Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Ψηφιακός Πολιτισμός, Έξυπνες Πόλεις, IoT και Προηγμένες Ψηφιακές Τεχνολογίες»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Βαθιά Μάθηση για ανίχνευση αντικειμένων σε πραγματικό χρόνο με Raspberry Pi** **Deep Learning for Real-Time Object Detection on Raspberry Pi**
Ονοματεπώνυμο Φοιτητή	**Ιωάννα Οικονομίδου**
Πατρώνυμο	**Βασίλειος**
Αριθμός Μητρώου	**ΨΠΟΛ/19043**
Επιβλέπων	**Μιχαήλ Φιλιππάκης**, **Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Ιούλιος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Μιχαήλ Φιλιππάκης	Δημήτριος Βέργαδος	Άγγελος Μιχάλας
Αναπληρωτής Καθηγητής	Καθηγητής	Καθηγητής

Στην Οικογένειά μου

και στη Λίζι,

Ευχαριστιες

Πρώτα από όλα θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Φιλιππάκη Μιχαήλ, για τη δυνατότητα που μου έδωσε να πραγματοποιήσω την παρούσα διπλωματική και για την καθοδήγηση και τις συμβουλές που μου παρείχε καθ' όλη τη διάρκεια διεκπεραίωσης της.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου για την ανυπολόγιστη υποστήριξη που μου παρείχε και το Βασίλη Χαλδεάκη για όλη την βοήθεια και την έμπνευση σε όλα τα χρόνια των σπουδών μου.

Περιληψη

Η ανίχνευση αντικειμένων χρησιμοποιώντας πραγματικά δεδομένα είναι μια πολύ δημοφιλής εφαρμογή στην Τεχνητή Νοημοσύνη. Με την εξέλιξη της βαθιάς μάθησης, παρατηρείται συνεχής αύξηση της απόδοσης τεχνικών ανίχνευσης αντικειμένων με την βοήθεια των νευρωνικών δικτύων τόσο ως προς την ταχύτητα όσο και ως προς την ακρίβεια. Το γεγονός αυτό καθιστά εφικτή την προσπάθεια υλοποίησης αλγορίθμων εντοπισμού αντικειμένων μέσω κάμερας ακόμα και σε πραγματικό χρόνο.

Ιδιαίτερο ενδιαφέρον φαίνεται να υπάρχει στην ανάπτυξη αρχιτεκτονικών νευρωνικών δικτύων κατάλληλων για εφαρμογή σε ενσωματωμένες συσκευές. Στην παρούσα διπλωματική θα μελετηθεί ο αλγόριθμος YOLO για τον εντοπισμό αντικειμένων σε καταγεγραμμένο υλικό εικόνων και βίντεο αλλά και πραγματικό χρόνο τόσο σε υπολογιστικές συσκευές όσο και σε ενσωματωμένες, όπως το Raspberry pi. Θα πραγματοποιηθούν πειράματα με διαφορετικές εκδόσεις και αρχιτεκτονικές του YOLO για την μελέτη της ταχύτητας και της απόδοσης αυτών.

Abstract

Object detection using real data is a very popular task in Artificial Intelligence. The recent advancement of deep learning has led to greatly increase of the performance of object detection techniques in both speed and accuracy. That fact makes it possible to run object detection algorithms with high accuracy even with real time speed on desktop computer systems.

Of growing interest seems to be the development of neural networks' architectures suitable for embedded devices. In this thesis, the YOLO algorithm will be studied for object detection in recorded images and videos, but also in real-time in both computing and embedded devices, such as Raspberry Pi. Experiments will be performed with different versions and YOLO architectures to compare their speed and performance.

Πινακας περιεχομενων

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

1. Εισαγωγη

Στις σημερινές κοινωνίες είναι γεγονός πως η τεχνολογία καθιστά όλο και πιο εύκολη την ζωή των ανθρώπων παρέχοντας τους δυνατότητες που πριν από μερικά χρόνια φάνταζαν επιστημονική φαντασία. Για τον εντοπισμό αντικειμένων όπως οχήματα, ζώα, άνθρωποι, καθημερινά αντικείμενα έχουν προταθεί και χρησιμοποιούνται αλγόριθμοι βασισμένοι στην Τεχνητή Νοημοσύνη. Χρησιμοποιώντας βασικές αρχές της Μηχανικής Μάθησης και των Νευρωνικών Δικτύων, έχουν δημιουργηθεί σημαντικές βάσεις για την ανίχνευση και χωροθέτηση αντικειμένων.

