Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αλγόριθμοι μηχανικής μάθησης για την αναγνώριση αντικειμένων

Βασίλειος Μπαλάφας (ΑΜ: 1041)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

18 Φεβρουαρίου 2021

Περίληψη

Η ανίχνευση και αναγνώριση των διαφορετικών αντικειμένων σε μια εικόνα γρήγορα και αξιόπιστα είναι ένα σημαντικό πεδίο της υπολογιστικής όρασης. Οι άνθρωποι ρίχνουν μια ματιά σε μια εικόνα και γνωρίζουν αμέσως τι αντικείμενα υπάρχουν στην εικόνα, που βρίσκονται και πως αλληλεπιδρούν. Το ανθρώπινο οπτικό σύστημα είναι γρήγορο και ακριβές, επιτρέποντάς μας να εκτελέσουμε πολύπλοκες εργασίες, όπως οδήγηση με λίγη συνετή σκέψη. Αντιθέτως, το πρόβλημα του εντοπισμού αντικειμένων από έναν υπολογιστή δεν είναι τόσο απλό. Σκοπός αυτής της διπλωματικής εργασίας είναι η παρουσίαση, η ανάλυση και η υπολογιστική σύγκριση αλγορίθμων μηχανικής μάθησης και υπολογιστικής όρασης για την ανίχνευση αντικειμένων σε μια εικόνα ή σε ένα βίντεο.

Λέξεις κλειδιά: Ανίχνευση αντικειμένων, Υπολογιστική όραση, Μηχανική μάθηση, Νευρωνικά δίκτυα, Python

Abstract

The detection and recognition of different objects in an image quickly and reliably is an important field of computer vision. People look at an image and immediately know what objects are in the image, where they are located and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks, such as driving with a little prudent thinking. In contrast, the problem of locating objects from a computer is not so simple. The purpose of this thesis is the presentation and analysis of machine learning and computer vision algorithms for detecting objects in an image or video.

Keywords: Object detection, Computer vision, Machine learning, Neural networks, Python

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Επίκουρο Καθηγητή Νικόλαο Πλόσκα. Η υποστήριξη και η καθοδήγησή του καθ΄ όλη τη διάρκεια της εκπόνησης της εργασίας αποτέλεσε σπουδαία βοήθεια, συμβάλλοντας στην ολοκλήρωσή της, παρέχοντας μεταξύ άλλων ιδιαίτερα χρήσιμες συμβουλές και υποδείξεις.

Θα ήθελα, επίσης, να ευχαριστήσω την οικογένειά μου για την αγάπη και την υποστήριξή τους, χωρίς την οποία δε θα ήταν δυνατή η ολοκλήρωση της παρούσας εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους πολύ καλούς φίλους που απέκτησα κατά τη διάρκεια των σπουδών μου στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών για τη στήριξη και τη συνεργασία όλα αυτά τα χρόνια.

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Αλγόριθμοι μηχανικής μάθησης για την αναγνώριση αντικειμένων" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Νικόλαου Πλόσκα αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν το συγγραφέα και μόνο.

Copyright (C) Βασίλειος Μπαλάφας & Νικόλαος Πλόσκας, 2021, Κοζάνη

Αλγόριθμοι μηχανικής μάθησης για την αναγνώριση αντικειμένων

Βασίλειος Μπαλάφας

ece01041@uowm.gr

19 Φεβρουαρίου 2021

Περιεχόμενα

1 Εισαγωγή **12**

1.1 Περιγραφή του προβλήματος 12

1.2 Στόχος και συνεισφορά της διπλωματικής 12

1.3 Διάρθρωση της εργασίας 13

2 Έννοιες και Ορισμοί **14**

2.1 Τεχνητή Νοημοσύνη 14

2.2 Μηχανική Μάθηση 15

2.3 Βαθιά Μάθηση 16

2.4 Μηχανική Όραση 17

2.5 Νευρωνικά Δίκτυα 19

2.6 Τεχνητά Νευρωνικά Δίκτυα 19

2.7 Συνελικτικά Νευρωνικά Δίκτυα 21

2.7.1 Συνελικτικό επίπεδο / Convolutional layer 23

2.7.2 Συναρτήσεις ενεργοποίησης 23

2.7.3 Επίπεδα χωρικής υποδειγματοληψίας / Pooling layers 26

2.7.4 Πλήρως συνδεδεμένα επίπεδα 27

2.7.5 Επίπεδο μαζικής κανονικοποίησης 27

2.7.6 Επίπεδο κανονικοποίησης 28

2.7.7 Επίπεδα εξόδου 30

2.7.8 Συναρτήσεις απόκλισης 30

2.8 Οπισθοδιάδοση 32

2.9 Παλινδρόμηση 32

2.10 Μηχανές υποστήριξης διανυσμάτων 33

3 Ανίχνευση Αντικειμένων **35**
3.1 Λειτουργίες και τύποι ανίχνευσης αντικειμένων 36
3.2 Πώς λειτουργεί η ανίχνευση αντικειμένων 36
3.3 Πλαίσιο ανίχνευσης αντικειμένων Viola-Jones 37
3.4 Ιστογράμματα Προσανατολισμένων Κλίσεων 38
3.5 Αλγόριθμοι Προτάσεων Περιοχής 41
3.5.1 R-CNN 41
3.5.2 Fast R-CNN 42
3.5.3 Faster-RCNN 44
3.5.4 Mask R-CNN 46
3.6 Ανιχνευτής Πολλαπλών Θυρίδων μιας Λήψης (SSD) 48
3.7 RetinaNet 53
3.8 EfficientDet 58
3.9 YOLOv3 66
3.10 YOLOv4 77

4 Λογισμικά και Τεχνολογίες **84**
4.1 JetBrains PyCharm IDE 84
4.2 Python 84
4.3 Jupyter Notebooks 86
4.4 OpenCV 87
4.5 CUDA 88
4.6 TensorFlow 88
4.7 TensorBoard 89
4.8 Keras 89
4.9 PyTorch 91
4.10 Darknet 91
4.11 Google Colab 91

5 Υπολογιστική μελέτη **93**
5.1 Απαιτήσεις συστήματος 93
5.2 Δημιουργία συνόλου δεδομένων εικόνων για εκπαίδευση 95
5.2.1 Διαθέσιμα σύνολα δεδομένων για εκπαίδευση 95

5.2.2 Προγραμματιστικός τρόπος απόκτησης εικόνων από το Open Images Dataset 97
5.2.3 Το σύνολο δεδομένων της εργασίας 99
5.3 Αλγόριθμοι και πλαίσια εργασίας 100
5.4 Εκπαίδευση ανιχνευτή αντικειμένων με TensorFlow 2 Object Detection API 101
5.5 Εκπαίδευση ανιχνευτή αντικειμένων με Darknet 113
5.6 Μετρήσεις για την αξιολόγηση των ανιχνευτών αντικειμένων 121
5.7 Συγκριτικά στοιχεία 127

6 Συμπεράσματα και μελλοντικές επεκτάσεις 131
6.1 Συμπεράσματα 131
6.2 Μελλοντικές επεκτάσεις 132

Α΄ Ακρωνύμια και συντομογραφίες 136

Κατάλογος σχημάτων

2.1 Επισκόπηση της σχέσης της υπολογιστικής όρασης με την τεχνητή νοημοσύνη και την μηχανική μάθηση. 18
2.2 Διάγραμμα ενός ανθρώπινου νευρώνα [1]. 19
2.3 Σχηματική αναπαράσταση της δομής ενός απλού νευρωνικού δικτύου. 20
2.4 Οι υπολογιστές βλέπουν μια εικόνα ως πίνακα εικονοστοιχείων ο οποίος εξαρτάται από την ανάλυση της εικόνας. Με βάση την ανάλυση της εικόνας, θα δει Ύψος x Πλάτος x Διάσταση. 22
2.5 Οπτικοποίηση ενός CNN για την αναγνώριση χειρογράφων ψηφίων [2]. 22
2.6 Η αριστερή εικόνα δείχνει ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο. Το κάθε επίπεδο συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Η δεξιά εικόνα δείχνει ένα συνελικτικό νευρωνικό δίκτυο. Κάθε επίπεδο έχει νευρώνες διατεταγμένους σε τρεις διαστάσεις με πλάτος, ύψος και βάθος [3]. 23
2.7 (a) Sigmoid (b) Υπερβολική Εφαπτομένη (c) ReLu (d) Leaky ReLu . . 24
2.8 Συνάρτηση Softmax [4]. 25
2.9 Γραφική παράσταση της συνάρτησης ενεργοποίησης Mish 26
2.10 Επίπεδα χωρικής υποδειγματοληψίας ενός CNN [3]. 27
2.11 Παράδειγμα ενός πλήρως συνδεδεμένου επιπέδου [5]. 27
2.12 Η αριστερή εικόνα δείχνει το νευρωνικό δίκτυο χωρίς απόσυρση. Η δεξιά εικόνα δείχνει το νευρωνικό δίκτυο μετά την εφαρμογή της απόσυρσης. Οι νευρώνες αποσύρονται τυχαία [6]. 30
2.13 Παράδειγμα μηχανής υποστήριξης διανυσμάτων. 34

3.1 Παράδειγμα ανίχνευσης αντικειμένων σε μία εικόνα. 35
3.2 Ο αλγόριθμος του πλαισίου ανίχνευσης αντικειμένων Viola-Jones [7]. . 38

3.3 Ιστογράμματα Προσανατολισμένων Κλίσεων: Για κάθε ένα εικονοστοιχείο, κοιτάει και τα εικονοστοιχεία που το περιβάλλουν [8]. . . . 39
3.4 Παράδειγμα σύγκρισης ενός εικονοστοχείου με τα γύρω εικονοστοιχεία για το που γίνονται πιο σκοτεινά [8]. 39
3.5 Παράδειγμα ροής κλίσεων από το φως στο σκοτάδι σε μια εικόνα [8]. 40
3.6 Παράδειγμα ανίχνευσης προσώπου με Ιστογράμματα Προσανατολισμένων Κλίσεων βάση μοτίβων που εξήχθησαν από ένα πλήθος άλλων προσώπων εκπαίδευσης. [8]. 40
3.7 Δύο παραδείγματα της επιλεκτικής αναζήτησης που δείχνουν την αναγκαιότητα διαφορετικών κλιμάκων. Στα αριστερά βρίσκουμε πολλά αντικείμενα μεγέθη. Στα δεξιά βρίσκουμε απαραιτήτως τα αντικείμενα σε διαφορετικές κλίμακες, καθώς το κορίτσι περιέχεται από την τηλεόραση [9]. 41
3.8 Η αρχιτεκτονική του R-CNN [10]. 42
3.9 Η αρχιτεκτονική του Fast R-CNN [11]. 43
3.10 Σύγκριση χρόνων αλγορίθμων ανίχνευσης αντικειμένων R-CNN και Fast R-CNN [12]. 43
3.11 Η αρχιτεκτονική του Faster R-CNN [13]. 45
3.12 Σύγκριση χρόνων αλγορίθμων ανίχνευσης αντικειμένων R-CNN, Fast R-CNN και Faster R-CNN [12]. 45
3.13 Παράδειγμα τμηματοποίησης αντικειμένων σε μια εικόνα με το Mask-RCNN. 46
3.14 Η αρχιτεκτονική του Mask-RCNN για τμηματοποίηση αντικειμένων [14]. 48
3.15 Εικόνα εισόδου με τα πλαίσια οριοθέτησης και τις ετικέτες των αντικειμένων [15]. 49
3.16 8×8 χάρτης χαρακτηριστικών με το πλέγμα των σημείων αγκύρωσης το οποίο τοποθετείται πάνω από την εικόνα εισόδου [15]. 49
3.17 4×4 χάρτης χαρακτηριστικών με την τοποθεσία και την βαθμολογία εμπιστοσύνης των κλάσεων αντικειμένων [15]. 49
3.18 Αρχιτεκτονική του SSD [15]. 50
3.19 Αρχιτεκτονική VGG (η είσοδος είναι 224x224x3) [16]. 50

3.20 Αρχιτεκτονική πολυεπίπεδης συνελικτικής πρόβλεψης θέσης και εμπιστοσύνης του multibox [16]. 51
3.21 Οπτικοποίηση του χάρτη χαρακτηριστικών ενός μοντέλου VGG [17]. . 53
3.22 Η αρχιτεκτονική του RetinaNet [18]. 54
3.23 Αρχιτεκτονική δικτύου πυραμιδικών χαρακτηριστικών (FPN) [19] . . . 59
3.24 (a) Το FPN εισάγει ένα μονοπάτι από πάνω προς τα κάτω για την ασφάλεια λειτουργιών πολλαπλών κλιμάκων. (b) Το PANet προσθέτει ένα επιπλέον μονοπάτι από πάνω προς τα πάνω στην κορυφή του FPN. (c) NAS-FPN χρησιμοποιεί αναζήτηση αρχιτεκτονικής με χρήση νευρωνικών δικτύων για να βρει μια τοπολογία δικτύου χαρακτηριστικών και στη συνέχεια να εφαρμοστεί επανειλημμένα το ίδιο μπλοκ. (d) Το BiFPN, με καλύτερη αντιστάθμιση και αποτελεσματικότητα [20] 61
3.25 Αρχιτεκτονική EfficientDet - Χρησιμοποιεί το EfficientNet [21] ως δίκτυο backbone, BiFPN ως δίκτυο δυνατοτήτων και κοινόχρηστο δίκτυο προβλέψεων τάξης / πλαισίου. Τόσο τα επίπεδα BiFPN όσο και τα επίπεδα καθαρού κλάσης / πλαισίου επαναλαμβάνονται πολλές φορές με βάση διαφορετικούς περιορισμούς πόρων [20] 64
3.26 Κλιμάκωση ρυθμίσεων για EfficientDet D0-D6. Ο φ είναι ο σύνθετος συντελεστής που ελέγχει όλες τις άλλες διαβαθμίσεις κλίμακας. Τα BiFPN, το box/class net, και το input size κλιμακώνονται σύμφωνα με τις εξισώσεις 3.9, 3.10, 3.11, αντίστοιχα [20] 66
3.27 Μια απλοποιημένη απεικόνιση της λογικής του ανιχνευτή αντικειμένων YOLO [22]. 67
3.28 Ο YOLOv3 διαιρεί την εικόνα σε κελιά 13 x 13, σύμφωνα με τις διαστάσεις του χάρτη χαρακτηριστικών. Το κόκκινο κελί είναι υπεύθυνο για την ανίχνευση του σκύλου στην εικόνα [23]. 69
3.29 Οι μαθηματικοί τύποι των bx, by, bw, bh [24]. 70
3.30 Διαστάσεις του πλαισίου οριοθέτησης του YOLOv3 [24]. 72
3.31 Προβλέψεις χάρτη χαρακτηριστικών σε διαφορετικά μεγέθη [22]. . . . 74
3.32 Πολλαπλά grids μπορεί να ανιχνεύουν το ίδιο αντικείμενο. Η μη μέγιστη καταστολή χρησιμοποιείται για να απαλείψει τις διπλότυπες ανιχνεύσεις [22]. 75

3.33 Διάγραμμα της αρχιτεκτονικής δικτύου του YOLOv3 [23]. 76
3.34 Το YOLOv4 Βελτιώνει το AP και το FPS του YOLOv3 κατά 10% και 12%, αντίστοιχα [25]. 77
3.35 Παράδειγμα γεωμετρικών παραμορφώσεων [26]. 78
3.36 Παράδειγμα τυχαίων διαγραφών στην ανίχνευση αντικειμένων [27]. . 79
3.37 (α) Εικόνα εισαγωγής σε συνελικτικό νευρωνικό δίκτυο. Οι πράσινες περιοχές στα (β) και (γ) περιλαμβάνουν μονάδες ενεργοποίησης που περιέχουν σημασιολογικές πληροφορίες στην εικόνα εισόδου. Η κατάργηση των ενεργοποιήσεων με τυχαίο τρόπο δεν είναι αποτελεσματική για την αφαίρεση σημασιολογικών πληροφοριών, επειδή οι κοντινές ενεργοποιήσεις περιέχουν πολύ σχετικές πληροφορίες. Αντίθετα, η απομάκρυνση συνεχών περιοχών μπορεί να αφαιρέσει ορισμένες σημασιολογικές πληροφορίες (π.χ. κεφάλι ή πόδια) και κατά συνέπεια να επιβάλει τις υπόλοιπες μονάδες για να μάθουν χαρακτηριστικά για την κατηγοριοποίηση της εικόνας εισόδου [28]. 79
3.38 Bag of Freebies στο YOLOv4 [25]. 80
3.39 Τροποποιημένο επίπεδο PAN YOLOv4 [25]. 81
3.40 Το Head του YOLOv4. 82
3.41 Η ομαλοποίηση παρτίδας υπολογίζει στατιστικά στοιχεία ενεργοποίησης από 4 διαφορετικές εικόνες σε κάθε επίπεδο [25]. 83

4.1 Αρχείο λίστας απαιτήσεων πακέτων Python (requirements.txt). 86
4.2 Περιβάλλον Jupyter Notebook. 87
4.3 Περιβάλλον TensorBoard για την αστικοποίηση της ακρίβειας του μοντέλου 90
4.4 Εκτέλεση σημειωματάριου εκπαίδευσης αλγορίθμου SSD στο Google Colab. 92

5.1 Δομή ενός CPU και ενός GPU [29]. 94
5.2 GPU 95
5.3 Tensor Processing Unit 3.0 [30]. 95
5.4 Λογισμικό LabelImg για σχόλια των αντικειμένων επί των εικόνων. . . 97

5.5 Λήψη εικόνων μαζί με τους σχολιασμούς για εκπαίδευση μέσω του εργαλείου OIDv4 ToolKit. 98
5.6 Οπτικοποίηση του συνόλου δεδομένων που θα χρησιμοποιηθεί στην εκπαίδευση των αλγορίθμων. Τα κίτρινα πλαίσια αποτελούν τα πλαίσια οριοθέτησης των αντικειμένων του συνόλου δεδομένων. 99
5.7 Διάγραμμα της διαδικασίας ανάπτυξης ενός ανιχνευτή αντικειμένου με το Tensorflow 2 Object Detection API. 101
5.8 Βήματα εκπαίδευσης ανιχνευτή Faster R-CNN με το TensorFlow 2 Object Detection API. 107
5.9 Γράφημα της μείωση της συνολικής απόκλισης κατά τη εκπαίδευση του Faster R-CNN από το TensorBoard. 108
5.10 Βήματα εκπαίδευσης Faster R-CNN με το TensorFlow 2 Object Detection API 109
5.11 Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου Faster R-CNN. Αυτή η εικόνα δεν είναι μέρος του συνόλου εκπαίδευσης ή επαλήθευσης του μοντέλου. 112
5.12 Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου Faster R-CNN. Αυτή η εικόνα δεν είναι μέρος του συνόλου εκπαίδευσης ή επαλήθευσης του μοντέλου. 112
5.13 Διάγραμμα της διαδικασίας ανάπτυξης ενός ανιχνευτή αντικειμένου με το Darknet. 113
5.14 Εκπαίδευση του μοντέλου YOLOv4 με Darknet. 115
5.15 Γράφημα απόκλισης και mAP σε YOLOv3 με Darknet 116
5.16 Γράφημα απόκλισης και mAP σε YOLOv4 με Darknet 117
5.17 Υπολογισμός mAP εκπαιδευμένου μοντέλου YOLOv3 με Darknet. Το αποτέλεσμα είναι 89.67% σε IoU=50%. 118
5.18 Υπολογισμός mAP εκπαιδευμένου μοντέλου YOLOv4 με Darknet. Το αποτέλεσμα είναι 94.40% σε IoU=50%. 118
5.19 Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv3 με Darknet. 119
5.20 Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv3 με Darknet. 119

5.21 Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv4 με Darknet. 120
5.22 Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv4 με Darknet. 120
5.23 Παράδειγμα IOU μεταξύ ενός πλαισίου οριοθέτησης αλήθειας εδάφους (σε πράσινο) και ενός εντοπισμένου πλαισίου οριοθέτησης (με κόκκινο χρώμα) [31]. 122
5.24 Καμπύλη ανάκλησης-ακρίβειας (precision-recall) [32]. 123
5.25 Καμπύλη ανάκλησης-IoU [32]. 124
5.26 Μέση ακρίβεια στο COCO. 128
5.27 FPS στο COCO. 128

6.1 Παράδειγμα ανιχνευτή τήρησης κοινωνικών αποστάσεων με χρήση YOLOv3 και OpenCV [33] 134
6.2 Παράδειγμα εντοπισμού θέσεων στάθμευσης και κατηγοριοποίηση ως ελεύθερες ή κατειλημμένες [34]. 135

Κατάλογος πινάκων

5.1 Υλικό μέρος συστήματος εργασίας. 95
5.2 Framework που χρησιμοποιήθηκε για τον κάθε αλγόριθμο. 100
5.3 Σύγκριση της αποδοτικότητας των αλγορίθμων στο COCO. 128
5.4 Σύγκριση της απόδοσης των αλγορίθμων σύμφωνα με τις μετρήσεις της πρόκλησης COCO. 129
5.5 Σύγκριση των διαφορετικών αποκλίσεων των αλγορίθμων κατά την εκπαίδευση τους. 129
5.6 Σύγκριση χρόνων ανίχνευσης αντικείμενων σε CPU και GPU. 130

Κατάλογος απεικονίσεων

5.1 Ορισμός μοντέλου που θα εκπαιδεύσουμε με το TensorFlow. 102
5.2 Ορισμός των διαδρομών των αρχείων δεδομένων και ρυθμίσεων για το TensorFlow. 103
5.3 Αρχείο χάρτη ετικετών label_map.pbtxt 104
5.4 Λήψη του προ-εκπαιδευμένου μοντέλου από το TensorFlow Model Zoo.105
5.5 Διαχωρισμός σε σετ εκπαίδευσης και σετ δοκιμών. 105
5.6 Μετατροπή ετικετών xml σε csv. 106
5.7 Μετατροπή σε μορφή TFRecord (.record). 106
5.8 Έναρξη της εκπαίδευσης. 107
5.9 Οπτικοποίηση δεδομένων με το TensorBoard. 108
5.10 Αξιολόγηση του εκπαιδευμένου μοντέλου. 108
5.11 Εξαγωγή του εκπαιδευμένου μοντέλου. 109
5.12 Φόρτωση όλων των εικόνων δοκιμής για ανίχνευση με το εκπαιδευμένο μοντέλο. 110
5.13 Φόρτωση του εκπαιδευμένου μοντέλου. 110
5.14 Δημιουργία προβλέψεων στο σύνολο δεδομένων των εικόνων δοκιμής. 111
5.15 Δημιουργία αρχείου ρυθμίσεων δεδομένων gun_detection.txt και αρχείου train.txt με τις διαδρομές όλων των εικόνων για εκπαίδευση . . 114
5.16 Λήψη του προ-εκπαιδευμένου δικτύου YOLOv3 για εκπαίδευση. . . . 114
5.17 Λήψη του προ-εκπαιδευμένου δικτύου YOLOv4 για εκπαίδευση. . . . 114
5.18 Εκπαίδευση YOLOv3 με Darknet. 115
5.19 Εκπαίδευση YOLOv4 με Darknet. 116
5.20 Υπολογισμός mAP σε YOLOv3 με Darknet. 117
5.21 Υπολογισμός mAP σε YOLOv4 με Darknet 118
5.22 Ανίχνευση αντικειμένων με YOLOv3. 119
5.23 Ανίχνευση αντικειμένων με YOLOv4. 120

Κεφάλαιο 1

Εισαγωγή

Σήμερα, το πρόβλημα της αναγνώρισης και του εντοπισμού αντικειμένων σε εικόνες επιλύεται χρησιμοποιώντας Νευρωνικά Δίκτυα Συνέλιξης. Το γεγονός ότι τα Νευρωνικά Δίκτυα Συνέλιξης, σε συνδυασμό με κάρτες γραφικών, δίνουν τη δυνατότητα επίλυσης προβλημάτων αναγνώρισης και εντοπισμού αντικειμένων σε μικρό χρόνο, τα καθιστά ικανά να χρησιμοποιηθούν σε εφαρμογές πραγματικού χρόνου.

1.1 Περιγραφή του προβλήματος

Οι άνθρωποι ρίχνουν μια ματιά σε μια εικόνα και ξέρουν αμέσως τι αντικείμενα υπάρχουν στην εικόνα, πού βρίσκονται και πώς αλληλεπιδρούν. Το ανθρώπινο οπτικό σύστημα είναι γρήγορο και ακριβές, επιτρέποντάς μας να εκτελέσουμε πολύπλοκες εργασίες, όπως οδήγηση με λίγη συνετή σκέψη. Αντιθέτως, το πρόβλημα του εντοπισμού αντικειμένων από έναν υπολογιστή δεν είναι τόσο απλό. Γρήγοροι και ακριβείς αλγόριθμοι για την ανίχνευση αντικειμένων θα επέτρεπαν στους υπολογιστές να οδηγούν αυτοκίνητα χωρίς ειδικούς αισθητήρες, να επιτρέπουν σε βοηθητικές συσκευές να μεταφέρουν πληροφορίες πραγματικού χρόνου σε ανθρώπινους χρήστες και να ξεκλειδώνουν τις δυνατότητες γενικής χρήσης για ανταποκρινόμενα ρομποτικά συστήματα.

1.2 Στόχος και συνεισφορά της διπλωματικής

Στόχος της παρούσας διπλωματικής εργασίας είναι να κάνει μια γενικευμένη ανάλυση των επικρατέστερων αλγορίθμων μηχανικής μάθησης για την ανίχνευση

αντικειμένων που είναι διαθέσιμοι σήμερα, να μελετήσει σε βάθος τη χρήση νευρωνικών δικτύων συνέλιξης μέσω αλγορίθμων μηχανικής μάθησης σε εφαρμογές ανίχνευσης και εντοπισμού αντικειμένων (object recognition and localization) σε εικόνες, βίντεο αλλά και σε εφαρμογές που απαιτούν ανίχνευση αντικειμένων σε πραγματικό χρόνο.

Επίσης, στοχεύει στην εκπαίδευση μοντέλων ανίχνευσης αντικειμένων και στην εφαρμογή τους για τον εντοπισμό όπλων σε εικόνες, σε βίντεο αλλά και σε πραγματικό χρόνο. Τέλος, παρουσιάζονται μετρήσεις για την απόδοση και την αποτελεσματικότητα αυτών των αλγορίθμων στο πρόβλημα του εντοπισμού όπλων.

1.3 Διάρθρωση της εργασίας

Η διάρθρωση της παρούσας διπλωματικής εργασίας είναι η εξής: Στο κεφάλαιο 2 θα αναπτυχθούν οι έννοιες της τεχνητής νοημοσύνης, της μηχανικής μάθησης και της υπολογιστικής όρασης και στη συνέχεια θα αναπτυχθούν οι έννοιες της αρχιτεκτονικής δομής και των επιπέδων (layers) των CNNs καθώς και των τεχνικών εκπαίδευσης (training), χωρικής υποδειγματοληψίας (pooling), κανονικοποίησης (normalization) και ενεργοποίησης (activation) που τα διέπουν.

Στη συνέχεια, στο κεφάλαιο 3, γίνεται μία ιστορική αναδρομή των αλγορίθμων που έθεσαν τα θεμέλια για την ανάπτυξη της ανίχνευσης αντικειμένων και θα μελετηθούν οι πιο σύγχρονοι αλγόριθμοι αναγνώρισης και εντοπισμού αντικειμένων Faster R-CNN, SSD, RetinaNet, EfficentDet, YOLOv3 και YOLOv4. Στο κεφάλαιο 4 γίνεται αναφορά στα εργαλεία και στις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη και την εκπαίδευση των ανιχνευτών αντικειμένων. Έπειτα, στο κεφάλαιο 5, γίνεται ενδελεχής μελέτη στην εκπαίδευση μοντέλων ανίχνευσης αντικειμένων και παρουσιάζονται συγκριτικά στοιχεία απόδοσης και ακρίβειας των αλγορίθμων που μελετάει η παρούσα διπλωματική εργασία. Τέλος, στο κεφάλαιο 6 παρουσιάζονται τα συμπεράσματα της διπλωματικής αυτής εργασίας και παρουσιάζονται μερικές δυνατότητες που προσφέρουν σε ένα ευρύ φάσμα εφαρμογών και επιστημών.

Κεφάλαιο 2

Έννοιες και Ορισμοί

Σε αυτό το κεφάλαιο παρουσιάζεται όλο το θεωρητικό πλαίσιο των εννοιών της διπλωματικής εργασίας.

2.1 Τεχνητή Νοημοσύνη

Ο Alan Turing όρισε την Τεχνητή Νοημοσύνη (Artificial Intelligence - AI) ως εξής: ”Αν υπάρχει μηχανή πίσω από μια κουρτίνα και ένας άνθρωπος αλληλεπιδρά μαζί της (με οποιονδήποτε τρόπο, π.χ. ήχος ή μέσω πληκτρολόγησης κλπ.) και αν αυτός ο άνθρωπος αισθάνεται ότι είναι σαν να αλληλεπιδρά με έναν άλλο άνθρωπο, τότε το μηχάνημα είναι τεχνητά έξυπνο” [35]. Αυτός είναι ένας αρκετά μοναδικός τρόπος για να ορίσουμε το AI. Δεν στοχεύει άμεσα στην έννοια της νοημοσύνης, αλλά επικεντρώνεται στην αλληλεπίδραση μηχανών-ανθρώπων που μοιάζει με αυτή μεταξύ των ανθρώπων. Στην πραγματικότητα, ο στόχος αυτός είναι ακόμη ευρύτερος από την απλή νοημοσύνη. Από αυτή την προοπτική, το AI δε στοχεύει στην κατασκευή μιας εξαιρετικά έξυπνης μηχανής που μπορεί να λύσει οποιοδήποτε πρόβλημα σε χρόνο μηδέν, αλλά στοχεύει την ανάπτυξη μιας μηχανής ικανής για ανθρώπινη συμπεριφορά.

Σύμφωνα με τη σύγχρονη αντίληψη, όταν μιλάμε για AI, εννοούμε μηχανές που είναι ικανές να εκτελέσουν μία ή περισσότερες από αυτές τις εργασίες: κατανόηση της ανθρώπινης γλώσσας, εκτέλεση μηχανικών εργασιών που περιλαμβάνουν πολύπλοκους υπολογισμούς, επίλυση περίπλοκων προβλημάτων που βασίζονται σε υπολογιστή και που ενδέχεται να περιλαμβάνουν μεγάλα δεδομένα σε πολύ σύντομο χρονικό διάστημα και να δίνουν απαντήσεις με ανθρώπινο τρόπο.

Με άλλα λόγια, η τεχνητή νοημοσύνη καθιστά τις μηχανές ικανές να μαθαίνουν

από την εμπειρία, να προσαρμόζονται σε νέα εισαγόμενα δεδομένα και να εκτελούν ανθρωπομορφικά έργα. Τα περισσότερα επιτεύγματα με AI (από τους υπολογιστές που παίζουν σκάκι έως τα αυτο-οδηγούμενα αυτοκίνητα) βασίζονται σε μεγάλο βαθμό στη βαθιά μάθηση (deep learning). Με τη χρήση των τεχνολογιών αυτών, οι υπολογιστές μπορούν να εκπαιδευτούν, ώστε να επιτελούν συγκεκριμένα καθήκοντα με επεξεργασία μεγάλων ποσοτήτων δεδομένων και αναγνώριση μορφών στα δεδομένα.

2.2 Μηχανική Μάθηση

Ο όρος "Μηχανική Μάθηση" (machine learning) ή ML εν συντομία, προτάθηκε το 1959 από τον Arthur Samuel στο πλαίσιο της επίλυσης του παιχνιδιού Ντάμα με μηχανή [36]. Ο όρος αναφέρεται σε ένα πρόγραμμα υπολογιστή που μπορεί να μάθει να παράγει μια συμπεριφορά που δεν είχε ρητά προγραμματιστεί από τον συγγραφέα του προγράμματος. Πόσο μάλλον είναι ικανό να δείξει κάποια συμπεριφορά την οποία ο συγγραφέας μπορεί να αγνοεί εντελώς. Αυτή η συμπεριφορά «μαθαίνεται» βάση τριών παραγόντων:

1. δεδομένα που λαμβάνονται ως είσοδος και επεξεργάζονται από το πρόγραμμα
2. μια μέτρηση που ποσοτικοποιεί το σφάλμα ή κάποια μορφή απόστασης μεταξύ της τρέχουσας συμπεριφοράς και της ιδανικής συμπεριφοράς
3. και ένας μηχανισμός ανατροφοδότησης που χρησιμοποιεί το ποσοτικοποιημένο σφάλμα για την καθοδήγηση του προγράμματος για την παραγωγή καλύτερης συμπεριφοράς στα επόμενα γεγονότα.

Όπως φαίνεται, ο δεύτερος και ο τρίτος παράγοντας καθιστούν την έννοια της αφηρημένη. Αυτές οι μέθοδοι, στη θεωρία της μηχανικής μάθησης, είναι απαραίτητες στη δημιουργία τεχνητά έξυπνων συστημάτων.

Η μηχανική μάθηση επικαλύπτεται σημαντικά με τη στατιστική, αφού και τα δύο πεδία μελετούν την ανάλυση δεδομένων. Οι ιδέες της μηχανικής μάθησης, από τις μεθοδολογικές αρχές μέχρι τα θεωρητικά εργαλεία, προϋπάρχουν στη στατιστική.

Ο Leo Breiman [37] διέκρινε δύο υποδείγματα στατιστικής μοντελοποίησης: το μοντέλο δεδομένων και το αλγοριθμικό μοντέλο. Το αλγοριθμικό μοντέλο ταυτίζεται σχεδόν με αλγορίθμους μηχανικής μάθησης. Οι αλγόριθμοι μηχανικής μάθησης

κατηγοριοποιούνται ανάλογα με το επιθυμητό αποτέλεσμα του αλγορίθμου. Οι συνηθέστερες κατηγορίες είναι οι εξής:

Επιβλεπόμενη μάθηση (supervised learning): Το υπολογιστικό πρόγραμμα δέχεται τις παραδειγματικές εισόδους καθώς και τα επιθυμητά αποτελέσματα από έναν «δάσκαλο» και ο στόχος είναι να μάθει έναν γενικό κανόνα, προκειμένου να αντιστοιχίσει τις εισόδους με τα αποτελέσματα. Οι πιο ευρέως χρησιμοποιούμενοι αλγόριθμοι επιβλεπόμενης μάθησης είναι:

- Υποστήριξη διανυσματικών μηχανών (Support Vector Machines - SVMs)
- Γραμμική παλινδρόμηση (Linear regression)
- Λογιστική παλινδρόμηση (Logistic regression)
- Δέντρα αποφάσεων (Decision tree)
- Κ κοντινότεροι γείτονες (K Nearest Neighbors - KNN)
- Αφελής κατηγοριοποιητής Bayes (Naive Bayes classifier)

Μη επιβλεπόμενη μάθηση (unsupervised learning): Χωρίς να παρέχεται κάποια εμπειρία στον αλγόριθμο μάθησης, πρέπει να βρει τη δομή των δεδομένων εισόδου. Διαφορετικά, ονομάζεται και μάθηση από παρατήρηση, καθώς το σύστημα πρέπει από μόνο του να ανακαλύψει συσχετίσεις στο σύνολο δεδομένων του αλγορίθμου. Η μη επιβλεπόμενη μάθηση μπορεί να είναι αυτοσκοπός (ανακαλύπτοντας κρυμμένα μοτίβα σε δεδομένα) ή μέσο για ένα τέλος (χαρακτηριστικό της μάθησης).

Ενισχυτική μάθηση (reinforcement learning): Ένα πρόγραμμα υπολογιστή αλληλεπιδρά με ένα δυναμικό περιβάλλον, στο οποίο πρέπει να επιτευχθεί ένας συγκεκριμένος στόχος (όπως η οδήγηση ενός οχήματος), χωρίς κάποιος δάσκαλος να του λέει ρητά αν έχει φτάσει κοντά στον στόχο του. Ένα άλλο παράδειγμα είναι να μάθει να παίζει ένα παιχνίδι εναντίον κάποιου αντιπάλου.

2.3 Βαθιά Μάθηση

Η βαθιά μάθηση (Deep learning) είναι ένας τομέας της μηχανικής μάθησης που χρησιμοποιεί έναν αλγόριθμο οπισθοδιάδοσης (backpropagation) έτσι ώστε να αλλάζει τις εσωτερικές παραμέτρους που ονομάζονται βάρη. Οι ενημερώσεις των βαρών

βασίζονται σε μια συνάρτηση σφάλματος η οποία υπολογίζει τις αποκλίσεις. Χρησιμοποιώντας αυτές τις αποκλίσεις, τα βάρη κάθε νευρώνα ενημερώνονται μετά από κάθε επανάληψη, έτσι ώστε η έξοδος του συστήματος βαθιάς μάθησης να είναι πιο κοντά στην ιδανική έξοδο ή στη διάλεκτο της μηχανικής εκμάθησης στο αποτέλεσμα-στόχο.

Οι μέθοδοι βαθιάς μάθησης είναι πλέον σύγχρονες όχι μόνο στον τομέα της μηχανικής όρασης, αλλά και στην Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing - NLP), και σε πολλούς άλλους τομείς. Οι μέθοδοι βαθιάς μάθησης έχουν μελετηθεί από τη δεκαετία του '90, αλλά τα τελευταία δέκα χρόνια, τα νευρωνικά δίκτυα έχουν αναγεννηθεί, κυρίως λόγω της διαθεσιμότητας ισχυρότερων υπολογιστών και μεγαλύτερων συνόλων δεδομένων.

Στις αρχές της δεκαετίας του 2000, βρέθηκε ότι τα νευρωνικά δίκτυα θα μπορούσαν να εκπαιδευτούν χρησιμοποιώντας με ευκολία μονάδες επεξεργασίας γραφικών. Οι GPUs είναι πιο αποτελεσματικές για την εργασία αυτή από τις παραδοσιακές CPUs και παρέχουν μια σχετικά φθηνή εναλλακτική λύση σε σχέση με το εξειδικευμένο υλικό. Σήμερα, οι ερευνητές χρησιμοποιούν συνήθως κάρτες υψηλού επιπέδου για τους καταναλωτές, όπως η GPU NVIDIA Tesla K40. Άλλες πιο θεωρητικές ανακαλύψεις περιλαμβάνουν την αντικατάσταση του σφάλματος μέσου τετραγώνου με συναρτήσεις βασισμένες σε εγκάρσια εντροπία και αντικατάσταση της σιγμοειδούς συνάρτησης ενεργοποίησης με διορθωμένες γραμμικές μονάδες [38].

Με τη βαθιά μάθηση, υπάρχει λιγότερη ανάγκη για λύσεις με χειροκίνητη μηχανική εκμάθηση όπως γίνονταν προηγουμένως [38]. Για παράδειγμα, σε ένα κλασσικό σύστημα ανίχνευσης προτύπων περιλαμβάνεται μια φάση ανίχνευσης χαρακτηριστικών με το χέρι πριν από την φάση μηχανικής μάθησης. Το ισοδύναμο της βαθιάς μάθησης αποτελείται από ένα νευρωνικό δίκτυο. Τα χαμηλότερα επίπεδα του νευρωνικού δικτύου μαθαίνουν να αναγνωρίζουν τα βασικά χαρακτηριστικά, τα οποία στη συνέχεια τροφοδοτούνται στα υψηλότερα επίπεδα του δικτύου.

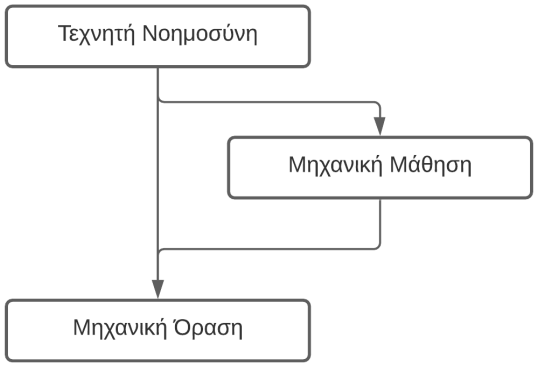
2.4 Μηχανική Όραση

Η μηχανική όραση είναι ένα διεπιστημονικό πεδίο της τεχνητής νοημοσύνης το οποίο επιχειρεί να αναπαράγει αλγοριθμικά την αίσθηση της όρασης. Από μηχανικής απόψεως προσπαθεί να κατανοήσει και να αυτοματοποιήσει λειτουργίες τις

οποίες κάνει το ανθρωπινό οπτικό σύστημα. Είναι ένα διεπιστημονικό πεδίο που θα μπορούσε γενικά να κατηγοριοποιηθεί ως υποπεδίο της τεχνητής νοημοσύνης και της μηχανικής μάθησης, το οποίο περιλαμβάνει τη χρήση εξειδικευμένων μεθόδων και τη χρήση γενικών αλγορίθμων μάθησης. Οι εφαρμογές της μηχανικής όρασης περιλαμβάνουν ταξινόμηση εικόνας, οπτική ανίχνευση, 3D ανακατασκευή σκηνής από 2D εικόνες, ανάκτηση εικόνας, επαυξημένη πραγματικότητα, μηχανική όραση και αυτοματοποίηση.

Σήμερα, η μηχανική όραση είναι απαραίτητο συστατικό πολλών αλγορίθμων. Τέτοιοι αλγόριθμοι μπορούν να περιγραφούν ως συνδυασμός επεξεργασίας εικόνας και μηχανικής μάθησης. Οι αποτελεσματικές λύσεις απαιτούν αλγορίθμους που μπορούν να αντεπεξέλθουν στην τεράστια ποσότητα οπτικών πληροφοριών που περιέχονται στις εικόνες και για κρίσιμες εφαρμογές, να πραγματοποιήσουν υπολογισμούς σε πραγματικό χρόνο. Σε αφηρημένο επίπεδο, ο στόχος της μηχανικής όρασης είναι η χρήση των δεδομένων εικόνας για να συμπεράνουμε κάτι για τον κόσμο.

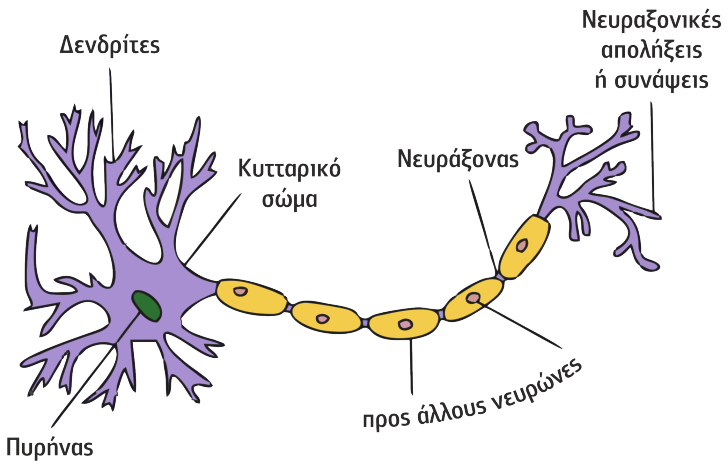
Στο σχήμα 2.1 παρουσιάζεται η σχέση της υπολογιστικής όρασης με την τεχνητή νοημοσύνη και τη μηχανική μάθηση. Από αυτό συμπεραίνουμε ότι η υπολογιστική όραση αποτελεί σημαντικό υποπεδίο της τεχνητής νοημοσύνης και κατ' επέκταση της μηχανικής μάθησης.



Σχήμα 2.1: Επισκόπηση της σχέσης της υπολογιστικής όρασης με την τεχνητή νοημοσύνη και την μηχανική μάθηση.

2.5 Νευρωνικά Δίκτυα

Ο όρος Νευρωνικά Δίκτυα (Neural Networks) περιγράφει έναν αριθμό από διαφορετικά μαθηματικά μοντέλα, εμπνευσμένα από αντίστοιχα βιολογικά μοντέλα, δηλαδή μοντέλα που προσπαθούν να μιμηθούν τη συμπεριφορά των νευρώνων του ανθρώπινου εγκεφάλου. Ήδη από τον 19ο αιώνα, οι επιστήμονες παραδέχονται ότι ο εγκέφαλος αποτελείται από διακριτά στοιχεία, τους νευρώνες (neurons), που επικοινωνούν ο ένας με τους άλλους. Οι νευρώνες αποτελούν το βασικό δομικό κομμάτι του ανθρώπινου εγκεφάλου. Υπολογίζεται ότι ο εγκέφαλος περιέχει 10 δισεκατομμύρια περίπου νευρώνες τοποθετημένους σε ομάδες, καθεμία από τις οποίες συνιστά ένα φυσικό νευρωνικό δίκτυο. Έτσι, ο ανθρώπινος εγκέφαλος περιέχει εκατοντάδες φυσικά νευρωνικά δίκτυα, καθένα από τα οποία περιέχει χιλιάδες διασυνδεδεμένους νευρώνες με μέσο αριθμό διασυνδέσεων ανά νευρώνα 1000 με 10.000 [1].



Σχήμα 2.2: Διάγραμμα ενός ανθρώπινου νευρώνα [1].

2.6 Τεχνητά Νευρωνικά Δίκτυα

Τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks) αποτελούν έναν από τους πιο γνωστούς τύπους μοντέλων εποπτευόμενης μάθησης. Ο όρος «τεχνητά νευρωνικά δίκτυα» προέκυψε επειδή αναπτύχθηκαν για να προσομοιάζουν τον τρόπο με τον οποίο δουλεύει το ανθρώπινο νευρικό σύστημα. Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από ένα σύνολο νευρώνων που συνδέονται μεταξύ τους αλλά και

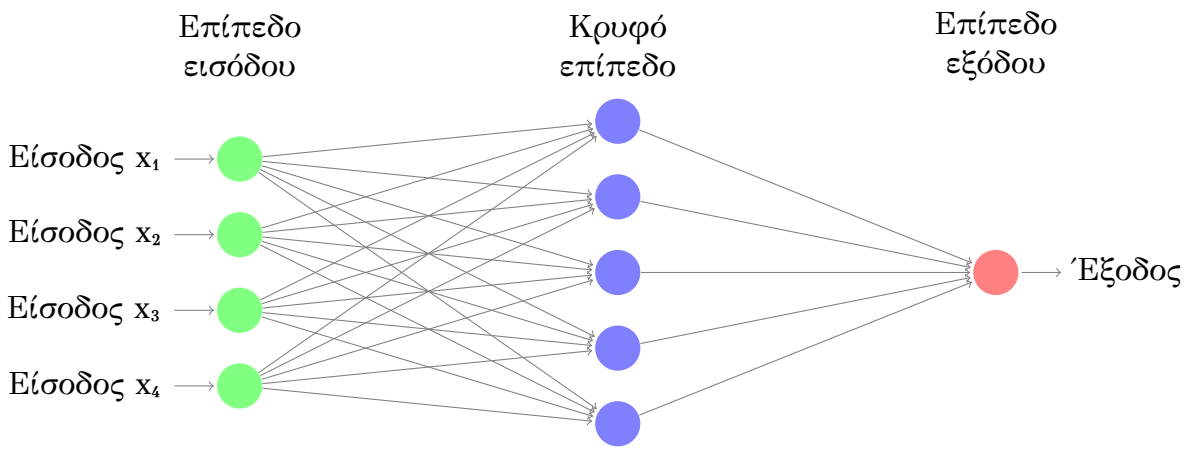
ομαδοποιούνται σε επίπεδα για να αναπαράγουν τη νευρική λειτουργία του ανθρώπινου εγκεφάλου. Τα τεχνητά νευρωνικά δίκτυα επεξεργάζονται πληροφορίες ανταποκρινόμενα δυναμικά σε εξωτερικά ερεθίσματα (εισόδους). Κάθε τεχνητός νευρώνας αποτελείται από πολλές εισόδους x_i και μία μόνο έξοδο y. Κάθε είσοδος x_i «ζυγίζεται» με ένα βάρος w_i και τα αποτελέσματα αθροίζονται μέσω της συνάρτησης αθροίσματος F, της οποίας η μαθηματική αναπαράσταση παρουσιάζεται στον τύπο 2.1.

$$F = \sum_{i}^{n} x_i w_i \tag{2.1}$$

Ο τεχνητός νευρώνας δίνει έξοδο μέσω της συνάρτησης μετάβασης (transfer function), μόνο όταν το ζυγισμένο άθροισμα των εισόδων είναι μεγαλύτερο μιας ορισμένης τιμής κατωφλίου (threshold) θ, δηλαδή όταν

$$\sum_{i}^{n} x_i w_i \; - \; \theta \; > \; 0 \tag{2.2}$$

Στο σχήμα 2.3 παρουσιάζεται η βασική δομή ενός απλού τεχνητού νευρωνικού δικτύου.



Σχήμα 2.3: Σχηματική αναπαράσταση της δομής ενός απλού νευρωνικού δικτύου.

Μια ξεχωριστή περίπτωση νευρωνικών δικτύων, τα οποία ονομάζονται συνελικτικά νευρωνικά δίκτυα (CNN), είναι η βασική εστίαση αυτής της διπλωματικής διατριβής. Ο τύπος αυτών των νευρωνικών δικτύων παρουσιάζεται στην επόμενη ενότητα.

