστους. Κατά την εκπαίδευση του δικτύου, για κάθε λανθασμένη πρόβλεψη του μοντέλου με Ground Truth την κλάση μειονότητας, το μοντέλο μέσω του pos weight και της BCE λαμβάνει αυξημένη ποινή, ενισχύοντας έτσι την κλάση με την μειοψηφία δεδομένων. Στην περίπτωση του EfficientNet και της ανίχνευσης συγκρούσεων, η επιλογή του pos weight προηγήθηκε εκ των υπολοίπων, καθώς πέρα από την επίλυση του προβλήματος της ανισορροπίας των δεδομένων (όπου το μοντέλο θα γίνονταν biased), αναδεικνύει την κλάση ενδιαφέροντος Accident, όπου εκ φύσεως αποτελεί μία δύσκολη πρόβλεψη. [25]

```
- accident_fr=len(list(Path(data_path+"/Accident").glob("*")))
- non_accident_fr=len(list(Path(data_path+"/Non_Accident").glob("*")))
- p_w=torch.tensor([non_accident_fr/accident_fr]).to(dvc)
```

Το μοντέλο εστιάζει κατά $c = \text{non_accident_fr} / \text{accident_fr}$ (περίπου 2,5) περισσότερες φορές στην κλάση Accident.

Optimizer (Βελτιστοποιητής)

Για την αποδοτική ενημέρωση των βαρών και των παραμέτρων του δικτύου, αλλά και για την ελαχιστοποίηση της απώλειας, υιοθετήθηκε ο βελτιστοποιητής (Optimizer) Adam (Adaptive Moment Estimation). Αποτελεί έναν από τους πιο γνωστούς, αποτελεσματικούς και γρήγορους optimizers. Ο Adam ενημερώνει τα βάρη του δικτύου συγκροτώντας πληροφορίες τόσο για την κατεύθυνση όσο και για το μέγεθος των gradients των βαρών. Έτσι, αντιλαμβάνεται ποιο πρέπει να είναι το κατάλληλο μέγεθος του βήματος ενημέρωσης. Επιπλέον, ενδείκνυται για δίκτυα που διαθέτουν μεγάλο όγκο παραμέτρων και διαχειρίζονται αυξημένο φόρτο δεδομένων. Τα συγκεκριμένα χαρακτηριστικά τον καθιστούν ιδανικό για την εκπαίδευση του δικτύου της αναγνώρισης συγκρούσεων [23].

Warm-up Phase:

```
- optimizer=torch.optim.Adam(filter(lambda par:
par.requires_grad,effnet_accident_det.parameters()),lr=1e-4)
- sched=torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,'min',patie
nce=2)
```

Fine-Tune Phase:

```
- optimizer=torch.optim.Adam(params=effnet_accident_det.parameters(),lr=
1e-5)
- sched=torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,'mi
n',patience=2)
```

Οι τιμές **lr=1e-4** και **lr=1e-5** αποτελούν το αρχικό βήμα με το οποίο ενημερώνει ο **Adam** τα βάρη. Ακόμη, χρησιμοποιείται scheduler **ReduceLROnPlateau** με **patience=2**, για την μείωση του **lr** με βάση την απώλεια της αξιολόγησης κάθε epoch. Αν δεν υπάρξει βελτίωση της απώλειας της αξιολόγησης για 2 epochs, τότε τίθεται σε λειτουργία η μείωση του **lr**.

Για τον υπολογισμό της ακρίβειας του μοντέλου τόσο για το training, όσο και για το testing εφαρμόστηκε η παρακάτω συνάρτηση:

```
- def calculation_of_accuracy(y_true_labels,y_predicts):
-     correct_preds=torch.eq(y_true_labels,y_predicts).sum().item()
-     calculated_accuracy=(correct_preds/len(y_predicts)*100)
-     return calculated_accuracy
```

όπου, **y_true_labels** οι σωστές ετικέτες του σετ και **y_predicts** οι προβλέψεις του μοντέλου.

Training και Testing Loop

Το μοντέλο αναγνώρισης συγκρούσεων, εκπαιδεύτηκε σε πρώτη φάση για 30 epochs (Warm-Up) και σε δεύτερη φάση (Fine-Tune), για άλλα 30 epochs.

```
- for p1 in effnet_accident_det.parameters():
-     p1.requires_grad=False
- for p2 in effnet_accident_det.classifier.parameters():
```

```
- p2.requires_grad=True
```

Αρχικά στο warm up, οι παράμετροι στο backbone παραμένουν παγωμένοι, ενώ ο κατηγοριοποιητής εκπαιδεύεται κανονικά.

```
- for i in range(1,3):
-     if i==2:
-         print("phase2: fine-tuning")
-         for f in effnet_accident_det.parameters():
-             f.requires_grad=True
-         optimizer=torch.optim.Adam(params=effnet_accident_det.parameters(),lr=1e-5)
-         sched=torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,'min',patience=2)
-         epochs=epochs_fine_tune
-     else:
-         print("phase1: warm-up")
-         epochs=epochs_warm_up
```

Έπειτα στο fine tune, ξεπαγώνεται το backbone και εκπαιδεύεται όλο το δίκτυο όπως είναι. Το δίκτυο ξεκινάει να εκπαιδεύεται για 30 epochs στην warm up φάση, με διαφορετικό lr στον βελτιστοποιητή από ότι στην fine tune φάση όπου εκπαιδεύεται για άλλα 30 epochs. Επίσης και αυτή η φάση, χρησιμοποιεί βελτιστοποιητή με διαφορετικό lr και συγκεκριμένα μικρότερο, για την πιο λεπτομερή εκμάθηση του δικτύου.

```
- for epoch in tqdm(range(epochs)):
-     effnet_accident_det.train()
-     loss_from_training=0
-     acc_from_training=0
-     for batch,(X_training,y_training) in enumerate(train_dataloader):
-         X_training=X_training.to(dvc)
-         y_training=y_training.float().unsqueeze(1).to(dvc)
-         train_logits=effnet_accident_det(X_training)
-         y_train_logsToProbs=torch.sigmoid(train_logits)
-         y_train_probsToLabels=torch.round(y_train_logsToProbs)
-         loss=bce_ls_function_loss(train_logits,y_training)
-         accuracy=calculation_of_accuracy(y_training,y_train_probsToLabels)
-
-         optimizer.zero_grad()
-         loss.backward()
-         optimizer.step()
-         loss_from_training+=loss.item()
-         acc_from_training+=accuracy
-     loss_from_training/=len(train_dataloader)
-     acc_from_training/=len(train_dataloader)
```

Στην συνέχεια, ακολουθεί η εκπαίδευση του δικτύου με την μορφή επανάληψης for. Οι εικόνες σε μορφή logits (**X_training**) και οι ετικέτες τους (**y_training**) φορτώνονται, από τον DataLoader

(**train_dataloader**). Έπειτα, μεταφέρονται στην κάρτα γραφικών (**.to(dvc)**) για την αξιοποίηση της τεχνολογίας **cuda** της **PyTorch**. Το **X_training** περνάει στο μοντέλο **effnet_accident_det** και παράγεται η πρόβλεψη σε μορφή logits (**train_logits**). Τα **train_logits**, με την βοήθεια της συνάρτησης **sigmoid**, μετατρέπονται σε πιθανότητες στο διάστημα [0,1] (**y_train_logsToProbs**) και στην συνέχεια με την **round**, εξάγονται σε ετικέτες 0 και 1 (**0: Accident, 1: Non Accident**). Η ροή συνεχίζεται με τον υπολογισμό της απώλειας **loss**, μέσω της **bce_ls-function_loss** και της ακρίβειας **accuracy**, μέσω της συνάρτησης **calculation_of_accuracy**. Τέλος, με την **optimizer.zero_grad()**, μηδενίζονται τα **gradients** των **parameters** που προπονοούνται, για να μην αλλοιωθεί η εκπαίδευση, ενώ με τις **loss.backward()** και **optimizer.step()**, το δίκτυο ενημερώνεται για το τωρινό υπολογισμένο **loss** του και ο **optimizer** ενεργοποιείται για την βελτιστοποίηση των βαρών με βάση το **loss**, αντίστοιχα.

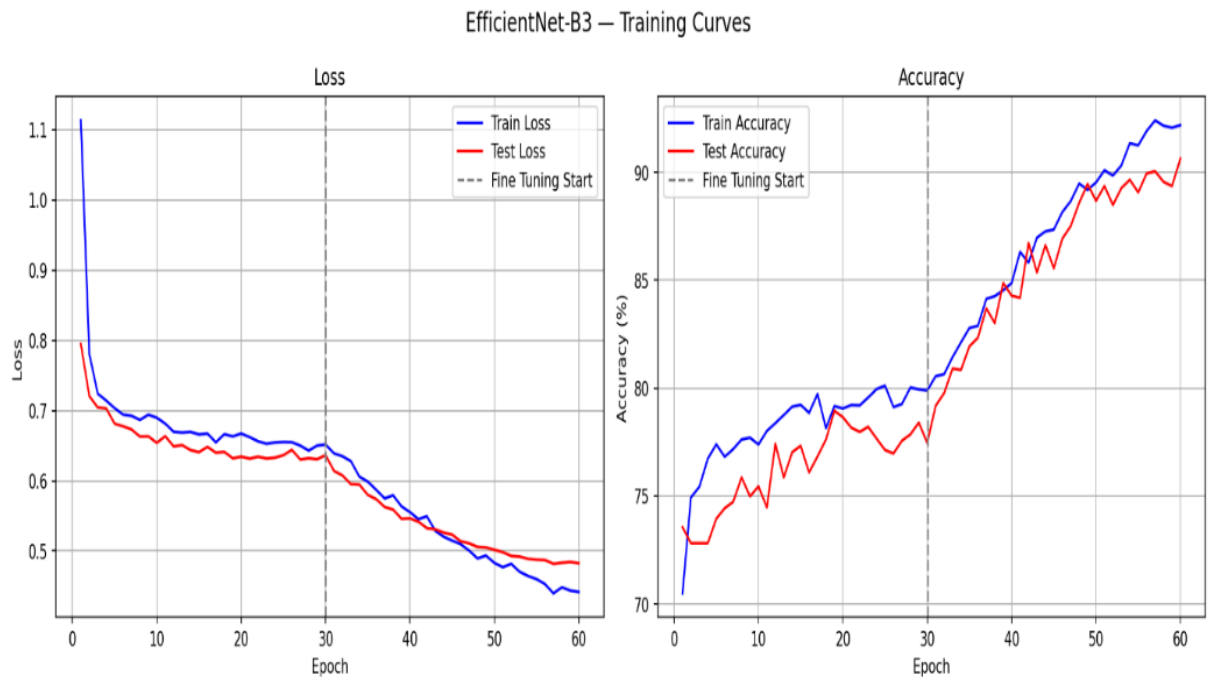
```
-     effnet_accident_det.eval()
-         with torch.inference_mode():
-             for X_testing,y_testing in test_dataloader:
-                 X_testing=X_testing.to(dvc)
-                 y_testing=y_testing.float().unsqueeze(1).to(dvc)
-                 test_logits=effnet_accident_det(X_testing)
-                 test_logsToProbs=torch.sigmoid(test_logits)
-                 test_probsToLabels=torch.round(test_logsToProbs)
-                 loss=bce_ls_function_loss(test_logits,y_testing)
-                 accuracy=calculation_of_accuracy(y_testing,test_probsToLabels)
-
-                 loss_from_testing+=loss.item()
-                 acc_from_testing+=accuracy
-                 loss_from_testing/=len(test_dataloader)
-                 acc_from_testing/=len(test_dataloader)
```

