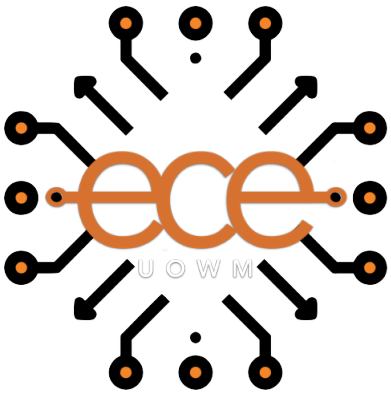


Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αλγόριθμοι μηχανικής μάθησης για δημιουργία λεζάντας σε εικόνες

Ζαφείριος Κυπριώτης (ΑΜ: 1024)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

15 Μαρτίου 2022

Περίληψη

Η εύρεση λεζάντας για μια εικόνα είναι ένα πρόβλημα στο οποίο έχει γίνει μεγάλη πρόοδος για την επίλυσή του τον τελευταίο καιρό. Αυτό οφείλεται στην εξέλιξη των αλγορίθμων μηχανικής μάθησης, και ειδικότερα των νευρωνικών δικτύων. Με την εξέλιξη των αλγορίθμων αυτών, έχουν εξελιχθεί και οι απαιτήσεις που έχουμε από αυτούς. Δηλαδή το πρόβλημα της εύρεσης λεζάντας σε μια εικόνα έχει μετασχηματιστεί στην εύρεση λεζάντας των διαφορετικών αντικειμένων που υπάρχει σε μια εικόνα. Αυτή την πιο γενική κατηγορία προβλημάτων την ονομάζουμε Εύρεση Αντικειμένων (Object Detection).

Για την αντιμετώπιση του προβλήματος της εύρεσης αντικειμένων έχουν δημιουργηθεί πολλαπλοί αλγόριθμοι, οι οποίοι ακολουθούν διαφορετικές μεθοδολογίες ώστε να καταφέρουν να έχουν μεγαλύτερη αξιοπιστία και ταχύτητα ώστε να μπορέσουν να αξιοποιηθούν σε εφαρμογές στον πραγματικό κόσμο. Σε αυτήν την εργασία θα ασχοληθούμε με την εξέλιξη των νευρωνικών δικτύων για την αντιμετώπιση του προβλήματος της εύρεσης αντικειμένων, θα συγκρίνουμε πέντε από τους πιο δημοφιλείς αλγορίθμους [2] [24] [17] [4] [28], θα εξηγήσουμε τις στρατηγικές που ακολουθούν και θα τους αξιολογήσουμε με τη χρήση συγκεκριμένων μετρικών [3] [5]. Τέλος, θα δούμε αν οι συγκεκριμένες στρατηγικές που ακολουθούν τους εξειδικεύουν στην επίλυση συγκεκριμένων προβλημάτων.

Λέξεις κλειδιά: Μηχανική μάθηση, Εύρεση αντικειμένων, Λεζάντες, Νευρωνικά δίκτυα, Python

Abstract

Image labeling is a fundamental problem in computer science, and in the last few years, great progress has been made towards its solution. This progress is caused by breakthroughs in machine learning algorithms, with the most prominent one being Neural Networks. Due to the rapid evolution of these algorithms, the problems that we can solve by using them have become more complex. The task of simply finding the label of an image has evolved to finding the label of all objects that an image contains. This new category of problems is called Object Detection.

In order to tackle this new problem, multiple machine learning algorithms have been created, each one trying to approach the problem from a different angle, striving for maximum precision and inference speed, with the main goal of applying those algorithms in order to solve real world problems. In this experiment we will compare five of the most popular algorithms [2] [24] [17] [4] [28], and we will describe the different strategies that they utilize, then we will compare them using certain standardized metrics [3] [5]. And lastly, we will draw conclusions regarding their effectiveness, and if there are certain types of problems that some of those algorithms might specialize in.

Keywords: Machine learning, Object detection, Labeling, Neural networks, Python

Δήλωση Πνευματικών Δικαιωμάτων

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 δημιουργία λεζάντας σε εικόνες'' 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. Νικόλαο 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 τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Ορισμός του Προβλήματος

Ο τομέας της μηχανικής μάθησης έχει εξελιχθεί ραγδαία τον τελευταίο καιρό λόγω της εκθετικής αύξησης της επεξεργαστικής δύναμης και των δυνατοτήτων αποθήκευσης δεδομένων. Το αποτέλεσμα είναι η δημιουργία όλο και πιο σύνθετων αλγόριθμων, οι οποίοι μπορούν να εφαρμοστούν σε όλο και μεγαλύτερο πλήθος δεδομένων. Ιδιαίτερη προσοχή έχει δοθεί τον τελευταίο καιρό στους αλγορίθμους των νευρωνικών δικτύων (neural networks) λόγω της μεγάλης προσαρμοστικότητας τους σε διάφορα είδη προβλημάτων τα οποία έχουν πολλές εφαρμογές στη καθημερινότητα μας [30]. Ειδικότερα έχει δοθεί μεγάλη προσοχή στον κλάδο της εύρεσης αντικειμένων (object detection) και ταξινόμησης εικόνων (image classification). Οι αλγόριθμοι εύρεσης αντικειμένων δεν ήταν αρκετά αποτελεσματικοί για τη χρήση σε πρακτικές εφαρμογές. Αυτό άλλαξε το 2012 με τη δημοσίευση του μοντέλου AlexNet [12], το οποίο κατάφερε να επιτύχει μεγάλα ποσοστά ακρίβειας και ταχύτητας, με αποτέλεσμα να κινήσει το ενδιαφέρον των ερευνητών. Οπότε επειδή η έρευνα στον τομέα αυτό είναι σχετικά πρόσφατη, και επειδή υπάρχουν πάρα πολλές προσεγγίσεις για την αρχιτεκτονική των μοντέλων που επιλύει το πρόβλημα της εύρεσης των αντικειμένων, είναι απαραίτητο να βρούμε κάποια μέτρα σύγκρισης για να δούμε τι πλεονεκτήματα μπορεί να προσφέρει η κάθε προσέγγιση.

Στις περισσότερες δημοσιεύσεις όταν θέλουν να συγκρίνουν οι συγγραφείς τα μοντέλα τους σε σχέση με τα προηγούμενα, πολλές φορές κάνουν μια αναφορά στην ταχύτητα εντοπισμού και στο μετρικό της μέσης ακρίβειας (mean average precision) που υπολογίζεται μέσω των κανόνων του διαγωνισμού COCO [3] και

PASCAL VOC [5], χωρίς να επεκτείνουν πολύ στις αδυναμίες που μπορεί να έχουν για τον εντοπισμό αντικειμένων με συγκεκριμένα χαρακτηριστικά.

Οπότε θα χρειαστεί να μελετήσουμε τα μοντέλα αυτά, να δούμε αν οι μεθοδολογίες που έχουν ακολουθήσει για τη βελτιστοποίηση της απόδοσής τους, αν δημιουργούν κάποιες αδυναμίες που δεν εμφανίζονται σε μια γενική τιμή μέσης ακρίβειας. Πιο συγκεκριμένα θα δούμε αν υπάρχει κάποια σημαντική επιρροή στα αποτελέσματα των μοντέλων αυτών η επιλογή της αρχιτεκτονικής των στρωμάτων και η χρήση διαφορετικών μεθοδολογιών για την εύρεση των περιβαλλόντων κουτιών.

1.2 Διάρθρωση του Κειμένου

Στο κεφάλαιο 2, κάνουμε μια σύντομη εισαγωγή για τον τρόπο λειτουργίας των απλών νευρωνικών δικτύων και των συνελικτικών νευρωνικών δικτύων. Και στη συνέχεια βλέπουμε το πως αξιοποιούνται τα συνελικτικά νευρωνικά δίκτυα για την επίλυση του προβλήματος της εύρεσης αντικειμένων και το πώς θα μπορούμε να συγκρίνουμε τους διαφορετικούς αλγορίθμους.

Στο κεφάλαιο 3 θα δούμε μερικές βασικές δομές νευρωνικών δικτύων, οι οποίες εμφανίζονται ανάμεσα στα μοντέλα που θα εξετάσουμε. Και θα περιγράψουμε τον τρόπο λειτουργίας τους.

Το κεφάλαιο 4 περιέχει τις περιγραφές των μοντέλων που θα εξετάσουμε, και τις μεθοδολογίες που χρησιμοποιούν ώστε να καταφέρουν να εντοπίσουν αντικείμενα.

Στο κεφάλαιο 5 παρουσιάζουμε τα αποτελέσματα των συγκρίσεων των μοντέλων, και θα εξετάσουμε τις διαφορές των αποτελεσμάτων που έχουν μεταξύ τους.

Στο κεφάλαιο 6 βρίσκονται τα συμπεράσματα της εργασίας.

Κεφάλαιο 2

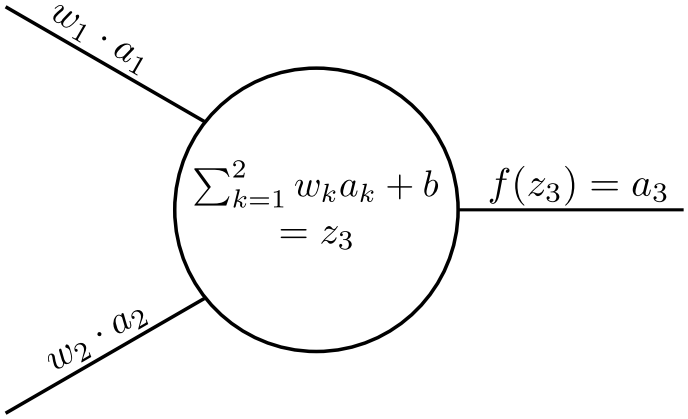
Εύρεση Αντικειμένων

2.1 Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα είναι μια κατηγορία αλγορίθμων του επιστημονικού κλάδου της μηχανικής μάθησης, όπου η βασική τους λειτουργία είναι εμπνευσμένη από τον τρόπο που λειτουργεί το νευρικό σύστημα. Το κάθε νευρωνικό δίκτυο αποτελείται από ένα σύνολο στοιχείων που ονομάζονται νευρώνες. Ό νευρώνας είναι το πιο απλό δομικό στοιχείο του δικτύου, και ανάλογα με το πλήθος και τη διάταξη των νευρώνων, μπορούν να σχηματιστούν μοντέλα για ένα πολύ μεγάλο εύρος εφαρμογών. Ο κάθε νευρώνας έχει πολλές εισόδους, συνήθως από άλλους νευρώνες, όπου η κάθε μια από αυτές τις εισόδους πολλαπλασιάζεται με ένα βάρος (weight) και συναθροίζονται και με μια τιμή τάσης (bias) για να βγάλουν μια τελική τιμή. Αυτή η τελική τιμή όμως δεν είναι η έξοδος του νευρώνα, πρέπει πρώτα να την εισάγουμε σε μια μη γραμμική συνάρτηση, γιατί αλλιώς, αν το δίκτυο αποτελούνταν μόνο από πολλαπλασιασμούς και προσθέσεις, τότε θα καταλήγαμε σε ένα μοντέλο εύρεσης καμπύλης (linear regressor). Αυτές οι μη γραμμικές συναρτήσεις ονομάζονται συναρτήσεις ενεργοποίησης (activation functions), οι οποίες πολλές φορές είναι επιλεγμένες ώστε να προσθέτουν μια σταθερότητα στο δίκτυο και να περιορίζουν το εύρος τιμών που μπορεί να έχει η έξοδος του κάθε νευρώνα. Η δομή του νευρώνα απεικονίζεται στο Σχήμα 2.1

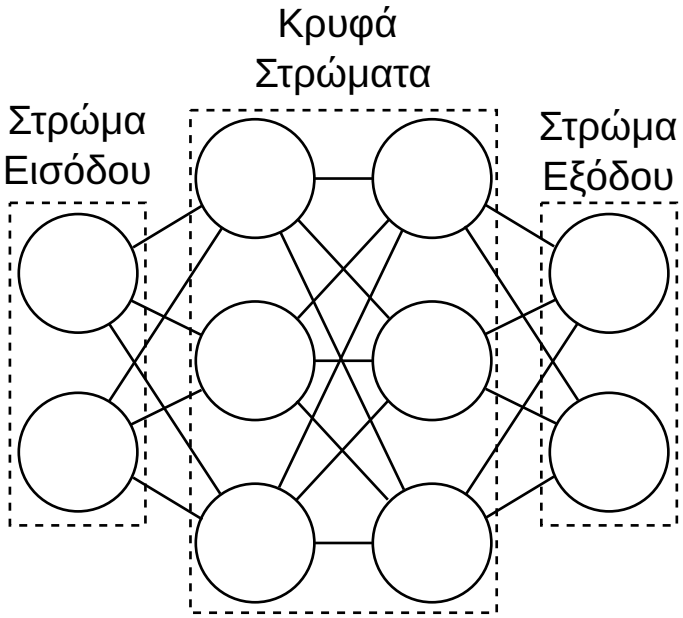
Οπότε όταν συνδυάσουμε αυτούς τους νευρώνες, δημιουργούμε ένα νευρωνικό δίκτυο, του οποίου η βασική δομή φαίνεται στο Σχήμα 2.2. Το νευρωνικό δίκτυο αποτελείται από στρώματα νευρώνων (layers), οπού το πρώτο στρώμα έχει την είσοδο των δεδομένων και το τελευταίο έχει την έξοδό του. Όλα τα ενδιάμεσα

Σχήμα 2.1: Δομή νευρώνα, Το a_1, a_2 είναι οι έξοδοι από τα προηγούμενα δίκτυα, w_1, w_2 είναι τα βάρη, b η τάση και f η συνάρτηση ενεργοποίησης.



στρώματα ονομάζονται κρυφά στρώματα (hidden layers), τα οποία σε διαφορετικές διατάξεις μπορούν να βελτιστοποιήσουν την ταχύτητά του δικτύου και την ακρίβεια των αποτελεσμάτων του. Όταν έχουμε καταφέρει να βρούμε τις τιμές για τα βάρη, και τις τιμές των τάσεων τότε μπορούμε να χρησιμοποιήσουμε το μοντέλο για την επίτευξη του σκοπού μας, θα μπορούμε απλά να εισάγουμε στο δίκτυο την πληροφορία και αυτή θα ταξιδέψει μέσα από κάθε νευρώνα. Στο τέλος, θα καταλήξει στους νευρώνες εξόδου όπου ανάλογα με τις τιμές τους μπορούμε να ερμηνεύσουμε το αποτέλεσμα.

Σχήμα 2.2: Παράδειγμα αρχιτεκτονικής νευρωνικού δικτύου



Για να μπορέσουμε να βρούμε τις κατάλληλες τιμές ώστε να έχουμε ικανοποιη-

τικά αποτελέσματα πρέπει να εκπαιδεύσουμε το δίκτυο. Στην αρχή της εκπαίδευσης, αναθέτουμε στα βάρη και στις τάσεις τυχαίες τιμές έτσι ώστε να μπορέσει να βγει ένα αρχικό αποτέλεσμα για μια είσοδο, στη συνέχεια έχουμε το στάδιο της εμπρόσθιας διάδοσης (forward pass), που στην ουσία υπολογίζουμε τις τιμές των νευρώνων ανά στρώμα ανάλογα με τα βάρη και τις τιμές εξόδων των νευρώνων του προηγούμενου στρώματος, και οι τιμές εξόδου που παράγονται, διαδίδονται σε νευρώνες του επόμενου επιπέδου μέχρι να καταλήξει η επεξεργασμένη πληροφορία στο στρώμα εξόδου.

Στο στρώμα εξόδου αξιολογούμε το αποτέλεσμα που έχει βγάλει το δίκτυο με το επιθυμητό αποτέλεσμα χρησιμοποιώντας μια συνάρτηση απώλειας C (cost/loss function, πολλές φορές συμβολίζεται και με L), αυτή η συνάρτηση έχει ως είσοδο ένα διάνυσμά a_L το οποίο έχει n διαστάσεις, όπου το n είναι το πλήθος των νευρώνων στο στρώμα εξόδου L του δικτύου, οπότε το συνολικό σφάλμα υπολογίζεται ως $C(a_L)$. Σε αυτό το σημείο χρησιμοποιούμε τον αλγόριθμο που αξιοποιείται σε μεγάλο βαθμό στον κλάδο της μηχανικής μάθησης: τον αλγόριθμο κατάβασης κλήσης (gradient descent). Οπού στην ουσία για τη συνάρτηση απώλειας που έχουμε επιλέξει πρέπει να βρούμε τη μερική παράγωγο για κάθε βάρος w ως προς τη συνάρτηση απώλειας $\frac{\partial C}{\partial w}$ για κάθε είσοδο. Το σύνολο αυτών των παραγώγων το ορίζουμε σαν ένα διάνυσμα και το συμβολίζουμε ως ∇C.

Για να βρούμε το ∇C θα πρέπει να αξιοποιήσουμε τον αλγόριθμο αντίστροφης αναδιάδοσης (back propagation). Επειδή το πλήθος των παραμέτρων σε ένα νευρωνικό δίκτυο είναι πολύ μεγάλο χρειαζόμαστε έναν γρήγορο τρόπο για να υπολογίσουμε των πίνακα των κλήσεων ∇C. Οπότε για ένα βάρος w^l_{jk}, οι τρεις δείκτες που ορίζουμε είναι οι εξής: Ο πρώτος δείκτης είναι l και θα αφορά το στρώμα του νευρώνα του οποίου χρησιμοποιεί το βάρος w στην είσοδό του. Ο δεύτερος δείκτης j καθορίζει στο στρώμα που είμαστε (l), ποιον νευρώνα εξετάζουμε. Και τέλος, ο τρίτος δείκτης είναι ο k, ο οποίος καθορίζει τον νευρώνα του προηγούμενου στρώματος $l-1$ που συνδέεται με τον νευρώνα που εξετάζουμε. Για τις τάσεις, επειδή δεν υπάρχει σύνδεσή με προηγούμενα στρώματα, δε θα χρειαστούμε τον δείκτη k, δηλαδή μας αρκεί ο ορισμός b^l_j. Και το ίδιο ισχύει για την έξοδο του νευρώνα, την οποία που θα ονομάσουμε a^l_j, και οποία περιγράφεται στην εξίσωση 2.1.

$$a_j^l = f(\sum_{k=0}^{n} w_{jk}^l a_k^{l-1} + b_j^l) \tag{2.1}$$

Επειδή υπάρχουν πολλές παράμετροι ανά στρώμα, σε αυτό το σημείο μπορούμε να απλοποιήσουμε τη διαδικασία της εύρεσης των μερικών παραγώγων με τη χρήση πινάκων. Λόγω της δομής των νευρωνικών δικτύων είναι εύκολο να προσαρμόσουμε τις σχέσεις σε πίνακες, και η εξίσωση 2.1 μπορεί να μετασχηματιστεί στην εξίσωση 2.2.

$$a^l = f(w^l a^{l-1} + b^l) \tag{2.2}$$

Στην εξίσωση 2.2 το a^l αναφέρεται στο διάνυσμα των τιμών εξόδου των νευρώνων του l στρώματος, το w^l αναφέρεται στο διάνυσμα των βαρών και το b^l αναφέρεται στο διάνυσμα των τάσεων. Επίσης, είναι καλό να ορίζουμε ένα διάνυσμα z^l το οποίο θα αντιπροσωπεύει τις εξόδους των νευρώνων, πριν χρησιμοποιηθεί η συνάρτηση ενεργοποίησης στο l επίπεδο. Το διάνυσμα αυτό το ορίζουμε ως $z^l = w^l a^{l-1} + b^l$. Ένα πράγμα που θα χρειαστεί να ορίσουμε ακόμα, είναι το σφάλμα του κάθε νευρώνα, το οποίο θα το συμβολίσουμε με δ_j^l. Το σφάλμα αυτό, ορίζεται στην εξίσωση 2.3.

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{2.3}$$

Χρησιμοποιώντας τους ορισμούς 2.3, 2.2 τώρα μπορούμε να περιγράψουμε τις βασικές εξισώσεις που χρησιμοποιεί ο αλγόριθμος της αντίστροφης αναδιάδοσης. Η πρώτη εξίσωση περιγράφει το σφάλμα στο τελευταίο στρώμα, και υπολογίζεται από την εξίσωση 2.4.

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} f'(z_j^L) \tag{2.4}$$

Στη συνάρτηση 2.4 το δ_j^L είναι ένα στοιχείο του διανύσματος δ^L. Οπότε κάνοντας την αντίστοιχη μετατροπή καταλήγουμε στην εξίσωση 2.5, η οποία χρησιμοποιεί μόνο διανύσματα ως μεταβλητές.

$$\delta^L = \nabla_a C \odot f'(z^L) \tag{2.5}$$

Στην εξίσωση 2.5, το $\nabla_a C$ είναι το διάνυσμα που περιέχει τις μερικές παραγώγους του a_j^L ως προς τη συνάρτηση απώλειας.

Η δεύτερη σημαντική εξίσωση είναι η 2.6, και περιγράφει το σφάλμα δ^l σε σχέση με το σφάλμα του επόμενου στρώματος δ^{l+1}.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot f'(z^l) \tag{2.6}$$

Η εξίσωση 2.6 μας δίνει τη δυνατότητα να υπολογίζουμε τα σφάλματα των προηγούμενων στρωμάτων με βάση τα σφάλματα των επόμενων. Δηλαδή, μπορούμε να βρούμε τα σφάλματα δ^L του τελευταίου επιπέδου και να υπολογίσουμε τα σφάλματα του προτελευταίου δ^{L-1}, και μπορούμε να επαναλάβουμε τη διαδικασία αυτή για όλα τα στρώματα του δικτύου, με αποτέλεσμα να βρίσκουμε όλες τις τιμές του δ.

Τέλος, οι τελευταίες δύο εξισώσεις είναι αυτές που μας δίνουν τη δυνατότητα να αυξομειώνουμε τα βάρη και τις τάσεις. Η εξίσωση που αφορά τις τάσεις είναι η 2.7.

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.7}$$

Μπορούμε να παρατηρήσουμε ότι η εξίσωση 2.7 είναι πολύ απλή επειδή αφορά μόνο μια σταθερή τιμή που υπάρχει μια φορά σε έναν νευρώνα. Και τέλος, η εξίσωση για τα βάρη είναι η 2.8.

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{2.8}$$

Έχουμε φτάσει λοιπόν στο σημείο που μπορούμε να βρούμε όλες τις παραγώγους χρησιμοποιώντας τις εξισώσεις 2.5, 2.6, 2.7, 2.8. Σε αυτό το σημείο έχουμε γνωστές όλες τις παραγώγους και τώρα μπορούμε να ξεκινήσουμε να βλέπουμε την επιρροή που έχει το κάθε βάρος και τάση στο σφάλμα για κάποια συγκεκριμένη εικόνα. Οπότε μπορούμε για ένα σύνολο εικόνων να προσδιορίσουμε τον μέσο όρο των παραγώγων για κάθε βάρος και τάση ως προς τη συνάρτηση απώλειας, και στο τέλος αυξάνουμε ή μειώνουμε τα βάρη και τάσεις σύμφωνα με τις τιμές των παραγώγων ώστε να καταφέρουμε να ελαχιστοποιήσουμε το σφάλμα.

Αξίζει να σημειωθεί ότι είναι πολύ σημαντική η επιλογή της συνάρτησης ενερ-

γοποίησης, διότι θα χρησιμοποιηθεί σε κάθε νευρώνα του δικτύου. Οπότε πρέπει να έχει μια παράγωγο που να μην έχει μεγάλο υπολογιστικό κόστος, διότι επηρεάζει την ταχύτητα εκπαίδευσης του δικτύου. Επίσης υπάρχουν συναρτήσεις όπου συμπιέζουν το σύνολο τιμών σε τιμές από $(-\infty, \infty)$ σε $(-1, 1)$ ώστε να μπορέσουν να σταθεροποιήσουν το δίκτυο κατά τη διαδικασία της εκπαίδευσης. Όμως με τη χρήση τέτοιων συναρτήσεων υπάρχει κίνδυνος να αντιμετωπίσουμε το πρόβλημα της φθίνουσας κλήσης (vanishing gradient). Δηλαδή μπορεί η παράγωγος να είναι πολύ μικρή όταν έχουμε μεγάλες τιμές εισόδου με αποτέλεσμα να φτάσει το δίκτυο να βγάζει πάρα πολύ μικρά σφάλματα και να παραμείνει στάσιμο στην εκπαίδευση.

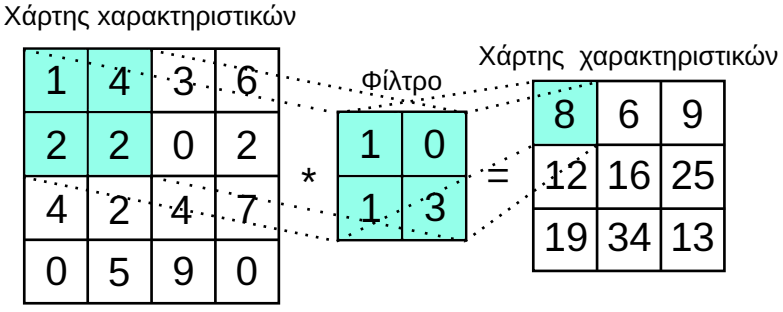
2.2 Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (convolutional neural networks) είναι τα δίκτυα που έχουν τη μεγαλύτερη δημοφιλία για την αντιμετώπιση του προβλήματος της ταξινόμησης εικόνων. Αυτή η κατηγορία νευρωνικών δικτύων καλύπτει πολλές αδυναμίες που έχει ένα απλό δίκτυο που αποτελείται μόνο από πλήρως συνδεδεμένα στρώματα (fully connected layers), διότι τα απλά δίκτυα, αν θέλουμε να τα χρησιμοποιήσουμε για να βρούμε την κλάση μιας εικόνας, η οποία αποτελείται από δυο διαστάσεις, ύψος και πλάτος, πρέπει να μετατρέψουμε μια δισδιάστατη πληροφορία σε μια μονοδιάστατη είσοδο. Αυτό έχει ένα πολύ μεγάλο μειονέκτημα, το οποίο είναι ότι χάνεται η τοπικότητα της πληροφορίας. Δηλαδή εκεί που μπορούσαμε να συγκρίνουμε ένα εικονοστοιχείο (pixel) με τα εικονοστοιχεία που το περικύκλωναν για να δούμε αν υπήρχε κάποιο μοτίβο, όπως η γωνία ενός αντικειμένου, τώρα απλά χάνουμε αυτή τη δυνατότητα. Ένα δεύτερο μειονέκτημα των απλών δικτύων είναι ότι είναι πολύ ευαίσθητα στη θέση του αντικειμένου. Δηλαδή αν ένα αντικείμενο μετακινηθεί σε οποιαδήποτε από τις δύο διαστάσεις υπάρχει μεγάλη πιθανότητα να μην μπορεί να το αναγνωρίσει το δίκτυο. Και το τελευταίο μειονέκτημα είναι ότι μια εικόνα περιέχει ένα πολύ μεγάλο όγκο πληροφοριών. Ακόμα και σε μια απλή εικόνα που μπορεί να έχει μια χαμηλή ανάλυση που αποτελείται από 256×256 εικονοστοιχεία, και το κάθε εικονστοιχείο είναι ένα διάνυσμα τριών τιμών για τα χρώματα μπλε, πράσινο και κόκκινο, καταλήγουμε να έχουμε 200000 παραμέτρους μόνο για μια εικόνα. Οπότε αυτήν αν τη συνδέσουμε με ένα στρώμα που αποτελείται έστω από 32 νευρώνες τότε θα χρειαστεί να εκπαιδεύ-

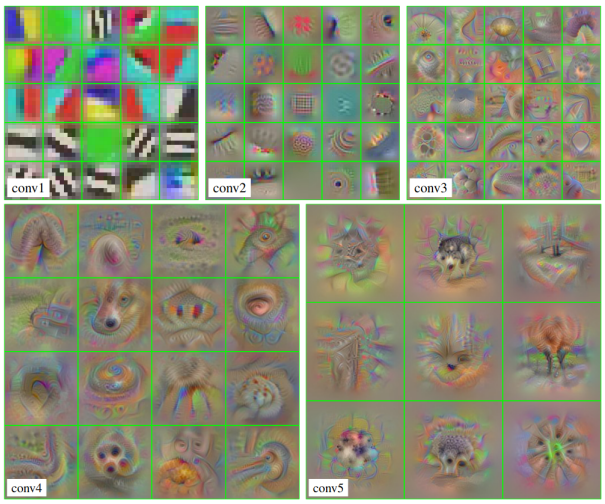
σουμε 6400000 βάρη μόνο για το πρώτο στρώμα. Και αν αρχίσουμε αυτές τις τιμές να τις διαδίδουμε σε στρώματα μεγαλύτερου βάθους, θα καταλήξουμε πολύ γρήγορα σε ένα δίκτυο που είναι πάρα πολύ αργό στην εκπαίδευση.

Όλα τα παραπάνω προβλήματα μας τα λύνει ένα συνελικτικό δίκτυο, το οποίο μειώνει τις παραμέτρους που χρειάζονται στην εκπαίδευση και κρατάει την πληροφορία της τοπικότητας, χρησιμοποιώντας μια καινούργια τεχνική που κάνει χρήση των φίλτρων (filters). Αυτά τα φίλτρα στην ουσία είναι ένας δισδιάστατος πίνακάς στον οποίο η κάθε τιμή που περιέχει είναι ένα βάρος. Αυτόν τον πίνακα τον πολλαπλασιάζουμε στοιχείο προς στοιχείο με τη σχετική περιοχή στην εικόνα και καταγράφουμε το αποτέλεσμά σε ένα καινούργιο στρώμα που ονομάζεται χάρτης χαρακτηριστικών (feature map). Μετά μετακινούμε αυτόν τον πίνακα για όλες τις πιθανές περιοχές με ένα συγκεκριμένο βήμα (stride). Ουσιαστικά είναι παρόμοια τεχνική με αυτή του ολισθόμενου παραθύρου. Αυτή η διαδικασία απεικονίζεται στο Σχήμα 2.3. Επίσης για κάθε στρώμα μπορούμε να επιλέξουμε πόσα φίλτρα θέλουμε ώστε να μπορέσουμε να ερμηνεύσουμε το αντίστοιχο πλήθος των μοτίβων. Τέλος, αυτά τα φίλτρα μπορούμε να τα εφαρμόσουμε πάνω σε χάρτες χαρακτηριστικών που έχουν παράξει τα φίλτρα ενός προηγούμενου επιπέδου. Δηλαδή αρχίζουμε και ψάχνουμε μοτίβα πάνω στα μοτίβα. Αυτό μας βοηθάει στο να εντοπίζουμε χαρακτηριστικά με μεγαλύτερη εννοιολογική σημασία όσο αυξάνουμε το βάθος του δικτύου. Αυτή η αύξηση εννοιολογικής σημασίας ανάλογα με το βάθος απεικονίζεται στο Σχήμα 2.4, όπου αρχικά εμφανίζονται χαρακτηριστικά χαμηλού επιπέδου, και όσο βαθύτερα προχωράμε σε στρώματα, αυτά συνδυάζονται και παράγουν μοτίβα, και μετά από αυτά παράγονται σχήματα που θυμίζουν αντικείμενα ή μέρη αντικειμένων.