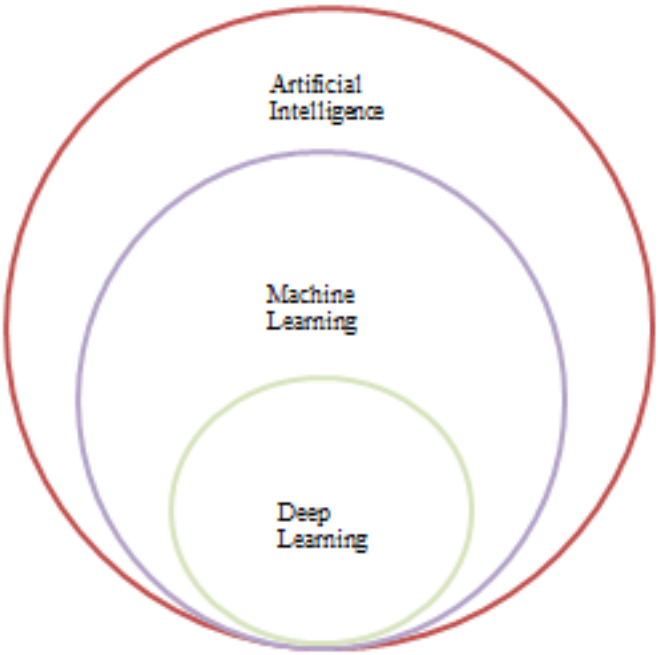
Ο σκοπός της παρούσας εργασίας είναι η μελέτη και παρουσίαση βασικών εννοιών μηχανικής μάθησης, η ανάλυση αρχιτεκτονικής των συνελικτικών νευρωνικών δικτύων ως βασικό εργαλείο για την αναγνώριση αντικειμένων και η υλοποίηση του αλγορίθμου YOLO στις 2 τελευταίες εκδόσεις του version 3 και version 4 καθώς και η σύγκριση τους. Τέλος, υλοποιούνται οι αλγόριθμοι yolo-tiny στις 2 αυτές εκδόσεις με εφαρμογή σε ανίχνευση αντικειμένων σε πραγματικό χρόνο με την χρήση συσκευής raspberry pi. Σε όλη την εργασία χρησιμοποιείται η γλώσσα προγραμματισμού Python, με βιβλιοθήκες που παρουσιάζονται στην Ενότητα 5. Τέλος, τα σύνολα δεδομένων που χρησιμοποιήθηκαν στην Ενότητα 3 για τα παραδείγματα των αλγορίθμων Μηχανικής Μάθησης είναι από τον ιστότοπο Kaggle.

Η εργασία ακολουθεί την εξής δομή. Στην Ενότητα 2 παρουσιάζονται βασικές θεωρητικές έννοιες. Στην Ενότητα 3 αναλύονται οι κατηγορίες μάθησης και παρουσιάζονται αναλυτικά , με παραδείγματα και διαγράμματα, οι πιο σημαντικοί αλγόριθμοι Μηχανικής Μάθησης ανά κατηγορία. Στην Ενότητα 4, μελετώνται θεωρητικά τα τεχνητά νευρωνικά δίκτυα και τα συνελικτικά. Τέλος, στην Ενότητα 5, περιγράφεται αναλυτικά ο αλγόριθμος YOLO στις δύο τελευταίες του εκδόσεις και γίνεται η υλοποίηση του αλγορίθμου για την ανίχνευση αντικειμένων σε ιστορικό και πραγματικό χρόνο με την ενσωματωμένη συσκευή Raspberry Pi.

2. Μηχανικη Μαθηση

Τεχνητή νοημοσύνη (Artificial Intelligence) είναι επιστήμη που δίνει τη δυνατότητα στους υπολογιστές να μιμούνται την ανθρώπινη νοημοσύνη και συμπεριφορές όπως είναι η λήψη αποφάσεων, η διαδικασία επεξεργασίας κειμένου, η οπτική αντίληψη. Αποτελεί μια έννοια που συνδέεται με πολλές επιστήμες πληροφορική, νευρολογία, ψυχολογία, γλωσσολογία και άλλες. Η τεχνητή νοημοσύνη είναι ένα ευρύτερο πεδίο που εμπεριέχει υποκατηγορίες όπως η Μηχανική μάθηση, ρομποτική και υπολογιστική όραση.