Παρόμοια λογική, υιοθετείται και για την αξιολόγηση του μοντέλου, όπου με την **effnet_accident_det.eval()**, το μοντέλο αποσπάται από την λειτουργία εκπαίδευσης, ενώ ταυτόχρονα με την **torch.inference_mode()**, απενεργοποιείται ο υπολογισμός **gradients** και το μοντέλο προβλέπει με μεγαλύτερη ταχύτητα και οικονομία.

3.1.3 Αποτελέσματα εκπαίδευσης EfficientNet-B3

Το μοντέλο EfficientNet-B3 έπειτα από 30 εποχές Warm-Up και 30 εποχές Fine-Tuning παρουσίασε ορισμένα αποτελέσματα απώλειας και ακρίβειας. Συγκεκριμένα, το μοντέλο εμφάνισε **Training Loss: 0,44** και **Testing Loss: 0,48**, ενώ όσον αφορά την ακρίβεια σημείωσε δείκτες **Training Accuracy: 92,1%** και **Testing Accuracy: 90,6%**. Για τα ευρέως γνωστά σύνολα δεδομένων (CIFAR-10, CIFAR-100, COCO, MNIST) ένα ποσοστό άνω του 90% θεωρείται εξαιρετικό. Στο μοντέλο της παρούσας εργασίας, εκδηλώνονται εξαιρετικά ποσοστά ακρίβειας, για ένα σύνθετο σύνολο δεδομένων με εικόνες εξαγόμενες από CCTV συστήματα, που διαθέτουν θόρυβο. Ακόμη, η διαφορά μεταξύ Training Accuracy και Testing Accuracy (**1,5%**) είναι αρκετά ικανοποιητική, καθώς δεν εμφανίζονται σημάδια υπερ-

εκπαίδευσης (overfitting). Τέλος, όπως παρατηρείται από το παρακάτω γράφημα, τόσο το Training Loss όσο και το Testing Loss, διασχίζουν κοινή πορεία και δεν αποκλείουν σημαντικά μεταξύ τους.



Εικόνα 31: Διάγραμμα ακρίβειας και απώλειας μετά την εκπαίδευση του μοντέλου

Ακόμη, για την περαιτέρω ανάλυση και αξιολόγηση των αποτελεσμάτων, υλοποιήθηκε και ένα Confusion Matrix σε δείγμα 1012 προβλέψεων. Συγκεκριμένα:

True Negative:223 (το μοντέλο πρόβλεψε ότι δεν υπάρχει σύγκρουση και όντως δεν υπήρχε)

False Positives: 83 (το μοντέλο πρόβλεψε ότι υπάρχει σύγκρουση, αλλά δεν υπήρχε)

False Negatives: 13 (το μοντέλο πρόβλεψε ότι δεν υπάρχει σύγκρουση, αλλά υπήρχε)

True Positives 693 (το μοντέλο πρόβλεψε ότι υπήρχε σύγκρουση και όντως υπήρχε)

Για την διαχείριση των αποτελεσμάτων του Confusion Matrix, θα εφαρμοστούν δύο μετρικές κατηγοριοποίησης Precision, Recall και ένας συνδυασμός τους, F1-Score [27].

Precision: Το Precision καθορίζεται από το σύνολο των προβλέψεων του μοντέλου (Accident), πόσες από αυτές πραγματικά ήταν Accident.

■ $Precision = P = TP / (TP + FP)$

Recall: Το Recall από την άλλη συνοψίζεται, από το σύνολο των πραγματικών συμβάντων (Accident), πόσα κατόρθωσε το μοντέλο να εντοπίσει.

■ $Recall = R = TP / (TP + FN)$

Το μοντέλο αναγνώρισης συγκρούσεων, παρουσίασε Precision=**89,3%** και Recall=**98,2%**. Στην ουσία, στις εκατό περιπτώσεις που προβλέπει Accident, οι 89 περιπτώσεις πράγματι είναι και στις εκατό περιπτώσεις πραγματικών Accident, εντοπίζει ορθά τις 98.

Ο συνδυασμός των δύο αυτών μετρικών κατηγοριοποίησης, ονομάζεται F1-Score. Το F1-Score αποτελείται από την ανάμειξη του Precision και του Recall, με βάση του αρμονικού μέσου.

■ $F1\text{-Score} = F1 = 2(P * R)/(P + R)$

Δηλαδή, το μοντέλο παρουσίασε **F1=95%**

Στην περίπτωση της αναγνώρισης συγκρούσεων αλλά και στις περισσότερες εφαρμογές, η αποτελεσματικότητα του μοντέλου δεν αποτυπώνεται αποκλειστικά από το Accuracy του μοντέλου, καθώς τα δεδομένα των δύο κλάσεων δεν είναι ισόποσα. Για αυτόν τον λόγο, δίνεται περισσότερη έμφαση στα Precision, Recall και F1-Score. Τα ποσοστά που αναφέρονται από αυτές τις μετρικές είναι πολύ ενθαρρυντικά. Συγκεκριμένα, το **98%** Recall αναδεικνύει την ικανότητα του μοντέλου να εντοπίζει ατυχήματα. Επιπλέον, ένα σύστημα ασφάλειας όπως η αναγνώριση συγκρούσεων, επιβάλλεται να παρουσιάζει σχεδόν βέλτιστα ποσοστά Recall. Δηλαδή, να εντοπίζει την συντριπτική πλειοψηφία των συμβάντων, με μικρή θυσία σε λάθους συναγερούς.

3.2 Σύστημα ανίχνευσης αναγνώρισης συγκρούσεων

Σε αυτό το σημείο, παρουσιάζεται η μεθοδολογία εύρεσης και ελέγχου πιθανών συγκρούσεων αυτοκινήτων, καθώς και η τροφοδοσία του EfficientNet με cropped frames για την λήψη της τελικής απόφασης.

Καταρχήν, χρησιμοποιήθηκε το YOLOv8n μοντέλο για την ανίχνευση αυτοκινήτων. Προκρίθηκε η συγκεκριμένη επιλογή, καθώς τόσο το YOLOv8 όσο και η nano έκδοσή του προσφέρουν την χρυσή τομή μεταξύ ακρίβειας, κόστους και ανταπόκρισης που απαιτείται σε μία τέτοια εφαρμογή. Έπειτα, υιοθετούνται ορισμένες τεχνικές για την αποδοτική εξαγωγή υποπτων frames. Δηλαδή αυτοκινήτων που προσεγγίζονται επικίνδυνα μεταξύ τους, καθώς και την διαδικασία εισροής τους στο δίκτυο EfficientNet.

Χρησιμοποιείται η βιβλιοθήκη OpenCV, για το άνοιγμα και την ανάγνωση των frames ενός βίντεο.

```
- input_video_detect_path="clip_03.mp4"
- video_reader=cv2.VideoCapture(input_video_detect_path)
-
- if not video_reader.isOpened():
-     print("error: video was not found ")
-     exit()
- while video_reader.isOpened():
-     r,frame=video_reader.read()
-     if not r:
-         break
```

Με την βοήθεια της **cv2.VideoCapture** της OpenCV, πραγματοποιείται η εκκίνηση και η ανάγνωση του βίντεο **input_video_detect_path**. Ανατίθεται στο **video_reader**, η ανάγνωση του βίντεο και με την εντολή συνθήκης **if not video_reader.isOpened()**, πραγματοποιείται έλεγχος εύρεσης και εκκίνησης του βίντεου. Με τον βρόγχο **while video_reader.isOpened()** και το **r,frame=video_reader.read()** γίνεται η

ανάγνωση των frames με την σειρά, όπου **frame** είναι το frame του βίντεο και **r** ένα Boolean που επιβεβαιώνει ότι η ανάγνωση του κάθε **frame** έγινε επιτυχώς.

Για τις ανάγκες της εφαρμογής, χρησιμοποιήθηκε το προ-εκπαιδευμένο YOLOv8n, για την ανίχνευση αυτοκινήτων.

```
- car_detection_yolo=YOLO('yolov8n.pt')
```

```
- yolov8n_outputs=car_detection_yolo.track(  
-     frame,  
-     classes=[2,5,7],  
-     conf=0.25,  
-     tracker="bytetrack.yaml",  
-     persist=True,  
-     verbose=False  
- )
```

Το YOLOv8n επιστρέφει τα αποτελέσματα του για κάθε **frame** του βίντεο στο **yolov8n_outputs**. Με την ρύθμιση **classes=[2,5,7]**, ο yolo tracker προσπαθεί να ανιχνεύσει αυτοκίνητα (κλάση 2), λεωφορεία (κλάση 5), φορτηγά (κλάση 7), με κατώφλι σιγουριάς 0.25 (**conf=0.25**). Το **tracker="bytetrack.yaml"**, αναθέτει σε κάθε πλαίσιο που εντοπίζει ένα ξεχωριστό ID, ενώ το **persist=True**, διατηρεί το ID κάθε πλαισίου από το προηγούμενο **frame**. Με το **verbose=False**, καθαρίζεται η κονσόλα από συνεχείς πληροφορίες για τα αποτελέσματα κάθε **frame**.