Σχήμα 2.3: Προσπέλαση φίλτρου για μία δισδιάστατη είσοδο



Σχήμα 2.4: Τιμές αρχικής εικόνας που μεγιστοποιούν την έξοδο των φίλτρων στο αντίστοιχο επίπεδο [18, Σχήμα 16]

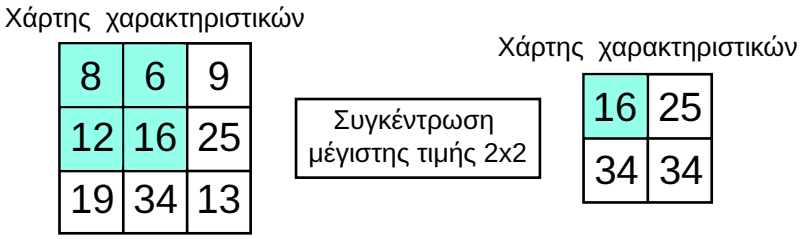


Ακόμα μια βελτιστοποίηση που κάνουν τα συνελικτικά νευρωνικά δίκτυα είναι η χρήση των στρωμάτων συγκέντρωσης (pooling layers). Αυτά τα στρώματα έχουν κύριο σκοπό τη μείωση των παραμέτρων του δικτύου ώστε να μπορούμε να εκπαιδεύσουμε αυτά τα δίκτυα πιο γρήγορα. Τα στρώματα συγκέντρωσης εφαρμόζονται πάνω στους χάρτες χαρακτηριστικών με μια παρόμοια λογική με τα φίλτρα, αλλά έχουν συνήθως μεγαλύτερο βήμα ώστε να μειωθούν οι παράμετροι σε ένα μεγαλύτερο βαθμό. Δηλαδή μπορούμε για μια περιοχή πάνω στον χάρτη χαρακτηριστικών να εφαρμόσουμε μια συνάρτηση που βρίσκει το στοιχείο με τη μέγιστη τιμή (όπως φαίνεται στο Σχήμα 2.5), ή μπορούμε να χρησιμοποιήσουμε τη μέση τιμή όλων των στοιχείων. Αυτή η απλοποίηση που προσθέτουν τα επίπεδα συγκέντρωσης δεν επηρεάζει σημαντικά την ακρίβεια, γιατί δε χρειαζόμαστε πολλές παραμέτρους για να εξάγουμε την εννοιολογική σημασία κάποιας περιοχής πάνω στον χάρτη χαρακτηριστικών. Παράλληλα αυτή η απλοποίησή μας βοηθάει στο να προσθέσουμε περισσότερα ενδιάμεσα στρώματα· αυξάνοντας το βάθος του δικτύου με μικρότερο υπολογιστικό κόστος.

Το κύριο πλεονέκτημα της μείωσης των παραμέτρων του δικτύου προέρχεται από την επαναχρησιμοποίηση των βαρών του κάθε φίλτρου σε διαφορετικές εισόδους και χάρτες χαρακτηριστικών. Οι κανόνες της αντίστροφης αναδιάδοσης είναι παρόμοιοι, αλλά με κάποιες αλλαγές για να μπορέσουν να χρησιμοποιηθούν οι βελτιστοποιήσεις που προσφέρουν τα φίλτρα και τα στρώματα συγκέντρωσης.

Η εφαρμογή της συνέλιξης στα νευρωνικά δίκτυα για μια είσοδο και ένα φίλτρο

Σχήμα 2.5: Παράδειγμα της τεχνικής ενός στρώματος συγκέντρωσης



που έχει διαστάσεις $k_1 \times k_2$ και με τάση b, περιγράφεται από την εξίσωση 2.9.

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^{C} K_{m,n,c} \cdot I_{i+m,j+n,c} + b \tag{2.9}$$

Το C στην εξίσωση 2.9 αντιπροσωπεύει το πλήθος των καναλιών που έχει το προηγούμενο στρώμα. Για την απλοποίηση των εξισώσεων δε θα χρησιμοποιηθεί παρακάτω. Επίσης, παρόλο που ο αλγόριθμος αντίστροφης αναδιάδοσης χρησιμοποιεί διαφορετικά σύμβολα στις παραμέτρους, επειδή αφορά δύο διαστάσεις, θα χρησιμοποιήσουμε τα ίδια σύμβολα με την προηγούμενη ενότητα για να μπορέσουμε να χαρακτηρίσουμε καλύτερα τους παρόμοιους τρόπους λειτουργίας των δύο διαφορετικών τύπων δικτύου. Οπότε το αποτέλεσμα ενός νευρώνα πριν εφαρμόσουμε τη συνάρτηση ενεργοποίησης θα το ορίσουμε με την εξίσωση 2.10.

$$z_{i,j}^{l} = \sum_{m} \sum_{n} w_{m,n}^{l} a_{i+m,j+1}^{l} - 1 + b^{l} \tag{2.10}$$

Οπού το a στην εξίσωση 2.10 είναι η έξοδος του νευρώνα και ορίζεται ως $a_{i,j} = f(x_{i,j}^{l})$, όπου το f είναι η συνάρτηση ενεργοποίησης.

Ομοίως με τα απλά δίκτυα ο σκοπός μας είναι να βρούμε το πόσο επηρεάζει ένα βάρος τη συνάρτηση σφάλματος. Οπότε για μια τιμή του φίλτρου w θέλουμε να βρούμε την εξής μερική παράγωγο: $\frac{\partial C}{\partial w_{m',n'}^{l}}$. Επίσης, για μια είσοδο με διαστάσεις $H \times W$, και ένα φίλτρο διαστάσεων $k_1 \times k_2$ τότε ο χάρτης χαρακτηριστικών εξόδου που θα παραχθεί θα είναι διαστάσεων $(H - k_1 + 1) \times (W - k_2 + 1)$. Τέλος, η εξίσωση που μας δίνει τη μερική παράγωγο της συνάρτησης απώλειας ως προς ένα στοιχείο του πίνακα βάρους είναι η 2.11.

$$\frac{\partial C}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{i,j} \frac{\partial z^l_{i,j}}{\partial w^l_{m',n'}} \tag{2.11}$$

Στην εξίσωση 2.11 το σφάλμα δ ορίζεται στη συνάρτηση 2.12.

$$\delta^l_{i,j} = \frac{\partial C}{\partial z^l_{i,j}} \tag{2.12}$$

Αξίζει να σημειωθεί ότι στην εξίσωση 2.11, υπάρχουν πολλά αθροίσματα για το ίδιο βάρος. Οπότε μπορούμε να δούμε το πλεονέκτημα των συνελικτικών δικτύων όπου καταφέρνει να μειώσει τις παραμέτρους του δικτύου, μοιράζοντας τα ίδια βάρη.

Σε αυτό το σημείο μπορούμε να περιγράψουμε την αναλογία $\frac{\partial z^l_{i,j}}{\partial w^l_{m',n'}}$ που υπάρχει στην εξίσωση 2.11, χρησιμοποιώντας τη συνάρτηση 2.10 με την εξίσωση 2.13.

$$\frac{\partial z^l_{i,j}}{\partial w^l_{m',n'}} = \frac{\partial}{\partial w^l_{m',n'}} (\sum_m \sum_n w^l_{m,n} a^{l-1}_{i+m,j+n} + b^l) \tag{2.13}$$

Και αν επεκτείνουμε τις αθροιστικές ακολουθίες στην εξίσωση 2.13 μπορούμε να δούμε ότι υπάρχουν πολλές απαλοιφές, με αποτέλεσμα να έχουμε μια αρκετά απλοποιημένη εξίσωση 2.14.

$$\begin{aligned} \frac{\partial z^l_{i,j}}{\partial w^l_{m',n'}} &= \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{0,0} a^{l-1}_{i+0,j+0} + \cdots + w^l_{m',n'} a^{l-1}_{i+m',j+n'} + \cdots + b^l \right) \\ &= \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{m',n'} a^{l-1}_{i+m',j+n'} \right) \\ &= a^{l-1}_{i+m',j+n'} \end{aligned} \tag{2.14}$$

Οπότε μπορούμε να αντικαταστήσουμε την εξίσωση 2.14 στην εξίσωση 2.11 και να καταλήξουμε στην εξίσωση 2.15.

$$\frac{\partial C}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{i,j} a^{l-1}_{i+m',j+n'} \tag{2.15}$$

Οπότε ακολουθώντας τους κανόνες της αλυσιδωτής παραγώγισης μπορούμε να επεκτείνουμε τον ορισμό του δ και να τον χαρακτηρίσουμε με την εξίσωση 2.16.

$$\delta^l_{i',j'} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} \frac{\partial z^{l+1}_{i'-m,j'-n}}{\partial z^l_{i',j'}} \tag{2.16}$$

Σε αυτό το σημείο μένει να βρούμε την εξίσωση που περιγράφει τη σχέση $\frac{\partial z^{l+1}_{i'-m,j'-n}}{\partial z^{l}_{i',j'}}$. Αυτό μπορούμε να το πετύχουμε βρίσκοντας τη μερική παράγωγο της εξίσωσης 2.10. Για να το καταφέρουμε αυτό, χρησιμοποιούμε την εξίσωση 2.17.

$$\begin{aligned}\frac{\partial z^{l+1}_{i'-m,j'-n}}{\partial z^{l}_{i',j'}} &= \frac{\partial}{\partial z^{l}_{i',j'}}\left(\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} a^{l}_{i'-m+m',j'-n+n'} + b^{l+1}\right) \\ &= \frac{\partial}{\partial z^{l}_{i',j'}}\left(\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} f\left(z^{l}_{i'-m+m',j'-n+n'}\right) + b^{l+1}\right)\end{aligned} \tag{2.17}$$

Αν επεκτείνουμε τα αθροίσματα στην εξίσωση 2.17 θα δούμε ότι υπάρχουν πολλές απαλοιφές, και ότι μένουν μόνο τα βάρη για τα οποία ισχύει $m' = m$ και $n' = n$, άρα καταλήγουμε στην εξίσωση 2.18.

$$\begin{aligned}\frac{\partial z^{l+1}_{i'-m,j'-n}}{\partial z^{l}_{i',j'}} &= \frac{\partial}{\partial z^{l}_{i',j'}}\left(w^{l+1}_{m',n'} f\left(z^{l}_{0-m+m',0-n+n'}\right) + \cdots + w^{l+1}_{m,n} f\left(z^{l}_{i',j'}\right) + \cdots + b^{l+1}\right) \\ &= \frac{\partial}{\partial z^{l}_{i',j'}}\left(w^{l+1}_{m,n} f\left(z^{l}_{i',j'}\right)\right) \\ &= w^{l+1}_{m,n}\frac{\partial}{\partial z^{l}_{i',j'}}\left(f\left(z^{l}_{i',j'}\right)\right) \\ &= w^{l+1}_{m,n} f'\left(z^{l}_{i',j'}\right)\end{aligned} \tag{2.18}$$

Οπότε μπορούμε να αντικαταστήσουμε την εξίσωση 2.18 στην εξίσωση 2.16 και καταλήγουμε στη σχέση 2.19.

$$\delta^{l}_{i',j'} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} w^{l+1}_{m,n} f'\left(z^{l}_{i',j'}\right) \tag{2.19}$$

Αφού καταλήξαμε λοιπόν στην εξίσωση 2.19 μπορούμε να τη χρησιμοποιήσουμε για να εκτελέσουμε τα βήματά της αντίστροφης αναδιάδοσης με παρόμοιο τρόπο που εκτελείται και στα απλά νευρωνικά δίκτυα. Διότι μας δίνει την ίδια ιδιότητα να διαδίδουμέ το σφάλμα προς τα πίσω με σχετικά χαμηλό υπολογιστικό κόστος.

Αξίζει να αναφερθεί ότι τα στρώματα συγκέντρωσης δε χρησιμοποιούν παραμέτρους τις οποίες βρίσκουμε μέσω της εκπαίδευσης του δικτύου, οπότε δε συμμετέχουν με κάποιον σημαντικό βαθμό στη διαδικασία της αντίστροφης αναδιάδοσης. Στην περίπτωση που υπάρχουν επίπεδα συγκέντρωσης μέγιστου (max pooling) τότε απλά βρίσκουμε το σφάλμα μόνο στις παραμέτρους που συμβάλουν στη μέγιστη

τιμή, και στα στρώματα συγκέντρωσης μέσου όρου διαιρούμε απλά το σφάλμα με το πλήθος των στοιχείων που υπάρχει στο φίλτρο του στρώματος συγκέντρωσης.

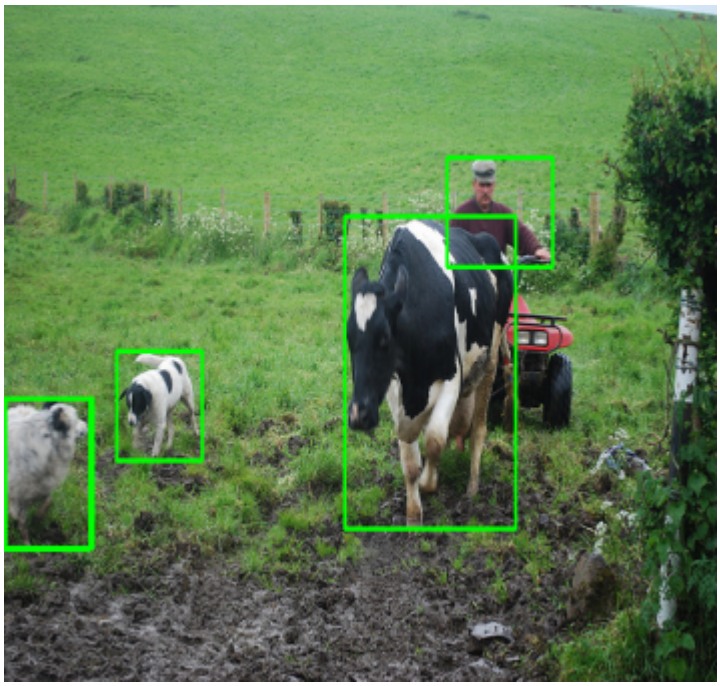
2.3 Ανίχνευση Αντικειμένων

Η ανίχνευση αντικειμένων στον κλάδο της μηχανικής μάθησης έχει ως σκοπό, όχι μόνο να αναγνωρίσει μια εικόνα και το τι αντικείμενο περιέχει, αλλά και να βρει τις συντεταγμένες του. Για την περιγραφή της τοποθεσίας του αντικειμένου πάνω στην εικόνα, οι συντεταγμένες που χρησιμοποιούνται είναι αυτές των γωνιών ενός ορθογώνιου παραλληλόγραμμου, όπου η κάθε του ακμή θέτει τα όρια του αντικείμενου. Αυτό το παραλληλόγραμμο ονομάζεται περιβάλλων κουτί (bounding box) και φαίνεται στο Σχήμα 2.6. Αυτή η επιπλέον πληροφορία της τοποθεσίας είναι πολύ χρήσιμη γιατί έχει πολλές εφαρμογές στα προβλήματα που αντιμετωπίζουμε στην καθημερινότητα. Για παράδειγμα, μπορούμε με αυτήν την επιπρόσθετη πληροφορία να ανιχνεύσουμε τις θέσεις που έχει το κάθε εμπόδιο στον δρόμο ενός αυτοκινήτου. Ή ακόμα να δούμε σε ποια σημεία μιας ακτινογραφίας εμφανίζονται σημάδια κάποιας πάθησης.

Για την εύρεση αυτών των συντεταγμένων δεν έχει καθιερωθεί μια κύρια μεθοδολογία με βάση της οποίας θα εκπαιδεύεται το δίκτυο, είναι ένας κλάδος της μηχανικής μάθησης που ακόμα ερευνάται. Έχουν υπάρξει διαφορετικές μεθοδολογίες για την εύρεση των περιβαλλόντων κουτιών. Οι αρχικές υλοποιήσεις χρησιμοποιούσαν ένα εκπαιδευμένο συνελικτικό δίκτυο και το διαπερνούσαν στην εικόνα πολλές φορές. Και για τα σημεία που υπήρχε μεγάλη αυτοπεποίθηση, τότε καθόριζαν ότι εκείνη η περιοχή περιείχε ένα αντικείμενο σε εκείνες τις συντεταγμένες. Αργότερα δημιουργήθηκαν μοντέλα όπως το R-CNN [7] όπου είχαν ως κύρια μεθοδολογία τη χρήση ενός εξωτερικού εργαλείου όπως το Selective Search [29] για να εντοπίσουν μια περιοχή που έχει μια μεγάλη πιθανότητα να περιέχει ένα αντικείμενο. Το επόμενο βήμα εξέλιξης ήταν η χρήση των συνελικτικών νευρωνικών δικτύων για την εύρεση των περιβάλλοντων κουτιών με αποτέλεσμα να αυξηθεί η ταχύτητα εντοπισμού, πολλές φορές σε ένα τέτοιο βαθμό που να μπορούμε να τα χρησιμοποιήσουμε σε πραγματικό χρόνο.

Επειδή τα μοντέλα που έχουν φτιαχτεί για να επιλύσουν το πρόβλημα της εύρεσης αντικειμένων εξελίσσονται με ραγδαίους ρυθμούς τον τελευταίο καιρό, είναι

Σχήμα 2.6: Περιβάλλοντα κουτιά σε μια εικόνα



απαραίτητο να μπορέσουμε να βρούμε κάποια μετρικά ώστε να μπορέσουμε να τα συγκρίνουμε. Στον κλάδο της μηχανικής μάθησης, ο συνηθέστερος τρόπος για να αξιολογηθεί ένας αλγόριθμος ταξινόμησης είναι με τη χρήση του πίνακα σύγχυσης (confusion matrix) όπου έστω ότι στο πρόβλημα ταξινόμησης που έχουμε n κλάσεις, τότε οι διαστάσεις αυτού του πίνακα θα είναι $n \times n$, η μία διάσταση αντιπροσωπεύει τις πραγματικές τιμές και η άλλη περιέχει τις τιμές που έχει προβλέψει το μοντέλο, κάθε στοιχείο του πίνακα περιέχει το πλήθος των αντίστοιχων στοιχείων που προβλέψαμε. Το Σχήμα 2.7 περιγράφει τη δομή ενός πίνακα σύγχυσης.

Ο πίνακας σύγχυσης έχει τέσσερις διαφορετικούς χαρακτηρισμούς για τα αποτελέσματα που μπορεί να παράξει ένα μοντέλο: Τα αληθώς θετικά (Αθ, True positive) ονομάζουμε τα στοιχεία στα οποία η κλάση που εξετάζουμε είναι ίση με την πραγματική και την προβλεπόμενη κλάση. Τα αληθώς αρνητικά (ΑΑ, True Negative) είναι τα στοιχεία για τα οποία η προβλεπόμενη και η πραγματική κλάση είναι διαφορετική από την οποία εξετάζουμε. Τα ψευδώς Θετικά (ΨΘ, False Positive) είναι τα στοιχεία στα οποία η προβλεπόμενη κλάση και η κλάση που εξετάζουμε είναι ίδιες, αλλά η πραγματική κλάση είναι διαφορετική. Και τέλος τα ψευδός αρνητικά (ΨΑ, False Negative) είναι τα στοιχεία στα οποία η κλάση που εξετάζουμε και η πραγματική κλάση είναι ίδιες, αλλά η προβλεπόμενη κλάση είναι διαφορετική.

Από τον πίνακα σύγχυσης μπορούμε να αντλήσουμε πολλά μετρικά χρησιμο-

Σχήμα 2.7: Πίνακας σύγχυσης για τέσσερις κλάσεις, και οι χαρακτηρισμοί των στοιχείων όταν εξετάζουμε προς την κλάση 2

		Προβλεπόμενες Κλάσεις			
		Κλάση 1	Κλάση 2	Κλάση 3	Κλάση 4
Πραγματικές Κλάσεις	Κλάση 1	Αληθώς Αρνητικό	Ψευδός Θετικό	Αληθώς Αρνητικό	Αληθώς Αρνητικό
	Κλάση 2	Ψευδός Αρνητικό	Αληθώς Θετικό	Ψευδός Αρνητικό	Ψευδός Αρνητικό
	Κλάση 3	Αληθώς Αρνητικό	Ψευδός Θετικό	Αληθώς Αρνητικό	Αληθώς Αρνητικό
	Κλάση 4	Αληθώς Αρνητικό	Ψευδός Θετικό	Αληθώς Αρνητικό	Αληθώς Αρνητικό

ποιώντας τις αναλογίες του πλήθος των στοιχείων. Τα μετρικά στα οποία θα επικεντρωθούμε εμείς είναι η ακρίβεια (precision) και η ανάκληση (recall). Η ακρίβεια είναι ένα μετρικό που υπολογίζεται από την εξίσωση 2.20.

$$\text{Ακρίβεια} = \frac{\text{ΑΘ}}{\text{ΑΘ} + \Psi\text{Θ}} \tag{2.20}$$

Το μετρικό της ακρίβειας στην ουσία περιγράφει για την κλάση που προβλέψαμε, τι ποσοστό τιμών έχουμε προβλέψει σωστά.

Η ανάκληση περιγράφεται με την εξίσωση 2.21.

$$\text{Ανάκληση} = \frac{\text{ΑΘ}}{\text{ΑΘ} + \Psi\text{Α}} \tag{2.21}$$

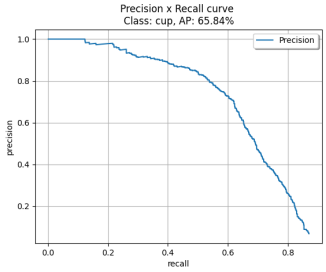
Η ανάκληση μας δίνει την εικόνα, για το πόσα στοιχεία έχουμε προβλέψει σωστά για την εξεταζόμενη κλάση ως προς τα πόσα πραγματικά στοιχεία υπάρχουν για τη συγκεκριμένη κλάση. Αυτό το μετρικό είναι χρήσιμο για να σιγουρέψουμε ότι ο αλγόριθμος ο οποίος εξετάζεται δεν είναι υπερβολικά αυστηρός, με αποτέλεσμα να ταξινομεί σωστά μόνο ένα μικρό ποσοστό των πραγματικών κλάσεων.

Για να μπορέσουμε να αξιολογήσουμε το μοντέλο χρειαζόμαστε και τα δύο μετρικά. Δε θέλουμε ένα μοντέλο να βρίσκει πολλά θετικά αποτελέσματα, με το να χαρακτηρίζει τα περισσότερα δείγματα ως θετικά, και ούτε θέλουμε να βρίσκει

μόνο ένα θετικό ενώ υπάρχουν πολλά. Το πόσο μας ενδιαφέρει η μια παράμετρός σε σχέση με την άλλη εξαρτάται από το πρόβλημα που αντιμετωπίζει το μοντέλο.

Για το πρόβλημα της εύρεσης αντικειμένων το κύριο μετρικό που έχει καθιερωθεί είναι η μέση ακρίβεια (average precision). Η μέση ακρίβεια υπολογίζεται από το εμβαδόν της καμπύλης ανάκλησης-ακρίβειας (precision-recall curve). Η καμπύλη αυτή μπορεί να υπολογιστεί όταν ο αλγόριθμος μηχανικής μάθησης που εξετάζουμε περιέχει και έναν δείκτη αυτοπεποίθησης για την πρόβλεψη που έχει κάνει. Οπότε χρησιμοποιώντας αυτόν τον δείκτη, πρώτα επιλέγουμε τη μέγιστή του τιμή και καταγράφουμε την ακρίβεια και ανάκληση μόνο για όσα στοιχεία έχουν ταξινομηθεί με τιμή του δείκτη αυτοπεποίθησης μεγαλύτερη ή ίση με την τιμή που έχουμε επιλέξει. Για πολύ μεγάλες τιμές στον δείκτη αυτοπεποίθησης η ακρίβεια είναι μεγάλη αλλά η ανάκληση είναι μικρή. Μετά σταδιακά μειώνουμε το όριο αυτοπεποίθησης και μετράμε τα αποτελέσματα μέχρι να φτάσει η ανάκληση τη μέγιστη τιμή της, με αποτέλεσμα να καταλήγουμε σε ένα γράφημα, παρόμοιο με αυτό του Σχήματος 2.8. Το γράφημα αυτό όμως αντιπροσωπεύει τα μετρικά για μία μόνο κλάση. Για να έχουμε μια γενικότερη εικόνα μπορούμε να πάρουμε τον μέσο όρο των εμβαδών των γραφημάτων για κάθε κλάση και να δημιουργήσουμε το μετρικό του μέσου όρου της μέσης ακρίβειας (mAP) (mean average precision).

Σχήμα 2.8: Καμπύλη ανάκλησης-ακρίβειας



Για να μπορέσουμε να δημιουργήσουμε το γράφημα αυτό, θα πρέπει να ορίσουμε το όριο της τιμής αυτοπεποίθησης ως τ, οπότε όταν η τιμή αυτοπεποίθησης έχει τιμή μικρότερη του τ, τότε θα θεωρήσουμε ότι δεν έχει προσδιοριστεί η συγκεκριμένη κλάση. Η συνάρτηση $Pr(\tau)$ πού ορίζεται από τη εξίσωση 2.22 είναι η ακρίβεια για το συγκεκριμένο όριο αυτοπεποίθησης και αντίστοιχα η συνάρτηση $Rc(\tau)$, που ορίζεται από την εξίσωση 2.23 είναι η ανάκληση που υπολογίζεται για το συγκεκριμένο όριο.

$$Pr(\tau) = \frac{\sum_{n=1}^{S} {}_n(\tau)}{\sum_{n=1}^{S} {}_n(\tau) + \sum_{n=1}^{N-S} {}_n(\tau)} \tag{2.22}$$

$$Rc(\tau) = \frac{\sum_{n=1}^{S} {}_n(\tau)}{\sum_{n=1}^{S} {}_n(\tau) + \sum_{n=1}^{N-S} {}_n(\tau)} \tag{2.23}$$

Επειδή όμως τα σημεία (Pr, Rc) έχουν μια ιδιαιτερότητα να εμφανίζουν ανεβοκατεβάσματα στην καμπύλη όταν αφορούν πραγματικά δεδομένα λόγω θορύβου των αποτελεσμάτων, θέλουμε για τον υπολογισμό του AP να εγγυηθούμε ότι η καμπύλη της συνάρτησης για την οποία θα υπολογίσουμε το εμβαδόν είναι μονοτονική. Για να το πετύχουμε αυτό, για τον δείκτη αυτοπεποίθησης τ θα ορίσουμε μια μεταβλητή k που να ικανοποιεί την εξίσωση 2.24.

$$\tau(k), k = 1, 2, ..., K \quad \text{ώστε} \quad \tau(i) > \tau(j) \quad \text{για} \quad i > j \tag{2.24}$$

Επίσης, για κάθε τιμή ανάκλησης R_r που θα επιλέγουμε στον οριζόντιο άξονα του γραφήματος, θα πρέπει να ορίσουμε ένα δείκτη n που να ικανοποιεί τη σχέση 2.25.

$$R_r(n), n = 1, 2, ..., N \quad \text{ώστε} \quad R_r(m) > R_r(n) \quad \text{για} \quad m > n \tag{2.25}$$

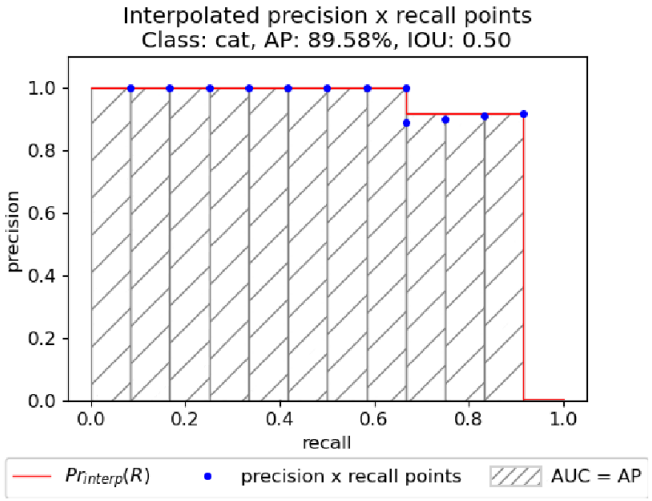
Τέλος, η τελευταία μεταβλητή που χρειαζόμαστε για να ορίσουμε τη μέση ακρίβεια είναι η $Pr_{interp}(R)$, που στην ουσία είναι μια βοηθητική τιμή ακρίβειας ώστε να μπορέσουμε να δημιουργήσουμε μια μονοτονική καμπύλη. Αυτή η τιμή ορίζεται με την εξίσωση 2.26.

$$Pr_{interp}(R) = \max_{k|Rc(\tau(k))\geq R} \{Pr(\tau(k))\} \tag{2.26}$$

Σε αυτό το σημείο, αφού έχουμε καταλήξει σε μια μονοτονική συνάρτηση μπορούμε να ορίσουμε τη μέση ακρίβεια ως το εμβαδόν της περιοχής που βρίσκεται στη κάτω μεριά της καμπύλης που παράγει η εξίσωση 2.27. Με αποτέλεσμα να καταλήγουμέ σε ένα γράφημα παρόμοιο με το Σχήμα 2.9.

$$AP = \sum_{k=0}^{K} (R_r(k) - Rr(k+1)) Pr_{interp}(R_r(k)) \tag{2.27}$$

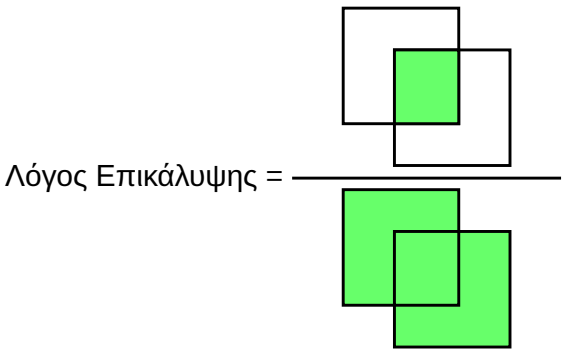
Σχήμα 2.9: Υπολογισμός μέσης ακρίβειας [20, Σχήμα 6]



Οπότε έχουμε καταφέρει να ορίσουμε το βασικότερο μετρικό για το οποίο θα συγκρίνουμε τα μοντέλα στα επόμενα κεφάλαια.

Το πρόβλημα της εύρεσης αντικειμένων δεν περιλαμβάνει μόνο την εύρεση των κλάσεων αλλά και την τοποθεσία τους στην εικόνα. Πρέπει λοιπόν να να προσαρμόσουμε το μετρικό της μέσης ακρίβειας για να καλύπτει αυτόν τον επιπλέον παράγοντα, ο οποίος είναι τα περιβάλλοντα κουτιά. Για τη λύση αυτού του προβλήματος τα μετρικά εκτίμησης απόδοσης του COCO (COCO Evaluation Metrics) [3] και τα μετρικά του Pascal VOC [5] χρησιμοποιούν τις ανιχνεύσεις που έχουν ένα συγκεκριμένο λόγο επικάλυψης. Ο λόγος επικάλυψης, η αλλιώς IOU (intersection over union) ορίζεται ως το εμβαδόν της τομής του πραγματικού περιβάλλοντος κουτιού με του προβλεπόμενου περιβάλλοντος κουτιού ως προς το εμβαδόν της ένωσης τους. Αυτός ο λόγος φαίνεται στο Σχήμα 2.10.

Σχήμα 2.10: Λόγος επικάλυψης



Για να μπορέσουν να αξιολογήσουν τα μοντέλα, οι δημιουργοί του συνόλου δεδομένων COCO έχουν χρησιμοποιήσει μια πληθώρα μετρικών ώστε να μπορέσουν να

συγκρίνουν τις δυνατότητες του μοντέλου σε διαφορετικές περιπτώσεις. Τα μετρικά που έχουν χρησιμοποιηθεί είναι: AP όπου είναι ο μέσος όρος της μέσης ακρίβειας για τα IOU από το 0.5 έως το 0.95 με βήμα 0.5, το AP_{50} και AP_{75} είναι η μέση ακρίβεια για τα IOU 0.50 και 0.75, αντίστοιχα, και το $AP_{small}, AP_{medium}, AP_{large}$ είναι οι τιμές της μέσης ακρίβειας που χαρακτηρίζονται με βάση το εμβαδόν της περιοχής των πραγματικών περιβαλλόντων κουτιών. Ο δείκτης $small$ αναφέρεται σε περιοχές μικρότερες των 32^2 εικονοστοιχείων, ο δείκτης $medium$ αναφέρεται σε περιοχές μεταξύ 32^2 και 96^2 εικονοστοιχείων, και τέλος, ο δείκτης $large$ αναφέρεταί σε περιοχές που είναι μεγαλύτερες των 96^2 εικονοστοιχείων.