Η Μηχανική Μάθηση (Machine Learning) είναι υποπεδίο της Τεχνητής Νοημοσύνης που δίνει την ικανότητα στους υπολογιστές να αποκτούν εμπειρία και να βελτιώνουν ένα δοσμένο έργο. Είναι μια μέθοδος ανάλυσης δεδομένων που αυτοματοποιεί τη δημιουργία αναλυτικών μοντέλων εκπαιδεύοντας μια μηχανή πως να μαθαίνει. Οι υπολογιστικές μηχανές μπορούν να μάθουν χωρίς να προγραμματιστούν ρητά. Ουσιαστικά, το έργο της μηχανικής εκμάθησης είναι να αναλύονται μεγάλα δεδομένα και να εξάγονται αυτόματα πληροφορίες. Με αυτές τις πληροφορίες η μηχανή μπορεί να κάνει προβλέψεις, να αποκρυπτογραφήσει εάν η πρόβλεψη ήταν σωστή και, αν είναι λανθασμένη, να μάθει από αυτήν για να κάνει στο μέλλον μια πιο σωστή πρόβλεψη. Η Μηχανική Μάθηση είναι ένας τομέας που μελετά και κατασκευάζει αλγόριθμους που μπορούν να μάθουν και στο τέλος να προβλέψουν διάφορα αποτελέσματα πάνω σε ένα σύνολο δεδομένων . Οι αλγόριθμοι αυτοί, για την λήψη αποφάσεων και την πρόβλεψη, βασίζονται αποκλειστικά στα σύνολα δεδομένων με τη δημιουργία κατάλληλων μοντέλων από δειγματικά δεδομένα. Ένα ακόμα υποσύνολο της μηχανικής μάθησης είναι η βαθιά μάθηση (Deep Learning), το οποίο θα μελετήσουμε αναλυτικά στο Κεφάλαιο 4. Σχηματικά η σχέση των διαφορετικών αυτών πεδίων αναπαρίσταται στην Εικόνα 1 .



Εικόνα 1: Πεδία Τεχνητής Νοημοσύνης

3. Κατηγοριες Μαθησης

Οι αλγόριθμοι της Μηχανικής Μάθησης χωρίζονται ανάλογα με το είδος του αποτελέσματος, αν είναι γνωστό εκ των προτέρων ή όχι, στις κατηγορίες:

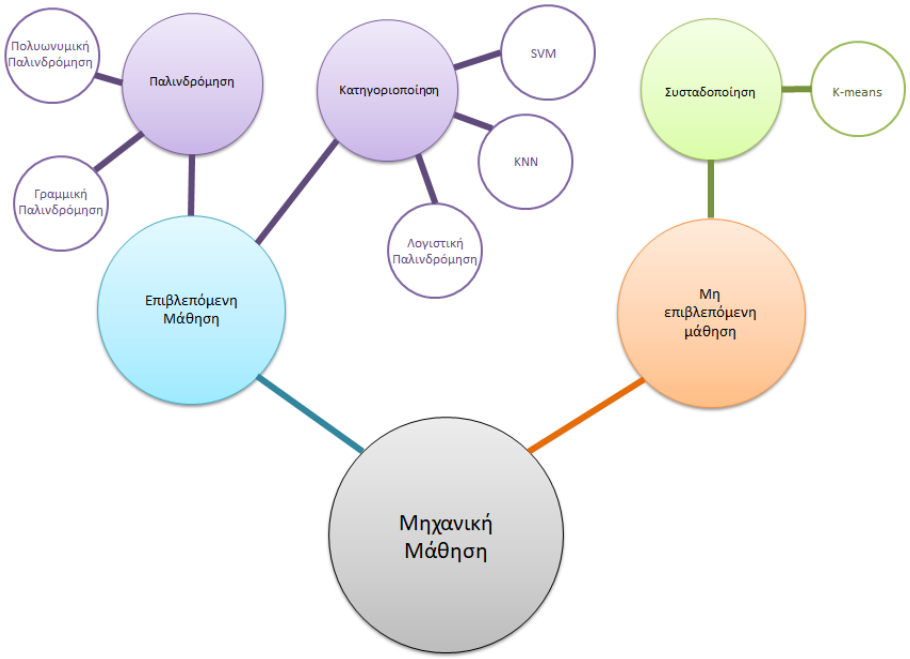
- Επιβλεπόμενη μάθηση (Supervised Learning)