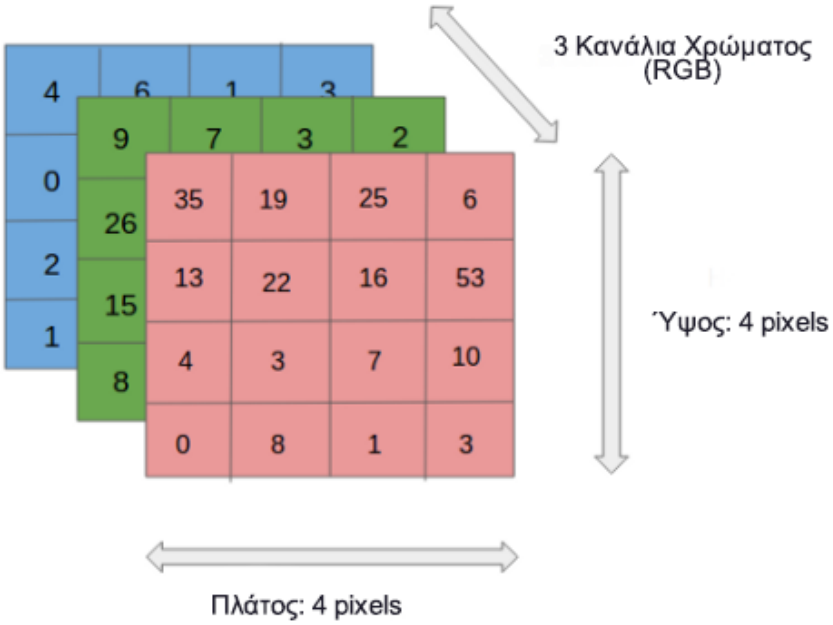
2.7 Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (CNN ή ConvNets) αποτελούν έναν αλγόριθμο βαθιάς μηχανικής μάθησης που μπορεί να λάβει μια εικόνα εισόδου, να εκχωρήσει σημασία (βάρη και πολώσεις) σε διάφορες πτυχές / αντικείμενα της εικόνας και να μπορεί να διαφοροποιεί το ένα από το άλλο. Η ονομασία των συνελικτικών νευρωνικών δικτύων προέρχεται από την ομώνυμη μαθηματική πράξη της συνέλιξης. Η προ-επεξεργασία που απαιτείται σε ένα CNN είναι πολύ χαμηλότερη σε σύγκριση με άλλους αλγορίθμους κατηγοριοποίησης. Ενώ στις πρωτόγονες μεθόδους τα φίλτρα είναι χειροποίητα, με αρκετή εκπαίδευση, τα συνελικτικά νευρωνικά δίκτυα έχουν τη δυνατότητα να μάθουν αυτά τα φίλτρα / χαρακτηριστικά.

Η αρχιτεκτονική ενός CNN είναι ανάλογη με τη συνδεσιμότητα των νευρώνων στον ανθρώπινο εγκέφαλο και εμπνεύστηκε από την οργάνωση του οπτικού φλοιού του εγκεφάλου. Οι μεμονωμένοι νευρώνες ανταποκρίνονται σε ερεθίσματα μόνο σε μια περιορισμένη περιοχή του οπτικού πεδίου που είναι γνωστό ως Δεκτικό Επίπεδο (Receptive Field). Μια συλλογή τέτοιων πεδίων αλληλοεπικαλύπτεται για να καλύψει ολόκληρη την οπτική περιοχή.

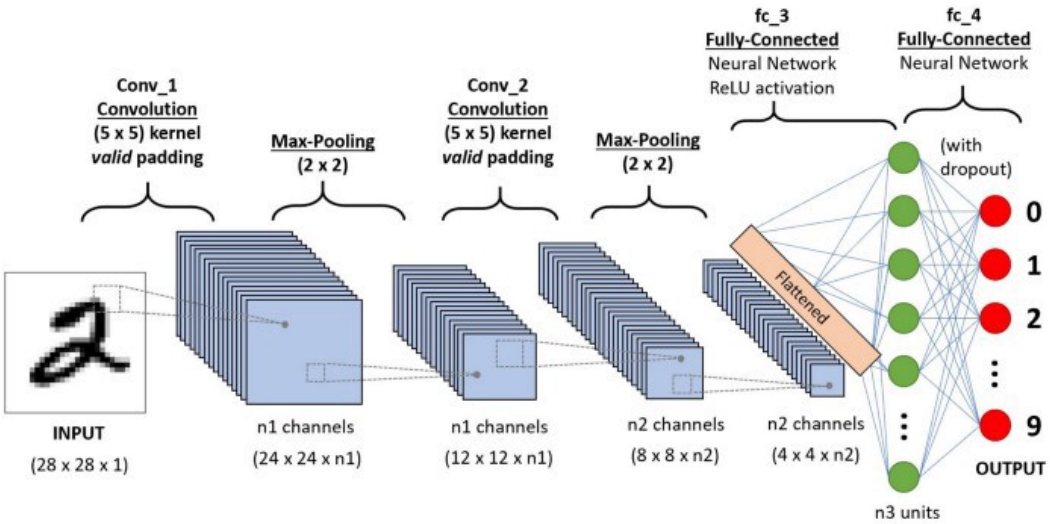
Στα νευρωνικά δίκτυα, τα συνελικτικά νευρωνικά δίκτυα είναι μια από τις κύριες κατηγορίες για την αναγνώριση εικόνων, κατηγοριοποιήσεις εικόνων, αναγνώριση προσώπων και φυσικά ανίχνευση αντικειμένων.

Για έναν υπολογιστή, μια εικόνα είναι απλώς μια σειρά τιμών. Συνήθως είναι ένας πίνακας τρισδιάστατων (RGB) τιμών pixel. Για παράδειγμα, μια αναπαράσταση εικόνας 4 * 4 RGB θα μοιάζει με αυτήν στο σχήμα 2.4, όπου κάθε εικονοστοιχείο έχει μια συγκεκριμένη τιμή κόκκινου, πράσινου και μπλε που αντιπροσωπεύει το χρώμα ενός δεδομένου εικονοστοιχείου.



Σχήμα 2.4: Οι υπολογιστές βλέπουν μια εικόνα ως πίνακα εικονοστοιχείων ο οποίος εξαρτάται από την ανάλυση της εικόνας. Με βάση την ανάλυση της εικόνας, θα δει Ύψος x Πλάτος x Διάσταση.

Το CNN επεξεργάζεται εικόνες χρησιμοποιώντας πίνακες βαρών που ονομάζονται φίλτρα (χαρακτηριστικά) που ανιχνεύουν συγκεκριμένα χαρακτηριστικά, όπως κάθετα άκρα, οριζόντια άκρα, κλπ. Ο απώτερος στόχος του CNN είναι να εντοπίσει τι συμβαίνει στη σκηνή. Στο σχήμα 2.5 παρουσιάζεται ένα παράδειγμα της δομής ενός CNN για την αναγνώριση χειρογράφων ψηφίων.



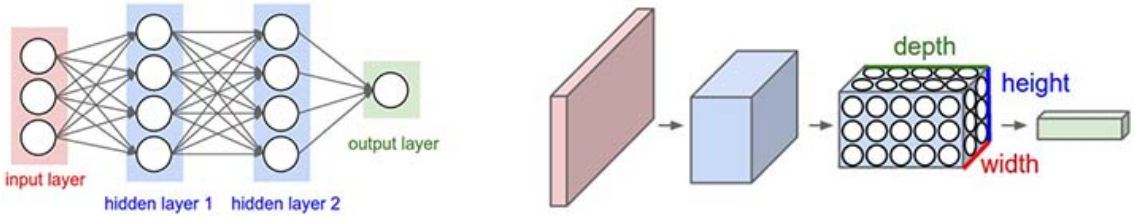
Σχήμα 2.5: Οπτικοποίηση ενός CNN για την αναγνώριση χειρογράφων ψηφίων [2].

Ένα συνελικτικό νευρωνικό δίκτυο αποτελείται από διαφορετικούς τύπους επι-

πέδων, τα οποία περιγράφονται στις παρακάτω υποενότητες.

2.7.1 Συνελικτικό επίπεδο / Convolutional layer

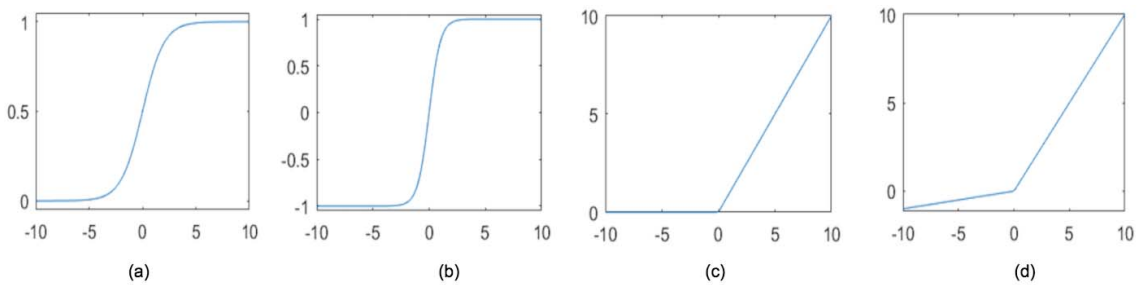
Το συνελικτικό επίπεδο είναι το βασικό δομικό στοιχείο ενός CNN. Ένας νευρώνας σε ένα συνελικτικό επίπεδο συνδέεται μόνο με ένα υποσύνολο συνεχών εξόδων του προηγούμενου επιπέδου αλλά σε όλο το βάθος τους. Κάθε νευρώνας είναι υπεύθυνος για την ενεργοποίηση τους κοιτάζοντας ένα συγκεκριμένο χαρακτηριστικό στην είσοδο του. Όλοι οι νευρώνες στη δισδιάστατη μεριά του βάθους, μοιράζονται τις ίδιες παραμέτρους (βάρη και πολώσεις), που σημαίνει ότι αναζητούν το ίδιο χαρακτηριστικό σε διαφορετικές τοποθεσίες μιας εικόνας. Αυτό μπορεί να θεωρηθεί ως συνελικτική λειτουργία όπου το ίδιο φίλτρο εφαρμόζεται σε όλες τις τοποθεσίες μιας εικόνας για να δώσει έναν χάρτη χαρακτηριστικών. Το μέγεθος του φίλτρου είναι το δεκτικό πεδίο του νευρώνα και το βάθος του φίλτρου είναι συχνά ίδιο με το βάθος του όγκου εισόδου.



Σχήμα 2.6: Η αριστερή εικόνα δείχνει ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο. Το κάθε επίπεδο συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Η δεξιά εικόνα δείχνει ένα συνελικτικό νευρωνικό δίκτυο. Κάθε επίπεδο έχει νευρώνες διατεταγμένους σε τρεις διαστάσεις με πλάτος, ύψος και βάθος [3].

2.7.2 Συναρτήσεις ενεργοποίησης

Ένα συνελικτικό επίπεδο σε ένα νευρωνικό δίκτυο ακολουθείται από μια συνάρτηση ενεργοποίησης που προσθέτει την απαιτούμενη μη γραμμικότητα στο δίκτυο. Πρακτικά αποτελούν έναν κόμβο που τοποθετείται στο τέλος ή ανάμεσα σε νευρωνικά δίκτυα και βοηθούν στο να ληφθεί η απόφαση για το αν θα ενεργοποιηθεί ένας νευρώνας ή όχι. Μια λειτουργία ενεργοποίησης παίρνει μια είσοδο και εκτελεί μια συγκεκριμένη μαθηματική λειτουργία σε αυτό. Ορισμένες από τις λειτουργίες ενεργοποίησης είναι οι παρακάτω.



Σχήμα 2.7: (a) Sigmoid (b) Υπερβολική Εφαπτομένη (c) ReLu (d) Leaky ReLu

1. **Sigmoid** Μια σιγμοειδής συνάρτηση παίρνει μια είσοδο και τη συμπιέζει στην περιοχή 0 έως 1. Η μαθηματική αναπαράσταση αυτής της συνάρτησης παρουσιάζεται στον τύπο 2.3.

$$sigmoid(x) \;=\; \frac{1}{1+e^{-x}} \tag{2.3}$$

2. **Υπερβολική Εφαπτομένη** Αυτή η μη γραμμικότητα συμπιέζει μια τιμή εισόδου στην περιοχή -1 έως 1. Η μαθηματική έκφραση της tanh παρουσιάζεται στον τύπο 2.4.

$$tanh(x) \;=\; \frac{1 \;-\; e^{-2x}}{1 \;+\; e^{-2x}} \tag{2.4}$$

Η έξοδος της tanh είναι μηδενική, ωστόσο είναι κορεσμένη σε πολύ υψηλές και πολύ χαμηλές τιμές όπου η κλίση είναι σχεδόν μηδέν.

3. **ReLU** Η Ανορθωμένη Γραμμική Συνάρτηση Ράμπας (Rectified linear unit) δέχεται μια είσοδο και επιστρέφει 0 εάν είναι αρνητική, αλλά για οποιαδήποτε θετική τιμή x, επιστρέφει αυτήν την ίδια τιμή πίσω. Έτσι δίνει μια έξοδο που κυμαίνεται από 0 έως άπειρο. Η μαθηματική αναπαράσταση της ReLU παρουσιάζεται στον τύπο 2.5.

$$relu\;(x) \;=\; max\;(0,\; x) \tag{2.5}$$

Η ReLU μπορεί να υπολογιστεί γρήγορα σε σύγκριση με την tanh και τη σιγμοειδή συνάρτηση που περιλαμβάνουν εκθετικά. Επίσης, η ReLU δεν έχει πρόβλημα διαβάθμισης κορεσμού. Αυτό κάνει τη ReLU την πιο συχνά χρησιμοποιούμενη λειτουργία ενεργοποίησης. Ωστόσο, οι μονάδες ReLU μπορεί να ”πεθάνουν” κατά τη διάρκεια εκπαίδευσης (”Dying ReLU”) όταν το ποσοστό

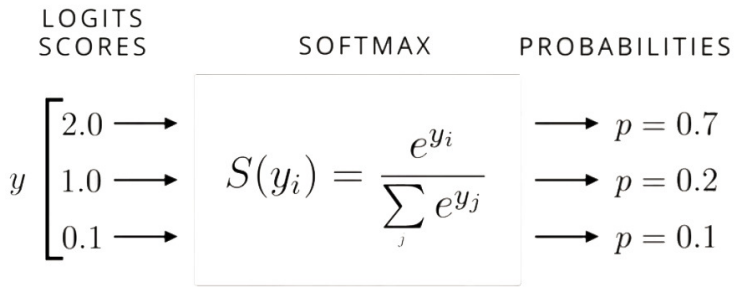
εκμάθησης είναι υψηλό.

4. **Leaky ReLU** Το Leaky ReLU αντιμετωπίζει το πρόβλημα του "Dying ReLU" καθώς δε χαρτογραφεί αρνητικά τιμές στο μηδέν, αντίθετα δίνει μια πολύ μικρή αρνητική κλίση σε αυτήν την περιοχή, όπως φαίνεται στο Σχήμα 2.7 (d). Λόγω αυτού, η κλίση δε θα γίνει ποτέ μηδέν. Η μαθηματική αναπαράσταση της Leaky ReLU παρουσιάζεται στον τύπο 2.6.

$$lrelu\ (x)\ =\ max(0,01x,\ x) \tag{2.6}$$

5. **Συνάρτηση Softmax** Η συνάρτηση Softmax είναι συνάρτηση ενεργοποίησης που μετατρέπει τους αριθμούς σε πιθανότητες που συνολικά το άθροισμά τους είναι 1. Η συνάρτηση Softmax εξάγει ένα διάνυσμα που αντιπροσωπεύει τις πιθανότητες από ένα σύνολο πιθανών αποτελεσμάτων. Η μαθηματική αναπαράσταση της Softmax παρουσιάζεται στον τύπο 2.7.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.7}$$



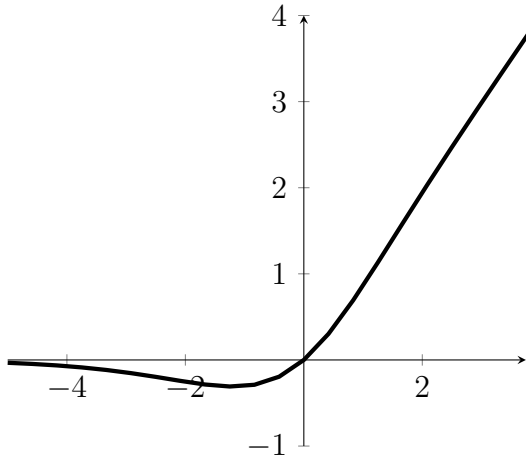
Σχήμα 2.8: Συνάρτηση Softmax [4].

Συνάρτηση Mish Η συνάρτηση ενεργοποίησης Mish είναι μια ομαλή, μη μονοτονική συνάρτηση ενεργοποίησης [39], η οποία μπορεί να οριστεί σύμφωνα με τον τύπο 2.8.

$$f(x) = x \cdot tanh(softplus(x)) \tag{2.8}$$

όπου $softplus(x) = ln(1 + e^x)$, είναι η συνάρτηση ενεργοποίησης softmax.

Ο λόγοι για τους οποίους η συνάρτηση Mish χρησιμοποιείται είναι λόγω του χαμηλού κόστους και των διαφόρων ιδιοτήτων της, όπως η ομαλή και μη μονοτονική φύση της, δεν υπάρχει άνω όριο, οριοθετείται κάτω από την ιδιότητα και έτσι βελτιώνει την απόδοσή της σε σύγκριση με άλλες δημοφιλείς συναρτήσεις ενεργοποίησης όπως η συνάρτηση ReLU και βοηθά στην επίτευξη ισχυρών αποτελεσμάτων κανονικοποίησης (ταιριάζει σωστά στο μοντέλο). Στο σχήμα 2.9 παρουσιάζεται η γραφική παράσταση της συνάρτησης Mish.



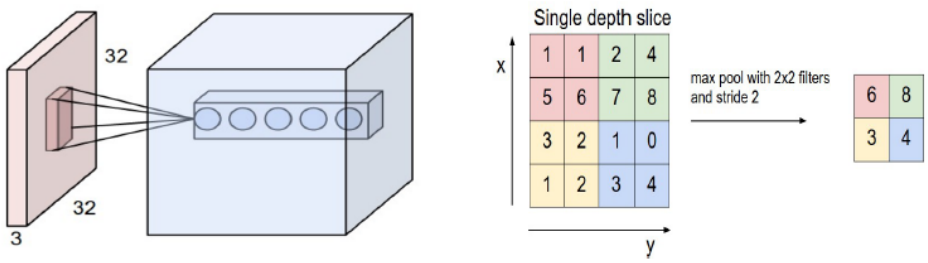
Σχήμα 2.9: Γραφική παράσταση της συνάρτησης ενεργοποίησης Mish

Οι γραφικές παραστάσεις της σιγμοειδούς συνάρτησης, της υπερβολικής εφαπτομένης, της ReLu και της Leaky ReLu παρουσιάζονται στο σχήμα 2.7.

2.7.3 Επίπεδα χωρικής υποδειγματοληψίας / Pooling layers

Ένα στρώμα συγκέντρωσης εφαρμόζεται συνήθως μετά τη συνάρτηση και τη λειτουργία ενεργοποίησης σε ένα CNN. Το στρώμα συγκέντρωσης μειώνει το μέγεθος του όγκου εισόδου χωρικά και συνεπώς μειώνεται ο αριθμός των παραμέτρων που απαιτούνται για την εκπαίδευση του CNN. Βοηθά επίσης στην αποφυγή της υπερεκπαίδευσης (overfitting), όταν δηλαδή ένα μοντέλο μαθαίνει τις λεπτομέρειες και τον θόρυβο στα δεδομένα της εκπαίδευσης σε βαθμό που επηρεάζει αρνητικά την απόδοση του μοντέλου σε νέα δεδομένα [40]. Η πιο συχνά χρησιμοποιούμενη λειτουργία συγκέντρωσης είναι η μέγιστη συγκέντρωση, στην οποία το μέγιστο μιας περιοχής $n * n$ λαμβάνεται και αντιστοιχείται στην έξοδο, όπου n είναι το μέγεθος του φίλτρου συγκέντρωσης (Σχήμα 2.10(b)). Ένας άλλος κοινός συνδυασμός είναι ο μέσος όρος συγκέντρωσης στην οποία ο μέσος όρος της περιοχής λαμβάνεται και

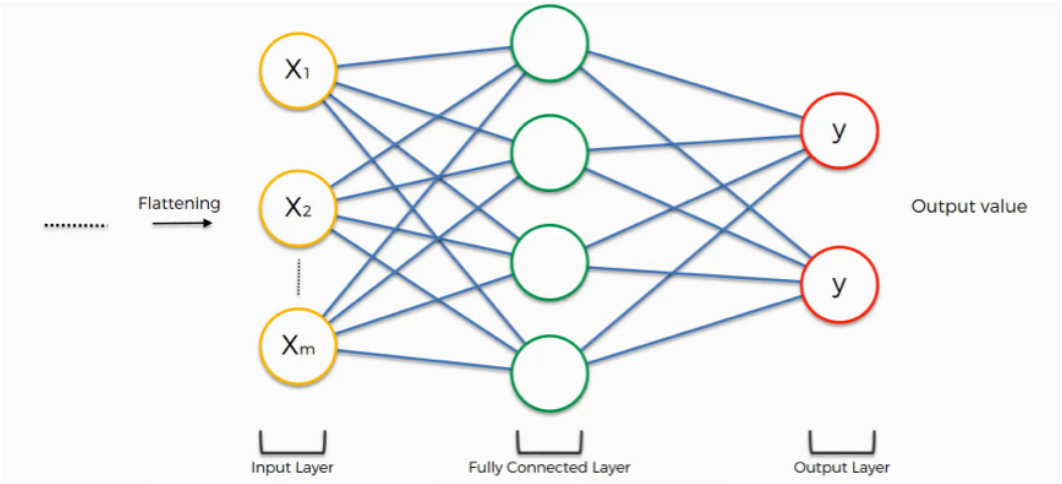
αντιστοιχεί στην έξοδο. Το φίλτρο συγκέντρωσης εφαρμόζεται ανεξάρτητα σε όλα τα βάθη του όγκου εισόδου και ως εκ τούτου το βάθος του όγκου εξόδου είναι ίδιο με το βάθος του όγκου εισόδου.



Σχήμα 2.10: Επίπεδα χωρικής υποδειγματοληψίας ενός CNN [3].

2.7.4 Πλήρως συνδεδεμένα επίπεδα

Ένα πλήρως συνδεδεμένο επίπεδο εφαρμόζεται συνήθως μετά από μια σειρά συνελικτικών επιπέδων και επιπέδων χωρικής υποδειγματοληψίας σε ένα CNN. Σε ένα πλήρως συνδεδεμένο επίπεδο, κάθε νευρώνας συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Έτσι, το επίπεδο είναι υπεύθυνο για τη συσσώρευση όλης της πληροφορίας από χαρακτηριστικά χαμηλότερων επιπέδων. Τέλος, δε μπορεί να εφαρμοστεί κάποιο συνελικτικό επίπεδο μετά από ένα πλήρως συνδεδεμένο επίπεδο σε ένα CNN.



Σχήμα 2.11: Παράδειγμα ενός πλήρως συνδεδεμένου επιπέδου [5].

2.7.5 Επίπεδο μαζικής κανονικοποίησης

Το επίπεδο κανονικοποίησης εισήχθη από τους Ioffe & Szegedy [41] για την αντιμετώπιση του προβλήματος της εσωτερικής μεταβλητής μεταβολής σε βαθιά νευρω-

νικά δίκτυα. Η εσωτερική μεταβλητή μεταβολή είναι η κατανομή των αλλαγών της εισόδου σε εσωτερικά επίπεδα του δικτύου, με μεγαλύτερο αντίκτυπο στα βαθύτερα επίπεδα. Δεδομένου ότι η είσοδος σε οποιοδήποτε επίπεδο επηρεάζεται από τις παραμέτρους όλων των προηγούμενων επιπέδων, οι μικρές αλλαγές στα αρχικά επίπεδα ενισχύονται σε βαθύτερα στρώματα και έτσι οδηγούν σε αλλαγή στη κατανομή. Το επίπεδο μαζικής κανονικοποίησης κάνει τις εισόδους μιας παρτίδας να κανονικοποιηθούν με εκ νέου κεντράρισμα και κλιμάκωση. Σε μαθηματική αναπαράσταση, εάν το m είναι ο μέσος όρος και το var είναι η διακύμανση της παρτίδας εισόδου, η κανονικοποιημένη έξοδος y'_i για την είσοδο y_i αναπαρίσταται σύμφωνα με τον τύπο 2.9.

$$y'_i = \frac{y_i - m}{\sqrt{var \ + e}} \tag{2.9}$$

όπου το e είναι μια μικρή σταθερή τιμή. Μετά την κανονικοποίηση, οι τιμές κλιμακώνονται και μετατοπίζονται σύμφωνα με τον τύπο 2.10.

$$z_i = \ \gamma * y'_i + \beta \tag{2.10}$$

όπου τα γ και β είναι παράμετροι που μαθαίνουμε. Με αυτόν τον τρόπο, μπορούν να χρησιμοποιηθούν υψηλότερα ποσοστά μάθησης σε ένα δίκτυο μετά την προσθήκη αυτού του επιπέδου.

2.7.6 Επίπεδο κανονικοποίησης

Η κανονικοποίηση γίνεται στα νευρωνικά δίκτυα για την αποφυγή του προβλήματος της υπερεκπαίδευσης (overfitting). Καθώς ένα νευρωνικό δίκτυο έχει μεγάλο αριθμό παραμέτρων, μπορεί εύκολα να εκπαιδευτεί ένα μεγάλο σύνολο δεδομένων αλλά μπορεί να μην είναι σε θέση να το γενικεύσει καλά. Οι πιο γνωστές μέθοδοι κανονικοποίησης είναι η κανονικοποίηση L1, η κανονικοποίηση L2 και η απόσυρση (dropout).

Κανονικοποίηση L2

Η κανονικοποίηση L2 προσθέτει έναν όρο κανονικοποίησης στη συνάρτηση απόκλισης, έτσι ώστε τα βάρη που τροποποιούνται κατά τη διάρκεια της εκπαίδευσης

να μη μπορούν να επιτύχουν πολύ υψηλές τιμές. Η κανονικοποίηση L2 τιμωρεί το άθροισμα των τετραγωνικών βαρών σύμφωνα με τον τύπο 2.11.

$$L = \sum_x L_x + \lambda \, \frac{1}{2} \sum_j \left| w_j \right|^2 \tag{2.11}$$

όπου λ είναι μια σταθερά που ελέγχει τη στάθμιση του όρου κανονικοποίησης.

Κανονικοποίηση L1

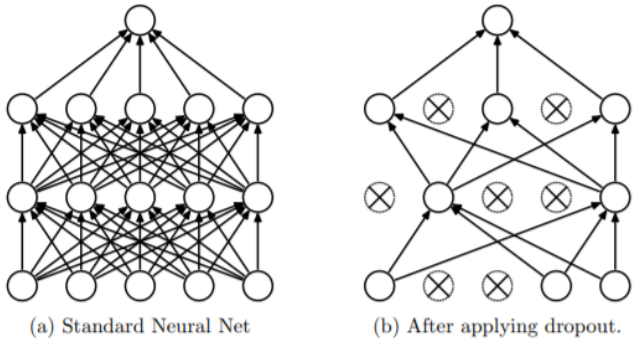
Αυτή η μέθοδος είναι παρόμοια με την κανονικοποίηση L2, αλλά ο όρος κανονικοποίησής της χρησιμοποιεί άθροισμα απόλυτων βαρών (και όχι υψωμένο στο τετράγωνο). Η μαθηματική αναπαράσταση της κανονικοποίησης L1 παρουσιάζεται στον τύπο 2.12.

$$L = \sum_x L_x + \lambda \, \frac{1}{2} \sum_j \left| w_j \right| \tag{2.12}$$

όπου και πάλι το λ είναι μια σταθερά που ελέγχει τη στάθμιση του όρου κανονικοποίησης.

Απόσυρση

Η απόσυρση (dropout) αποτελεί τεχνική των νευρωνικών δικτύων βαθιάς μάθησης με στόχο να αντιμετωπίσει το πρόβλημα της υπερβολικής εκπαίδευσης [6]. Η βασική της ιδέα είναι να αποσυρθούν τυχαία μερικοί νευρώνες μαζί με τις συνδέσεις τους κατά τη διάρκεια της εκπαίδευσης όπως φαίνεται στο Σχήμα 2.12. Σε κάθε επανάληψη της εκπαίδευσης τυχαίοι νευρώνες αποσύρονται, έτσι ένα διαφορετικό νευρωνικό δίκτυο εκπαιδεύεται κάθε φορά, δίνοντας ένα πιο γενικευμένο δίκτυο τη στιγμή της δοκιμής. Η απόσυρση μπορεί να θεωρηθεί ως εκπαίδευση σε σύνολο όπου περισσότεροι από ένας αδύναμοι κατηγοριοποιητές εκπαιδεύονται ξεχωριστά και έτσι μαθαίνει διαφορετικές πτυχές των δεδομένων. Τη στιγμή της δοκιμής, τα αποτελέσματα όλων των αδύναμων κατηγοριοποιητών συνδυάζονται για να δώσουν έναν τελικό κατηγοριοποιητή, ο οποίος είναι πιο γενικευμένος.



Σχήμα 2.12: Η αριστερή εικόνα δείχνει το νευρωνικό δίκτυο χωρίς απόσυρση. Η δεξιά εικόνα δείχνει το νευρωνικό δίκτυο μετά την εφαρμογή της απόσυρσης. Οι νευρώνες αποσύρονται τυχαία [6].

2.7.7 Επίπεδα εξόδου

Η έξοδος από συνελικτικά και συγκεντρωτικά επίπεδα αντιπροσωπεύουν χαρακτηριστικά υψηλού επιπέδου της εικόνας εισόδου. Μετά την εξαγωγή χαρακτηριστικών τα δεδομένα κατηγοριοποιούνται σε διάφορες κατηγορίες. Αυτό μπορεί να γίνει χρησιμοποιώντας ένα πλήρως συνδεδεμένο επίπεδο. Πλήρως συνδεδεμένα επίπεδα ενεργούν με τον ίδιο τρόπο όπως ένα κανονικό νευρωνικό δίκτυο, καθώς έχουν πλήρη σύνδεση με όλες τις ενεργοποιήσεις του προηγούμενου επιπέδου.

2.7.8 Συναρτήσεις απόκλισης

Ένα νευρωνικό δίκτυο αναπροσαρμόζεται από μια συνάρτηση απόκλισης κατά τη στιγμή της εκπαίδευσης. Μια συνάρτηση απόκλισης (ονομάζεται επίσης συνάρτηση κόστους) μετρά πόσο καλά μαθαίνει το δίκτυο συγκρίνοντας την αναμενόμενη έξοδο με την πραγματική έξοδο. Μια συνάρτηση απόκλισης δίνει μια μεμονωμένη τιμή η οποία λέει πόσο «καλό» είναι το δίκτυο. Μια συνάρτηση απόκλισης έχει τις ακόλουθες απαιτήσεις:

- Πρέπει να είναι δυνατή η καταγραφή της συνολικής απόκλισης ως μέσος όρος της απόκλισης του δείγματος.

$$L = \frac{1}{N} \sum_{x} L_x \tag{2.13}$$

  όπου N είναι ο συνολικός αριθμός δειγμάτων στην εκπαίδευση και L_x είναι η απόκλιση για κάθε ένα δείγμα εκπαίδευσης x.

- Πρέπει να είναι δυνατή η καταγραφή της απόκλισης μόνο ως συνάρτηση των εξόδων του δικτύου, χωρίς τη χρήση των εξόδων των εσωτερικών επιπέδων.

Οι πιο συχνά χρησιμοποιούμενες συναρτήσεις απόκλισης είναι:

- **Τετραγωνική απόκλιση**

  Η τετραγωνική απόκλιση (Square Loss) είναι μια απόκλιση για τη διαφορά στις προβλεπόμενες και αναμενόμενες εξόδους. Χρησιμοποιείται κυρίως στην παλινδρόμηση. Η μαθηματική μορφή για αυτήν τη συνάρτηση παρουσιάζεται στον τύπο 2.14.

$$L_{SQ} = \frac{1}{2}(t-y)^2 \tag{2.14}$$

  όπου t είναι η αναμενόμενη έξοδος και y είναι η προβλεπόμενη έξοδος. Αυτή η απόκλιση είναι κυρτή και διαφοροποιήσιμη και ως εκ τούτου μπορεί να ελαχιστοποιηθεί με κλίση κατάβασης.

- **Απόκλιση Hinge**

  Η απόκλιση Hinge χρησιμοποιείται κυρίως για την κατηγοριοποίηση στις μηχανές υποστήριξης διανυσμάτων. Για δυαδικό κατηγοριοποιητή, η μαθηματική μορφή της συνάρτησης παρουσιάζεται στον τύπο 2.15.

$$L_{HG} = max(0, 1 - t \cdot y) \tag{2.15}$$

  όπου t μπορεί να έχει τιμή είτε +1 ή -1 αντιστοιχώντας σε δύο τάξεις και το y είναι η βαθμολογία κατάταξης που δίνεται από τον κατηγοριοποιητή.

- **Λογαριθμική απόκλιση**

  Η λογαριθμική συνάρτηση απόκλισης χρησιμοποιείται στην κατηγοριοποίηση και η μαθηματική μορφή της παρουσιάζεται στον τύπο 2.16.

$$L_{LL} = -\sum_{c}^{C} y_c log(p_c) \tag{2.16}$$

  όπου C είναι ο συνολικός αριθμός κλάσεων, το y_c είναι μια δυαδική μεταβλητή που δείχνει εάν το c είναι η πραγματική κλάση για αυτό το δείγμα και το p_c είναι η βαθμολογία τάξης που προβλέπεται για την κατηγορία c.

- **Λογιστική απόκλιση**

  Η λογιστική απόκλιση είναι η λογαριθμική απόκλιση σε συνδυασμό με τη σιγμοειδή συνάρτηση. Για δυαδική κατηγοριοποίηση παρουσιάζεται σύμφωνα με τον τύπο 2.17.

$$L_{LO} = \ log\ (1\ +\ exp(-t \cdot y)) \tag{2.17}$$

  όπου t μπορεί να έχει τιμή +1 ή -1 και το y είναι η βαθμολογία πρόβλεψης. Αυτή η συνάρτηση ομαλοποιεί τη βαθμολογία σε τιμή πιθανότητας χρησιμοποιώντας τη σιγμοειδή συνάρτηση.

2.8 Οπισθοδιάδοση

Η μέθοδος οπισθοδιάδοσης (backpropagation) του λάθους είναι η πιο γνωστή μέθοδος σήμερα για την εκπαίδευση ενός νευρωνικού δικτύου που αποτελείται από πολλά επίπεδα, και χρησιμοποιείται στις πιο πολλές εφαρμογές. Η μέθοδος οπισθοδιάδοσης ακολουθεί έναν αλγόριθμο βελτιστοποίησης βασιζόμενο στην κλίση που εκμεταλλεύεται τον κανόνα της αλυσίδας [42]. Το κύριο χαρακτηριστικό της οπισθοδιάδοσης είναι η επαναληπτική, αναδρομική και αποδοτική μέθοδος για τον υπολογισμό των ανανεώσεων των βαρών για τη βελτίωση του δικτύου έως ότου είναι σε θέση να εκτελέσει το έργο για το οποίο εκπαιδεύεται. Είναι στενά συνδεδεμένη με τον αλγόριθμο Gauss-Newton. Η κεντρική ιδέα είναι αρκετά απλή: το δίκτυο ξεκινά τη διαδικασία μάθησης από τυχαίες τιμές των βαρών του. Εάν δώσει λάθος απάντηση, που είναι και το πιο πιθανό, τότε τα βάρη διορθώνονται έτσι ώστε το λάθος να γίνει μικρότερο. Η ίδια διαδικασία επαναλαμβάνεται αρκετές φορές έτσι ώστε σταδιακά το λάθος ελαττώνεται μέχρι να γίνει πολύ μικρό και ανεκτό. Στο σημείο αυτό λέμε ότι το δίκτυο έχει μάθει τα παραδείγματα που του δώσαμε για εκπαίδευση με την ακρίβεια που θέλαμε να μάθει.

2.9 Παλινδρόμηση

Η παλινδρόμηση είναι μια ευρέως χρησιμοποιούμενη στατιστική τεχνική μοντελοποίησης για την έρευνα της συσχέτισης μεταξύ μίας εξαρτώμενης μεταβλητής και

μιας ή περισσότερων ανεξάρτητων μεταβλητών. Χρησιμοποιείται με σκοπό την εκχώρηση δεδομένων σε μία πραγματική μεταβλητή πρόβλεψης, όπως ισχύει και στην περίπτωση της κατηγοριοποίησης όταν είναι διακριτή, αλλιώς καλείται παλινδρόμηση αν η μεταβλητή είναι συνεχής. Η παλινδρόμηση προϋποθέτει ότι τα σχετικά δεδομένα ταιριάζουν με μερικά γνωστά είδη συνάρτησης και μετά καθορίζει την καλύτερη συνάρτηση αυτού του είδους που μοντελοποιεί τα δεδομένα που έχουν δοθεί. Αποτέλεσμα της παλινδρόμησης όταν χρησιμοποιείται ως τεχνική εξόρυξης δεδομένων, αποτελεί ένα μοντέλο που χρησιμοποιείται αργότερα για να προβλέψει τις τιμές της κατηγορίας για τα νέα δεδομένα. Τέτοια παραδείγματα εφαρμογής της παλινδρόμησης αποτελεί η πρόβλεψη της ζήτησης για ένα νέο προϊόν ή υπηρεσία συναρτήσει των δαπανών διαφήμισης ή ο υπολογισμός της ταχύτητας του ανέμου σε σχέση με τη θερμοκρασία, την υγρασία και την ατμοσφαιρική πίεση του περιβάλλοντος.

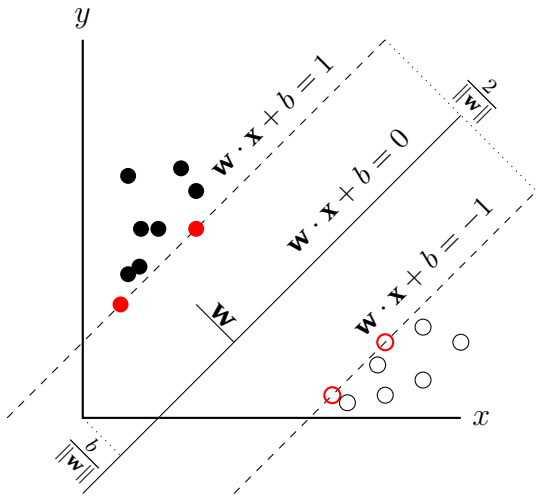
2.10 Μηχανές υποστήριξης διανυσμάτων

Οι Μηχανές Υποστήριξης Διανυσμάτων (Support Vector Machines - SVM) είναι ένα εποπτευόμενο μοντέλο μάθησης που μπορεί να χρησιμοποιηθεί τόσο για προβλήματα ταξινόμησης όσο και για παλινδρόμηση. Επιτυγχάνει τον σκοπό του κατασκευάζοντας έναν πολυδιάστατο υπερεπίπεδο (hyperplane) χώρο ο οποίος διαχωρίζει τις κλάσεις και λειτουργεί ως συνάρτηση απόφασης. Αυτό επιτυγχάνεται με τον προσδιορισμό του βέλτιστου υπερεπιπέδου, που ονομάζεται όριο απόφασης, το οποίο διαχωρίζει τα δεδομένα εκπαίδευσης με το μέγιστο περιθώριο. Η αρχική έκδοση των SVM προτάθηκε από τους Vapnik και Chervonenkis το 1963 [43]. Έπειτα οι Boser et al. [44], εισήγαγαν έναν αποτελεσματικό τρόπο χαρτογράφησης των δεδομένων εισόδου για εκπαίδευση σε υψηλότερο ή άπειρο χώρο διάστασης. Αυτό είναι γνωστό ως Kernel Trick. Το αποτέλεσμα είναι ότι επιτρέπει στις SVM να καταγράφουν αποτελεσματικά πιο πολύπλοκες σχέσεις μεταξύ των σημείων δεδομένων χωρίς να χρειάζεται να εκτελεί εκτεταμένους μετασχηματισμούς.

Οι SVM προκάλεσαν το ενδιαφέρον πολλών ερευνητών και εφαρμόστηκαν για την ανάπτυξη μοντέλων σε πλήθος προβλημάτων κατηγοριοποίησης. Μερικά κοινά παραδείγματα, πέρα από την ανίχνευση αντικειμένων είναι, για την ανίχνευση προσώπου, την ταξινόμηση κειμένου και υπερκειμένου, την ταξινόμηση εικόνων, τη βιο-

πληροφορική, την αναγνώριση χειρογράφων χαρακτήρων και την ανάλυση γεωχωρικών δεδομένων.

Στο σχήμα 2.13 παρουσιάζεται η βασική δομή μιας μηχανής υποστήριξης διανυσμάτων.

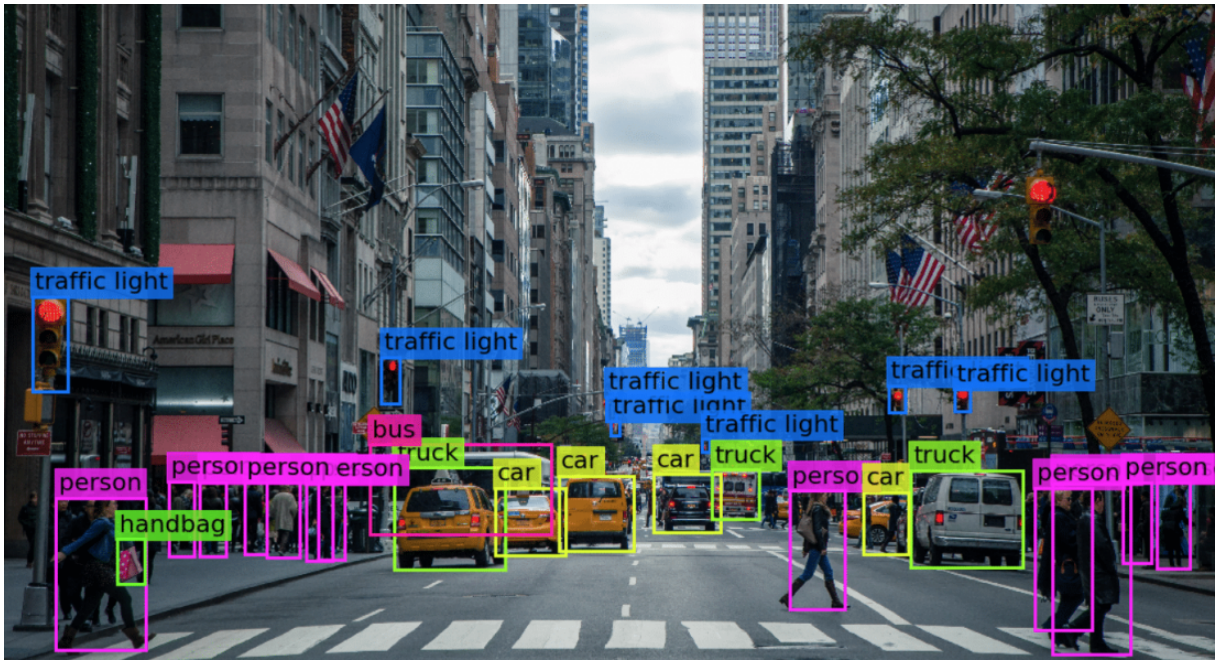


Σχήμα 2.13: Παράδειγμα μηχανής υποστήριξης διανυσμάτων.

Κεφάλαιο 3

Ανίχνευση Αντικειμένων

Η **Ανίχνευση αντικειμένων** (object detection) είναι μια τεχνική όρασης μέσω υπολογιστή (computer vision) που μας επιτρέπει να ανιχνεύουμε και να εντοπίζουμε αντικείμενα σε μια εικόνα ή σε ένα βίντεο. Με αυτό το είδος ταυτοποίησης και εντοπισμού, η ανίχνευση αντικειμένων μπορεί να χρησιμοποιηθεί για την καταμέτρηση αντικειμένων σε μια σκηνή και για τον προσδιορισμό και την παρακολούθηση των ακριβών τοποθεσιών τους, όλα αυτά με την ακριβή επισήμανση τους.



Σχήμα 3.1: Παράδειγμα ανίχνευσης αντικειμένων σε μία εικόνα.

Η ανίχνευση αντικειμένων συγχέεται συνήθως με την αναγνώριση εικόνων, οπότε πριν προχωρήσουμε, είναι σημαντικό να διευκρινίσουμε τις διαφορές μεταξύ τους. Η αναγνώριση εικόνας εκχωρεί μία ετικέτα σε μία εικόνα. Μια φωτογραφία ενός σκύλου λαμβάνει την ετικέτα ''σκύλος''. Μία φωτογραφία δύο σκύλων, εξακολουθεί

να λαμβάνει την ετικέτα ”σκύλος”. Ανίχνευση αντικειμένων, από την άλλη πλευρά, σχεδιάζει ένα πλαίσιο οριοθέτησης γύρω από κάθε σκύλο και επισημαίνει το πλαίσιο με την ετικέτα ”σκύλος”. Το μοντέλο προβλέπει πόσο είναι κάθε αντικείμενο και ποια ετικέτα πρέπει να εφαρμοστεί. Με αυτόν τον τρόπο, η ανίχνευση αντικειμένων παρέχει περισσότερες πληροφορίες σχετικά με μια εικόνα από την αναγνώριση.

3.1 Λειτουργίες και τύποι ανίχνευσης αντικειμένων

Σε γενικές γραμμές, η ανίχνευση αντικειμένων μπορεί να αναλυθεί σε προσεγγίσεις βασισμένες στη μηχανική μάθηση και προσεγγίσεις βασισμένες στη βαθιά μάθηση. Σε πιο παραδοσιακές προσεγγίσεις που βασίζονται στη μηχανική μάθηση, οι τεχνικές μηχανικής όρασης χρησιμοποιούνται για να εξετάσουν διάφορα χαρακτηριστικά μιας εικόνας, όπως το χρωματικό ιστόγραμμα ή τα άκρα, για να προσδιορίσουν ομάδες εικονοστοιχείων που μπορεί να ανήκουν σε ένα αντικείμενο. Αυτά τα χαρακτηριστικά τροφοδοτούνται στη συνέχεια σε ένα μοντέλο παλινδρόμησης που προβλέπει τη θέση του αντικειμένου μαζί με την ετικέτα του. Από την άλλη πλευρά, οι προσεγγίσεις που βασίζονται στη βαθιά μάθηση χρησιμοποιούν συνελικτικά νευρωνικά δίκτυα (CNNs) για την πραγματοποίηση ανίχνευσης αντικειμένων χωρίς επιτήρηση, όπου τα χαρακτηριστικά δε χρειάζεται να οριστούν και να εξαχθούν χωριστά.

3.2 Πώς λειτουργεί η ανίχνευση αντικειμένων

Τα μοντέλα ανίχνευσης αντικειμένων βαθιάς μάθησης έχουν συνήθως δύο μέρη. Ένας κωδικοποιητής λαμβάνει μία εικόνα ως είσοδο και τρέχει μέσω μιας σειράς επιπέδων που μαθαίνουν να εξάγουν στατιστικά χαρακτηριστικά που χρησιμοποιούνται για τον εντοπισμό και την επισήμανση αντικειμένων. Οι έξοδοι από τον κωδικοποιητή διαβιβάζονται έπειτα σε έναν αποκωδικοποιητή, ο οποίος προβλέπει πλαίσια και ετικέτες για κάθε αντικείμενο.

Ο απλούστερος αποκωδικοποιητής είναι ένας καθαρός παλινδρομητής (regressor). Ο καταγραφέας είναι συνδεδεμένος με την έξοδο του κωδικοποιητή και προβλέπει τη θέση και το μέγεθος κάθε πλαισίου οριοθέτησης απευθείας. Η έξοδος του μοντέλου είναι το ζεύγος συντεταγμένων X, Y για το αντικείμενο και η έκτασή του

στην εικόνα. Αν και είναι απλός, αυτός ο τύπος μοντέλου είναι αρκετά περιορισμένος. Πρέπει να καθοριστεί ο αριθμός των πλαισίων από πριν. Αν, για παράδειγμα, η εικόνα που λαμβάνει ως είσοδο έχει δύο σκυλιά, αλλά το μοντέλο σχεδιάστηκε για να εντοπίσει ένα μόνο αντικείμενο, κάποιο από τα δύο θα εμφανιστεί χωρίς ετικέτα "σκύλος". Ωστόσο, αν γνωρίζετε τον αριθμό των αντικειμένων που πρέπει να προβλέψετε σε κάθε εικόνα από πριν, τα καθαρά μοντέλα που βασίζονται σε παλινδρομητή θα μπορούσαν να είναι μια καλή επιλογή.

Στη συνέχεια παρουσιάζονται πιο εξελιγμένοι αλγόριθμοι για εντοπισμό και ανίχνευση αντικειμένων.