Επίσης, το COCO έχει ένα επιπλέον μετρικό, τη μέση ανάκληση (AR), η οποία υπολογίζεται από τη σχέση 2.28.

$$AR = \frac{1}{O} \sum_{o=1}^{O} \max_{k|Pr_{t(o)}(\tau(k))>0} \{Rc_{t(o)}(\tau(k))\} \tag{2.28}$$

Το O στη συνάρτηση 2.28 είναι το πλήθος των λόγων επικάλυψης για τον οποίο μετράμε τη μέση ακρίβεια. Στην ουσία, για να υπολογίσουμε το μετρικό της μέσης ανάκλησης, αρκεί να πάρουμε τον μέσο όρο των μεγαλύτερων τιμών ανακλήσεων για κάθε ποσοστό επικάλυψης.

Τα μετρικά που χρησιμοποιεί το PASCAL VOC για την αξιολόγηση των μοντέλων είναι πιο απλά, αλλά διαφορετικά από τα μετρικά που χρησιμοποιεί το COCO και για αυτόν τον λόγο έχουν συμπεριληφθεί στις αξιολογήσεις των μοντέλων. Το PASCAL υπολογίζει τη μέση ακρίβεια αυστηρά για τις ανιχνεύσεις που έχουν IOU μεγαλύτερο του 0.5, και τις χωρίζει ανά κλάση, οπότε μπορούμε να δούμε πιο αναλυτικά αν είναι πιο δύσκολο να εντοπίσουμε συγκεκριμένες κατηγορίες αντικειμένων.

Κεφάλαιο 3

Θεμελιώδη Νευρωνικά Δίκτυα

3.1 Εισαγωγή

Θεμελιώδη νευρωνικά δίκτυα συνήθως ονομάζουμε κάποιες αρχιτεκτονικές νευρωνικών δικτύων όπου χρησιμοποιούνται από άλλα, πιο σύνθετα νευρωνικά δίκτυα. Αυτά τα δίκτυα έχουν ανεπίσημα τρεις κατηγοριοποιήσεις η οποίες είναι: Η ραχοκοκαλιά (backbone) που χρησιμοποιείται για την εξαγωγή χαρακτηριστικών. Ο λαιμός (neck) που ο κύριος του σκοπός είναι να συνδυάζει τους χάρτες χαρακτηριστικών από διαφορετικά στάδια του δικτύου. Και τέλος, είναι η κεφαλή (head), η οποία κάνει τις προβλέψεις στους χάρτες χαρακτηριστικών και εξάγει τα αποτελέσματα. Ένα δίκτυο δεν είναι απαραίτητο να χρησιμοποιεί αυτές τις δομές, αυτή η ορολογία απλά μας βοηθάει να καταλάβουμε πιο εύκολα τον λόγο της ύπαρξης κάποιας δομής στα μοντέλα που θα εξετάσουμε. Αυτές οι δομές πολλές φορές ξεκίνησαν ως ανεξάρτητα ολοκληρωμένα μοντέλα τα οποία χειριζόντουσαν όλες τις λειτουργίες ενός νευρωνικού δικτύου, αλλά πολλοί συγγραφείς έχουν δημιουργήσει μοντέλα με πιο σύνθετη λογική από ένα δίκτυο που αποτελείται μόνο από συνελικτικά στρώματα (convolutional layers), και πολλές φορές αυτά τα δίκτυα μπορούν να εκπαιδευτούν ξεχωριστά και να προστεθούν σε ένα μεγαλύτερο δίκτυο το οποίο θα περιέχει την πιο σύνθετη λογική.

3.2 VGG

Το VGG [25] είναι ένα από τα πρώτα παραδείγματα μιας αποτελεσματικής δομής, η απλότητα της και το μικρό βάθος που έχει κάνει αυτή τη δομή πιο εύχρηστη στη συμπερίληψη της σε άλλα μοντέλα. Ο συγγραφέας αυτής της αρχιτεκτονικής

παρατήρησε ότι τα δίκτυα εκείνης της περιόδου είχαν ένα περιορισμένο αριθμό στρωμάτων όσο αφορά το βάθος. Για να μπορέσουν λοιπόν οι συγγραφείς να επεκτείνουν το βάθος του δικτύου, χρησιμοποίησαν πολύ μικρά φίλτρα 3×3 σε κάθε συνελικτικό στρώμα του δικτύου. Επίσης έχουν σχεδιάσει μια πληθώρα δομών με αυτή τη λογική για να χρησιμοποιηθούν ανάλογα με τις ανάγκες του δικτύου, οι οποίες φαίνονται στο Σχήμα 3.1.

Σχήμα 3.1: Διαφορετικές διαμορφώσεις του VGG [25, Πίνακας 1]

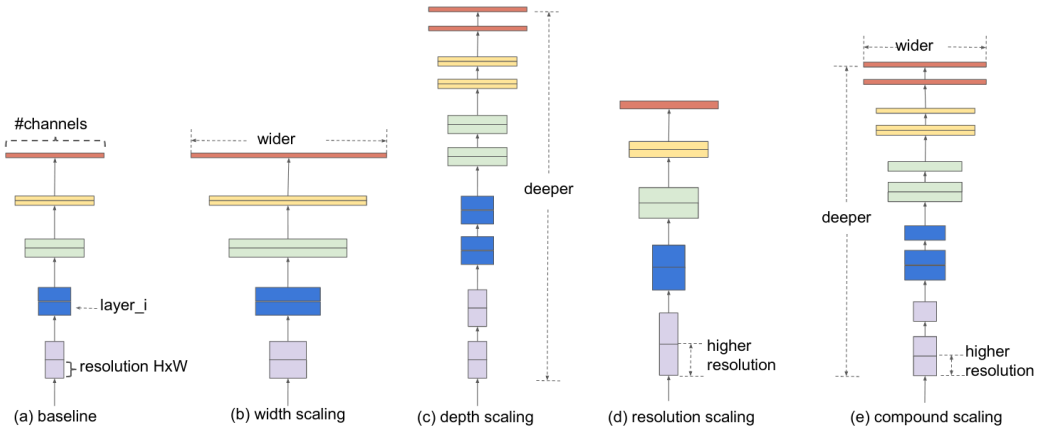
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

3.3 EfficientNet

Το EfficientNet [27], όπως αναφέρει και το όνομα του είναι ένα μοντέλο που δίνει μεγάλη προτεραιότητα στην αποδοτικότητα. Ο συγγραφέας αναφέρει ότι ο πιο συνηθισμένος τρόπος για την αύξηση της αποδοτικότητας τον περισσότερων μοντέλων είναι η απλή αύξηση μιας από τις υπερπαραμέτρους του μοντέλου, όπως το πλήθος των στρωμάτων, το πλάτος των στρωμάτων ή την ανάλυση της εικόνας. Το Σχήμα 3.2 απεικονίζει τους διαφορετικούς τρόπους αύξησης των υπερπαραμέτρων. Τις περισσότερες φορές αυτές οι παράμετροι επιλέγονται εμπειρικά και είναι βελτιστοποιημένες χειροκίνητα, καταλαμβάνοντας αρκετό χρόνο και προσπάθεια και αρκετά συχνά, χωρίς να παραχθούν τα επιθυμητά αποτελέσματα.

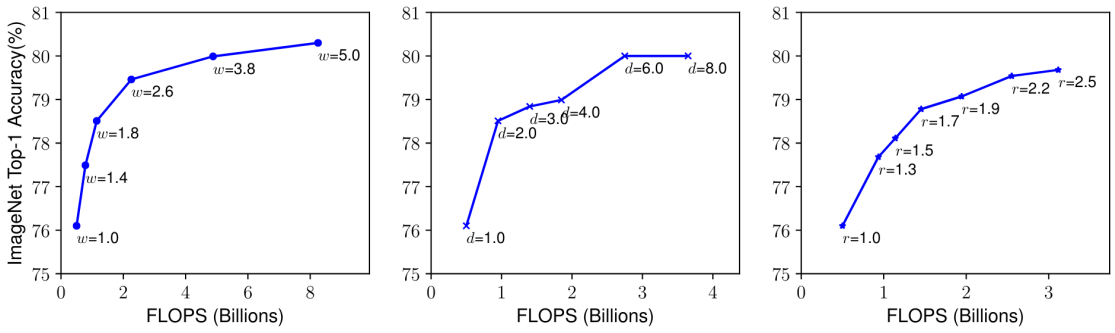
Επίσης, υπάρχει ένα μειονέκτημα με την αύξηση μιας μόνο υπερπαράμετρου, το

Σχήμα 3.2: Παραδείγματα κλιμάκωσης [27, Σχήμα 2]



οποίο είναι ότι αρχίζουμε και αντιμετωπίζουμε αρκετά γρήγορα προβλήματα που αφορούν τον νόμο των φθινουσών αποδόσεων. Οπότε οι συγγραφείς σκέφτηκαν έναν τρόπο για να αυξήσουν αυτές τις βασικές παραμέτρους ταυτόχρονα, ώστε να μετριάσουν τις επιπτώσεις του νόμου των φθινουσών αποδόσεων. Το Σχήμα 3.3 δείχνει την επιρροή του νόμου αυτού στα πειραματικά αποτελέσματα των συγγραφέων.

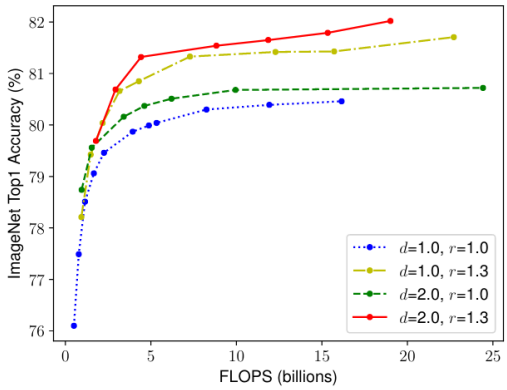
Σχήμα 3.3: Ο νόμος των φθινουσών αποδόσεων, αυξάνοντας τις διαστάσεις του δικτύου με ανάλογα με το φάρδος, ύψος και ανάλυση [27, Σχήμα 3]



Επίσης, ανάλογα με την παράμετρο που αυξάνουμε, βελτιστοποιούμε διαφορετικά σημεία του μοντέλου. Όταν αυξάνουμε το πλάτος, δηλαδή το πλήθος των φίλτρων, μπορούμε και εντοπίζουμε χαρακτηριστικά που αποτελούνται από πολύ μικρές λεπτομέρειες, αλλά το μειονέκτημα με αυτήν την αύξηση είναι ότι γίνεται περισσότερο δύσκολο να εντοπίσουμε χαρακτηριστικά με μεγάλη εννοιολογική σημασία, αυξάνοντας το βάθος μπορούμε και ανιχνεύουμε χαρακτηριστικά μεγαλύτερης εννοιολογικής σημασίας αλλά κάνει τα μοντέλα πολύ πιο δύσκολα στην εκπαίδευση και είναι πιο εύκολα να προκύψει το πρόβλημα της φθίνουσας κλήσης. Και τέλος, αυξάνοντας την ανάλυση του δικτύου είναι πιο εύκολο να εντοπιστούν

μοτίβα (patterns) με μικρότερες λεπτομέρειες. Η ταυτόχρονη αυξομείωση των διαφορετικών διαστάσεων του δικτύου φαίνεται στο Σχήμα 3.2.

Σχήμα 3.4: Παράδειγμα που δείχνει πως ο συνδυασμός αυξήσεων φάρδους και ανάλυσης είναι πιο αποτελεσματικός από την αύξηση μίας παραμέτρου [27, Σχήμα 4]



Ο τύπος που κατάληξαν οι συγγραφείς για την παραμετροποίηση της αύξησης υπερπαραμέτρων του δικτύου περιγράφεται από την εξίσωση 3.1.

$$\begin{aligned} &\text{βάθος: } d = \alpha^{\phi} \\ &\text{πλάτος: } w = \beta^{\phi} \\ &\text{ανάλυση: } r = \gamma^{\phi} \\ &\text{όπου } \alpha \times \beta^2 \times \gamma^2 \approx 2 \end{aligned} \tag{3.1}$$

Στην εξίσωση 3.1 Η παράμετρος ϕ είναι μια παράμετρος επιλεγμένη χειροκίνητα, και την προσδιορίζουμε με βάση τους πόσους πόρους θα χρειαστεί το μοντέλο για την εφαρμογή που το θέλουμε. Η αύξηση μιας μονάδας στο ϕ αυξάνει τους υπολογιστικούς πόρους (FLOPS) που θα χρειαστεί το μοντέλο κατά 2^{ϕ} φορές. Για τις τιμές των α, β και γ οι συγγραφείς χρησιμοποιούν έναν απλό αλγόριθμο αναζήτησης πλέγματος (grid search algorithm). Και αφού βρεθούν αυτές οι τιμές για το μοντέλο που χρησιμοποιούμε, μπορούμε μετά να το προσαρμόσουμε στις ανάγκες μας χρησιμοποιώντας την παράμετρο ϕ.

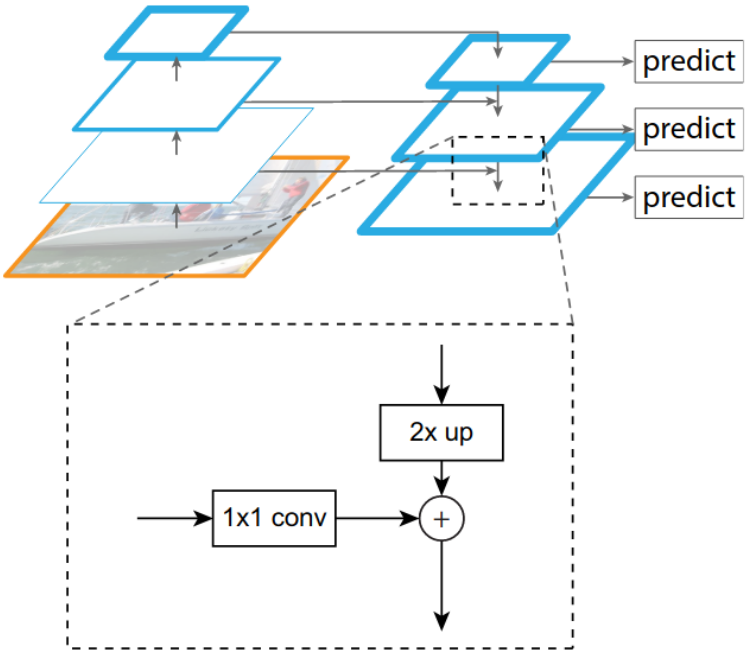
Το αρχικό μοντέλο στο οποίο βασίζεται το EfficientNet είναι το MnasNet [26] που είναι ήδη ένα μοντέλο που έχει ως στόχο την αποδοτικότητα. Οπότε βρίσκοντας τις κατάλληλες παραμέτρους έχουν καταφέρει οι συγγραφείς να δημιουργήσουν ένα δίκτυο που σύμφωνα με τα πειράματα τους, μειώνει δραματικά το πλήθος των παραμέτρων και την επεξεργαστική δύναμη που χρησιμοποιεί ένα δίκτυο, παράλληλα

διατηρώντας πολύ ανταγωνιστικά ποσοστά ακρίβειας.

3.4 Feature Pyramid Network [14]

Παραδοσιακά, για να μπορέσει ένα μοντέλο να κάνει προβλέψεις πάνω σε ένα μεγάλο εύρος μεγεθών χρειαζόταν να υλοποιήσει μια δομή που είχε ως κορμό τη μεθοδολογία που ονομάζεται "Πυραμίδα Χαρακτηριστικών" (Feature Pyramid). Αυτές οι πυραμίδες χαρακτηριστικών σαν κύριο σκεπτικό έχουν την κλιμάκωση της αρχικής εικόνας σε ένα πλήθος από διαφορετικά μεγέθη, και μετά για κάθε στρώμα διαφορετικής κλίμακας, τρέχουμε έναν ταξινομητή. Όμως η κλιμάκωση της ίδιας εικόνας πολλές φορές δημιουργεί πρόβλημα στην απόδοση των μοντέλων ειδικά στο στάδιο της πρόβλεψης, καθιστώντας τη χρήση αυτής της μεθοδολογίας απαγορευτική για εφαρμογές πραγματικού χρόνου. Ο συγγραφέας κάνει μια αναφορά στη προσπάθεια που έχουν κάνει οι δημιουργοί του μοντέλου SSD [16] για να αποφύγουν αυτό το πρόβλημα, όπου το κύριο σκεπτικό είναι αντί να χρησιμοποιήσουν την ίδια εικόνα πολλές φορές στην ταξινόμηση, να χρησιμοποιήσουν τους χάρτες χαρακτηριστικών διαφορετικών κλιμάκων που παράγονται από αυτήν ώστε να μπορέσουν να μειώσουν το υπολογιστικό κόστος. Όμως αυτή η στρατηγική δεν είναι πλήρως αποτελεσματική διότι από τη φύση των συνελικτικών νευρωνικών δικτύων, οι χάρτες χαρακτηριστικών στα αρχικά επίπεδά του δικτύου έχουν πολύ μικρότερη εννοιολογική σημασία, και μόνο στους χάρτες χαρακτηριστικών των επόμενων επιπέδων αναβαίνει η εννοιολογική σημασία. Με αποτέλεσμα είτε να εκμεταλλεύεταί το δίκτυο χάρτες χαρακτηριστικών υψηλής ανάλυσης και χαμηλής εννοιολογικής σημασίας, είτε να εκμεταλλεύεται χάρτες υψηλής εννοιολογικής σημασίας που έχουν χαμηλή ανάλυση. Με αποτέλεσμα να μην μπορεί το μοντέλο να εκμεταλλευτεί ομοιόμορφα όλη την πληροφορία της αρχικής εικόνας. Οι συγγραφείς του FPN (Feature Pyramid Network), έχουν σκεφτεί μια λύση για να αντιμετωπίσουν τα προβλήματα αυτά που είχαν οι προηγούμενες προσεγγίσεις, συνδυάζοντας την πληροφορία μεγάλης εννοιολογικής σημασίας που παρέχουν οι χάρτες χαρακτηριστικών των τελευταίων στρωμάτων με την πληροφορία πλούσια σε λεπτομέρειες που παρέχουν τα αρχικά στρώματα. Τελικά οι συγγραφείς καταλήξανε σε μια αρχιτεκτονική που απεικονίζεται στο Σχήμα 3.5, και καταφέρνουν να δημιουργήσουν μια ολοκληρωμένη λύση, ταυτόχρονα διατηρώντας την απόδοσή σε ένα ικανοποιητικό βαθμό.

Σχήμα 3.5: Αρχιτεκτονική FPN [14, Σχήμα 1]

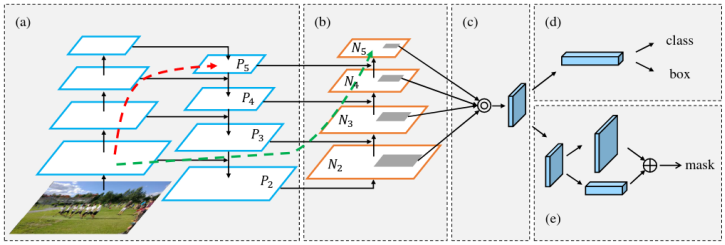


3.5 PANet

Το PANet [15] είναι μια αρχιτεκτονική που επικεντρώνεται περισσότερο για τη λύση προβλημάτων εύρεσης περιγράμματος αντικειμένων (image segmentation), αυτό σημαίνει ότι δεν αρκεί μόνο να βρεθούν τα περιβάλλοντα κουτιά που έχει μια εικόνα αλλά, να βρεθεί και το περίγραμμα του αντικειμένου που προσπαθούμε να ταξινομήσουμε. Αυτό το δίκτυο χρησιμοποιεί την αρχιτεκτονική FPN [14] για την εξαγωγή των χαρτών χαρακτηριστικών για διαφορετικές κλίμακες, και παράλληλα προσθέτουν ένα στάδιο στην αρχιτεκτονική FPN διότι, παρόλο που διαδίδεται η πληροφορία από τα χαμηλότερα στρώματα στα υψηλότερα όπως φαίνεται και στο Σχήμα 3.5 υπάρχει σημαντική αλλοίωση της πληροφορίας και χάνεται η ευκαιρία να εμπλουτιστούν τα κορυφαία στρώματα με τις αρκετά λεπτομερές πληροφορίες των κατώτερων στρωμάτων.

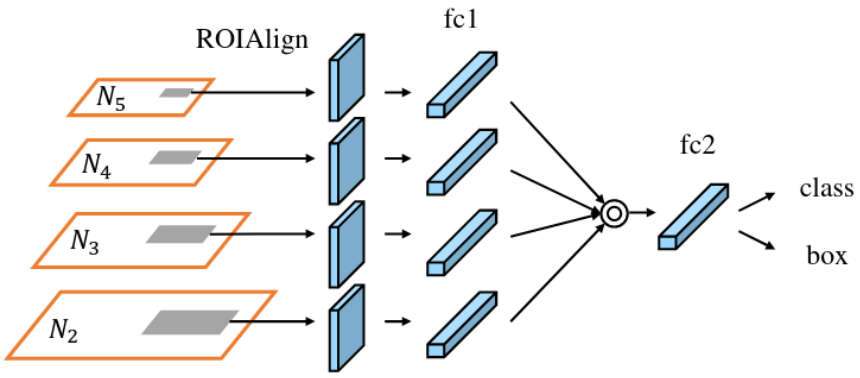
Αυτή η από κάτω προς τα πάνω μετάδοση της πληροφορίας δε χρησιμοποιεί ένα μεγάλο πλήθος συνελικτικών στρωμάτων, αλλά είναι μια απλή πρόσθεση της πληροφορίας που για κάθε επίπεδο περνάει μόνο από δύο 3×3 συνελικτικά στρώματα που τον μόνο σκοπό που έχουν είναι η μείωση του πλήθους παραμέτρων. Οπότε για κάθε επίπεδο i προσθέτουμε τον χάρτη χαρακτηριστικών N_{i-1} με τον

Σχήμα 3.6: Αρχιτεκτονική PANET [15, Σχήμα 1]



χάρτη χαρακτηριστικών P_i, με αποτέλεσμα να μειώνουμε το πλήθος στρωμάτων που χρειάζεται για να περνάει μια πληροφορία χαμηλού επιπέδου ώστε να συνδυαστεί με την πληροφορία του υψηλού επιπέδου. Αυτή η διάδοση απεικονίζεται στο Σχήμα 3.6. Αξιοποιώντας αυτή τη τεχνική, καταφέρνουμε και κρατάμε πιο λεπτομερή πληροφορία για την τοποθεσία των χαρακτηριστικών με υψηλότερες εννοιολογικές σημασίες, πράγμα που είναι κρίσιμο για το πρόβλημα του διαχωρισμού εικόνας.

Σχήμα 3.7: Δομή Feature Pooling PANet [15, Σχήμα 4]



Τέλος, αναφέρουν οι συγγραφείς ότι οι προηγούμενες υλοποιήσεις είχαν το μειονέκτημα ότι ταξινομούσαν μικρά αντικείμενα μόνο με βάση των χαρακτηριστικών των χαμηλότερων επιπέδων, και τα μεγάλα αντικείμενα μόνο με χαρακτηριστικά των μεγαλύτερων επιπέδων. Γι' αυτόν τον λόγο προτείνουν μια μεθοδολογία που την ονομάζουν Feature Pooling, η οποία περιγράφεται στο Σχήμα 3.7, όπου για κάθε υποψήφιο περιβάλλων κουτί, αθροίζουμε τις πληροφορίες από όλα τα στρώματα καλύπτοντας περιπτώσεις λάθους ταξινόμησης λόγω μικρών διαφορών στην τοποθεσία του αντικειμένου.

3.6 ResNet

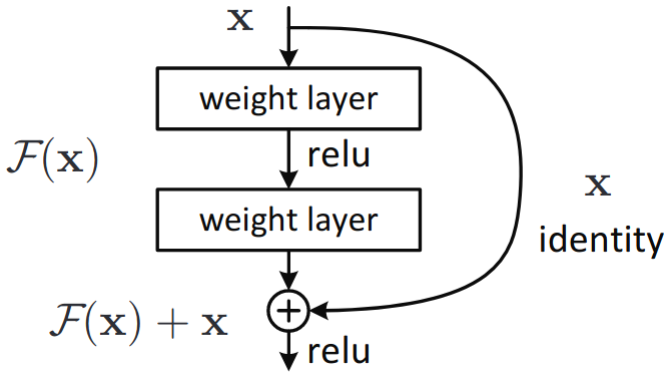
Η εφαρμογή των υπολειπόμενων νευρωνικών δικτύων (residual neural networks) στα προβλήματα εύρεσης αντικειμένων πηγάζει από την ανάγκη που υπάρχει για βαθύτερα νευρωνικά δίκτυα. Όμως η εκπαίδευση των νευρωνικών δικτύων είναι ένα εξαιρετικά δύσκολο πρόβλημα, που γίνεται πιο δύσκολο με την προσθήκη περισσότερων στρωμάτων. Διότι για κάθε στρώμα που προσθέτουμε, αυξάνεται η ακρίβεια του δικτύου αλλά μετά από ένα σημείο υπάρχει κορεσμός. Αν συνεχίσουμε να προσθέτουμε στρώματα μετά από το σημείο κορεσμού, βλέπουμε ότι αρχίζει και μειώνεται η ακρίβειά του δικτύου (degradation problem). Αυτό συμβαίνει διότι τα βαθύτερα δίκτυα έχουν μεγαλύτερο σφάλμα εκπαίδευσης (training error). Επίσης ένα άλλο πρόβλημα που δημιουργείται από τη χρήση πολλαπλών στρωμάτων είναι το πρόβλημα της φθίνουσας κλίσης που αναφέρθηκε στο προηγούμενο κεφάλαιο.

Ο κύριος σκοπός του ResNet [10] είναι να ελαχιστοποιήσει το σφάλμα εκπαίδευσης του δικτύου και να περιορίσει το πρόβλημα της φθίνουσας κλήσης, ώστε να μπορέσει να κάνει τα μοντέλα με πολλαπλά στρώματα πιο εφικτά στην εκπαίδευση τους. Αυτό το πρόβλημα αποφάσισαν να το επιλύσουν με τη χρήση των συνδέσεων παράληψης (skip connections), οι οποίες προσθέτουν ακόμα ένα μονοπάτι που παρακάμπτει μερικά στρώματα του δικτύου.

Για να αιτιολογήσει ο συγγραφέας το πώς κατέληξε σε αυτή τη μεθοδολογία, ανέφερε ότι θεωρητικά μπορούμε να προσθέσουμε περισσότερα στρώματα στο δίκτυο κατά τη διάρκεια της εκπαίδευσης αν αυτά τα στρώματα έχουν έξοδο ίση με την είσοδο (identity mapping). Διότι αν το κάθε στρώμα απλά έχει τη συμπεριφορά του προηγούμενου τότε και το σφάλμα εκπαίδευσης θα είναι ίσο με το προηγούμενο, πράγμα που επιλύει το πρόβλημά της απώλειας ακρίβειας του δικτύου. Αλλά μέχρι στιγμής δεν έχει βρεθεί κάποια ικανοποιητική λύση που να προσθέτει στρώματα στο δίκτυο με αυτόν τον τρόπο. Οπότε οι συγγραφείς ακολούθησαν μια άλλη λογική. Αντί να προσθέσουν περισσότερα στρώματα κατά τη διάρκεια της εκπαίδευσης του δικτύου, τα έχουν ήδη προσθέσει τα στρώματα αυτά εξ αρχής. Όπου αντί να περιμένουμε να προσεγγίσει η έξοδος μια είσοδο μέσω των συνελικτικών στρωμάτων θα συνδέσουμε απευθείας την είσοδο με την έξοδο. Οπότε πρακτικά, έστω ότι η έξοδος που θέλουμε να προσεγγίσουμε είναι ή $\mathcal{H}(x)$ και η συνάρτηση

που συνθέτεται από τα συνελικτικά στρώματα είναι ή $\mathcal{F}(x)$. Τότε τη σχέση αναμεταξύ τους μπορούμε να την περιγράψουμε ως $\mathcal{F}(x) + x = \mathcal{H}$, η εφαρμογή αυτής της σχέσης στο δίκτυο περιγράφεται στο Σχήμα 3.8.

Σχήμα 3.8: Σύνδεση παράληψης δυο συνελικτικών στρωμάτων [10, Σχήμα 2]



Οπότε για την εφαρμογή της μεθοδολογίας αυτής στο νευρωνικό δικτύου, χρησιμοποιούμε την εξίσωση 3.2.

$$y = \mathcal{F}(x, W_i) + x \tag{3.2}$$

Το x και y στην εξίσωση 3.2 είναι τα διανύσματα εισόδου και εξόδου αντίστοιχα.

Και τέλος για τα δομικά στοιχεία που δεν έχουν της ίδιες διαστάσεις εισόδου και εξόδου μπορούμε να προσθέσουμε ένα διάνυσμα W_s ώστε οι διαστάσεις εισόδου να είναι ίσες με τις διαστάσεις εξόδου. Με αποτέλεσμα να καταλήξουμε στην εξίσωση 3.3

$$y = \mathcal{F}(x, W_i) + W_s x \tag{3.3}$$

Στα πειραματικά αποτελέσματα, οι συγγραφείς είδαν μεγάλες βελτιώσεις στην ακρίβειά του μοντέλου, λόγω της αύξησής του βάθους που τους επέτρεψε αυτή η υλοποίηση.

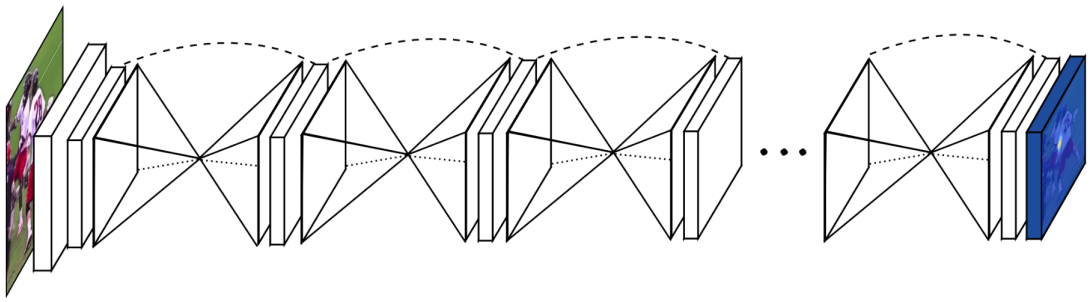
3.7 Hourglass

Η αρχιτεκτονική Hourglass [19] δημιουργήθηκε από τους συγγραφείς για να μπορέσουν να φτιάξουν ένα δίκτυο που να αναγνωρίζει τις στάσεις των ανθρώπων σε μια εικόνα. Μια από τις απαιτήσεις που είχαν από αυτό το δίκτυο είναι η

ακριβείς θέση των αρθρώσεων του ανθρώπινου σώματος. Διότι αν υπάρχει μεγάλη ακρίβεια στη θέση των αρθρώσεων, τότε θα υπάρχει και μεγάλη ακρίβεια στην εκτίμηση της στάσης που έχει ο άνθρωπος στην εικόνα. Και με την ακριβή εύρεση της ανθρώπινης στάσης, μπορούμε να προσδιορίσουμε με μεγαλύτερη ακρίβεια την ενέργεια που κάνει ο άνθρωπος.