Το σύστημα καλείται να μάθει ένα σύνολο από δεδομένα, που αποτελεί την περιγραφή του μοντέλου. Ο αλγόριθμος εκμάθησης λαμβάνει ένα σύνολο εισόδων μαζί με τις αντίστοιχες σωστές εξόδους και μαθαίνει συγκρίνοντας την πραγματική του έξοδο με τις σωστές εξόδους για να βρει λάθη, ενώ στη συνέχεια τροποποιεί ανάλογα το μοντέλο. Η επιβλεπόμενη μάθηση χρησιμοποιείται συνήθως σε εφαρμογές όπου επιθυμούμε από τα ήδη υπάρχοντα δεδομένα να προβλέψουμε τα πιθανά μελλοντικά αποτελέσματα. Κύριες περιοχές της επιβλεπόμενης μάθησης είναι η παλινδρόμηση και η κατηγοριοποίηση.

- Μη επιβλεπόμενη μάθηση

Το σύστημα πρέπει μόνο του να ανακαλύψει συσχετίσεις σε ένα σύνολο δεδομένων και να δημιουργήσει εκ νέου πρότυπα. Η βασικότερη περιοχή της μη επιβλεπόμενης μάθησης είναι η συσταδοποίηση, δηλαδή την ομαδοποίηση ενός συνόλου δεδομένων με σκοπό να ανήκουν στην ίδια συστάδα. Παρακάτω θα μελετήσουμε κάποιους από τους σημαντικότερους αλγόριθμους τόσο στην επιβλεπόμενη όσο και στην μη επιβλεπόμενη μάθηση.

Στη συνέχεια, θα μελετήσουμε κάποιους από τους πιο γνωστούς αλγόριθμους και των δύο κατηγοριών τόσο σε θεωρητικό επίπεδο, εξηγώντας ποια είναι τα βήματα που ακολουθούν, όσο και σε πρακτικό με την υλοποίησή τους και την παρουσίαση των αποτελεσμάτων.



Εικόνα 2: Κατηγορίες Μηχανικής Μάθησης

3.1 Τεχνικες Παλινδρομησης

Σε πολλά προβλήματα με τυχαίες μεταβλητές χρειάζεται να προσδιοριστεί ο τρόπος με τον οποίο συσχετίζονται αυτές. Τέτοιες μεταβλητές μπορεί να είναι για παράδειγμα:

- Η διάρκεια ζωής ενός οργανισμού σε μια περιοχή και το ποσοστό μόλυνσης στην περιοχή αυτή.
- Το ύψος των αποδοχών των υπαλλήλων μια εταιρείας και τα χρόνια υπηρεσίας τους.
- Οι δαπάνες κατανάλωσης σε ένα νοικοκυριό και το διαθέσιμο εισόδημα της οικογένειας.

Η ανάπτυξη ενός μαθηματικού μοντέλου αποτελεί μια στατιστική διαδικασία που συμβάλλει στην παραγωγή εξισώσεων που περιγράφουν τη σχέση μεταξύ των ανεξάρτητων μεταβλητών και της εξαρτημένης. Τα μοντέλα παλινδρόμησης χρησιμοποιούνται για την πρόβλεψη μιας συνεχούς πραγματικής τιμής.

3.1.1 Απλη Γραμμικη Παλινδρομηση

Η απλή γραμμική παλινδρόμηση είναι η πιο απλή μορφή παλινδρόμησης. Υπάρχει μόνο μία ανεξάρτητη μεταβλητή x και μία εξαρτημένη μεταβλητή y, που προσεγγίζεται ως μια γραμμική συνάρτηση του x. Η τιμή y_i της y, για κάθε τιμή x_i της x, δίνεται από τη σχέση :

$$y_i = b_0 + b_1 * x_i + e_i$$

Το b_1 είναι ο συντελεστής της ανεξάρτητης μεταβλητής και b_0 η σταθερά. Το πρόβλημα της γραμμικής παλινδρόμησης είναι η εύρεση των παραμέτρων b_0 και b_1 που εκφράζουν καλύτερα τη γραμμική εξάρτηση της y από τη x. Για κάθε τιμή b_0 και b_1 ορίζεται μια διαφορετική γραμμική σχέση όπου γεωμετρικά εκφράζεται από μια ευθεία με τις εξής παραμέτρους:

- Η σταθερά b_0 είναι η τιμή του y για x=0
- Ο συντελεστής b_1 του x είναι η κλίση (slope) της ευθείας ή αλλιώς ο συντελεστής παλινδρόμησης (regression coefficient). Εκφράζει τη μεταβολή της μεταβλητής y όταν η μεταβλητή x μεταβληθεί κατά μία μονάδα.