3.3 Πλαίσιο ανίχνευσης αντικειμένων Viola-Jones

Το πλαίσιο ανίχνευσης αντικειμένων Viola – Jones είναι το πρώτο πλαίσιο ανίχνευσης αντικειμένων που παρέχει ανταγωνιστικά ποσοστά ανίχνευσης αντικειμένων σε πραγματικό χρόνο που προτάθηκαν το 2001 από τους Paul Viola και Michael Jones [45].

Παρόλο που μπορούσε να εκπαιδευτεί για να ανιχνεύσει πολλές κατηγορίες αντικειμένων, υποκινήθηκε κυρίως από το πρόβλημα της ανίχνευσης προσώπου.

Στη δημοσίευση του αλγορίθμου είχε παρουσιαστεί ο εντοπισμός προσώπων σε πραγματικό χρόνο μέσω μίας web κάμερας. Ήταν η πιο εκπληκτική επίδειξη της όρασης του υπολογιστή και των δυνατοτήτων της εκείνη τη εποχή. Σύντομα, υλοποιήθηκε και στο OpenCV και η ανίχνευση προσώπου έγινε συνώνυμη με τον αλγόριθμο Viola-Jones.

Η βασική λογική του αλγορίθμου είναι η παρακάτω:

1. Δέχεται ένα σύνολο δεδομένων από πρόσωπα ως είσοδο.

2. Χρησιμοποιεί hardcoded χαρακτηριστικά προσώπων.

3. Εκπαιδεύει μια μηχανή υποστήριξης διανυσμάτων στο σύνολο των χαρακτηριστικών των προσώπων.

4. Χρησιμοποιεί αυτή τη μηχανή υποστήριξης διανυσμάτων για να ανιχνεύσει πρόσωπα.

```
Algorithm: Viola-Jones Face Detection Algorithm
Input: original test image
Output: image with face indicators as rectangles
for i ← 1 to num of scales in pyramid of images do
    Downsample image to create image_i
    Compute integral image, image_ii
    for j ← 1 to num of shift steps of sub-window do
        for k ← 1 to num of stages in cascade classifier do
            for l ← 1 to num of filters of stage k do
                Filter detection sub-window
                Accumulate filter outputs
            end for
            if accumulation fails per-stage threshold then
                Reject sub-window as face
                Break this k for loop
            end if
        end for
        if sub-window passed all per-stage checks then
            Accept this sub-window as a face
        end if
    end for
end for
```

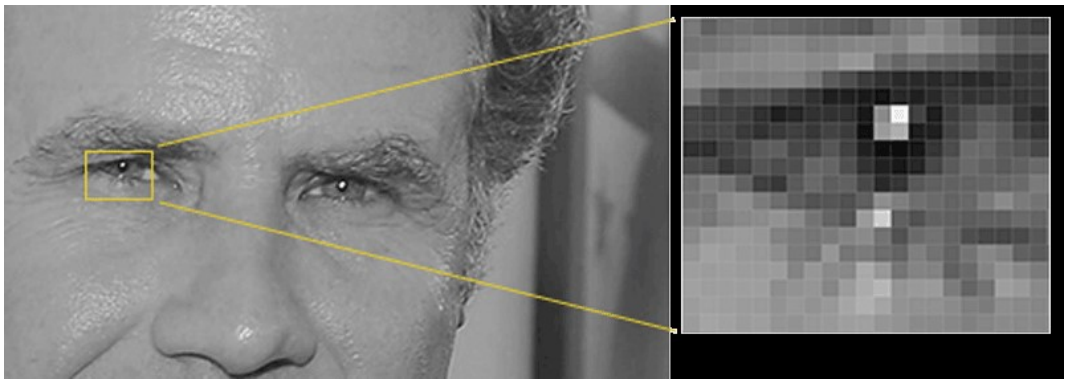
Σχήμα 3.2: Ο αλγόριθμος του πλαισίου ανίχνευσης αντικειμένων Viola-Jones [7].

Βασικό μειονέκτημα του αλγορίθμου ήταν ότι δε μπορούσε να αναγνωρίσει πρόσωπα σε άλλους προσανατολισμούς ή διαμορφώσεις (με κλίση, ανάποδα, φορώντας μάσκα κ.λ.π.)

3.4 Ιστογράμματα Προσανατολισμένων Κλίσεων

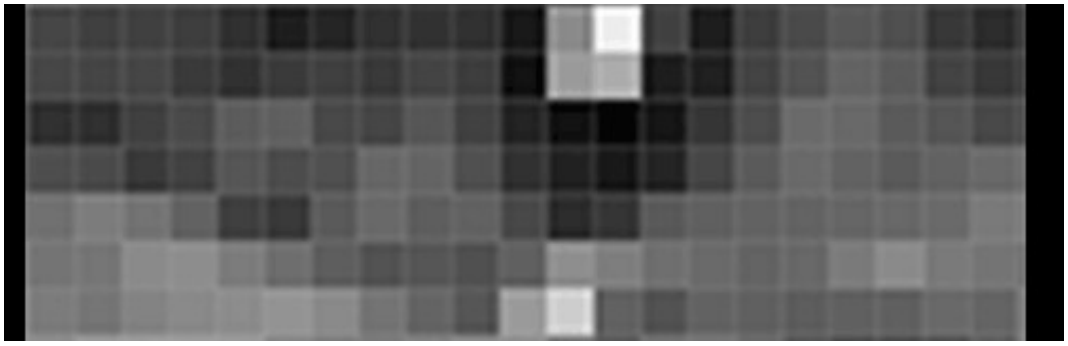
Ο αλγόριθμος Ιστογραμμάτων Προσανατολισμένων Κλίσεων (Histograms of Oriented Gradients) ή HOG προτάθηκε το 2005 από τους Navneet Dalal και Bill Triggs για εντοπισμό των περαστικών σε δρόμους και ήταν πιο αποδοτικός από τον αλγόριθμο Viola-Jones [46]. Η περιγραφή των χαρακτηριστικών τους, τα ιστογράμματα των προσανατολισμένων διαβαθμίσεων, ξεπέρασαν σημαντικά τους υπάρχοντες αλγορίθμους σε αυτήν την εργασία. Υπήρχαν αρκετά hardcoded χαρακτηριστικά όπως και στον προηγούμενο αλγόριθμο. Για κάθε εικονοστοιχείο (pixel), ο HOG, κοιτάει και τα εικονοστοιχεία που το περιβάλλουν άμεσα, όπως φαίνεται στο σχήμα 3.3.

Ο στόχος είναι πόσο πιο σκοτεινό είναι το τρέχον εικονοστοιχείο σε σύγκριση



Σχήμα 3.3: Ιστογράμματα Προσανατολισμένων Κλίσεων: Για κάθε ένα εικονοστοιχείο, κοιτάει και τα εικονοστοιχεία που το περιβάλλουν [8].

με τα γύρω εικονοστοιχεία. Στη συνέχεια σχεδιάζει ένα βέλος που δείχνει σε ποια κατεύθυνση γίνεται πιο σκοτεινή η εικόνα.



Σχήμα 3.4: Παράδειγμα σύγκρισης ενός εικονοστοχείου με τα γύρω εικονοστοιχεία για το που γίνονται πιο σκοτεινά [8].

- Επαναλαμβάνεται αυτή η διαδικασία για κάθε εικονοστοιχείο στην εικόνα.
- Κάθε εικονοστοιχείο αντικαθίσταται από ένα βέλος. Αυτά τα βέλη ονομάζονται ντεγκραντέ (gradient).
- Οι κλίσεις δείχνουν τη ροή από το φως στο σκοτάδι σε ολόκληρη την εικόνα.

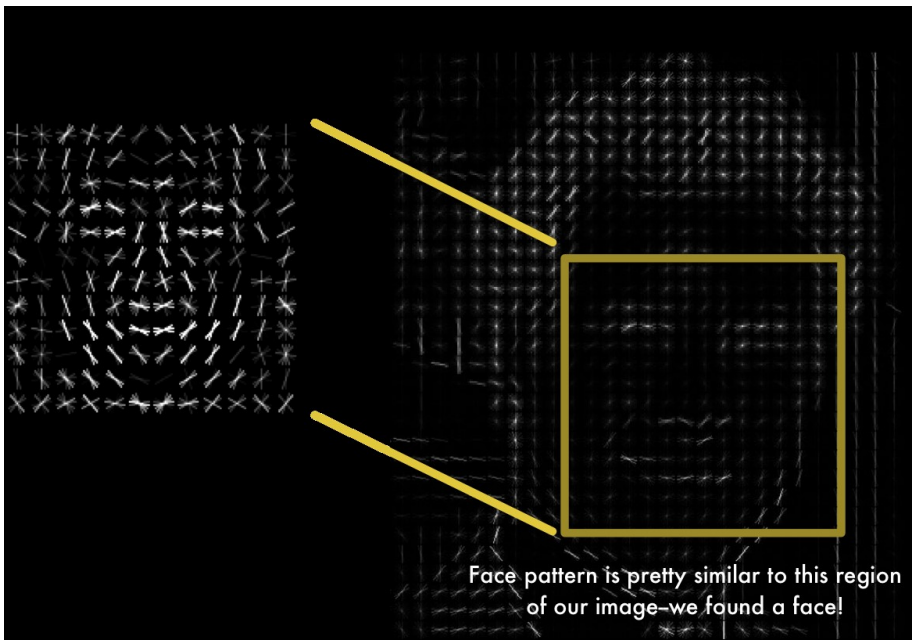
Έπειτα:

- Χωρίζει την εικόνα σε μικρά τετράγωνα των 16x16 εικονοστοιχεία το καθένα.
- Σε κάθε τετράγωνο, μετράει πόσες κλίσεις δείχνουν σε κάθε μεγάλη κατεύθυνση.
- Τελικό αποτέλεσμα είναι: η αρχική εικόνα μετατράπηκε σε απλή αναπαράσταση που καταγράφει τη βασική δομή ενός προσώπου με απλό τρόπο.

Σχήμα 3.5: Παράδειγμα ροής κλίσεων από το φως στο σκοτάδι σε μια εικόνα [8].

- Στη συνέχεια, αντικαθιστά αυτό το τετράγωνο στην εικόνα με τις οδηγίες βέλους που ήταν οι ισχυρότερες.

Η ανίχνευση προσώπων σημαίνει ότι βρίσκουμε το τμήμα της εικόνας που μοιάζει περισσότερο με ένα γνωστό μοτίβο HOG που εξήχθη από ένα πλήθος άλλων προσώπων εκπαίδευσης.



Σχήμα 3.6: Παράδειγμα ανίχνευσης προσώπου με Ιστογράμματα Προσανατολισμένων Κλίσεων βάση μοτίβων που εξήχθησαν από ένα πλήθος άλλων προσώπων εκπαίδευσης. [8].

Βασικό μειονέκτημα αυτού του αλγορίθμου: Παρά το γεγονός ότι είναι καλό σε πολλές εφαρμογές, χρησιμοποιούσε ακόμα hardcoded δυνατότητες που αποτύγχαναν σε πιο γενικευμένες καταστάσεις με πολύ θόρυβο και περισπασμούς στο παρασκήνιο.

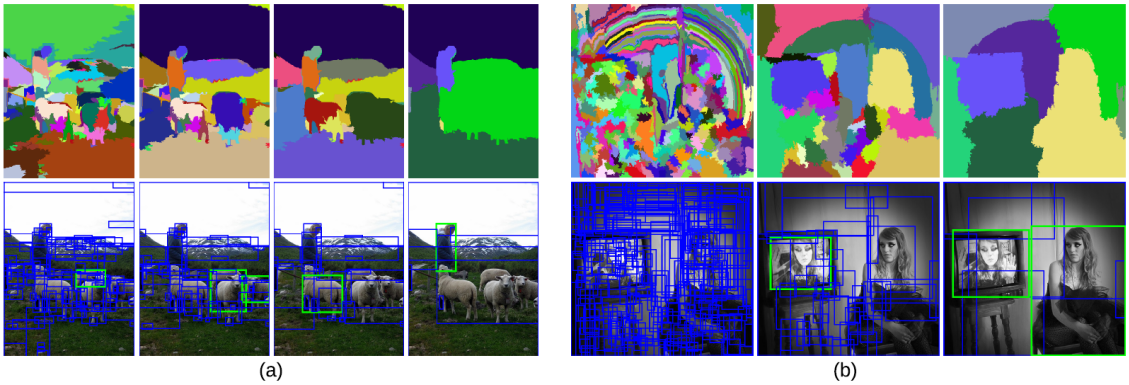
3.5 Αλγόριθμοι Προτάσεων Περιοχής

Με τη χρήση της βαθιάς μάθησης εμφανίζονται οι περιοχές με χαρακτηριστικά που αποτελούν πρωτοποριακή προσέγγιση στην ανίχνευση αντικειμένων. Σε αυτήν την ενότητα, θα δούμε το R-CNN και μια σειρά βελτιώσεων που έγιναν σε αυτό: Fast R-CNN , Faster R-CNN και Mask R-CNN.

3.5.1 R-CNN

Η εποχή του Deep Learning ξεκινάει με τον αλγόριθμο R-CNN [10]. Τα μοντέλα επιλέγουν πρώτα πολλές προτεινόμενες περιοχές από μια εικόνα (για παράδειγμα, τα πλαίσια αγκύρωσης είναι ένας τύπος μεθόδου επιλογής) και στη συνέχεια επισημαίνουν τις κατηγορίες τους και τα πλαίσια οριοθέτησης. Έπειτα, χρησιμοποιούν ένα νευρωνικό δίκτυο συνέλιξης για να εκτελούν υπολογισμούς προς τα εμπρός για να εξαγάγουν χαρακτηριστικά από κάθε προτεινόμενη περιοχή. Τέλος, χρησιμοποιούν τα χαρακτηριστικά κάθε προτεινόμενης περιοχής για να προβλέψουν τις κατηγορίες και τα πλαίσια οριοθέτησης τους.

Το R-CNN δημιουργεί πλαίσια οριοθέτησης ή προτάσεις περιοχής, χρησιμοποιώντας μια διαδικασία που ονομάζεται Επιλεκτική Αναζήτηση (Selective search) [9]. Αναλυτικότερα, η Επιλεκτική Αναζήτηση εξετάζει την εικόνα μέσω παραθύρων διαφορετικών μεγεθών και για κάθε μέγεθος προσπαθεί να ομαδοποιήσει γειτονικά εικονοστοιχεία ανά υφή, χρώμα ή ένταση για να αναγνωρίσει αντικείμενα.



Σχήμα 3.7: Δύο παραδείγματα της επιλεκτικής αναζήτησης που δείχνουν την αναγκαιότητα διαφορετικών κλιμάκων. Στα αριστερά βρίσκουμε πολλά αντικείμενα μεγέθη. Στα δεξιά βρίσκουμε απαραιτήτως τα αντικείμενα σε διαφορετικές κλίμακες, καθώς το κορίτσι περιέχεται από την τηλεόραση [9].

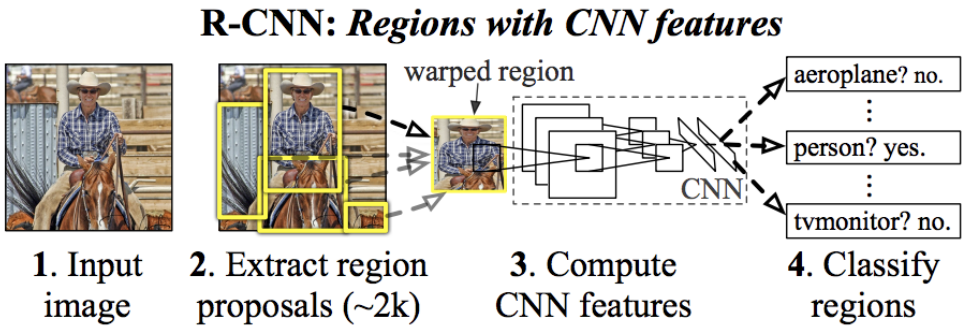
Στη συνέχεια, ο R-CNN:

1. Δημιουργεί ένα σύνολο προτάσεων για πλαίσια οριοθέτησης.

2. Περνάει τις εικόνες με πλαίσια οριοθέτησης μέσω ενός προ-εκπαιδευμένου μοντέλου και τέλος χρησιμοποιεί μια μηχανή υποστήριξης διανυσμάτων (SVM) για να αναγνωρίσει το κάθε ένα αντικείμενο στα πλαίσια οριοθέτησης.

3. Τέλος, ελέγχει το κάθε πλαίσιο μέσω ενός μοντέλου γραμμικής παλινδρόμησης για να εξάγει αυστηρότερες συντεταγμένες μόλις το αντικείμενο αναγνωριστεί.

Στο σχήμα 3.8 παρουσιάζεται η αρχιτεκτονική του R-CNN.



Σχήμα 3.8: Η αρχιτεκτονική του R-CNN [10].

Προβλήματα με το R-CNN

- Χρειάζεται πάρα πολύ χρόνος για να εκπαιδευτεί το δίκτυο, καθώς θα πρέπει να κατηγοριοποιηθούν 2000 προτάσεις περιοχής ανά εικόνα.

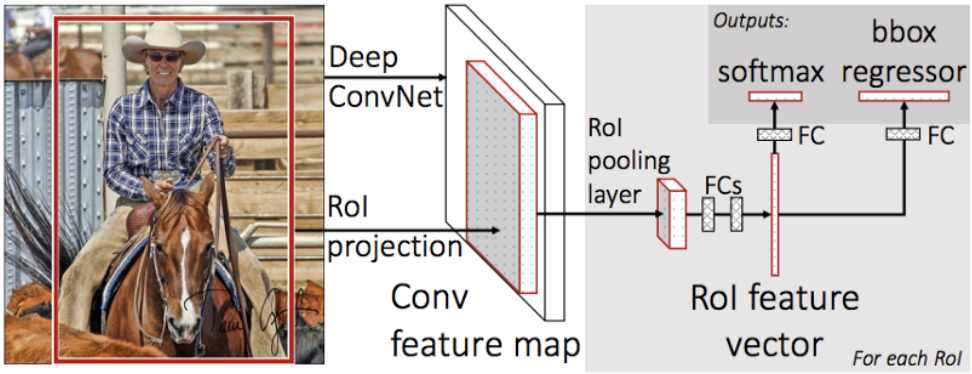
- Δε μπορεί να εφαρμοστεί σε πραγματικό χρόνο, διότι χρειάζονται περίπου 47 δευτερόλεπτα για κάθε εικόνα.

- Ο αλγόριθμος επιλεκτικής αναζήτησης είναι ένας σταθερός αλγόριθμος. Επομένως, δεν υπάρχει μάθηση σε αυτό το στάδιο. Αυτό θα μπορούσε να οδηγήσει στη δημιουργία κακών υποψήφιων προτάσεων για περιοχές.

3.5.2 Fast R-CNN

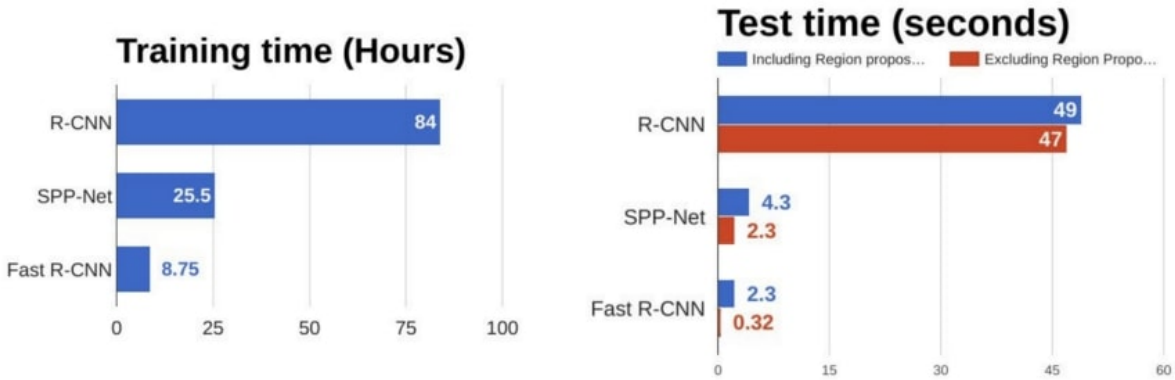
Ο ίδιος συγγραφέας της προηγούμενης εργασίας (R-CNN) έλυσε μερικά από τα μειονεκτήματα του R-CNN για να δημιουργήσει έναν ταχύτερο αλγόριθμο ανίχνευσης αντικειμένων, τον Fast R-CNN [11]. Η προσέγγιση είναι παρόμοια με τον

αλγόριθμο R-CNN. Όμως, αντί να τροφοδοτούμε τις προτάσεις της περιοχής στο CNN, τροφοδοτούμε την εικόνα εισόδου στο CNN για να δημιουργήσουμε ένα χάρτη χαρακτηριστικών. Από το χάρτη των συνελικτικών χαρακτηριστικών, εντοπίζουμε την περιοχή των προτάσεων, τις παραμορφώνουμε σε τετράγωνα και χρησιμοποιώντας ένα στρώμα ομαδοποίησης RoI, τα αναδιαμορφώνουμε σε σταθερό μέγεθος, έτσι ώστε να μπορεί να τροφοδοτηθεί σε ένα πλήρως συνδεδεμένο επίπεδο. Από το διάνυσμα χαρακτηριστικών RoI, χρησιμοποιούμε ένα επίπεδο softmax για να προβλέψουμε την κλάση της προτεινόμενης περιοχής και επίσης τις τιμές μετατόπισης για το πλαίσιο οριοθέτησης.



Σχήμα 3.9: Η αρχιτεκτονική του Fast R-CNN [11].

Ο λόγος για τον οποίο το Fast R-CNN είναι ταχύτερο από το R-CNN είναι επειδή δε χρειάζεται να τροφοδοτείται το συνελικτικό νευρωνικό δίκτυο με 2000 προτάσεις περιοχής κάθε φορά. Αντ' αυτού, η λειτουργία συνέλιξης γίνεται μόνο μία φορά ανά εικόνα και από αυτήν δημιουργείται ένας χάρτης χαρακτηριστικών.



Σχήμα 3.10: Σύγκριση χρόνων αλγορίθμων ανίχνευσης αντικειμένων R-CNN και Fast R-CNN [12].

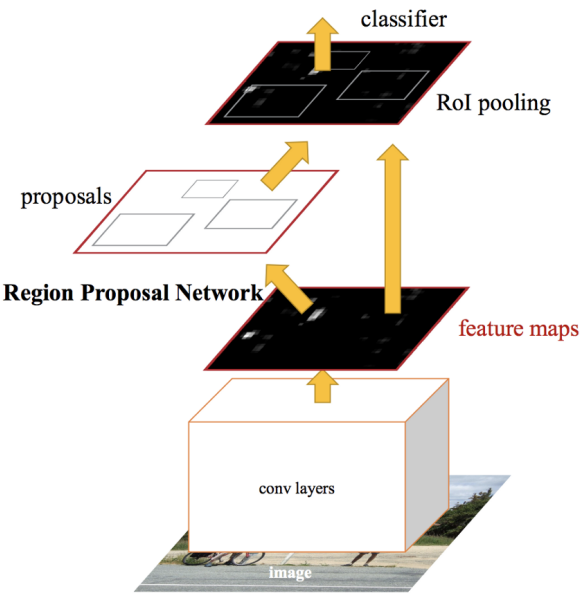
Από τα γραφήματα του σχήματος 3.10, διαπιστώνουμε ότι το Fast R-CNN είναι πολύ πιο γρήγορο τόσο σε εκπαίδευση όσο και ανιχνεύσεις σε σχέση με το R-CNN. Όταν ελέγχουμε την απόδοση του Fast R-CNN κατά τη διάρκεια του χρόνου δοκιμής, συμπεριλαμβανομένων των προτάσεων περιοχής επιβραδύνεται σημαντικά ο αλγόριθμος σε σύγκριση με τη μη χρήση προτάσεων περιοχής. Επομένως, οι προτάσεις περιοχής γίνονται εμπόδια στον αλγόριθμο Fast R-CNN καθώς επηρεάζουν την απόδοσή του.

Προβλήματα με το Fast R-CNN

Ο Fast R-CNN έχει και αυτός ορισμένα προβλήματα. Χρησιμοποιεί επίσης την επιλεκτική αναζήτηση ως μέθοδο προτάσεων για να βρει τις περιοχές ενδιαφέροντος, η οποία είναι μια αργή και χρονοβόρα διαδικασία. Χρειάζονται περίπου 2 δευτερόλεπτα ανά εικόνα για τον εντοπισμό αντικειμένων, κάτι που είναι πολύ καλύτερο σε σύγκριση με το R-CNN. Αλλά όταν εξετάζουμε μεγάλα σύνολα δεδομένων σε πραγματικό χρόνο, τότε ακόμη και ο Fast R-CNN δε φαίνεται πλέον τόσο γρήγορος.

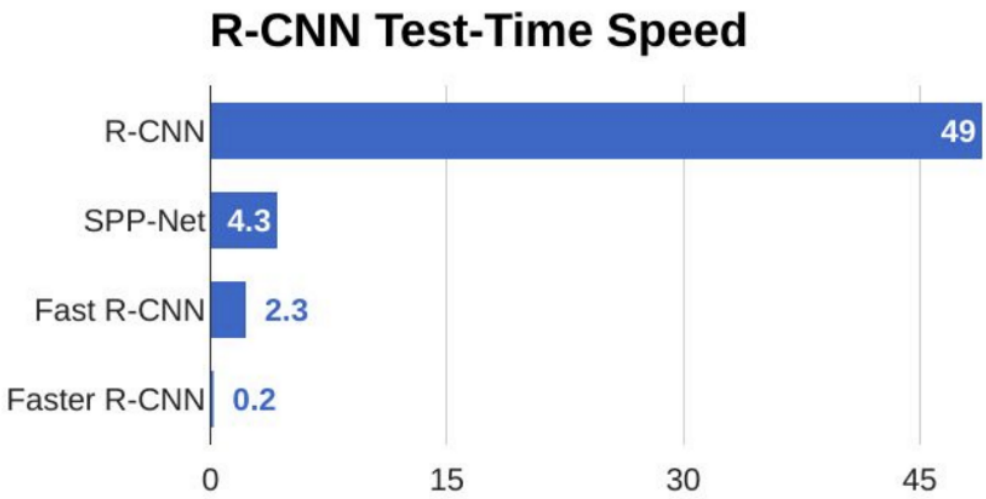
3.5.3 Faster-RCNN

Και οι δύο παραπάνω αλγόριθμοι (R-CNN & Fast R-CNN) χρησιμοποιούν επιλεκτική αναζήτηση για να μάθουν τις προτάσεις της περιοχής. Η επιλεκτική αναζήτηση είναι μια αργή και χρονοβόρα διαδικασία που επηρεάζει την απόδοση του δικτύου. Επομένως, οι Shaoqing Ren, Kaiming He, Ross Girshick και Jian Sun πρότειναν έναν αλγόριθμο ανίχνευσης αντικειμένων που εξαλείφει τον αλγόριθμο επιλεκτικής αναζήτησης και επιτρέπει στο δίκτυο να μάθει τις προτάσεις περιοχής [13].



Σχήμα 3.11: Η αρχιτεκτονική του Faster R-CNN [13].

Όπως και με το Fast R-CNN, η εικόνα παρέχεται ως είσοδος σε ένα συνελικτικό δίκτυο, το οποίο παρέχει έναν χάρτη συνελικτικών χαρακτηριστικών. Αντί να χρησιμοποιεί αλγόριθμο επιλεκτικής αναζήτησης στον χάρτη δυνατοτήτων για τον προσδιορισμό των προτάσεων περιοχής, χρησιμοποιείται ένα ξεχωριστό δίκτυο για την πρόβλεψη των προτάσεων περιοχής. Έπειτα, οι προβλεπόμενες προτάσεις περιοχής αναδιαμορφώνονται χρησιμοποιώντας ένα επίπεδο συγκέντρωσης περιοχών ενδιαφέροντος (RoI), το οποίο στη συνέχεια χρησιμοποιείται για την κατάταξη της εικόνας εντός της προτεινόμενης περιοχής και την πρόβλεψη των τιμών για τα πλαίσια οριοθέτησης.



Σχήμα 3.12: Σύγκριση χρόνων αλγορίθμων ανίχνευσης αντικειμένων R-CNN, Fast R-CNN και Faster R-CNN [12].

Από το παραπάνω γράφημα, φαίνεται ξεκάθαρα ότι η υλοποίησή του Faster R-CNN είναι πολύ πιο γρήγορη από τους R-CNN και Fast R-CNN. Επομένως, μπορεί να χρησιμοποιηθεί και για ανίχνευση αντικειμένων σχεδόν σε πραγματικό χρόνο καθώς ο χρόνος ανίχνευσης των αντικειμένων σε μια εικόνα είναι στα περίπου 0.2 δευτερόλεπτα.

Προβλήματα με το Faster R-CNN

Ο αλγόριθμος Faster R-CNN απαιτεί πολλά περάσματα σε κάθε εικόνα για την εξαγωγή όλων των αντικειμένων. Παράλληλα, καθώς αποτελείται από διαφορετικά συστήματα που λειτουργούν το ένα μετά το άλλο, η απόδοση των πιο μπροστά συστημάτων εξαρτάται από την απόδοση των προηγούμενων συστημάτων.

3.5.4 Mask R-CNN

Το Mask R-CNN είναι ένα υπερσύγχρονο μοντέλο για τμηματοποίηση αντικειμένων σε εικόνες και βίντεο, που αναπτύχθηκε μετά το Faster R-CNN. Με άλλα λόγια, μπορεί να χωρίσει διαφορετικά αντικείμενα σε μια εικόνα ή ένα βίντεο. Δέχεται μία εικόνα ως είσοδο, και έπειτα δίνει πλαίσια οριοθέτησης αντικειμένων, κλάσεις και τις μάσκες πάνω από τα αντικείμενα [47].

Σχήμα 3.13: Παράδειγμα τμηματοποίησης αντικειμένων σε μια εικόνα με το Mask-RCNN.

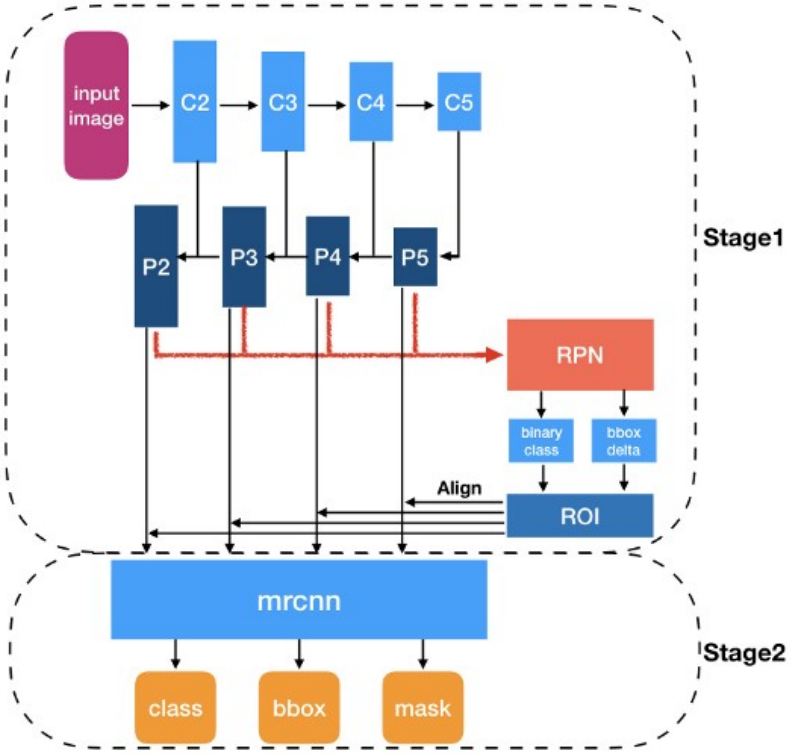
Υπάρχουν δύο στάδια του Mask R-CNN. Πρώτον, δημιουργεί προτάσεις σχετικά με τις περιοχές όπου μπορεί να υπάρχει ένα αντικείμενο βάσει της εικόνας εισαγωγής. Δεύτερον, προβλέπει την κλάση του αντικειμένου, βελτιώνει το πλαίσιο οριοθέτησης και δημιουργεί μια μάσκα σε επίπεδο pixel του αντικειμένου με βάση την πρόταση του πρώτου σταδίου. Και τα δύο επίπεδα συνδέονται με τη δομή του δικτύου backbone.

Το Backbone είναι ένα βαθύ νευρωνικό δίκτυο τύπου FPN [19]. Αποτελείται από ένα κάτω προς τα πάνω μονοπάτι, ένα πάνω-κάτω μονοπάτι και πλευρικές συνδέσεις. Η κατώτατη διαδρομή μπορεί να είναι οποιοδήποτε CNN, συνήθως ResNet ή VGG, το οποίο εξάγει λειτουργίες από ακατέργαστες εικόνες. Το μονοπάτι από πάνω προς τα κάτω δημιουργεί χάρτη πυραμίδας χαρακτηριστικών (FPN) που είναι παρόμοιο σε μέγεθος με το μονοπάτι από κάτω προς τα πάνω. Οι πλευρικές συνδέσεις είναι συνελίξεις και προσθέτουν λειτουργίες μεταξύ δύο αντίστοιχων επιπέδων των δύο διαδρομών. Το FPN ξεπερνά τα άλλα μεμονωμένα CNN κυρίως για τον λόγο ότι διατηρεί ισχυρά σημασιολογικά χαρακτηριστικά σε διάφορες κλίμακες ανάλυσης.

Στο πρώτο στάδιο του Mask R-CNN ένα μικρό νευρωνικό δίκτυο, που ονομάζεται RPN, σαρώνει όλα τα μονοπάτια FPN (χάρτης χαρακτηριστικών) από πάνω προς τα κάτω και προτείνει περιοχές που μπορεί να περιέχουν αντικείμενα. Ενώ η σάρωση του χάρτη χαρακτηριστικών είναι ένας αποτελεσματικός τρόπος, χρειάζεται μια μέθοδο για να συνδέσουμε τις λειτουργίες με την πρώτη θέση της εικόνας. Σε αυτό το σημείο εμφανίζονται οι άγκυρες. Οι άγκυρες (anchors) είναι ένα σύνολο πλαισίων με προκαθορισμένες τοποθεσίες και κλίμακες σε σχέση με τις εικόνες. Οι κλάσεις (μόνο το αντικείμενο ή το δυαδικό υπόβαθρο κατηγοριοποιούνται σε αυτό το στάδιο) και τα πλαίσια οριοθέτησης αντιστοιχίζονται σε μεμονωμένες αγκυρώσεις σύμφωνα με κάποια τιμή του IoU. Καθώς οι άγκυρες με διαφορετικές κλίμακες συνδέονται με διαφορετικά επίπεδα του χάρτη χαρακτηριστικών, το RPN χρησιμοποιεί αυτές τις άγκυρες για να καταλάβει από ποιο σημείο στο χάρτη «πρέπει» να πάρει ένα αντικείμενο και ποιο μέγεθος του πλαισίου οριοθέτησης του.

Στο δεύτερο στάδιο, ένα άλλο νευρωνικό δίκτυο παίρνει τις προτεινόμενες περιοχές από το πρώτο στάδιο και τις εκχωρεί σε αρκετές συγκεκριμένες περιοχές ενός επιπέδου χάρτη χαρακτηριστικών, σαρώνει αυτές τις περιοχές και δημιουργεί τά-

ξεις αντικειμένων (πολυ-κατηγοριοποιημένη ταξινόμηση), πλαίσια οριοθέτησης και μάσκες. Η διαδικασία μοιάζει με το RPN. Οι διαφορές είναι ότι χωρίς τη βοήθεια των αγκυρών, το δεύτερο στάδιο χρησιμοποιεί μια τεχνική που ονομάζεται ROIAlign για να εντοπίσει τις σχετικές περιοχές του χάρτη χαρακτηριστικών και υπάρχει ένας κλάδος που δημιουργεί μάσκες για κάθε αντικείμενο σε επίπεδο pixel.



Σχήμα 3.14: Η αρχιτεκτονική του Mask-RCNN για τμηματοποίηση αντικειμένων [14].

3.6 Ανιχνευτής Πολλαπλών Θυρίδων μιας Λήψης (SSD)

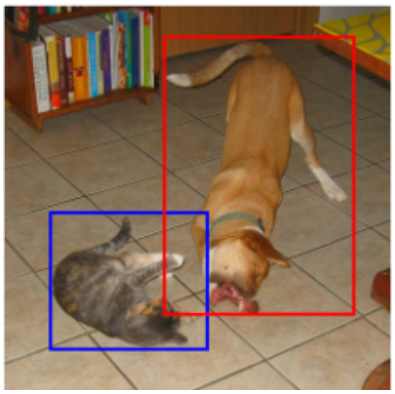
Η δημοσίευση SSD: Single Shot MultiBox Detector [15] κυκλοφόρησε στα τέλη Νοεμβρίου 2016 από τους Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu και Alexander C. Berg και έφτασε σε νέα ρεκόρ, όσον αφορά την απόδοση και την ακρίβεια για εργασίες εντοπισμού αντικειμένων, σημειώνοντας πάνω από 74% mAP (μέση ακρίβεια) στα 59 καρέ ανά δευτερόλεπτο σε τυπικά σύνολα δεδομένων, όπως Pascal VOC και COCO. Για να κατανοήσουμε καλύτερα το SSD, ας αναλύσουμε το όνομα αυτού του αλγορίθμου:

Single Shot: σημαίνει ότι οι διαδικασίες εντοπισμού και ταξινόμησης αντικειμένων εκτελούνται σε μία μόνο κίνηση προς τα εμπρός του δικτύου.

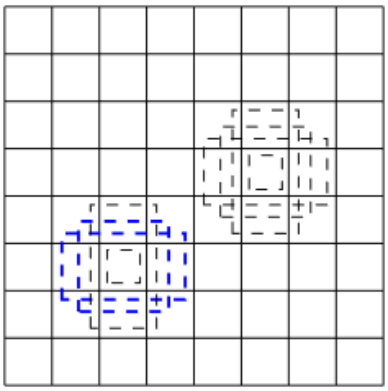
MultiBox: αυτό είναι το όνομα της τεχνικής για την παλινδρόμηση του πλαισίου οριοθέτησης.

Detector: Το δίκτυο είναι ένας ανιχνευτής αντικειμένων που κατηγοριοποιεί επίσης αυτά τα αντικείμενα που εντοπίστηκαν.

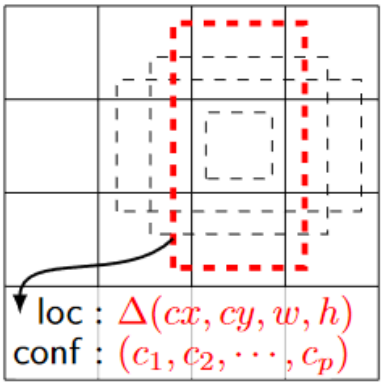
Οι SSD ανιχνευτές, σε αντίθεση με τον Faster R-CNN ο οποίος χρησιμοποιεί ένα υποδίκτυο για να προτείνει περιοχές, βασίζονται σε ένα σύνολο προκαθορισμένων περιοχών. Ένα πλέγμα σημείων αγκύρωσης τοποθετείται πάνω από την εικόνα εισόδου, και σε κάθε σημείο αγκύρωσης, πλαίσια πολλαπλών σχημάτων και μεγεθών χρησιμεύουν ως περιοχές. Για κάθε πλαίσιο σε κάθε σημείο αγκύρωσης, το μοντέλο εξάγει μια πρόβλεψη για το εάν υπάρχει ή όχι ένα αντικείμενο εντός της περιοχής και τροποποιήσεις στη θέση και το μέγεθος του πλαισίου για να ταιριάζει πιο κοντά στο αντικείμενο. Επειδή υπάρχουν πολλά πλαίσια σε κάθε σημείο αγκύρωσης και τα σημεία αγκύρωσης μπορεί να είναι κοντά το ένα με το άλλο, οι SSD ανιχνευτές παράγουν πολλές πιθανές ανιχνεύσεις που επικαλύπτονται. Στο τέλος, εκτελείται μια μετα-επεξεργασία στην έξοδο του SSD, προκειμένου να απομακρυνθούν οι περισσότερες από αυτές τις προβλέψεις και να επιλεγεί η καλύτερη.



Σχήμα 3.15: Εικόνα εισόδου με τα πλαίσια οριοθέτησης και τις ετικέτες των αντικειμένων [15].

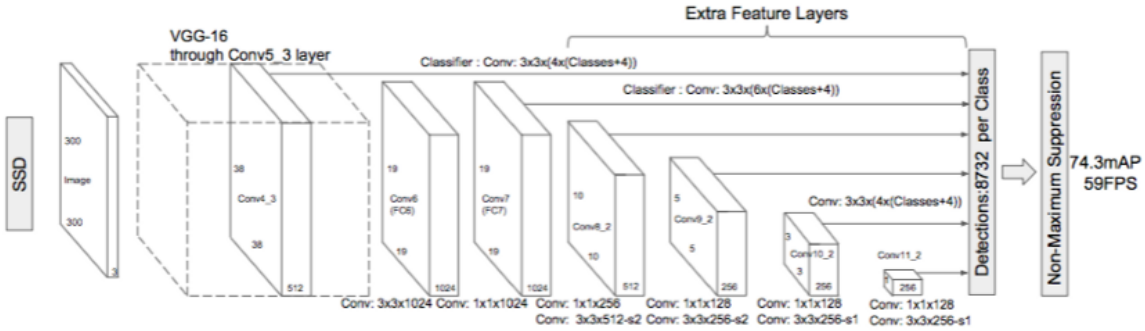


Σχήμα 3.16: 8×8 χάρτης χαρακτηριστικών με το πλέγμα των σημείων αγκύρωσης το οποίο τοποθετείται πάνω από την εικόνα εισόδου [15].



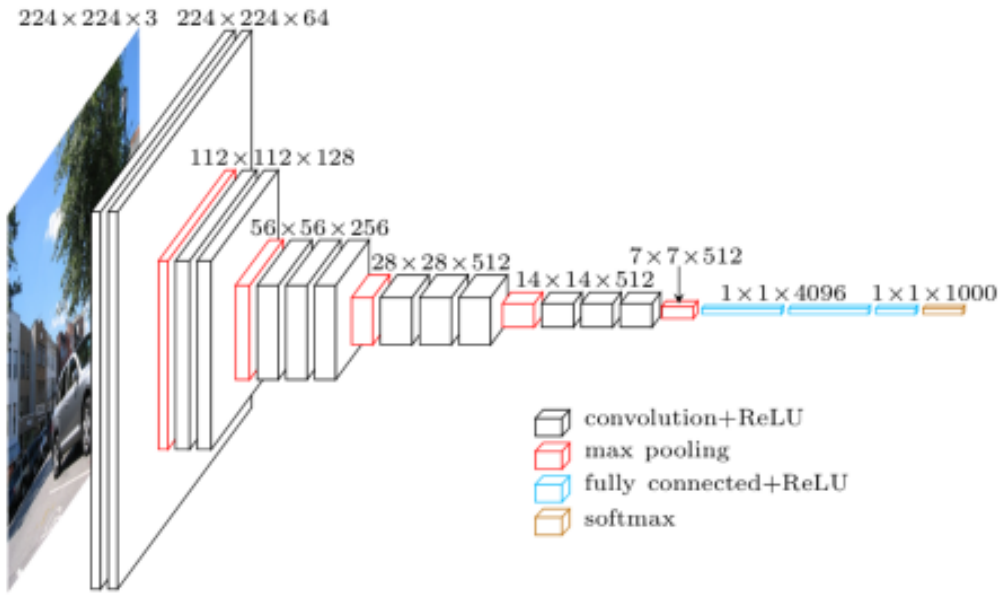
Σχήμα 3.17: 4×4 χάρτης χαρακτηριστικών με την τοποθεσία και την βαθμολογία εμπιστοσύνης των κλάσεων αντικειμένων [15].

Αρχιτεκτονική του δικτύου



Σχήμα 3.18: Αρχιτεκτονική του SSD [15].

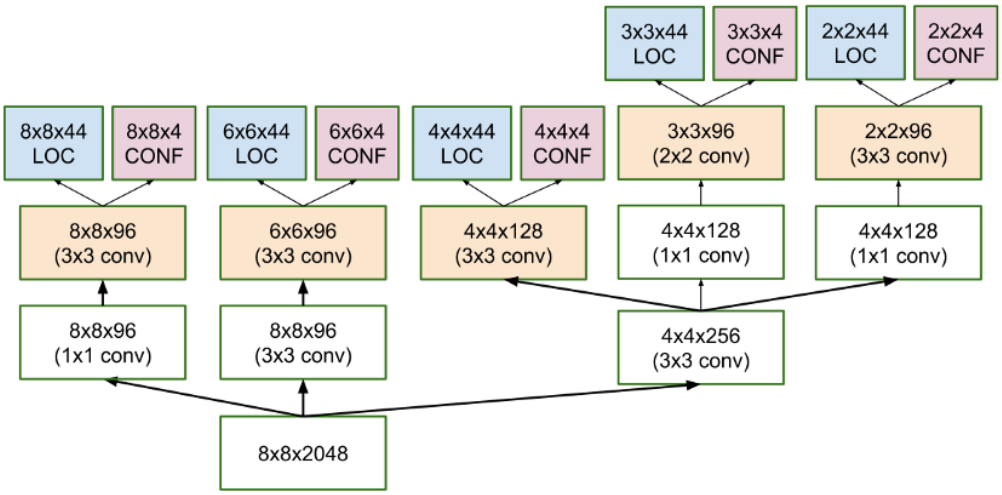
Όπως φαίνεται στο σχήμα 3.18, η αρχιτεκτονική του SSD βασίζεται στην αρχιτεκτονική του VGG-16 [48], αλλά χωρίς τα πλήρως συνδεδεμένα επίπεδα. Ο λόγος για τον οποίο το VGG-16 χρησιμοποιήθηκε ως το βασικό δίκτυο είναι λόγω της ισχυρής απόδοσής του σε υψηλής ποιότητας εργασίες κατηγοριοποίησης εικόνων σε κλάσεις και της δημοτικότητάς του για προβλήματα όπου η μεταφορά μάθησης βοηθά στη βελτίωση των αποτελεσμάτων. Αντί για τα αρχικά πλήρως συνδεδεμένα επίπεδα VGG, προστέθηκε ένα σύνολο βοηθητικών συνελικτικών στρωμάτων (από το conv6 και μετά), επιτρέποντας έτσι την εξαγωγή χαρακτηριστικών σε πολλαπλές κλίμακες και μειώνοντας σταδιακά το μέγεθος της εισόδου σε κάθε επόμενο επίπεδο.



Σχήμα 3.19: Αρχιτεκτονική VGG (η είσοδος είναι 224x224x3) [16].

MultiBox

Η τεχνική παλινδρόμησης οριοθέτησης του αλγορίθμου SSD εμπνέεται από την εργασία του Szegedy στο MultiBox, η οποία αποτελεί μια μέθοδο για γρήγορες προτάσεις οριοθετημένων πλαισίων [49]. Για την ανάπτυξη του MultiBox χρησιμοποιείται ένα συνελικτικό δίκτυο τύπου Inception [50]. Οι συνελίξεις 1x1 βοηθούν στη μείωση των διαστάσεων του δικτύου, καθώς ο αριθμός των διαστάσεων θα μειωθεί αλλά το πλάτος και το ύψος θα παραμείνουν τα ίδια.



Σχήμα 3.20: Αρχιτεκτονική πολυεπίπεδης συνελικτικής πρόβλεψης θέσης και εμπιστοσύνης του multibox [16].

Η συνάρτηση απόκλισης του MultiBox συνδυάζει επίσης δύο κρίσιμα στοιχεία που μπήκαν στο SSD:

- **Απόκλιση εμπιστοσύνης**: υπολογίζει πόσο σίγουρο είναι το δίκτυο για την αντικειμενικότητα του υπολογισμένου πλαισίου οριοθέτησης. Η κατηγοριοποιημένη εγκάρσια εντροπία (cross-entropy) χρησιμοποιείται για τον υπολογισμό αυτής της απώλειας.

- **Απόκλιση τοποθεσίας**: υπολογίζει πόσο μακριά είναι τα προβλεπόμενα πλαίσια οριοθέτησης του δικτύου από αυτά που δόθηκαν στο σύνολο δεδομένων της εκπαίδευσης. Σε αυτό το σημείο χρησιμοποιείται κανονικοποίηση L2. Η μαθηματική έκφραση για την απόκλιση, η οποία μετρά πόσο μακριά βρέθηκε από την πρόβλεψη, είναι:

$$multibox_loss \;=\; confidence_loss + alpha * location_loss$$

Ο όρος alpha μας βοηθά στην εξισορρόπηση της συμβολής της απόκλισης τοποθεσίας. Στη βαθιά μάθηση, ο στόχος είναι να βρούμε τις τιμές παραμέτρων που μειώνουν βέλτιστα τη συνάρτηση απώλειας, φέρνοντας έτσι τις προβλέψεις του μοντέλου πιο κοντά στις πραγματικές.