Εμπνευσμένοι οι συγγραφείς από τη μεγάλη πρόοδο των συνελικτικών νευρωνικών δικτύων στην επίλυση του προβλήματος της εύρεσης της ανθρώπινης στάσης, δημιούργησαν μια αρχιτεκτονική στην οποία συνδυάζουν πολλαπλές δομές "Κλεψύδρας" (Hourglass) με σκοπό την αξιοποίηση της πληροφορίας σε όλες τις κλίμακες μιας εικόνας. Μια γενική εικόνα της αρχιτεκτονικής φαίνεται στο Σχήμα 3.9.

Σχήμα 3.9: Συνοπτική εικόνα αρχιτεκτονικής Hourglass [19, Σχήμα 1]

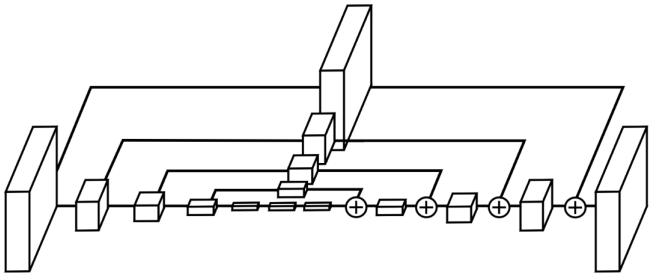


Για τον σχεδιασμό της αρχιτεκτονικής του δικτύου οι συγγραφείς αναφέρουν ότι είναι χρήσιμά τα χαρακτηριστικά χαμηλού επιπέδου, δηλαδή μια άρθρωση όπως ένας ώμος η ένας αγκώνας. Αλλά μπορούμε να έχουμε καλύτερα αποτελέσματα αν το δίκτυο λαμβάνει υπ' όψιν τα χαρακτηριστικά μεγάλου επιπέδου που αφορούν το σύνολό της πληροφορίας στάσης του ανθρώπινου σώματος, και όταν έχουμε μια γενική εικόνα καταφέρνουμε και προσδιορίζουμε με μεγαλύτερη αυτοπεποίθηση τις αρθρώσεις του ανθρώπινου σώματος.

Η αρχιτεκτονική του δικτύου περιγράφεται ως εξής: Κάθε δομικό στοιχείο Hourglass παίρνει την αρχική πληροφορία και την περνάει από μια σειρά συνελικτικών στρωμάτων και στρωμάτων συγκέντρωσης μέγιστης τιμής (max pooling layers) υποδιαιρώντας την ανάλυσή της εικόνας, μετά αφού έχει φτάσει η πληροφορία στη μικρότερη ανάλυση, στο επόμενο στάδιο αυξάνουμέ την ανάλυσή της πληροφορίας από στρώμα σε στρώμα χρησιμοποιώντας την πληροφορία του προηγούμενου στρώματος με τη χρήση του αλγορίθμου του κοντινότερου γείτονα (nearest neighbor) και την απλή πρόσθεση πληροφορίας ανά στοιχείο με το προηγούμενο στρώμα ίσης ανά-

λυσης. Οπότε για κάθε στρώμα που μειώνει την ανάλυση έχουμε ένα που να την αυξάνει, με αποτέλεσμα να υπάρχει μια συμμετρία. Στο τέλος του κάθε δομικού στοιχείου υπάρχουν συνελικτικά στρώματα 1×1 ώστε να μπορέσουν να κρατήσουν σταθερό τον όγκο της πληροφορίας. Τελικά, αυτά τα δομικά στοιχεία μπορούμε να τα προσθέσουμε το ένα μετά το άλλο για να εμβαθύνουμε το δίκτυο μας. Μια συνοπτική αναπαράσταση του δικτύου εμφανίζεται στο Σχήμα 3.10.

Σχήμα 3.10: Δομικό στοιχείο Hourglass [19, Σχήμα 3]



Κεφάλαιο 4

Μοντέλα

4.1 EfficientDet

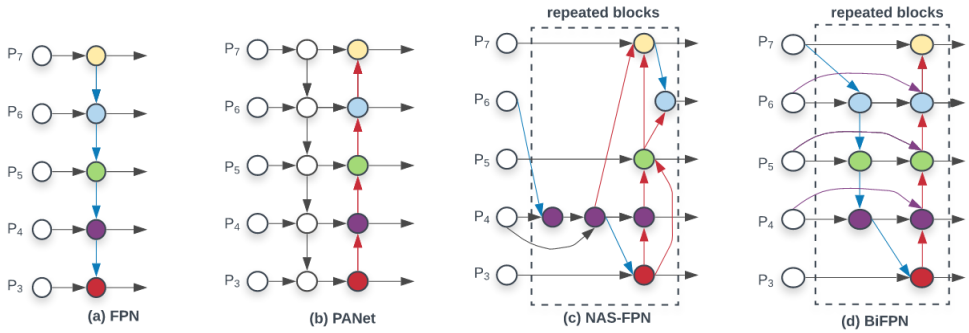
Το πρώτο μοντέλο που θα εξετάσουμε είναι το EfficientDet [28]. Το άρθρο εξηγεί πως ο κύριος στόχος του συγκεκριμένου μοντέλου είναι η μείωση του πλήθους τον παραμέτρων και των FLOP που χρησιμοποιεί, χωρίς να υπάρχει μείωση ακρίβειας. Με κύριο κίνητρό την πιο εύκολη προσάρτησή του μοντέλου σε εφαρμογές που έχουν περιορισμένους φυσικούς πόρους λόγω απαιτήσεων οικονομίας και βάρους όπως τα αυτό-οδηγούμενα αυτοκίνητα και πολλές εφαρμογές στο πεδίο της ρομποτικής.

Ο συγγραφέας αναφέρεται σε περιπτώσεις διαφορετικών μοντέλων που έχουν βρει τρόπους για τη μείωση των απαιτήσεων όσο αφορά τους υπολογιστικούς πόρους. Πιο συγκεκριμένα το SSD [16] και Yolo [21] που έχουν μειώσει το υπολογιστικό κόστος επειδή χρησιμοποιούν ένα στάδιο για την ταξινόμηση και την εύρεση περιβάλλων κουτιών. Ή ακόμα και απόπειρες όπως το CornerNet [13] που έχουν αποφύγει τη χρήση προεπιλεγμένων κουτιών (anchor boxes). Αλλά τονίζει ότι αυτές οι μεθοδολογίες είχαν αρνητική επιρροή στην ακρίβειά των συγκεκριμένων μοντέλων, και ότι η βελτιστοποιήσεις που έκαναν αφορούσαν πολύ συγκεκριμένες απαιτήσεις μειώνοντας ως ένα μεγάλο βαθμό την επεκτασιμότητά (scalability) των μοντέλων. Για αυτό η κύρια φιλοσοφία του μοντέλου έχει λάβει υπ' όψιν και την ευελιξία που χρειάζεται ως προς το εύρος των απαιτήσεων, ώστε να μπορεί να αξιοποιεί όσο το δυνατόν καλύτερα τους διαθέσιμους πόρους ανεξαρτήτως των περιορισμών της επεξεργαστικής δύναμής των συσκευών που το χρησιμοποιούν.

Για τον εντοπισμό τον αντικειμένων οι συγγραφείς βασίστηκαν σε μια παραλλαγή της αρχιτεκτονικής [14] λόγω της ακρίβειας που προσφέρει για τον εντο-

πισμό αντικειμένων διαφορετικών κλιμάκων την οποία την ονόμασαν BiFPN (Bi-Directional Feature Pyramid Network). Το BiFPN δημιουργήθηκε από παρατηρήσεις που είχαν οι συγγραφείς για τις αρχιτεκτονικές FPN [14], PANet[15], NAS-FPN[6]. Το Σχήμα 4.1 περιγράφει συνοπτικά τις διαφορές των διαφορετικών αρχιτεκτονικών. Στις συγκρίσεις που έκαναν οι συγγραφείς, αναφέρουν ότι το NAS-FPN παρόλο που καταλήγει σε ένα αρκετά βελτιστοποιημένο δίκτυο, αυξάνει κατά πολύ τον χρόνο της εκπαίδευσης. Επίσης το PANet έχει καλύτερα αποτελέσματα ακρίβειας, με το μειονέκτημα όμως ότι χρησιμοποιεί περισσότερες παραμέτρους. Τελικά καταλήξανε σε ένα μοντέλο που πλησιάζει περισσότερο το PANet[15] με μερικές βελτιστοποιήσεις.

Σχήμα 4.1: Διαφορές αρχιτεκτονικών FPN[14], PANet[15], NAS-FPN[6], BiFPN [28, Σχήμα 2]



Η πρώτη από αυτές τις βελτιστοποιήσεις είναι η αφαίρεση των κόμβων που έχουν μόνο μια είσοδο, διότι δεν συνεισφέρουν στον συνδυασμό της πληροφορίας από τα στρώματα διαφορετικών επιπέδων. Η δεύτερη βελτιστοποίηση είναι ότι προσθέτουν μια επιπλέον σύνδεση στα στρώματα ίδιου επιπέδου ώστε να μπορέσουν να εκμεταλλευτούν περισσότερη πληροφορία χωρίς να επιβαρύνουν σημαντικά το δίκτυο. Και τέλος επαναλαμβάνουν αυτή τη δομή πολλές φορές στο δίκτυο για να εξάγουν χαρακτηριστικά μεγαλύτερης εννοιολογικής σημασίας, όπου το πλήθος των επαναλήψεων μπορούν να το παραμετροποιήσουν σύμφωνα με τους υπολογιστικούς πόρους τους οποίους το μοντέλο θα μπορεί να χρησιμοποιήσει.

Η προηγούμενες υλοποιήσεις όπως το PANet [15] για τον συνδυασμό των χαρτών χαρακτηριστικών των προηγούμενων σταδίων έκαναν απλές πράξεις πρόσθεσης, αλλά οι συγγραφείς παρατήρησαν ότι αυτή η μεθοδολογία δεν είναι βέλτιστη επειδή οι χάρτες χαρακτηριστικών διαφορετικών επιπέδων δεν έχουν ίση συνεισφορά στο δίκτυο, γι' αυτόν τον λόγο έχουν προσθέσει μια μεταβλητή βάρους στον συνδυασμό

στην είσοδο των χαρτών χαρακτηριστικών και κατέληξαν στη συνάρτηση 4.1.

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i \tag{4.1}$$

Στην εξίσωση 4.1, το w_i μπορεί να είναι ένας πίνακας η διάνυσμα το οποίο υπολογίζεται κατά τη διαδικασία της εκπαίδευσης του δικτύου. το και O είναι η είσοδος και η έξοδος της συνάρτησης. Όπως μπορεί να παρατηρηθεί, στη συνάρτηση 4.1 υπάρχει το κομμάτι $\frac{w_i}{\epsilon+\sum_j w_j}$ το οποίο θυμίζει τη συνάρτηση Soft-Max, οι συγγραφείς αναφέρανε ότι είχαν κάνει μια προσπάθεια με τη χρήση της συνάρτησης Soft-Max, αλλά αποδείχτηκε ότι προσθέτει μεγάλο υπολογιστικό κόστος, οπότε κατέληξαν σε μια απλοποιημένη μορφή της.

Πέρα από τη χρήση της καινούργιας δομής BiFPN, οι συγγραφείς επέλεξαν να ακολουθήσουν το σκεπτικό που είχε το EfficientNet [27] με τη χρήση της μεταβλητής ϕ η οποία καθορίζει τις υπερπαράμετρους του δικτύου ώστε να προσαρμόζεται εύκολα για κάθε ανάγκη. Παράλληλα επέλεξαν να μην χρησιμοποιήσουν ένα αλγόριθμο αναζήτησης πλέγματος σε τόσο μεγάλο βαθμό όπως έχει υλοποιήσει το EfficientNet [27] διότι επιβαρύνει πάρα πολύ την εύρεση των υπερπαραμέτρων στα δίκτυα εντοπισμού αντικειμένων.

Τελικά το μοντέλο χρησιμοποιεί το EfficientNet [27] ως θεμελιώδεις αρχιτεκτονική την οποία παραμετροποιεί ανάλογα με τη τιμή του ϕ. Και για την παραμετροποίηση του BiFPN οι συγγραφείς επιλέξανε μια γραμμική συσχέτιση μεταξύ του βάθους D_{bifpn} και της μεταβλητής ϕ. Και για το πλάτος W_{bifpn} επιλέξανε να χρησιμοποιήσουν μια εκθετική συσχέτιση. Για να βρουν τον εκθετικό συντελεστή της W_{bifpn} χρησιμοποίησαν τον αλγόριθμο αναζήτησης πλέγματος σε ένα μικρό σύνολο τιμών. Τελικά καταλήξανε στις εξισώσεις 4.2.

$$W_{bifpn} = 64 \cdot (1.35^{\phi}), \qquad D_{bifpn} = 3 + \phi \tag{4.2}$$

Για την παραμετροποίηση του μοντέλου που αφορά την εύρεση των περιβάλλων κουτιών και του μοντέλου που κάνει τις ταξινομήσεις το πλάτος W_{pred} είναι ίδιο με το W_{bifpn}. αλλά το βάθος D_{box} υπολογίζεται διαφορετικά από το D_{bifpn} και περιγράφεται από την εξίσωση 4.3.

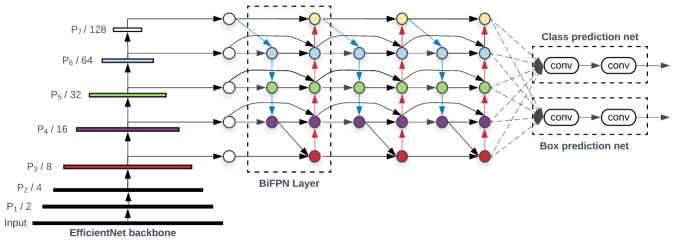
$$D_{box} = D_{class} = 3 + \left\lfloor \frac{\phi}{3} \right\rfloor \tag{4.3}$$

Και τέλος για την ανάλυση της εικόνας εισόδου έχουμε την εξίσωση 4.4.

$$R_{input} = 512 + \phi \cdot 128 \tag{4.4}$$

Οπότε συνοπτικά έχουμε μια ολοκληρωμένη αρχιτεκτονική η οποία καταφέρνει και κρατάει σε ισορροπία τις απαιτήσεις ακρίβειας και ταχύτητας με μεγάλη ευκολία στην προσαρμογή των υπερπαραμέτρων, εκμεταλλευόμενοι τη φιλοσοφία του EfficientNet [27]. Η συνολική εικόνα της αρχιτεκτονικής αυτής φαίνεται στο Σχήμα 4.2

Σχήμα 4.2: Ολοκληρωμένη εικόνα αρχιτεκτονικής EfficientDet [28, Σχήμα 3]



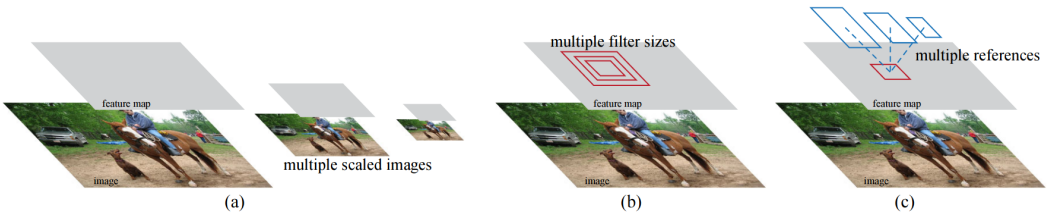
4.2 Faster R-CNN

Το Faster R-CNN [24] είναι τρίτο στην εξέλιξη των δικτύων R-CNN [7] και Fast R-CNN[8]. Όπου το R-CNN είναι ένα μοντέλο που χρησιμοποιούσε τον αλγόριθμο Selective Search [29] για να βρει τις περιοχές που ήταν περισσότερο πιθανό να υπάρχει κάποιο αντικείμενο, και στο τέλος στις προτεινόμενες περιοχές, έκανε προβλέψεις για τις κλάσεις ένα συνελικτικό δίκτυο. Ο συγγραφέας σχολιάζει πως αυτή η λύση που έτρεχε τον αλγόριθμο Selective Search είχε ως αποτέλεσμα να το κάνει το δίκτυο πιο εξαρτώμενο στην απόδοση του CPU παρά του GPU, δημιουργώντας μια συμπλοκή, μειώνοντας την ταχύτητα σε βαθμό μιας τάξη μεγέθους. Αναφέρεται ότι θα μπορούσε ο αλγόριθμος να υλοποιηθεί για να χρησιμοποιήσει τους πόρους του GPU αλλά θα συνέχιζε να έχει αυτή η φιλοσοφία το μειονέκτημα που αποκόβεται η δομή εύρεσης περιβάλλων κουτιών με το υπόλοιπο δίκτυο και χάνονται ευκαιρίες στον διαμοιρασμό τον πόρων.

Οπότε προτείνεται για το βήμα της εύρεσης περιοχών η χρήση ενός ακόμα νευρωνικού δικτύου, το οποίο ονόμασαν Region Propasal Network (RPN) που δένει πολύ καλύτερα στα βήματα της εκπαίδευσης του δικτύου και μειώνει δραματικά τη χρονοκαθυστέρηση για την πρόταση περιοχών σε κάθε εικόνα. Αυτό που ενέπνευσε τη δημιουργία του RPN ήταν μια παρατήρηση των συγγραφών ότι οι χάρτες χαρακτηριστικών που παράγουν τα CNN μπορούν να χρησιμοποιηθούν για την πρόταση περιοχών, επίσης πρόσθεσαν μερικά ακόμα στρώματα με σκοπό την εξακρίβωση των περιοχών και την πιθανότητα τους να περιέχουν ένα αντικείμενο.

Το RPN είναι σχεδιασμένο ώστε να μπορεί να κάνει προτάσεις για ένα μεγάλο εύρος αντικειμένων, ανεξαρτήτως μεγέθους και αναλογίας. Χρησιμοποιώντας τη μέθοδο των προεπιλεγμένων περιβάλλων κουτιών, αποφεύγουν το μεγάλο κόστος που έχουν άλλες μεθοδολογίες που περιλαμβάνουν φίλτρα διαφορετικών διαστάσεων, η την κλιμάκωση της ίδιας εικόνας σε διαφορετικά μεγέθη. Αυτές οι διαφορετικές μεθοδολογίες περιγράφονται συνοπτικά στο Σχήμα 4.3.

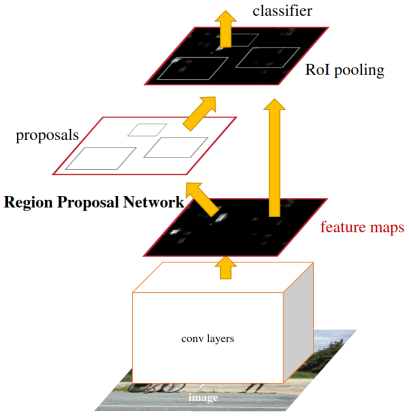
Σχήμα 4.3: Σύγκριση μεθοδολογιών για εύρεση περιοχών που περιέχουν αντικείμενα. a) Μεταβολή μεγέθους αρχικής εικόνας και ταξινόμηση της για κάθε ανάλυση. b) Χρήση φίλτρων διαφορετικού μεγέθους για εύρεση αντικειμένων. c) Μεθοδολογία με προεπιλεγμένα περιβάλλοντα κουτιά για εύρεση ορίων αντικειμένων [24, Σχήμα 1]



Επίσης, για την εκπαίδευση του μοντέλου η μεθοδολογία που χρησιμοποιείται εναλλάσσει την εκπαίδευση του RPN και την εκπαίδευση του δικτύου ταξινόμησης. Με αυτή την τεχνική τα δύο δίκτυα σταθεροποιούνται γρήγορα στην παραγωγή ενός ενιαίου μοντέλου. Αυτά τα δύο δίκτυα περιγράφονται στο Σχήμα 4.4 δείχνει τη γενική διαρρύθμιση του δικτύου.

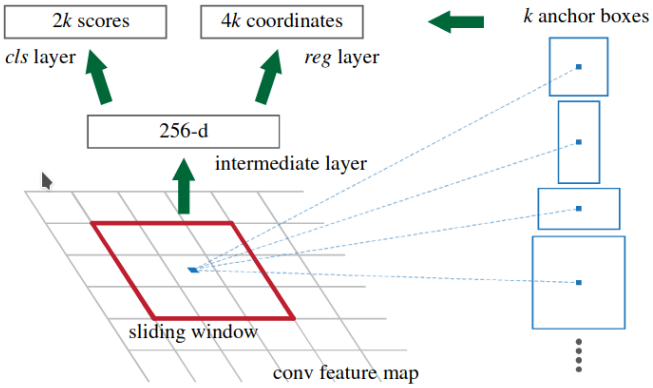
Ένα δίκτυο RPN σαν είσοδο έχει μια εικόνα ανεξαρτήτου μεγέθους και σαν έξοδο έχει τα περιβάλλοντα κουτιά με μια βαθμολογία που ορίζει την πιθανότητα να υπάρχει ένα αντικείμενο σε αυτά. Για να μπορέσει το RPN να βρει συντεταγμένες για τις περιοχές που περιέχουν αντικείμενα, χρησιμοποιεί ένα ολισθόμενο παράθυρο 3×3, και η έξοδος αυτού του παραθύρου πηγαίνει σε δύο πλήρες συνδεδεμένα δίκτυα, ένα για την εύρεση των περιβαλλόντων κουτιών (box-regression layer) και ένα για

Σχήμα 4.4: Αρχιτεκτονική Faster R-CNN [24, Σχήμα 2]



την ταξινόμηση των αντικειμένων (box-classification layer). Για κάθε τοποθεσία του παραθύρου το δίκτυο εξετάζει πολλαπλά περιβάλλοντα κουτιά με πλήθος K, για το δίκτυο των προτάσεων μεταφέρονται $4K$ μεταβλητές που έχουν τις συντεταγμένες της κάθε περιοχής, και για το δίκτυο της ταξινόμησης μεταφέρονται $2K$ μεταβλητές που περιέχουν την πιθανότητα να υπάρχει η να μην υπάρχει αντικείμενο στην περιοχή. Το Σχήμα 4.5 περιγράφει αυτή τη μεθοδολογία. Οι συγγραφείς επέλεξαν 3 διαφορετικά μεγέθη και 3 διαφορετικές αναλογίες με αποτέλεσμα το K να είναι ίσο με 9 για τον κάθε πιθανό συνδυασμό.

Σχήμα 4.5: Στιγμιότυπο ολισθόμενου παραθύρου [24, Σχήμα 3]



Για την εκπαίδευση των RPN υπάρχουν δύο κλάσεις για κάθε σημείο που εξετάζεται, μία για τον χαρακτηρισμό ύπαρξης του αντικειμένου και μία για τον χαρακτηρισμό μη ύπαρξης. Η επιλογή των περιβαλλόντων κουτιών γίνεται με δύο τρόπους: Ο πρώτος είναι να επιλέξουμε τα περιβάλλοντα κουτιά με τον υψηλότερο λόγο επικάλυψης για το αντικείμενο που θέλουμε να εντοπίσουμε. Και ο δεύτερος τρόπος είναι να επιλέξουμε τα κουτιά για το οποία λόγος επικάλυψης να είναι με-

γαλύτερος από 0.7 για οποιαδήποτε αντικείμενο που υπάρχει σε εκείνη την περιοχή. Οι συγγραφείς καταλήγουν σε μια συνάρτηση απώλειας που ορίζεται από τη σχέση 4.5.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{4.5}$$

Στη συνάρτηση 4.5 το i είναι ο δείκτης του περιβάλλοντος κουτιού για τις συγκεκριμένες συντεταγμένες στην εικόνα. Το p_i^* είναι 0 ή 1 ανάλογα με τη λεζάντα της συγκεκριμένης περιοχής. Το t_i αντιπροσωπεύει ένα διάνυσμα με τις συντεταγμένες των γωνιών του περιβάλλοντος κουτιού. Το t_i^* είναι οι συντεταγμένες του πραγματικού περιβάλλων κουτιού. Το L_{cls} είναι η απώλεια ταξινόμησης των κλάσεων, η οποία περιγράφεται με μια λογαριθμική συνάρτηση. Το L_{reg} είναι η απώλεια συντεταγμένων περιβάλλων κουτιών για την οποία χρησιμοποιείται η συνάρτηση Robust Loss [1] (smooth L_1). Μετά οι δύο αθροιστικοί όροι κανονικοποιούνται από τις παραμέτρους N_{reg} και $_{cls}$, όπου το λ είναι μια τιμή για τη ρύθμιση της ισορροπίας των δύο απωλειών όπου το $_{cls}$ αντιπροσωπεύει το πλήθος των περιβάλλων κουτιών, το N_{reg} είναι το πλήθος των σημείων των οποίων έχουν τα υποψήφια περιβάλλοντα κουτιά ως κέντρο.

Για την εύρεση των διαστάσεων του περιβάλλων κουτιού έχουμε τις σχέσεις 4.6

$$\begin{aligned}
t_x &= (x - x_a)/w_a, \quad & t_y &= (y - y_a)/h_a, \\
tw &= log(w/w_a), & th &= log(h/h_a), \\
t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a, \\
tw^* &= log(w^*/w_a), & th^* &= log(h^*/h_a)
\end{aligned} \tag{4.6}$$

Στην εξίσωση 4.6 το x, y, w, h αναπαριστούν τις συντεταγμένες του κέντρου του περιβάλλοντος κουτιού, το πλάτος του, και το ύψος του αντίστοιχα. Οι μεταβλητές που έχουν τον δείκτη a αναφέρονται στα αρχικά περιβάλλοντα κουτιά και οι μεταβλητές που χρησιμοποιούν τον δείκτη $*$ αναφέρονται στα πραγματικά περιβάλλοντα κουτιά.

Επίσης, οι συγγραφείς αναφέρανε ότι οι προηγούμενες λύσεις που είχαν χρησιμοποιήσει αυτή τη τεχνική με τα προεπιλεγμένα περιβάλλοντα κουτιά, είχαν το μειονέκτημα ότι έλειπε μεθοδικότητα για την εύρεση των διαστάσεων τους, και τα

βάρη που προέκυπταν από την εκπαίδευση του δικτύου μοιραζόντουσαν μεταξύ των διαφορετικών αρχικών κουτιών. Για την αποφυγή του μειονεκτήματος αυτού, οι συγγραφείς χρησιμοποίησαν διαφορετικά βάρη για το κάθε προεπιλεγμένο περιβάλλων κουτί, καταφέρνοντας να εντοπίσουν και να περιγράψουν με ακρίβεια αντικείμενα διαφορετικών μεγεθών επειδή έχει προστεθεί μια ευελιξία για το κάθε αρχικό κουτί, του οποίου οι διαστάσεις είναι πολύ συγκεκριμένες.

Το νευρωνικό δίκτυο που θα αξιοποιεί τις προτάσεις που βγάζει το RPN είναι το Fast-RCNN [8]. Αξίζει να αναφερθεί ότι το δίκτυο RPN μαζί με το δίκτυο Fast-RCNN θα πρέπει να εκπαιδευτούν παράλληλα, διότι αν γίνει προσπάθεια να εκπαιδευτούν τα δίκτυα RPN και Fast-RCNN ξεχωριστά θα δημιουργηθούν προβλήματα επειδή τα δίκτυα αυτά μπορούν να καταλήξουν σε εντελώς διαφορετικά φίλτρα χωρίς να υπάρχει κάποια συμβατότητα μεταξύ των δύο αυτών δικτύων, καταλήγοντας με εντελώς διαφορετικά χαρακτηριστικά. Για να μπορέσει να επιλυθεί αυτό το πρόβλημα θα πρέπει να εκπαιδευτούν και τα δύο δίκτυα ταυτόχρονα. Για να υπάρξει ταυτόχρονη εκπαίδευση των δικτύων, οι συγγραφείς χρησιμοποιούν τη μεθοδολογία που αναφέρθηκε προηγουμένως όπου αρχικά εκπαιδεύεται το RPN και μετά χρησιμοποιούνται οι προτάσεις για να εκπαιδευτεί το Fast-RCNN μετά στον επόμενο κύκλο για την εκπαίδευση του RPN χρησιμοποιούνται τα αποτελέσματα του Fast-RCNN από τον προηγούμενο κύκλο, καταλήγοντας σε δύο δίκτυα που συμβαδίζουν.

4.3 SSD Resnet

Η έμπνευση για το μοντέλο SSD [16] (Single Shot MultiBox Detector) προήλθε από μια παρατήρηση που είχαν οι συγγραφείς: Ότι τα δημοφιλέστερα δίκτυα χρησιμοποιούσαν πρώτα μια δομή εύρεσης των περιοχών που μπορεί να υπάρχουν για μια εικόνα και μετά την κάθε υποψήφια περιοχή την εισήγαγαν σε ένα δίκτυο ταξινόμησης. Παρόλο που αυτή η προσέγγιση είχε πολύ καλά αποτελέσματα όσο αφορά τις μετρικές ακρίβειας, είχαν έλλειψη στον τομέα της ταχύτητας, που ακόμα και τα πιο γρήγορα μοντέλα που ακολουθούσαν αυτή τη μεθοδολογία όπως το Faster-RCNN[24] είχαν στην καλύτερη περίπτωση ταχύτητα αναγνώρισης 7 εικόνες το δευτερόλεπτο. Και τα δίκτυα που είχαν μεγάλες ταχύτητες ως αντάλλαγμα είχαν μεγάλη πτώση στην ακρίβεια τους.

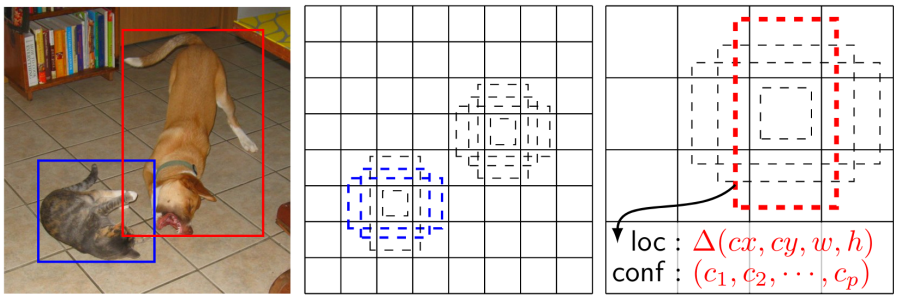
Το SSD [16] είναι από τα πρώτα μοντέλα που ως φιλοσοφία έχει να περνάει την εικόνα μόνο μια φορά από το δίκτυο για να βγάλει συμπεράσματα για τις διαστάσεις των περιβάλλων κουτιών και για την ταξινόμηση των κλάσεων των περιοχών που περιέχουν. Παράλληλα διατηρώντας τη μεγάλη ακρίβεια των μοντέλων που χρησιμοποιούν τη προσέγγιση των δύο σταδίων (2-stage detector) και ταυτόχρονα αυξάνοντας την ταχύτητα εύρεσης αντικειμένων. Τελικά κατέληξαν οι συγγραφείς σε ένα μοντέλο που έχει ταχύτητα μεγαλύτερη από το Faster-RCNN και να έχει 1% μεγαλύτερο mAP στο Pascal VOC 2007 τεστ.

Οι συγγραφείς αναφέρουν ότι δεν είναι οι πρώτοι που έχουν δοκιμάσει αυτή την προσέγγιση αναφέροντας το YoloV1[21] αλλά είναι οι πρώτοι που έχουν κάνει αρκετές βελτιστοποιήσεις ώστε να ανεβάσουν κατά ένα μεγάλο βαθμό την απόδοση. Οι τρεις κύριες βελτιώσεις που έχουν κάνει στο δίκτυο είναι το να χρησιμοποιήσουν ένα μικρό συνελικτικό φίλτρο για την πρόβλεψη των διαστάσεων και τις κλάσεις του αντικειμένου. Τη χρήση διαφορετικών φίλτρων για κάθε αναλογία και μέγεθος. Και τέλος, εφαρμόζουν αυτά τα φίλτρα και σε βαθύτερα μέρη του δικτύου επιτυγχάνοντας μεγαλύτερη ακρίβεια ανεξαρτήτως της κλίμακας μεγέθους του αντικειμένου.