Για την αποτελεσματική υλοποίηση του συστήματος ανίχνευσης συγκρούσεων, υιοθετούνται ορισμένες συναρτήσεις, που επιλύουν σημαντικά τεχνικά ζητήματα και βελτιστοποιούν την απόδοση του συστήματος. Παρακάτω, παρατίθεται μία συνοπτική εξήγηση για την περιγραφή τους και στην συνέχεια θα αναλυθούν σε λεπτομέρεια με την ροή του κώδικα.

rectangles_approaching(): Η συνάρτηση αυτή, χρησιμοποιείται για τον έλεγχο προσέγγισης δύο bounding boxes.

same_vehicle_checking(): Εφαρμόζεται για την επιβεβαίωση, ότι δύο bounding boxes περικλείουν και αποτελούν στην ουσία το ίδιο όχημα.

motion_blur_add(): Με αυτήν την συνάρτηση εισάγουμε motion blur στο **frame**, για τον εμπλουτισμό του **frame** με την έννοια της κίνησης, κατά την σύγκρουση των αυτοκινήτων.

effnet_classification(): Σε αυτήν, πραγματοποιείται η εισροή του **frame** στο μοντέλο **EfficientNet** και εξάγεται η πρόβλεψη του.

creation_of_superbox(): Εφόσον τηρούνται οι προϋποθέσεις της **rectangles_approaching()**, δημιουργείται ένα bounding box (super box), που περικλείει και τα δύο οχήματα.

Για την εισαγωγή της έννοιας της κίνησης στο μοντέλο, υλοποιήθηκε η συνάρτηση **motion_blur_add()**. Επειδή, η σύγκρουση αυτοκινήτων είναι ένα γεγονός που χαρακτηρίζεται από την κίνηση των εμπλεκόμενων αυτοκινήτων, υιοθετήθηκε η ιδέα εμφάνισης motion blur στο **frame**. Συγκεκριμένα, πραγματοποιήθηκε η αφαίρεση των pixels μεταξύ του προτελευταίου **frame** με του τελευταίου και έπειτα εφαρμόζεται ανάμειξη της διαφοράς των δύο προηγούμενων **frame** με του

τωρινού. Το αποτέλεσμα της ανάμειξης, είναι ένα **frame** που περιλαμβάνει ένα μεγάλο ποσοστό της πραγματικής εικόνας και ένα ποσοστό motion blur που δημιουργήθηκε από την διαφορά pixels των προηγούμενων frames. Με αυτήν την τεχνική, το μοντέλο λαμβάνει ένα **frame** που περιέχει motion blur και αντιλαμβάνεται ότι υπάρχει μία στοιχειώδη κίνηση κατά την σύγκρουση. Σε περίπτωση, που το μοντέλο λάμβανε μία στατική εικόνα, θα κατανοούσε μόνο τα σχήματα των αυτοκινήτων, ενώ τώρα, αντιλαμβάνεται και τα σχήματα και την κίνηση της σύγκρουσης.

```
- def motion_blur_add(buff_of_frames,present_frame):  
-     if len(buff_of_frames) <2:  
-         return present_frame  
-     pxl_abs_dif=cv2.absdiff(buff_of_frames[-2],buff_of_frames[-1])  
-     motion_blur_frame=cv2.addWeighted(present_frame,0.6,pxl_abs_dif,0.  
4,0)  
-     return motion_blur_frame
```

Η συνάρτηση λαμβάνει, μία ουρά με τα προηγούμενα frames την **buff_of_frames** και το τωρινό **frame** (**present_frame**). Εάν δεν επαρκούν τα προηγούμενα frames (**if len(buff_of_frames)<2**), επειδή το βίντεο μόλις άρχισε, τότε αναγκαστικά χρησιμοποιείται το υπάρχον **frame** όπως είναι. Στην περίπτωση, που το βίντεο έχει προχωρήσει έστω και ελάχιστα και τα frames είναι αρκετά, υλοποιείται η τεχνική. Με την **pxl_abs_dif=cv2.absdiff(buff_of_frames[-2],buff_of_frames[-1])**, υπολογίζεται η απόλυτη διαφορά μεταξύ των δύο προηγούμενων frames και δημιουργείται το motion blur που επιθυμείτε. Τέλος, με την **motion_blur_frame=cv2.addWeighted(present_frame,0.6,pxl_abs_dif,0.4,0)** εφαρμόζεται η ανάμειξη της διαφοράς (40%) με του **present_frame** (60%), χωρίς bias. Το τελικό αναμειγμένο **motion_blur_frame** επιστρέφεται.

```
- differ_buffer=deque(maxlen=3)
```

Η ουρά με μέγιστο μήκος 3 frames. Το **deque**, υποδηλώνει ότι σε περίπτωση που η ουρά είναι γεμάτη και πραγματοποιείται εισαγωγή ενός **frame**, τότε διαγράφεται από την ουρά το πιο παλιό **frame**.

```
- differ_buffer.append(frame.copy())  
- if len(differ_buffer) >=2:  
-     effnet_frame=motion_blur_add(differ_buffer,frame)  
- else:  
-     effnet_frame=frame
```

Διατηρείται αντίγραφο του **frame** για την εισαγωγή του στην ουρά **differ_buffer**, προτού τεθεί υπό επεξεργασία. Καλείται η συνάρτηση, όταν υπάρχουν διαθέσιμα frames, αλλιώς η συνέχεια πραγματοποιείται με το υπάρχον **frame** αυτούσιο. Το **effnet_frame**, είναι αυτό που θα σταλεί στο **EfficientNet**.

Στην συνέχεια, δημιουργείται ένα άδειο σύνολο **ids_present**, το οποίο θα δέχεται τα ids των αυτοκινήτων που βλέπει αυτήν την στιγμή το **YOLOv8n** (στο υπάρχον **frame**).

```
- ids_present=set()
```

Έπειτα, φιλτράρονται τα πιθανά errors των outputs του **YOLOv8n**, και αποθηκεύονται τα id στο και οι συντεταγμένες των bounding boxes.

```
-         if yolov8n_outputs[0].boxes.id is not None:
-             detected_ids=yolov8n_outputs[0].boxes.id.cpu().numpy().astype(
int)
-             cords_xyxy=yolov8n_outputs[0].boxes.xyxy.cpu().numpy()
```

Τα id κρατούνται στο **detected_ids** και οι συντεταγμένες στο **cords_xyxy**. Πραγματοποιείται η μεταφορά τους από την GPU στην CPU, από **PyTorch tensor** σε **Numpy** πίνακα και από τα **ids** από float σε integers (ακεραίους).

Στην συνέχεια, με την εντολή επανάληψης for, γίνεται ενημέρωση του συνόλου **ids_present** με τα **id** που εντοπίζει το YOLOv8n στο τρέχον **frame**.

```
-         for id,rec_b in zip(detected_ids,cords_xyxy):
-             x1,y1,x2,y2=map(int,rec_b)
-             if (x2-x1)<50 or (y2-y1)<50:
-                 continue
-             centerx=(x1+x2)//2
-             centery=(y1+y2)//2
-             history_of_ids[id].append((centerx,centery))
-             last_detected_box[id]=(x1,y1,x2,y2)
-             frames_id_miss_now[id]=0
-             ids_present.add(id)
-             cv2.rectangle(frame,(x1,y1),(x2,y2),(0,255,0),1)
-             cv2.putText(frame,f"bbox_id:{id}",(x1,y1-
5),cv2.FONT_HERSHEY_SIMPLEX,0.4,(0,255,0),1)
```

Η εντολή **zip(detected_ids,cords_xyxy)**, συνδυάζει τα **id** και τα **coordinates** σε ζευγάρια. Έπειτα, για κάθε **id** και bounding box (**rec_b**), κρατούνται οι συντεταγμένες του πλαισίου σε ακέραια μορφή (δηλαδή pixels), καθώς φιλτράρονται τα bounding boxes που παρουσιάζουν μέγεθος κάτω από 50X50 pixels (**if (x2-x1)<50 or (y2-y1)<50**). Αυτή η διάκριση, αποτρέπει την ανάλυση πλαισίων που εμφανίζονται μακριά από το πεδίο λήψης, που διαθέτουν χαμηλή ποιότητα ανάλυσης και το μοντέλο αδυνατεί να τα επεξεργαστεί και να αντιληφθεί μία πιθανή σύγκρουση. Ακόμη υπολογίζεται το κέντρο (**centerx και centery**) του bounding box και στην συνέχεια αποθηκεύεται στο **history_of_ids**, για την επεξεργασία του στο μέλλον. Επιπλέον, συγκροτούνται οι συντεταγμένες του κουτιού στην **last_detected_box**, ως η τελευταία εντοπισμένη θέση του του πλαισίου, καθώς και το **id** του πλαισίου στην **ids_present**, ενώ παράλληλα γίνεται αρχικοποίηση του μετρητή **frames_id_miss_now**, διότι το πλαίσιο εντοπίζεται στο τρέχον **frame**. Τα πλαίσια που εντοπίζει το **YOLOv8n** στο υπάρχον **frame**, χρωματίζονται με πράσινο.