Ο όρος e_i ονομάζεται σφάλμα παλινδρόμησης (regression error) ή απόκλιση. Στην πράξη είναι η διαφορά της παρατηρούμενης τιμής y_i δεδομένης της τιμής x_i από την τιμή της πρόβλεψης που προκύπτει από το μοντέλο.

Για την εύρεση της ευθείας παλινδρόμησης που ταιριάζει καλύτερα στα δεδομένα μας θα χρησιμοποιούμε μια από τις κυριότερες μεθόδους υπολογισμού εκτιμητή ευθείας, την μέθοδο ελαχίστων τετραγώνων. Ουσιαστικά πρέπει να εκτιμήσουμε τους παραμέτρους του της παλινδρόμησης, b_0 και b_1. Ο αριθμός των ζευγών αυτών είναι άπειρος και αναζητούμε την ευθεία που περιγράφει με τον καλύτερο δυνατό τρόπο τη σχέση μεταξύ των δύο μεταβλητών. Η γραμμή παλινδρόμησης πρέπει να περνάει κοντά από τα σημεία που αντιστοιχούν τα ζεύγη παρατηρήσεων (x,y) έτσι ώστε να ελαχιστοποιούνται τα σφάλματα πρόβλεψης.

Παράδειγμα

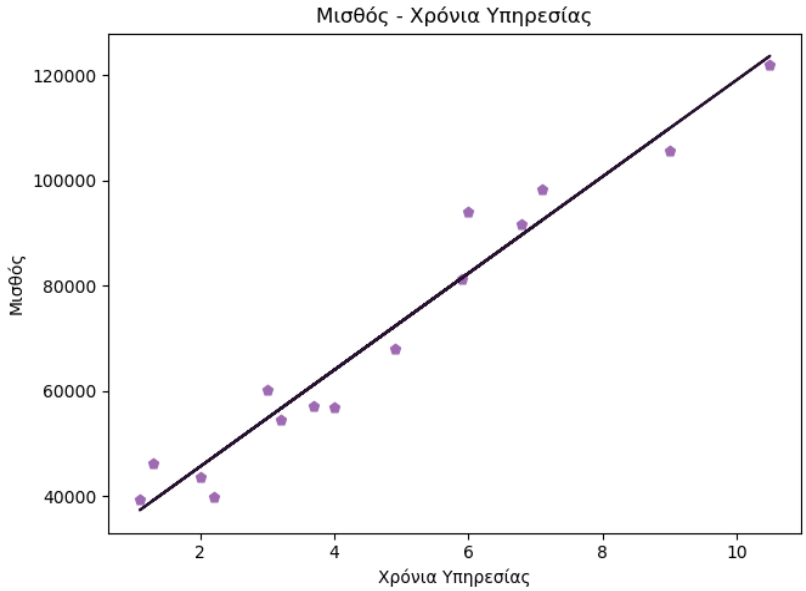
Έστω πως ο άξονας x αναπαριστά τον μισθό και ο y τα χρόνια υπηρεσίας. Θέλουμε να μελετήσουμε πώς ο μισθός κάποιου επηρεάζεται από την εμπειρία του. Στο παρακάτω σχήμα φαίνονται κάποιες παρατηρήσεις για το πως οι μισθοί είναι κατανεμημένοι μεταξύ των ανθρώπων που δουλεύουν σε κάποια εταιρεία με διαφορετικά επίπεδα εμπειρίας. Η εξίσωση της απλής γραμμικής παλινδρόμησης θα είναι:

μισθός = b_0 + b_1 * χρόνια υπηρεσίας

Με δεδομένο το σύνολο δεδομένων που βλέπουμε στην εικόνα, η ευθεία της παλινδρόμησης θα είναι η εξής:

Χρόνια Υπηρεσίας	Μισθός
6.8	91738
1.3	46205
10.5	121872
3	60150
2.2	39891
5.9	81363
6	93940
3.7	57189
3.2	54445
9	105582
2	43525
1.1	39343
7.1	98273
4.9	67938
4	56957

Εικόνα 3: Σύνολο Δεδομένων Απλής Γραμμικής Παλινδρόμησης



Εικόνα 4: Ευθεία Απλής Γραμμικής Παλινδρόμησης

3.1.2 Πολλαπλη Γραμμικη Παλινδρομηση

Σε πολλά πρακτικά προβλήματα χρειάζονται δύο ή περισσότερες ανεξάρτητες μεταβλητές προκειμένου να υπάρχει μεγαλύτερη ακρίβεια και να βγουν σωστότερα συμπεράσματα. Η πολλαπλή παλινδρόμηση έχει εφαρμογών σε πολλά διαφορετικά πεδία.