Για την πρώτη έκδοση του SSD οι συγγραφείς αξιοποίησαν την αρχιτεκτονική VGG-16[25] και αφαίρεσαν τα στρώματα που αφορούν την ταξινόμηση του αντικειμένου, και τα αντικατέστησαν με περισσότερα συνελικτικά στρώματα, που ο μόνος τους σκοπός τους, είναι το να προβλέπουν τα περιβάλλοντα κουτιά με βάση τους χάρτες χαρακτηριστικών που έχει εξάγει το VGG-16 από την αρχική εικόνα. Επίσης το μοντέλο κάνει προβλέψεις σε όλους τους χάρτες χαρακτηριστικών που έχουν διαφορετικά μεγέθη, παράλληλα διατηρώντας το μέγεθος του περιβάλλοντος κουτιού. Αυτό έχει ως αποτέλεσμα το ίδιο περιβάλλων κουτί, να καλύπτει διαφορετικές κλίμακες λόγω των διαφορετικών διαστάσεων των χαρτών χαρακτηριστικών. Αυτή η μεθοδολογία φαίνεται στο Σχήμα 4.6.

Οπότε στην ουσία οι συγγραφείς για τη σχεδίαση αυτού του μοντέλου πήραν το VGG-16[25] και αφαίρεσαν τα στρώματα που αφορούν την ταξινόμηση, και πρόσθεσαν μια δικιά τους υλοποίηση η οποία αποτελείται από στρώματα διαφορετικών διαστάσεων για να μπορούν να διατηρήσουν μια ακρίβεια ανεξαρτήτως της κλίμακας του αντικειμένου. Μετά από αυτό το βήμα το υπόλοιπο μοντέλο χρησιμοποιεί 3×3 φίλτρα σταδιακά μειώνοντας τις διαστάσεις του μοντέλου. Στη συνέχεια, για

Σχήμα 4.6: Εύρεση περιβάλλοντα κουτιού αντικειμένων διαφορετικού μεγέθους [16, Σχήμα 1]



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

το κάθε στρώμα $m \times n$ που έχουν παράξει αυτά τα φίλτρα οι συγγραφείς αναθέτουν k περιβάλλοντα κουτιά για κάθε κελί, και το κάθε περιβάλλων κουτί κρατάει τιμές για την πιθανότητα να υπάρχει μια κλάση σε εκείνο το σημείο, και τις πιθανές αποστάσεις του αρχικού περιβάλλοντος κουτιού με του πραγματικού περιβάλλοντος κουτιού. Σε αυτό το σημείο αναφέρουν οι συγγραφείς ότι η ιδέα των αρχικών περιβαλλόντων κουτιών είναι παρόμοια με του Faster-RCNN[24], αλλά το πλεονέκτημα που έχει αυτή η μεθοδολογία είναι ότι τα περιβάλλοντα κουτιά τα προβλέπουν για πολλαπλούς χάρτες χαρακτηριστικών διαφορετικών μεγεθών, και ότι η χρήση αυτών των περιβάλλων κουτιών με τις σταθερές διαστάσεις, τους επιτρέπει να προσπελάσουν την εικόνα με μεγαλύτερη αποδοτικότητα.

Για την εκπαίδευση του δικτύου χρειάζεται να μεταφέρουμε την πληροφορία των πραγματικών συντεταγμένων του αντικειμένου σε κάποιες συγκεκριμένες εξόδους των ταξινομητών. Πράγμα που διαφοροποιεί αυτή τη μεθοδολογία με τις μεθοδολογίες άλλων δικτύων που χρησιμοποιούν τους πιο παραδοσιακούς αλγορίθμους για την εύρεση περιοχών, Π.Χ selective search[29]. Για να μπορέσουν να εκπαιδεύσουν το δίκτυο όσον αφορά την τοποθεσία του αντικειμένου οι συγγραφείς απλά για να κρίνουν ικανοποιητικό ένα περιβάλλων κουτί έχουν ως κριτήριο το IOU να είναι μεγαλύτερο του 0.5. Αυτή η τακτική έχει το πλεονέκτημα ότι μπορεί να βρει πολλαπλά περιβάλλοντα κουτιά που έχουν μεγάλη αυτοπεποίθηση για τη συγκεκριμένη περιοχή, χωρίς να τα αποκλείουν επειδή μπορεί να έχουν μια χαμηλή αρχική επικάλυψη, με το σκεπτικό ότι θα καταφέρουν να καταλήξουν στις σωστές διαστάσεις κατά την εκπαίδευση του δικτύου (regression).

Για τη δημιουργία της συνάρτησης απώλειας οι συγγραφείς ορίζουν μια μεταβλητή $x_{ij}^p = \{1, 0\}$ ως μια τιμή που αντιστοιχίζει το i-οστό περιβάλλων κουτί

πρόβλεψης με το j-οστό πραγματικό περιβάλλων κουτί για την p κλάση. Οπότε για τη συνάρτηση απώλειας έχουμε την απώλεια που προκύπτει από τη μη σωστή τοποθέτηση του περιβάλλοντος κουτιού (localization loss), και την απώλεια μη σωστής ταξινόμησης, η σχέση αναμεταξύ τους περιγράφεται με τον τύπο: $L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$. Όπου N είναι ο αριθμός των αντιστοιχισμένων αρχικών κουτιών. Η απώλεια τοποθέτησης είναι τύπου smooth f1 χρησιμοποιώντας τις παραμέτρους του προβλεπόμενου περιβάλλοντος κουτιού με σύμβολο l, και του πραγματικού περιβάλλοντος κουτιού με σύμβολο g. Οπότε η συνάρτηση που περιγράφει την απώλεια τοποθέτησης είναι η 4.7.

$$
\begin{gathered}
L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m) \\
\hat{g}_j^{cx} = (g_j^{cx} - d_j^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_j^{cy})/d_i^h \\
\hat{g}_j^w = log(\frac{g_j^w}{d_i^w}) \qquad \hat{g}_j^h = log(\frac{g_j^h}{d_i^h})
\end{gathered}
\tag{4.7}
$$

Και για να υπολογιστεί η απώλεια αυτοπεποίθησης χρησιμοποιείται η συνάρτηση 4.8. Που στην ουσία είναι μια συνάρτηση SoftMax η οποία εφαρμόζεται σε πολλαπλές τιμές αυτοπεποίθησης.

$$
L_{conf}(x, c) = -\sum_{i \in Pos}^{N} - \sum_{i \in Neg} x_{ij}^{p} \quad log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{exp(c_i^p)}{\sum exp c_i^p} \tag{4.8}
$$

Όπως περιγράφηκε και πριν, το μοντέλο χρησιμοποιεί τα ίδια αρχικά περιβάλλοντα κουτιά αλλά σε χάρτες χαρακτηριστικών διαφορετικών επιπέδων, ώστε να μπορεί να συμπεριλάβει στις προβλέψεις για αντικείμενα με μεγάλη διαφορά στην κλίμακα τους. Για να παραμετροποιήσουν την κλίμακα των αρχικών περιβάλλων κουτιών οι συγγραφείς χρησιμοποιούν τη συνάρτηση 4.9.

$$
S_k = s_{min} + \frac{S_{max} - S_{min}}{m - 1}(k - 1), \quad k \in [1, m] \tag{4.9}
$$

Στη συνάρτηση 4.9 το S_{min} και το S_{max} έχουν τιμές 0.2 και 0.9, αντίστοιχα, και έχουν επιλεχθεί με βάση των πειραματισμών των συγγραφών. Επίσης, την κάθε αναλογία τη συμβολίζουν με το σύμβολο a_r το οποίο είναι ένα στοιχείο που ανήκει σε ένα σύνολο που έχει τις εξής τιμές $\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. Το ύψος και το πλάτος των περιβάλλων κουτιών μπορούν να υπολογιστούν από τους τύπους $w_k^a = s_k\sqrt{a_r}, \quad h_k^a = s_k/\sqrt{a_r}$

Επιπλέον προστίθεται ένα επιπλέον περιβάλλων κουτί με αναλογία 1 με την ακόλουθη κλίμακα $s'_k = \sqrt{s_k s_{k+1}}$ οπότε για κάθε σημείο στον χάρτη χαρακτηριστικών δημιουργούμε 6 περιβάλλοντα κουτιά με κέντρο $\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|}$ όπου το $|f_k|$ είναι το μέγεθος του k-οστού χάρτη χαρακτηριστικών όπου $i, j \in [0, |f_k|]$. Επίσης, έχει υλοποιηθεί μια βελτιστοποίηση για την πιο σταθερή εκπαίδευση του μοντέλου που αγνοεί τα περισσότερα υποψήφια περιβάλλοντα κουτιά με βάση την απώλεια αυτοπεποίθησης που έχουν. Και τέλος, προσπαθούν να τηρήσουν μια αναλογία στην οποία για κάθε περιβάλλων κουτί που έχει προβλεφθεί σωστά, να υπάρχουν 3 που απορρίπτονται.

Σε επόμενο στάδιο η υλοποίησή του SSD βελτιστοποιήθηκε από τους Xin Lu κ.α. [17] με την κύρια συνεισφορά την αλλαγή του νευρωνικού δικτύου ραχοκοκαλιάς από VGG σε ResNet. Αυτή η βελτιστοποίηση προήλθε από μια παρατήρηση που είχαν οι συγγραφείς, είδαν ότι η χρήση του VGG [25] είχε το μειονέκτημα ότι περιόριζε το βάθος του δικτύου, και επίσης το VGG δεν ήταν επαρκείς για να βρει με ικανοποιητική ακρίβεια αντικείμενα που είχαν έναν περιορισμένο αριθμό παραδειγμάτων στα σύνολα δεδομένων εκπαίδευσης, και επίσης είχε δυσκολία στο να εντοπίζει αντικείμενα που καταλαμβάνουν ένα μικρό μέρος της εικόνας. Με τα πειραματικά αποτελέσματα των συγγραφέων, είδαν ικανοποιητικές βελτιώσεις με την αλλαγή του δικτύου ραχοκοκαλιάς.

4.4 CenterNet

Το CenterNet είναι ένα μοντέλο που χρησιμοποιεί μια πρωτότυπη λύση για την εύρεση των περιβάλλων κουτιών. Το μοντέλο αυτό είναι βασισμένο στο CornerNet[13], το οποίο δημιουργήθηκε από μια παρατήρησή που είχε ο συγγραφέας του, ότι τα μοντέλα που έχουν την πιο μεγάλη αποδοτικότητα τον τελευταίο καιρό βασίζονται συνήθως στη λογική των αρχικών κουτιών (Π.Χ Faster-RCNN [24], SSD[16]). Όμως οι υλοποιήσεις που χρησιμοποιούσαν τα αρχικά περιβάλλοντα κουτιά είχαν κάποια μειονεκτήματα που ήθελε να αποφύγει ο συγγραφέας τα σημαντικότερα από αυτά ήταν τα εξής: τα μοντέλα που ακολουθούν τη μεθοδολογία αυτή, αναγκάζονται να επιλέξουν έναν μεγάλο αριθμό περιβαλλόντων κουτιών για να μπορούν να καλύψουν ένα μεγάλο σύνολο αναλογιών, ώστε να καταφέρουν να φτάσουν τα κατάλληλα επίπεδα αυτοπεποίθησης κατά το στάδιο της εκπαίδευσης του δικτύου. Ένα άλλο μειονέκτημα είναι το γεγονός ότι αυτές οι αναλογίες πρέπει να προσ-

διοριστούν χειροκίνητα από το άτομο που σχεδιάζει το μοντέλο, και πολλές φορές τα αρχικά περιβάλλοντα κουτιά δεν διασταυρώνονται κατάλληλα με το πραγματικό περιβάλλων κουτί του αντικειμένου, πράγμα που δημιουργεί τριβές στον σχεδιασμό του μοντέλου.

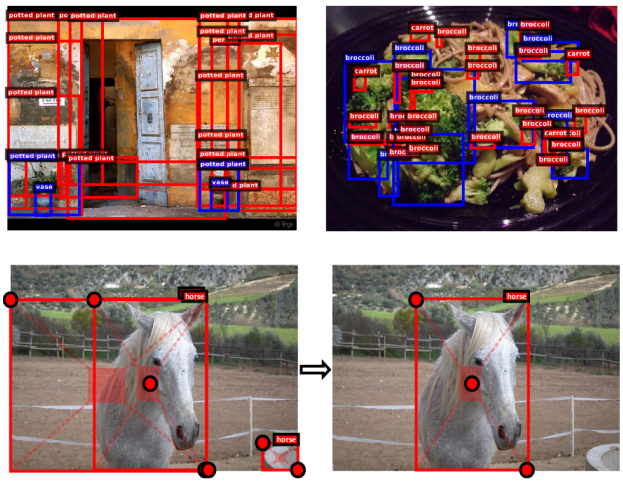
Θέλοντας λοιπόν ο συγγραφέας του CornerNet να αποφύγει αυτά τα μειονεκτήματα, δημιούργησε ένα μοντέλο που το κάθε αντικείμενο αντιπροσωπεύονταν από ένα ζευγάρι συντεταγμένων που αντιστοιχούσε στις γωνίες του περιβάλλοντος κουτιού, έτσι κατάφερε να αποφύγει εξ' ολοκλήρου τη λογική των αρχικών περιβαλλόντων κουτιών με αποτέλεσμα να έχει ένα πολύ αποδοτικό μοντέλο.

Όμως ακολουθώντας αυτή τη νέα προσέγγιση δημιουργήθηκαν κάποιες αδυναμίες: Η πρώτη είναι ότι το μοντέλο δεν μπορεί αξιοποιήσει όλα τα χαρακτηριστικά του αντικειμένου γιατί αγνοεί τη γενική εικόνα και βλέπει μόνο τις άκρες του, με αποτέλεσμα να είναι υπερβολικά ευαίσθητο στα όρια που ορίζεται το αντικείμενο. Η δεύτερη αδυναμία είναι ότι επειδή δεν υπάρχει καμία παραμετροποίηση σχετικά με τις αναλογίες και τις διαστάσεις που πρέπει να έχει ένα αντικείμενο, υπάρχουν αρκετές περιπτώσεις που δεν μπορεί να αντιστοιχιστούν σωστά τα περιβάλλοντα κουτιά στα αντικείμενα που βρίσκονται πολλαπλές φορές στην ίδια εικόνα, πράγμα που μπορεί να διορθωθεί εύκολα χρησιμοποιώντας επιπλέον πληροφορίες, όπως την αναλογία του αντικειμένου.

Οι συγγραφείς οπότε για να καλύψουν τις αδυναμίες του CornerNet έχουν δημιουργήσει το CenterNet[4] στο οποίο πέρα από των εντοπισμό των αντικείμενων χρησιμοποιώντας τις γωνίες, προσθέτει ένα τρίτο σημείο κλειδί στο κέντρο του αντικειμένου για λόγους επαλήθευσης, με την απλή λογική ότι αν για ένα αντικείμενο βρούμε τα δύο σημεία κλειδιά που αντιστοιχούν στις γωνίες του, τότε υπάρχει μια πολύ μεγάλη πιθανότητα το κεντρικό σημείο κλειδί να ανήκει στο ίδιο αντικείμενο. Αυτό το τρίτο σημείο κλειδί φαίνεται στο Σχήμα 4.7.

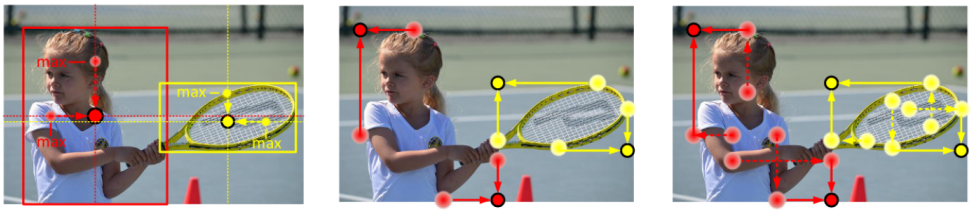
Παράλληλα έχουν προστεθεί κάποιες τεχνικές ώστε να βελτιώσουν τις προτάσεις για τα κεντρικά, και για τα γωνιακά σημεία κλειδιά με τη χρήση κάποιων τεχνικών που προσθέτουν παραπάνω πληροφορίες. Η πρώτη τεχνική είναι η κεντρική συγκέντρωση (center pooling) που στην ουσία είναι μια παρόμοια στρατηγική με τη γωνιακή συγκέντρωση (corner pooling) που χρησιμοποιούσε το CornerNet και αφορά τον εντοπισμό των κεντρικών σημείων κλειδιών, χρησιμοποιώντας τις μέ-

Σχήμα 4.7: Σύγκριση μεθοδολογιών CornerNet και CenterNet [4, Σχήμα 1]



γιστες τιμές στις οριζόντιες και κάθετες κατευθύνσεις του αντικειμένου από ένα χάρτη θερμότητας. Η δεύτερη τεχνική ονομάζεται παλινδρομική γωνιακή συγκέντρωση (cascade corner pooling) η οποία ενισχύει τη γωνιακή συγκέντρωση που είχε ήδη το CornerNet. Αυτός η μεθοδολογία συγκέντρωσης, στην ουσία εισάγει πληροφορίες για το κάθε σημείο κλειδί όχι μόνο από τις οριζόντιες και τις κάθετες κατευθύνσεις του, αλλά και από τις κατευθύνσεις που δείχνουν προς το κέντρο του αντικειμένου, βελτιώνοντας τη σταθερότητα και την ακρίβεια των προβλέψεων του δικτύου. Οι διαφορετικοί τύποι συγκέντρωσης φαίνονται στο Σχήμα 4.8.

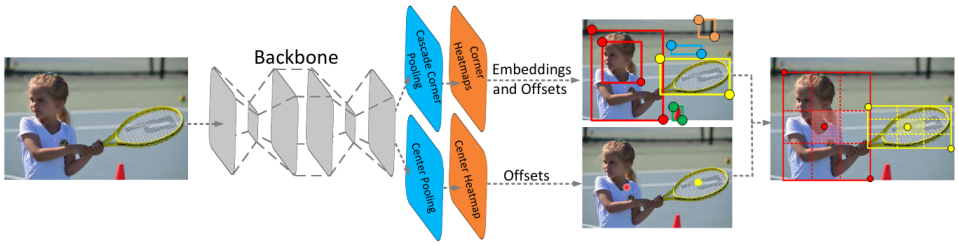
Σχήμα 4.8: Κεντρική συγκέντρωση, Γωνιακή συγκέντρωση και Cascade Corner Pooling αντίστοιχα [4, Σχήμα 4]



Οπότε ακολουθώντας το παράδειγμα του CornerNet αυτή η αρχιτεκτονική χρησιμοποιεί τις ίδιες βάσεις, δηλαδή περνάει την εικόνα μέσα από ένα συνελικτικό νευρωνικό δίκτυο για να βγάλει έναν χάρτη χαρακτηριστικών, και μετά για αυτόν τον χάρτη χαρακτηριστικών δημιουργεί 2 χάρτες θερμότητας (heatmap), έναν για τις γωνιακές συντεταγμένες και έναν χάρτη θερμότητας για τις κεντρικές συντεταγμένες του δικτύου, αυτοί οι χάρτες θερμότητας δημιουργούνται με τις μεθοδολογίες συγκέντρωσης που αναφέρθηκαν, και μετά πάνω σε αυτούς τους χάρτες θερμό-

τητας προσπαθούμε να βρούμε τα σημεία κλειδιά των υποψήφιων περιβαλλόντων κουτιών. Αυτό επιτυγχάνεται με τη χρήση ενός δείκτη αυτοπεποίθησης, ο οποίος υπολογίζεται από τον μέσο όρο της βαθμολογίας αυτοπεποίθησης των τριών σημείων κλειδιών. Μετά διαλέγουμε τα πρώτα περιβάλλοντα κουτιά ανάλογα με την τιμή αυτοπεποίθησης που έχουν, και στη συνέχεια απορρίπτουμε αυτά στα οποία τα γωνιακά σημεία κλειδιά τους έχουν μεγάλη απόσταση στα διανύσματα ένθεσης (embedding vectors) τους, διότι υπάρχει μεγάλη πιθανότητα να ανήκουν σε διαφορετικά αντικείμενα. Και τέλος, στο τελευταίο βήμα βλέπουμε αν το περιβάλλων κουτί που έχει προβλεφθεί έχει ένα σημείο κλειδί της ίδιας κλάσης στη κεντρική περιοχή του. Αυτό το τελικό βήμα είναι ένας επιπλέον παράγοντας που διαφοροποιεί το CenterNet από το CornerNet με αποτέλεσμα να υπάρχουν σημαντικές αυξήσεις στα μετρικά ακρίβειας. Αυτή η ροή περιγράφεται στο Σχήμα 4.9.

Σχήμα 4.9: Αρχιτεκτονική CenterNet [4, Σχήμα 2]

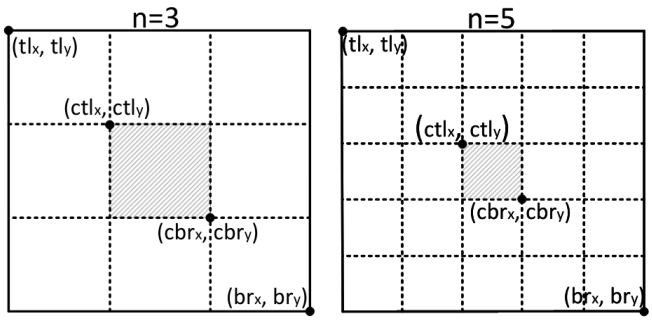


Όσο αναφορά την επαλήθευση των περιβάλλων κουτιών με βάση τα κεντρικά σημεία κλειδιά, οι συγγραφείς έχουν ορίσει μια μεταβλητή n που σχετίζεται με το μέγεθος του αντικειμένου, επειδή όσο μικραίνει το μέγεθος ενός αντικειμένου σε σχέση με το μέγεθος μιας εικόνας, μικραίνει η πληροφορία που μπορούμε να χρησιμοποιήσουμε. Γι' αυτόν τον λόγο χρειάζεται να είμαστε λιγότερο αυστηροί με τα κεντρικά σημεία κλειδιά των αντικειμένων που πιάνουν λίγο χώρο στην εικόνα και περισσότερο αυστηροί όταν τα αντικείμενα καταλαμβάνουν περισσότερο χώρο στην εικόνα. Οπότε χρησιμοποιώντας τη μεταβλητή n μπορούμε να καθορίσουμε τις συντεταγμένες της κεντρικής περιοχής με τη συνάρτηση 4.10.

$$\begin{cases} ctl_x = \frac{(n+1)tl_x+(n-1)br_x}{2n} \\ ctl_y = \frac{(n+1)tl_y+(n-1)br_y}{2n} \\ cbr_x = \frac{(n-1)tl_x+(n+1)br_x}{2n} \\ cbr_y = \frac{(n-1)tl_x+(n+1)br_x}{2n} \end{cases} \tag{4.10}$$

Στη συνάρτηση 4.10 το tl_x, tl_y και br_x, br_y είναι οι συντεταγμένες των πάνω αριστερά και κάτω δεξιά γωνιών του υποψήφιου περιβάλλοντος κουτιού, και το ctl_x, ctl_y και cbr_x, cbr_y είναι οι συντεταγμένες των πάνω αριστερά και κάτω δεξιά γωνιών της κεντρικής περιοχής που θα κάνουμε την επαλήθευση του υποψήφιου περιβάλλοντος κουτιού. Οι τιμές του n που επιλέχθηκαν είναι οι εξής: 3 αν το μέγεθος του περιβάλλοντος κουτιού να είναι μικρότερο από 150 εικονοστοιχεία και 5 αν είναι μεγαλύτερο η ίσο από 150 εικονοστοιχεία. Το Σχήμα 4.10 περιγράφει αυτή τη παραμετροποίηση.

Σχήμα 4.10: Εικονική αναπαράσταση μεγέθους κεντρικής περιοχής σε ένα περιβάλλων κουτί [4, Σχήμα 3]



Για την εκπαίδευση του μοντέλου οι συγγραφείς χρησιμοποιούν τις συναρτήσεις απώλειας που χρησιμοποιεί το CornerNet. Αρχικά πρέπει να οριστεί η συνάρτηση απώλειας για τους χάρτες θερμότητας. Για κάθε σημείο κλειδί υπάρχουν C χάρτες θερμότητας και η τιμή της παραμέτρου αυτής είναι ίση με το πλήθος των κλάσεων που έχει το σύνολο δεδομένων μας, επίσης έχουμε το H και το W που είναι οι διαστάσεις του ύψους και του πλάτους αντίστοιχα. Για την εύρεση των γωνιών κάθε αντικείμενου, ορίζουν οι συγγραφείς μια παράμετρο t οπού αντιπροσωπεύει μια ελάχιστη τιμή IOU για την οποία αν το υποψήφιο περιβάλλων κουτί την ξεπερνάει, τότε θα το λάβουμε υπ' όψιν. Επίσης για να υπολογιστεί το σφάλμα για κάθε γωνία, χρησιμοποιείται μια δισδιάστατη συνάρτηση Gauss και καταλήγουμε στη συνάρτηση εστιακής απώλειας (focal loss) η οποία είναι η 4.11.

$$L_{det} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1 - p_{cij})^{\alpha} log(p_{cij}) & y_{cij} = 1 \\ (1 - y_{cij})^{\beta} (p_{cij})^{\alpha} log(1 - p_{cij}) & y_{cij} \neq 1 \end{cases} \tag{4.11}$$

Στη συνάρτηση 4.11 το p_{cij} είναι η τιμή της συνάρτησης Gauss με βάση το σημείο (i, j) που έχουμε προβλέψει και το y_{cij} είναι η τιμή της συνάρτησης Gauss με βάση

το πραγματικό σημείο. Το N είναι το πλήθος των αντικειμένων στην εικόνα και το α και β είναι υπερπαράμετροι που έχουν ορίσει οι συγγραφείς για να μπορούν να ρυθμίσουν τη συνεισφορά που θα έχει το κάθε σημείο.

Ένα ακόμη πρόβλημα που έχουν λύσει οι συγγραφείς κατά τη διάρκεια της εκπαίδευσης είναι η στρογγυλοποίηση των συντεταγμένων. Αυτό προκύπτει επειδή οι χάρτες θερμότητας έχουν χαμηλότερη ανάλυση από την εικόνα εισόδου. Στα μεγάλα αντικείμενα δεν δημιουργεί σοβαρό πρόβλημα, αλλά στα μικρά αντικείμενα επηρεάζεται σημαντικά το IOU, οπότε για να αντιμετωπίσουν αυτό το πρόβλημα οι συγγραφείς πρόσθεσαν μια συνάρτηση απώλειας που περιγράφει το σφάλμα στρογγυλοποίησης. Για να περιγράψουμε το σφάλμα στρογγυλοποίησης ορίζουμε μια μεταβλητή o_k η οποία ορίζεται από την εξίσωση 4.12.

$$o_k = (\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor) \tag{4.12}$$

Οπότε η συνάρτηση απώλειας στρογγυλοποίησης μπορεί να περιγραφεί με την εξίσωση 4.13 .

$$L_{off} = \frac{1}{n} \sum_{k=1}^{N} \text{SmoothL1Loss}(o_k, \hat{o}_k) \tag{4.13}$$

Στη συνάρτηση 4.11 το $\hat{o}_k$ είναι η τιμή που έχει προβλέψει το δίκτυο.

Και τέλος οι τελευταίες δυο συναρτήσεις απώλειας που χρησιμοποιεί το δίκτυο έχουν ως στόχο την εύρεση των διανυσμάτων ένθεσης. Τα διανύσματα ένθεσης στην ουσία καθορίζουν αν μια πάνω αριστερή και μια κάτω δεξιά γωνία ενός περιβάλλοντος κουτιού ανήκουν στο ίδιο αντικείμενο, όταν τα διανύσματα έχουν μικρή απόσταση αναμεταξύ τους, τότε θεωρούμε ότι ανήκουν οι γωνίες του αντικειμένου στο ίδιο περιβάλλων κουτί, και αντίστροφα, όταν έχουν μεγάλη απόσταση αναμεταξύ τους, ανήκουν σε διαφορετικά περιβάλλοντα κουτιά. Τα διανύσματα αυτά θα τα ορίσουμε ως εξής: e_{t_k} αφορά το διάνυσμα ένθεσης για την άνω αριστερή γωνία του αντικείμενου k, το e_{b_k} αφορά το διάνυσμα ένθεση της κάτω αριστερής γωνίας και τέλος το e_k είναι ο μέσος των δύο αυτών διανυσμάτων. Αξιοποιώντας αυτές τις μεταβλητές μπορούμε να ορίσουμε δυο συναρτήσεις απώλειας, η πρώτη είναι η L_{pull} που δείχνει το σφάλμα αν τα διανύσματα ανήκουν στο ίδιο αντικείμενο και το L_{push} είναι το σφάλμα αν οι γωνίες ανήκουν σε διαφορετικά αντικείμενα. Αυτές οι

δύο συναρτήσεις απώλειας περιγράφονται με τις συναρτήσεις 4.14 και 4.15.

$$L_{pull} = \frac{1}{N}\sum_{k=1}^{N}[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2] \tag{4.14}$$

$$L_{push} = \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{j=1,j\neq k}^{N}\max(0, \Delta - |e_k - e_j|) \tag{4.15}$$

Οπότε, χρησιμοποιώντας τις συναρτήσεις απώλειας 4.11, 4.14, 4.15, 4.13 μπορούμε να συνθέσουμε τη γενική συνάρτηση απώλειας που θα αξιοποιήσει το δίκτυο, καταλήγοντας στη σχέση 4.16.

$$L = L_{det}^{co} + L_{det}^{ce} + \alpha L_{pull}^{co} + \beta L_{push}^{co} + \gamma(L_{off}^{co} + L_{off}^{ce}) \tag{4.16}$$

Στη συνάρτηση 4.16 ο δείκτης co αφορά την απώλεια για τα γωνιακά σημεία και ο ce για το κεντρικό σημείο. Οι μεταβλητές α , β και γ είναι υπερπαράμετροι του δικτύου.

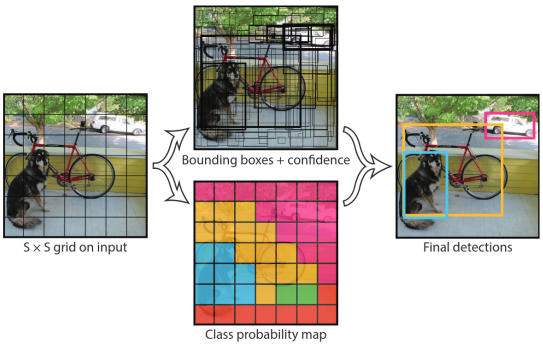
4.5 YoloV4

Η πρώτη έκδοση του Yolo [21] (You Only Look Once) βγήκε το 2016, όπου ο συγγραφέας του ήταν δυσαρεστημένος με τη μεθοδολογία τον τότε μοντέλων, γιατί οι πρώτες υλοποιήσεις ήταν στην ουσία απλά δίκτυα ταξινομητών που διαπερνούσαν την εικόνα με τη μεθοδολογία του ολισθόμενου παραθύρου, και για τις θέσεις που παρήγαγε ο ταξινομητής μια αξιολόγηση με ικανοποιητική τιμή αυτοπεποίθησης τότε θεωρούσαν ότι η περιοχή εκείνη περιέχει ένα αντικείμενο. Αυτή η μεθοδολογία είχε ελάχιστες βελτιστοποιήσεις και ήταν πάρα πολύ αργή. Επίσης εκείνη την εποχή ήταν πολύ διαδεδομένη η χρήση των μοντέλων δύο σταδίων, όπως το [8], οπού στην ουσία τρέχουν δύο ξεχωριστές δομές, η μία είναι υπεύθυνη για την πρόβλεψη των περιβάλλων κουτιών και η άλλη είναι υπεύθυνη για την εύρεση των κλάσεων. Η μεθοδολογίες δύο σταδίων ήταν ανεπιθύμητες στον συγγραφέα γιατί χρειαζόταν να εκπαιδεύσει ξεχωριστά δύο δίκτυα, με αποτέλεσμα να δυσκολεύουν την εύρεση των υπερπαραμέτρων. Οπότε θέλοντας ο συγγραφέας να βελτιώσει την απόδοση, υλοποίησε αυτό το μοντέλο το οποίο ήταν από τα πρώτα που χρησιμοποίησαν τη λογική της μίας διαπέρασης (single shot). Όπου η εύρεση των περιβάλλων κουτιών και

η ταξινόμηση των αντικειμένων που περιείχαν γινόταν μόνο σε ένα στάδιο, απλοποιώντας τη γενική δομή του δικτύου, μειώνοντας τα προβλήματα εκπαίδευσης και παράλληλα να βελτιώνοντας την ταχύτητα εύρεσης. Όλες αυτές οι βελτιώσεις έδωσαν στο δίκτυο τη δυνατότητα να επεξεργάζεται δεδομένα σε πραγματικό χρόνο.