Αμέσως μετά, υλοποιείται κα δεύτερη εντολή επανάληψης for, για τον εντοπισμό πλαισίων που χάθηκαν και δεν κατάφεραν να ανιχνευθούν σε μελλοντικά **frame**. Κατά την δημιουργία της εφαρμογής, σε ορισμένες περιπτώσεις παρατηρήθηκε ότι στην διάρκεια της σύγκρουσης δύο αυτοκινήτων, το **YOLOv8n** λόγω της παραμόρφωσης των αυτοκινήτων, δεν εντόπιζε το ένα από τα δύο ή και τα δύο αυτοκίνητα, με σκοπό να μην δημιουργείτε **super box** και να χάνεται η σύγκρουση. Για τον λόγο αυτό, εφαρμόστηκε μία διαχείριση των πλαισίων που χάνονται. Όταν ένα πλαίσιο εξαφανίζεται σε ένα **frame**,

κρατείται η τελευταία εντοπισμένη θέση του και πραγματοποιείται έλεγχος μεταξύ οποιουδήποτε άλλου πλαισίου που είναι κοντά με αυτό.

```
-     for id in list(last_detected_box.keys()):
-         if id not in ids_present:
-             frames_id_miss[id]+=1
-             if frames_id_miss[id]<=15:
-                 x1,y1,x2,y2=last_detected_box[id]
-                 cv2.rectangle(frame,(x1,y1),(x2,y2),(0,255,255),1)
-                 cv2.putText(frame,f" id : {id} (wasnt found
{frames_id_miss[id]}f)",cv2.FONT_HERSHEY_SIMPLEX,0.4,(0,255,255),1)
-             else:
-                 del last_detected_box[id]
-                 del frames_id_miss[id]
```

Πραγματοποιείται σάρωση, για κάθε **id** της **last_detected_box**, η οποία περιλαμβάνει τις τελευταίες γνωστές συντεταγμένες κάθε πλαισίου. Ελέγχεται, αν το **id** της **last_detected_box** υπάρχει και στην **ids_present**, η οποία με την σειρά της περιέχει τα **id** των πλαισίων που βλέπει αυτήν την στιγμή το **YOLOv8n**. Αν δεν ταυτίζονται, τότε σημαίνει ότι σε αυτό το **frame** το **YOLOv8n** έχασε το πλαίσιο και δεν το εντόπισε, οπότε πραγματοποιείται αύξηση του μετρητή απώλειας πλαισίου **frames_id_miss_now**. Αν το πλαίσιο, εξαφανιστεί για λιγότερα από 15 **frames**, κρατούνται οι συντεταγμένες του και χρωματίζεται με κίτρινο πλαίσιο, ως η τελευταία εντοπισμένη θέση του πλαισίου. Αν εξαφανιστεί για παραπάνω από 15 **frames**, τότε διαγράφεται εντελώς.

Στο παρακάτω dictionary **all_correct_bb**, διοχετεύονται όλα τα **id** και τα bounding boxes (**rec_b**), τόσο των πλαισίων στο υπάρχον **frame**, όσο και των πλαισίων που έχουν χαθεί στο υπάρχον **frame**.

```
-     all_correct_bb={}
-     for id,rec_b in last_detected_box.items():
-         all_correct_bb[id]=rec_b
```

```
-     boolean_collision_happened=False
-     ids=list(all_correct_bb.keys())
```

Πλέον, διατίθενται σε μία λίστα **ids**, όλα τα **id** και τα πλαίσια (εντοπισμένα και χαμένα) όπου μπορούν να πραγματοποιηθούν οι απαραίτητες συγκρίσεις μεταξύ τους, για τον εντοπισμό της σύγκρουσης.

Με την χρήση δύο **for**, πραγματοποιείται έλεγχος μεταξύ ζευγαριών πλαισίων, για την δημιουργία ενός **super box** και στην συνέχεια για την εύρεση της σύγκρουσης.

```
-     for i1 in range(len(ids)):
-         for i2 in range(i1+1,len(ids)):
-             id_a=ids[i1]
-             id_b=ids[i2]
-             b1=all_correct_bb[id_a]
-             b2=all_correct_bb[id_b]
```

Κάθε ζευγάρι πλαισίων ελέγχεται από μία φορά.

```
- if same_vehicle_checking(b1,b2):  
-     continue
```

Έπειτα, τα δύο πλαίσια εισέρχονται στην συνάρτηση **same_vehicle_checking(b1,b2)** και πραγματοποιείται μία διαδικασία ελέγχου, η οποία διαπιστώνει αν τα δύο πλαίσια αντιπροσωπεύουν το ίδιο αυτοκίνητο. Αν αναφέρονται στο ίδιο αυτοκίνητο, η ερεύνα συνεχίζεται στο επόμενο ζευγάρι. Η συνάρτηση αυτή είναι απαραίτητη καθώς, το **YOLO** αρκετές φορές εμφανίζει δύο **bounding box** για το ίδιο αυτοκίνητο και αυτό θα αποτελούσε μεγάλο εμπόδιο για την δημιουργία αξιόπιστων **super box**.

```
- def same_vehicle_checking(b1,b2):  
-     x_point_l=max(b1[0],b2[0])  
-     y_point_t=max(b1[1],b2[1])  
-     x_point_r=min(b1[2],b2[2])  
-     y_point_b=min(b1[3],b2[3])  
-     if x_point_r < x_point_l or y_point_b < y_point_t:  
-         return False  
-     marg_inn=(x_point_r-x_point_l)*(y_point_b-y_point_t)  
-  
-  
-     e1=(b1[2]-b1[0])*(b1[3]-b1[1])  
-     e2=(b2[2]-b2[0])*(b2[3]-b2[1])  
-     return (marg_inn/min(e1,e2))>0.6
```

Το **x_point_l** κρατάει το δεξιότερο σημείο της επικάλυψης μεταξύ των δύο αριστερών συντεταγμένων των πλαισίων (**b1[0], b2[0]**). Αντίστοιχα, το **x_point_r**, κρατάει το αριστερότερο σημείο της επικάλυψης μεταξύ των δύο δεξιών συντεταγμένων των πλαισίων (**b1[3], b2[3]**). Στην ουσία, στην **x_point_l** που αρχίζει η επικάλυψη στον οριζόντιο άξονα και στην **x_point_r** που τελειώνει η επικάλυψη οριζόντια. Από την άλλη, στα **x_point_t** και **x_point_b** αποθηκεύονται τα σημεία που αρχίζει και τελειώνει η επικάλυψη στον κάθετο άξονα αντίστοιχα. Επιπρόσθετα, με τον όρο συνθήκης **if**, πραγματοποιείται έλεγχος τόσο οριζόντια όσο και κάθετα. Αν, το δεξιότερο σημείο της επικάλυψης βρίσκεται πιο αριστερά από το αριστερότερο σημείο ή το κατώτερο είναι πιο πάνω από το υψηλότερο της επικάλυψης τότε στην ουσία δεν υπάρχει κάποια επικάλυψη. Παράλληλα, υπολογίζονται τα εμβαδά των δύο πλαισίων, καθώς και της επικάλυψής τους. Τέλος, πραγματοποιείται υπολογισμός του ποσοστού του μικρότερου πλαισίου που καταλαμβάνεται από την επικάλυψη. Αν το ποσοστό είναι μεγαλύτερο του **60%**, τότε τα δύο πλαίσια απεικονίζουν το ίδιο αυτοκίνητο.

Στην συνέχεια το ζευγάρι πλαισίων, εισέρχεται στην συνάρτηση **rectangles_approaching**.

```
- if not rectangles_approaching(b1,b2,35):  
-     continue
```

Με την συνάρτηση **rectangles_approaching(b1,b2,35)**, τα δύο πλαίσια εξετάζονται αν επικαλύπτονται αρκετά ώστε να δημιουργηθεί ένα **super box** που θα περικλείει και τα δύο πλαίσια (δηλαδή την στιγμή της σύγκρουσης) και θα αποστέλλεται στο **EfficientNet** για την ταυτοποίηση του (**Accident/Non Accident**).

```

- def rectangles_approaching(b1,b2,m):
-
-     b1_x1_min,b1_y1_min,b1_x1_max,b1_y1_max=b1
-     b2_x2_min,b2_y2_min,b2_x2_max,b2_y2_max=b2
-     check_x=(b1_x1_min -m <b2_x2_max) and (b1_x1_max+m >b2_x2_min)
-     check_y=(b1_y1_min -m <b2_y2_max) and (b1_y1_max +m >b2_y2_min)
-
-     return check_x and check_y

```

Γίνεται λήψη των συντεταγμένων κάθε πλαισίου (**x1_min=αριστερά, y1_min=πάνω, x1_max=δεξιά, y1_max=κάτω**) και έπειτα στην **check_x** και στην **check_y** πραγματοποιείται οριζόντιος και κάθετος έλεγχος με επικάλυψη **m=35 pixels**. Τα δύο πλαίσια των αυτοκινήτων πρέπει να είναι ταυτόχρονα κοντά και οριζόντια και κάθετα για να θεωρηθούν ότι τα δύο αυτοκίνητα προσεγγίζονται μεταξύ τους. Με το **b1_x1_min -m <b2_y2_max** πραγματοποιείται επιβεβαίωση ότι το **b1** δεν απέχει πάρα πολύ δεξιά από το **b2**, ενώ με την **b1_x1_max +m >b2_x2_min** ότι το **b1** δεν απέχει πάρα πολύ αριστερά από το **b2** και συνεπώς υπάρχει επαφή των δύο πλαισίων. Ίδια λογική εφαρμόζεται και στον κατακόρυφο άξονα (**b1_y1_min -m <b2_y2_max, b1_y1_max +m >b2_y2_min**). Πρέπει να ισχύουν και οι δύο συνθήκες για κάθε άξονα για να επιστρέψει η συνάρτηση **True**.

Η επικάλυψη **m** των **35 pixels**, υιοθετήθηκε καθώς το YOLO αρκετές φορές εξάγει πλαίσια για τα αντικείμενα που εντοπίζει με **margin error**. Δηλαδή, το πλαίσιο που περικλείει το αυτοκίνητο είναι πολύ στενό και δεν αφήνει περιθώρια ανοχής στην συνάρτηση να ενώσει ένα ζευγάρι πλαισίων. Με το **margin**, είναι δυνατόν να ενωθούν δύο πλαίσια κατά την στιγμή της σύγκρουσης, όπου σε άλλη περίπτωση τα δύο πλαίσια δεν θα ακουμπούσαν πλήρως και θα χανόταν η σύγκρουση. Η υιοθέτηση των **35 pixels**, αποτελεί επιλογή έπειτα από πειραματικές διαδικασίες. Στην ουσία, το **margin** πρέπει να είναι αρκετό ώστε να μην χάνονται συγκρούσεις, ενώ ταυτόχρονα, δεν πρέπει να είναι υπέρογκο ώστε να δημιουργούνται ανούσια **super boxes**.