MultiBox Priors

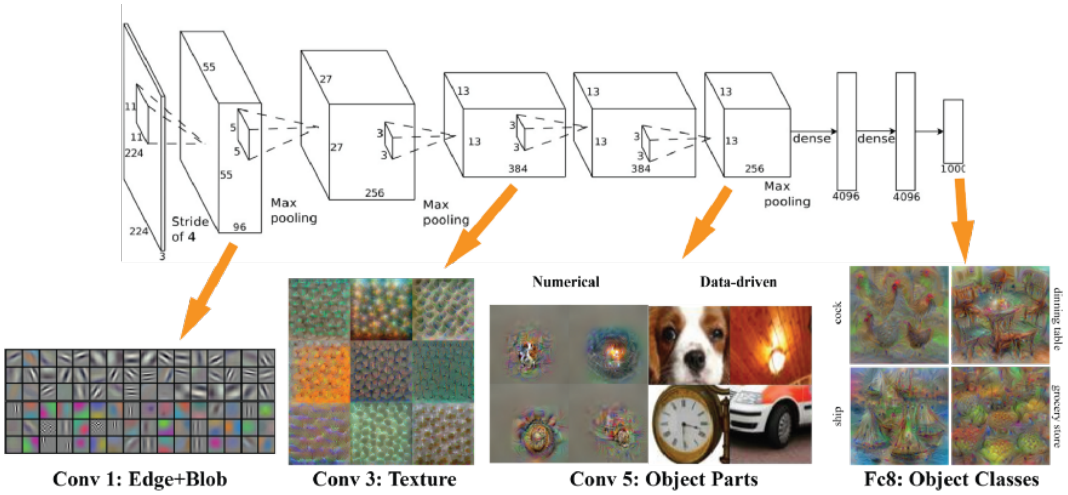
Η λογική που περιστρέφεται γύρω από τη δημιουργία οριοθετημένων πλαισίων είναι στην πραγματικότητα πιο περίπλοκη.

Στο MultiBox, οι ερευνητές δημιούργησαν τα Priors (ή άγκυρες στην ορολογία Faster-R-CNN), τα οποία είναι προ-υπολογισμένα, σταθερού μεγέθους πλαίσια οριοθέτησης που ταιριάζουν στενά με τη διανομή των αρχικών πλαισίων αλήθειας εδάφους (ground-truth boxes) [49]. Στην πραγματικότητα, αυτά τα προηγούμενα επιλέγονται με τέτοιο τρόπο ώστε η αναλογία IoU να είναι μεγαλύτερη από 0,5. Ως εκ τούτου, το MultiBox ξεκινά με τα προηγούμενα ως προβλέψεις και προσπαθεί να φτάσει τις προβλέψεις του πιο κοντά στα πλαίσια οριοθέτησης που ορίστηκαν στο σετ της εκπαίδευσης.

Αυτή η αρχιτεκτονική (σχήμα 3.20) περιέχει 11 priors ανά κελί χαρτών χαρακτηριστικών (8x8, 6x6, 4x4, 3x3, 2x2) και μόνο ένα στο χάρτη χαρακτηριστικών 1x1, συνολικά 1420 priors ανά εικόνα, επιτρέποντας έτσι ισχυρή κάλυψη εικόνων εισόδου σε πολλαπλές κλίμακες, για τον εντοπισμό διαφόρων μεγεθών αντικειμένων. Στο τέλος, το MultiBox διατηρεί μόνο τις κορυφαίες προβλέψεις K που έχουν ελαχιστοποιήσει τόσο τις αποκλίσεις τοποθεσίας όσο και τις αποκλίσεις εμπιστοσύνης.

Χάρτες χαρακτηριστικών

Οι χάρτες χαρακτηριστικών, με άλλα λόγια τα αποτελέσματα των συνελικτικών επιπέδων, είναι μια αναπαράσταση των κυρίαρχων χαρακτηριστικών της εικόνας σε διαφορετικές κλίμακες, επομένως η εκτέλεση του MultiBox σε πολλούς χάρτες χαρακτηριστικών αυξάνει την πιθανότητα οποιουδήποτε αντικειμένου (μεγάλου και μικρού) να ανιχνευτεί, να τοποθετηθούν κατάλληλα τα πλαίσια οριοθέτησης γύρω του και να εντοπιστεί σωστά η κλάση του. Στο σχήμα 3.21 παρουσιάζεται οπτικοποίηση του πως το δίκτυο ”βλέπει” μια εικόνα μεταξύ των διαφόρων χαρτών χαρακτηριστικών του:



Σχήμα 3.21: Οπτικοποίηση του χάρτη χαρακτηριστικών ενός μοντέλου VGG [17].

Ανίχνευση αντικειμένων με SSD

Η διαδικασία της πρόβλεψης είναι απλή. Με την τροφοδοσία μιας εικόνας στο δίκτυο, κάθε prior θα έχει ένα σύνολο πλαισίων και ετικετών οριοθέτησης. Ωστόσο έχουμε πολλές προβλέψεις priors για το ίδιο αντικείμενο. Για να αφαιρεθούν τα διπλότυπα priors πάνω από κάθε αντικείμενο που ανιχνεύθηκε, χρησιμοποιείται η μη μέγιστη καταστολή (Non-Maximum Suppression - NMS).

Μη μέγιστη καταστολή

Το NMS διατηρεί μόνο τα κουτιά οριοθέτησης με τις μεγαλύτερες πιθανότητες και μεγαλύτερα IoU και αφαιρεί τα κουτιά οριοθέτησης που έχουν χαμηλότερες πιθανότητες σε σχέση με τα διατηρημένα.

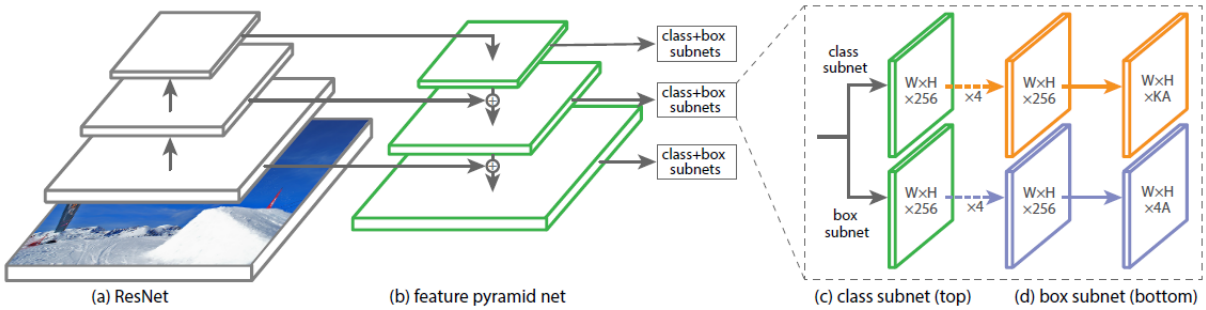
Μειονεκτήματα του SSD

Τα ρηχά επίπεδα στο νευρωνικό δίκτυο του SSD ενδέχεται να μη δημιουργούν αρκετά χαρακτηριστικά υψηλού επιπέδου για να κάνουν προβλέψεις για μικρά αντικείμενα. Επομένως, το SSD δεν κάνει καλές προβλέψεις για μικρότερα αντικείμενα σε σχέση με τα μεγαλύτερα αντικείμενα.

3.7 RetinaNet

Το RetinaNet είναι ένα από τα καλύτερα μοντέλα ανίχνευσης αντικειμένων ενός σταδίου που έχει αποδειχθεί ότι λειτουργεί καλά με πυκνά και μικρά αντικείμενα [18]. Το RetinaNet έχει διαμορφωθεί κάνοντας δύο βελτιώσεις σε σχέση με τα υπάρ-

χοντα μοντέλα ανίχνευσης αντικειμένων ενός σταδίου - Feature Pyramid Networks (FPN) [51] και Focal Loss [18].



Σχήμα 3.22: Η αρχιτεκτονική του RetinaNet [18].

Αρχιτεκτονική RetinaNet

Υπάρχουν τέσσερα κύρια συστατικά της αρχιτεκτονικής του μοντέλου RetinaNet (Σχήμα 3.22):

1. **Διαδρομή από κάτω προς τα πάνω**: Το backbone δίκτυο (π.χ. ResNet) που υπολογίζει τους χάρτες χαρακτηριστικών σε διαφορετικές κλίμακες, είναι ανεξάρτητο από το μέγεθος της εικόνας εισαγωγής ή το backbone δίκτυο.

2. **Διαδρομές από πάνω προς τα κάτω και πλευρικές συνδέσεις**: Η διαδρομή από πάνω προς τα κάτω δίνει δείγματα στους χονδρικούς χάρτες χαρακτηριστικών χωρικών από υψηλότερα επίπεδα πυραμίδας και οι πλευρικές συνδέσεις συγχωνεύουν τα επίπεδα από πάνω προς τα κάτω και από τα κάτω προς τα πάνω επίπεδα με το ίδιο χωρικό μέγεθος.

3. **Υποδίκτυο ταξινόμησης**: Προβλέπει την πιθανότητα παρουσίας ενός αντικειμένου σε κάθε χωρική θέση για κάθε πλαίσιο αγκύρωσης και κλάση αντικειμένων.

4. **Υποδίκτυο παλινδρόμησης**: Παρουσιάζει την αντιστάθμιση για τα πλαίσια οριοθέτησης από τα πλαίσια αγκύρωσης για κάθε αντικείμενο αλήθειας εδάφους.

Backbone

Το RetinaNet υιοθετεί το Feature Pyramid Network (FPN) [19] που προτάθηκε από τους Lin, Dollar, et al. (2017) ως backbone του δικτύου, το οποίο με τη σειρά

του είναι χτισμένο πάνω από το ResNet (ResNet-50, ResNet-101 ή ResNet-152) με απόλυτα συνελικτικό τρόπο. Η πλήρως συνελικτική φύση επιτρέπει στο δίκτυο να λαμβάνει μια εικόνα ενός αυθαίρετου μεγέθους και εξάγει χάρτες χαρακτηριστικών ανάλογου μεγέθους σε πολλά επίπεδα στην πυραμίδα χαρακτηριστικών. Η κατασκευή του FPN περιλαμβάνει δύο διαδρομές που συνδέονται με πλευρικές συνδέσεις.

Υποδίκτυο ταξινόμησης

Το υποδίκτυο ταξινόμησης είναι ένα πλήρως συνελικτικό δίκτυο (FCN) συνδεδεμένο σε κάθε επίπεδο FPN. Το υποδίκτυο αποτελείται από τέσσερα 3 × 3 συνελικτικά επίπεδα με 256 φίλτρα, ακολουθούμενα από ενεργοποιήσεις RELU. Στη συνέχεια, υπάρχει ένα άλλο συνελικτικό στρώμα 3 × 3 με φίλτρα K × A, ακολουθούμενο από σιγμοειδή ενεργοποίηση (αντί για ενεργοποίηση softmax). Το υποδίκτυο έχει κοινόχρηστες παραμέτρους σε όλα τα επίπεδα. Οι διαστάσεις του χάρτη χαρακτηριστικών εξόδου είναι της μορφής (W, H, KA), όπου τα W και H είναι ανάλογα με το πλάτος και το ύψος του χάρτη χαρακτηριστικών εισόδου, K και A είναι οι αριθμοί της κλάσης αντικειμένων και των αγκυρών.

Υποδίκτυο παλινδρόμησης

Το υποδίκτυο παλινδρόμησης βρίσκεται σε κάθε χάρτη δυνατοτήτων του FPN παράλληλα με το υποδίκτυο ταξινόμησης. Ο σχεδιασμός του υποδικτύου παλινδρόμησης είναι πανομοιότυπος με αυτόν του υποδικτύου ταξινόμησης, εκτός από το ότι το τελευταίο συνελικτικό επίπεδο είναι 3 × 3 με φίλτρα 4A. Επομένως, οι διαστάσεις του χάρτη χαρακτηριστικών εξόδου θα είναι (W, H, 4A).

Εστιακή απόκλιση

Η εστιακή απόκλιση (Focal Loss - FL) είναι μια βελτίωση έναντι της Cross-Entropy Loss (CE) και εισάγεται για να χειριστεί το πρόβλημα ανισορροπίας της κλάσης με μοντέλα ανίχνευσης αντικειμένων ενός σταδίου. Τα μοντέλα ενός σταδίου υποφέρουν από ένα ακραίο πρόβλημα ανισορροπίας κλάσης προσκηνίου-φόντου λόγω της πυκνής δειγματοληψίας αγκυρών (πιθανές θέσεις αντικειμένων) [52]. Στο RetinaNet, σε κάθε στρώμα πυραμίδας μπορεί να υπάρχουν χιλιάδες κουτιά αγκύ-

ρωσης. Μόνο λίγα θα εκχωρηθούν σε ένα αντικείμενο αλήθειας εδάφους, ενώ η συντριπτική πλειοψηφία θα ανήκει στην κλάση του αντικειμένου. Αυτά τα εύκολα παραδείγματα (ανιχνεύσεις με μεγάλες πιθανότητες) αν και οδηγούν σε μικρές τιμές απώλειας μπορούν συλλογικά να κατακλύσουν το μοντέλο. Η εστιακή απόκλιση μειώνει τη συμβολή απώλειας από εύκολα παραδείγματα και αυξάνει τη σημασία της διόρθωσης παραδειγμάτων που δεν έχουν κατηγοριοποιηθεί σωστά.

Συνάρτηση απόκλισης

Η απόκλιση του RetinaNet είναι μια απόκλιση πολλαπλών εργασιών που περιέχει δύο όρους: έναν για εντοπισμό (δηλώνεται ως L_{loc}) και έναν για κατηγοριοποίηση (με την ένδειξη L_{cls}). Η συνολική απόκλιση των 2 εργασιών μπορεί να γραφτεί ως:

$$L = \lambda L_{loc} + L_{cls} \tag{3.1}$$

όπου λ είναι μια υπέρ-παράμετρος που ελέγχει την ισορροπία μεταξύ των δύο αποκλίσεων.

Απόκλιση κατηγοριοποίησης

Η απόκλιση κατηγοριοποίησης που υιοθετήθηκε από το RetinaNet είναι μια παραλλαγή της εστιακής απώλειας, η οποία είναι η πιο καινοτόμος σχεδίαση του ανιχνευτή. Η απόκλιση για κάθε άγκυρα μπορεί να οριστεί σύμφωνα με τον τύπο 3.2.

$$L_{cls} = -\sum_{i=1}^{K}(y_i log(p_i)(1-p_i)^{\gamma}\alpha_i + (1-y_i)log(1-p_i)p_i^{\gamma}(1-\alpha_i)) \tag{3.2}$$

όπου δηλώνει τον αριθμό των κλάσεων, το y_i ισούται με 1 εάν η αλήθεια εδάφους ανήκει στην i κλάση, διαφορετικά το y_i ισούται με 0 . Το p_i είναι η προβλεπόμενη πιθανότητα για την i κλάση. Το $\gamma \in (0, +\infty)$ είναι μια παράμετρος εστίασης. Το $\alpha_i \in [0, 1]$ είναι μια παράμετρος στάθμισης για την i κλάση. Η απόκλιση είναι παρόμοια με την κατηγορική εγκάρσια εντροπία και είναι ισοδύναμη εάν $\gamma = 0$ και $\alpha_i = 1$.

Απόκλιση παλινδρόμησης

Η απόκλιση παλινδρόμησης υπολογίζεται με βάση κατά πόσο ένα πλαίσιο αγκύρωσης συνδυάζεται με ένα πλαίσιο αλήθειας εδάφους. Ας υποδείξουμε αυτά τα

ζευγάρια που ταιριάζουν ως $(A_i, G_i), i = 1, \cdots, N$, όπου το A αντιπροσωπεύει μια άγκυρα, το G αντιπροσωπεύει μια αλήθεια εδάφους και N είναι ο αριθμός των πλαισίων που ταιριάζουν.

Για κάθε άγκυρα που ταιριάζει, το υποδίκτυο παλινδρόμησης προβλέπει τέσσερις αριθμούς, τους οποίους δηλώνουμε ως $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$. Οι δύο πρώτοι αριθμοί καθορίζουν τη μετατόπιση μεταξύ των κέντρων αγκύρωσης A_i και εδάφους-αλήθειας G_i, ενώ οι δύο τελευταίοι αριθμοί καθορίζουν τη μετατόπιση μεταξύ του πλάτους/ύψους της άγκυρας και της αλήθειας εδάφους. Αντίστοιχα, για καθεμία από αυτές τις προβλέψεις, υπάρχει ένας στόχος παλινδρόμησης T_i που υπολογίζεται ως η αντιστάθμιση μεταξύ της άγκυρας και της αλήθειας εδάφους:

$$\begin{aligned} T^i_x &= (G^i_x - A^i_x)/A^i_w \\ T^i_y &= (G^i_y - A^i_y)/A^i_h \\ T^i_w &= log(G^i_w/A^i_w) \\ T^i_h &= log(G^i_h/A^i_h) \end{aligned} \tag{3.3}$$

Με τις παραπάνω συμβολές, η απόκλιση παλινδρόμησης μπορεί να οριστεί σύμφωνα με τον τύπο 3.4.

$$L_{loc} = \sum_{j \in \{x,y,w,h\}} smooth_{L1}(P^i_j - T^i_j) \tag{3.4}$$

όπου $smooth_{L1}(x)$ είναι η ομαλή απόκλιση L1 που μπορεί να οριστεί ως:

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & |x| \geq 1 \end{cases} \tag{3.5}$$

Ανίχνευση αντικειμένων με RetinaNet

Τέλος, για να δημιουργήσει προβλέψεις το RetinaNet μόλις εκπαιδευτεί ακολουθεί την παρακάτω διαδικασία. Για κάθε εικόνα εισόδου: υπάρχουν $\sum_{l=3}^{7} W_l \times H_l \times A$ πλαίσια αγκύρωσης από όλα τα επίπεδα FPN. Για κάθε πλαίσιο αγκύρωσης, το υποδίκτυο ταξινόμησης προβλέπει τους αριθμούς που υποδηλώνουν την πιθανότητα κατανομής των κατηγοριών αντικειμένων, ενώ το υποδίκτυο παλινδρόμησης προβλέπει 4 αριθμούς που δείχνουν την μετατόπιση μεταξύ κάθε αγκύρωσης και

του αντίστοιχου πλαισίου οριοθέτησης.

Για λόγους απόδοσης, το RetinaNet επιλέγει το πολύ 1000 πλαίσια αγκύρωσης που έχουν την υψηλότερη βαθμολογία εμπιστοσύνης (δηλαδή προβλεπόμενη πιθανότητα για κάθε κλάση αντικειμένου) από κάθε επίπεδο FPN, και έχουν βαθμολογία μεγαλύτερη από 0,05. Μόνο αυτές οι άγκυρες θα συμπεριληφθούν στα ακόλουθα βήματα.

Σε αυτό το στάδιο, ένα αντικείμενο στην εικόνα μπορεί να προβλεφθεί από πολλά πλαίσια αγκύρωσης. Για να αφαιρεθεί αυτός ο πλεονασμός πλαισίων, εφαρμόζεται η μη μέγιστη καταστολή (NMS) σε κάθε τάξη ανεξάρτητα, η οποία επιλέγει επαναληπτικά ένα πλαίσιο αγκύρωσης με την υψηλότερη βαθμολογία και αφαιρεί τυχόν επικαλυπτόμενα πλαίσια αγκύρωσης με IoU μεγαλύτερο από 0,5.

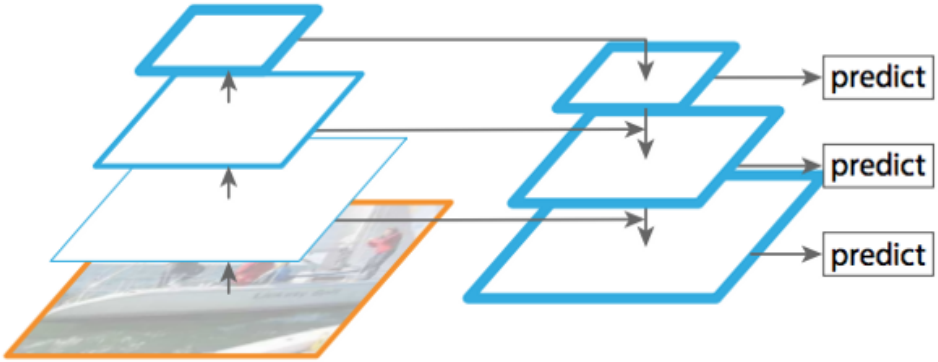
Στο τελευταίο στάδιο, για κάθε πλαίσιο αγκύρωσης που έχει απομείνει, το υποδίκτυο παλινδρόμησης δίνει προβλέψεις αντιστάθμισης που μπορούν να χρησιμοποιηθούν για να βελτιώσουν την αγκύρωση, έτσι ώστε να έχει και πρόβλεψη του πλαισίου οριοθέτησης του αντικειμένου που περικλείει.

3.8 EfficientDet

Πρόκειται για μια σειρά 8 αλγορίθμων από την Google οι οποίοι είναι πολύ πιο αποδοτικοί, ακριβείς και πολύ πιο γρήγοροι σε σχέση με τους προϋπάρχοντες ανιχνευτές [20]. Σε διαφορετικές συσκευές, μπορεί να αποδώσει καλά αποτελέσματα και είναι πάντα συγκρίσιμο σε ένα ευρύ φάσμα περιορισμών πόρων. Ειδικά στην περίπτωση ενός μοντέλου και μιας κλίμακας, το EfficientDet-D7 έχει φτάσει στα 52,2% mAP στο σύνολο δοκιμής COCO, με 52M παραμέτρους και 325B FLOPs, σε σύγκριση με τον προηγούμενο αλγόριθμο, η ποσότητα των παραμέτρων μειώνεται κατά 4 έως 9 φορές, τα FLOPs έχουν μειωθεί κατά 13 έως 42 φορές. Για την ανάπτυξη αυτού του ανιχνευτή αντικειμένων η Google, πρώτον, πρότεινε ένα σταθμισμένο δίκτυο διπλής κατεύθυνσης πυραμίδας χαρακτηριστικών (BiFPN), το οποίο επιτρέπει την απλή και γρήγορη συγχώνευση πολλαπλών κλιμάκων και δεύτερον, μια μέθοδο κλιμάκωσης σύνθετων χαρακτηριστικών πυραμίδας δικτύου, ομοιόμορφη κλίμακα της ανάλυσης, του βάθους και του πλάτους, του δικτύου δυνατοτήτων και του δικτύου προβλέψεων box/class όλων των δικτύων backbone.

Δίκτυα πυραμίδων χαρακτηριστικών (FPN)

Προτού μελετήσουμε το BiFPN, είναι σημαντικό να επανεξετάσουμε την ιδέα του FPN και να μελετήσουμε τα πλεονεκτήματα και τα μειονεκτήματα αυτής της προσέγγισης. Παρόλο που το FPN δεν είναι καινούριο, η ιδέα της χρήσης εγγενών πολυεπίπεδων ιεραρχικών πυραμίδων χαρτών χαρακτηριστικών σε ένα βαθύ CNN παρουσιάστηκε για πρώτη φορά το 2017 στο άρθρο Feature Pyramid Networks for Object Detection [19].



Σχήμα 3.23: Αρχιτεκτονική δικτύου πυραμιδικών χαρακτηριστικών (FPN) [19]

.

Ένα από τα δυσκολότερα προβλήματα, όσον αφορά την ανίχνευση αντικειμένων, είναι η ανίχνευση αντικειμένων σε διαφορετικές κλίμακες. Παλαιότερα, οι ερευνητές συνήθιζαν να συνδυάζουν χαρακτηριστικά που βασίζονται σε πυραμίδες εικόνας (Featurized Image Pyramids) για την αντιμετώπιση αυτού του προβλήματος. Αυτές οι πυραμίδες είναι αναλλοίωτες υπό την έννοια ότι η αλλαγή κλίμακας ενός αντικειμένου αντισταθμίζεται μετατοπίζοντας το επίπεδό του στην πυραμίδα. Διαισθητικά, αυτή η ιδιότητα επιτρέπει σε ένα μοντέλο να ανιχνεύει αντικείμενα σε ένα μεγάλο εύρος κλιμάκων σαρώνοντας το μοντέλο και στις δύο θέσεις και στα επίπεδα της πυραμίδας. Αν και αυτό λειτουργεί καλά, είναι πολύ ακριβό όσον αφορά τον υπολογισμό και μπορεί να χρησιμοποιηθεί μόνο κατά τη διάρκεια της δοκιμής, δημιουργώντας αναντιστοιχία μεταξύ των διαδικασιών εκπαίδευσης και δοκιμών.

Τα CNN, από την άλλη πλευρά, εκτός από το ότι είναι ικανά να αντιπροσωπεύουν σημασιολογία υψηλότερου επιπέδου, είναι επίσης πιο ανθεκτικά στη διακύμανση στην κλίμακα και έτσι διευκολύνουν την αναγνώριση από χαρακτηριστικά που υπολογίζονται σε μία κλίμακα εισόδου. Αν και τα CNN είναι ισχυρά και πα-

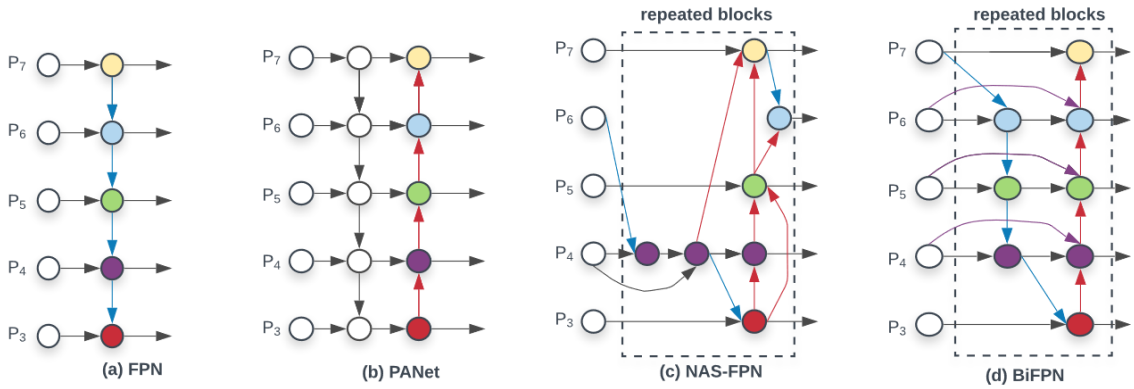
ρέχουν υψηλές σημασιολογικές δυνατότητες, οι ανιχνεύσεις από μία μόνο κλίμακα εισόδου δεν είναι αρκετά καλές. Ένα από τα πιο σημαντικά πράγματα για τα CNN είναι ότι διαφορετικά επίπεδα υπολογίζουν χαρακτηριστικά χαρτών σε διαφορετικά επίπεδα και διαφορετικές κλίμακες. Αυτή η ιεραρχία χαρακτηριστικών εντός δικτύου έχει εγγενή πολυκλίμακα, πυραμιδική μορφή. Στο σχήμα 3.23 παρουσιάζεται η αρχιτεκτονική του FPN σε υψηλό επίπεδο.

Αν και έχουμε την πυραμίδα χαρακτηριστικών, όλοι οι χάρτες χαρακτηριστικών αυτής της πυραμίδας είναι σε διαφορετικές κλίμακες και έχουν τεράστια σημασιολογικά κενά που προκαλούνται από διαφορετικά βάθη στρωμάτων στο δίκτυο. Οι χάρτες υψηλής ανάλυσης έχουν χαρακτηριστικά χαμηλού επιπέδου που βλάπτουν την ικανότητα αναπαραγωγής τους για αναγνώριση αντικειμένων.

Ωστόσο, οι χάρτες υψηλής ανάλυσης σε ένα CNN έχουν χαμηλές σημασιολογικές δυνατότητες, ενώ οι χάρτες χαμηλής ανάλυσης έχουν υψηλές σημασιολογικές δυνατότητες και χρειαζόμαστε και τα δύο για να κάνουμε μια ισχυρή ανίχνευση σε διαφορετικές κλίμακες.

Τα FPN επιτυγχάνουν αυτό συνδυάζοντας τη διαδρομή από κάτω προς τα πάνω με τη διαδρομή από πάνω προς τα κάτω χρησιμοποιώντας πλευρικές συνδέσεις. Εν ολίγοις, στις λειτουργίες υψηλού επιπέδου λαμβάνονται πρώτα τα δείγματα και στη συνέχεια συνδυάζονται με τα χαρακτηριστικά χαμηλού επιπέδου χρησιμοποιώντας μια πλευρική σύνδεση που βασικά είναι μια συνέλιξη 1x1 ακολουθούμενη από άθροισμα.

BiFPN



Σχήμα 3.24: (a) Το FPN εισάγει ένα μονοπάτι από πάνω προς τα κάτω για την ασφάλεια λειτουργιών πολλαπλών κλιμάκων. (b) Το PANet προσθέτει ένα επιπλέον μονοπάτι από πάνω προς τα πάνω στην κορυφή του FPN. (c) NAS-FPN χρησιμοποιεί αναζήτηση αρχιτεκτονικής με χρήση νευρωνικών δικτύων για να βρει μια τοπολογία δικτύου χαρακτηριστικών και στη συνέχεια να εφαρμοστεί επανειλημμένα το ίδιο μπλοκ. (d) Το BiFPN, με καλύτερη αντιστάθμιση και αποτελεσματικότητα [20]

Το συμβατικό FPN συγκεντρώνει χαρακτηριστικά πολλαπλών κλιμάκων με τρόπο από πάνω προς τα κάτω:

$$P_7^{out} = Conv\left(P_7^{in}\right)$$

$$P_6^{out} = Conv\left(P_7^{in} + Resize\left(P_7^{out}\right)\right)$$

$$\ldots$$

$$P_3^{out} = Conv\left(P_3^{in} + Resize\left(P_6^{out}\right)\right)$$

Το πρόβλημα με το συμβατικό FPN όπως φαίνεται στο Σχήμα 3.24 (a) είναι ότι περιορίζεται από τη μονόδρομη (πάνω-κάτω) ροή πληροφοριών. Για να αντιμετωπίσει αυτό το ζήτημα, το PANet προσθέτει ένα επιπλέον δίκτυο συγκέντρωσης διαδρομής από κάτω προς τα πάνω, όπως φαίνεται στο Σχήμα 3.24 (b). Επίσης, υπάρχουν πολλές δημοσιεύσεις, όπως το NAS-FPN, που μελετάνε τις διασυνδέσεις για τη λήψη καλύτερης σημασιολογίας. Εν ολίγοις, το νόημα αφορά τη σύνδεση λειτουργιών χαμηλού επιπέδου με λειτουργίες υψηλού επιπέδου και το αντίστροφο για τη λήψη καλύτερης σημασιολογίας.

Το πρόβλημα με το NAS είναι ότι βασίζεται στην ενισχυτική μάθηση που απαιτεί χιλιάδες GPU ή/και TPU ώρες για να βρει τις καλύτερες συνδέσεις. Επίσης, διαπιστώθηκε ότι οι τελικές διασταυρωμένες συνδέσεις που βρέθηκαν από τη NAS, όπως φαίνεται στο 3.24 (c), ήταν ακανόνιστες και δύσκολο να ερμηνευθούν. Ακόμα και μετά από αυτό, το PANet κέρδισε τόσο το FPN όσο και το NAS-FPN όσον αφορά την ακρίβεια, αλλά έχει επιπλέον κόστος υπολογισμού. Έτσι, παίρνουμε το PANet ως

βάση και βελτιώνουμε τις συνδέσεις τόσο για αποδοτικότητα όσο και για ακρίβεια. Για να επιτευχθούν βελτιστοποιημένες διασυνδέσεις, οι συγγραφείς πρότειναν τα εξής:

- Αφαίρεση των κόμβων που έχουν μόνο ένα άκρο εισόδου. Εάν ένας κόμβος έχει μόνο ένα άκρο εισόδου χωρίς συγχώνευση χαρακτηριστικών, τότε θα έχει λιγότερη συνεισφορά στο δίκτυο δυνατοτήτων που στοχεύει στη συγχώνευση διαφορετικών λειτουργιών. Αυτό οδηγεί σε έναν απλοποιημένο PANET όπως φαίνεται στο σχήμα 3.24 (e).

- Πρόσθεση ενός επιπλέον άκρου από την αρχική είσοδο στον κόμβο εξόδου, εάν βρίσκονται στο ίδιο επίπεδο, προκειμένου να συγχωνεύσει περισσότερες δυνατότητες χωρίς να προσθέσει πολύ κόστος, όπως φαίνεται στο σχήμα 3.24 (f).

- Σε αντίθεση με το PANET που έχει μόνο μία διαδρομή από πάνω προς τα κάτω και μία από κάτω προς τα πάνω, οι συγγραφείς αντιμετωπίζουν κάθε διαδρομή διπλής κατεύθυνσης (από πάνω προς τα κάτω και κάτω προς τα πάνω) ως ένα επίπεδο δικτύου χαρακτηριστικών και επαναλαμβάνουν το ίδιο επίπεδο πολλές φορές για να επιτρέψουν περισσότερα υψηλά επίπεδα χαρακτηριστικών συγχώνευσης. Το αποτέλεσμα αυτών των βελτιστοποιήσεων είναι το νέο δίκτυο χαρακτηριστικών που ονομάζεται BiFPN όπως φαίνεται στο σχήμα 3.24 (f).

Σταθμισμένο χαρακτηριστικό συγχώνευσης

Οι προηγούμενες εργασίες επανακατασκευάζουν τους χάρτες χαμηλής ανάλυσης και, στη συνέχεια, τους συνοψίζουν με διαφορετικές δυνατότητες εισαγωγής που τους αντιμετωπίζουν εξίσου. Διαισθητικά, εάν τα χαρακτηριστικά βρίσκονται σε διαφορετική ανάλυση, η συνεισφορά αυτών των χαρακτηριστικών στα τελικά χαρακτηριστικά εξόδου θα ήταν γενικά άνιση. Ως εκ τούτου, απαιτείται κάποια στρατηγική στάθμισης για την εκχώρηση διαφορετικού βάρους σε διαφορετικούς χάρτες χαρακτηριστικών. Έτσι οι συγγραφείς του EfficientDet εισήγαγαν τρεις διαφορετικές σταθμισμένες στρατηγικές συγχώνευσης:

- Συγχώνευση χωρίς περιορισμούς (Unbounded Fusion)

$$O = w_i \ * \ I_i \tag{3.6}$$

Εδώ είναι ένα μαθησιακό βάρος που μπορεί να είναι κλιμακωτό (ανά χαρακτηριστικό), διάνυσμα (ανά κανάλι) ή πολυδιάστατο τανυστή (ανά pixel). Οι συγγραφείς ανακάλυψαν ότι μια κλίμακα μπορεί να επιτύχει συγκρίσιμη ακρίβεια με άλλες προσεγγίσεις με ελάχιστο υπολογιστικό κόστος. Ωστόσο, δεδομένου ότι το βαθμωτό βάρος δεν είναι περιορισμένο, θα μπορούσε ενδεχομένως να προκαλέσει αστάθεια στην εκπαίδευση.

- Συγχώνευση με βάση τη Softmax (Softmax-based Fusion)

$$O = \Sigma_i \frac{e^{w_i}}{\Sigma_j e^{w_j}} \ * \ I_i \tag{3.7}$$

Εάν θέλουμε να περιοριστούν οι τιμές χωρίς περιορισμό στο εύρος 0-1, ένας από τους καλύτερους τρόπους είναι να εφαρμοστεί η συνάρτηση softmax και να μετατραπούν οι τιμές σε κατανομή πιθανότητας όπου οι τιμές αντιπροσωπεύουν τη σημασία των βαρών. Το μόνο μειονέκτημα αυτής της προσέγγισης είναι ότι η συνάρτηση softmax είναι υπολογιστικά ακριβή και αυξάνει την καθυστέρηση του δικτύου.

- Γρήγορη κανονικοποιημένη συγχώνευση (Fast Normalized Fusion)

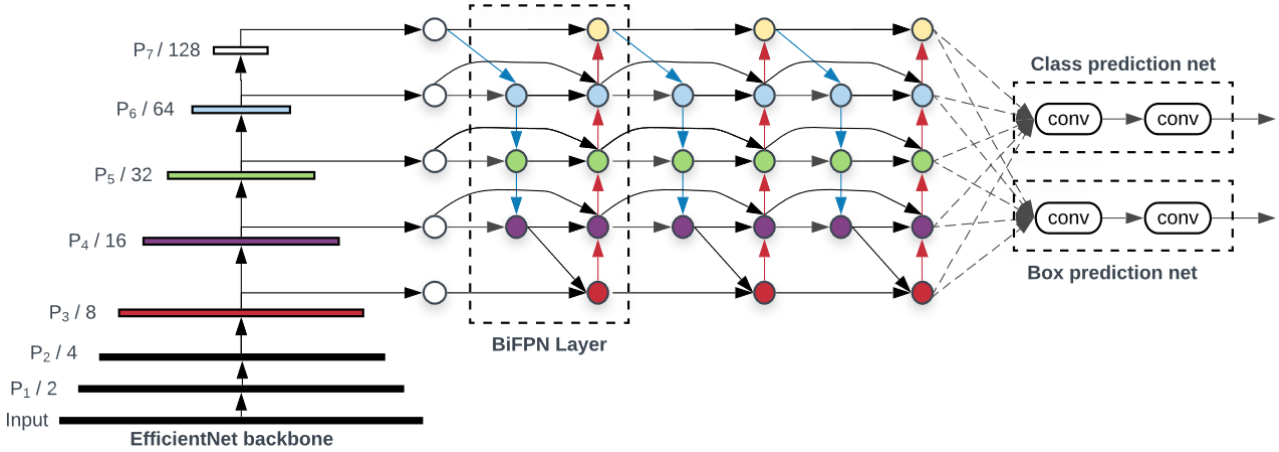
$$O = \Sigma_i \frac{w_i}{\varepsilon + \Sigma_j e^{w_j}} \ * \ I_i \tag{3.8}$$

Θέλουμε τα βάρη μας να κυμαίνονται από 0 έως 1. Ένας τρόπος για να βεβαιωθούμε ότι $w_i \geq 0$ είναι να εφαρμοστεί η συνάρτηση ReLU. Τώρα, μπορούμε να ομαλοποιήσουμε τις τιμές, έτσι ώστε το ανώτερο όριο να είναι 1. Αυτή η μέθοδος είναι εξαιρετικά απλή και πολύ αποτελεσματική σε σύγκριση με την προσέγγιση που βασίζεται στη συνάρτηση softmax.

Όπως υποδηλώνει το όνομα, χρησιμοποιώντας το EfficientNets ως δίκτυο backbone μαζί με BiFPNs, έχουμε μια νέα οικογένεια ανιχνευτών που ονομάζεται EfficientDet.

Αρχιτεκτονική

Οι ανιχνευτές EfficientDet είναι ανιχνευτές μίας λήψης όπως το SSD και το RetinaNet. Το backbone δικτύου του EfficientDet είναι το EfficientNet προ-εκπαιδευμένο στο σύνολο δεδομένων ImageNet. Το προτεινόμενο BiFPN χρησιμεύει ως δίκτυο δυνατοτήτων, το οποίο λαμβάνει τις δυνατότητες επιπέδου 3-7 P3, P4, P5, P6, P7 από το backbone δίκτυο και εφαρμόζει επανειλημμένα συγχωνεύσεις διπλής κατεύθυνσης από πάνω προς τα κάτω και από κάτω προς τα πάνω. Αυτές οι συγχωνευμένες δυνατότητες τροφοδοτούνται σε ένα δίκτυο κλάσης και πλαισίου για την παραγωγή προβλέψεων κατηγορίας αντικειμένων και οριοθέτησης πλαισίου αντίστοιχα. Τα βάρη δικτύου τάξης και πλαισίου μοιράζονται σε όλα τα επίπεδα λειτουργιών.



Σχήμα 3.25: Αρχιτεκτονική EfficientDet - Χρησιμοποιεί το EfficientNet [21] ως δίκτυο backbone, BiFPN ως δίκτυο δυνατοτήτων και κοινόχρηστο δίκτυο προβλέψεων τάξης / πλαισίου. Τόσο τα επίπεδα BiFPN όσο και τα επίπεδα καθαρού κλάσης / πλαισίου επαναλαμβάνονται πολλές φορές με βάση διαφορετικούς περιορισμούς πόρων [20]

.

Σύνθετη κλιμάκωση

Οι συγγραφείς πρότειναν μια νέα μέθοδο σύνθετης κλιμάκωσης για την ανίχνευση αντικειμένων, ο οποίος χρησιμοποιεί έναν απλό συντελεστή σύνθεσης ϕ για την από κοινού αναβάθμιση όλων των διαστάσεων του δικτύου backbone, του δικτύου BiFPN, του δικτύου κλάσης/πλαισίου και της ανάλυσης. Επιπλέον, οι συγγραφείς χρησιμοποίησαν μια ευρετική προσέγγιση κλιμάκωσης, αλλά εξακολουθούν να ακολουθούν την κύρια ιδέα της από κοινού αύξησης όλων των διαστάσεων.

- **Backbone δίκτυο**: Χρησιμοποιούνται οι ίδιοι συντελεστές κλιμάκωσης πλά-

τους / βάθους EfficientNet-B0 έως B6, έτσι ώστε να μπορούν να χρησιμοποιηθούν προκαθορισμένα σημεία ελέγχου ImageNet.

- **Δίκτυο BiFPN**: Οι συγγραφείς αυξάνουν εκθετικά το πλάτος BiFPN (αριθμός καναλιών) όπως γίνεται στο EfficientNets, αλλά αυξάνουν γραμμικά το βάθος (αριθμός επιπέδων) καθώς το βάθος πρέπει να στρογγυλοποιείται σε μικρούς ακέραιους αριθμούς.

$$W_{bifpn} = 64 * (1.35^{\varphi}), \; D_{bifpn} = 2 + \varphi \tag{3.9}$$

- **Δίκτυο πρόβλεψης πλαισίου / κλάσης**: Το πλάτος διατηρείται ίδιο με το BiFPN αλλά το βάθος (αριθμός επιπέδων) αυξάνεται γραμμικά.

$$D_{box} = D_{class} = 3 + [\varphi/3] \tag{3.10}$$

- **Ανάλυση εικόνας εισόδου**: Δεδομένου ότι το επίπεδο χαρακτηριστικών 3-7 χρησιμοποιείται στο BiFPN, η ανάλυση εισόδου πρέπει να διαιρεθεί με $2^7 = 128$, οπότε αυξάνουμε γραμμικά τις αναλύσεις χρησιμοποιώντας εξίσωση:

$$R_{input} = 512 + \varphi * 128 \tag{3.11}$$

Τέλος, χρησιμοποιώντας τις εξισώσεις (3.9), (3.10) και (3.11), και διαφορετικές τιμές του ϕ, μπορούμε να πάμε από Efficient-D0 (ϕ = 0) στο Efficient-D6 (ϕ = 6). Μοντέλα κλιμακωτά με ϕ ≥ 7 δε μπόρεσαν να χωρέσουν στη μνήμη, εκτός εάν αλλάξει το μέγεθος παρτίδας ή άλλες ρυθμίσεις. Επομένως, οι συγγραφείς επέκτειναν το D6 σε D7 διευρύνοντας μόνο το μέγεθος εισόδου διατηρώντας ταυτόχρονα όλες τις άλλες διαστάσεις ίδιες, έτσι ώστε να μπορούμε να χρησιμοποιήσουμε τις ίδιες ρυθμίσεις εκπαίδευσης για όλα τα μοντέλα. Ακολουθεί ένας πίνακας που συνοψίζει όλες αυτές τις ρυθμίσεις:

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

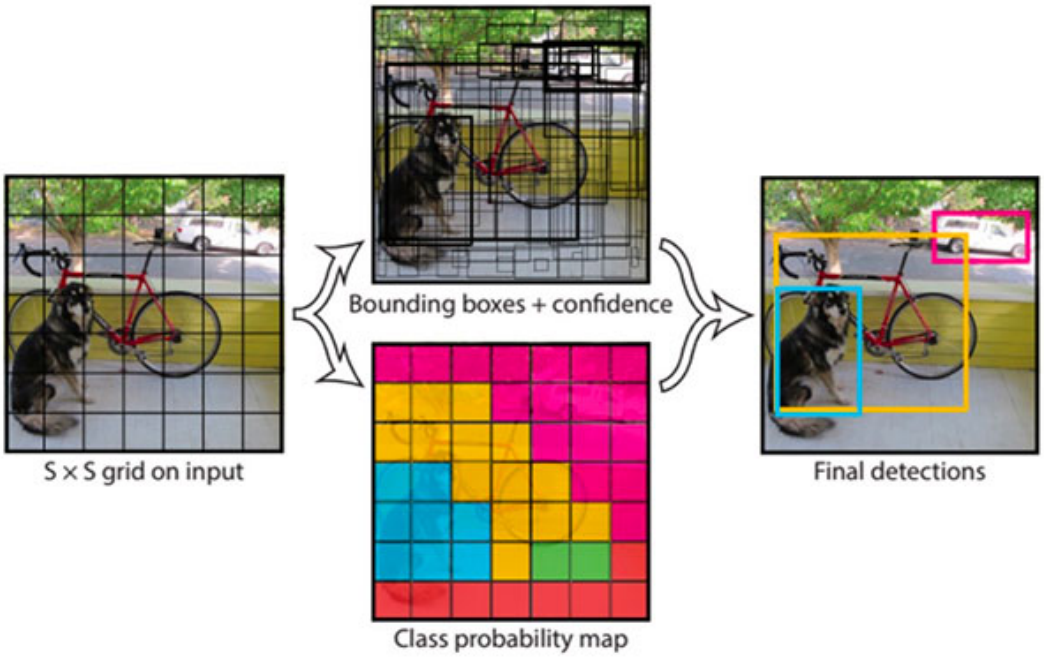
Σχήμα 3.26: Κλιμάκωση ρυθμίσεων για EfficientDet D0-D6. Ο φ είναι ο σύνθετος συντελεστής που ελέγχει όλες τις άλλες διαβαθμίσεις κλίμακας. Τα BiFPN, το box/class net, και το input size κλιμακώνονται σύμφωνα με τις εξισώσεις 3.9, 3.10, 3.11, αντίστοιχα [20]

.

Το δίκτυο EfficientDet εμπνέεται σε μεγάλο βαθμό από τη δουλειά που έγινε στα μοντέλα EfficientNet. Χρησιμοποιώντας σύνθετη κλιμάκωση, η απόδοση του μοντέλου από την άποψη της ακρίβειας και της απόδοσης έχει βελτιωθεί σε σύγκριση με άλλα σύγχρονα μοντέλα ανίχνευσης αντικειμένων. Η οικογένεια μοντέλων EfficientDet έχει αποδειχθεί ότι έχει σημαντική επιτάχυνση σε GPU και CPU που είναι κρίσιμη για εφαρμογές που απαιτούν χαμηλό λανθάνοντα χρόνο.

3.9 YOLOv3

Όλοι οι προηγούμενοι αλγόριθμοι ανίχνευσης αντικειμένων χρησιμοποιούν περιοχές για να εντοπίσουν το αντικείμενο μέσα στην εικόνα. Το δίκτυο δε βλέπει την πλήρη εικόνα πάρα μόνο μερικά τμήματα της εικόνας που έχουν μεγάλες πιθανότητες να περιέχουν αντικείμενο. Το YOLO ή You Only Look Once [53] [54] είναι ένας αλγόριθμος ανίχνευσης αντικειμένων πολύ διαφορετικός από τους αλγορίθμους που βασίζονται στις περιοχές. Στο YOLO ένα ενιαίο συνελικτικό δίκτυο προβλέπει τα πλαίσια οριοθέτησης και τις πιθανότητες αντικειμένου. Δεδομένου ότι ολόκληρη η διαδικασία ανίχνευσης είναι σε ένα μεμονωμένο δίκτυο, μπορεί να βελτιστοποιηθεί πλήρως η απόδοση της ανίχνευσης.



Σχήμα 3.27: Μια απλοποιημένη απεικόνιση της λογικής του ανιχνευτή αντικειμένων YOLO [22].

Το YOLO χρησιμοποιεί μόνο συνελικτικά επίπεδα, καθιστώντας το ένα πλήρως συνελικτικό δίκτυο (FCN). Διαθέτει 75 συνελικτικά επίπεδα, με συνδέσεις παράλειψης και επίπεδα δειγματοληψίας. Δε χρησιμοποιείται καμία μορφή ομαδοποίησης και ένα συνελικτικό στρώμα με βήμα (stride) 2 χρησιμοποιείται για να υποδείξει τους χάρτες χαρακτηριστικών. Αυτό βοηθά στην πρόληψη της απώλειας χαρακτηριστικών χαμηλού επιπέδου που συχνά αποδίδονται στην ομαδοποίηση.

Καθώς πρόκειται για FCN, ο αλγόριθμος YOLO είναι αμετάβλητος από το μέγεθος της εικόνας εισόδου. Ωστόσο, στην πράξη, θέλουμε να παραμείνουμε σε ένα σταθερό μέγεθος εισόδου των εικόνων λόγω διαφόρων προβλημάτων που διαπιστώνουμε όταν εφαρμόζουμε τον αλγόριθμο.