Η Κύρια μεθοδολογία του Yolo είναι το να διαχωρίζει το ύψος και το πλάτος της εικόνας σε $S \times S$ κελιά, και για το κάθε κελί δημιουργούνται B περιβάλλων κουτιά οπού για το καθένα από αυτά υπάρχει ένας δείκτης αυτοπεποίθησης που δείχνει την πιθανότητα να περιέχει ένα αντικείμενο. Αυτή η μεθοδολογία φαίνεται στο Σχήμα 4.11. Η τιμή του δείκτη αυτοπεποίθησης ορίζεται ως $Pr(Object) * IOU_{pred}^{truth}$. Κάθε περιβάλλων κουτί αποτελείται από 5 παραμέτρους, τις συντεταγμένες του κέντρου του (x, y) το πλάτος και το ύψος (w, h) και τέλος τη βαθμολογία αυτοπεποίθησης. Παράλληλα έχουμε μια παράμετρο C όπου αποτελείται από τόσα στοιχεία, όσα το πλήθος των κλάσεων, όπου το κάθε στοιχείο είναι η τιμή αυτοπεποίθησης για μια συγκεκριμένη κλάση.

Σχήμα 4.11: Αναπαράσταση διαχωρισμού της εικόνας [21, Σχήμα 2]



Μετά από την πρώτη έκδοση του Yolo ο συγγραφέας παρατήρησε ότι υπήρχε ένα πρόβλημα σε σχέση με το σύνολο δεδομένων που μπορούν να αξιοποιήσουν τα μοντέλα για τα προβλήματα εύρεσης αντικειμένων. Λόγω αυτής της ανεπάρκειας, ο συγγραφέας αποφάσισε να βελτιώσει την πρώτη έκδοση του Yolo ώστε να μπορεί να χρησιμοποιήσει σύνολα δεδομένων που χρησιμοποιούνται για το πρόβλημα της εύρεσης λεζάντας, και παράλληλα προσθέτοντας μια πληθώρα Από βελτιστοποιήσεις για να κάνει το μοντέλο πιο ανταγωνιστικό, Έτσι λοιπόν δημιουργήθηκε το YoloV2 [22].

Αυτές οι βελτιστοποιήσεις είναι: Κανονικοποίηση παρτίδας (batch normalization) [11], αυτή η τεχνική αυξάνει τη μέση ακρίβεια του δικτύου και βοηθάει στα προβλή-

ματα σύγκλισης που μπορεί να έχουμε κατά τη διάρκεια της εκπαίδευσης. Αύξηση στην ανάλυση του δικτύου ταξινόμησης από 224×224 σε $448x448$. Αντικατάσταση των πλήρως συνδεδεμένων στρωμάτων (fully connected layers) με τη μεθοδολογία των αρχικών περιβαλλόντων κουτιών για την πρόβλεψη των συντεταγμένων, όπου αυτή η δομή είναι εμπνευσμένη από το παράδειγμα του Faster-RCNN με τελικό αποτέλεσμα να μειώνεται λίγο η ακρίβεια αλλά να βελτιώνεται αρκετά η ανάκληση. καλύτερη επιλογή των αρχικών περιβαλλόντων κουτιών χρησιμοποιώντας k-means στο σύνολο δεδομένων με αποτέλεσμα να καθορίσουν 5 διαφορετικές αναλογίες. Χρήση ενός πλέγματος με περισσότερες υποδιαιρέσεις με $S = 26$ ώστε να βελτιώσουν τον εντοπισμό μικρότερων αντικειμένων. Και τέλος, εκπαιδεύουν το δίκτυο με διαφορετικές διαστάσεις ανά 10 παρτίδες (batches) ώστε να μπορεί να είναι αποτελεσματικό σε ένα μεγάλο εύρος διαστάσεων που θα διαμορφώνεται με τις απαιτήσεις και τους πόρους της συσκευής που θα κάνει τις προβλέψεις.

Μετά από το YoloV2 ο συγγραφέας έκανε διάφορες βελτιστοποιήσεις, ώστε να μπορέσει να βελτιώσει την ακρίβεια των αποτελεσμάτων του μοντέλου και το αποτέλεσμα αυτών τον βελτιστοποιήσεων είναι το YoloV3 [23]. Μια αδυναμία που εξακολουθούσε να υπάρχει στις προηγούμενες εκδόσεις αυτού του μοντέλου ήταν η μη ακριβής πρόβλεψη συσσωρευμένων μικρών αντικειμένων, αυτή η αδυναμία ωφελούνταν στη θεμελιώδη αρχιτεκτονική του μοντέλου, επειδή χρησιμοποιεί έναν περιορισμένο αριθμό περιβαλλόντων κουτιών. Για να προσπαθήσει να ελαχιστοποιήσει αυτό το πρόβλημα, ο συγγραφέας παραδειγματίσθηκε από τον τρόπο λειτουργίας των αρχιτεκτονικών όπως το FPN [14] που για μια εικόνα εξάγουν διαφορετικούς χάρτες χαρακτηριστικών από διαφορετικά επίπεδα του δικτύου στους οποίους προσπαθούν να προβλέψουν τα διαφορετικά περιβάλλοντα κουτιά. Οπότε ακολουθώντας αυτήν τη λογική, οι συγγραφείς επιλέξανε 3 διαφορετικούς χάρτες χαρακτηριστικών για να καλύψουν 3 βασικές κλίμακες. Επίσης μια ακόμη βελτιστοποίηση που έχει γίνει στο δίκτυο είναι η δημιουργία ενός θεμελιώδους νευρωνικού δικτύου με περισσότερα στρώματα και τη χρήση υπολειπόμενων δικτύων (residual networks) για τη διατήρηση της ταχύτητας.

Μετά την τρίτη έκδοση του Yolo, ο συγγραφέας του αρχικού σταμάτησε την ενασχόληση του με τον τομέα της εύρεσης αντικειμένων, οπότε η τέταρτη έκδοση του μοντέλου, η YoloV4 έχει προέλθει από άλλους συγγραφείς, οι οποίοι έχουν κα-

ταφέρει να προσθέσουν και άλλες βελτιστοποιήσεις στο δίκτυο αυτό, αυξάνοντας την ακρίβεια του δικτύου και διατηρώντας την ταχύτητα του. Οι βελτιώσεις που πρόσφεραν οι καινούργιοι συγγραφείς στο μοντέλο είναι η ενίσχυση του δικτύου ραχοκοκαλιάς με τη μεθοδολογία CSP [31] για την αύξηση της ταχύτητας του μοντέλου. Για τον λαιμό του δικτύου χρησιμοποίησαν τη δομή SPP [9] διότι τους δίνει τη δυνατότητα να αυξήσουν το υποδεκτικό πεδίο χωρίς να επιβαρύνεται η ταχύτητα του δικτύου. Μετά από τη δομή SPP υπάρχει η δομή [15] για τον συνδυασμό της πληροφορίας από τα διαφορετικά επίπεδα του δικτύου. Και τέλος η κεφαλή για την εύρεση περιβαλλόντων κουτιών ακολουθεί την ίδια μεθοδολογία με το YoloV3.

Έχουν γίνει και κάποιες επιπρόσθετες βελτιστοποιήσεις, οι οποίες είναι: Πρόσθεση καινούργιων μεθοδολογιών για τον εμπλουτισμό δεδομένων εκπαίδευσης (data augmentation) ποιο συγκεκριμένα, Mosaic Augmentation όπου στην ουσία συνδυάζονται τα δεδομένα από 4 διαφορετικές εικόνες με τα δεδομένα τους. Μια καινούργια τεχνική αυτό-αντιμέτωπης εκπαίδευσης (SAT, Self-Adversarial training), όπου το δίκτυο κατά τη διαδικασία της εκπαίδευσης λειτουργεί με δύο στάδια τα οποία εναλλάσσει, ένα στάδιο που χρησιμοποιεί το δίκτυο με έναν τρόπο που τροποποιεί την εικόνα ώστε να μην υπάρχουν αντικείμενα σε αυτή, και ένα στάδιο εύρεσης αντικειμένων με είσοδο την τροποποιημένη εικόνα. Επίσης έχουν ενημερώσει το κομμάτι που αφορούσε την κανονικοποίησης παρτίδας (batch normalization) [11] [33] για ταχύτερη εκπαίδευση. Και έχει επίσης προστεθεί μια δομή SAM [32] για τη βελτιστοποίηση ακρίβειας με μικρό κόστος στην ταχύτητα του δικτύου.

Οπότε στην ουσία το YOLOv4 είναι ένα μοντέλο το οποίο έχει περάσει από πολλά στάδια βελτιστοποιήσεων και αλλαγών, και έχει καταφέρει να είναι από τα πιο αποτελεσματικά δίκτυα σε ότι αφορά ακρίβεια σε συνδυασμό με την ταχύτητα εκπαίδευσης και εύρεσης.

Κεφάλαιο 5

Υπολογιστική μελέτη

5.1 Μεθοδολογία

Τα μοντέλα που έχουν συγκριθεί είναι τα ακόλουθα:

Faster-RCNN [24] με ανάλυση εικόνας εισόδου 512×512 και 1024×1024.

SSD-ResNet [17][16] με ανάλυση εικόνας εισόδου 512×512 και 1024×1024.

EfficientDet [28] με παράμετρο ϕ για τις τιμές $0, 1, 2, 3, 4, 5, 6, 7$.

CenterNet [4] με ανάλυση εικόνας εισόδου 512×512 και 1024×1024.

YoloV4 [2] με ανάλυση εικόνας εισόδου 416×416, 512×512 και 608×608.

Όλα τα μοντέλα είναι προεκπαιδευμένα με βάση το COCO train2017 σύνολο δεδομένων και η αξιολόγηση τους έχει γίνει πάνω στο σύνολο δεδομένων COCO val2017 [3]. Η αξιολόγηση των μοντέλων έχει γίνει με βάση τις μετρικές Pascal VOC [5] και COCO [3] χρησιμοποιώντας το ίδιο εργαλείο που χρησιμοποιήθηκε στο [20].

Το πείραμα εκτελέστηκε σε κάρτα γραφικών GTX 1080Ti με έκδοση CUDA 11.4. Το τρέξιμο των μοντέλων Faster-RCNN, SSD-ResNet, EfficientDet, CenterNet έγινε με Python 3 και βιβλιοθήκες Tensorflow-2.4.1 και Tensorflow-gpu-2.4.1, και η εκτίμηση του μοντέλου YoloV4 έγινε με C++ 11 και βιβλιοθήκες OpenCV-4.5.2 και CUDNN 8.2.1.

Το σύνολο δεδομένων που χρησιμοποιήθηκε για την αξιολόγηση των μοντέλων αποτελείται από 5000 εικόνες, οι οποίες έχουν στο σύνολο 36781 περιβάλλοντα κουτιά. Επίσης το σύνολο δεδομένον στο οποίο εκπαιδεύτηκαν τα μοντέλα αποτελείται από 118287 εικόνες, η οποίες έχουν στο σύνολο 860001 περιβάλλοντα κουτιά. Η

κατανομή των κλάσεων είναι στο Σχήμα 5.1. Και τέλος η ταχύτητα πρόβλεψης υπολογίστηκε από το πόσο γρήγορα έβγαλε αποτελέσματα ο κάθε αλγόριθμος για το κάθε συγκεκριμένο μοντέλο.

Μπορεί να παρατηρήσει κάποιος από το Σχήμα 5.1 ότι δεν υπάρχει ίσος αριθμός περιβαλλόντων κουτιών για κάθε κλάση, και στο σύνολο δεδομένων αξιολόγησης, και στο σύνολο δεδομένων εκπαίδευσης. Οπότε θα χρειαστεί να το έχουμε αυτό υπ' όψιν για τις παρακάτω συγκρίσεις.

Όπως μπορούμε να παρατηρήσουμε από το Σχήμα 5.2 μπορούμε να δούμε ότι το πλήθος των δεδομένων εκπαίδευσης δεν έχει κάποια σημαντική επιρροή στη μέση ακρίβεια τον μοντέλων. Οι κλάσεις, όπως "toaster" και "hair dryer", παρόλο που έχουν ένα πολύ μικρό πλήθος περιβαλλόντων κουτιών στα δεδομένα εκπαίδευσης (για την κλάση "toaster" 225 και για τη κλάση "hair dryer" 198), έχουν μεγάλες διαφορές στο μετρικό της μέσης ακρίβειας, και γενικότερα παρατηρώντας το γράφημα μπορούμε να δούμε ότι δεν υπάρχει κάποια δυνατή συσχέτιση μεταξύ του πλήθους δεδομένων και μέσης ακριβείας, δηλαδή η ακρίβεια κατά κλάση επηρεάζεται σημαντικότερα και από άλλους παράγοντες.

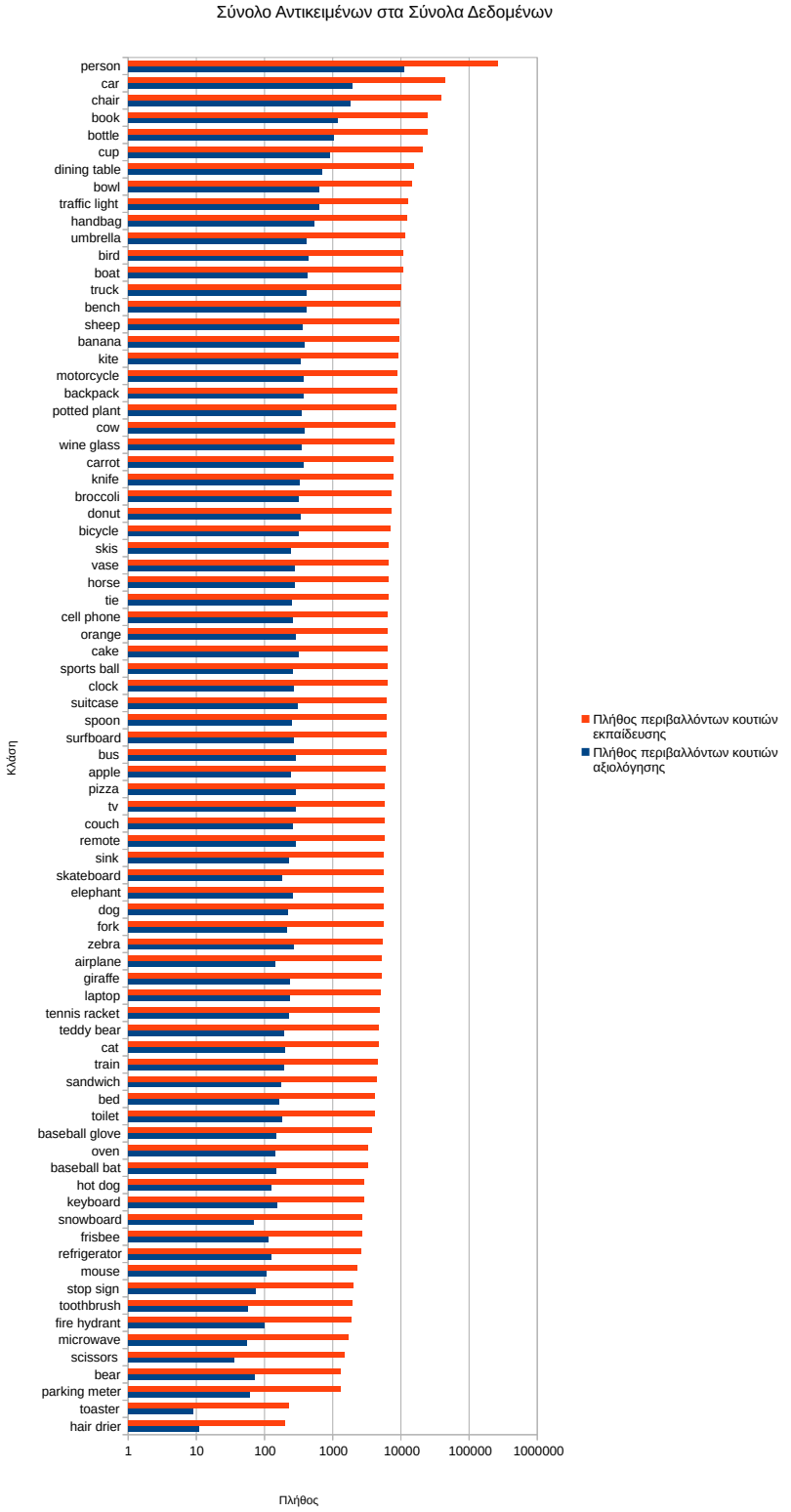
5.2 Μέση Ακρίβεια Μοντέλων ανά Κλάση

Η μέση ακρίβεια ακρίβεια ανά κλάση προκύπτει από τα μετρικά που χρησιμοποιεί το Pascal VOC [5] όπου για να θεωρηθεί αληθώς θετικό ένα αποτέλεσμα για ένα αντικείμενο, θα πρέπει το IOU του περιβάλλοντος κουτιού να είναι μεγαλύτερο του 0.5 και η προβλεπόμενη κλάση να είναι η σωστή.

Αρχικά θα εξετάσουμε αν κάποιες αρχιτεκτονικές εντοπίζουν κάποιες κλάσεις πιο εύκολα από άλλες. Για να το πετύχουμε αυτό θα πάρουμε τον μέσο όρο για κάθε αρχιτεκτονική με τις διαφορετικές παραμετροποιήσεις της, και μετά θα τις συγκρίνουμε ανά κλάση αυτή η σύγκριση είναι στο Σχήμα 5.3.

Από το Σχήμα 5.3 μπορούμε να δούμε ότι γενικά το πιο αδύναμο μοντέλο, που η μέση ακρίβεια του είναι χαμηλότερη από όλα τα άλλα είναι το Faster-RCNN [24], πράγμα που δείχνει την ηλικία του. Πέρα από τους χαμηλούς δείκτες μέσης ακρίβειας που έχει, η κάθε τιμή μέσης ακρίβειας για όλες τις κλάσεις που έχει προβλέψει είναι κάτω από τον μέσο όρο. Παράλληλα μπορούμε να δούμε ότι για μερικές κλάσεις η τιμή μέση ακρίβειας είναι η μισή σε σύγκριση με τον μέσο όρο, ποιο χα-

Σχήμα 5.1: Κατανομή κλάσεων και περιβαλλόντων κουτιών στα σύνολα δεδομένων αξιολόγησης και εκπαίδευσης



ρακτηριστικά είναι οι κλάσεις ”refrigerator”, ”donut” και ”cow”. Αυτή η δραματική διαφορά μπορεί να οφείλεται στον τρόπο εντοπισμού του αντικειμένου, επειδή το

Faster-RCNN [24] κάνει την εκπαίδευση του και την πρόβλεψη αντικειμένων σε δύο στάδια, μπορεί να υπάρχει κάποια αστάθεια με την αύξηση των παραμέτρων. Τα υπόλοιπα δίκτυα που είναι σχεδιασμένα για ένα στάδιο, δεν παρουσιάζουν αυτή την ιδιαιτερότητα στη μέση ακρίβειά τους για αυτές τις κλάσεις.

Επίσης μια άλλη ιδιαιτερότητα που γίνεται εμφανής από το Σχήμα 5.3 είναι μερικές προβλέψεις από το CenterNet, όπου παρόλο που φαίνεται πιο αδύναμο γενικά σε σχέση με τα άλλα δίκτυα, υπάρχουν μερικές κλάσεις για τις οποίες η μέση ακρίβεια είναι μεγαλύτερη σε σχέση με τον μέσο όρο, οι κλάσεις αυτές είναι οι ”sports ball”, ”skis” και ”traffic light”. Μπορεί λοιπόν να παρατηρηθεί ότι το CenterNet[4] έχει μια τάση να εντοπίζει με μεγαλύτερη ευκολία αντικείμενα στα οποία η κεντρική τους περιοχή είναι αρκετά αντιπροσωπευτική για το υπόλοιπο αντικείμενο, πράγμα που οφείλεται στην αρχιτεκτονική του μοντέλου επειδή επιβεβαιώνει το περιβάλλων κουτί με βάση τη κεντρική περιοχή του.

Μια τελευταία παρατήρηση είναι ότι η κλάσεις που έχουν τη μεγαλύτερη ακρίβεια συνήθως αφορούν κλάσεις στις οποίες τα αντικείμενα που περιλαμβάνουν έχουν μικρές εμφανισιακές διαφορές. Δηλαδή οι κλάσεις που αφορούν ζώα όπως η κλάση ”zebra” και ”giraffe” εντοπίζονται με πολύ μεγάλη ακρίβεια διότι δεν υπάρχουν πολλές εμφανισιακές διαφορές ανάμεσα στα δείγματα που υπάρχουν στο σύνολο δεδομένων.

Το Faster-RCNN συνεχίζει να έχει κάποιες ιδιαιτερότητες, ακόμα και σε σύγκρισή με τον εαυτό του. Διότι για μερικές κλάσεις που προβλέπει, υπάρχει σημαντική πτώση ή αύξηση στο μετρικό της μέσης ακρίβειας για τις διαφορετικές αναλύσεις εισόδου του, οι οποίες είναι πολύ μεγαλύτερες σε σχέση με τις προβλέψεις για τις υπόλοιπες κλάσεις. Επίσης, ενώ υπάρχει ο γενικός κανόνας ότι η μεγαλύτερη ανάλυση εικόνας εισόδου έχει ως αποτέλεσμα μεγαλύτερη μέση ακρίβεια. Υπάρχουν περιπτώσεις για τις οποίες ο κανόνας παραβιάζεται σε μεγάλο βαθμό. Δηλαδή μπορούμε από το Σχήμα 5.4 να δούμε ότι για την κλάση ”sheep”, η ακρίβεια του μοντέλου με ανάλυση εισόδου 512×512 είναι 25% μεγαλύτερη από το μοντέλο με ανάλυση εισόδου 1024×1024. Άλλη μια ιδιαιτερότητα είναι ότι στη κλάση ”toaster” η ακρίβεια του μοντέλου με τη μεγαλύτερη ανάλυση εισόδου έχει 40% διαφορά με τη μέση ακρίβεια του μοντέλου που έχει τη μικρότερη ανάλυση. Οπότε σαν μοντέλο παρουσιάζει απρόβλεπτη συμπεριφορά σε ορισμένες περιπτώσεις, πράγμα

που μπορεί να οφείλεται στο γεγονός ότι είναι δίκτυο δύο σταδίων.

Από το Σχήμα 5.5 μπορούμε να δούμε η ιδιαιτερότητα για την κλάση "toaster" συνεχίζει να υπάρχει, δηλαδή το μοντέλο με την ανάλυση εισόδου 1024×1024 έχει αρκετά μεγαλύτερη μέση ακρίβεια από το μοντέλο με ανάλυση εισόδου $512x512$. Επίσης υπάρχει μια σημαντική διαφορά για την κλάση "bed" όπου το μοντέλο με τις λιγότερες παραμέτρους έχει μεγαλύτερη μέση ακρίβεια.

Στο EfficientDet[28] δεν υπάρχουν περιπτώσεις στις οποίες τα μοντέλα με της λιγότερες παραμέτρους να έχουν πολύ μεγαλύτερη ακρίβεια σε σχέση με αυτά που έχουν μεγαλύτερο πλήθος παραμέτρων. Αλλά αξίζει να σημειωθεί ότι παρόλο που στις περισσότερες περιπτώσεις η διαφορά της μέσης ακρίβειας μεταξύ των μεγαλύτερων μοντέλων μικρότερων μοντέλων είναι σημαντική. Υπάρχουν Περιπτώσεις όπου η αύξηση του πλήθους παραμέτρων δε βελτιώνει σημαντικά τη μέση ακρίβεια. Αυτό μπορούμε να το δούμε στις κλάσεις "bed","brocoli" και "dining table".

Το SSD-ResNet έχει τη μικρότερη διαφορά στη μέση ακρίβεια για κάθε έκοδση του σε σχέση με τα άλλα μοντέλα διότι η διαφορά στην ανάλυση εικόνας εισόδου είναι μικρότερη για τις διαφορετικές παραμετροποιήσεις του, σε σχέση με τα υπόλοιπα μοντέλα. Η ελάχιστη ανάλυση εικόνας εισόδου είναι 640×640 και η μέγιστη είναι 1024×1024. Επίσης παρατηρούμε και εδώ, ότι υπάρχει ακόμη η ιδιαιτερότητα με το μοντέλο που έχει τις λιγότερες παραμέτρους να έχει παρόμοια η καλύτερη μέση ακρίβεια για τις κλάσεις "bed", "brocoli".

Και τέλος για το μοντέλο YoloV4 μπορούμε να δούμε από το Σχήμα 5.8 ότι η πτώση μέσης ακρίβειας ανά κλάση είναι πολύ μικρότερη σε σχέση με τα υπόλοιπα μοντέλα, επειδή, ομοίως με το SSD-ResNet το μοντέλο έχει μικρή διαφορά στη μέση ακρίβεια για της διαφορετικές παραμετροποιήσεις. Δηλαδή η ελάχιστη ανάλυση εισόδου είναι 416×416 και η μέγιστη 608×608 Επίσης είναι αξιοσημείωτο το γεγονός ότι ακόμα και στις κλάσεις με μικρό πλήθος δεδομένων εκπαίδευσης, η αρχιτεκτονική αυτή καταφέρνει και διατηρεί αξιοπρεπείς ποσοστά ακρίβειας, αυτό μπορεί να οφείλεται στους καινούργιους τρόπους επαύξησης δεδομένων (data augmentation) που χρησιμοποιεί το που χρησιμοποιεί το YoloV4.

5.3 Ακρίβεια και Ανάκληση ανά Μέγεθος

Ένας άλλος τρόπος για να εξετάσουμε τα μοντέλα είναι η μέση ακρίβεια και ανάκληση ανά μέγεθος αντικειμένου. Αυτά τα μετρικά έχουν βγει σύμφωνα με τις προδιαγραφές του διαγωνισμού COCO [3] και θα μας βοηθήσουν στο να βρούμε γενικές αδυναμίες ανά περιοχή που έχει το κάθε μοντέλο.

Βλέποντας το Σχήμα 5.9 μπορούμε να δούμε ότι η αύξηση της ανάλυσης εικόνας εισόδου έχει πάντα θετικά αποτελέσματα για κάθε περιοχή. Η μόνη εξαίρεση σε σε αυτόν τον κανόνα είναι το YoloV4 που για την ανάλυση εικόνας εισόδου 608×608 βλέπουμε ότι υπάρχει μια μείωση στη μέση ακρίβεια που αφορά τα αντικείμενα που καταλαμβάνουν μεγάλο χώρο στην εικόνα. Αυτό οφείλεται στο γεγονός ότι το υποδεκτικό πεδίο για τη συγκεκριμένη παραμετροποίηση δεν αυξάνεται με την αύξηση της ανάλυσης της εικόνας, με αποτέλεσμα να υπάρχει μια μικρή πτώση μέσης ακρίβειας.

Ένα άλλο συμπέρασμα που μπορούμε να βγάλουμε είναι ότι γενικά όλα τα μοντέλα έχουν αδυναμία στην πρόβλεψη αντικειμένων που έχουν μικρά περιβάλλοντα κουτιά. Αξίζει να σημειωθεί ότι οι παραμετροποιήσεις του EfficientDet που έχουν τη μεγαλύτερη ανάλυση εικόνας εισόδου, έχουν μεγαλύτερη ευκολία στην εύρεση αντικειμένων σε σχέση με τα υπόλοιπα μοντέλα.

Η μέση ανάκληση ανά μοντέλο και περιοχή φαίνεται στο Σχήμα 5.10. Η μέση ανάκληση υπολογίζεται για τους λόγους επικάλυψης από 0.50 έως 0.95, και στα σημεία που αναφερόμαστε για μέση ανάκληση για ένα πλήθος προβλέψεων, σημαίνει ότι αφήνουμε το δίκτυο να κάνει μέχρι n προβλέψεις και στην συνέχεια βλέπουμε την ανάκληση που έχει για ως προς το σύνολο αντικειμένων. Άρα όταν αφήνουμε μικρό αριθμό προβλέψεων, ο οποίος είναι μικρότερος από το πλήθος αντικειμένων στην εικόνα, η μέση ανάκληση θα είναι αναγκαστικά μικρότερη της μονάδας.

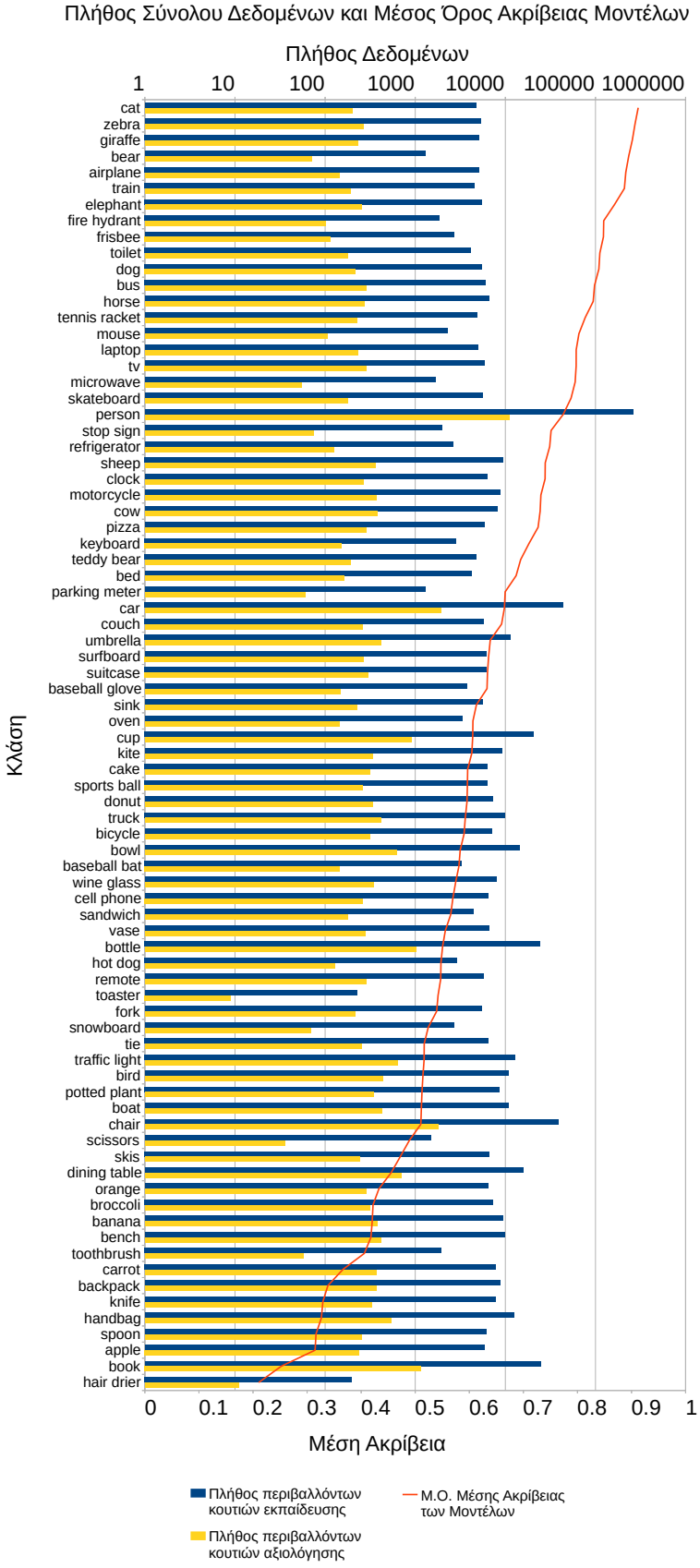
Παρατηρώντας λοιπόν το Σχήμα 5.9 μπορούμε να δούμε ότι η κατηγορία που ευνοείται περισσότερο με την αύξηση ανάλυσης εισόδου είναι η ανάκληση που αφορά μικρά αντικείμενα. Αυτό οφείλεται στο γεγονός ότι όταν έχουμε μεγαλύτερη ανάλυση εισόδου μπορούμε και έχουμε μεγαλύτερη ακρίβεια σε ότι αφορά την τοποθεσία του αντικειμένου και του Σχήματος του. Επίσης για το YoloV4 εξακολουθεί να υπάρχει η ιδιαιτερότητα που είχε για τη μέση ακρίβεια, όταν λειτουργεί με ανά-

λυση εισόδου 608×608 βλέπουμε ότι υπάρχει μια μικρή πτώση στη μέση ανάκληση μεγάλων αντικειμένων, που πάλι οφείλεται στη μη αύξηση του υποδεκτικού πεδίου.