Η διαδικασία συνεχίζεται με την δημιουργία ενός **super box**, που περιλαμβάνει το ζευγάρι πλαισίων, προϋποθέτοντας ότι το ζευγάρι έχει περάσει με επιτυχία τους απαραίτητους ελέγχους των συναρτήσεων **same_vehicle_checking(b1,b2)** και **rectangles_approaching(b1,b2,35)**. Η υλοποίηση πραγματοποιείται με την συνάρτηση **creation_of_superbox**.

```

- superx1,supery1,superx2,supery2=creation_of_superbox(b1,b2,frame.shape
, pad=10)

```

Η συνάρτηση δέχεται το ζευγάρι πλαισίων, το σχήμα του τρέχον **frame** καθώς και το **padding** που θα υπάρχει γύρω από το **super box** (10 pixels). Η έξοδος της συνάρτησης περιλαμβάνει τις συντεταγμένες του **super box** που δημιουργήθηκε.

```

- def creation_of_superbox(b1,b2,frame_shape,pad=10):
-
-
-     f_hei,f_wid=frame_shape[:2]
-     superx1=max(0,min(b1[0],b2[0]) -pad)
-     supery1=max(0,min(b1[1],b2[1]) -pad)
-     superx2=min(f_wid,max(b1[2],b2[2]) +pad)
-     supery2=min(f_hei,max(b1[3],b2[3]) +pad)
-     return superx1,supery1,superx2,supery2

```

Η συνάρτηση κρατάει το height (**f_hei**) και το width (**f_wid**) του **frame** και στην συνέχεια υπολογίζει τις συντεταγμένες του super box (**superx1**, **supery1**, **superx2**, **supery2**) με padding 10 pixels (**pad=10**) γύρω από το πλαίσιο και στην συνέχεια τις επιστρέφει.

Στην συνέχεια, δημιουργείται το **super box** με βάση τις συντεταγμένες που υπολόγισε η **creation_of_superbox**, πάνω στο **frame** που έγινε προσθήκη **motion blur** (**effnet_frame**, με την βοήθεια της συνάρτησης **motion_blur_add** που αναλύθηκε προηγουμένως για την εισαγωγή κίνησης στο **frame**) και όχι στο αυθεντικό.

```
- effnet_crop=effnet_frame[supery1:supery2,superx1:superx2].copy()
```

Έπειτα, το **motion blur frame** (**effnet_frame**), γίνεται cropped και εξάγεται το **super box** με την αισθητική κίνησης την στιγμή της σύγκρουσης (λόγω **motion blur**).

```
- if effnet_crop.size==0:
-     continue
- effnet_probs=effnet_classification(effnet_crop)
```

Αν το **cropped frame** έχει μηδενικό μέγεθος, παραλείπεται για την αποφυγή πιθανών error. Σε περίπτωση, που όλα κυλήσουν ομαλά το **super box** αποστέλλεται στο **EfficientNet** μέσω της συνάρτησης **effnet_classification**, για την αναγνώριση της σύγκρουσης.

```
- def effnet_classification(frame_cropped):
-     image_to_rgb=cv2.cvtColor(frame_cropped,cv2.COLOR_BGR2RGB)
-     image_tnsr=transformations(image_to_rgb).unsqueeze(0).to(dvc)
-     with torch.inference_mode():
-         predicted_log=effnet_collision_model(image_tnsr)
-         predicted_pr=torch.sigmoid(predicted_log).item()
-     return predicted_pr
```

Το **cropped frame** που εισέρχεται στην συνάρτηση, μετατρέπεται σε **RGB** (Red-Green-Blue) καθώς το **PyTorch** χρησιμοποιεί **RGB** σε αντίθεση με το **OpenCV** (BGR). Στην συνέχεια εφαρμόζονται τα απαραίτητα **transforms (PIL->Resize->Tensor->Normalize)** που είναι τα ίδια που χρησιμοποιήθηκαν και κατά το testing του **EfficientNet**. Η εικόνα με το **unsqueeze(0)**, δέχεται ένα επιπλέον dimension για την εισαγωγή του στο δίκτυο (σε μορφή batch). Έπειτα, με το **torch.inference_mode()** απενεργοποιεί περιττούς υπολογισμούς και η πρόβλεψη πραγματοποιείται ταχύτερα και οικονομικότερα. Τέλος, το **cropped frame** εισάγεται στο **EfficientNet**. Το δίκτυο εξάγει την πρόβλεψη σε μορφή **logits** και με την χρήση της **sigmoid** γίνεται μετατροπή των **logits** σε πιθανότητα μεταξύ **[0,1]**. Όσο πιο κοντά στο **0**, το δίκτυο προβλέπει **Accident**, ενώ όταν προσεγγίζει το **1** προβλέπει **Non_Accident**.

Η τελευταία υλοποίηση του συστήματος, αφορά την τεχνική Time Smoothing. Το σύστημα πριν το Time Smoothing, χρειαζόταν 4 διαδοχικά frames που αξιολογούνταν από το EfficientNet ως Accident, για την επιβεβαίωση μίας σύγκρουσης. Με το Time Smoothing, όταν ένα frame χαρακτηρίζεται ως Accident, ο μετρητής σύγκρουσης αυξάνεται κατά 1, ενώ όταν κάποιο frame δεν αξιολογείται ως Non Accident, μειώνεται κατά 1. Χωρίς την συγκεκριμένη τεχνική, θα πραγματοποιούνταν αρχικοποίηση του μετρητή στο 0 στην εμφάνιση Non Accident frames και έτσι θα χανόταν το progress.

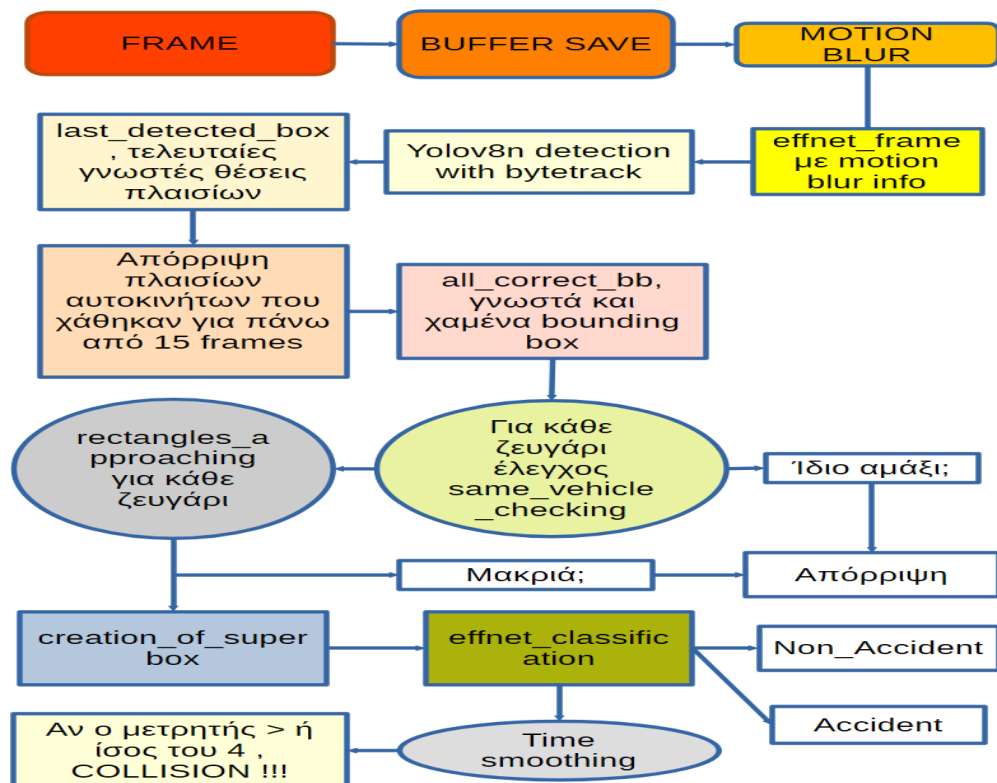
Συγκεκριμένα, το Time Smoothing εισάγει ελαστικότητα στο σύστημα, καθώς δεν απαιτεί αυστηρά 4 διαδοχικά frames για την επιβεβαίωση μίας σύγκρουσης, αλλά στην ουσία, την πλειοψηφία των frames να χαρακτηρίζονται ως Accident. Την στιγμή που ο μετρητής ξεπεράσει το κατώφλι των τεσσάρων frame, μετά τις αυξομειώσεις που έχει υποστεί, τότε το σύστημα μπορεί με βεβαιότητα να επιβεβαιώσει την ανίχνευση μίας σύγκρουσης.

```
- if effnet_probs < 0.40:
-     b=True
```

Σε περίπτωση που οι πιθανότητες που εξάγονται από το EfficientNet (**effnet_probs**), είναι μεγαλύτερες από **60%** για την κλάση Accident, τότε το frame χαρακτηρίζεται ως Accident, αλλιώς ως Non Accident. Στην περίπτωση Accident, τίθεται μία Boolean μεταβλητή **b** ως True. Η μεταβλητή αυτή, βοηθάει στην καταμέτρηση των Accident frames στο Time Smoothing.

```
- if b:
-     coll_counter+=1
- else:
-     coll_counter=max(0,coll_counter -1)
-     coll_counter=min(coll_counter,4 +10)
```

Όταν η μεταβλητή **b** είναι True (δηλαδή το frame είναι Accident), τότε ο μετρητής αυξάνεται κατά 1 (**coll_counter+=1**). Αν όχι, τότε μειώνεται κατά 1 χωρίς να πέσει κάτω από το 0 (**coll_counter=max(0,coll_counter -1)**), ενώ παράλληλα, δεν μπορεί να υπερβεί το όριο των δεκατεσσάρων frames (**coll_counter=min(coll_counter,4 +10)**), ώστε να μην εκπέμπει ανελλιπώς ο συναγερμός.