- Εκτίμηση της δράσης των χημικών συστατικών ενός τροφίμου στις οργανοληπτικές ιδιότητές του.
- Εκτίμηση του βαθμού επίδοσης του προσωπικού μιας εταιρείας.

- Διαχείριση του τύπου οδοστρώματος και είδους μεταφορικού μέσου στο χρόνο εκπλήρωσης μιας μετακίνησης.
- Ατμοσφαιρική και υδρόβια ρύπανση με προεκτάσεις στη διαφύλαξη της δημόσιας υγείας.

Το γραμμικό μοντέλο πολλαπλής παλινδρόμησης με n ανεξάρτητες μεταβλητές, είναι της μορφής:

$$y_n = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n + e_i$$

Η πολλαπλή παλινδρόμηση έχει κάποιες προϋποθέσεις. Αυτές είναι:

- Γραμμικότητα: Κάθε μια από τις εξαρτημένες μεταβλητές να συσχετίζεται γραμμικά με την εξαρτημένη μεταβλητή. Τα διαγράμματα (scatterplot) μπορούν να δείξουν εάν υπάρχει γραμμική ή καμπυλική σχέση.
- Η διακύμανση των όρων σφάλματος είναι παρόμοια στις τιμές των ανεξάρτητων μεταβλητών. Ένα διάγραμμα τυποποιημένων υπολειμμάτων έναντι των προβλεπόμενων τιμών μπορεί να δείξει εάν τα σημεία κατανέμονται εξίσου σε όλες τις τιμές των ανεξάρτητων μεταβλητών.
- Τα σφάλματα (οι διαφορές μεταξύ της παρατηρούμενης τιμής της εξαρτημένης μεταβλητής y και της προβλεπόμενης τιμής $\hat{y}$) ακολουθούν κανονική κατανομή.
- Ανεξαρτησία σφαλμάτων.
- Οι ανεξάρτητες μεταβλητές δεν συσχετίζονται ιδιαίτερα μεταξύ τους.

Για παράδειγμα, προκειμένου να δημιουργηθεί ένα μοντέλο που θα μπορεί να προβλέπει τη μεγιστοποίηση του κέρδους μιας Startup εταιρίας για μια οικονομική χρονιά με σκοπό την επένδυση ενός φορέα σε αυτή και να δίνει πληροφορίες για την συσχέτιση κάθε μεταβλητής με το επιθυμητό αποτέλεσμα, θα χρησιμοποιηθεί ένα μοντέλο πολλαπλής παλινδρόμησης. Η εξαρτημένη μεταβλητή θα είναι το ετήσιο κέρδος και οι ανεξάρτητες τα χρήματα δαπάνησε στην έρευνα και ανάπτυξη, τα χρήματα δαπάνησε στην διοίκηση, τα χρήματα δαπάνησε στο μάρκετινγκ και η πολιτεία που βρίσκεται.

3.2 Κατηγοριοποιηση

3.2.1 Λογιστικη Παλινδρομηση

Η Λογιστική Παλινδρόμηση αποτελεί μια τεχνική που έχει ως στόχο την ανάλυση των δεδομένων σχετικά με την πρόβλεψη των τιμών κάποιας κατηγορικής εξαρτημένης μεταβλητής μέσω κάποιων ποσοτικών και ποιοτικών ανεξάρτητων μεταβλητών. Χρησιμοποιούμε την λογιστική παλινδρόμηση όταν χρειάζεται να προβλέψουμε την ύπαρξη ή την απουσία ενός χαρακτηριστικού ή ενός συμβάντος.

Η πρόβλεψη για το αν κάτι πρόκειται να συμβεί ή όχι είναι ένα αρκετά συχνό πρόβλημα. Στην ουσία αυτό που επιδιώκουμε είναι να υπολογίσουμε μια πιθανότητα. Την πιθανότητα με την οποία η εξαρτημένη μεταβλητή θα λάβει κάποια συγκεκριμένη τιμή. Εξ ορισμού, η τιμή της πιθανότητας παίρνει τιμές μεταξύ του 0 και του 1 ενώ με τη χρήση της Γραμμικής Παλινδρόμησης μπορεί να προκύψουν τιμές μεγαλύτερες του 1 ή και μικρότερες του 0. Επομένως, τα μοντέλα της γραμμικής παλινδρόμησης που έχουμε εξετάσει μέχρι τώρα είναι ακατάλληλα για την εκτίμηση της τιμής της εξαρτημένης μεταβλητής από τις ανεξάρτητες (Εικόνα 6).