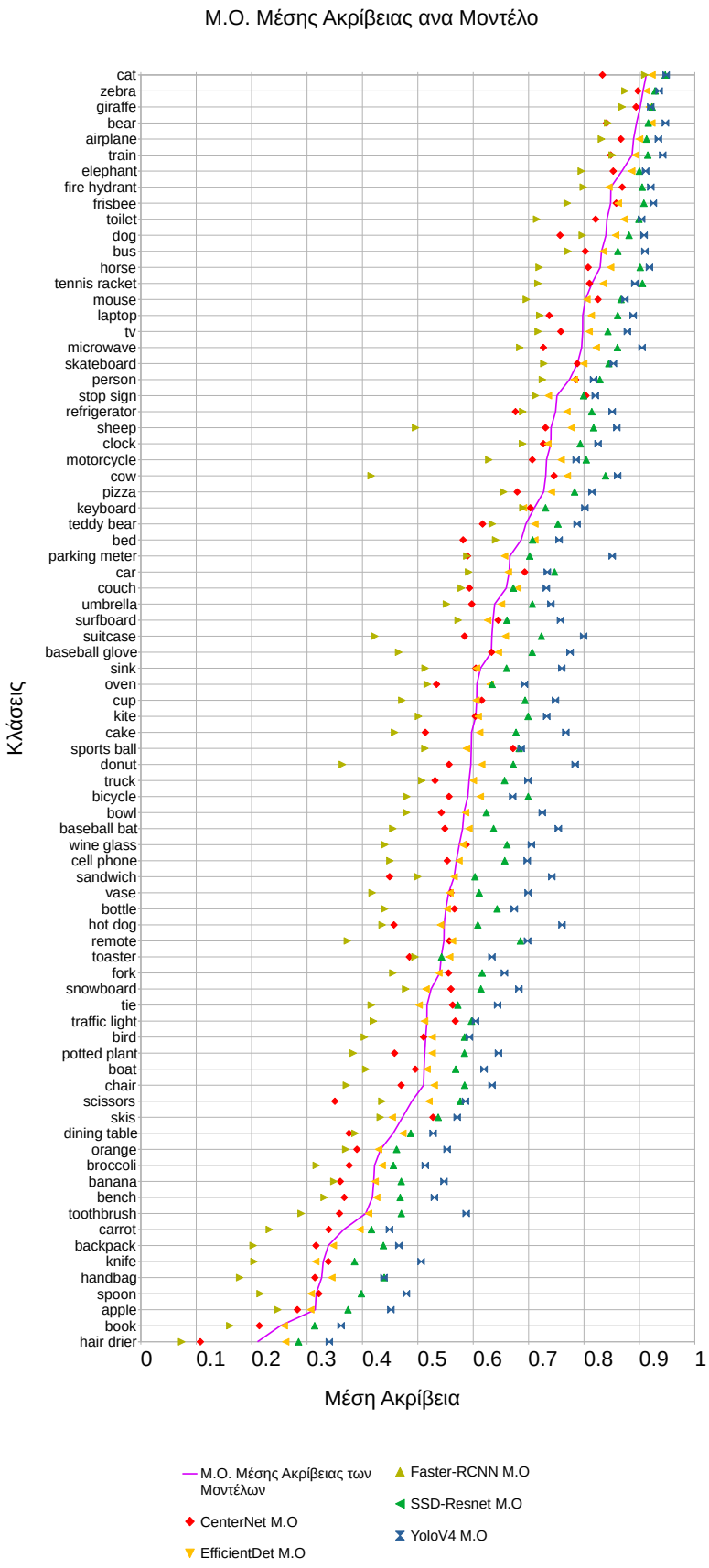
5.4 Ταχύτητα Μοντέλων

Από το Σχήμα 5.11 μπορούμε να δούμε ότι από άποψη ισορροπίας της ταχύτητας εντοπισμού και μέσης ακρίβειας, ο YoloV4 είναι το πιο αποτελεσματικό μοντέλο. Απο άποψη μέσης ακρίβειας, το EfficientDet είναι το πιο αποτελεσματικό μοντέλο με ένα μικρό πλεονέκτημα ως προς το YoloV4. Το CenterNet έχει μεγαλύτερη ακρίβεια και ταχύτητα από το EfficientDet D2 και μετά μένει πίσω στα μετρικά ακρίβειας σε σχέση με τις εκδόσεις του EfficientDet που έχουν περισσότερες παραμέτρους. Επίσης μπορούμε να δούμε ότι τα μοντέλα SSD-ResNet και Faster-RCNN δεν έχουν κάποιο σημαντικό πλεονέκτημα σε σχέση με τις άλλες αρχιτεκτονικές.

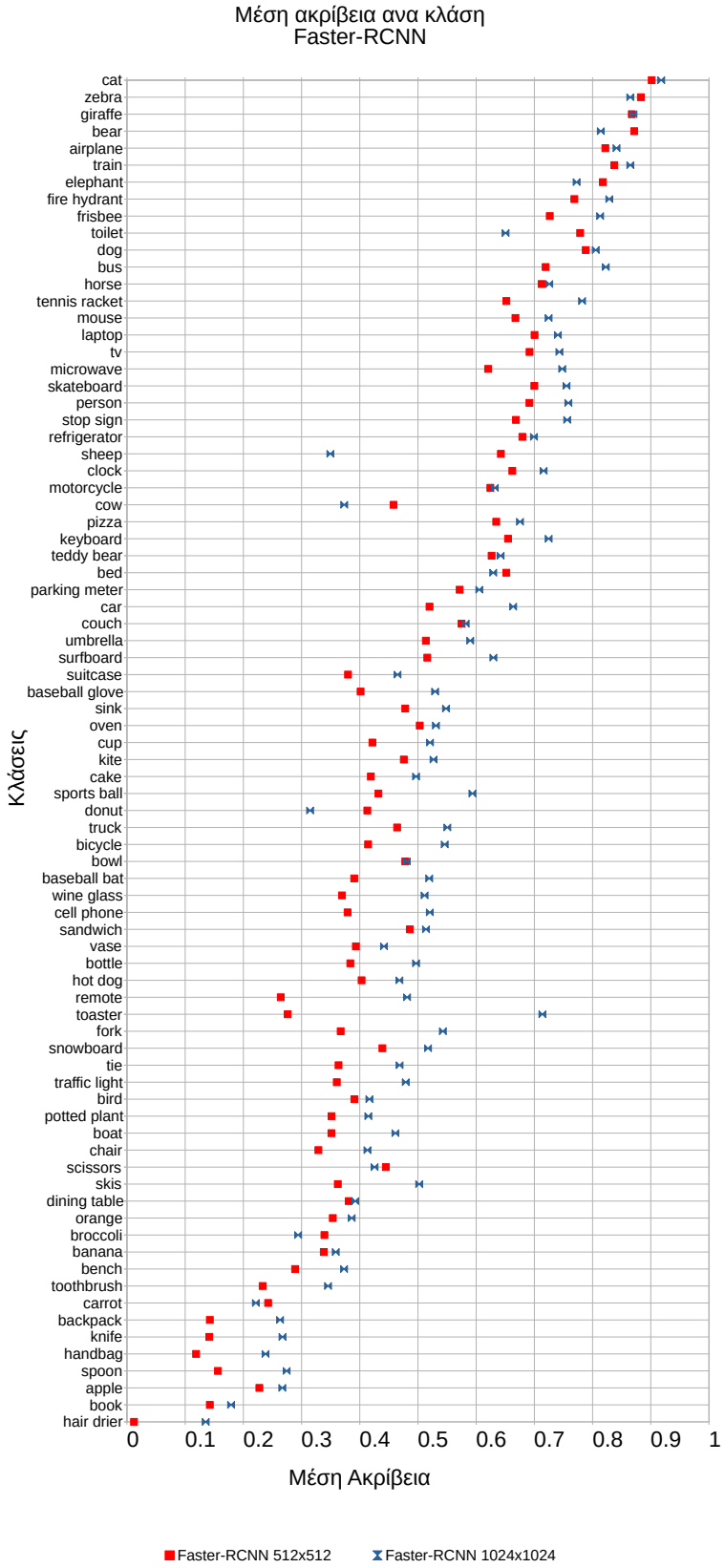
Σχήμα 5.2: Πλήθη συνόλων δεδομένων, ταξινομημένα ανάλογα με τη μέση ακρίβεια



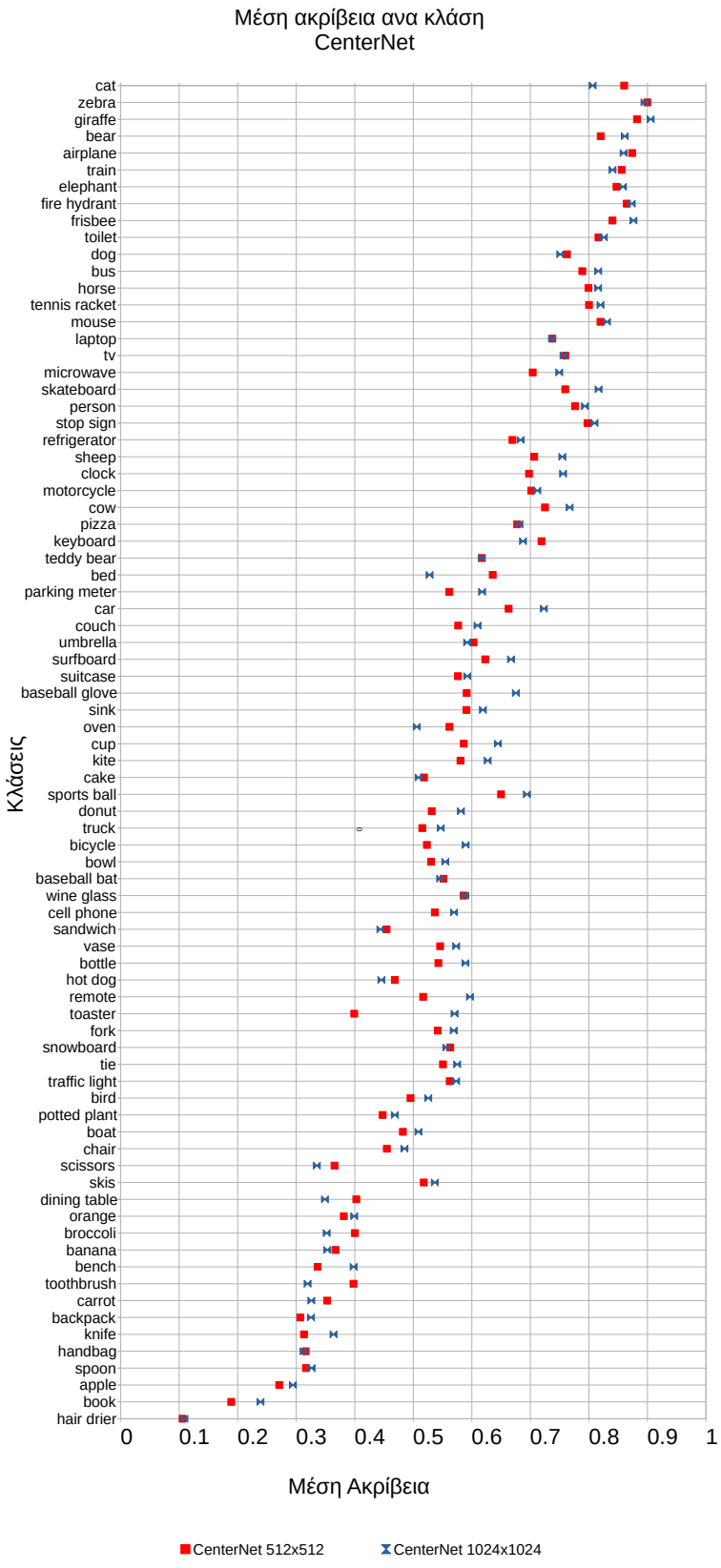
Σχήμα 5.3: Μέσος όρος διαφορετικών παραμετροποιήσεων μοντέλων ανά κλάση



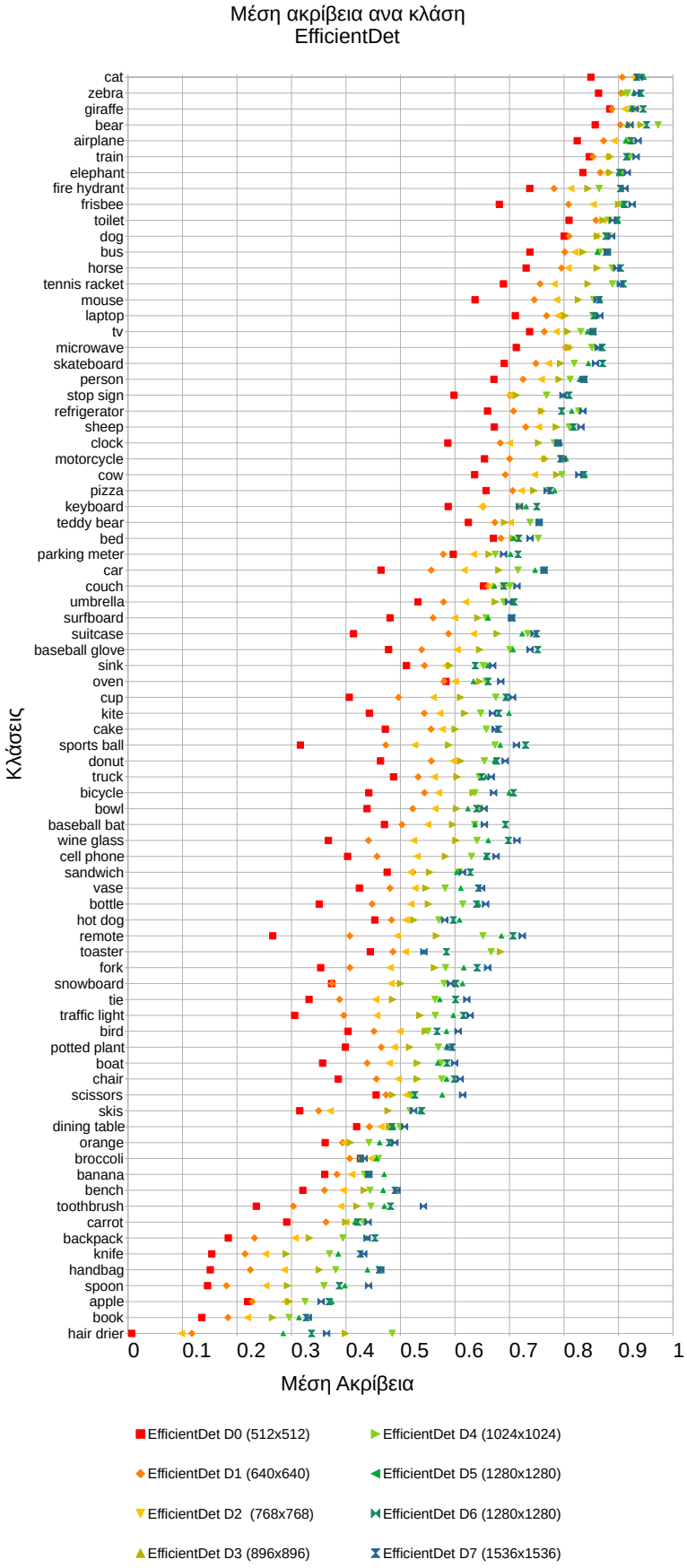
Σχήμα 5.4: Μέση ακρίβεια ανά κλάση για το μοντέλο Faster-RCNN



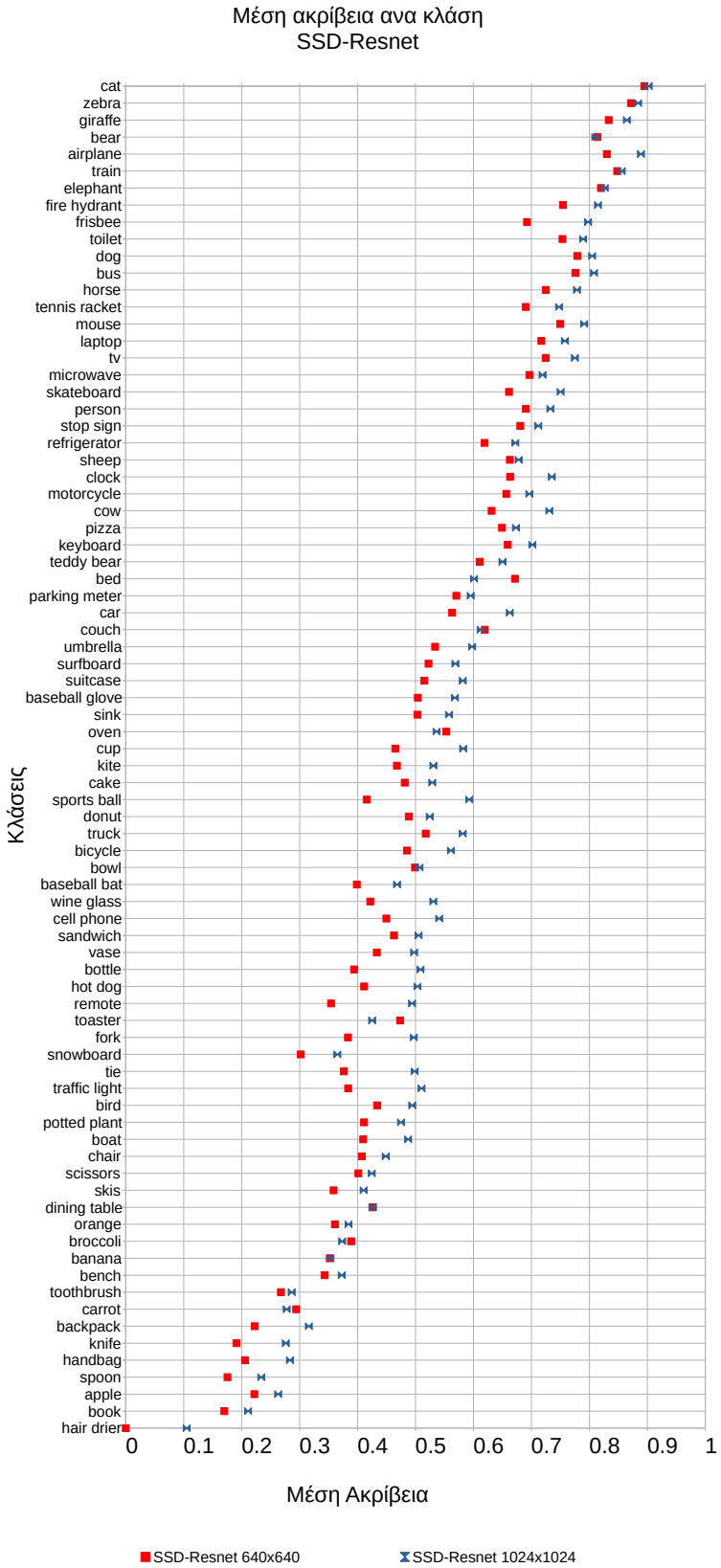
Σχήμα 5.5: Μέση ακρίβεια ανά κλάση για το μοντέλο CenterNet



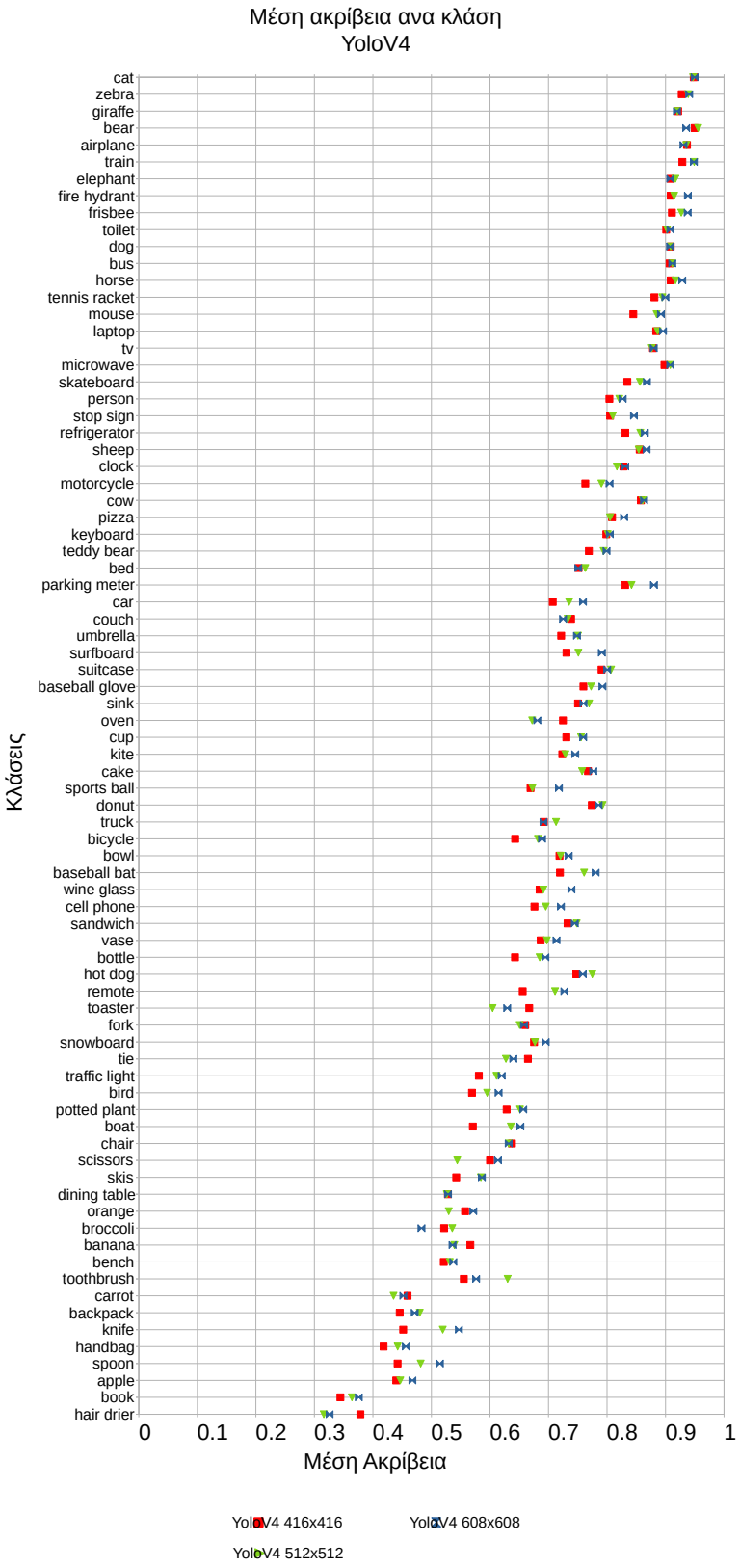
Σχήμα 5.6: Μέση ακρίβεια ανά κλάση για το μοντέλο EfficientDet



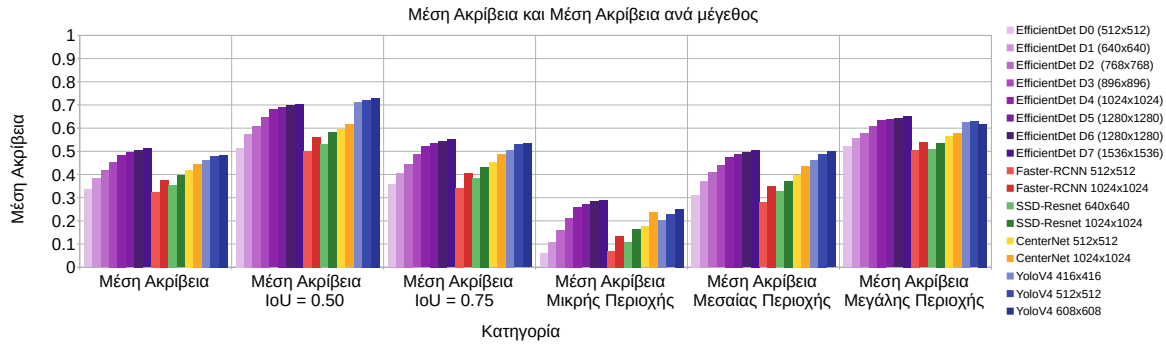
Σχήμα 5.7: Μέση ακρίβεια ανά κλάση για το μοντέλο SSD ResNet



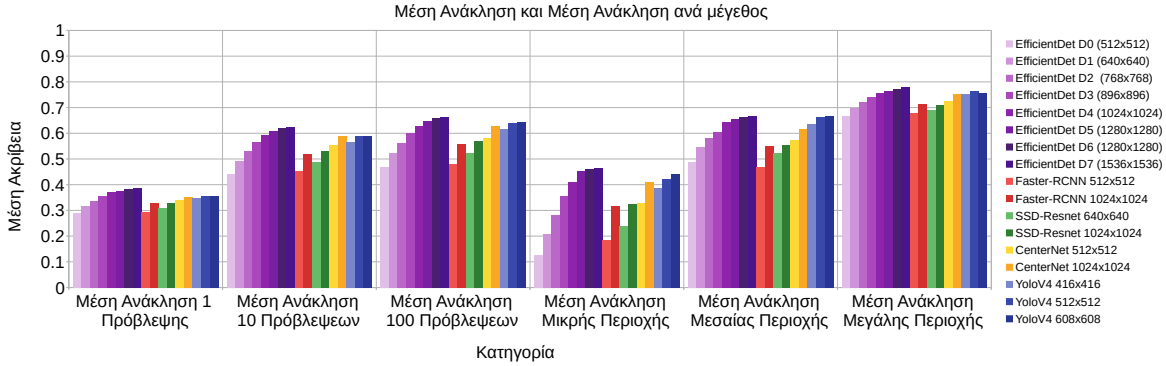
Σχήμα 5.8: Μέση ακρίβεια ανά κλάση για το μοντέλο YoloV4



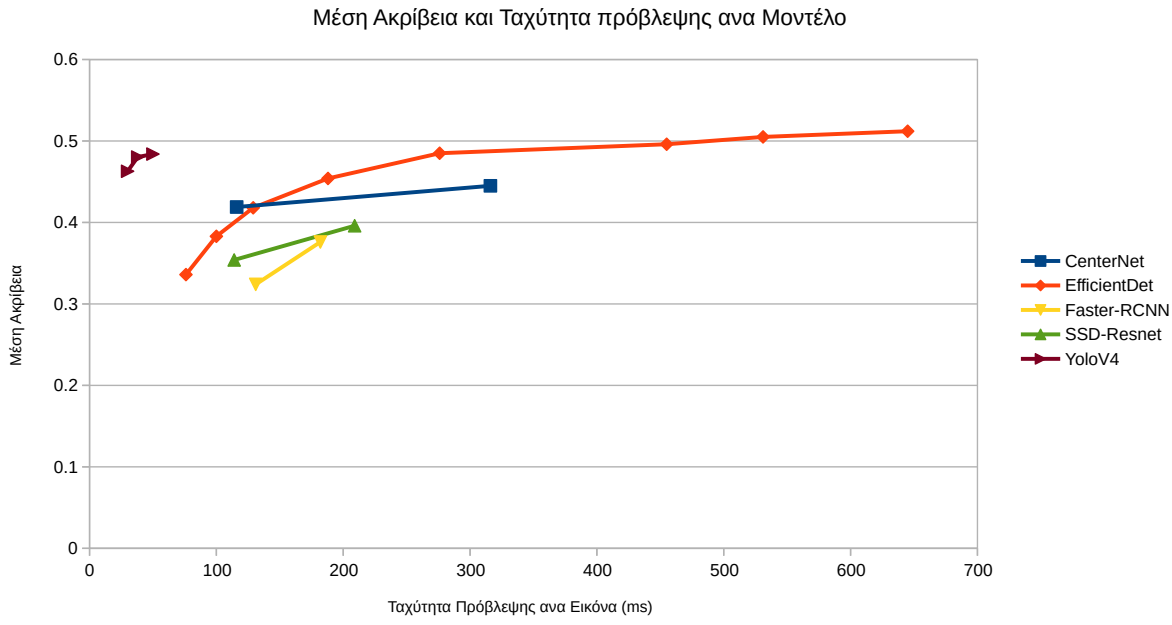
Σχήμα 5.9: Μέση ακρίβεια και μέση ακρίβεια ανά μέγεθος



Σχήμα 5.10: Μέση ανάκληση και μέση ανάκληση ανά μέγεθος



Σχήμα 5.11: Ταχύτητα εντοπισμού μοντέλων, ανάλογα με τη μέση ακρίβεια



Κεφάλαιο 6

Συμπεράσματα

Συνοψίζοντας, σε αυτήν την εργασία έχουμε αναλύσει το πως λειτουργούν τα πιο ευρέως διαδεδομένα μοντέλα εύρεσης αντικειμένων σε εικόνες. Είδαμε για ποιους λόγους έχουν επιλεχθεί κάποιες συγκεκριμένες δομές και το πως τις συνδύασαν οι συγγραφείς για να δημιουργήσουν ένα πιο αποτελεσματικό μοντέλο. Μετά, χρησιμοποιώντας αυτά τα μοντέλα, που έχουν εκπαιδευτεί με βάση το COCO σύνολο δεδομένων [3], εξάγαμε αποτελέσματα και μετρικά ώστε να καταλήξουμε σε κάποια συμπεράσματα, και να δούμε αν εμφανίζονται κάποιες ιδιομορφίες στη συμπεριφορά των μοντέλων λόγω της αρχιτεκτονικής τους.

Τα κύρια συμπεράσματα που καταλήξαμε για τη σύγκριση των μοντέλων είναι τα εξής: η αύξηση του πλήθους παραμέτρων του δικτύου, παρόλο που αυξάνει τη μέση ακρίβεια και τη μέση ανάκληση, αυτό δεν ισχύει πάντα. Πιο συγκεκριμένα, στο μοντέλο YoloV4 παρατηρήσαμε ότι με την αύξηση της ανάλυσης εισόδου έχουμε μια πτώση στην ακρίβεια που αφορά αντικείμενα τα οποία καταλαμβάνουν μεγάλο χώρο στην εικόνα, πράγμα που σημαίνει ότι υπάρχει μείωση του υποδεκτικού πεδίου.