Εικόνα 32: Ροή ενός frame στο σύστημα

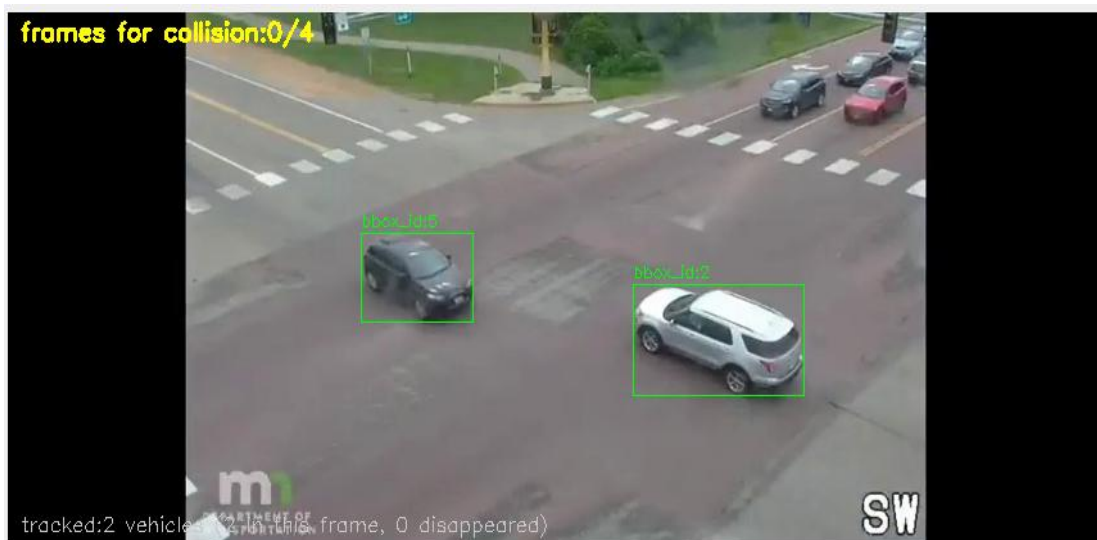
3.3 Αποτελέσματα και δοκιμή συστήματος

Για την δοκιμή και την αξιολόγηση του συστήματος, χρησιμοποιήθηκε βιντεολιπτικό υλικό από συστήματα CCTV, που περιέχει συγκρούσεις αυτοκινήτων. Το σύστημα, αξιολογείται τόσο για την ικανότητά του να εντοπίζει την σύγκρουση που υπάρχει σε κάθε βίντεο, όσο και για την ικανότητα του να εξάγει σωστούς συναγερμούς. Στην πλειοψηφία των περιπτώσεων, το σύστημα εντοπίζει αποτελεσματικό τρόπο τις συγκρούσεις που συμβαίνουν και καταγράφονται από τα συστήματα καμερών CCTV, που βρίσκονται τοποθετημένα σε διασταυρώσεις του οδικού δικτύου.

Παρακάτω, παρατίθενται frames από την δοκιμή του συστήματος και αναλύονται λεπτομερώς, για την αξιολόγηση της υλοποίησης. Τα συγκεκριμένα frames, αναπαριστούν την συμπεριφορά του συστήματος στις εξής περιπτώσεις του συγκεκριμένου βίντεο:

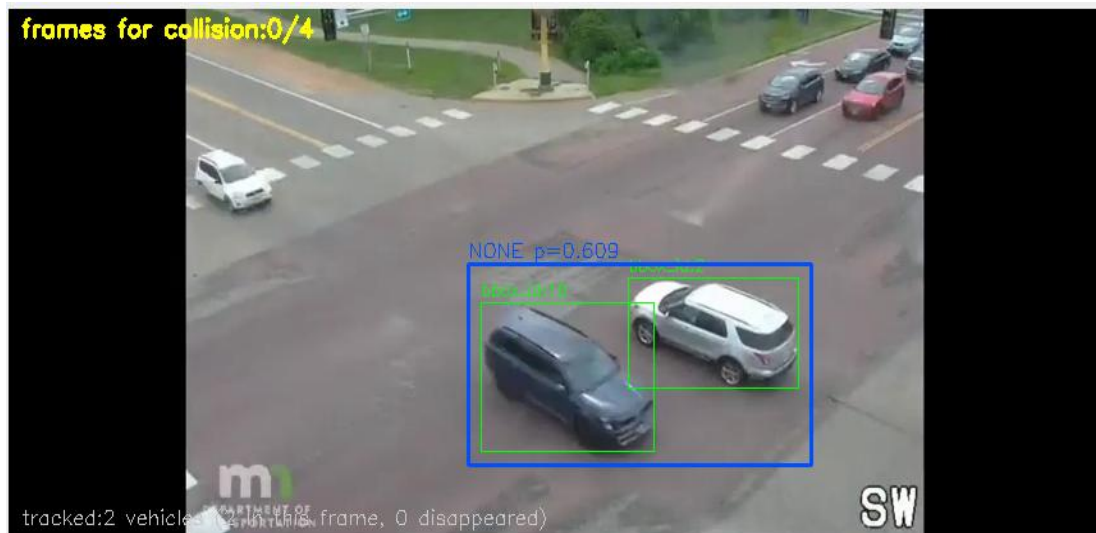
- Σε ομαλή οδική κυκλοφορία
- Την στιγμή της σύγκρουσης
- Αμέσως μετά την σύγκρουση

■ Σε ομαλή οδική κυκλοφορία



Εικόνα 33: Frame από την δοκιμή του συστήματος. Δύο αυτοκίνητα με τα πλαίσια τους

Το YOLO, εντοπίζει τα δύο αυτοκίνητα και τους αναθέτει ξεχωριστό id, ενώ τα bounding boxes χρωματίζονται με πράσινο. Μέχρι στιγμής, τα δύο αυτοκίνητα δεν έχουν πλησιάσει αρκετά, ώστε να δημιουργηθεί ένα super box και έπειτα να αποσταλεί στο EfficientNet για αναγνώριση. Παράλληλα, ο μετρητής σύγκρουσης δεν αυξανέται, καθώς δεν εντοπίζεται κάποιο frame σύγκρουσης μεταξύ των δύο αυτοκινήτων.



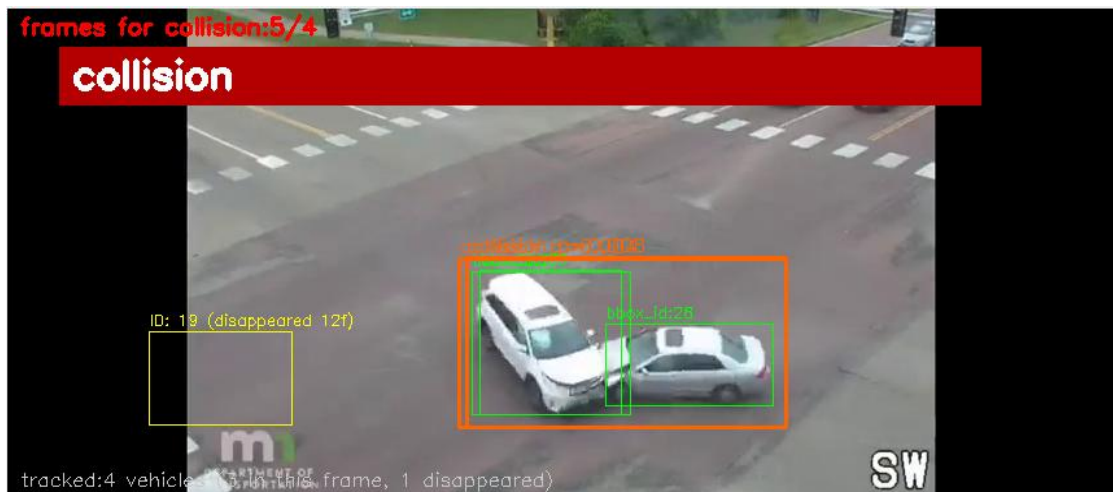
Εικόνα 34: Frame από την δοκιμή του συστήματος. Δύο αυτοκίνητα προσεγγίζονται μεταξύ τους, αλλά το σύστημα δεν βρίσκει σύγκρουση



Εικόνα 35: Frame από την δοκιμή του συστήματος. Διαφορετική περίπτωση αυτοκινήτων που προσεγγίζονται μεταξύ τους, αλλά το σύστημα δεν βρίσκει σύγκρουση

Στην συνέχεια σε δύο παρόμοιες περιπτώσεις, τα δύο αυτοκίνητα προσεγγίζονται τόσο μεταξύ τους, ώστε να δημιουργηθεί super box για τον έλεγχο μίας πιθανής σύγκρουσης. Το superbox χρωματίζεται με μπλε για τον διαχωρισμό του από τα υπόλοιπα πλαίσια. Το EfficientNet πρόβλεψε ότι δεν υπάρχει σύγκρουση με $p=0,609$ και $p=0,788$ αντίστοιχα, ενώ ο μετρητής frames for collision παραμένει ορθά σταθερός στο 0.

- Την στιγμή της σύγκρουσης



Εικόνα 36: Frame από την δοκιμή του συστήματος. Τα αυτοκίνητα συγκρούονται και το σύστημα ορθά εντοπίζει την σύγκρουση

Εδώ παρατηρείτε, ότι το σύστημα εντοπίζει αποτελεσματικά την σύγκρουση των αυτοκινήτων και εμφανίζει το super box που την περικλείει με κόκκινο χρώμα. Ο μετρητής έχει αυξηθεί και έχει περάσει το φράγμα των τεσσάρων frame. Για αυτόν τον λόγο, το σύστημα εκπέμπει συναγερμό. Επιπλέον, παρατηρείτε ένα κίτρινο πλαίσιο, το οποίο αναπαριστά την τελευταία εντοπισμένη θέση ενός αυτοκινήτου που προφανώς εξαφανίστηκε από την σκηνή. Η θέση αυτή, κρατείται στην μνήμη μέχρι τα 15 frames, για μία πιθανή σύγκριση που μπορεί να προκύψει και έπειτα διαγράφεται.