Ένα μεγάλο από αυτά τα προβλήματα είναι ότι εάν θέλουμε να επεξεργαστούμε τις εικόνες μας σε παρτίδες (για να αξιοποιήσουμε τις δυνατότητες επεξεργασίας μιας GPU) πρέπει να έχουμε όλες τις εικόνες σταθερού ύψους και πλάτους. Αυτό απαιτείται για τη συνένωση πολλαπλών εικόνων σε μια μεγάλη παρτίδα.

Το δίκτυο υποδειγματοποιεί την εικόνα με έναν παράγοντα που ονομάζεται βήμα (stride) του δικτύου. Για παράδειγμα, εάν το βήμα του δικτύου είναι 32, τότε μία εικόνα εισόδου μεγέθους 416 x 416 θα αποδώσει έξοδο μεγέθους 13 x 13. Γενικά, το βήμα οποιουδήποτε επιπέδου στο δίκτυο είναι ίσο με τον παράγοντα με τον οποίο η έξοδος του κάθε επιπέδου είναι μικρότερη από το μέγεθος της εικόνας εισόδου

στο δίκτυο.

Ερμηνεία της εξόδου

Συνήθως, όπως συμβαίνει σε όλους τους ανιχνευτές αντικειμένων, τα χαρακτηριστικά που μαθαίνουν τα συνελικτικά επίπεδα μεταφέρονται σε έναν παλινδρομητή (regressor) που κάνει την πρόβλεψη ανίχνευσης (συντεταγμένες των πλαισίων οριοθέτησης, την ετικέτα κλάσης κ.λπ.).

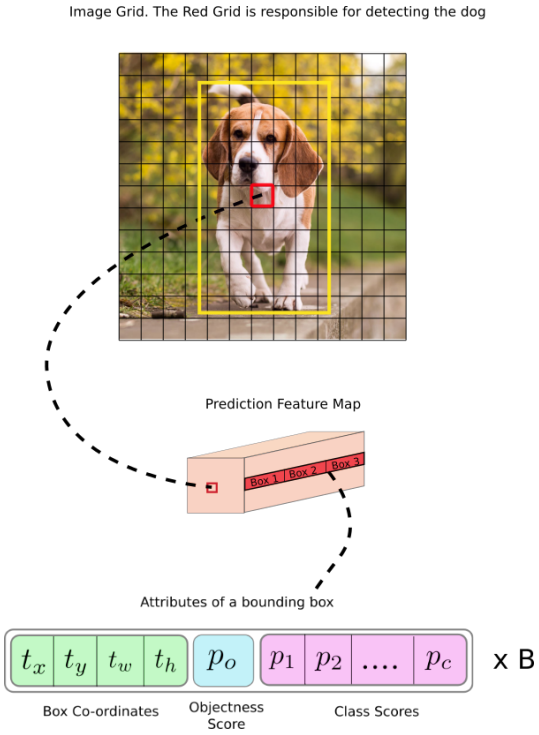
Στο YOLO, η πρόβλεψη γίνεται χρησιμοποιώντας ένα συνελικτικό επίπεδο που χρησιμοποιεί 1 x 1 συνελίξεις. Η έξοδος του είναι ένας χάρτης χαρακτηριστικών. Δεδομένου ότι έχουμε χρησιμοποιήσει 1 x 1 συνελίξεις, το μέγεθος του χάρτη προβλέψεων είναι ακριβώς όσο το μέγεθος του χάρτη χαρακτηριστικών πριν από αυτόν. Στο YOLO ο τρόπος με τον οποίο ερμηνεύεται αυτός ο χάρτης πρόβλεψης είναι ότι κάθε κελί μπορεί να προβλέψει έναν καθορισμένο αριθμό πλαισίων οριοθέτησης. Αν και τεχνικά ο σωστός όρος για την περιγραφή μιας μονάδας στο χάρτη χαρακτηριστικών είναι νευρώνας, αποκαλώντας το ως κελί το καθιστά πιο διαισθητικό στην περίπτωση του αλγορίθμου.

Αναλυτικότερα, έχουμε $(Bx(5+C))$ καταχωρήσεις στο χάρτη χαρακτηριστικών. Το B αντιπροσωπεύει τον αριθμό των πλαισίων οριοθέτησης που μπορεί να προβλέψει κάθε κελί. Σύμφωνα με τη δημοσίευση του αλγορίθμου, καθένα από αυτά τα πλαίσια οριοθέτησης B μπορεί να ειδικεύεται στην ανίχνευση ενός συγκεκριμένου είδους αντικειμένου. Κάθε ένα από τα πλαίσια οριοθέτησης έχει 5 + C χαρακτηριστικά, τα οποία περιγράφουν τις κεντρικές συντεταγμένες, τις διαστάσεις, τη βαθμολογία του αντικειμένου και τις C κλάσεις βαθμολογίες εμπιστοσύνης για κάθε πλαίσιο οριοθέτησης. Το YOLO προβλέπει 3 πλαίσια οριοθέτησης για κάθε κελί.

Κάθε κελί του χάρτη δυνατοτήτων μπορεί να προβλέψει ένα αντικείμενο μέσω ενός από τα πλαίσια οριοθέτησης, εάν το κέντρο του αντικειμένου πέσει στο δεκτικό πεδίο αυτού του κελιού. Αυτό σχετίζεται με τον τρόπο εκπαίδευσης του YOLO, όπου μόνο ένα πλαίσιο οριοθέτησης είναι υπεύθυνο για την ανίχνευση ενός αντικειμένου. Πρώτον, πρέπει να εξακριβώσουμε σε ποια κελιά ανήκει αυτό το πλαίσιο οριοθέτησης.

Για να γίνει αυτό, διαιρούμε την εικόνα εισόδου σε ένα πλέγμα διαστάσεων ίσων με εκείνες του τελικού χάρτη χαρακτηριστικών.

Στο σχήμα 3.28 παρουσιάζεται ένα παράδειγμα, όπου η εικόνα εισόδου είναι 416 x 416 και το βήμα του δικτύου είναι 32. Όπως επισημάνθηκε νωρίτερα, οι διαστάσεις του χάρτη χαρακτηριστικών θα είναι 13 x 13. Στη συνέχεια διαιρούμε την εικόνα εισόδου σε 13 x 13 κελιά



Σχήμα 3.28: Ο YOLOv3 διαιρεί την εικόνα σε κελιά 13 x 13, σύμφωνα με τις διαστάσεις του χάρτη χαρακτηριστικών. Το κόκκινο κελί είναι υπεύθυνο για την ανίχνευση του σκύλου στην εικόνα [23].

Στη συνέχεια, επιλέγεται το κελί (στην εικόνα εισόδου) που περιέχει το κέντρο του επίγειου αληθούς πλαισίου ενός αντικειμένου ως υπεύθυνο για την πρόβλεψη του αντικειμένου. Στην εικόνα, είναι το κελί που επισημαίνεται με κόκκινο χρώμα, το οποίο περιέχει το κέντρο του κιβωτίου αλήθειας εδάφους (σημειωμένο κίτρινο). Το κόκκινο κελί είναι το 7ο κελί στην 7η σειρά στο πλέγμα. Αναθέτουμε το 7ο κελί στην 7η σειρά στο χάρτη χαρακτηριστικών (αντίστοιχο κελί στο χάρτη χαρακτηριστικών) ως αυτό που είναι υπεύθυνο για την ανίχνευση του σκύλου. Αυτό το κελί μπορεί να προβλέψει τρία πλαίσια οριοθέτησης. Για να καταλάβουμε ποιο κελί θα ανατεθεί στην ετικέτα αλήθειας του σκύλου θα χρειαστούμε τις άγκυρες. Το κελί για το οποίο μιλάμε εδώ είναι ένα κελί στον χάρτη δυνατοτήτων πρόβλεψης. Διαιρούμε την εικόνα εισαγωγής σε ένα πλέγμα μόνο για να προσδιοριστεί ποιο κελί του χάρτη δυνατοτήτων πρόβλεψης είναι υπεύθυνο για την πρόβλεψη.

Πλαίσια αγκύρωσης

Μπορεί να έχει νόημα να προβλέψουμε το πλάτος και το ύψος του πλαισίου οριοθέτησης, αλλά στην πράξη, αυτό οδηγεί σε ασταθείς κλίσεις κατά τη διάρκεια της εκπαίδευσης. Αντ' αυτού, οι περισσότεροι από τους σύγχρονους ανιχνευτές αντικειμένων προβλέπουν μετασχηματισμούς χώρου καταγραφής ή απλώς αντισταθμίζουν τα προκαθορισμένα προεπιλεγμένα πλαίσια οριοθέτησης που ονομάζονται άγκυρες.

Στη συνέχεια, αυτοί οι μετασχηματισμοί εφαρμόζονται στα πλαίσια αγκύρωσης για να ληφθεί η πρόβλεψη. Το YOLO έχει τρεις άγκυρες, οι οποίες έχουν ως αποτέλεσμα την πρόβλεψη τριών πλαισίων οριοθέτησης ανά κελί.

Το πλαίσιο οριοθέτησης που είναι υπεύθυνο για την ανίχνευση του σκύλου στην παραπάνω εικόνα θα είναι εκείνο του οποίου η άγκυρα έχει το υψηλότερο IoU με το χαμηλότερο πλαίσιο.

Δημιουργία προβλέψεων

Οι ακόλουθοι τύποι περιγράφουν τον τρόπο μετατροπής της εξόδου του δικτύου για τη λήψη των προβλέψεων οριοθέτησης.

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

Σχήμα 3.29: Οι μαθηματικοί τύποι των bx, by, bw, bh [24].

bx, by, bw, bh είναι οι x, y κεντρικές συντεταγμένες, πλάτος και ύψος της πρόβλεψης. Τα tx, ty, tw, th είναι αυτό που εξάγει το δίκτυο. cx και cy είναι οι άνω-αριστερές συντεταγμένες του πλέγματος. Τα pw και ph είναι διαστάσεις αγκύρωσης για το πλαίσιο οριοθέτησης.

Συντεταγμένες κέντρων

Η πρόβλεψη των συντεταγμένων του κέντρου ενός αντικειμένου γίνεται μέσω μιας σιγμοειδούς συνάρτησης. Αυτό αναγκάζει την τιμή της εξόδου να κυμαίνεται

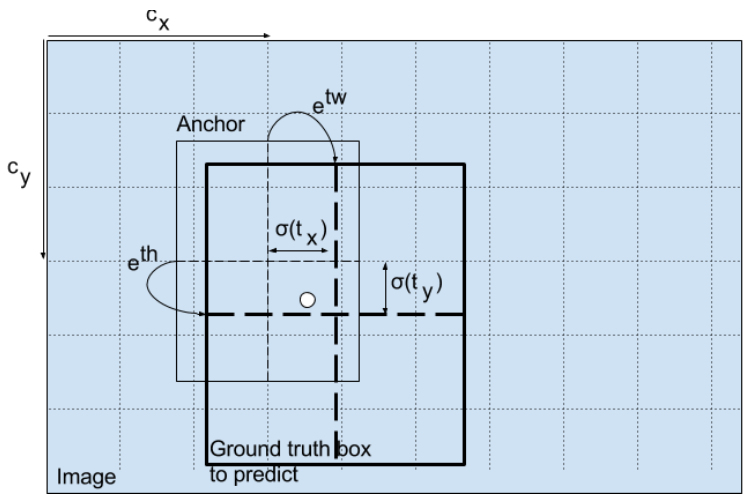
μεταξύ 0 και 1. Κανονικά, το YOLO δεν προβλέπει τις απόλυτες συντεταγμένες του κέντρου του πλαισίου οριοθέτησης. Προβλέπει αντισταθμίσεις που είναι:

- Σχετικά με την επάνω αριστερή γωνία του κελιού πλέγματος που προβλέπει το αντικείμενο.
- Και ομαλοποιείται από τις διαστάσεις του κελιού από το χάρτη χαρακτηριστικών, δηλαδή 1.

Για παράδειγμα, στην περίπτωση της εικόνας του σκύλου εάν η πρόβλεψη για το κέντρο είναι (0.4, 0.7), τότε αυτό σημαίνει ότι το κέντρο βρίσκεται στο (6.4, 6.7) στο χάρτη χαρακτηριστικών 13 x 13 (δεδομένου ότι οι άνω-αριστερές συντεταγμένες του κόκκινου κελιού είναι (6,6)). Αλλά στην περίπτωση που οι προβλεπόμενες συντεταγμένες x, y είναι περισσότερες από μία, π.χ. (1.2, 0.7) τότε αυτό σημαίνει ότι το κέντρο βρίσκεται στο (7.2, 6.7). Παρατηρούμε ότι το κέντρο βρίσκεται τώρα ακριβώς στο κόκκινο κελί, ή στο 8ο κελί στην 7η σειρά. Αυτό σπάει τη θεωρία πίσω από το YOLO γιατί αν υποθέσουμε ότι το κόκκινο πλαίσιο είναι υπεύθυνο για την πρόβλεψη του σκύλου, το κέντρο του σκύλου πρέπει να βρίσκεται στο κόκκινο κελί και όχι δίπλα σε αυτό. Επομένως, για την επίλυση αυτού του προβλήματος, η έξοδος περνά μέσω μιας σιγμοειδούς συνάρτησης, η οποία συμπιέζει την έξοδο σε εύρος από 0 έως 1, διατηρώντας αποτελεσματικά το κέντρο στο πλέγμα που προβλέπει.

Διαστάσεις του πλαισίου οριοθέτησης

Οι διαστάσεις του πλαισίου οριοθέτησης προβλέπονται εφαρμόζοντας έναν μετασχηματισμό χώρου-καταγραφής στην έξοδο και μετά πολλαπλασιάζοντας με μια άγκυρα.



Σχήμα 3.30: Διαστάσεις του πλαισίου οριοθέτησης του YOLOv3 [24].

Οι προκύπτουσες προβλέψεις, bw και bh, ομαλοποιούνται από το ύψος και το πλάτος της εικόνας (οι ετικέτες εκπαίδευσης επιλέγονται με αυτόν τον τρόπο). Έτσι, εάν οι προβλέψεις bx και by για το πλαίσιο που περιέχει το σκύλο είναι (0.3, 0.8), τότε το πραγματικό πλάτος και ύψος στο χάρτη χαρακτηριστικών 13 x 13 είναι (13 x 0.3, 13 x 0.8).

Βαθμολογία πιθανότητας αντικειμένου

Η βαθμολογία πιθανότητας αντικειμένου αντιπροσωπεύει την πιθανότητα ότι ένα αντικείμενο περιέχεται μέσα σε ένα πλαίσιο οριοθέτησης. Θα πρέπει να είναι σχεδόν 1 για το κόκκινο και τα γειτονικά πλέγματα, ενώ σχεδόν 0 για το πλέγμα στις γωνίες.

Η βαθμολογία πιθανότητας αντικειμένου περνά επίσης από ένα σιγμοειδές, καθώς πρέπει να ερμηνευτεί ως πιθανότητα.

Εμπιστοσύνη κλάσης

Οι εμπιστευτικές τάξεις αντιπροσωπεύουν τις πιθανότητες του αντικειμένου που ανιχνεύεται να ανήκει σε μια συγκεκριμένη τάξη (σκύλος, γάτα, μήλο, αυτοκίνητο κ.λπ.). Πριν από το v3, το YOLO συνήθιζε να κάνει χρήση της συνάρτησης softmax για τις βαθμολογίες των κλάσεων. Ωστόσο, αυτή η επιλογή σχεδιασμού έχει αφαιρεθεί στο v3 και οι συγγραφείς επέλεξαν να χρησιμοποιήσουν τη σιγμοειδή συνάρτηση. Ο λόγος είναι με ότι τη χρήση της softmax θεωρείται ότι οι κλάσεις είναι αμοιβαία αποκλειόμενες μεταξύ τους. Ως αποτέλεσμα, εάν ένα αντικείμενο ανήκει σε μία κλάση, τότε είναι εγγυημένο ότι δε μπορεί να ανήκει σε άλλη κλάση. Αυτό ισχύει για το σύνολο δεδομένων COCO [55] στην οποία βασίζεται το προ-εκπαιδευμένο

μοντέλο YOLOv3. Ωστόσο, αυτές οι υποθέσεις μπορεί να μην ισχύουν όταν έχουμε κλάσεις όπως *Άνθρωπος* και *Πρόσωπο*. Αυτός είναι ο λόγος που το YOLO απέφυγε τη χρήση μιας συνάρτησης ενεργοποίησης Softmax.

Πρόβλεψη σε διαφορετικές κλίμακες

Το YOLO v3 κάνει προβλέψεις σε 3 διαφορετικές κλίμακες. Το επίπεδο ανίχνευσης χρησιμοποιείται για την ανίχνευση σε χάρτες χαρακτηριστικών τριών διαφορετικών μεγεθών, με βήματα 32, 16 και 8, αντίστοιχα. Αυτό σημαίνει, με είσοδο 416 x 416, κάνουμε ανιχνεύσεις σε κλίμακες 13 x 13, 26 x 26 και 52 x 52.

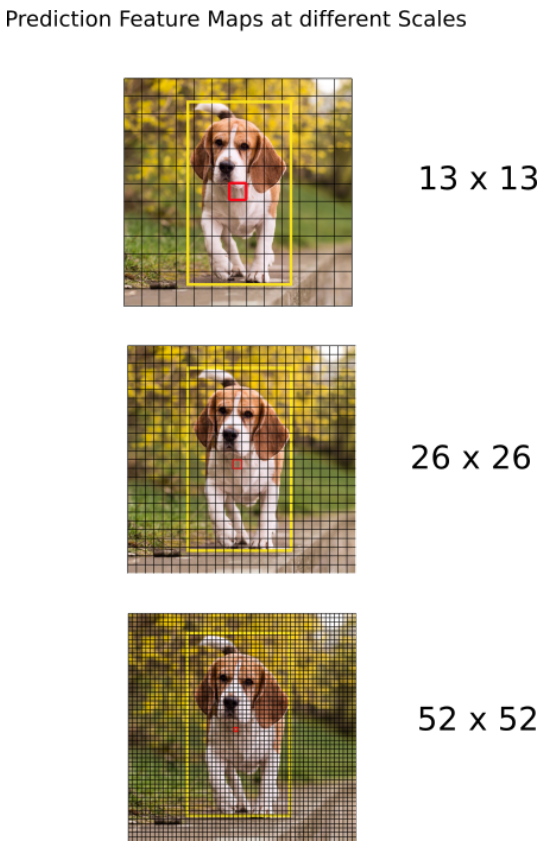
Το δίκτυο υποδειγματοποιεί την εικόνα εισόδου μέχρι το πρώτο επίπεδο ανίχνευσης, όπου γίνεται μια ανίχνευση χρησιμοποιώντας χάρτες χαρακτηριστικών ενός επιπέδου με βήμα 32. Επιπλέον, τα επίπεδα προστίθενται σε δείγμα με συντελεστή 2 και συνδυάζονται με τους χάρτες χαρακτηριστικών των προηγούμενων επιπέδων που έχουν τον ίδιου μεγέθους χάρτη χαρακτηριστικών. Μια άλλη ανίχνευση γίνεται και στο επίπεδο με το βήμα 16. Επαναλαμβάνεται η ίδια διαδικασία δειγματοληψίας και η τελική ανίχνευση γίνεται στο επίπεδο του βήματος 8.

Σε κάθε κλίμακα, κάθε κελί προβλέπει 3 πλαίσια οριοθέτησης χρησιμοποιώντας 3 άγκυρες, κάνοντας τον συνολικό αριθμό αγκυρών που χρησιμοποιούνται 9. Οι άγκυρες είναι διαφορετικές για διαφορετικές κλίμακες.

Οι συγγραφείς αναφέρουν ότι αυτό βοηθά το YOLO v3 να βελτιωθεί στον εντοπισμό μικρών αντικειμένων, πράγμα που αποτελούσε συχνό παράπονο με τις προηγούμενες εκδόσεις του YOLO. Η δειγματοληψία μπορεί να βοηθήσει το δίκτυο να μάθει λεπτομερείς λειτουργίες που είναι καθοριστικές για τον εντοπισμό μικρών αντικειμένων.

Επεξεργασία της εξόδου του δικτύου

Για μια εικόνα μεγέθους 416 x 416, το YOLO προβλέπει ((52 x 52) + (26 x 26) + 13 x 13)) x 3 = 10647 πλαίσια οριοθέτησης. Ωστόσο, στην περίπτωση της εικόνας στο σχήμα 3.32, υπάρχει μόνο ένα αντικείμενο, ένας σκύλος. Για να μειώσει το YOLO τις ανιχνεύσεις από 10647 σε 1 χρησιμοποιεί τις 2 παρακάτω μεθόδους, θέτει ένα ελάχιστο κατώφλι για την εμπιστοσύνη των αντικειμένων (threshold) και χρησιμοποιεί τη μη μέγιστη καταστολή.



Σχήμα 3.31: Προβλέψεις χάρτη χαρακτηριστικών σε διαφορετικά μεγέθη [22].

Ελάχιστο όριο για την εμπιστοσύνη αντικειμένων

Τα πλαίσια οριοθέτησης που ανιχνεύθηκαν με βαθμολογία κάτω από ένα όριο αγνοούνται.

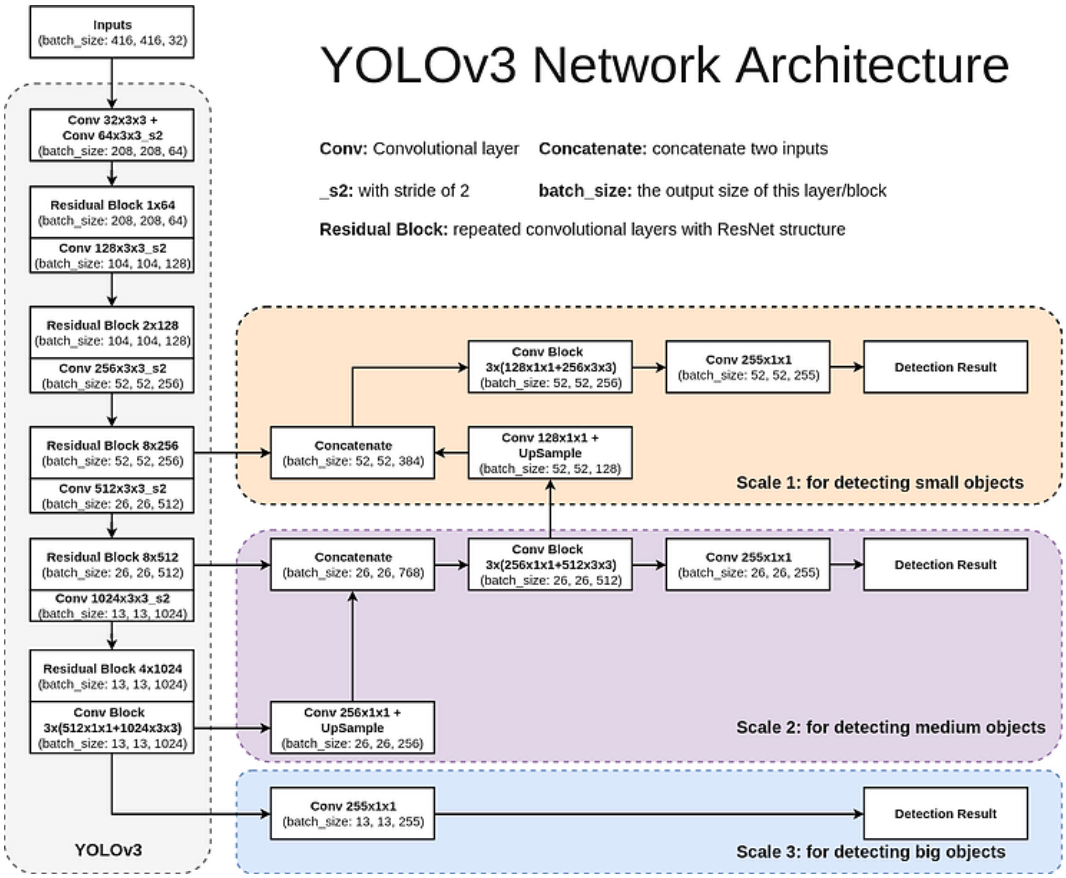
Μη μέγιστη καταστολή (Non-maximum Suppression)

Η μη μέγιστη καταστολή λύνει το πρόβλημα πολλαπλών ανιχνεύσεων ενός αντικειμένου στην ίδια εικόνα. Για παράδειγμα, και τα 3 πλαίσια οριοθέτησης του κελιού του πλέγματος μπορεί να ανιχνεύουν ένα πλαίσιο ή τα παρακείμενα κελιά μπορεί να ανιχνεύουν το ίδιο αντικείμενο.

Σχήμα 3.32: Πολλαπλά grids μπορεί να ανιχνεύουν το ίδιο αντικείμενο. Η μη μέγιστη καταστολή χρησιμοποιείται για να απαλείψει τις διπλότυπες ανιχνεύσεις [22].

Η αρχιτεκτονική του YOLO

ΤΟ YOLO αποτελεί ένα δίκτυο βασισμένο στη μάθηση χαρακτηριστικών και υιοθετεί 75 συνελικτικά επίπεδα ως το πιο ισχυρό εργαλείο του. Δε χρησιμοποιείται κανένα πλήρως συνδεδεμένο επίπεδο. Αυτή η δομή καθιστά δυνατή την αντιμετώπιση εικόνων με οποιαδήποτε μέγεθος. Επίσης, δε χρησιμοποιούνται επίπεδα συγκέντρωσης. Αντ 'αυτού, χρησιμοποιείται ένα συνελικτικό επίπεδο με το βήμα 2 για να υποδειγματογραφήσει το χάρτη χαρακτηριστικών. Επιπλέον, χρησιμοποιεί δομές που μοιάζει με το ResNet και το FPN για τη βελτίωση της ακρίβειας του. Στο σχήμα 3.38 παρουσιάζεται η αρχιτεκτονική δικτύου του YOLOv3.



Σχήμα 3.33: Διάγραμμα της αρχιτεκτονικής δικτύου του YOLOv3 [23].

Περιορισμοί και μειονεκτήματα του ανιχνευτή αντικειμένων YOLOv3

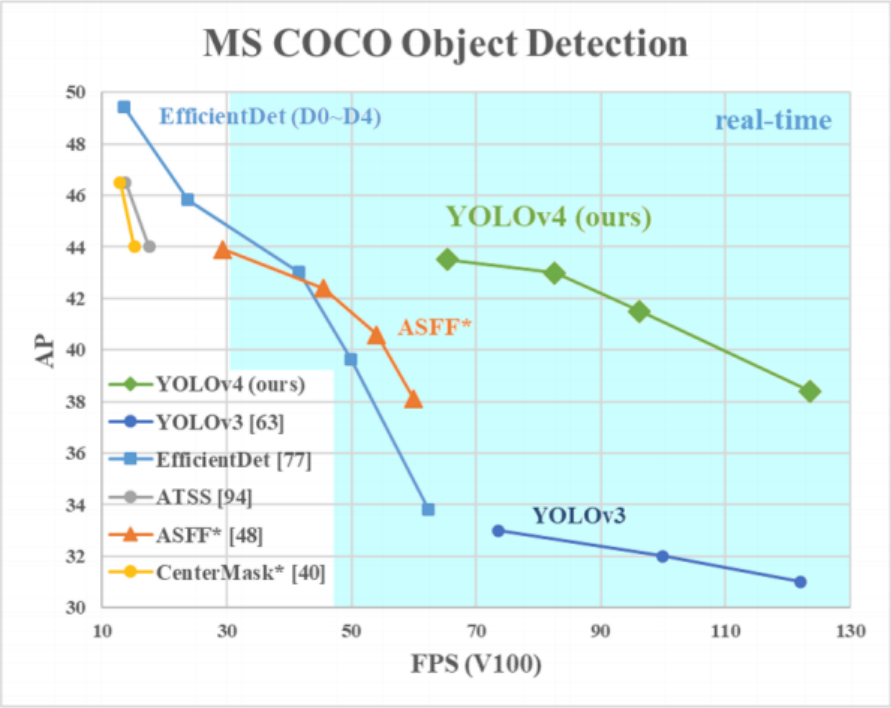
Αναμφισβήτητα ο μεγαλύτερος περιορισμός και μειονέκτημα του ανιχνευτή αντικειμένων YOLO είναι ότι δεν εντοπίζει πάντα καλά τα μικρά αντικείμενα και ειδικότερα δε χειρίζεται καλά αντικείμενα ομαδοποιημένα μεταξύ τους. Αυτό προκύπτει επειδή ο YOLO διαιρεί μια εικόνα εισόδου σε ένα πλέγμα SxS όπου κάθε κελί στο πλέγμα προβλέπει μόνο ένα αντικείμενο. Εάν υπάρχουν πολλά, μικρά αντικείμενα σε ένα μόνο κελί, τότε το YOLO δε θα μπορεί να τα εντοπίσει, οδηγώντας τελικά σε απώλειες ανιχνεύσεων αντικειμένων.

Βελτιώσεις του YOLO

Η τελευταία έκδοση του αλγορίθμου YOLO είναι το YOLOv4 [25] από τους συγγραφείς Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. Στην ενότητα 3.10 γίνεται ενδελεχής μελέτη του αλγορίθμου YOLOv4.

3.10 YOLOv4

Το YOLOv4 είναι μια σημαντική βελτίωση του YOLOv3. Η εφαρμογή μιας νέας αρχιτεκτονικής στο Backbone και οι τροποποιήσεις στο Neck έχουν επιτύχει εξαιρετικά αποτελέσματα με ταχύτητα ανίχνευσης σε πραγματικό χρόνο. Στο σύνολο δεδομένων MS COCO, το YOLOv4 επιτυγχάνει μέση ακρίβεια ίση με 43,5% AP και ταχύτητα στα 65 FPS σε μια GPU Tesla V100 [25]. Για να επιτύχει αυτά τα αποτελέσματα, συνδυάζει δυνατότητες όπως σταθμισμένες συνδέσεις, Cross-Stage-Partial-συνδέσεις (CSP), Self-Adversial-training (SAT), Mish ενεργοποιήσεις, επαυξήσεις δεδομένων, κανονικοποίηση DropBlock και απόκλιση CIoU.



Σχήμα 3.34: Το YOLOv4 Βελτιώνει το AP και το FPS του YOLOv3 κατά 10% και 12%, αντίστοιχα [25].

Από το σχήμα 3.34 μπορούμε να διακρίνουμε ότι το EfficientDet D4-D3 επιτυγχάνει καλύτερο AP από τα μοντέλα YOLO v4, αλλά λειτουργούν με ταχύτητα < 30 FPS σε GPU V100. Από την άλλη πλευρά, το YOLO είναι σε θέση να τρέχει με πολύ μεγαλύτερη ταχύτητα (60+ FPS) με πολύ καλή ακρίβεια. Τα μοντέλα που εμπίπτουν στη γαλάζια περιοχή στο σχήμα 3.34 θεωρούνται ανιχνευτές αντικειμένων σε πραγματικό χρόνο (30+ FPS).

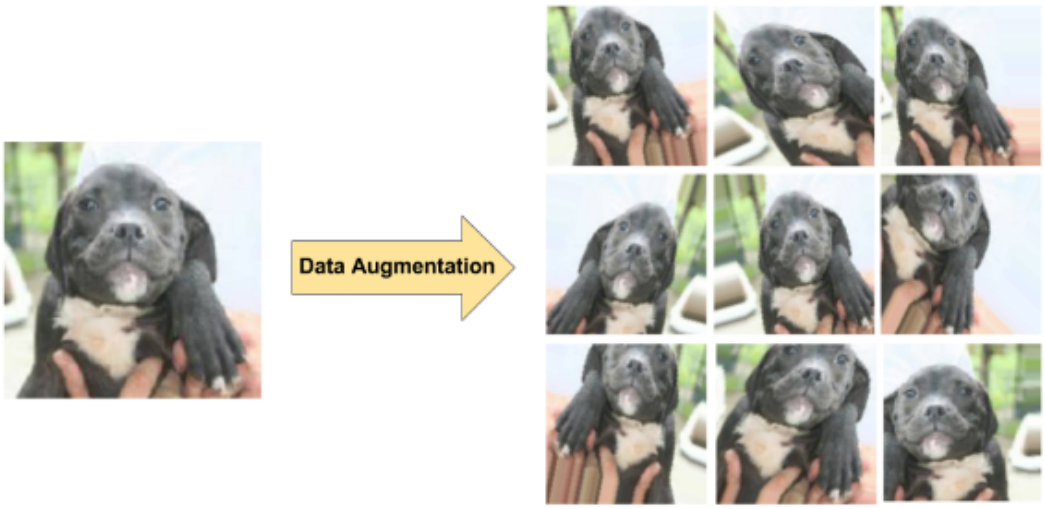
Η αρχιτεκτονική του backbone του YOLOv4 αποτελείται από τρία βασικά μέρη:

- Το Bag of freebies
- Το Bag of specials
- Και το CSPDarknet53

Bag of freebies

Το Bag of freebies είναι το σύνολο μεθόδων που μπορούν να κάνουν τον ανιχνευτή αντικειμένων να λαμβάνει καλύτερη ακρίβεια χωρίς να αυξάνεται το κόστος συμπερασμάτων. Αυτές οι μέθοδοι αλλάζουν μόνο τη στρατηγική εκπαίδευσης ή αυξάνουν μόνο το κόστος της εκπαίδευσης [25].

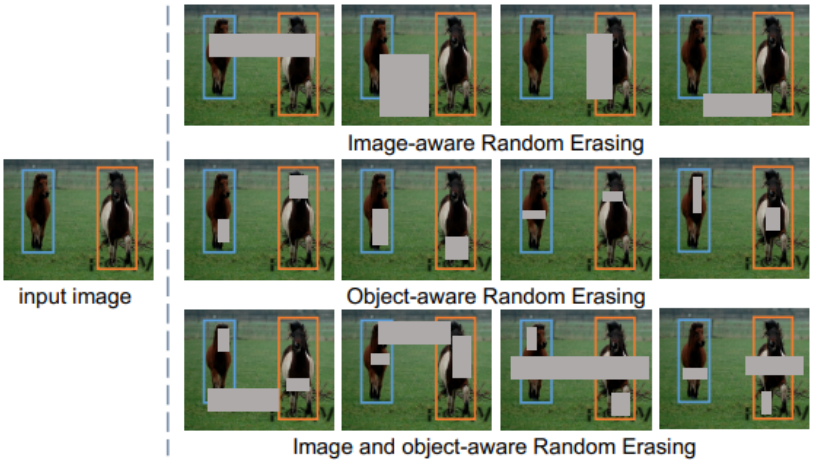
Ένα παράδειγμα του Bag of freebies είναι η επαύξηση δεδομένων (data augmentation), η οποία ενισχύει την ικανότητα γενίκευσης του μοντέλου. Για να το κάνουμε αυτό μπορούμε να κάνουμε φωτομετρικές παραμορφώσεις όπως: αλλαγή της φωτεινότητας, του κορεσμού, της αντίθεσης και του θορύβου ή μπορούμε να κάνουμε γεωμετρική παραμόρφωση μιας εικόνας, όπως περιστροφή, περικοπή κ.λπ. Οι πιο συχνά χρησιμοποιούμενες μέθοδοι είναι η φωτομετρική παραμόρφωση, η γεωμετρική παραμόρφωση, το MixUp, το CutMix και τα GAN.



Σχήμα 3.35: Παράδειγμα γεωμετρικών παραμορφώσεων [26].

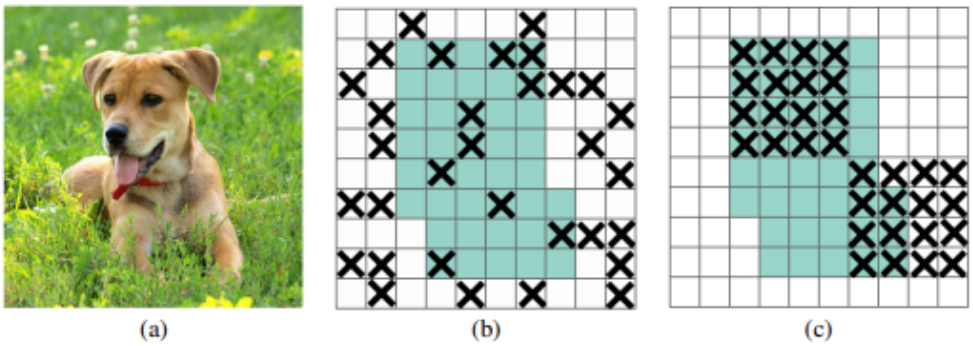
Υπάρχουν και άλλες ενδιαφέρουσες τεχνικές επαύξησης των εικόνων όπως το CutOut [56] το οποίο αποκρύπτει τυχαία τις τετραγωνικές περιοχές της εικόνας εισόδου κατά τη διάρκεια της εκπαίδευσης. Αυτό αποδείχθηκε ότι βελτιώνει την

ευρωστία και την απόδοση των CNN. Ομοίως, η Τυχαία Διαγραφή (Random Erasing) [27] επιλέγει τυχαία ορθογώνιες περιοχές σε μια εικόνα και σβήνει τα pixel αυτής της επιλογής. Παράδειγμα τυχαίων διαγραφών παρουσιάζεται στο σχήμα 3.36.

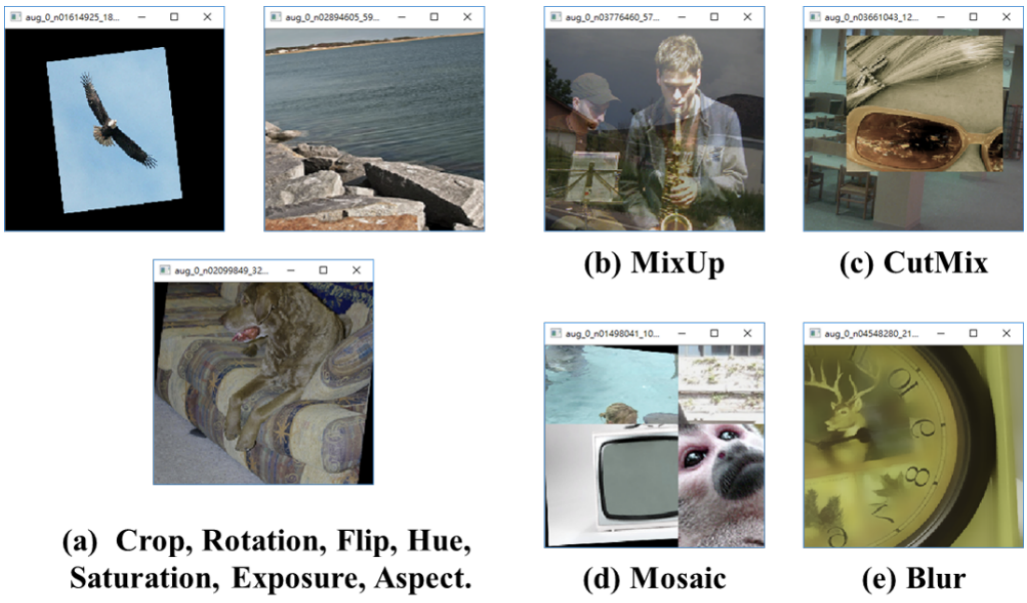


Σχήμα 3.36: Παράδειγμα τυχαίων διαγραφών στην ανίχνευση αντικειμένων [27].

Άλλες τεχνικές του Bag of Freebies είναι οι τεχνικές κανονικοποίησης που χρησιμοποιούνται για την αποφυγή υπερβολικής εφαρμογής κατά την εκπαίδευση (overfitting), όπως: DropOut, DropConnect και DropBlock [28]. Το DropBlock δείχνει στην πραγματικότητα πολύ καλά αποτελέσματα στα CNN και χρησιμοποιείται στο YOLO v4.



Σχήμα 3.37: (α) Εικόνα εισαγωγής σε συνελικτικό νευρωνικό δίκτυο. Οι πράσινες περιοχές στα (β) και (γ) περιλαμβάνουν μονάδες ενεργοποίησης που περιέχουν σημασιολογικές πληροφορίες στην εικόνα εισόδου. Η κατάργηση των ενεργοποιήσεων με τυχαίο τρόπο δεν είναι αποτελεσματική για την αφαίρεση σημασιολογικών πληροφοριών, επειδή οι κοντινές ενεργοποιήσεις περιέχουν πολύ σχετικές πληροφορίες. Αντίθετα, η απομάκρυνση συνεχών περιοχών μπορεί να αφαιρέσει ορισμένες σημασιολογικές πληροφορίες (π.χ. κεφάλι ή πόδια) και κατά συνέπεια να επιβάλει τις υπόλοιπες μονάδες για να μάθουν χαρακτηριστικά για την κατηγοριοποίηση της εικόνας εισόδου [28].



Σχήμα 3.38: Bag of Freebies στο YOLOv4 [25].

Bag of specials

Το Bag of specials είναι ένα σύνολο ειδικών μεθόδων που αυξάνουν το κόστος της ανίχνευσης κατά μια μικρή τιμή, αλλά βελτιώνουν σημαντικά την ακρίβεια του ανιχνευτή. Αυτές οι μέθοδοι περιλαμβάνουν την εισαγωγή μηχανισμών προσοχής (Squeeze-and-Excitation και Spatial Attention Module), διεύρυνση του δεκτικού πεδίου του μοντέλου και ενίσχυση της δυνατότητας ενσωμάτωσης χαρακτηριστικών. Τα κοινά τμήματα που χρησιμοποιούνται για τη βελτίωση του δεκτικού πεδίου είναι SPP, ASPP και RFB. Το YOLO v4 χρησιμοποιεί το SPP.

Αυτές οι πυραμίδες χαρακτηριστικών εμπίπτουν επίσης στην κατηγορία του Bag of Specials, όπως και οι συναρτήσεις ενεργοποίησης. Από τότε που κυκλοφόρησε το ReLU, υπήρξαν πολλές παραλλαγές του, όπως το LReLU, το PReLU και το ReLU6. Συναρτήσεις ενεργοποίησης όπως το ReLU6 και το hard-Swish έχουν σχεδιαστεί ειδικά για κβαντοποιημένα δίκτυα που χρησιμοποιούνται για να δίνουν συμπεράσματα σε ενσωματωμένες συσκευές, όπως στο Google Coral Edge TPU.

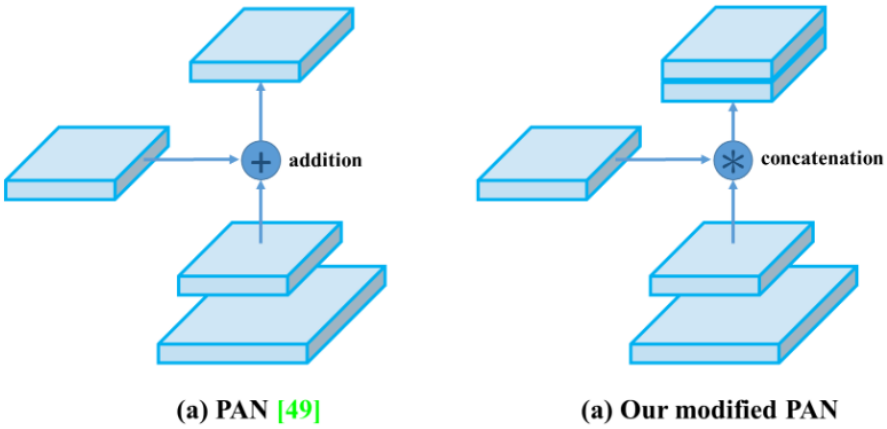
Από την άλλη πλευρά, το YOLO v4 χρησιμοποιεί αρκετά τη συνάρτηση ενεργοποίησης Mish στο backbone, η οποία αναφέρεται στην ενότητα 2.7.2. Το Mish είναι μια συνάρτηση ενεργοποίησης που έχει σχεδιαστεί για να ωθεί τα σήματα προς τα αριστερά και προς τα δεξιά. Στο YOLO v4 αποδεικνύεται ότι αυτή η συνάρτηση ενεργοποίησης παρουσιάζει πολύ καλύτερα αποτελέσματα.

Επίσης, το YOLOv4 ενσωματώνει το DIoU NMS [57] για να διαχωρίσει τα προβλεπόμενα πλαίσια οριοθέτησης. Το δίκτυο μπορεί να προβλέψει πολλαπλά πλαίσια οριοθέτησης πάνω από ένα αντικείμενο και πρέπει να επιλέξει το καλύτερο από αυτά.

Αρχιτεκτονική του YOLOv4

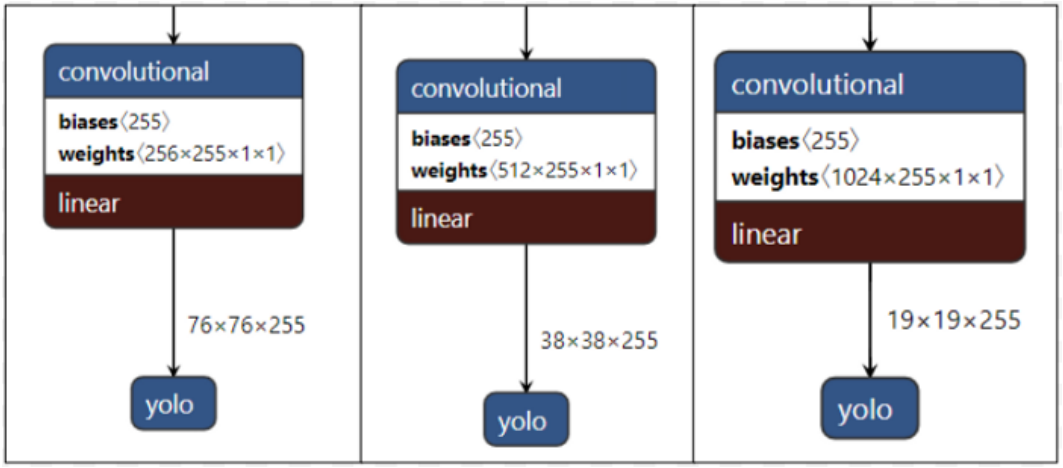
- **Backbone**: Χρησιμοποιεί το CSPDarknet53 ως το μοντέλο εξαγωγής χαρακτηριστικών για την έκδοση GPU. Το CSPDarknet53 είναι ένα νέο backbone που μπορεί να βελτιώσει τη μαθησιακή ικανότητα του CNN. Το μπλοκ χωρικής πυραμίδας προστίθεται μέσω του CSPDarknet53 για να αυξήσει το δεκτικό πεδίο και να διαχωρίσει τα πιο σημαντικά χαρακτηριστικά περιβάλλοντος. Ακόμα, αντί να χρησιμοποιηθεί το FPN όπως στο YOLOv3, χρησιμοποιείται το PANet ως μέθοδος για τη συγκέντρωση παραμέτρων για τα διαφορετικά επίπεδα του ανιχνευτή.

- **Neck**: Το YOLOv4 χρησιμοποιεί Spatial pyramid pooling (SPP) και Path Aggregation Network (PAN). Το PAN δεν είναι πανομοιότυπο με το αρχικό PAN, αλλά αποτελεί τροποποιημένη έκδοση που αντικαθιστά την προσθήκη μιας συνένωσης (concatenation). Στο σχήμα 3.39 παρουσιάζεται η τροποποιημένη έκδοση του PAN.



Σχήμα 3.39: Τροποποιημένο επίπεδο PAN YOLOv4 [25].

- **Head**: Το YOLOv4 χρησιμοποιεί την ίδια κεφαλή YOLO με το YOLOv3 για ανίχνευση με τα βήματα ανίχνευσης βάσει αγκύρωσης και τρία επίπεδα κοκκώδους ανίχνευσης.



Σχήμα 3.40: Το Head του YOLOv4.

Αυτές είναι οι κεφαλές που εφαρμόζονται σε διαφορετικές κλίμακες του δικτύου, για την ανίχνευση αντικειμένων διαφορετικού μεγέθους. Ο αριθμός των καναλιών είναι 255 και προκύπτει από τις 80 κλάσεις του συνόλου δεδομένων COCO + 1 για την βαθμολογία της πιθανότητας του αντικειμένου + 4 συντεταγμένες * 3 άγκυρες.

Πρόσθετες βελτιώσεις

Στο YOLOv4 εισάγεται μια νέα μέθοδος επαύξησης δεδομένων που ονομάζεται Μωσαϊκό. Αυτή η μέθοδος συνδυάζει 4 εικόνες του συνόλου δεδομένων εκπαίδευσης σε 1 εικόνα. Με αυτόν τον τρόπο, η ομαλοποίηση παρτίδας υπολογίζει στατιστικά στοιχεία ενεργοποίησης από 4 διαφορετικές εικόνες σε κάθε επίπεδο [25]. Ως αποτέλεσμα, μειώνεται σημαντικά η ανάγκη επιλογής μεγάλου μεγέθους μίνι παρτίδας για εκπαίδευση. Παράδειγμα επαύξησης δεδομένων με τη χρήση του Μωσαϊκού παρουσιάζεται στο σχήμα 3.41.