Η εξίσωση της λογιστικής παλινδρόμησης ακολουθεί την εξίσωση της γραμμικής, δηλαδή:

$$y = b_0 + b_1 * x$$

αν πάνω σε αυτή την εξίσωση εφαρμόσουμε τη σιγμοειδή συνάρτηση

$\sigma = \frac{1}{1+e^{-y}}$, λύνοντας ως προς y προκύπτει η σχέση:

$$\ln\left(\frac{\sigma}{1-\sigma}\right) = b_0 + b_1 * x$$

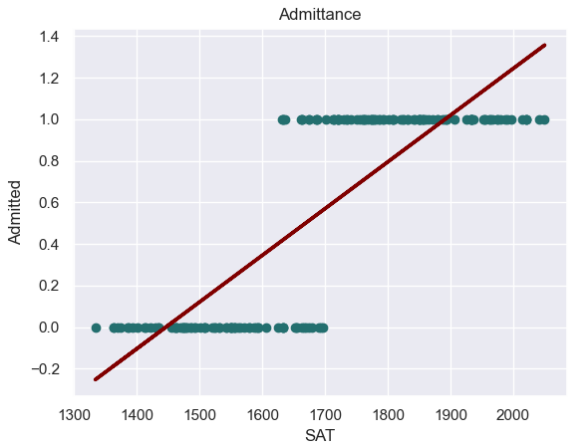
έτσι η γραφική παράσταση της λογιστικής παλινδρόμησης παίρνει τη μορφή που φαίνεται και στο σχήμα (Εικόνα 7).

Παράδειγμα

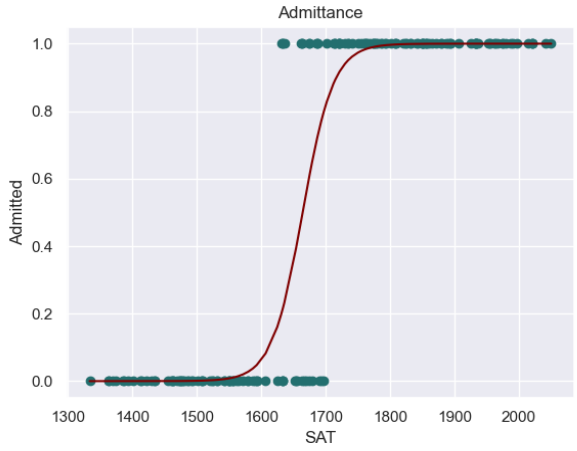
Έστω ότι θέλουμε να εξετάσουμε την πιθανότητα που ένας μαθητής θα εισαχθεί στο πανεπιστήμιο. Το σύνολο δεδομένων δείχνει τον βαθμό SAT, τον βαθμό που λαμβάνει ο μαθητής στις εξετάσεις εισαγωγής και αν με βάση αυτόν τον βαθμό έγινε αποδεκτός ή όχι. Στην εικόνα βλέπουμε την αναπαράσταση των δεδομένων μας. Φαίνεται γιατί η εφαρμογή της γραμμικής παλινδρόμησης δεν είναι η σωστή προσέγγιση για αυτά τα δεδομένα. Όπως είπαμε παραπάνω, ψάχνουμε να βρούμε πιθανότητες. Οι πιθανότητες δεν μπορούν να ξεφύγουν από τα όρια 0 και 1.

SAT	Admitted
1363	No
1792	Yes
1954	Yes
1653	No
1593	No
1755	Yes
1775	Yes
1887	Yes
1893	Yes
1580	No
1857	Yes
1880	Yes
1664	Yes
1364	No
1693	No
1850	Yes