Εικόνα 37: Frame από την δοκιμή του συστήματος. Τα αυτοκίνητα συγκρούονται και το σύστημα ορθά εντοπίζει την σύγκρουση, με τελευταία εντοπισμένη θέση

Λόγω της σύγκρουσης και της παραμόρφωσης των δύο αυτοκινήτων, το αριστερό αυτοκίνητο δεν εντοπίστηκε από το YOLO και έτσι χρησιμοποιήθηκε η τελευταία εντοπισμένη θέση του που εμφανίζεται με κίτρινο χρώμα. Με βάση την εντοπισμένη θέση του δεξιού αυτοκινήτου (πράσινο χρώμα) και της τελευταίας γνωστής (κίτρινο χρώμα), δημιουργείται το super box κόκκινο χρώμα (collision with lost bb). Σύμφωνα με το EfficientNet, το super box με $p=1 - 0,152=0,848$, αναπαριστά σκηνή σύγκρουσης, ενώ και πάλι ορθά ο μετρητής **frames for collision** συνεχίζεται να αυξάνεται.

■ Αμέσως μετά την σύγκρουση



Εικόνα 38: Frame από την δοκιμή του συστήματος. Σκηνή μετά την σύγκρουση των δύο αυτοκινήτων

Με το που τελειώσει η σύγκρουση, ο μετρητής μειώνεται και το super box χρωματίζεται μπλε. Αυτή η συμπεριφορά, αναδεικνύει την ικανότητα του μοντέλου, να εντοπίζει με ακρίβεια την σύγκρουση, την στιγμή που υφίσταται.

■ Σε ομαλή οδική κυκλοφορία



Εικόνα 39: Frame από την δοκιμή του συστήματος. Τα αυτοκίνητα παρουσιάζονται πριν την σύγκρουση μεταξύ τους, σε διαφορετικό βίντεο

Σε διαφορετικό βίντεο, παρατηρείται ότι σε ομαλή οδική κυκλοφορία το σύστημα ορθά δεν εντοπίζει κάποια σύγκρουση και οριοθετεί σωστά τα αυτοκίνητα με τα πλαίσια που εξάγει το YOLO. Παράλληλα, δημιουργείται super box μεταξύ των σταματημένων αυτοκινήτων, που το σύστημα δεν ταυτοποιεί ως σύγκρουση με $p=0,961$.

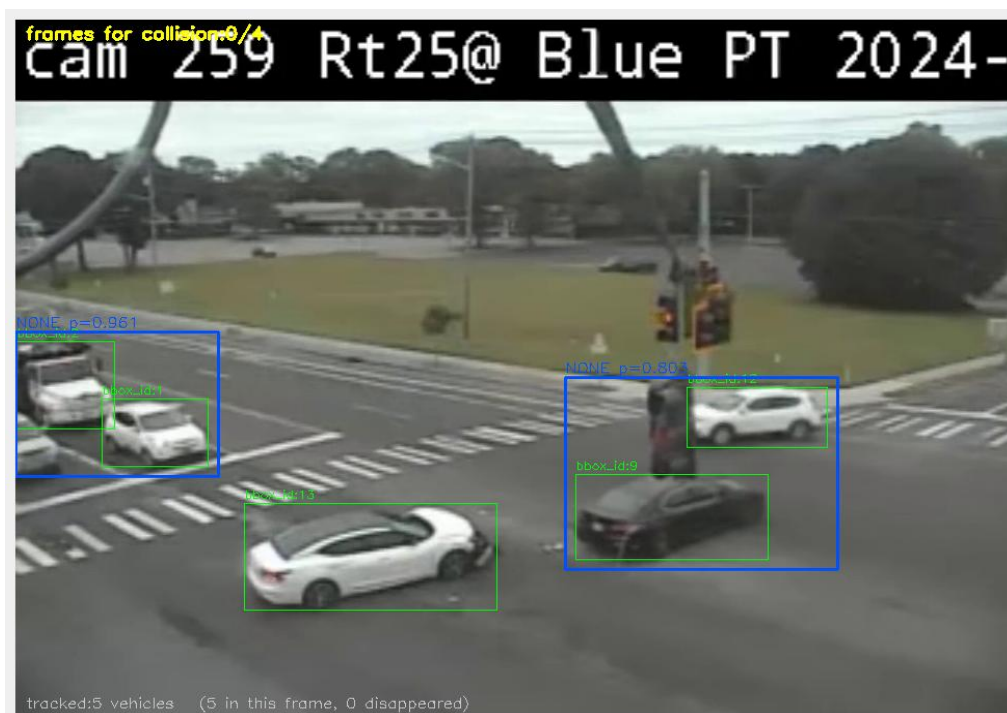
- Την στιγμή της σύγκρουσης



Εικόνα 40: Frame από την δοκιμή του συστήματος. Σκηνή με την σύγκρουση των δύο αυτοκινήτων

Σε διαφορετικό βίντεο, πραγματοποιείται ξανά επιτυχώς η ανίχνευση της σύγκρουσης των δύο αυτοκινήτων με $p=1 - 0,172=0,828$. Το σύστημα έχει εντοπίσει και οριοθετήσει σε πλαίσια, τα δύο εμπλεκόμενα αυτοκίνητα της σύγκρουσης. Το super box που δημιουργείται περικλείει τα δύο αυτοκίνητα και αποστέλλεται στο EfficientNet, για τον εντοπισμό της σύγκρουσης. Ο μετρητής αυξάνεται στα 4 frames και ακαριαία το σύστημα επιβεβαιώνει την σύγκρουση.

- Αμέσως μετά την σύγκρουση



Εικόνα 41: Frame από την δοκιμή του συστήματος. Σκηνή μετά την σύγκρουση των δύο αυτοκινήτων

Μολίς το συμβάν λάβει τέλος και τα δύο αυτοκίνητα απομακρυνθούν, ο μετρητής ελαχιστοποιείται στο 0 και δεν συνεχίζονται να εκπέμπονται συναγερμοί. Τα πλαίσια των αυτόκιντων εξακολουθούν να εντοπίζονται, με το δεξί αυτοκίνητο της σύγκρουσης να δημιουργεί super box με άλλο διερχόμενο αυτοκίνητο, χωρίς να αναγνωρίζεται σύγκρουση με $p=0,803$.

3.4 Συμπεράσματα

Με την χρήση και την εκπαίδευση του δικτύου EfficientNet για την αναγνώριση συγκρούσεων, καθώς και με την χρήση του αλγορίθμου ανίχνευσης αντικειμένων YOLO (You Only Look Once), για την τροφοδοσία του EfficientNet με frames για την ταυτοποίηση μίας πιθανής σύγκρουσης, υλοποιήθηκε ένα αποδοτικό σύστημα αναγνώρισης συγκρούσεων αυτοκινήτων, από υλικό καμερών τύπου CCTV. Επιπλέον, το σύστημα αναγνώρισης συγκρούσεων υιοθετεί ορισμένες τεχνικές, για την περαιτέρω ανάλυση και αξιοποίηση των πλαισίων, που οριοθετεί ο αλγόριθμος YOLO (You Only Look Once). Το σύστημα παρέχει στο EfficientNet πλαίσια, που αναπαριστούν αυτοκίνητα που συγκλίνουν επικίνδυνα μεταξύ τους, με το ίδιο το δίκτυο έπειτα, να είναι σε θέση να τα αναλύσει και να προβλέψει με μεγάλη ακρίβεια μία πιθανή σύγκρουση.

Το υπάρχον σύστημα, είναι ικανό να προσφέρει ένα χέρι βοήθειας προς τις αρχές και τις υπηρεσίες έκτακτης ανάγκης, για την αποτελεσματική αντιμετώπιση των ολέθριων συνεπειών ενός τέτοιου σημαντικού συμβάντος, όπως η σύγκρουση οχημάτων. Ταυτόχρονα, αναδεικνύει την χρησιμότητα των τεχνολογιών της βαθιάς μάθησης (Deep Learning), στην υλοποίηση ωφέλιμων συστημάτων προς την κοινωνία.

3.5 Πιθανές μελλοντικές βελτιώσεις

Το σύστημα, με την πρόοδο της τεχνολογίας είναι ικανό να παράγει αποτελεσματικότερες και ασφαλέστερες προβλέψεις. Συγκεκριμένα, με την εξέλιξη των μοντέλων YOLO για την αναπαραγωγή σταθερότερων και εγκυρότερων πλαισίων οριοθέτησης των οχημάτων, καθώς και με την αναβάθμιση του EfficientNet, για την λεπτομερή ανάλυση και κατηγοριοποίηση των frames. Ακόμη, η περαιτέρω επάρκεια δεδομένων εκπαίδευσης, θα ενισχύσει την ικανότητα του μοντέλου να ταυτοποιεί διαφόρων ειδών συγκρούσεων, όπως συγκρούσεις που περιλαμβάνουν μηχανές, ποδήλατα και οχήματα βαρέους τύπου. Τέλος, ο εκσυγχρονισμός των υπολογιστικών πόρων και των συστημάτων CCTV, θα αποτελέσει την κινητήρια δύναμη για την μελλοντική βελτίωση ενός τέτοιου συστήματος.