Επιπρόσθετα, χρησιμοποιούν την Αυτό-Αντιπαραθετική εκπαίδευση (Self-Adversarial Training - SAT), η οποία λειτουργεί σε δύο στάδια, προς τα εμπρός και προς τα πίσω. Στο 1ο στάδιο, το νευρωνικό δίκτυο τροποποιεί την αρχική εικόνα αντί για τα βάρη του δικτύου. Με αυτόν τον τρόπο, το νευρωνικό δίκτυο εκτελεί μια εχθρική επίθεση στον εαυτό του, αλλάζοντας την αρχική εικόνα για να δημιουργήσει την εξαπάτηση ότι δεν υπάρχει επιθυμητό αντικείμενο στην εικόνα. Στο 2ο στάδιο, το νευρωνικό δίκτυο εκπαιδεύεται για να ανιχνεύει ένα αντικείμενο σε αυτήν την τροποποιημένη εικόνα με τον κανονικό τρόπο.

Σχήμα 3.41: Η ομαλοποίηση παρτίδας υπολογίζει στατιστικά στοιχεία ενεργοποίησης από 4 διαφορετικές εικόνες σε κάθε επίπεδο [25].

Tiny Yolo

Μαζί με το YOLO, κάνει την εμφάνιση του και ο αλγόριθμος Tiny-YOLO. Ο Tiny-YOLO αποτελεί μια μικρότερη έκδοση του συνελικτικού νευρωνικού δικτύου του YOLO και είναι εξαιρετικά γρήγορος. Η αρχιτεκτονική του Tiny-YOLO είναι περίπου 442% ταχύτερη από το YOLO, επιτυγχάνοντας πάνω από 244 FPS σε μία μόνο GPU. Ωστόσο, ο Tiny-YOLO έχει σχεδόν τη μισή ακρίβεια από το YOLO. Αυτό συμβαίνει καθώς χρησιμοποιεί ένα ελαφρύτερο μοντέλο με λιγότερα επίπεδα σε σύγκριση με το κανονικό YOLO, αλλά η εικόνα εισόδου διατηρεί το ίδιο μέγεθος με αυτή του YOLO. Το μικρό μέγεθος του μοντέλου (<50MB) και η γρήγορη ταχύτητα συμπερασμάτων καθιστούν τον ανιχνευτή αντικειμένων Tiny-YOLO κατάλληλο για ενσωματωμένες συσκευές υπολογιστικής όρασης και βαθιάς μάθησης υπολογιστή όπως το Raspberry Pi, το Google Coral και το NVIDIA Jetson Nano.

Κεφάλαιο 4

Λογισμικά και Τεχνολογίες

Σε αυτήν την ενότητα αναφέρεται όλο το λογισμικό που χρησιμοποιήθηκε για την υλοποίηση της διπλωματικής εργασίας. Η εργασία αναπτύχθηκε με τη γλώσσα προγραμματισμού Python και με τη χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης (IDE) JetBrains PyCharm και του περιβάλλοντος JupyterLab. Παράλληλα, βασικό ρόλο στο κομμάτι των αλγορίθμων έχουν οι βιβλιοθήκες OpenCV και Tensorflow και το πλαίσιο εργασίας Darknet.

4.1 JetBrains PyCharm IDE

Το PyCharm είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που χρησιμοποιείται στον προγραμματισμό με τη γλώσσα Python [58]. Αναπτύχθηκε από την τσεχική εταιρεία JetBrains. Παρέχει ανάλυση κώδικα, γραφικό περιβάλλον εντοπισμού σφαλμάτων, επισήμανση σφαλμάτων εν κινήσει και γρήγορες επιδιορθώσεις, ενσωματωμένη μονάδα δοκιμών (unit test), διασύνδεση με συστήματα ελέγχου εκδόσεων (Git) και υποστηρίζει την επιστήμη δεδομένων με το Anaconda. Διατίθεται στις εκδόσεις PyCharm Community Edition που είναι δωρεάν για όλους και στην PyCharm Professional Edition επί πληρωμή.

4.2 Python

Η Python μια από τις δημοφιλέστερες γλώσσες προγραμματισμού υψηλού επιπέδου. Είναι διερμηνευόμενη (interpreted) και γενικού σκοπού (general-purpose) [59]. Ανήκει στις γλώσσες προστακτικού προγραμμα-

τισμού (Imperative programming) και υποστηρίζει τόσο το διαδικαστικό (procedural programming) όσο και το αντικειμενοστρεφές (object-oriented programming) προγραμματιστικό υπόδειγμα (programming paradigm).
Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (garbage collection).

Δημιουργήθηκε από τον Ολλανδό Guido van Rossum στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989 και κυκλοφόρησε για πρώτη φορά το 1991. Η πιο πρόσφατη έκδοση της Python είναι η 3.9.1 (20 Δεκεμβρίου 2020).

Η Python, καθώς είναι μια από τις πιο γνωστές γλώσσες προγραμματισμού, διαθέτει παρά πολλά πακέτα όπου ο κάθε προγραμματιστής μπορεί να τα κατεβάσει από τα αποθετήρια πακέτων και να τα εισάγει στον κώδικά του.

Παράλληλα με την εγκατάσταση της Python θα εγκαταστήσουμε και τους διαχειριστές πακέτων pip και Anaconda προκειμένου να εγκατασταθούν όλα τα απαραίτητα πακέτα για την υλοποίηση της διπλωματικής εργασίας.

PIP

Το PIP είναι ο διαχειριστής πακέτων για πακέτα Python. Ένα σημαντικό πλεονέκτημα του pip είναι η ευκολία της διεπαφής γραμμής εντολών, η οποία καθιστά την εγκατάσταση πακέτων λογισμικού Python τόσο εύκολη όσο η έκδοση μιας εντολής:
pip install some-package-name
Οι χρήστες μπορούν επίσης να καταργήσουν εύκολα το πακέτο μέσω της εντολής:
pip uninstall some-package-name

Το πιο σημαντικό είναι ότι το pip διαθέτει μια δυνατότητα διαχείρισης πλήρων λιστών πακέτων και αντίστοιχων αριθμών έκδοσης, που είναι δυνατή μέσω ενός αρχείου ''απαιτήσεων''. Έτσι επιτρέπει την αποτελεσματική επαναδημιουργία ολόκληρης ομάδας πακέτων σε ξεχωριστό περιβάλλον (π.χ. άλλον υπολογιστή) ή εικονικό περιβάλλον. Αυτό μπορεί να επιτευχθεί με ένα ειδικά μορφοποιημένο αρχείο και την ακόλουθη εντολή, όπου το requirements.txt είναι το όνομα του αρχείου:
pip install -r requirements.txt
Παράδειγμα αρχείου requirements.txt παρουσιάζεται στο σχήμα 4.1.

```
pandas
numpy
tensorflow-gpu==1.15
opencv-python
matplotlib
scikit-image
flask
setuptools
cython
pycocotools
h5py
keras
```

Σχήμα 4.1: Αρχείο λίστας απαιτήσεων πακέτων Python (requirements.txt).

Anaconda

Το Anaconda είναι μια δωρεάν και ανοιχτού κώδικα διανομή των γλωσσών προγραμματισμού Python και R για επιστημονικούς υπολογιστές (επιστήμη δεδομένων, εφαρμογές μηχανικής μάθησης, επεξεργασία δεδομένων μεγάλης κλίμακας, προγνωστικά αναλυτικά στοιχεία κ.λπ.), που στοχεύει στην απλοποίηση της διαχείρισης πακέτων και στη διαχείριση του περιβάλλοντος προγραμματισμού. Οι εκδόσεις πακέτων διαχειρίζονται από το σύστημα διαχείρισης πακέτων [60]. Η διανομή Anaconda περιλαμβάνει πακέτα επιστήμης δεδομένων κατάλληλα για Windows, Linux και macOS. Χρησιμοποιούμε αυτό το πακέτο λογισμικού για να δημιουργήσουμε συγκεκριμένο περιβάλλον για την εγκατάσταση και τη διαχείριση των πακέτων λογισμικού της Python για την υλοποίηση της διπλωματικής εργασίας.

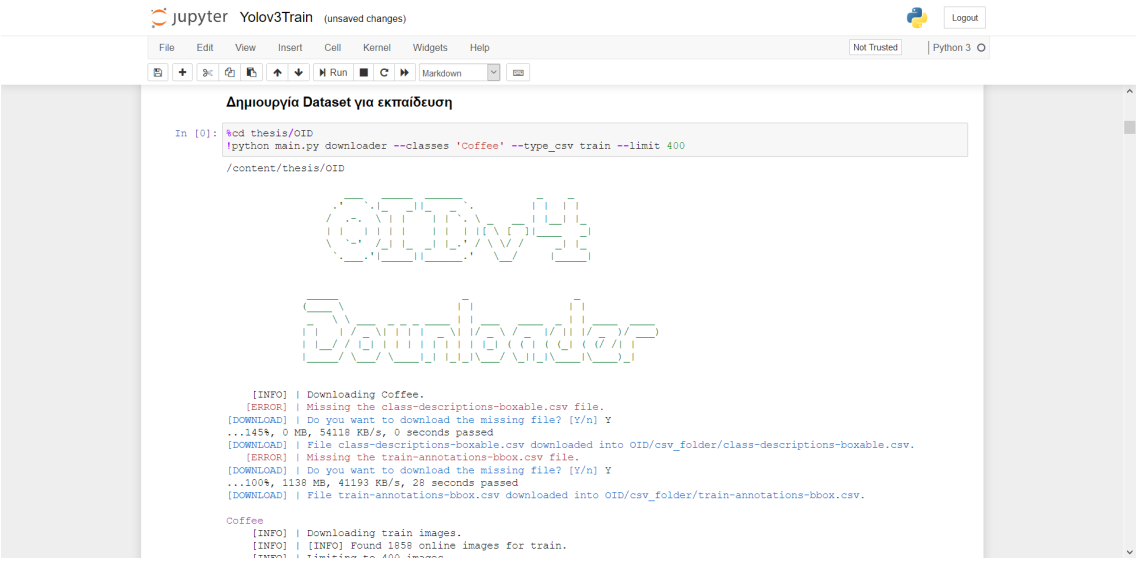
Για να δημιουργήσουμε ένα νέο περιβάλλον χρησιμοποιούμε την εντολή:
conda create –name ThesisEnvironment
και έπειτα ενεργοποιούμε το περιβάλλον με την εντολή:
source conda acticate ThesisEnvironment
Έπειτα μπορούμε να εγκαταστήσουμε τα πακέτα που θέλουμε σε αυτό το περιβάλλον είτε μέσω conda (conda install -c conda-forge όνομα-του-πακέτου) είτε μέσω pip (pip install όνομα-του-πακέτου).

4.3 Jupyter Notebooks

Το Jupyter Notebooks είναι μια διαδικτυακή εφαρμογή ανοιχτού κώδικα που επιτρέπει τη δημιουργία και τον διαμοιρασμό εγγράφων που περιέχουν ζωντανό κώδικα, εξισώσεις, οπτικοποιήσεις και αφηγηματικό κείμενο. Οι χρήσεις περιλαμβάνουν: καθαρισμό και μετατροπή δεδομένων, αριθμητική προσομοίωση, στατιστική μοντελοποίηση, οπτικοποίηση δεδομένων, μηχανική μάθηση και πολλά άλλα [61].

Πάνω σε Jupyter Notbooks τρέχουν και όλοι οι αλγόριθμοι της διπλωματικής εργασίας τόσο στο κομμάτι της εκπαίδευσης (που θα δούμε στο επόμενο κεφάλαιο), όσο και στο κομμάτι της ανίχνευσης αντικείμενων σε εικόνες και βίντεο.

Στο σχήμα 4.2 παρουσιάζεται το περιβάλλον του Jupyter Notebook που τρέχει ένα notebook για την εκπαίδευση του αλγορίθμου YOLOv3.



Σχήμα 4.2: Περιβάλλον Jupyter Notebook.

4.4 OpenCV

Το OpenCV είναι μια βιβλιοθήκη ανοιχτού κώδικα, υπολογιστικής όρασης για εξαγωγή και επεξεργασία σημαντικών δεδομένων από εικόνες [62]. Αυτά τα σημαντικά δεδομένα ενδέχεται να περιλαμβάνουν την εύρεση όλων ή τμημάτων αντικειμένων, την αναγνώριση όλων ή τμημάτων αντικειμένων, την παρακολούθηση της κίνησης (τμημάτων) αντικειμένων σε 2D ή 3D μεταξύ διαδοχικών εικόνων, προσδιορισμό του σχήματος 2D ή 3D αντικειμένων από μία ή περισσότερες εικόνες, και συσχέτιση των δεδομένων εικόνας με μια κατη-

γορηματική έννοια, όπως η χαρτογράφηση ενός handwave στην κατηγορία ”αντίο”.

4.5 CUDA

Το CUDA (Compute Unified Device Architecture) είναι μια παράλληλη πλατφόρμα υπολογιστών και μοντέλο διεπαφής προγραμματισμού εφαρμογών (API) που δημιουργήθηκε από την Nvidia [63]. Επιτρέπει στους προγραμματιστές λογισμικού να χρησιμοποιούν μια μονάδα επεξεργασίας γραφικών με δυνατότητα CUDA (GPU) για επεξεργασία γενικού σκοπού - μια προσέγγιση που ονομάζεται GPGPU (υπολογισμός γενικής χρήσης για μονάδες επεξεργασίας γραφικών). Η πλατφόρμα CUDA είναι ένα επίπεδο λογισμικού που παρέχει άμεση πρόσβαση στο εικονικό σύνολο εντολών της GPU και στα παράλληλα υπολογιστικά στοιχεία για την εκτέλεση υπολογιστικών πυρήνων.

Η πλατφόρμα CUDA έχει σχεδιαστεί για να λειτουργεί με γλώσσες προγραμματισμού όπως C, C++ και Fortran. Αυτή η προσβασιμότητα διευκολύνει τους ειδικούς στον παράλληλο προγραμματισμό να χρησιμοποιούν πόρους GPU, σε αντίθεση με προηγούμενα API όπως το Direct3D και το OpenGL, τα οποία απαιτούσαν προηγμένες δεξιότητες στον προγραμματισμό γραφικών. Οι GPUs που υποστηρίζονται από το CUDA υποστηρίζουν επίσης πλαίσια προγραμματισμού, όπως OpenACC και OpenCL, και HIP με τη σύνταξη τέτοιου κώδικα στο CUDA.

4.6 TensorFlow

Το TensorFlow είναι μια ολοκληρωμένη πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Διαθέτει ένα ολοκληρωμένο, ευέλικτο οικοσύστημα εργαλείων, βιβλιοθηκών και κοινοτικών πόρων που επιτρέπει στους ερευνητές να προωθήσουν την τελευταία λέξη της τεχνολογίας στο ML και οι προγραμματιστές να κατασκευάζουν και να αναπτύσσουν εύκολα εφαρμογές που υποστηρίζονται από ML [64].

Το TensorFlow χρησιμοποιεί γραφήματα ροής δεδομένων για την αναπαράσταση υπολογισμού, κοινή κατάσταση και τις λειτουργίες που μεταλλάσσονται σε αυτήν την κατάσταση. Αντιστοιχίζει τους κόμβους ενός γραφήματος ροής δεδομένων σε

πολλούς υπολογιστές σε ένα σύμπλεγμα και μέσα σε ένα μηχάνημα σε πολλές υπολογιστικές συσκευές, συμπεριλαμβανομένων επεξεργαστών πολλαπλών πυρήνων, GPU γενικής χρήσης και ειδικά σχεδιασμένων ASICs που είναι γνωστές ως μονάδες επεξεργασίας τανυστών (TPUs). Το TensorFlow υποστηρίζει μια ποικιλία εφαρμογών, με έμφαση στην εκπαίδευση και τη δοκιμή σε βαθιά νευρωνικά δίκτυα. Αρκετές υπηρεσίες της Google χρησιμοποιούν TensorFlow στο παραγωγικό τους επίπεδο. Το TensorFlow κυκλοφορεί ως έργο ανοικτού κώδικα και χρησιμοποιείται ευρέως για την έρευνα στη μηχανική μάθηση. Το σύστημα είναι ευέλικτο και μπορεί να χρησιμοποιηθεί για μια μεγάλη ποικιλία αλγορίθμων, ιδιαίτερα για μοντέλα βαθιάς μάθησης νευρωνικών δικτύων, για τη διεξαγωγή έρευνας και για την ανάπτυξη συστημάτων μηχανικής μάθησης στην πληροφορική και σε άλλους τομείς, συμπεριλαμβανομένης της αναγνώρισης ομιλίας, της μηχανικής όρασης, της ρομποτικής, της ανάκτησης πληροφοριών, την επεξεργασία φυσικής γλώσσας (NLP), εξαγωγή γεωγραφικών πληροφοριών και υπολογιστική ανακάλυψη φαρμάκων.

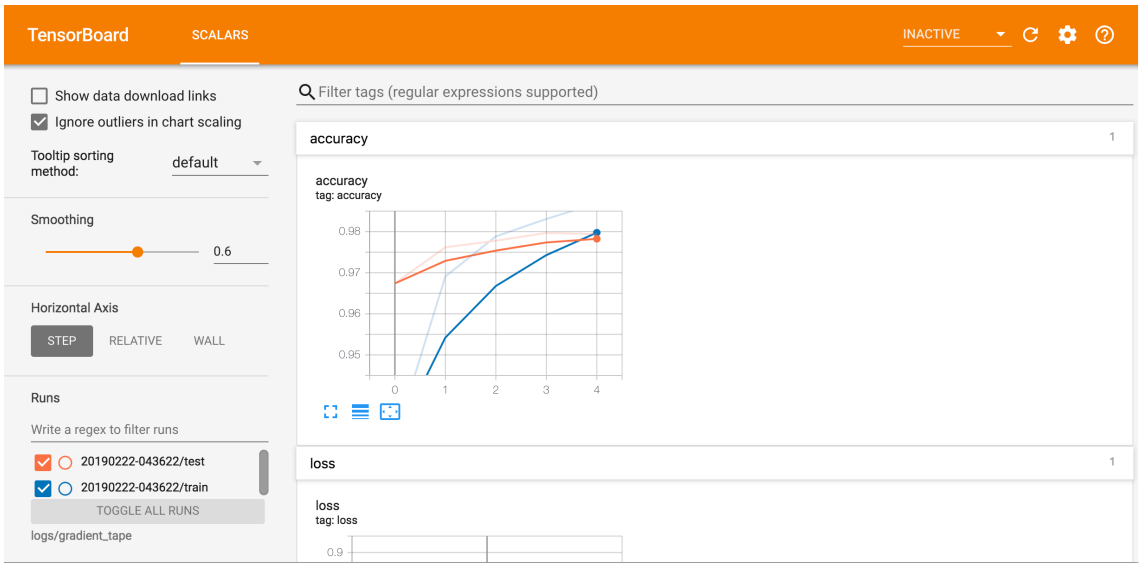
Το TensorFlow εμφανίζεται σε 2 βασικές εκδόσεις, την έκδοση TensorFlow 1.x και TensorFlow 2.x

4.7 TensorBoard

Στη μηχανική μάθηση, για να βελτιώσουμε κάτι, θα πρέπει να μπορούμε και να το μετρήσουμε. Το TensorBoard είναι ένα εργαλείο για την παροχή των μετρήσεων και των οπτικοποιήσεων που απαιτούνται κατά τη ροή εργασίας της μηχανικής μάθησης. Επιτρέπει την παρακολούθηση μετρήσεων πειράματος όπως απώλεια και ακρίβεια, οπτικοποίηση του γραφήματος μοντέλου, προβολή ενσωματώσεων σε χώρο χαμηλότερης διάστασης, εμφάνιση εικόνων, κειμένου και δεδομένων ήχου, προγραμματισμός προγραμμάτων TensorFlow και πολλά άλλα [65].
Στο σχήμα 4.3 παρουσιάζεται το περιβάλλον του λογισμικού TensorBoard.

4.8 Keras

Το Keras [66] είναι ένα API βαθιάς μάθησης γραμμένο στην Python, που τρέχει πάνω από την πλατφόρμα μηχανικής εκμάθησης TensorFlow. Είναι σχεδιασμένο για να επιτρέπει γρήγορο πειραματισμό με βαθιά νευ-



Σχήμα 4.3: Περιβάλλον TensorBoard για την αστικοποίηση της ακρίβειας του μοντέλου

ρωνικά δίκτυα και εστιάζει στο να είναι φιλικό προς τους χρήστες, διαμορφώσιμο και επεκτάσιμο. Το Keras περιέχει πολυάριθμες υλοποιήσεις συνηθισμένων δομικών μονάδων νευρωνικών δικτύων, όπως επίπεδα, στόχους, συναρτήσεις ενεργοποίησης, βελτιστοποιητές και πολλά εργαλεία για να διευκολύνει την εργασία με δεδομένα εικόνας και κειμένου και για να απλοποιήσει τη διαδικασία σύνταξης κώδικα για βαθιά νευρωνικά δίκτυα. Εκτός από τα τυπικά νευρωνικά δίκτυα, το Keras παρέχει υποστήριξη και για συνελικτικά και επαναλαμβανόμενα νευρωνικά δίκτυα. Υποστηρίζει και άλλα κοινά επίπεδα χρησιμότητας, όπως απόσυρση (dropout), ομαλοποίηση παρτίδων και χωρικής υποδειγματοληψίας. Επιπλέον, επιτρέπει την κατανεμημένη εκπαίδευση μοντέλων βαθιάς μάθησης σε κάρτες επεξεργασίας γραφικών (GPU) και μονάδων επεξεργασίας τανυστών (TPU).

4.9 PyTorch

Το PyTorch είναι μια βιβλιοθήκη για την Python που διευκολύνει την κατασκευή έργων βαθιάς μάθησης. Δίνει έμφαση στην ευελιξία και επιτρέπει στα μοντέλα βαθιάς μάθησης να εκφράζονται σε Python [67]. Αυτή η προσιτότητα και ευκολία χρήσης έκανε το PyTorch να εξελιχθεί σε ένα από τα πιο εξέχοντα εργαλεία βαθιάς μάθησης σε ευρύ φάσμα εφαρμογών. Το PyTorch είναι εύκολο να χρησιμοποιηθεί λόγω της απλότητας του. Πολλοί ερευνητές και επαγγελματίες το βρίσκουν εύκολο να μάθουν, να χρησιμοποιήσουν, να επεκτείνουν και να εντοπίσουν σφάλματα. Όπως και το TensorFlow, έτσι και το PyTorch παρέχει μια βασική δομή δεδομένων, τον τανυστή, ο οποίος είναι ένας πολυδιάστατος πίνακας που μοιράζεται πολλές ομοιότητες με τις συστοιχίες NumPy. Γύρω από αυτή τη θεμελίωση, το PyTorch έρχεται με δυνατότητες για την εκτέλεση επιταχυνόμενων μαθηματικών λειτουργιών σε ειδικό υλικό, το οποίο καθιστά βολικό το σχεδιασμό αρχιτεκτονικών νευρωνικών δικτύων και την εκπαίδευση αυτών σε μεμονωμένα μηχανήματα ή σε πόρους παράλληλου υπολογισμού.

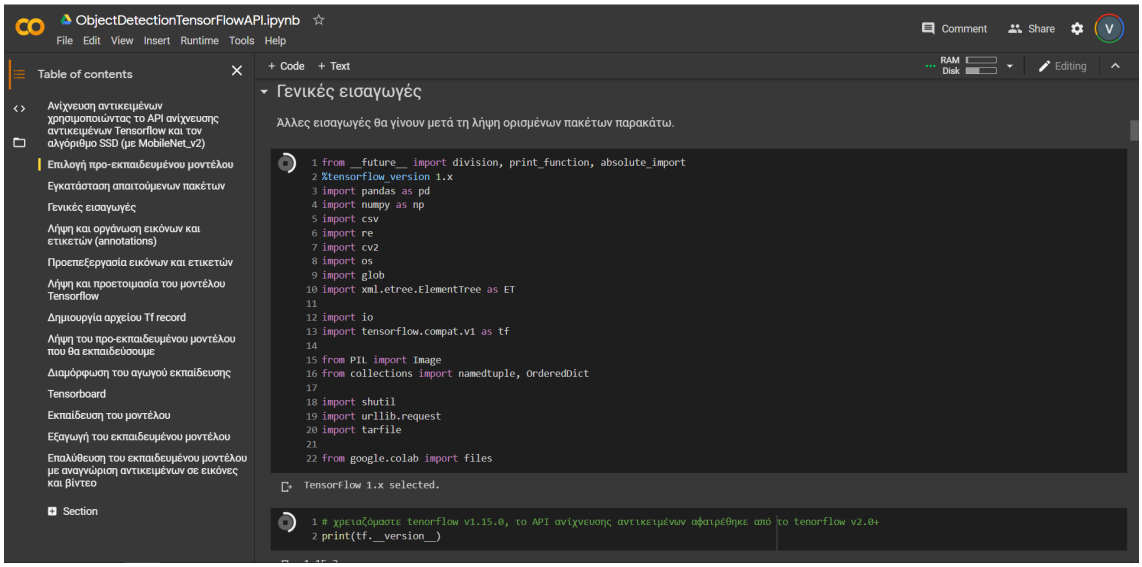
4.10 Darknet

Το Darknet είναι ένα ανοικτό πλαίσιο νευρωνικών δικτύων γραμμένο σε C και CUDA. Είναι γρήγορο, εύκολο να εγκατασταθεί, και υποστηρίζει υπολογισμό μέσω CPU και GPU. Δημιουργός του είναι ο Joseph Redmon, δημιουργός και του αλγορίθμου YOLO, και παρέχει build-in υποστήριξη για τον αλγόριθμο YOLO.

4.11 Google Colab

Το Colab είναι ένα δωρεάν περιβάλλον Jupyter notebooks που λειτουργεί εξ ολοκλήρου στο cloud. Τα σημειωματάρια Colab επιτρέπουν να συνδυαστεί εκτελέσιμος κώδικας και κείμενο σε ένα μόνο έγγραφο, μαζί με εικόνες, HTML, LaTeX και άλλα. Το πιο σημαντικό είναι ότι δεν απαιτεί ρυθμίσεις και υποστηρίζει πολλές δημοφιλείς βιβλιοθήκες μηχανικής μάθησης (PyTorch, TensorFlow, Keras), οι οποίες μπορούν να φορτωθούν εύκολα στα σημειωματάρια.

Επίσης, ιδιαίτερα σημαντικό χαρακτηριστικό του Colab είναι οτι παρέχει πρόσβαση σε GPU και TPU servers δωρεάν και έτσι επιταχύνονται οι εκπαιδεύσεις των νευρωνικών δικτύων.



Σχήμα 4.4: Εκτέλεση σημειωματάριου εκπαίδευσης αλγορίθμου SSD στο Google Colab.

Κεφάλαιο 5

Υπολογιστική μελέτη

Σε αυτό το κεφάλαιο θα μελετήσουμε την εκπαίδευση μοντέλων ανίχνευσης αντικειμένων με τους αλγορίθμους Faster R-CNN, RetinaNet, SSD, EfficientDet, YOLOv3 και YOLOv4. Ο ανιχνευτής που θα αναπτύξουμε θα μπορεί να εντοπίζει όπλα σε εικόνες και σε βίντεο αλλά και σε πραγματικό χρόνο. Τα βήματα είναι τα ακόλουθα. Αρχικά, θα επιλέξουμε ένα σύνολο δεδομένων με εικόνες που περιέχουν τα αντικείμενά μας αλλά και επισημάνσεις με τις συντεταγμένες που εμφανίζεται το κάθε αντικείμενο στην κάθε εικόνα. Στη συνέχεια, θα δώσουμε το σύνολο δεδομένων μας σαν είσοδο στους αλγόριθμους και θα ρυθμίσουμε τις παραμέτρους εκπαίδευσης του εκάστοτε αλγορίθμου. Μετά το τέλος της εκπαίδευσης, θα αποθηκεύσουμε τα εκπαιδευμένα μοντέλα μας και θα αξιολογήσουμε την αποτελεσματικότητά τους.

Αναλυτικές οδηγίες για την εγκατάσταση των λογισμικών και τη λήψη των απαραίτητων δεδομένων συμπεριλαμβάνονται μαζί με τον κώδικα της διπλωματικής εργασίας στο Github.

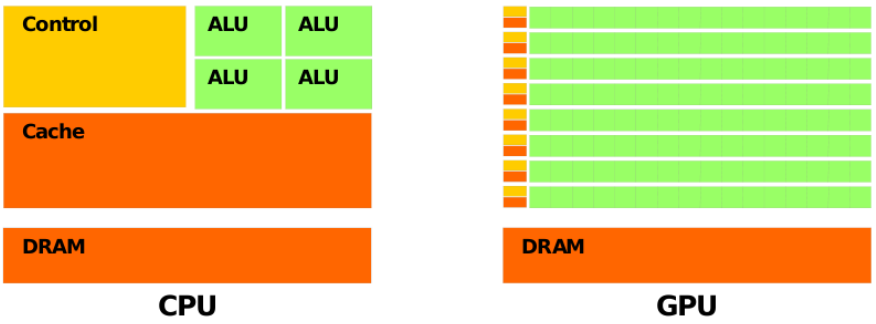
Πριν ξεκινήσουμε την εκπαίδευση του ανιχνευτή μας, ιδιαίτερα σημαντικό είναι να επισημάνουμε τις απαιτήσεις συστήματος σε υλικό κομμάτι (hardware) για την εκπαίδευση των μοντέλων.

5.1 Απαιτήσεις συστήματος

Στο κομμάτι της εκπαίδευσης του μοντέλου, το οποίο γενικά απαιτεί μεγάλη υπολογιστική ισχύ, η διαδικασία θα μπορούσε να είναι απογοητευτική εάν γίνει χωρίς το σωστό υλικό. Αυτό το εντατικό μέρος του νευρωνικού δικτύου αποτελείται από διάφορους πολλαπλασιασμούς πινάκων.

Αυτό μπορεί να επιτευχθεί απλά εκτελώντας όλες τις λειτουργίες ταυτόχρονα,

αντί να τις εκτελούμε σειριακά. Εδώ έρχεται η GPU, με χιλιάδες πυρήνες σχεδιασμένους να υπολογίζουν με σχεδόν 100% απόδοση. Αποδεικνύεται ότι αυτοί οι πυρήνες είναι κατάλληλοι για να εκτελέσουν τους υπολογισμούς που απαιτούνται από τα νευρωνικά δίκτυα.



Σχήμα 5.1: Δομή ενός CPU και ενός GPU [29].

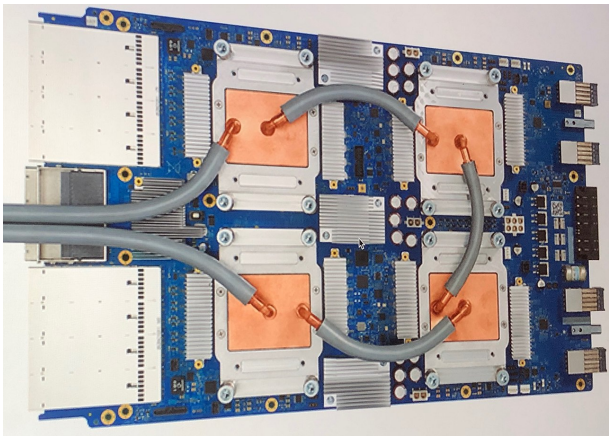
Ο αγώνας μεταξύ CPU και GPU ευνοεί τους τελευταίους λόγω του μεγάλου αριθμού πυρήνων GPU που αντισταθμίζουν τη μεγαλύτερη ταχύτητα 4-8 πυρήνων CPU έναντι των περίπου 3500 πυρήνων (GPU). Οι πυρήνες GPU είναι μια απλοποιημένη έκδοση των πιο περίπλοκων πυρήνων CPU, αλλά έχοντας τόσα πολλά από αυτά επιτρέπει στις GPU να έχουν υψηλότερο επίπεδο παραλληλισμού και επομένως καλύτερη απόδοση.

Οι GPUs έχουν σχεδιαστεί για τη δημιουργία γραφικών υπολογιστών που βασίζονται σε πολύγωνα. Τα τελευταία χρόνια, λόγω της ανάγκης για ρεαλισμό στα πρόσφατα ηλεκτρονικά παιχνίδια οι GPUs έχουν συσσωρεύσει μεγάλες δυνατότητες επεξεργασίας. Η GPU είναι μια παράλληλη ρύθμιση προγραμματισμού που περιλαμβάνει GPU και CPU που μπορούν να επεξεργαστούν και να αναλύσουν δεδομένα με παρόμοιο τρόπο όπως μια εικόνα ή οποιαδήποτε άλλη μορφή γραφικών. Οι GPUs δημιουργήθηκαν για καλύτερη επεξεργασία γραφικών, αλλά αργότερα διαπιστώθηκε ότι ταιριάζουν καλά στην επιστημονική πληροφορική.

Επιπρόσθετα, η εκπαίδευση ενός νευρωνικού δικτύου μπορεί να επιτευχθεί και με τη χρήση ενός TPU (Tensor Proccessing Unit). Πρόκειται για ένα ολοκληρωμένο κύκλωμα (ASIC) ειδικά για την εφαρμογή σε επιταχύνσεις AI το οποίο αναπτύχθηκε από την Google ειδικά για τη μηχανική εκμάθηση νευρωνικών δικτύων. Είναι μια από τις πιο προηγμένες πλατφόρμες εκπαίδευσης σε βάθος. Το TPU προσφέρει ώθηση απόδοσης 15-30x σε σχέση με τους σύγχρονους CPU και GPU και με υψηλότερη

αναλογία απόδοσης ανά watt 30-80x. Το TPU υποστηρίζεται από το λογισμικό TensorFlow.

Σχήμα 5.2: GPU



Σχήμα 5.3: Tensor Processing Unit 3.0 [30].

Για την εκπαίδευση των δικών μας νευρωνικών δικτύων θα χρησιμοποιήσουμε αποκλειστικά GPU.

Τα χαρακτηριστικά του βασικού συστήματος που θα αξιοποιηθεί για την ανάπτυξη και την εκπαίδευση των ανιχνευτών αναφέρονται στο πίνακα 5.1.

CPU	i7-7820X CPU @ 3.60GHz (16 Cores)
GPU	GeForce GTX 1080 Ti 11 GB
RAM	32 GB

Πίνακας 5.1: Υλικό μέρος συστήματος εργασίας.

Επιπλέον, για την επιτάχυνση των αλγορίθμων μέσω GPU χρησιμοποιήθηκε η έκδοση του CUDA Toolkit 11.

5.2 Δημιουργία συνόλου δεδομένων εικόνων για εκπαίδευση

Το πιο σημαντικό βήμα για την αποτελεσματική εκπαίδευση ενός μοντέλου είναι η σωστή δημιουργία ενός συνόλου δεδομένων για εκπαίδευση (train), ενός συνόλου δεδομένων για τεστ (test) και ενός συνόλου δεδομένων για επαλήθευση (validation).

5.2.1 Διαθέσιμα σύνολα δεδομένων για εκπαίδευση

Υπάρχουν διαθέσιμα μεγάλα σύνολα δεδομένων από τα οποία μπορούμε να βρούμε εικόνες για τα αντικείμενα για τα οποία θέλουμε να εκπαιδεύσουμε το μοντέλο. Τα πιο γνωστά σύνολα δεδομένων είναι τα PASCAL VOC, COCO Dataset και

το Google Open Images Dataset.

PASCAL Visual Object Classes

Το σύνολο δεδομένων PASCAL Visual Object Classes (VOC) παρέχει τυποποιημένα σύνολα δεδομένων εικόνας για αναγνώριση κλάσης αντικειμένων και επίσης παρέχει ένα κοινό σύνολο εργαλείων για την πρόσβαση στα σύνολα δεδομένων και τους σχολιασμούς και αποτελεί σημείο αναφοράς στην αναγνώριση και την ανίχνευση οπτικών αντικειμένων [68]. Το σύνολο δεδομένων PASCAL VOC έχει γίνει αποδεκτό ως σημείο αναφοράς για την ανίχνευση αντικειμένων.

COCO

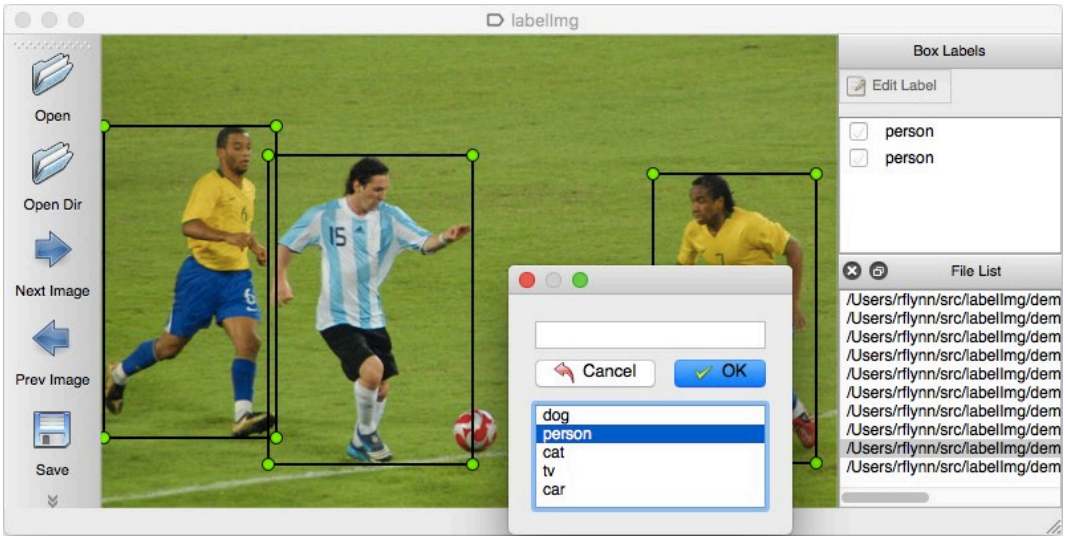
Το σύνολο δεδομένων COCO (Common Objects in Context) αποτελεί ένα νέο σύνολο δεδομένων με στόχο την πρόοδο της τελευταίας τεχνολογίας στην αναγνώριση αντικειμένων τοποθετώντας το ζήτημα της αναγνώρισης αντικειμένου στο πλαίσιο του ευρύτερου ζητήματος της κατανόησης της σκηνής. Αυτό επιτυγχάνεται με τη συλλογή σύνθετων εικόνων (330 χιλιάδες εικόνες, με ετικέτα περισσότερες από 200 χιλιάδες) καθημερινών σκηνών που περιέχουν κοινά αντικείμενα στο φυσικό τους περιβάλλον [69]. Τα αντικείμενα επισημαίνονται χρησιμοποιώντας τμηματοποιήσεις ανά εμφάνιση για να βοηθήσουν τον ακριβή εντοπισμό αντικειμένων. Το σύνολο δεδομένων μας περιέχει φωτογραφίες από 91 τύπους αντικειμένων που θα μπορούσαν να αναγνωριστούν εύκολα από ένα 4χρονο παιδί.

Open Images Dataset

Πρόκειται για ένα σύνολο περίπου 9 εκατομμυρίων εικόνων εμπλουτισμένο με επισημάνσεις των αντικειμένων που βρίσκονται στην κάθε εικόνα [70]. Οι εικόνες είναι πολύ διαφορετικές και συχνά περιέχουν πολύπλοκες σκηνές με πολλά αντικείμενα (8,4 ανά εικόνα κατά μέσο όρο). Περιλαμβάνει σχολιασμούς ετικετών σε επίπεδο εικόνας, πλαίσια οριοθέτησης αντικειμένων, τμηματοποίηση αντικειμένων, οπτικές σχέσεις, τοπικές αφηγήσεις και άλλα.

Προσαρμοσμένο σύνολο δεδομένων

Τέλος, μπορούμε και μόνοι μας να δημιουργήσουμε ένα σύνολο εικόνων για εκπαίδευση, προσθέτοντας για κάθε κλάση αντικειμένων αρκετές εικόνες για εκπαίδευση και για επαλήθευση. Μια καλή αναλογία είναι 80% για εκπαίδευση, 10% για επαλήθευση και 10% για τεστ. Πολύ σημαντικό στη δημιουργία ενός συνόλου δεδομένων για ανίχνευση αντικειμένων είναι ότι πρέπει να δημιουργηθούν "σχόλια" επί των εικόνων. Για αυτήν τη διαδικασία μπορούν να χρησιμοποιηθούν λογισμικά όπως τα LabelImg, MakeSense.AI, VGG Image Annotator καθώς και το λογισμικό Yolo Mark που μας επιτρέπει τη σήμανση οριοθετημένων κουτιών αντικειμένων στις εικόνες απευθείας για την εκπαίδευση του αλγορίθμου YOLO. Στο σχήμα 5.4 παρουσιάζεται το περιβάλλον του λογισμικού LabelImg.



Σχήμα 5.4: Λογισμικό LabelImg για σχόλια των αντικειμένων επί των εικόνων.

5.2.2 Προγραμματιστικός τρόπος απόκτησης εικόνων από το Open Images Dataset

Στη συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε τον προγραμματιστικό τρόπο για τη δημιουργία του dataset για την εκπαίδευση χρησιμοποιώντας το εργαλείο OIDv4 ToolKit. Πρόκειται για ένα εργαλείο γραμμένο σε Python3 το οποίο μας δίνει τη δυνατότητα να κατεβάσουμε αυτόματα τις εικόνες μαζί με τους σχολιασμούς τους με τα σχετικά κουτιά οριοθέτησης από το Open Image Dataset της Google [71]. Το εργαλείο περιέχεται μέσα στον κώδικα της διπλωματικής εργασίας στο φάκελο OID.

Για την εκκίνηση της αυτοματοποιημένης λήψης των εικόνων από το OID εκτελούμε την παρακάτω εντολή, προσθέτοντας στην παράμετρο *classes* όλες τις κατηγορίες αντικειμένων που γι αυτές θα θέλαμε να κατεβάσουμε τις εικόνες τους μαζί με τους σχολιασμούς τους.

```
python main.py downloader -y --classes 'Handgun' --type_csv train --limit 500
```

Με την παράμετρο type_csv ορίζουμε τον τύπο του dataset που θέλουμε να κατεβάσουμε. Αν, δηλαδή, θα προορίζεται για εκπαίδευση (train), για έλεγχο (test) ή για επαλήθευση (validation).

Με την παράμετρο limit ορίζουμε μέχρι πόσες εικόνες να κατεβάσει συνολικά από κάθε κλάση αντικειμένων. Έτσι για να κατεβάσουμε το test dataset και το validation dataset τρέχουμε και τις παρακάτω εντολές.

```
python main.py downloader -y --classes 'Handgun' --type_csv validation
python main.py downloader -y --classes 'Handgun' --type_csv test
```

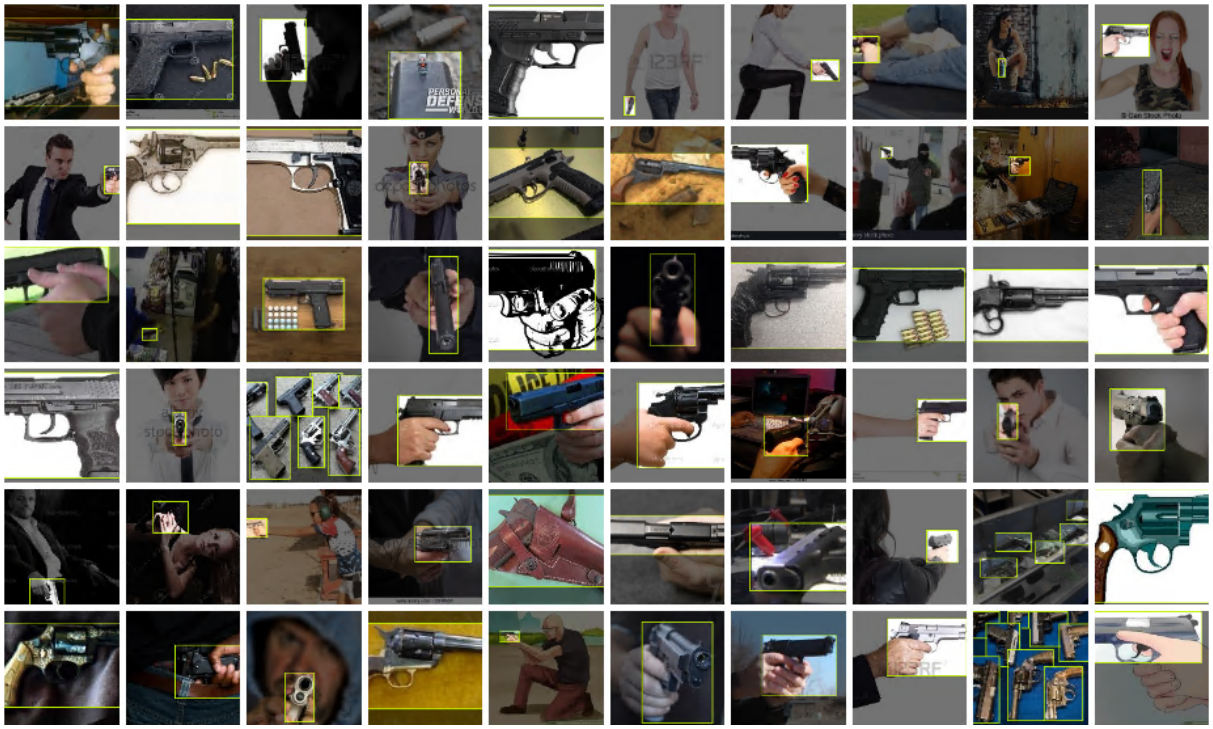
Σχήμα 5.5: Λήψη εικόνων μαζί με τους σχολιασμούς για εκπαίδευση μέσω του εργαλείου OIDv4 ToolKit.

5.2.3 Το σύνολο δεδομένων της εργασίας

Όπως προαναφέρθηκε, στην παρούσα διπλωματική εργασία θα εκπαιδεύσουμε και θα συγκρίνουμε ανιχνευτές αντικειμένων οι οποίοι θα μπορούν να εντοπίζουν όπλα τόσο σε εικόνες και βίντεο όσο και σε πραγματικό χρόνο μέσα από βίντεο από κάμερες ασφαλείας. Γι αυτό τον λόγο θα χρειαστούμε αλγορίθμους οι οποίοι να μπορούν να ανιχνεύουν αντικείμενα με ταχύτητα μεγαλύτερη από 30 FPS. Φυσικά, για να επιτευχθεί αυτό, πρωταρχικό ρόλο παίζει το σύνολο δεδομένων που θα δημιουργήσουμε ή θα χρησιμοποιήσουμε πάνω στο οποίο θα εκπαιδευτούν τα συνελικτικά νευρωνικά δίκτυα των αλγορίθμων. Το σύνολο δεδομένων που θα αξιοποιηθεί σε αυτήν τη διπλωματική εργασία είναι το Pistols Dataset από το Πανεπιστήμιο της Γρενάδας [72].

Αυτό το σύνολο δεδομένων περιέχει 2986 εικόνες και 3448 ετικέτες σε μία μόνο κατηγορία αντικειμένων: πιστόλι (pistol). Οι εικόνες είναι ευρείας εμβέλειας: πιστόλια στο χέρι, κινούμενα σχέδια και εικόνες ποιότητας όπλων σε στούντιο.

Στο σήμα 5.6 παρουσιάζεται η οπτικοποίηση του συνόλου δεδομένων (εικόνες μαζί με τα πλαίσια οριοθέτησης των αντικειμένων που μας ενδιαφέρουν) το οποίο θα χρησιμοποιηθεί στην εκπαίδευση των αλγορίθμων της διπλωματικής εργασίας



Σχήμα 5.6: Οπτικοποίηση του συνόλου δεδομένων που θα χρησιμοποιηθεί στην εκπαίδευση των αλγορίθμων. Τα κίτρινα πλαίσια αποτελούν τα πλαίσια οριοθέτησης των αντικειμένων του συνόλου δεδομένων.

5.3 Αλγόριθμοι και πλαίσια εργασίας

Τα δύο βασικά πλαίσια εργασίας (frameworks) που θα αξιοποιηθούν για την υπολογιστική μελέτη αυτής της διπλωματικής εργασίας είναι το TensorFlow 2 Object Detection API και το Darknet. Το TensorFlow 2 Object Detection API αποτελεί το πλαίσιο για τη δημιουργία ενός δικτύου βαθιάς μάθησης που επιλύει προβλήματα ανίχνευσης αντικειμένων [73]. Υπάρχουν προδιαμορφωμένα μοντέλα για τους αλγορίθμους ανίχνευσης αντικειμένων τα οποία είναι προ-εκπαιδευμένα στο σύνολο δεδομένων COCO. Από την άλλη πλευρά το Darknet [74] είναι ένα πλαίσιο εργασίας με τον οποίο έχουν υλοποιηθεί οι αλγόριθμοι YOLO.