Μια ακόμη παρατήρηση είναι ότι η ανάκληση για τα μικρά αντικείμενα βελτιώνεται με μεγαλύτερο ρυθμό με την αύξηση της ανάλυσης εισόδου, πράγμα που δείχνει ότι χάνεται αρκετή πληροφορία στα μικρότερα αντικείμενα με την μείωση της, και η πρόβλεψη περιβαλλόντων κουτιών είναι αρκετά πιο δύσκολη όταν δουλεύουμε με χαμηλές αναλύσεις.

Επίσης, είδαμε ότι μερικές κλάσεις όπως η κλάση ”bed” δεν επηρεάζεται πολύ από την ανάλυση εισόδου, και σε μερικές περιπτώσεις τα μοντέλα χαμηλότερης ανάλυσης κάνουνε πιο ακριβής προβλέψεις.

Η γενική εικόνα που μας έχουν δώσει τα γραφήματα είναι ότι στις περισσότερες περιπτώσεις τα δυσκολεύουν οι ίδιες κλάσεις, παρόλο που έχουν αρκετά διαφορετικές μεθοδολογίες. Δηλαδή η κλάσεις ''giraffe'' και ''zebra'' πάντα θα εντοπίζονται πιο εύκολα, και οι κλάσεις ''knife'',''handbag'',''toothbrush'' θα δυσκολεύουν τα περισσότερα μοντέλα. Ακόμα και το CenterNet που έχει μια αρκετά ιδιαίτερη προσέγγιση, μόνο για τρεις κλάσεις (''stop sign'',''sports ball'',''traffic light'') έχει διαφορετική συμπεριφορά με τα υπόλοιπα δίκτυα.

Οπότε η κύρια διαφορά ανάμεσα στα μοντέλα είναι η ταχύτητα πρόβλεψης και η μέση ακρίβεια τούς, αυτό σημαίνει ότι αν υπάρχει κάποια ανάγκη για να γίνουν προβλέψεις για μια συγκεκριμένη κλάση, στις περισσότερες περιπτώσεις αρκεί να κοιτάξουμε το μετρικό της μέσης ακρίβειας.

Από τους αλγόριθμους που εξετάσαμε, μπορούμε να δούμε ότι το EfficientDet έχει τη μεγαλύτερη ακρίβεια για όλες τις κατηγορίες των αντικειμένων, αλλά μόνο όταν χρησιμοποιήσουμε τις εκδόσεις του με τις περισσότερες παραμέτρους. Και τέλος, το πιο γρήγορο μοντέλο που συγκρίναμε είναι το YoloV4 το οποίο δεν έχει μόνο μεγάλη ταχύτητα εύρεσης (30ms), αλλά έχει και πολύ με καλά ποσοστά ακρίβειας. Αυτό μας δείχνει ότι ένα μοντέλο μπορεί να γίνει αρκετά αποτελεσματικό αν εφαρμόσουμε έναν μεγάλο αριθμό βελτιστοποιήσεων.

Τελικά μπορούμε να δούμε από τις μεγάλες διαφορές που υπάρχουν από μοντέλο σε μοντέλο, το πόσο γρήγορα εξελίσσονται τα δίκτυα για την εύρεση των αντικειμένων. Μπορούμε επίσης να δούμε ότι όσο περνάει ο χρόνος, η ταχύτητα πρόβλεψης και η μέση ακρίβεια των μοντέλων αυξάνεται. Σε αυτό ευθύνονται και οι καινούργιες μεθοδολογίες και μικρό-βελτιώσεις που ανακαλύπτουν διάφοροι ερευνητές. Αν συνεχίσουν αυτοί οι ρυθμοί εξέλιξης στον κλάδο, θα συνεχίζουμε να βλέπουμε δραματικές βελτιώσεις στην αποδοτικότητα των μοντέλων. Επίσης, μπορεί να βρεθούν καινούργια χρήσιμα μετρικά πέρα από αυτά της μέσης ακρίβειας και ανάκλησης. Επιπλέον με την πάροδο του χρόνου τα εργαλεία που συγκρίνουμε τα μοντέλα και οι τεχνικές υλοποιήσεις τους θα είναι πιο προσβάσιμα, με αποτέλεσμα να κάνει την εκτίμησή τους ακόμα πιο εύκολη και ακριβείς Ήδη γίνονται προσπάθειες για την τυποποίηση των μεθοδολογιών, όπως το Tensorflow και το Pytorch που προσπαθούν να περιγράψουν την κάθε αρχιτεκτονική και τις ιδιαιτερότητες που μπορεί να έχει, με όσο λιγότερο κώδικα γίνεται. Αυτό είναι χρήσιμο διότι τα

μοντέλα όπως το Yolo έχουν υλοποιηθεί με ένα συγκεκριμένο τρόπο με κάποιες συγκεκριμένες βιβλιοθήκες, οπότε στη σύγκριση των μοντέλων, αυτή η διαδικασία προσθέτει ένα στοιχείο θορύβου. Δηλαδή μπορεί μια βιβλιοθήκη να έχει κάποιες βελτιστοποιήσεις ή σφάλματα από έκδοση σε έκδοση. Οπότε στο κοντινό μέλλον θα μπορούμε να αξιολογήσουμε με μεγαλύτερη ακρίβεια τις ιδιαιτερότητες που έχουν τα μοντέλα αυτά, και το πως έχουν εξελιχθεί.

Παραρτήματα

Πίνακας 1: Ταχύτητα εντοπισμού μοντέλων

Μοντέλο	**Ταχύτητα (ms)**
Faster-RCNN 512x512	131
Faster-RCNN 1024x1024	182
CenterNet 512x512	116
CenterNet 1024x1024	316
SSD-Resnet 640x640	114
SSD-Resnet 1024x1024	209
EfficientDet D0 (512x512)	76
EfficientDet D1 (640x640)	100
EfficientDet D2 (768x768)	129
EfficientDet D3 (896x896)	188
EfficientDet D4 (1024x1024)	276
EfficientDet D5 (1280x1280)	455
EfficientDet D6 (1280x1280)	531
EfficientDet D7 (1536x1536)	645
YoloV4 416x416	30
YoloV4 512x512	38
YoloV4 608x608	50

Πίνακας 2: Μετρικά μέσης ακρίβειας και ανάκλησης ανα περιοχή για το μοντέλο Faster-RCNN

Κλάση	**512x512**	**1024x1024**
Μέση Ακρίβεια	0.32	0.38
Μέση Ακρίβεια IoU=0.50	0.50	0.56
Μέση Ακρίβεια IoU=0.75	0.34	0.40
Μέση Ακρίβεια Μικρή Περιοχή	0.07	0.14
Μέση Ακρίβεια Μεσαία Περιοχή	0.28	0.35
Μέση Ακρίβεια Μεγάλη Περιοχή	0.51	0.54
Μέση Ανάκληση 1 πρόβλεψη	0.29	0.33
Μέση Ανάκληση 10 προβλέψεις	0.45	0.52
Μέση Ανάκληση 100 προβλέψεις	0.48	0.56

Μέση Ανάκληση Μικρή Περιοχή	0.18	0.32
Μέση Ανάκληση Μεσαία Περιοχή	0.47	0.55
Μέση Ανάκληση Μεγάλη Περιοχή	0.68	0.71

Πίνακας 3: Μετρικά μέσης ακρίβειας και ανάκλησης ανα περιοχή για το μοντέλο CenterNet

Κλάση	**512x512**	**1024x1024**
Μέση Ακρίβεια	0.42	0.45
Μέση Ακρίβεια IoU=0.50	0.60	0.62
Μέση Ακρίβεια IoU=0.75	0.45	0.49
Μέση Ακρίβεια Μικρή Περιοχή	0.18	0.24
Μέση Ακρίβεια Μεσαία Περιοχή	0.40	0.44
Μέση Ακρίβεια Μεγάλη Περιοχή	0.57	0.58
Μέση Ανάκληση 1 πρόβλεψη	0.34	0.35
Μέση Ανάκληση 10 προβλέψεις	0.55	0.59
Μέση Ανάκληση 100 προβλέψεις	0.58	0.63
Μέση Ανάκληση Μικρή Περιοχή	0.33	0.41
Μέση Ανάκληση Μεσαία Περιοχή	0.57	0.62
Μέση Ανάκληση Μεγάλη Περιοχή	0.72	0.75

Πίνακας 4: Μετρικά μέσης ακρίβειας και ανάκλησης ανα περιοχή για το μοντέλο SSD-ResNet

Κλάση	**512x512**	**1024x1024**
Μέση Ακρίβεια	0.35	0.40
Μέση Ακρίβεια IoU=0.50	0.53	0.58
Μέση Ακρίβεια IoU=0.75	0.39	0.43
Μέση Ακρίβεια Μικρή Περιοχή	0.11	0.16
Μέση Ακρίβεια Μεσαία Περιοχή	0.33	0.37
Μέση Ακρίβεια Μεγάλη Περιοχή	0.51	0.54
Μέση Ανάκληση 1 πρόβλεψη	0.31	0.33
Μέση Ανάκληση 10 προβλέψεις	0.49	0.53
Μέση Ανάκληση 100 προβλέψεις	0.53	0.57

Μέση Ανάκληση Μικρή Περιοχή	0.24	0.33
Μέση Ανάκληση Μεσαία Περιοχή	0.52	0.56
Μέση Ανάκληση Μεγάλη Περιοχή	0.69	0.71

Πίνακας 5: Μετρικά μέσης ακρίβειας και ανάκλησης ανα περιοχή για το μοντέλο YoloV4

Κλάση	**416x416**	**512x512**	**608x608**
Μέση Ακρίβεια	0.46	0.48	0.48
Μέση Ακρίβεια IoU=0.50	0.71	0.72	0.73
Μέση Ακρίβεια IoU=0.75	0.51	0.53	0.53
Μέση Ακρίβεια Μικρή Περιοχή	0.20	0.23	0.25
Μέση Ακρίβεια Μεσαία Περιοχή	0.46	0.49	0.50
Μέση Ακρίβεια Μεγάλη Περιοχή	0.63	0.63	0.62
Μέση Ανάκληση 1 πρόβλεψη	0.35	0.36	0.36
Μέση Ανάκληση 10 προβλέψεις	0.57	0.59	0.59
Μέση Ανάκληση 100 προβλέψεις	0.62	0.64	0.64
Μέση Ανάκληση Μικρή Περιοχή	0.39	0.42	0.44
Μέση Ανάκληση Μεσαία Περιοχή	0.64	0.66	0.67
Μέση Ανάκληση Μεγάλη Περιοχή	0.75	0.76	0.76

Πίνακας 6: Μετρικά μέσης ακρίβειας και ανάκλησης ανα περιοχή για το μοντέλο EfficientDet

Κλάση	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
Μέση Ακρίβεια	0.34	0.38	0.42	0.45	0.49	0.50	0.51	0.51
Μέση Ακρίβεια IoU=0.50	0.52	0.57	0.61	0.65	0.68	0.69	0.70	0.70
Μέση Ακρίβεια IoU=0.75	0.36	0.41	0.45	0.49	0.52	0.54	0.55	0.55
Μέση Ακρίβεια Μικρή Περιοχή	0.06	0.11	0.16	0.21	0.26	0.27	0.29	0.29
Μέση Ακρίβεια Μεσαία Περιοχή	0.31	0.37	0.41	0.44	0.48	0.49	0.50	0.51
Μέση Ακρίβεια Μεγάλη Περιοχή	0.52	0.56	0.58	0.61	0.64	0.64	0.64	0.65
Μέση Ανάκληση 1 πρόβλεψη	0.29	0.32	0.34	0.36	0.37	0.38	0.38	0.39
Μέση Ανάκληση 10 προβλέψεις	0.44	0.49	0.53	0.57	0.59	0.61	0.62	0.62
Μέση Ανάκληση 100 προβλέψεις	0.47	0.53	0.56	0.60	0.63	0.65	0.66	0.66
Μέση Ανάκληση Μικρή Περιοχή	0.13	0.21	0.28	0.35	0.41	0.46	0.46	0.47
Μέση Ανάκληση Μεσαία Περιοχή	0.49	0.55	0.58	0.61	0.64	0.65	0.66	0.67

Μέση Ανάκληση Μεγάλη Περιοχή	0.67	0.70	0.72	0.74	0.76	0.77	0.77	0.78

Πίνακας 7: Μέση ακρίβεια ανά κλάση του μοντέλου Faster-RCNN

Κλάση	**512x512**	**1024x1024**
Μέσος Όρος	0.50	0.56
airplane	0.82	0.84
apple	0.23	0.27
backpack	0.14	0.26
banana	0.34	0.36
baseball bat	0.39	0.52
baseball glove	0.40	0.53
bear	0.87	0.81
bed	0.65	0.63
bench	0.29	0.37
bicycle	0.41	0.55
bird	0.39	0.42
boat	0.35	0.46
book	0.14	0.18
bottle	0.38	0.50
bowl	0.48	0.48
broccoli	0.34	0.29
bus	0.72	0.82
cake	0.42	0.50
car	0.52	0.66
carrot	0.24	0.22
cat	0.90	0.92
cell phone	0.38	0.52
chair	0.33	0.41
clock	0.66	0.72
couch	0.57	0.58
cow	0.46	0.37
cup	0.42	0.52
dining table	0.38	0.39
dog	0.79	0.81
donut	0.41	0.31
elephant	0.82	0.77
fire hydrant	0.77	0.83
fork	0.37	0.54
frisbee	0.73	0.81
giraffe	0.87	0.87
hair drier	0.01	0.14
handbag	0.12	0.24
horse	0.71	0.73
hot dog	0.40	0.47
keyboard	0.65	0.72
kite	0.48	0.53
knife	0.14	0.27
laptop	0.70	0.74
microwave	0.62	0.75
motorcycle	0.62	0.63
mouse	0.67	0.72
orange	0.35	0.39
oven	0.50	0.53
parking meter	0.57	0.61
person	0.69	0.76
pizza	0.63	0.68
potted plant	0.35	0.41
refrigerator	0.68	0.70
remote	0.26	0.48
sandwich	0.49	0.51
scissors	0.44	0.43
sheep	0.64	0.35
sink	0.48	0.55
skateboard	0.70	0.76
skis	0.36	0.50
snowboard	0.44	0.52
spoon	0.16	0.27
sports ball	0.43	0.59
stop sign	0.67	0.76
suitcase	0.38	0.46
surfboard	0.52	0.63

teddy bear	0.63	0.64
tennis racket	0.65	0.78
tie	0.36	0.47
toaster	0.28	0.71
toilet	0.78	0.65
toothbrush	0.23	0.35
traffic light	0.36	0.48
train	0.84	0.86
truck	0.46	0.55
tv	0.69	0.74
umbrella	0.51	0.59
vase	0.39	0.44
wine glass	0.37	0.51

Πίνακας 8: Μέση ακρίβεια ανά κλάση του μοντέλου CenterNet

Κλάση	**512x512**	**1024x1024**
Μέσος Όρος	0.59	0.60
airplane	0.87	0.86
apple	0.27	0.29
backpack	0.31	0.33
banana	0.37	0.35
baseball bat	0.55	0.55
baseball glove	0.59	0.68
bear	0.82	0.86
bed	0.64	0.53
bench	0.34	0.40
bicycle	0.52	0.59
bird	0.50	0.53
boat	0.48	0.51
book	0.19	0.24
bottle	0.54	0.59
bowl	0.53	0.55
broccoli	0.40	0.35
bus	0.79	0.82
cake	0.52	0.51
car	0.66	0.72
carrot	0.35	0.33
cat	0.86	0.81
cell phone	0.54	0.57
chair	0.45	0.48
clock	0.70	0.76
couch	0.58	0.61
cow	0.72	0.77
cup	0.59	0.64
dining table	0.40	0.35
dog	0.76	0.75
donut	0.53	0.58
elephant	0.85	0.86
fire hydrant	0.86	0.87
fork	0.54	0.57
frisbee	0.84	0.88
giraffe	0.88	0.91
hair drier	0.11	0.11
handbag	0.32	0.31
horse	0.80	0.82
hot dog	0.47	0.45
keyboard	0.72	0.69
kite	0.58	0.63
knife	0.31	0.36
laptop	0.74	0.74
microwave	0.70	0.75
motorcycle	0.70	0.71
mouse	0.82	0.83
orange	0.38	0.40
oven	0.56	0.51
parking meter	0.56	0.62
person	0.78	0.79
pizza	0.68	0.68
potted plant	0.45	0.47
refrigerator	0.67	0.68
remote	0.52	0.60
sandwich	0.45	0.44

scissors	0.37	0.34
sheep	0.71	0.75
sink	0.59	0.62
skateboard	0.76	0.82
skis	0.52	0.54
snowboard	0.56	0.56
spoon	0.32	0.33
sports ball	0.65	0.69
stop sign	0.80	0.81
suitcase	0.58	0.59
surfboard	0.62	0.67
teddy bear	0.62	0.62
tennis racket	0.80	0.82
tie	0.55	0.57
toaster	0.40	0.57
toilet	0.82	0.83
toothbrush	0.40	0.32
traffic light	0.56	0.57
train	0.86	0.84
truck	0.52	0.55
tv	0.76	0.76
umbrella	0.60	0.59
vase	0.55	0.57
wine glass	0.59	0.59
zebra	0.90	0.89

Πίνακας 9: Μέση ακρίβεια ανά κλάση του μοντέλου SSD-ResNet

Κλάση	**512x512**	**1024x1024**
Μέσος Όρος	0.52	0.57
airplane	0.83	0.89
apple	0.22	0.26
backpack	0.22	0.32
banana	0.35	0.35
baseball bat	0.40	0.47
baseball glove	0.50	0.57
bear	0.81	0.81
bed	0.67	0.60
bench	0.34	0.37
bicycle	0.49	0.56
bird	0.43	0.49
boat	0.41	0.49
book	0.17	0.21
bottle	0.39	0.51
bowl	0.50	0.51
broccoli	0.39	0.37
bus	0.78	0.81
cake	0.48	0.53
car	0.56	0.66
carrot	0.29	0.28
cat	0.89	0.90
cell phone	0.45	0.54
chair	0.41	0.45
clock	0.66	0.74
couch	0.62	0.61
cow	0.63	0.73
cup	0.47	0.58
dining table	0.43	0.43
dog	0.78	0.80
donut	0.49	0.52
elephant	0.82	0.83
fire hydrant	0.75	0.81
fork	0.38	0.50
frisbee	0.69	0.80
giraffe	0.83	0.86
hair drier	0.00	0.11
handbag	0.21	0.28
horse	0.72	0.78
hot dog	0.41	0.50
keyboard	0.66	0.70
kite	0.47	0.53
knife	0.19	0.28
laptop	0.72	0.76

microwave	0.70	0.72
motorcycle	0.66	0.70
mouse	0.75	0.79
orange	0.36	0.38
oven	0.55	0.54
parking meter	0.57	0.60
person	0.69	0.73
pizza	0.65	0.67
potted plant	0.41	0.47
refrigerator	0.62	0.67
remote	0.35	0.49
sandwich	0.46	0.51
scissors	0.40	0.42
sheep	0.66	0.68
sink	0.50	0.56
skateboard	0.66	0.75
skis	0.36	0.41
snowboard	0.30	0.37
spoon	0.18	0.23
sports ball	0.42	0.59
stop sign	0.68	0.71
suitcase	0.52	0.58
surfboard	0.52	0.57
teddy bear	0.61	0.65
tennis racket	0.69	0.75
tie	0.38	0.50
toaster	0.47	0.43
toilet	0.75	0.79
toothbrush	0.27	0.29
traffic light	0.38	0.51
train	0.85	0.86
truck	0.52	0.58
tv	0.72	0.77
umbrella	0.53	0.60
vase	0.43	0.50
wine glass	0.42	0.53
zebra	0.87	0.88

Πίνακας 10: Μέση ακρίβεια ανά κλάση του μοντέλου YoloV4

Κλάση	**416x416**	**512x512**	**608x608**
Μέσος Όρος	0.72	0.73	0.74
airplane	0.94	0.94	0.93
apple	0.44	0.45	0.47
backpack	0.45	0.48	0.47
banana	0.57	0.54	0.54
baseball bat	0.72	0.76	0.78
baseball glove	0.76	0.77	0.79
bear	0.95	0.96	0.94
bed	0.75	0.76	0.75
bench	0.52	0.53	0.54
bicycle	0.64	0.68	0.69
bird	0.57	0.60	0.61
boat	0.57	0.64	0.65
book	0.34	0.36	0.38
bottle	0.64	0.68	0.69
bowl	0.72	0.72	0.73
broccoli	0.52	0.54	0.48
bus	0.91	0.91	0.91
cake	0.77	0.76	0.78
car	0.71	0.74	0.76
carrot	0.46	0.44	0.45
cat	0.95	0.95	0.95
cell phone	0.68	0.70	0.72
chair	0.64	0.63	0.63
clock	0.83	0.82	0.83
couch	0.74	0.73	0.72
cow	0.86	0.86	0.86
cup	0.73	0.76	0.76
dining table	0.53	0.53	0.53
dog	0.91	0.91	0.91
donut	0.77	0.79	0.79
elephant	0.91	0.92	0.91

fire hydrant	0.91	0.91	0.94
fork	0.66	0.65	0.66
frisbee	0.91	0.93	0.94
giraffe	0.92	0.92	0.92
hair drier	0.38	0.32	0.33
handbag	0.42	0.44	0.46
horse	0.91	0.92	0.93
hot dog	0.75	0.77	0.76
keyboard	0.80	0.80	0.80
kite	0.72	0.73	0.75
knife	0.45	0.52	0.55
laptop	0.88	0.89	0.90
microwave	0.90	0.91	0.91
motorcycle	0.76	0.79	0.80
mouse	0.84	0.88	0.89
orange	0.56	0.53	0.57
oven	0.72	0.67	0.68
parking meter	0.83	0.84	0.88
person	0.80	0.82	0.83
pizza	0.81	0.81	0.83
potted plant	0.63	0.65	0.66
refrigerator	0.83	0.86	0.86
remote	0.66	0.71	0.73
sandwich	0.73	0.75	0.74
scissors	0.60	0.54	0.61
sheep	0.86	0.85	0.87
sink	0.75	0.77	0.76
skateboard	0.83	0.86	0.87
skis	0.54	0.58	0.59
snowboard	0.68	0.68	0.70
spoon	0.44	0.48	0.51
sports ball	0.67	0.67	0.72
stop sign	0.81	0.81	0.85
suitcase	0.79	0.81	0.80
surfboard	0.73	0.75	0.79
teddy bear	0.77	0.79	0.80
tennis racket	0.88	0.89	0.90
tie	0.66	0.63	0.64
toaster	0.67	0.60	0.63
toilet	0.90	0.90	0.91
toothbrush	0.56	0.63	0.58
traffic light	0.58	0.61	0.62
train	0.93	0.95	0.95
truck	0.69	0.71	0.69
tv	0.88	0.88	0.88
umbrella	0.72	0.75	0.75
vase	0.69	0.70	0.71
wine glass	0.68	0.69	0.74
zebra	0.93	0.94	0.94

Πίνακας 11: Μέση ακρίβεια ανά κλάση του μοντέλου EfficientDet

Κλάση	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
Μέσος Όρος	0.51	0.56	0.60	0.64	0.67	0.68	0.69	0.70
airplane	0.82	0.87	0.89	0.92	0.92	0.91	0.92	0.94
apple	0.22	0.23	0.29	0.30	0.33	0.37	0.37	0.35
backpack	0.18	0.23	0.31	0.33	0.39	0.44	0.45	0.44
banana	0.36	0.38	0.41	0.44	0.43	0.47	0.44	0.44
baseball bat	0.47	0.50	0.55	0.60	0.64	0.64	0.69	0.65
baseball glove	0.48	0.54	0.60	0.65	0.70	0.71	0.75	0.74
bear	0.86	0.90	0.92	0.94	0.97	0.92	0.95	0.92
bed	0.67	0.68	0.71	0.71	0.75	0.71	0.72	0.74
bench	0.32	0.36	0.39	0.43	0.44	0.47	0.49	0.49
bicycle	0.44	0.54	0.57	0.63	0.64	0.70	0.71	0.67
bird	0.40	0.45	0.50	0.55	0.55	0.58	0.57	0.61
boat	0.36	0.44	0.48	0.53	0.57	0.57	0.59	0.60
book	0.14	0.18	0.22	0.27	0.30	0.31	0.33	0.33
bottle	0.35	0.45	0.52	0.55	0.61	0.64	0.64	0.66
bowl	0.44	0.52	0.56	0.60	0.65	0.62	0.64	0.65
broccoli	0.43	0.41	0.45	0.43	0.46	0.46	0.43	0.43
bus	0.74	0.80	0.82	0.83	0.87	0.86	0.88	0.88
cake	0.47	0.56	0.58	0.60	0.66	0.68	0.68	0.67
car	0.46	0.56	0.62	0.68	0.72	0.75	0.76	0.76

carrot	0.29	0.36	0.40	0.40	0.43	0.42	0.42	0.44
cat	0.85	0.91	0.93	0.94	0.94	0.95	0.93	0.94
cell phone	0.40	0.46	0.53	0.58	0.63	0.66	0.66	0.68
chair	0.39	0.46	0.50	0.53	0.58	0.58	0.60	0.61
clock	0.59	0.68	0.70	0.75	0.78	0.79	0.79	0.79
couch	0.65	0.66	0.66	0.69	0.70	0.67	0.69	0.71
cow	0.64	0.69	0.75	0.79	0.80	0.84	0.84	0.83
cup	0.41	0.50	0.56	0.61	0.67	0.69	0.69	0.71
dining table	0.42	0.44	0.46	0.48	0.50	0.49	0.48	0.51
dog	0.80	0.81	0.86	0.86	0.88	0.88	0.88	0.89
donut	0.46	0.56	0.60	0.61	0.65	0.67	0.68	0.69
elephant	0.83	0.87	0.88	0.88	0.91	0.90	0.90	0.92
fire hydrant	0.74	0.78	0.81	0.84	0.86	0.90	0.90	0.91
fork	0.35	0.41	0.48	0.56	0.58	0.62	0.64	0.66
frisbee	0.68	0.81	0.85	0.90	0.91	0.91	0.91	0.92
giraffe	0.88	0.89	0.91	0.93	0.92	0.92	0.95	0.93
hair drier	0.01	0.12	0.10	0.40	0.48	0.28	0.34	0.36
handbag	0.15	0.22	0.29	0.35	0.38	0.44	0.46	0.46
horse	0.73	0.80	0.81	0.86	0.89	0.90	0.90	0.90
hot dog	0.45	0.48	0.51	0.52	0.57	0.61	0.60	0.58
keyboard	0.59	0.65	0.65	0.72	0.72	0.73	0.75	0.72
kite	0.44	0.54	0.57	0.62	0.65	0.70	0.68	0.67
knife	0.15	0.21	0.25	0.29	0.37	0.39	0.43	0.43
laptop	0.71	0.77	0.79	0.80	0.85	0.86	0.86	0.87
microwave	0.71	0.80	0.81	0.81	0.85	0.86	0.87	0.86
motorcycle	0.65	0.70	0.76	0.77	0.80	0.80	0.79	0.80
mouse	0.64	0.75	0.79	0.83	0.85	0.87	0.86	0.86
orange	0.36	0.39	0.40	0.41	0.44	0.46	0.48	0.49
oven	0.58	0.58	0.60	0.65	0.66	0.63	0.66	0.68
parking meter	0.60	0.58	0.63	0.66	0.67	0.70	0.72	0.69
person	0.67	0.73	0.76	0.79	0.81	0.83	0.84	0.84
pizza	0.66	0.71	0.72	0.74	0.77	0.78	0.77	0.77
potted plant	0.40	0.46	0.49	0.52	0.57	0.58	0.59	0.59
refrigerator	0.66	0.71	0.75	0.76	0.83	0.81	0.80	0.83
remote	0.27	0.41	0.49	0.57	0.65	0.69	0.71	0.72
sandwich	0.48	0.52	0.52	0.55	0.61	0.60	0.63	0.61
scissors	0.46	0.47	0.51	0.49	0.52	0.58	0.53	0.61
sheep	0.67	0.73	0.75	0.79	0.81	0.82	0.82	0.83
sink	0.51	0.54	0.58	0.59	0.65	0.66	0.64	0.67
skateboard	0.69	0.75	0.77	0.79	0.82	0.84	0.87	0.86
skis	0.32	0.35	0.37	0.48	0.52	0.54	0.54	0.52
snowboard	0.37	0.37	0.48	0.50	0.58	0.61	0.60	0.59
spoon	0.15	0.18	0.25	0.29	0.36	0.40	0.39	0.44
sports ball	0.32	0.47	0.53	0.59	0.67	0.68	0.73	0.71
stop sign	0.60	0.70	0.70	0.71	0.77	0.80	0.81	0.80
suitcase	0.41	0.59	0.63	0.68	0.73	0.72	0.75	0.75
surfboard	0.48	0.56	0.60	0.64	0.66	0.66	0.71	0.70
teddy bear	0.62	0.67	0.70	0.69	0.74	0.75	0.75	0.75
tennis racket	0.69	0.76	0.78	0.84	0.89	0.91	0.91	0.90
tie	0.33	0.39	0.45	0.49	0.56	0.57	0.60	0.62
toaster	0.44	0.49	0.51	0.68	0.67	0.54	0.58	0.54
toilet	0.81	0.86	0.87	0.87	0.88	0.90	0.90	0.89
toothbrush	0.24	0.30	0.39	0.42	0.45	0.47	0.48	0.54
traffic light	0.31	0.40	0.46	0.54	0.56	0.60	0.62	0.63
train	0.85	0.85	0.88	0.88	0.92	0.92	0.91	0.93
truck	0.49	0.53	0.56	0.60	0.64	0.66	0.65	0.67
tv	0.74	0.76	0.79	0.81	0.83	0.84	0.85	0.85
umbrella	0.53	0.58	0.62	0.67	0.69	0.71	0.71	0.70
vase	0.42	0.48	0.53	0.55	0.58	0.61	0.64	0.65
wine glass	0.37	0.44	0.52	0.60	0.64	0.66	0.70	0.71
zebra	0.86	0.91	0.91	0.91	0.92	0.93	0.94	0.93

Βιβλιογραφία

[1] Jonathan T. Barron. A more general robust loss function. *CoRR*, abs/1701.03077, 2017.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[3] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server, 2015.

[4] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. *CoRR*, abs/1904.08189, 2019.

[5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[6] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang, and Quoc V. Le. NAS-FPN: learning scalable feature pyramid architecture for object detection. *CoRR*, abs/1904.07392, 2019.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[8] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[13] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *CoRR*, abs/1808.01244, 2018.

[14] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[15] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *CoRR*, abs/1803.01534, 2018.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[17] Xin Lu, Xin Kang, Shun Nishide, and Fuji Ren. Object detection based on ssd-resnet. In *2019 IEEE 6th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 89–92, 2019.

[18] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *CoRR*, abs/1512.02017, 2015.

[19] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation, 2016.

[20] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[22] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[26] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile, 2019.

[27] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[28] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientnet: Scalable and efficient object detection. *CoRR*, abs/1911.09070, 2019.

[29] Jasper Uijlings, K. Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[30] Abdul Vahab, Maruti S Naik, Prasanna G Raikar, and SR Prasad. Applications of object detection system. *International Research Journal of Engineering and Technology (IRJET)*, 6(4):4186–4192, 2019.

[31] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. Cspnet: A new backbone that can enhance learning capability of CNN. *CoRR*, abs/1911.11929, 2019.

[32] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. *CoRR*, abs/1807.06521, 2018.

[33] Zhuliang Yao, Yue Cao, Shuxin Zheng, Gao Huang, and Stephen Lin. Cross-iteration batch normalization. *CoRR*, abs/2002.05712, 2020.