BIBΛΙΟΓΡΑΦΙΑ ΚΑΙ ΑΝΑΦΟΡΕΣ

- (1) <https://www.cohenjaffe.com/faqs/what-is-the-definition-of-a-car-accident/>
- (2) <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>
- (3) <https://www.paho.org/en/topics/post-crash-response>
- (4) <https://www.paessler.com/it-explained/cctvs>
- (5) <https://docs.pytorch.org/docs/2.13/data.html>
- (6) Chandreqowda Pachegow (2023). The Global Impact of AI-Artificial Intelligence: Recent Advantages and Future Directions. ArXiv cite as 2401.12223. Department of Professional Security Studies. New Jersey City University.
- (7) Stuart Russel and Peter Norvig (2020). Artificial Intelligence: A Modern Approach.
- (8) Christian Janiesch, Patrick Zschech, Kai Heinrich, (2021). Machine Learning and Deep Learning. ArXiv cite as: 2104.05314 .
- (9) Tala Talaei Khoei, Hadjar Ould Slimane, Naima Kaabouch (2023). Deep learning: systematic review, models, challenges, and research directions. Springer Natural Link.
- (10) Charu C. Aggarwal (2020), Νευρωνικά Δίκτυα και Βαθιά Μάθηση.
- (11) Zhang, Y., & Ling, C. (2018). A strategy to apply machine learning to small datasets in materials science. npj Computational Materials.
- (12) Gabriela Rangel, Juan C. Cuevas-Tello, Jose Nunez-Varela, Cesar Puente, Alejandra G. Silva-Trujillo (2024). A Survey on Convolutional Neural Networks and Their Performance Limitations in Image Recognition Tasks. ResearchGate.

- (13) Keiron O'Shea and Ryan Nash (2015). An Introduction to Convolutional Neural Networks. Department of Computer Science, Aberystwyth University, Ceredigion, School of Computing and Communications, Lancaster University, Lancashire. ArXiv cite as: 1511.08458
- (14) Mingxing Tan, Quoc V.Le. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv cite as: 1905.11946
- (15) Sinno Jialin Pan and Qiang Yang Fellow (2010). A Survey on Transfer Learning,. IEEE Transactions on Knowledge and Data Engineering.
- (16) Ranjan Sapkota, Marco Flores-Calero, Rizwan Qureshi, Chetan Badgujar, Upesh Nepal, Alwin Poullose, Peter Zeno, Uday Bhanu Prakash Vaddevolu, Sheheryar Khan, Maged Shoman, Hong Yan, Manoj Karkee (2025). YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series. Artificial Intelligence Review Springer
- (17) LEO THOMAS RAMO, ANGEL D. SAPPA (2025). A Decade of You Only Look Once (YOLO) for Object Detection: A Review,. Computer Vision Center, Universitat Autònoma de Barcelona
- (18) Nidhal Jegham, Chan Young Koh Marwan Abdelatti, and Abdeltawab Hendawi (2024). YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions. Tunis Business School, University of Tunis, Tunis, Tunisia, Computer Science Dept. University of Rhode IslandDepartment of Computer Science, Providence College.
- (19) Kyu Beom Lee Hyu Soung Shin Department of Extreme Environment Research Center Kyu Beom Lee Hyu Soung Shin (2019). An application of a deep learning algorithm for automatic detection of unexpected accidents under bad CCTV monitoring conditions in tunnels, 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML). Department of Extreme Environment Research Center, Hyu Soung Shin Department of Extreme Environment Research Center Korea Institute of Civil Engineering and Building Technology (KICT)
- (20) Charith Chitraranjan, Vipooshan Vipulananthan, Thuvarakan Sritharan (2025). Vision-Based Collision Warning Systems with Deep Learning: A Systematic Review. Department of Computer Science and Engineering, University of Moratuwa Sri Lanka
- (21) Connor Shorten, Taghi M. Khoshgoftaar (2019). A survey on Image Data Augmentation for Deep Learning. Springer Natural Link

(22) Shiv Ram Dubey , Satish Kumar Singh , Bidyut Baran Chaudhuri (2021). Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. Computer Vision and Biometrics Laboratory, Indian Institute of Information Technology, Allahabad, India

(23) Diederik P. Kingma, Jimmy Lei Ba (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. University of Amsterdam, University of Toronto

(24) Rafael Müller, Simon Kornblith, Geoffrey Hinton (2019). When Does Label Smoothing Help? Google Brain Toronto.

(25) Justin M. Johnson, Taghi M. Khoshgoftaar, (2019). Survey on deep learning with class imbalance. Journal Of Big Data

(26) Caiyi Li, Kaishuai Liu, Shuai Liu (2025). A Survey of Loss Functions in Deep Learning. School of Educational Science, Hunan Normal University, Institute of Interdisciplinary Studies, Hunan Normal University, China.

(27) Peter Christen, David J. Hand, Nishadi Kirielle (2023). A Review of the F-Measure: Its History, Properties, Criticism, and Alternatives. ACM Computing Surveys.

(28) https://www.google.com/imgres?q=cctv&imgurl=https%3A%2F%2Fupload.wikimedia.org%2Fwikipedia%2Fcommons%2Fthumb%2Fa%2Fa1%2FThree_Surveillance_cameras.jpg%2F1280px-Three_Surveillance_cameras.jpg%3Futm_source%3Den.wikipedia.org%26utm_campaign%3Dindex%26utm_content%3Dthumbnail&imgrefurl=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FClosed-circuit_television&docid=2HeZHD5mrCjSJM&tbnid=l0se1-JEzQqn0M&vet=12ahUKEwIU7InUh7mWAXOzAIHHYC9MbcQnPAOegQIThAA..i&w=1280&h=1040&hcb=2&ved=2ahUKEwIU7InUh7mWAXOzAIHHYC9MbcQnPAOegQIThAA

(29) https://www.google.com/imgres?q=automatic%20incident%20detection&imgurl=https%3A%2F%2Fkotaielectronics.com%2Fwp-content%2Fuploads%2F2024%2F06%2F012421dc-a3cd-4d18-a805-18c1bbb11f7d.jpeg&imgrefurl=https%3A%2F%2Fkotaielectronics.com%2Frole-of-ai-in-automatic-incident-detection%2F&docid=vMGr6vdmxh0nUM&tbnid=MmA4sBJZhdGcWM&vet=12ahUKEwiV_Jy1iLmWAXUKhf0HHb8QCMYQnPAOegUI8wQQAA..i&w=765&h=424&hcb=2&ved=2ahUKEwiV_Jy1iLmWAXUKhf0HHb8QCMYQnPAOegUI8wQQAA

(30) https://www.mdpi.com/sustainability/sustainability-14-00210/article_deploy/html/images/sustainability-14-00210-g001-550.jpg

- (31) https://www.researchgate.net/figure/Different-forms-of-Artificial-Intelligence-AI-uses-Modified-and-redrawn-from_fig6_377558634
- (32) [https://www.ntiva.com/hs-fs/hubfs/Machine%20learning%20\(1\).png?width=1200&height=627&name=Machine%20learning%20\(1\).png](https://www.ntiva.com/hs-fs/hubfs/Machine%20learning%20(1).png?width=1200&height=627&name=Machine%20learning%20(1).png)
- (33) <https://bigblue.academy/uploads/images/blog/supervised-learning/classification-vs-regression.jpg>
- (34) https://www.researchgate.net/figure/Supervised-and-unsupervised-machine-learning-a-Schematic-representation-of-an_fig3_351953193
- (35) <https://www.google.com/imgres?q=machine%20learning%20reinforcement%20learning&imgurl=https%3A%2F%2Ftechvidvan.com%2Ftutorials%2Fwp-content%2Fuploads%2F2020%2F08%2FREinforcement-Learning-in-ML-TV.jpg&imgrefurl=https%3A%2F%2Ftechvidvan.com%2Ftutorials%2FREinforcement-learning%2F&docid=Vsfr1URN2XUbKM&tbnid=ckZ7q8Oa1QEedM&vet=12ahUKEwir-eykrkWAXWT0QIHU5dDIEQnPAOegQIWBA..i&w=1200&h=628&hcb=2&ved=2ahUKEwir-eykrkWAXWT0QIHU5dDIEQnPAOegQIWBA>
- (36) <https://cdn.shopaccino.com/igmguru/images/what-is-artificial-neural-network-3350062279421351-12568976359074351.webp>
- (37) https://media.geeksforgeeks.org/wp-content/uploads/20251213152654254872/ai_ml.webp
- (38) https://cdn.builton.com/cdn-cgi/image/f=auto/sites/www.builton.com/files/styles/ckeditor_optimize/public/inline-images/3_what-is-deep-learning.png

- (39) https://media.geeksforgeeks.org/wp-content/uploads/20260217113409515144/working_of_cnn_.webp
- (40) <https://www.mygreatlearning.com/blog/wp-content/uploads/2025/04/how-transfer-learning-work.webp>
- (41) <https://storage.ghost.io/c/2c/8d/2c8d8c0d-1c15-4b6d-825e-02b78d61d40a/content/images/2024/04/image-1085.webp>
- (42) <https://media.cheggcdn.com/media/6a3/6a31dafb-7fb3-4ddd-a64c-5a78f74ead3d/php90cWP2>
- (43) https://www.researchgate.net/figure/Two-stage-vs-one-stage-object-detection-models_fig3_353284602
- (44) https://www.researchgate.net/figure/The-architecture-of-YOLO-consists-of-a-backbone-neck-and-head-5_fig1_386904984
- (45) <https://data-flair.training/blogs/wp-content/uploads/sites/2/2021/05/yolo-cnn-example.jpg>
- (46) https://www.researchgate.net/figure/YOLOv8-Architecture-consists-of-three-main-components-Backbone-Neck-and-Head_fig1_384979324
- (47) <https://user-images.githubusercontent.com/26456083/86477109-5a7ca780-bd7a-11ea-9cb7-48d9fd6848e7.jpg>
- (48) https://www.researchgate.net/figure/Block-diagram-of-YOLOv3-architecture_fig3_346963904
- (49) https://miro.medium.com/v2/resize:fit:720/format:webp/1*U07nDXlgshGNrt8MpBcZ0w.png
- (50) <https://www.hauptman-obrien.net/wp-content/uploads/2026/03/traffic-camera-and-car-accident.jpg>
- (51) https://www.articlecity.com/wp-content/uploads/2020/01/5ee5d5474c5bb114a6d98075c22f357b083edbe25a597048772e7f_640.jpg

(52) Smart Automatic Accident Detection & Information System, 2022, IEEE, Sandeep Rathor, Prateek Dubey, Shivam Samadhiya

(53) https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTSII9Q3MRD35kA4jK8pM84b46NdMTS5vg8RAgemIDOHgQoBi_eUHbgxHKc-&s=10

(54) https://miro.medium.com/1*sJBg-mtHkKCj8nkRhALJ-w.png