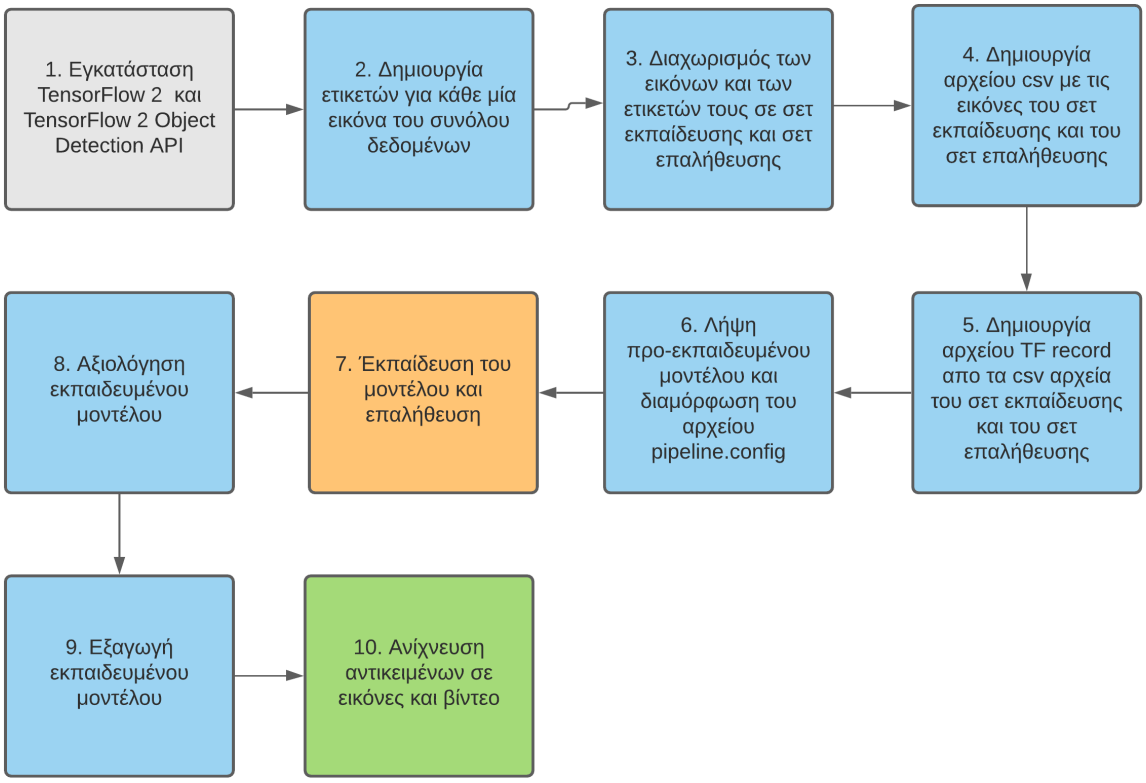
Στον πίνακα 5.2 αναφέρεται το framework που χρησιμοποιήθηκε για τον εκάστοτε αλγόριθμο της διπλωματικής εργασίας.

Αλγόριθμος	**Backbone**	**Framework**
Faster R-CNN	Resnet50	TensorFlow 2 Object Detection API
SSD	MobileNet v2	TensorFlow 2 Object Detection API
RetinaNet	Resnet50	TensorFlow 2 Object Detection API
EfficientDet	EfficientNet	TensorFlow 2 Object Detection API
YOLOv3	Darknet-53	Darknet
YOLOv4	CSPDarknet53	Darknet

Πίνακας 5.2: Framework που χρησιμοποιήθηκε για τον κάθε αλγόριθμο.

5.4 Εκπαίδευση ανιχνευτή αντικειμένων με TensorFlow 2 Object Detection API

Σε αυτήν την ενότητα της διπλωματικής εργασίας, θα χρησιμοποιήσουμε το TensorFlow 2 Object Detection API για την ανάπτυξη και εκπαίδευση ανιχνευτή αντικειμένων με τη χρήση των αλγορίθμων Faster R-CNN, SSD, RetinaNet και EfficientDet. Προκειμένου να προσαρμόσουμε το TensorFlow 2 Object Detection API στο τρέχον πρόβλημα, είναι σημαντικό να καθοριστεί η δομή του API, καθώς και ο τρόπος χειρισμού των δεδομένων από αυτό και όλες οι απαραίτητες τροποποιήσεις που πρέπει να γίνουν. Η διαδικασία της ανίχνευσης αντικειμένων αποτελείται από τα μπλοκ που απεικονίζονται στο σχήμα 5.7.



Σχήμα 5.7: Διάγραμμα της διαδικασίας ανάπτυξης ενός ανιχνευτή αντικειμένου με το Tensorflow 2 Object Detection API.

Καθώς το Tensorflow 2 Object Detection API υποστηρίζει πολλαπλά μοντέλα και αλγορίθμους, θα το χρησιμοποιήσουμε για να εκπαιδεύσουμε και να αξιολογήσουμε ένα μοντέλο, το οποίο θα μπορεί να ανιχνεύει όπλα σε εικόνες, βίντεο και σε πραγματικό χρόνο από κάμερες ασφαλείας. Με άλλα λόγια, το μοντέλο που θα

εκπαιδεύσουμε με τους αλγορίθμους Faster R-CNN, SSD, RetinaNet και EfficientDet θα μπορεί να εντοπίζει την κλάση αντικειμένων pistol.

Απεικόνιση 5.1: Ορισμός μοντέλου που θα εκπαιδεύσουμε με το TensorFlow.

```
# Models
MODELS_CONFIG = {
    'ssd_mobilenet_v2': {
        'model_name': 'ssd_mobilenet_v1_fpn_640x640_coco17_tpu-8',
        'model_path': '/models/tf2/ssd_mobilenet_v2_fpnlite_640x640/',
        'pipeline_file': 'pipeline.config'
    },
    'faster_rcnn_resnet50_v1': {
        'model_name': 'faster_rcnn_resnet50_v1_640x640_coco17_tpu-8',
        'model_path': '/models/tf2/faster_rcnn_resnet50_v1/',
        'pipeline_file': 'pipeline.config'
    },
      'retinanet_resnet50_v1_fpn': {
        'model_name': 'ssd_resnet50_v1_fpn_640x640_coco17_tpu-8',
        'model_path': '/models/tf2/ssd_resnet50_v1_fpn/',
        'pipeline_file': 'pipeline.config'
    },
      'efficientdet_d0': {
        'model_name': 'efficientdet_d0_coco17_tpu-32',
        'model_path': '/models/tf2/efficientdet_d0/',
        'pipeline_file': 'pipeline.config'
    }
}

# Select a model to use.
selected_model = 'faster_rcnn_resnet50_v1'
```

Στις μεταβλητές της απεικόνισης 5.2 αναθέτουμε τις διαδρομές όλων των βασικών αρχείων ρυθμίσεων και δεδομένων που θα ληφθούν ως είσοδος από το TensorFlow 2 Object Detection API.

Απεικόνιση 5.2: Ορισμός των διαδρομών των αρχείων δεδομένων και ρυθμίσεων για το TensorFlow.

```
model_name = MODELS_CONFIG[selected_model]['model_name']
model_path = MODELS_CONFIG[selected_model]['model_path']
pipeline_file = MODELS_CONFIG[selected_model]['pipeline_file']

# Set Repository Home Directory
working_directory = os.getcwd()

# Set Label Map (.pbtxt) path and pipeline.config path
label_map_pbtxt_fname = working_directory + '/annotations/label_map.pbtxt'
pipeline_fname = working_directory + model_path + pipeline_file

# Set .record path
test_record_fname = working_directory + '/annotations/test.record'
train_record_fname = working_directory + '/annotations/train.record'

# Set output directories and clean up
model_dir = working_directory + '/' + selected_model + '-training/'
output_dir = working_directory + '/' + selected_model + '-exported-models/'

!rm -rf {model_dir} {output_dir}
os.makedirs(model_dir, exist_ok=True)
os.makedirs(output_dir, exist_ok=True)
```

Όπως αναφέρθηκε προηγουμένως, χρησιμοποιούμε προ-εκπαιδευμένα μοντέλα που προσφέρει το Tensorflow, τα οποία πρέπει να προσαρμοστούν στα διαφορετικά σύνολα δεδομένων που λαμβάνει το συνελικτικό νευρωνικό δίκτυο του κάθε αλγορίθμου.

Το πρώτο βήμα της παραμετροποίησης είναι η δημιουργία του αρχείου του χάρτη ετικετών (label map) σε μορφή JSON, ο οποίος αντιστοιχεί συγκεκριμένα σε καθεμία από τις χρησιμοποιημένες ετικέτες σε ακέραιες τιμές. Αυτός ο χάρτης ετικετών χρησιμοποιείται τόσο από την εκπαίδευση όσο και από τις διαδικασίες ανίχνευσης.

Στη συγκεκριμένη μελέτη έχουμε μόνο μια κλάση αντικειμένων η οποία ονο-

μάζεται pistol και ως εκ τούτου, ο χάρτης ετικετών μας θα έχει τη μορφή που παρουσιάζεται στην απεικόνιση 5.3.

Απεικόνιση 5.3: Αρχείο χάρτη ετικετών label_map.pbtxt

```
item {
   id: 1
   name: 'pistol'
}
```

Το επόμενο βήμα της διαμόρφωσης της εκπαίδευσης είναι η παραμετροποίηση του αρχείου *pipeline.config* το οποίο περιέχει όλες τις ρυθμίσεις σχετικά με το μοντέλο που εκπαιδεύουμε αλλά και για το σύνολο δεδομένων. Το αρχείο pipeline.config είναι και αυτό σε μορφή JSON και οι παραμετροποιήσεις που θα κάνουμε για την εκπαίδευση του αλγορίθμου Faster R-CNN είναι οι παρακάτω.

1. **num_classes**: Ο συνολικός αριθμός των κλάσεων αντικειμένων που θα ανιχνεύει ο ανιχνευτής μας.

2. **type**: Το όνομα του προ-εκπαιδευμένου μοντέλου που κατεβάσαμε και θα χρησιμοποιήσουμε στην εκπαίδευση.

3. **batch_size**: Καθορίζει τον αριθμό των δειγμάτων που θα διαδοθούν μέσω του δικτύου σε μια επανάληψη. Στη δική μας περίπτωση το batch size είναι 8.

4. **fine_tune_checkpoint**: Διαδρομή στο checkpoint του προ-εκπαιδευμένου μοντέλου που θα κατέβει απο το TensorFlow Model Zoo με τον κώδικα της απεικόνισης 5.4.

5. **(train_input_reader) input_path**: Διαδρομή του αρχείου TFRecord της εκπαίδευσης.

6. **(train_input_reader) label_map_path**: Διαδρομή του αρχείου του χάρτη ετικετών (label map).

7. **(eval_input_reader) input_path**: Διαδρομή του αρχείου TFRecord της επαλήθευσης.

8. **(eval_input_reader) label_map_path**: Διαδρομή του αρχείου του χάρτη ετικετών (label map).

Απεικόνιση 5.4: Λήψη του προ-εκπαιδευμένου μοντέλου από το TensorFlow Model Zoo.

```
import os
import shutil
import glob
import urllib.request
import tarfile
MODEL_FILE = model_name + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/tf2/202
    00711/'
DEST_DIR = '/content/models/research/pretrained_model'

if not (os.path.exists(MODEL_FILE)):
    urllib.request.urlretrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)

tar = tarfile.open(MODEL_FILE)
tar.extractall()
tar.close()

# Set fine tune checkpoint
fine_tune_checkpoint = os.path.join(DEST_DIR, "checkpoint/ckpt-0")
print("fine_tune_checkpoint: ", fine_tune_checkpoint)
```

Σε επόμενο βήμα διαχωρίζουμε το σύνολο δεδομένων μας σε σετ εκπαίδευσης (90%) και σετ δοκιμών (10%).

Απεικόνιση 5.5: Διαχωρισμός σε σετ εκπαίδευσης και σετ δοκιμών.

```
# Split images to train:test = 9:1
!python scripts/partition_dataset.py -x -i images/ -r 0.1
```

Μετά το τέλος της διαδικασίας του διαχωρισμού του συνόλου δεδομένων, κάτω από το φάκελο images, θα δημιουργηθεί ένας υποφάκελος με το όνομα train, ο οποίος περιέχει τις εικόνες και τις ετικέτες που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου και ένας υποφάκελος με το valid, ο οποίος περιέχει τις εικόνες και τις ετικέτες που θα χρησιμοποιηθούν για την επαλήθευση του μοντέλου μετά

από κάθε εποχή εκπαίδευσης.

Καθώς το σύνολο δεδομένων που χρησιμοποιούμε, έχει τις ετικέτες των αντικειμένων σε μορφή PASCAL VOC XML θα χρειαστεί να τις μετατρέψουμε αρχικά σε CSV και στη συνέχεια σε TF records για να μπορέσει το TensorFlow να τα διαβάσει.

Απεικόνιση 5.6: Μετατροπή ετικετών xml σε csv.

```
# Generate train data:
!python scripts/xml_to_csv.py -i images/train -o annotations/train_labels.csv
# Generate test data:
!python scripts/xml_to_csv.py -i images/test -o annotations/test_labels.csv
```

Για να μπορέσουμε να εισάγουμε τις εικόνες μας στο API, οι εικόνες εισαγωγής και οι σχολιασμοί τους πρέπει να μετατραπούν σε αναγνώσιμη μορφή από το framework που ονομάζεται Tensorflow records (TFRecords). Αυτή η μορφή είναι συγκεκριμένη για το framework και αποθηκεύει δεδομένα ως ακολουθία δυαδικών συμβολοσειρών. Εκτός αυτού, τα δυαδικά δεδομένα μειώνουν τον χώρο των αρχικών δεδομένων και μπορούν να διαβαστούν πιο αποτελεσματικά.

Απεικόνιση 5.7: Μετατροπή σε μορφή TFRecord (.record).

```
# Generate train data:
!python scripts/generate_tfrecord.py -c annotations/train_labels.csv -i images/
    train -x images/train -o annotations/train.record -l annotations/label_map.
    pbtxt

# Generate test data:
!python scripts/generate_tfrecord.py -c annotations/test_labels.csv -i images/
    test -x images/test -o annotations/test.record -l annotations/label_map.
    pbtxt
```

Η διαδικασία προετοιμασίας των δεδομένων και της παραμετροποίησης έχει ολοκληρωθεί. Εκτελούμε την εντολή στην απεικόνιση 5.8 για την έναρξη της εκπαίδευσης του μοντέλου.

Απεικόνιση 5.8: Έναρξη της εκπαίδευσης.

```
%cd {working_directory}
!python /content/models/research/object_detection/model_main_tf2.py \
    --pipeline_config_path={pipeline_fname} \
    --model_dir={model_dir} \
    --alsologtostderr
```

Η εκπαίδευση και η επαλήθευση εκτελούνται ταυτόχρονα. Πριν από τη δοκιμή του ανιχνευτή, αυτό που είναι γνωστό ως γράφημα συμπερασμάτων εξάγεται και χρησιμοποιείται για την εκτέλεση ανίχνευσης αντικειμένου στο μέρος δοκιμής. Η δοκιμή πραγματοποιείται επιλέγοντας νέα σκορ διαφορετικά από αυτά που χρησιμοποιούνται για εκπαίδευση και επικύρωση. Η έξοδος αντικατοπτρίζει τα προβλεπόμενα οριακά πλαίσια οριοθέτησης από το νευρωνικό δίκτυο μαζί με την ετικέτα και τη βαθμολογία εμπιστοσύνης. Η εκπαίδευση τρέχει για 25.000 βήματα για τον κάθε αλγόριθμο.

```
INFO:tensorflow:Step 100 per-step time 0.317s loss=0.944
I1220 20:09:03.294398 140417237059392 model_lib_v2.py:648] Step 100 per-step time 0.317s loss=0.944
INFO:tensorflow:Step 200 per-step time 0.336s loss=0.559
I1220 20:09:35.946161 140417237059392 model_lib_v2.py:648] Step 200 per-step time 0.336s loss=0.559
INFO:tensorflow:Step 300 per-step time 0.326s loss=0.653
I1220 20:10:08.629013 140417237059392 model_lib_v2.py:648] Step 300 per-step time 0.326s loss=0.653
INFO:tensorflow:Step 400 per-step time 0.322s loss=0.460
I1220 20:10:41.244882 140417237059392 model_lib_v2.py:648] Step 400 per-step time 0.322s loss=0.460
INFO:tensorflow:Step 500 per-step time 0.331s loss=0.939
I1220 20:11:17.011125 140417237059392 model_lib_v2.py:648] Step 500 per-step time 0.331s loss=0.939
INFO:tensorflow:Step 600 per-step time 0.339s loss=0.569
I1220 20:11:49.807373 140417237059392 model_lib_v2.py:648] Step 600 per-step time 0.339s loss=0.569
INFO:tensorflow:Step 700 per-step time 0.324s loss=0.996
I1220 20:12:24.534740 140417237059392 model_lib_v2.py:648] Step 700 per-step time 0.324s loss=0.996
INFO:tensorflow:Step 800 per-step time 0.336s loss=0.802
I1220 20:12:57.456956 140417237059392 model_lib_v2.py:648] Step 800 per-step time 0.336s loss=0.802
INFO:tensorflow:Step 900 per-step time 0.345s loss=0.925
I1220 20:13:30.173893 140417237059392 model_lib_v2.py:648] Step 900 per-step time 0.345s loss=0.925
INFO:tensorflow:Step 1000 per-step time 0.320s loss=0.861
I1220 20:14:12.102819 140417237059392 model_lib_v2.py:648] Step 1000 per-step time 0.320s loss=0.861
INFO:tensorflow:Step 1100 per-step time 0.332s loss=1.101
I1220 20:15:06.415317 140417237059392 model_lib_v2.py:648] Step 1100 per-step time 0.332s loss=1.101
INFO:tensorflow:Step 1200 per-step time 0.336s loss=0.882
I1220 20:15:39.153424 140417237059392 model_lib_v2.py:648] Step 1200 per-step time 0.336s loss=0.882
INFO:tensorflow:Step 1300 per-step time 0.334s loss=0.682
I1220 20:16:12.192382 140417237059392 model_lib_v2.py:648] Step 1300 per-step time 0.334s loss=0.682
INFO:tensorflow:Step 1400 per-step time 0.317s loss=1.235
I1220 20:16:46.718531 140417237059392 model_lib_v2.py:648] Step 1400 per-step time 0.317s loss=1.235
INFO:tensorflow:Step 1500 per-step time 0.318s loss=0.530
I1220 20:17:19.490742 140417237059392 model_lib_v2.py:648] Step 1500 per-step time 0.318s loss=0.530
INFO:tensorflow:Step 1600 per-step time 0.330s loss=0.585
I1220 20:17:52.187473 140417237059392 model_lib_v2.py:648] Step 1600 per-step time 0.330s loss=0.585
INFO:tensorflow:Step 1700 per-step time 0.309s loss=0.380
I1220 20:18:26.063700 140417237059392 model_lib_v2.py:648] Step 1700 per-step time 0.309s loss=0.380
```

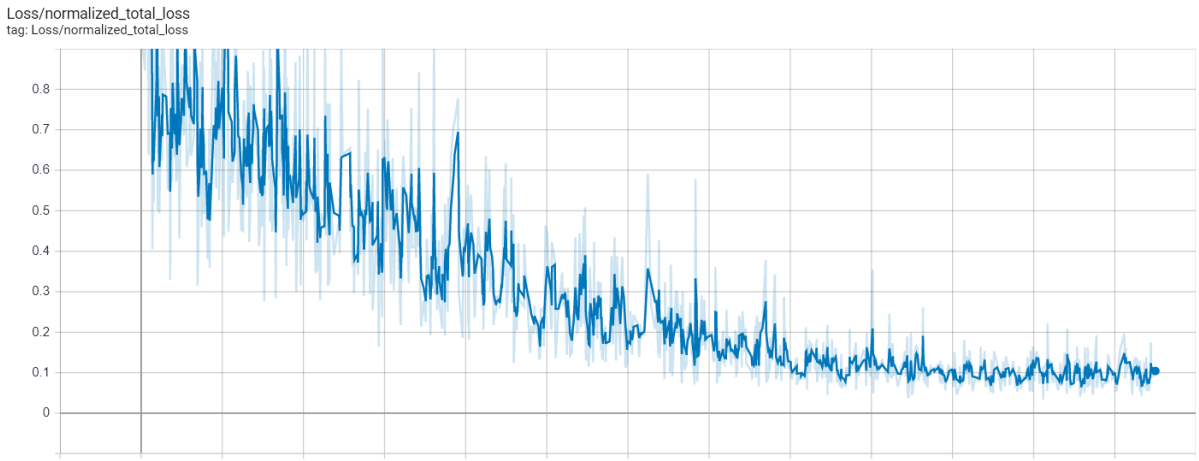
Σχήμα 5.8: Βήματα εκπαίδευσης ανιχνευτή Faster R-CNN με το TensorFlow 2 Object Detection API.

Εκτός αυτού, η εκπαίδευση και η επαλήθευση μπορούν να παρακολουθούνται

ζωντανά χρησιμοποιώντας το TensorBoard. Αυτό το λογισμικό επιτρέπει την οπτικοποίηση διαφορετικών μετρήσεων αξιολόγησης και για τα δύο σύνολα ενώ το δίκτυο εκπαιδεύεται. Εκτελούμε το Tensorboard με την εντολή στην απεικόνιση 5.9.

Απεικόνιση 5.9: Οπτικοποίηση δεδομένων με το TensorBoard.

```
# Start tensorboard and log directory for tensorboard to watch
!tensorboard --logdir={model_dir}
```



Σχήμα 5.9: Γράφημα της μείωση της συνολικής απόκλισης κατά τη εκπαίδευση του Faster R-CNN από το TensorBoard.

Ολοκληρώνοντας την εκπαίδευση του μοντέλου, αξιολογούμε την απόδοση του ανιχνευτή που εκπαιδεύτηκε με βάση της μετρήσεις της πρόκλησης COCO, οι οποίες περιγράφονται αναλυτικά στην ενότητα 5.6.

Απεικόνιση 5.10: Αξιολόγηση του εκπαιδευμένου μοντέλου.

```
%cd {repo_dir_path}
!python /content/models/research/object_detection/model_main_tf2.py \
    --pipeline_config_path={pipeline_fname} \
    --model_dir={model_dir} \
    --checkpoint_dir={model_dir} \
    --eval_timeout=60
```

Μόλις ολοκληρωθεί η εργασία της εκπαίδευσης, πρέπει να εξάγουμε το πρόσφατα εκπαιδευμένο γράφημα συμπερασμάτων, το οποίο θα μπορεί να χρησιμοποιηθεί αργότερα για την εκτέλεση του εντοπισμού αντικειμένων. Αυτό μπορεί να γίνει ως εξής:

```
I1221 00:46:16.076402 140087536568128 coco_evaluation.py:293] Performing evaluation on 573 images.
creating index...
index created!
INFO:tensorflow:Loading and preparing annotation results...
I1221 00:46:16.078691 140087536568128 coco_tools.py:116] Loading and preparing annotation results...
INFO:tensorflow:DONE (t=0.02s)
I1221 00:46:16.098630 140087536568128 coco_tools.py:138] DONE (t=0.02s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=2.05s).
Accumulating evaluation results...
DONE (t=0.35s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.906
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.990
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.980
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.850
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.872
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.918
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.774
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.928
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.929
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.870
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.895
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.942
INFO:tensorflow:Eval metrics at step 25000
I1221 00:46:18.533307 140087536568128 model_lib_v2.py:954] Eval metrics at step 25000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.905539
I1221 00:46:29.788581 140087536568128 model_lib_v2.py:957]      + DetectionBoxes_Precision/mAP: 0.905539
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 0.989872
I1221 00:46:29.790875 140087536568128 model_lib_v2.py:957]      + DetectionBoxes_Precision/mAP@.50IOU: 0.989872
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 0.979513
I1221 00:46:29.792619 140087536568128 model_lib_v2.py:957]      + DetectionBoxes_Precision/mAP@.75IOU: 0.979513
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (small): 0.850177
I1221 00:46:29.794220 140087536568128 model_lib_v2.py:957]      + DetectionBoxes_Precision/mAP (small): 0.850177
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (medium): 0.871834
I1221 00:46:29.795111 140087536568128 model_lib_v2.py:957]      + DetectionBoxes_Precision/mAP (medium): 0.871834
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (large): 0.918206
I1221 00:46:29.795989 140087536568128 model_lib_v2.py:957]      + DetectionBoxes_Precision/mAP (large): 0.918206
```

Σχήμα 5.10: Βήματα εκπαίδευσης Faster R-CNN με το TensorFlow 2 Object Detection API

Απεικόνιση 5.11: Εξαγωγή του εκπαιδευμένου μοντέλου.

```
%cd {repo_dir_path}
!python /content/models/research/object_detection/exporter_main_v2.py \
   --input_type image_tensor \
   --pipeline_config_path {pipeline_fname} \
   --trained_checkpoint_dir {model_dir} \
   --output_directory {output_dir}
```

Αφού ολοκληρωθεί η παραπάνω διαδικασία, θα πρέπει να βρούμε έναν νέο φάκελο exported-models που έχει την ακόλουθη δομή:

```
object-detection-guns/
  ...
  exported-models/
    faster_rcnn_resnet50_v1-exported-models/
      checkpoint/
```

```
            saved_model/
            pipeline.config
        ...
```

Ανίχνευση αντικειμένων με το αποθηκευμένο μοντέλο

Απεικόνιση 5.12: Φόρτωση όλων των εικόνων δοκιμής για ανίχνευση με το εκπαιδευμένο μοντέλο.

```
IMAGE_DIR = os.path.join(repo_dir_path, "images", "test")
IMAGE_PATHS = []

for file in os.listdir(IMAGE_DIR):
    if file.endswith(".jpg") or file.endswith(".png"):
        IMAGE_PATHS.append(os.path.join(IMAGE_DIR, file))
```

Εκτελούμε τον κώδικα της απεικόνισης 5.13 για τη φόρτωση του εκπαιδευμένου μοντέλου που έχουμε εξαγάγει σε προηγούμενο βήμα.

Απεικόνιση 5.13: Φόρτωση του εκπαιδευμένου μοντέλου.

```
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils

# Label Map path
PATH_TO_LABELS = label_map_pbtxt_fname
# Saved model path
PATH_TO_SAVED_MODEL = os.path.join(output_dir, "saved_model")

# Load saved model and build the detection function
detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL)

# Set category index
category_index = label_map_util.create_category_index_from_labelmap(
    PATH_TO_LABELS, use_display_name=True)
```

Τέλος, για κάθε εικόνα που θέλουμε να ανιχνεύσουμε αντικείμενα, τη μεταφέρουμε στο αντίστοιχο φάκελο και εκτελούμε τον κώδικα της απεικόνισης 5.17, ο

οποίος θα ανιχνεύσει όλα τα αντικείμενα που μπορεί να ανιχνεύσει το εκπαιδευμένο μοντέλο που φορτώσαμε παραπάνω. Στη συνέχεια για κάθε εικόνα θα σχεδιάσει το πλαίσιο οριοθέτησης του κάθε αντικειμένου που εντοπίστηκε μαζί με το όνομα και το ποσοστό εμπιστοσύνης της κλάσης.

Απεικόνιση 5.14: Δημιουργία προβλέψεων στο σύνολο δεδομένων των εικόνων δοκιμής.

```
for image_path in [IMAGE_PATHS]:
    image_np = np.array(Image.open(image_path))
    input_tensor = tf.convert_to_tensor(image_np)
    input_tensor = input_tensor[tf.newaxis, ...]
    object_detections = detect_fn(input_tensor)
    num_detections = int(object_detections.pop('num_detections'))
    object_detections = {key: value[0, :num_detections].numpy()
                         for key, value in object_detections.items()}
    object_detections['num_detections'] = num_detections
    object_detections['detection_classes'] = object_detections['detection_classes'].astype(np.
        int64)
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            object_detections['detection_boxes'],
            object_detections['detection_classes'],
            object_detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=30,
            min_score_thresh=.40,
            agnostic_mode=False)

    plt.figure(figsize = (12,8))
    plt.imshow(image_np_with_detections)
plt.show()
```

Στο σχήμα 5.11 και στο σχήμα 5.12 εμφανίζονται τα αποτελέσματα της ανίχνευσης όπλων με τη χρήση του εκπαιδευμένου μοντέλου σε εικόνες.

Σχήμα 5.11: Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου Faster R-CNN. Αυτή η εικόνα δεν είναι μέρος του συνόλου εκπαίδευσης ή επαλήθευσης του μοντέλου.

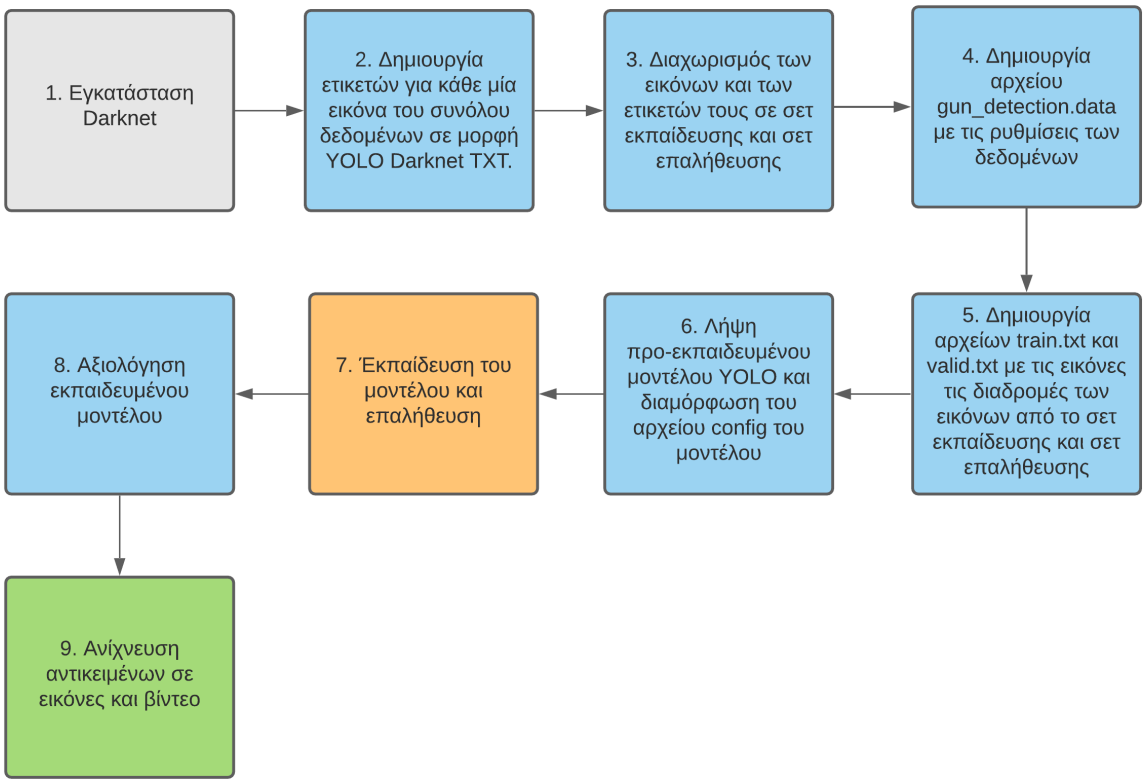
Σχήμα 5.12: Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου Faster R-CNN. Αυτή η εικόνα δεν είναι μέρος του συνόλου εκπαίδευσης ή επαλήθευσης του μοντέλου.

Παρατηρούμε ότι στις δύο παραπάνω εικόνες τα αποτελέσματα του Faster R-CNN είναι πάρα πολύ καλά. Η ίδια διαδικασία επαναλαμβάνεται για τους αλγορίθμους SSD, RetinaNet και EfficientDet τροποποιώντας μόνο τη μεταβλητή *selected_model*

στην απεικόνιση 5.1 με τον αλγόριθμο που εξετάζουμε και εκτελώντας πάλι όλα τα βήματα με την ίδια σειρά. Αναλυτικά αποτελέσματα σχετικά με την απόδοση των ανιχνευτών παρουσιάζονται στην ενότητα 5.7.

5.5 Εκπαίδευση ανιχνευτή αντικειμένων με Darknet

Σε αυτήν την ενότητα θα εκπαιδευτεί ο ανιχνευτής αντικείμενων YOLOv3 και YOLOv4 με τη χρήση του framework Darknet. Όπως και με το TensorFlow 2 Object Detection API έτσι και με το Darknet η διαδικασία της εκπαίδευσης των μοντέλων YOLO και της ανίχνευσης είναι παρόμοια και αποτελείται από τα μπλοκ που απεικονίζονται στο σχήμα 5.13.



Σχήμα 5.13: Διάγραμμα της διαδικασίας ανάπτυξης ενός ανιχνευτή αντικειμένου με το Darknet.

Έχοντας έτοιμο το σύνολο των εικόνων μας και στη μορφή που απαιτείται από το Darknet για να το διαβάσει, θα χρειαστεί να δημιουργηθούν 1 αρχείο ρυθμίσεων με τις διαδρομές των αρχείων δεδομένων και 2 αρχεία train.txt και valid.txt που περιέχουν σε κάθε μια σειρά τη διαδρομή κάθε εικόνας που συμπεριλαμβάνεται στο αντίστοιχο σετ.

Απεικόνιση 5.15: Δημιουργία αρχείου ρυθμίσεων δεδομένων gun_detection.txt και αρχείου train.txt με τις διαδρομές όλων των εικόνων για εκπαίδευση

```
with open('data/gun_detection.data', 'w') as out:
  out.write('classes = 1\n')
  out.write('train = data/train.txt\n')
  out.write('valid = data/valid.txt\n')
  out.write('names = data/gun_detection.names\n')
  out.write('backup = backup/')

#generate the train file
with open('data/train.txt', 'w') as out:
  for img in [f for f in os.listdir('train') if f.endswith('jpg')]:
    out.write('data/obj/' + img + '\n')

#generate the valid file
with open('data/valid.txt', 'w') as out:
  for img in [f for f in os.listdir('valid') if f.endswith('jpg')]:
    out.write('data/obj/' + img + '\n')
```

Παράλληλα, για κάθε έναν από τους αλγορίθμους YOLOv3 και YOLOv4 θα χρειαστεί να επεξεργαστούμε τα αρχεία ρυθμίσεων yolov3.cfg και yolov4.cfg, αντίστοιχα, τα οποία ρυθμίζουν τις υπερπαραμέτρους του κάθε δικτύου αλλά και τις παραμέτρους εκπαίδευσης.

Σε επόμενο βήμα, θα χρειαστεί να κατεβάσουμε τα προ-εκπαιδευμένα μοντέλα για τους αλγορίθμους YOLOv3 και YOLOv4 αντίστοιχα.

Απεικόνιση 5.16: Λήψη του προ-εκπαιδευμένου δικτύου YOLOv3 για εκπαίδευση.

```
!wget https://thesis.vasilis.pw/data/darknet53.conv.74
```

Απεικόνιση 5.17: Λήψη του προ-εκπαιδευμένου δικτύου YOLOv4 για εκπαίδευση.

```
!wget https://thesis.vasilis.pw/data/yolov4.conv.137
```

Ακόμα, ιδιαίτερα σημαντικό είναι να επεξεργαστούμε τα αρχεία ρυθμίσεων yolov3.cfg και yolov4.cfg, προσαρμόζοντας τις υπερπαραμέτρους των δικτύων των αλγορίθμων YOLOv3 και YOLOv4, αντίστοιχα.

Αρχικά, ορίζουμε τις τιμές των $batch$ ίση με 64 και $subdivisions$ ίση με 16. Μπορούμε να αυξήσουμε τιμή των batches, εφόσον διαθέτουμε αρκετούς υπολογιστικούς πόρους, για να εκτελεστεί η εκπαίδευση γρηγορότερα. Ο μέγιστος αριθμός επαναλήψεων για τις οποίες πρέπει να εκπαιδευτεί το δίκτυό μας ορίζεται με την παράμετρο $max_batches = 25000$. Επίσης ορίζουμε τα $steps = 20000, 22500$ που είναι 80%, 90% των $max_batches$.

Θα χρειαστεί να ενημερώσουμε τις παραμέτρους $classes$ και $filters$ των επιπέδων [yolo] και [convolutional] που βρίσκονται ακριβώς πριν από τα επίπεδα [yolo].

Σε αυτό το παράδειγμα, δεδομένου ότι έχουμε μια κλάση αντικειμένων (pistol) θα ενημερώσουμε τις παραμέτρους $classes$ στα επίπεδα [yolo] σε 1.

Παρομοίως, θα χρειαστεί να ενημερώσουμε την παράμετρο $filters$ με τον τύπο $filters = (classes + 5) * 3$. Για μία κλάση αντικειμένων πρέπει να ορίσουμε την τιμή τον φίλτρων ίση με 18.

Τέλος, εκκινούμε την εκπαίδευση εκτελώντας την εντολή που παρουσιάζεται στην απεικόνιση 5.18.

Απεικόνιση 5.18: Εκπαίδευση YOLOv3 με Darknet.

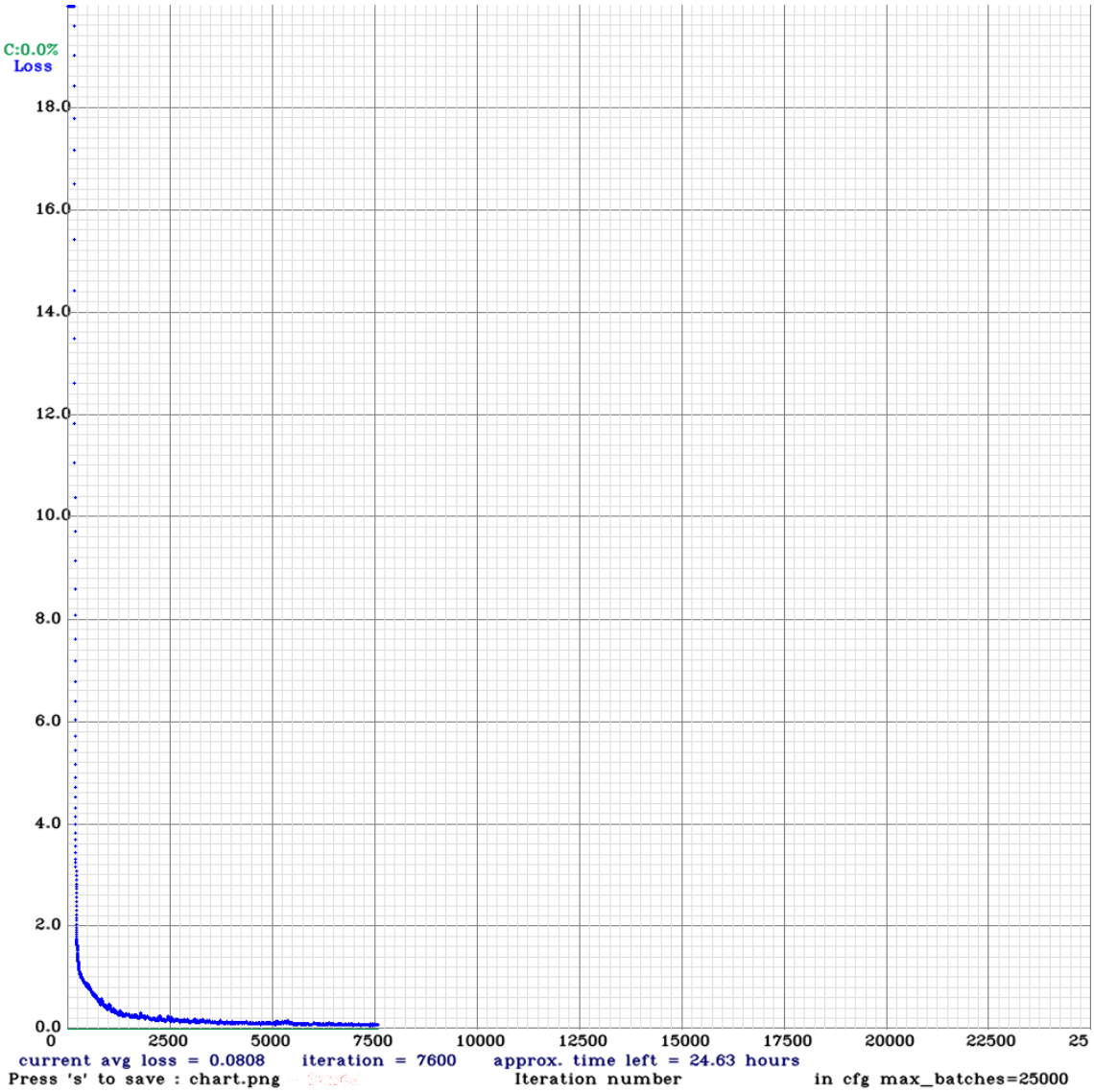
```
!./darknet detector train data/obj.data data/yolov3.cfg darknet53.conv.74 -map
```

```
 CUDA allocate done!
Loaded: 0.000068 seconds
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 4667.824707, iou_loss = 0.00000
0, total_loss = 4667.824707
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.344278), count: 2, class_loss = 1044.875122, iou_loss = 0.20910
6, total_loss = 1045.084229
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.421452), count: 12, class_loss = 293.471771, iou_loss = 0.27209
5, total_loss = 293.743866
 total_bbox = 14, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 4674.942871, iou_loss = 0.00000
0, total_loss = 4674.942871
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.189858), count: 2, class_loss = 1056.870605, iou_loss = 0.08203
1, total_loss = 1056.952637
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.408351), count: 18, class_loss = 293.563446, iou_loss = 0.46118
2, total_loss = 294.024628
 total_bbox = 34, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.198748), count: 1, class_loss = 4674.161133, iou_loss = 0.09228
5, total_loss = 4674.253418
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.346964), count: 8, class_loss = 1048.166504, iou_loss = 1.20385
7, total_loss = 1049.370361
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.327927), count: 14, class_loss = 294.029541, iou_loss = 0.35287
5, total_loss = 294.382416
 total_bbox = 57, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 4686.206055, iou_loss = 0.00000
0, total_loss = 4686.206055
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.437758), count: 6, class_loss = 1047.040771, iou_loss = 1.62304
7, total_loss = 1048.663818
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.389994), count: 16, class_loss = 292.865265, iou_loss = 0.84762
6, total_loss = 293.712891
 total_bbox = 79, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 4685.200195, iou_loss = 0.00000
0, total_loss = 4685.200195
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.343670), count: 3, class_loss = 1047.671387, iou_loss = 0.56286
6, total_loss = 1048.234253
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.365563), count: 3, class_loss = 292.144714, iou_loss = 0.03250
1, total_loss = 292.177216
 total_bbox = 85, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.300522), count: 3, class_loss = 4671.500488, iou_loss = 3.22070
3, total_loss = 4674.721191
```

Σχήμα 5.14: Εκπαίδευση του μοντέλου YOLOv4 με Darknet.

Στο σχήμα 5.15 παρουσιάζεται το γράφημα που εξάγεται από το Darknet κατά

την εκπαίδευση του μοντέλου YOLOv3 και δείχνει τη μείωση της απόκλισης μετά από κάθε βήμα της εκπαίδευσης (iteration).



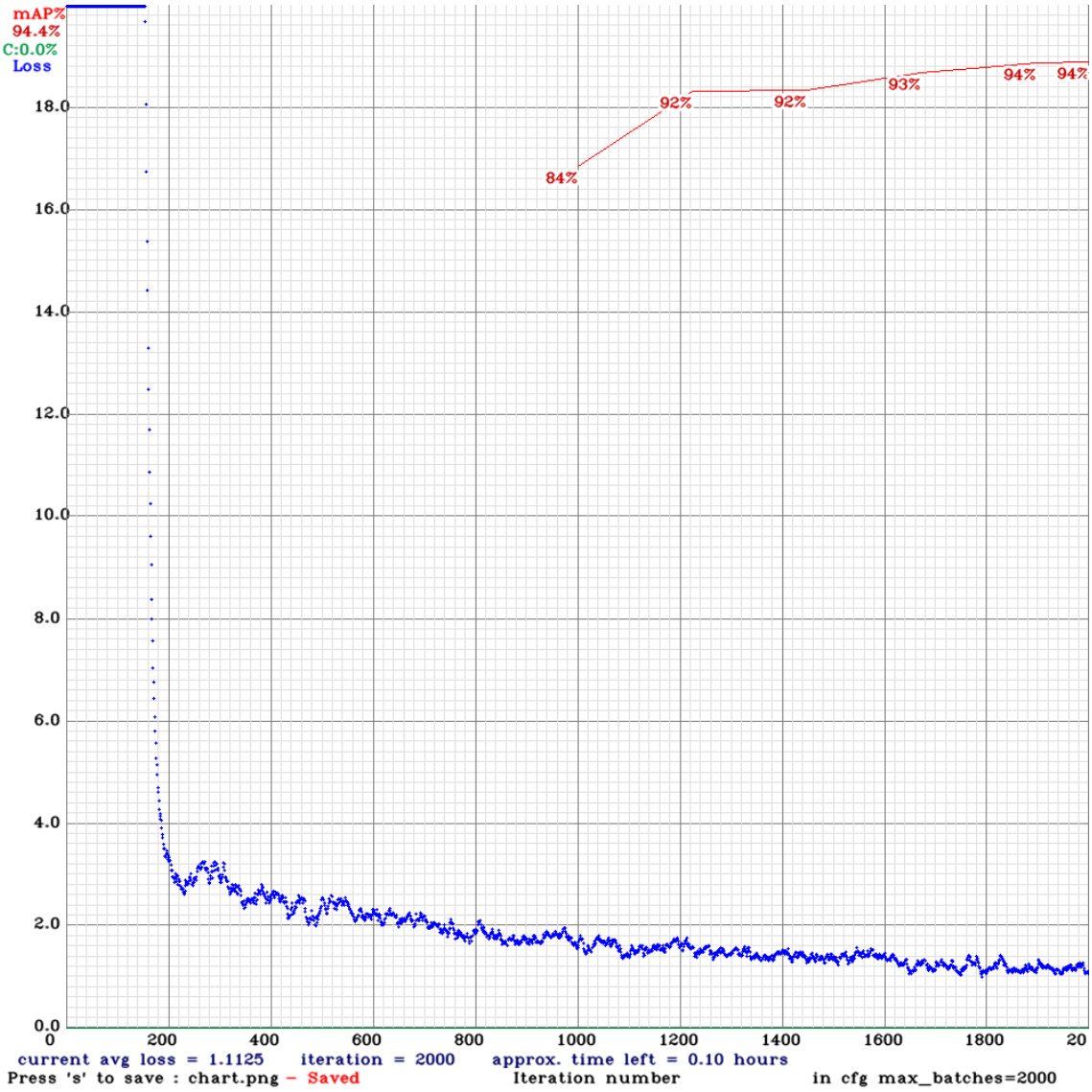
Σχήμα 5.15: Γράφημα απόκλισης και mAP σε YOLOv3 με Darknet

Αντίστοιχα, και για το YOLOv4, εκκινούμε την εκπαίδευση εκτελώντας την εντολή που παρουσιάζεται στην απεικόνιση 5.23.

Απεικόνιση 5.19: Εκπαίδευση YOLOv4 με Darknet.

```
1 !./darknet detector train data/obj.data cfg/yolov4.cfg yolov4.conv.137 -map
```

Με την ίδια λογική, και για τον αλγόριθμο YOLOv4, παρουσιάζεται στο σχήμα 5.16 το γράφημα που εξάγεται κατά την εκπαίδευση του μοντέλου YOLOv4 και δείχνει τη μείωση της απόκλισης μετά από κάθε βήμα της εκπαίδευσης αλλά και τον υπολογισμό του mAP.



Σχήμα 5.16: Γράφημα απόκλισης και mAP σε YOLOv4 με Darknet

Για την αξιολόγηση του εκπαιδευμένου μοντέλου εκτελούμε το Darknet με την παράμετρο map για τον υπολογισμό του mAP, όπως φαίνεται στην απεικόνιση 5.20.

Απεικόνιση 5.20: Υπολογισμός mAP σε YOLOv3 με Darknet.

```
1 !./darknet detector map data/obj.data cfg/yolov3.cfg backup/yolov3-guns.weights
```

Στο σχήμα 5.17 παρουσιάζεται το αποτέλεσμα του υπολογισμού του mAP σε IoU 50% του εκπαιδευμένου μοντέλου με YOLOv3. Το αποτέλεσμα του mAP είναι 89.67%.

```
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.304
avg_outputs = 516723
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3_last.weights...
 seen 64, trained: 972 K-images (15 Kilo-batches_64)
Done! Loaded 107 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 82 - type = 28
 Detection layer: 94 - type = 28
 Detection layer: 106 - type = 28
300
 detections_count = 454, unique_truth_count = 340
class_id = 0, name = pistol, ap = 89.67%         (TP = 297, FP = 34)

 for conf_thresh = 0.25, precision = 0.90, recall = 0.87, F1-score = 0.89
 for conf_thresh = 0.25, TP = 297, FP = 34, FN = 43, average IoU = 74.20 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.896741, or 89.67 %
Total Detection Time: 6 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Σχήμα 5.17: Υπολογισμός mAP εκπαιδευμένου μοντέλου YOLOv3 με Darknet. Το αποτέλεσμα είναι 89.67% σε IoU=50%.

Αντίστοιχα, και για το μοντέλο με YOLOv4.

Απεικόνιση 5.21: Υπολογισμός mAP σε YOLOv4 με Darknet

```
1 !./darknet detector map data/obj.data cfg/yolov4.cfg backup/yolov4-guns.weights
```

Στο σχήμα 5.18 παρουσιάζεται το αποτέλεσμα του υπολογισμού του mAP σε IoU 50% του εκπαιδευμένου μοντέλου με YOLOv4. Το αποτέλεσμα του mAP είναι 94.40%.

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 59.563
avg_outputs = 489778
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/custom-yolov4-detector_final.weights...
 seen 64, trained: 96 K-images (1 Kilo-batches_64)
Done! Loaded 162 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
300
 detections_count = 1022, unique_truth_count = 340
class_id = 0, name = pistol, ap = 94.40%         (TP = 321, FP = 59)

 for conf_thresh = 0.25, precision = 0.84, recall = 0.94, F1-score = 0.89
 for conf_thresh = 0.25, TP = 321, FP = 59, FN = 19, average IoU = 68.69 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.943973, or 94.40 %
Total Detection Time: 6 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Σχήμα 5.18: Υπολογισμός mAP εκπαιδευμένου μοντέλου YOLOv4 με Darknet. Το αποτέλεσμα είναι 94.40% σε IoU=50%.

Ανίχνευση αντικειμένων με το εκπαιδευμένο μοντέλο

Απεικόνιση 5.22: Ανίχνευση αντικειμένων με YOLOv3.

```
!./darknet detect cfg/yolov3.cfg backup/yolov3-guns.weights ../test/gun_1.jpg -dont-show
```

Σχήμα 5.19: Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv3 με Darknet.

Σχήμα 5.20: Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv3 με Darknet.

Απεικόνιση 5.23: Ανίχνευση αντικειμένων με YOLOv4.

```
!./darknet detect cfg/yolov4.cfg backup/yolov4-guns.weights ../test/gun_1.jpg -dont-show
```

Σχήμα 5.21: Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv4 με Darknet.

Σχήμα 5.22: Αποτελέσματα ανίχνευσης με τη χρήση του εκπαιδευμένου μοντέλου YOLOv4 με Darknet.

5.6 Μετρήσεις για την αξιολόγηση των ανιχνευτών αντικειμένων

Για την επίλυση της αυξανόμενης ανάγκης για ακριβή μοντέλα ανίχνευσης αντικειμένων χρησιμοποιήθηκαν διάφορες προσεγγίσεις. Πιο πρόσφατα, με τη διάδοση των συνελικτικών νευρωνικών δικτύων (CNN) και των επιταχυνόμενων GPU πλαισίων βαθιάς μάθησης, άρχισαν να αναπτύσσονται αλγόριθμοι εντοπισμού αντικειμένων από μια νέα προοπτική. Ειδικότερα, τα συνελικτικά νευρωνικά δίκτυα που μελετάμε σε αυτήν τη διπλωματική εργασία, όπως R-CNN, Fast R-CNN, Faster R-CNN, SSD και Yolo έχουν αυξήσει πολύ τα πρότυπα απόδοσης.

Μετά την εκπαίδευση ενός ανιχνευτή αντικειμένων, το επόμενο βήμα είναι να γνωρίζουμε την απόδοσή του. Σίγουρα, αυτό μπορεί να γίνει βλέποντας ότι το μοντέλο βρίσκει όλα τα αντικείμενα στις εικόνες που το λαμβάνει ως είσοδο. Δεδομένου ότι η εργασία κατηγοριοποίησης αξιολογεί μόνο την πιθανότητα εμφάνισης του αντικειμένου κλάσης στην εικόνα, είναι απλή εργασία για έναν κατηγοριοποιητή να εντοπίσει τις σωστές προβλέψεις από τις εσφαλμένες. Ωστόσο, η εργασία ανίχνευσης αντικειμένου εντοπίζει το αντικείμενο με περαιτέρω πληροφορίες, όπως ένα πλαίσιο οριοθέτησης που σχετίζεται με την αντίστοιχη βαθμολογία εμπιστοσύνης για να αναφέρει τον τρόπο με τον οποίο ανιχνεύεται το πλαίσιο οριοθέτησης της κλάσης αντικειμένων.

Αυτή η ενότητα παρουσιάζει τις ακόλουθες μετρήσεις: μέση ακρίβεια (AP), μέση τιμή της μέσης ακρίβειας (mAP), μέση ανάκληση (AR) και μέση τιμή της μέσης ανάκλησης (mAR), οι οποίες χρησιμοποιούνται για την αξιολόγηση της ακρίβειας και της αποδοτικότητας των ανιχνευτών αντικειμένων.

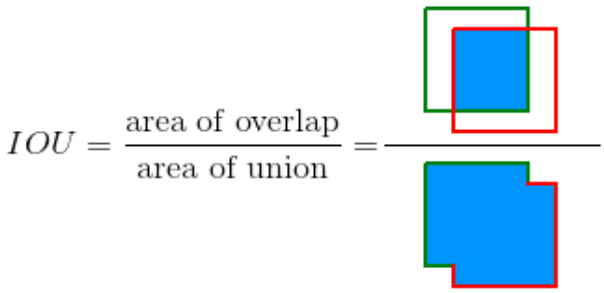
Intersection Over Union (IOU)

Η τομή πάνω από την ένωση (Intersection Over Union - IOU) είναι μέτρο βάσει του δείκτη Jaccard στη στατιστική το οποίο αξιολογεί την αλληλεπικάλυψη μεταξύ δύο οριοθετημένων πλαισίων. Απαιτεί ένα πλαίσιο οριοθέτησης αλήθειας εδάφους (ground truth) B_{gt} και ένα προβλεπόμενο πλαίσιο οριοθέτησης B_p. Εφαρμόζοντας το IOU μπορούμε να πούμε εάν μια ανίχνευση είναι έγκυρη (True Positive) ή όχι (False Positive). Το υψηλότερο IOU δίνεται από την αλληλεπικαλυπτόμενη περιοχή μεταξύ του προβλεπόμενου πλαισίου οριοθέτησης και του πλαισίου οριοθέτησης αλήθειας

εδάφους διαιρούμενο με την περιοχή ένωσης μεταξύ τους όπως φαίνεται στον τύπο 5.1.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{5.1}$$

Το σχήμα 5.23 απεικονίζει το IOU μεταξύ ενός πλαισίου οριοθέτησης αλήθειας εδάφους (σε πράσινο) και ενός εντοπισμένου πλαισίου οριοθέτησης (με κόκκινο χρώμα).



Σχήμα 5.23: Παράδειγμα IOU μεταξύ ενός πλαισίου οριοθέτησης αλήθειας εδάφους (σε πράσινο) και ενός εντοπισμένου πλαισίου οριοθέτησης (με κόκκινο χρώμα) [31].

Ένα $IOU > 0.5$ θεωρείται κανονικά μια «καλή» πρόβλεψη.

Μερικές βασικές έννοιες που χρησιμοποιούνται στις μετρήσεις είναι οι παρακάτω [75].

- **Αληθώς θετικά [TP]**: Αριθμός ανιχνεύσεων με $IoU > 0.5$
- **Ψευδώς θετικά [FP]**: Αριθμός ανιχνεύσεων με $IoU <= 0.5$ ή με περισσότερες από μία ανιχεύσεις
- **Ψευδώς αρνητικά [FN]**: Αριθμός αντικειμένων που δεν ανιχνεύθηκαν ή ανιχνεύθηκαν με $IoU <= 0.5$

Ακρίβεια (Precision)

Η ακρίβεια ορίζεται ως ο αριθμός των πραγματικών θετικών διαιρεμένος με το άθροισμα των αληθινών θετικών και των ψευδών θετικών. Η μαθηματική αναπαράσταση της ακρίβειας παρουσιάζεται στον τύπο 5.2.

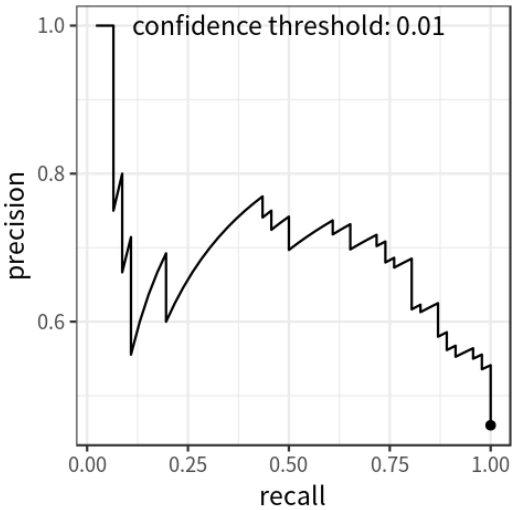
$$precision = \frac{TP}{TP + FP} \tag{5.2}$$

Ανάκληση

Η ανάκληση είναι η ικανότητα ενός μοντέλου να εντοπίζει όλες τις σχετικές περιπτώσεις (όλα τα πλαίσια περιορισμού της αλήθειας του εδάφους). Είναι το ποσοστό των αληθινών θετικών που εντοπίζονται μεταξύ όλων των σχετικών επίγειων αληθειών και δίνεται από τον τύπο 5.3.

$$recall = \frac{TP}{TP + FN} \tag{5.3}$$

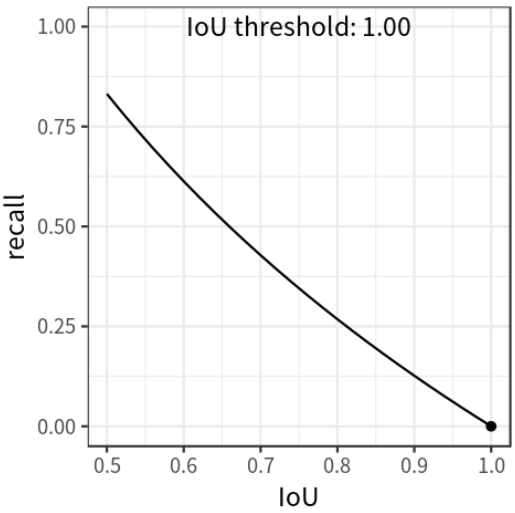
Ορίζοντας το κατώτατο όριο για βαθμολογία εμπιστοσύνης σε διαφορετικά επίπεδα, έχουμε διαφορετικά ζεύγη ακρίβειας και ανάκλησης. Με την ανάκληση στον άξονα x και την ακρίβεια στον άξονα y, μπορούμε να σχεδιάσουμε μια **καμπύλη ανάκλησης ακριβείας**, η οποία υποδεικνύει τη σχέση μεταξύ των δύο μετρήσεων. Το σχήμα 5.24 δείχνει μια προσομοιωμένη γραφική παράσταση της καμπύλης ανάκλησης ακρίβειας.



Σχήμα 5.24: Καμπύλη ανάκλησης-ακρίβειας (precision-recall) [32].

Καθώς το κατώτατο όριο για τη βαθμολογία της εμπιστοσύνης μειώνεται, η ανάκληση αυξάνεται μονοτονικά. Η ακρίβεια μπορεί να ανέβει και να κατεβαίνει, αλλά η γενικότερη τάση είναι να μειωθεί. Εκτός από την καμπύλη ανάκλησης ακρίβειας, υπάρχει και ένα άλλο είδος καμπύλης που ονομάζεται καμπύλη ανάκλησης-IoU. Παραδοσιακά, αυτή η καμπύλη χρησιμοποιείται για την αξιολόγηση της αποτελεσματικότητας των προτάσεων ανίχνευσης, αλλά είναι επίσης το θεμέλιο μιας μέτρησης που ονομάζεται μέση ανάκληση, η οποία παρουσιάζεται παρακάτω.

Με τον καθορισμό του ορίου για το IoU σε διαφορετικά επίπεδα, ο ανιχνευτής θα επιτύχει διαφορετικά επίπεδα ανάκλησης. Με αυτές τις τιμές, μπορούμε να σχεδιάσουμε την καμπύλη ανάκλησης-IoU κρατώντας τις τιμές IoU στον άξονα x και ανάκληση στον άξονα y.



Σχήμα 5.25: Καμπύλη ανάκλησης-IoU [32].

Average precision (AP)

Ένας άλλος τρόπος για να συγκριθεί η απόδοση των ανιχνευτών αντικειμένων είναι να υπολογιστεί η περιοχή κάτω από την καμπύλη ακρίβειας - ανάκλησης. Καθώς οι καμπύλες AP είναι συχνά καμπύλες ζιγκ-ζαγκ πάνω και κάτω, η σύγκριση διαφορετικών καμπυλών (διαφορετικών ανιχνευτών) στην ίδια γραφική παράσταση συνήθως δεν είναι εύκολη υπόθεση επειδή οι καμπύλες τείνουν να διασταυρώνονται μεταξύ τους πολύ συχνά. Γι΄ αυτό η μέση ακρίβεια (Average precision - AP) αποτελεί μια αριθμητική μέτρηση που μπορεί να βοηθήσει στη σύγκριση διαφορετικών ανιχνευτών. Στην πράξη, το AP είναι η μέση ακρίβεια όλων των τιμών ανάκλησης μεταξύ

0 και 1.

Από το 2010 και μετά, η μέθοδος υπολογισμού του AP από την πρόκληση PASCAL VOC έχει αλλάξει. Προς το παρόν, η παρεμβολή που πραγματοποιείται από την πρόκληση PASCAL VOC χρησιμοποιεί όλα τα σημεία δεδομένων, αντί να παρεμβάλλει μόνο 11 σημεία που βρίσκονται σε απόσταση μεταξύ τους όπως αναφέρεται στη δημοσίευση τους [68].

Παρεμβολή 11 σημείων

Η παρεμβολή 11 σημείων προσπαθεί να συνοψίσει το σχήμα της καμπύλης ακρίβειας - ανάκλησης με μέσο όρο την ακρίβεια σε ένα σύνολο έντεκα επιπέδων ανάκλησης με ίσες αποστάσεις [0, 0.1, 0.2, ..., 1]. Στον τύπο 5.4 παρουσιάζεται η μαθηματική αναπαράσταση του AP με την παρεμβολή 11 σημείων.

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1}) \tag{5.4}$$

όπου

$$p_{interp}(r) = \max_{r' \geq r} p(r') \tag{5.5}$$

και $p(r')$ είναι η μετρηθείσα ακρίβεια κατά την ανάκληση r'.

Αντί να χρησιμοποιείται η ακρίβεια που παρατηρείται σε κάθε σημείο, το AP λαμβάνεται παρεμβάλλοντας την ακρίβεια μόνο στα 11 επίπεδα r λαμβάνοντας τη μέγιστη ακρίβεια της οποίας η τιμή ανάκλησης είναι μεγαλύτερη από r.

Mean average precision (mAP)

Ο υπολογισμός του AP περιλαμβάνει μόνο μία τάξη. Ωστόσο, κατά την ανίχνευση αντικειμένων, υπάρχουν συνήθως περισσότερες κλάσεις. Η μέση τιμή της μέσης ακρίβειας (mAP) ορίζεται ως ο μέσος όρος του AP σε όλες τις κλάσεις K:

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{5.6}$$

Average recall (AR)

Όπως το AP, η μέση ανάκληση (AR) είναι επίσης μια αριθμητική μέτρηση που μπορεί να χρησιμοποιηθεί για τη σύγκριση της απόδοσης του ανιχνευτή. Στην ουσία, το AR είναι η ανάκληση κατά μέσο όρο σε όλα τα $IoU \in [0.5, 1.0]$ και μπορεί να υπολογιστεί ως δύο φορές η περιοχή κάτω από την καμπύλη ανάκλησης-IoU:

$$AR = 2 \int_{0.5}^{1} recall(o) do \tag{5.7}$$

όπου o είναι το IoU και $recall(o)$ είναι η αντίστοιχη ανάκληση.

Mean average recall (mAR)

Η μέση τιμή της μέσης ανάκλησης (Mean average recall - mAR) ορίζεται ως ο μέσος όρος του AR σε όλες τις κλάσεις K:

$$mAR = \frac{\sum_{i=1}^{K} AR_i}{K} \tag{5.8}$$

όπου O είναι το IoU και $recall(o)$ είναι η αντίστοιχη ανάκληση.

Οι μετρήσεις του COCO

- AP: AP στα IoU=.50:.05:.95, αντιστοιχεί σε μέσο όρο του AP σε 10 IOU (δηλ. 0,50, 0,55, 0,60,..., 0,95). Αποτελεί την κύρια μέτρηση πρόκλησης COCO.
- $AP^{\text{IoU=.50}}$: AP σε IoU=.50 (μέτρηση του PASCAL VOC)

- $AP^{\text{IoU=.75}}$: AP σε IoU=.75 (αυστηρή μέτρηση)
- AP^{small}: AP για μικρά αντικείμενα: εμβαδόν $< 32^2$
- AP^{medium}: AP για μεσαία αντικείμενα: $32^2 <$ εμβαδόν $< 96^2$
- AP^{large}: AP για μεγάλα αντικείμενα: εμβαδόν $> 96^2$
- $AR^{\text{max=1}}$: AR δεδομένου 1 εντοπισμού αντικειμένου ανά εικόνα
- $AR^{\text{max=10}}$: AR δεδομένου 10 εντοπισμών αντικειμένου ανά εικόνα
- $AR^{\text{max=100}}$: AR δεδομένου 100 εντοπισμών αντικειμένου ανά εικόνα
- AR^{small}: AR για μικρά αντικείμενα: εμβαδόν $< 32^2$
- AR^{medium}: AR για μεσαία αντικείμενα: $32^2 <$ εμβαδόν $< 96^2$
- AR^{large}: AR για μεγάλα αντικείμενα: εμβαδόν $> 96^2$

5.7 Συγκριτικά στοιχεία

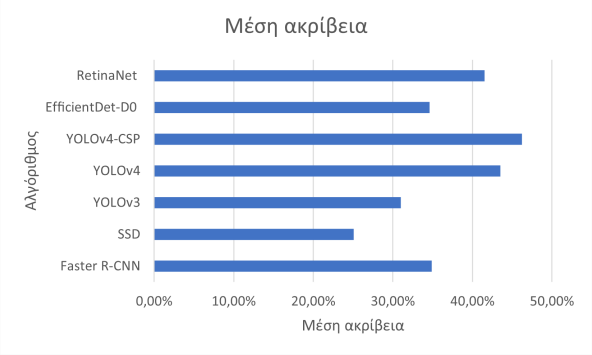
Σε αυτήν την υποενότητα παρουσιάζονται τα αποτελέσματα της υπολογιστικής σύγκρισης των αλγορίθμων που μελετάει η παρούσα διπλωματική εργασία ως προς την ακρίβεια τους αλλά και ως προς την ταχύτητα τους. Αρχικά, γίνεται μια αναφορά στη σύγκριση των αλγορίθμων μέσα από τη βιβλιογραφία, ενώ στη συνέχεια παρουσιάζονται τα αποτελέσματα της υπολογιστικής μελέτης που έγιναν στα πλαίσια εκπόνησης αυτής της διπλωματικής εργασίας. Στη βιβλιογραφία όλοι οι αλγόριθμοι πλέον συγκρίνονται στο σύνολο δεδομένων της πρόκλησης COCO, καθώς και χρησιμοποιούνται οι αντίστοιχες μετρήσεις της πρόκλησης COCO για τον υπολογισμό και τη σύγκριση της αποδοτικότητας του εκάστοτε αλγορίθμου.

Στον πίνακα 5.3 παρουσιάζονται τα συγκριτικά στοιχεία για την αξιολόγηση των αλγορίθμων στο σύνολο δεδομένων COCO, όπως παρουσιάζονται μέσα από τις επίσημες δημοσιεύσεις των αλγορίθμων.

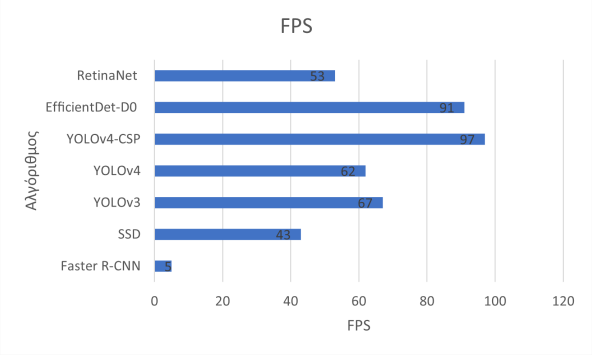
Στο σχήμα 5.26 παρουσιάζεται σχηματικά η μέση ακρίβεια των αλγορίθμων στο σύνολο δεδομένων COCO, ενώ στο σχήμα 5.27 παρουσιάζεται η απόδοση τους σε καρέ ανά δευτερόλεπτο.

Αλγόριθμος	**Μέγεθος**	**FPS**	**AP**	**AP$_{50}$**	**AP$_{75}$**	**AP$_S$**	**AP$_M$**	**AP$_L$**
Faster R-CNN	640	5	34.9%	55.7%	37.4%	15.6%	38.7%	50.9%
SSD	300	43	25.1%	43.1%	25.8%	6.6%	25.9%	41.4%
YOLOv3	416	35	31.0%	55.3%	32.3%	15.2%	33.2%	42.8%
YOLOv4	608	62	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
YOLOv4-CSP	512	97	46.2%	64.8%	50.2%	24.6%	50.4%	61.9%
EfficientDet-D0	512	91	34.6%	53.0%	37.1%	12.4%	39.0%	52.7%
RetinaNet	640	53	41.5%	60.5%	44.6%	23.3%	45.0%	58.0%

Πίνακας 5.3: Σύγκριση της αποδοτικότητας των αλγορίθμων στο COCO.



Σχήμα 5.26: Μέση ακρίβεια στο COCO.



Σχήμα 5.27: FPS στο COCO.

Με βάση την υπολογιστική μελέτη που έγινε στην παρούσα διπλωματική εργασία, παρακάτω παρουσιάζονται τα αποτελέσματα στη σύγκριση των αλγορίθμων αναγνώρισης και ανίχνευσης αντικειμένων ως προς την αποδοτικότητα τους αλλά και ως προς την ταχύτητά τους. Για τη σωστή αξιολόγηση της απόδοσης των αλγορίθμων χρησιμοποιήθηκαν οι μετρήσεις αξιολόγησης αποδοτικότητας της πρόκλησης COCO, οι οποίες παρουσιάζονται αναλυτικά στην ενότητα 5.6.

Στον πίνακα 5.4 παρουσιάζονται αναλυτικά οι μετρήσεις για την αξιολόγηση των αλγορίθμων της παρούσας εργασίας, ενώ στον πίνακα 5.5 παρουσιάζονται αναλυτικά οι μετρήσεις απόκλισης (Loss) μετά την εκπαίδευση των αλγορίθμων. Ακόμα, κατά την εκπαίδευση των αλγορίθμων, προσπαθήσαμε να χρησιμοποιήσουμε όσο το δυνατόν παρόμοιες τις υπερπαραμέτρους των αλγορίθμων, για να είναι όσο το δυνατόν πιο δίκαιη η σύγκριση μεταξύ τους. Ωστόσο, με το framework Darknet, το οποίο χρησιμοποιήθηκε για την εκπαίδευση των μοντέλων των αλγορίθμων YOLOv3 και YOLOv4 δεν ήταν εφικτή η αξιολόγηση μερικών παραμέτρων της πρόκλησης COCO γι' αυτό και παραλήφθηκαν από τον πίνακα 5.4 και την αξιολόγηση.

Αξιολόγηση της απόδοσης των ανιχνευτών

Στους παρακάτω πίνακες παρουσιάζεται η αξιολόγηση των εκπαιδευμένων μοντέλων των αλγορίθμων Faster R-CNN, SSD, EfficientDet, RetinaNet, YOLOv3 και YOLOv4 και σύμφωνα με τις μετρήσεις που χρησιμοποιούνται για τον χαρακτηρισμό της απόδοσης ενός ανιχνευτή αντικειμένων στην πρόκληση COCO.

Αλγόριθμος	**AP**	**AP$_{50}$**	**AP$_{75}$**	**AP$_S$**	**AP$_M$**	**AP$_L$**	**AR$_1$**	**AR$_{10}$**	**AR$_{100}$**	**AR$_S$**	**AR$_M$**	**AR$_L$**
Faster R-CNN	0.638	0.884	0.682	0.204	0.383	0.741	0.589	0.688	0.708	0.267	0.520	0.795
SSD	0.547	0.789	0.597	0.151	0.304	0.644	0.527	0.634	0.682	0.261	0.485	0.770
EfficientDet	0.572	0.855	0.617	0.209	0.399	0.652	0.567	0.663	0.693	0.394	0.530	0.762
RetinaNet	0.535	0.762	0.564	0.264	0.288	0.631	0.506	0.610	0.667	0.422	0.490	0.738
YOLOv3	0.593	0.893	0.655	-	-	-	-	-	-	-	-	-
YOLOv4	0.604	0.940	0.709	-	-	-	-	-	-	-	-	-

Πίνακας 5.4: Σύγκριση της απόδοσης των αλγορίθμων σύμφωνα με τις μετρήσεις της πρόκλησης COCO.

Αλγόριθμος	**Απ. Εντοπισμού**	**Απ. Αναγνώρισης**	**Απ. Κανονικοποίησης**	**Συνολική απόκλιση**	**Βήματα**	**Χρόνος/Βήμα (sec)**
Faster R-CNN	0.168	0.094	0.000000	0.446	25000	0.320
SSD	0.177708	0.326	0.149493	0.653	22100	0.256
EfficientDet	0.006	0.547214	0.056154	0.609	90000	0.398
RetinaNet	0.183	0.308	0.100	0.592	25000	0.582
YOLOv3	-	-	-	0.0808	7600	9.020
YOLOv4	-	-	-	1.112	2000	4.521

Πίνακας 5.5: Σύγκριση των διαφορετικών αποκλίσεων των αλγορίθμων κατά την εκπαίδευση τους.

Με βάση τον πίνακα 5.4 συμπεραίνουμε ότι στο πρόβλημα της ανίχνευσης όπλων και δεδομένου του δικού μας συνόλου δεδομένων, τη μεγαλύτερη ακρίβεια σε 50% IoU είχε ο αλγόριθμος YOLOv4 με ακρίβεια 94%. Ακολούθως, οι αλγόριθμοι YOLOv3, EfficientDet και Faster R-CNN είχαν εξίσου καλή ακρίβεια με 89.3%, 85.5% και 88.4%, αντίστοιχα. Σε 75% IoU την καλύτερη απόδοση είχε ο YOLOv4 με ακρίβεια 70.9%. Στην ακρίβεια αναγνώρισης μικρών αντικειμένων ο RetinaNet φάνηκε πιο αποτελεσματικός σε σχέση με τους υπόλοιπους.

Στη συνέχεια, στον πίνακα 5.6 παρουσιάζονται συγκριτικά οι χρόνοι των αλγορίθμων για ανίχνευση αντικειμένων σε μια εικόνα. Η συμπερασματολογία γίνεται αποκλειστικά είτε με τη χρήση μόνο CPU είτε και με την επιτάχυνση με τη χρήση της GPU.

Αλγόριθμος	**GeForce GTX 1080 Ti GPU**	**i7-7820X CPU @ 3.60GHz**
Faster R-CNN	0.29 δευτ.	0.73 δευτ.
SSD	0.05 δευτ.	0.08 δευτ.
EfficientDet	0.07 δευτ.	0.16 δευτ.
RetinaNet	0.15 δευτ.	0.33 δευτ.
YOLOv3	0.11 δευτ.	6.11 δευτ.
YOLOv4	0.16 δευτ.	6.93 δευτ.

Πίνακας 5.6: Σύγκριση χρόνων ανίχνευσης αντικείμενων σε CPU και GPU.

Ο αλγόριθμος SSD είναι αρκετά γρήγορος και στην εκτέλεση σε GPU αλλά και σε CPU ωστόσο δεν είναι τόσο ακριβής. Ο Faster R-CNN ενώ είναι αρκετά ακριβής, είναι αρκετά αργός με αποτέλεσμα να μην μπορεί να χρησιμοποιηθεί σε ανιχνεύσεις πραγματικού χρόνου. Από την άλλη πλευρά ο EfficientDet παρέχει ικανοποιητικούς χρόνους ανίχνευσης σε GPU με καλή ακρίβεια. Επιπλέον, οι αλγόριθμοι YOLOv3 και YOLOv4 είναι αρκετά γρήγοροι και πολύ ακριβείς όταν εκτελούνται σε GPU.

Κεφάλαιο 6

Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Συμπεράσματα

Σε αυτή τη διατριβή, μελετήσαμε το πρόβλημα υπολογιστικής όρασης για την ανίχνευση αντικειμένων με χρήση αλγορίθμων μηχανικής μάθησης. Αναμφίβολα, η δυνατότητα ανίχνευσης αντικειμένων στο χώρο από έναν υπολογιστή σε πραγματικό χρόνο, αλλά και με ακρίβεια είναι αναγκαία τόσο για το παρόν όσο και για το μέλλον. Η βαθιά μάθηση, καθώς και τα πολλαπλά σύνολα δεδομένων που είναι διαθέσιμα, έχουν ως αποτέλεσμα σημαντική βελτίωση στην ανίχνευση αντικειμένων. Από την πλευρά των αλγορίθμων, τα R-CNN είναι ακριβή, ωστόσο είναι αρκετά αργά, ακόμη και όταν εκτελούνται σε GPU. Αντίθετα, οι ανιχνευτές ενός σταδίου (SSD, YOLO, RetinaNet, EfficientDet) είναι αρκετά γρήγοροι, σε ανιχνεύσεις αντικειμένων σε πραγματικό χρόνο όταν εκτελούνται σε GPU. Μέσα από τα δικά μας πειράματα συμπεράναμε και εμείς το ίδιο. Ο αλγόριθμος Faster R-CNN είχε μέση ακρίβεια ίση με 63.8% αλλά ο χρόνος εκτέλεσής του σε GPU ήταν περίπου 0.29 δευτερόλεπτα. Από την πλευρά των ανιχνευτών ενός σταδίου, ακρίβεια ίση με 60.4% είχε ο αλγόριθμος YOLOv4 και 0.16 δευτερόλεπτα μέσο χρόνο ανίχνευσης σε GPU. Στην πραγματικότητα, είναι πολύ δύσκολο να υπάρχει δίκαιη σύγκριση μεταξύ των διαφορετικών ανιχνευτών αντικειμένων. Δεν υπάρχει απλή απάντηση στο ποιο μοντέλο είναι το καλύτερο. Για πραγματικές εφαρμογές, κάνουμε επιλογές για την εξισορρόπηση της ακρίβειας και της ταχύτητας. Εκτός από τους τύπους και τους αλγορίθμους ανιχνευτών, υπάρχουν και άλλες επιλογές που επηρεάζουν την από-

δοση όπως οι εξαγωγείς χαρακτηριστικών (VGG16, ResNet, Inception, MobileNet), η ανάλυση της εικόνας εισόδου, το σύνολο δεδομένων της εκπαίδευσης, η συνάρτηση απώλειας, το λογισμικό βαθιάς μάθησης που χρησιμοποιείται και πολλά άλλα. Ωστόσο, καθώς το πρόβλημα του εντοπισμού όπλου που μελετήσαμε σε αυτήν τη διπλωματική απαιτεί συμπεράσματα σε πραγματικό χρόνο, καθώς εφαρμόζεται σε συστήματα παρακολούθησης πραγματικού χρόνου, ο αλγόριθμος YOLOv4 παρέχει την καλύτερη σχέση ακρίβειας και ταχύτητας και με αυτόν τον τρόπο μας εξυπηρετεί καλύτερα.

6.2 Μελλοντικές επεκτάσεις

Σε αυτήν την ενότητα παρουσιάζονται μερικές επεκτάσεις της ανίχνευσης αντικειμένων με εφαρμογές σε ερευνητικό, επιστημονικό τομέα και στον εμπορικό τομέα αλλά και με εφαρμογές που μπορούν να καλυτερεύσουν και να διευκολύνουν τη ζωή των ανθρώπων.

Αυτόνομα αυτοκίνητα

Η αυτόνομη οδήγηση είναι μια από τις πιο σημαντικές τεχνολογίες που έχουν επωφεληθεί άμεσα λόγω της βελτίωσης της ανίχνευσης αντικειμένων και της όρασης του υπολογιστή. Πολλοί κατασκευαστές αυτοκινήτων έχουν έχουν ήδη ενσωματώσει κάποιο επίπεδο αυτονομίας στα αυτοκίνητα παραγωγής τους. Για παράδειγμα, τα αυτοκίνητα Tesla έχουν ήδη ενσωματωμένη διατήρηση λωρίδων, υποβοήθηση οδήγησης και ανίχνευση σύγκρουσης [76].

Αρκετές εταιρείες αυτοκινήτων έχουν δηλώσει ότι θα έχουν πλήρη αυτονομία έως το 2025, επιτρέποντας στα αυτοκίνητα να οδηγούν χωρίς καμία είσοδο χρήστη. Τέτοιες εξελίξεις επιτεύχθηκαν λόγω των σημαντικών προόδων στα δίκτυα ανίχνευσης αντικειμένων. Η βαθιά μάθηση επιτρέπει στα αυτοκίνητα να κατανοήσουν καλύτερα τους δρόμους και να αναγνωρίσουν αντικείμενα όπως αυτοκίνητα, φορτηγά και πεζούς.

Ανίχνευση ανωμαλιών

Η ανίχνευση ανωμαλιών είναι περίπτωση χρήσης εντοπισμού αντικειμένων που εξηγείται καλύτερα μέσω συγκεκριμένων παραδειγμάτων της βιομηχανίας. Στη γεωρ-

γία, για παράδειγμα, ένα προσαρμοσμένο μοντέλο ανίχνευσης αντικειμένων θα μπορούσε να εντοπίσει με ακρίβεια πιθανές περιπτώσεις ασθένειας των φυτών, επιτρέποντας στους αγρότες να εντοπίσουν απειλές για τις αποδόσεις των καλλιεργειών τους, που διαφορετικά δεν θα μπορούσαν να γίνουν αντιληπτές με γυμνό ανθρώπινο μάτι. Και στην υγειονομική περίθαλψη, η ανίχνευση αντικειμένων θα μπορούσε να χρησιμοποιηθεί για τη θεραπεία καταστάσεων που έχουν συγκεκριμένες και μοναδικές συμπτωματικές βλάβες. Ένα τέτοιο παράδειγμα έρχεται με τη μορφή της φροντίδας του δέρματος και της θεραπείας της ακμής. Ένα μοντέλο ανίχνευσης αντικειμένων θα μπορούσε να εντοπίσει περιπτώσεις ακμής σε δευτερόλεπτα. Ακόμα, με την χρήση αυτής της τεχνογνωσίας, διαγνώσεις εξετάσεων οι οποίες χρειάζονται οπτική διάγνωση από ιατρό, όπως για παράδειγμα ο εντοπισμός μια πνευμονίας από μια ακτινογραφία, μπορούν να γίνονται εξίσου γρήγορα και αποτελεσματικά από έναν υπολογιστή με πολύ καλά εκπαιδευμένα μοντέλα ανίχνευσης.

Λιανικό εμπόριο

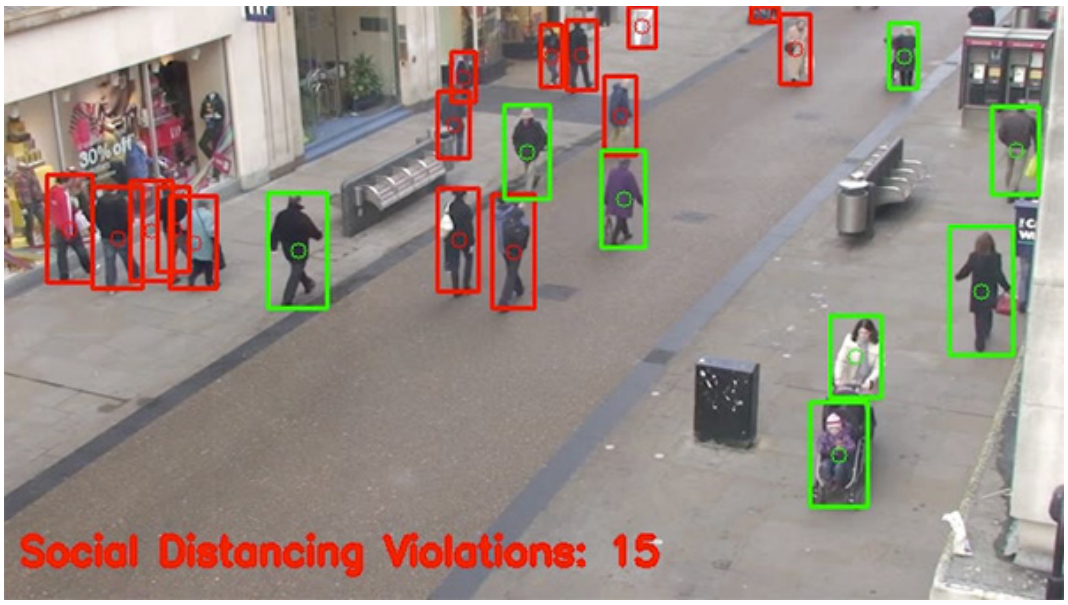
Σημαντικό παράδειγμα χρήσης της ανίχνευσης αντικειμένων στο λιανικό εμπόριο αποτελεί το κατάστημα Amazon Go στο Σιάτλ, το οποίο χρησιμοποιεί ανίχνευση αντικειμένων για να χρεώνει αυτόματα τους πελάτες τα προϊόντα που πήραν από τα ράφια [77]. Με τον εντοπισμό προϊόντων και παρατηρώντας αυτόματα την παρουσία ή την απουσία τους, οι λιανοπωλητές μπορούν να χρησιμοποιούν δίκτυα ανίχνευσης αντικειμένων για τον εντοπισμό κλοπής αντικειμένων και αποθεμάτων.

Επίβλεψη μέσω βίντεο

Καθώς οι προηγμένες τεχνικές ανίχνευσης αντικειμένων μπορούν να εντοπίσουν και να ανιχνεύσουν με ακρίβεια πολλαπλές εμφανίσεις ενός δεδομένου αντικειμένου σε μια σκηνή, αυτές οι τεχνικές φυσικά προσφέρονται για αυτοματοποίηση συστημάτων παρακολούθησης βίντεο. Για παράδειγμα, τα μοντέλα ανίχνευσης αντικειμένων είναι σε θέση να παρακολουθούν πολλά άτομα ταυτόχρονα, σε πραγματικό χρόνο, καθώς μετακινούνται σε μια δεδομένη σκηνή ή σε καρέ βίντεο. Από καταστήματα λιανικής έως δάπεδα βιομηχανικών εργοστασίων, αυτό το είδος λεπτομερούς παρακολούθησης θα μπορούσε να παρέχει ανεκτίμητες πληροφορίες σχετικά με την ασφάλεια, την απόδοση και την ασφάλεια των εργαζομένων και πολλά άλλα. Η

καταμέτρηση πλήθους είναι μια άλλη πολύτιμη εφαρμογή της ανίχνευσης αντικειμένων. Για πυκνοκατοικημένες περιοχές όπως θεματικά πάρκα, εμπορικά κέντρα και πλατείες πόλεων, η ανίχνευση αντικειμένων μπορεί να βοηθήσει τις επιχειρήσεις και τους δήμους να μετρήσουν αποτελεσματικότερα διαφορετικά είδη κυκλοφορίας.

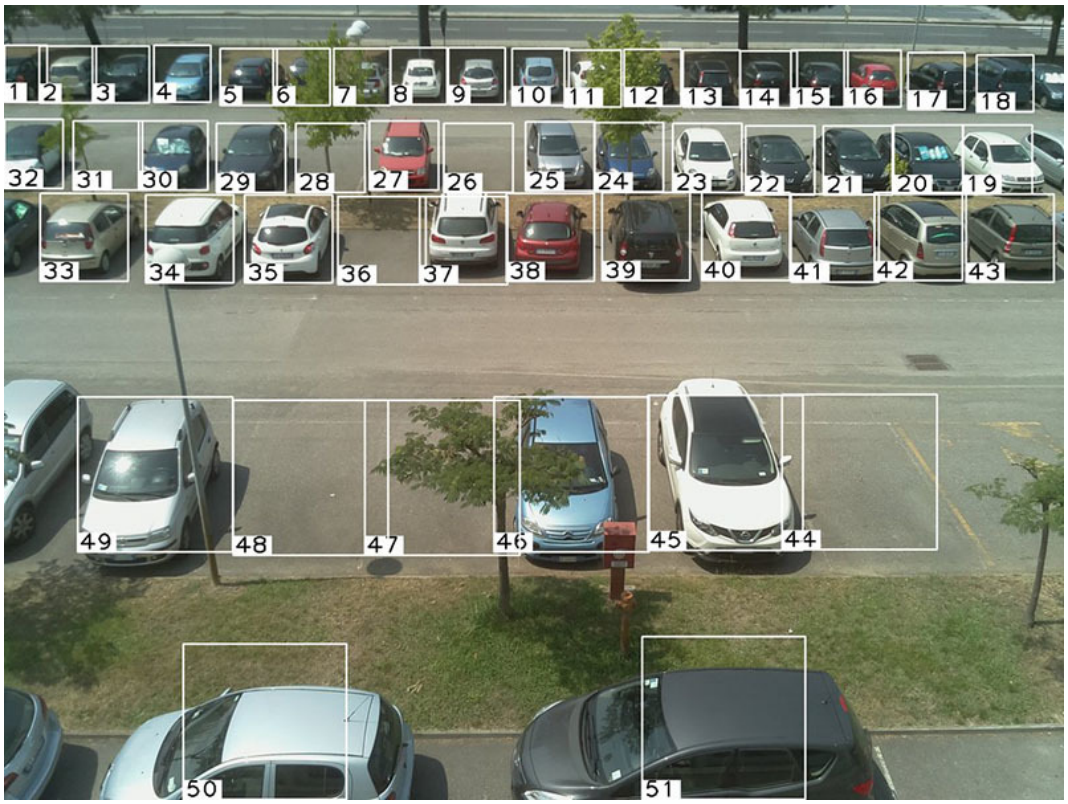
Στο σχήμα 6.2 παρουσιάζεται ένα παράδειγμα ανιχνευτή τήρησης κοινωνικών αποστάσεων με χρήση του αλγορίθμου YOLO και του OpenCV [33].



Σχήμα 6.1: Παράδειγμα ανιχνευτή τήρησης κοινωνικών αποστάσεων με χρήση YOLOv3 και OpenCV [33]

Εύρεση θέσης στάθμευσης

Οι τεχνικές και οι αλγόριθμοι της ανίχνευσης αντικειμένων μπορούν επίσης να χρησιμοποιηθούν για να εντοπίζουν τις κενές και τις δεσμευμένες θέσεις στάθμευσης αυτοκινήτων λαμβάνοντας ως είσοδο μόνο το βίντεο από κάμερες που βλέπουν στις αντίστοιχες θέσεις.



Σχήμα 6.2: Παράδειγμα εντοπισμού θέσεων στάθμευσης και κατηγοριοποίηση ως ελεύθερες ή κατειλημμένες [34].

Από όλα τα παραπάνω συμπεραίνουμε ότι η ανίχνευση αντικειμένων μέσω μηχανικής μάθησης αποτελεί σημαντικό τομέα στην έρευνα καθώς μέσα από πολυάριθμες εφαρμογές αναμένεται να διευκολύνει και να αναβαθμίσει τις ζωές των ανθρώπων.

Παράρτημα Α΄

Ακρωνύμια και συντομογραφίες

ML Machine Learning

CNN Convolutional neural network

GPU Graphics Processing Unit

GPU Graphics Processing Unit

HOG Histograms of Oriented Gradients

SVM Support Vector Machine

TPU Tensor Processing Units

IoU Intersection over Union

mAP mean Average Precision

ROI Regions of Interest

RPN Region Proposal Network

FPN Feature Pyramid Network

YOLO You Only Look Once

SSD Single Shot Detector

R-CNN Region Based Convolutional neural network

ΣΝΔ Συνελικτικό Νευρωνικό Δίκτυο

Βιβλιογραφία

[1] . Γεωργούλη, *Τεχνητή νοημοσύνη*. Αθήνα:Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, 2015, http://hdl.handle.net/11419/3381.

[2] Digit recognition-tutorial (cnn) - kaggle. [Online]. Available: https://www.kaggle.com/tarunkr/digit-recognition-tutorial-cnn-99-67-accuracy

[3] F.-F. Li, A. Karpathy, and J. Johnson, "Cs231n convolutional neural networks for visual recognition," https://cs231n.github.io/convolutional-networks/, 2020.

[4] Uniqtech. (2018) Understand the softmax function in minutes. [Online]. Available: https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

[5] SuperDataScience. (2018) Convolutional neural networks (cnn): Step 4 - full connection. [Online]. Available: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[7] P. Irgens, C. Bader, T. Lé, D. Saxena, and C. Ababei, "An efficient and cost effective fpga based implementation of the viola-jones face detection algorithm," *HardwareX*, vol. 1, pp. 68 – 75, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2468067216300116

[8] A. Geitgey, "Machine learning is fun! part 4: Modern face recognition with deep learning," Sep 2020. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

[9] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013. [Online]. Available: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013.

[11] R. Girshick, "Fast r-cnn," 2015.

[12] R. Gandhi, "R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms," Jul 2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[14] X. Zhang. (2018) Simple understanding of mask rcnn. [Online]. Available: https://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[16] E. Forson. (2017) Understanding ssd multibox — real-time object detection in deep learning. [Online]. Available: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

[17] Csci 1430: Introduction to computer vision - project 6. [Online]. Available: http://cs.brown.edu/courses/cs143/2017_Spring/proj6a/

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017.

[19] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[20] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[21] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[22] A. Kathuria. (2018) How to implement a yolo (v3) object detector from scratch in pytorch. [Online]. Available: https://www.kdnuggets.com/2018/05/implement-yolo-v3-object-detector-pytorch-part-1.html

[23] "A Closer Look at YOLOv3 – DMP BLOG." [Online]. Available: https://blog.dmprof.com/post/a-closer-look-at-yolov3/

[24] C. Bourez. Bounding box object detectors: understanding yolo, you look only once. [Online]. Available: http://christopher5106.github.io/object/detectors/2017/08/10/bounding-box-object-detectors-understanding-yolo.html

[25] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[26] [Online]. Available: https://developers.google.com/machine-learning/practica/image-classification/preventing-overfitting

[27] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," 2017.

[28] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," 2018.

[29] V. Thambawita, R. Ragel, and D. Elkaduwe, "To use or not to use: Graphics processing units for pattern matching algorithms," 12 2014.

[30] Wikipedia, "Tensor Processing Unit — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=Tensor%20Processing%20Unit&oldid=978594015, 2020, [Online; accessed 01-November-2020].

[31] "Object-Detection-Metrics: Most popular metrics used to evaluate object detection algorithms." [Online]. Available: https://github.com/rafaelpadilla/Object-Detection-Metrics

[32] "An Introduction to Evaluation Metrics for Object Detection | NickZeng|□□□." [Online]. Available: https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/

[33] "Opencv social distancing detector." [Online]. Available: https://www.pyimagesearch.com/2020/06/01/opencv-social-distancing-detector/

[34] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327–334, 2017.

[35] A. V. Joshi, *Machine Learning and Artificial Intelligence*. Springer Nature Switzerland AG, 2020.

[36] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM JOURNAL OF RESEARCH AND DEVELOPMENT*, pp. 71–105, 1959.

[37] L. Breiman, "Statistical modeling: The two cultures," *Cornell University Library*, 08 2015.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[39] D. Misra, "Mish: A self regularized non-monotonic activation function," 2020.

[40] J. Brownlee. (2016) Overfitting and underfitting with machine learning algorithms. [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," , vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[43] C. Cortes and V. Vapnik, "Support-vector networks," pp. 273–297, 1995.

[44] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifier," *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, vol. 5, 08 1996.

[45] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Comput. Vis. Pattern Recog*, vol. 1, 01 2001.

[46] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[49] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, "Scalable, high-quality object detection," 2015.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[51] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016.

[52] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015.

[54] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[55] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[56] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017.

[57] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," 2019.

[58] JetBrains, "Pycharm ide." [Online]. Available: https://www.jetbrains.com/pycharm/

[59] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[60] "Anaconda software distribution," 2020. [Online]. Available: https://docs.anaconda.com/

[61] Project Jupyter, "Jupyter notebook." [Online]. Available: https://jupyter.org/

[62] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[63] NVIDIA, P. Vingelmann, and F. H. Fitzek, "Cuda, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[64] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[65] Tensorboard: Tensorflow's visualization toolkit. [Online]. Available: https://www.tensorflow.org/tensorboard

[66] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[68] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[69] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014.

[70] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, 2020.

[71] A. Vittorio, "Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset," https://github.com/EscVM/OIDv4_ToolKit, 2018.

[72] Weapons detection for security and video surveillance | soft computing and intelligent information systems. [Online]. Available: https://sci2s.ugr.es/weapons-detection

[73] Tensorflow object detection api - github. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection

[74] Alexeyab/darknet: Yolov4v / scaled-yolov4 - neural networks for object detection (windows and linux version of darknet). [Online]. Available: https://github.com/AlexeyAB/darknet

[75] "Metrics to Use to Evaluate Deep Learning Object Detectors." [Online]. Available: https://www.kdnuggets.com/2020/08/metrics-evaluate-deep-learning-object-detectors.html

[76] "Autopilot ai." [Online]. Available: https://www.tesla.com/el_GR/autopilotAI

[77] "How the amazon go store's ai works." [Online]. Available: https://towardsdatascience.com/how-the-amazon-go-store-works-a-deep-dive-3fde9d9939e9