Εικόνα 5: Σύνολο Δεδομένων Λογιστικής Παλινδρόμησης



Εικόνα 6: Αποτελέσματα με εφαρμογή ευθείας Γραμμικής Παλινδρόμησης



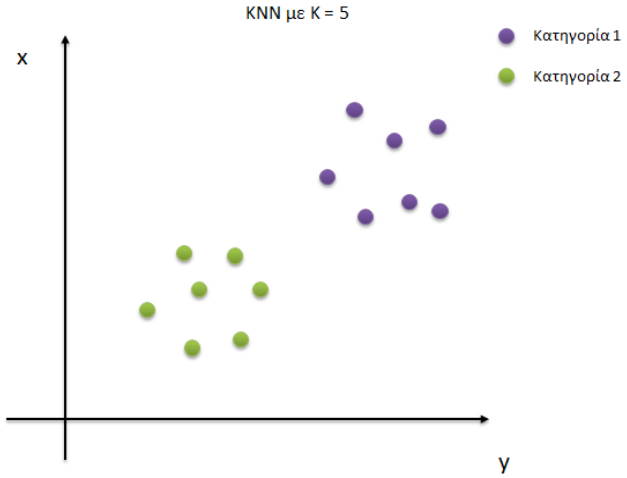
Εικόνα 7: Αποτελέσματα με εφαρμογή Λογιστικής Παλινδρόμησης

3.2.2 K-Nearest Neighbors

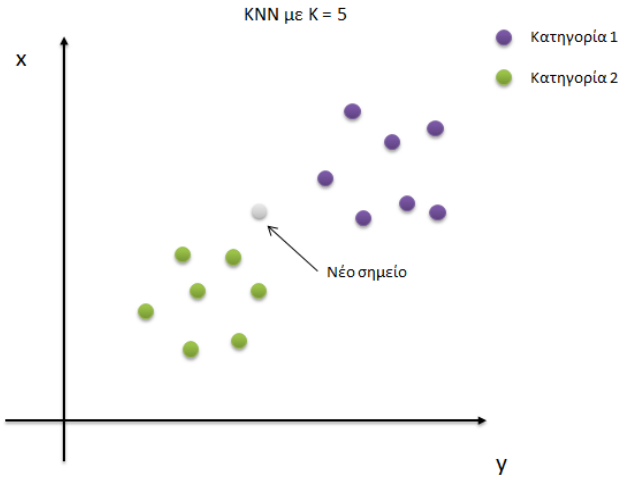
Ο αλγόριθμος KNN είναι μια πολύ γνωστή τεχνική ιδιαίτερα όταν υπάρχει μεγάλος όγκος δεδομένων. Ο ταξινομητής KNN κατηγοριοποιεί ένα άγνωστο δείγμα και χρησιμοποιεί σαν βάση την κυρίαρχη κατηγορία των k κοντινότερων γειτόνων του. Ως κοντινότερους γείτονες θεωρεί τα δείγματα που έχουν από αυτό την μικρότερη απόσταση. Τα βήματα που ακολουθεί ο αλγόριθμος είναι τα εξής:

1. Υπολογισμός της απόστασης του αντικειμένου x προς ταξινόμηση με κάθε μια από τις παρατηρήσεις x_i στο δείγμα εκπαίδευσης χρησιμοποιώντας την Ευκλείδεια απόσταση.
2. Εντοπισμός των k παρατηρήσεων με την μικρότερη απόσταση από το προς ταξινόμηση αντικείμενο και έλεγχος σε ποια κατηγορία ταξινόμησης ανήκουν .
3. Επανάληψη των βημάτων 1-2 για κάθε αντικείμενο.

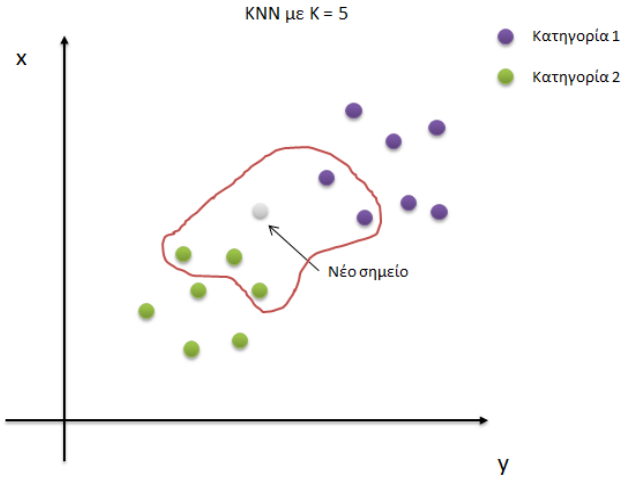
Αν έχουμε δύο κατηγορίες δεδομένων, όπως φαίνεται στο σχήμα, και θέλουμε να εξετάσουμε σε ποια κατηγορία θα ενταχθεί ένα νέο σημείο που προστίθεται στο σύνολο δεδομένων, θα χρησιμοποιήσουμε τον αλγόριθμο KNN που περιγράψαμε θέτοντας τον αριθμό των γειτόνων k.



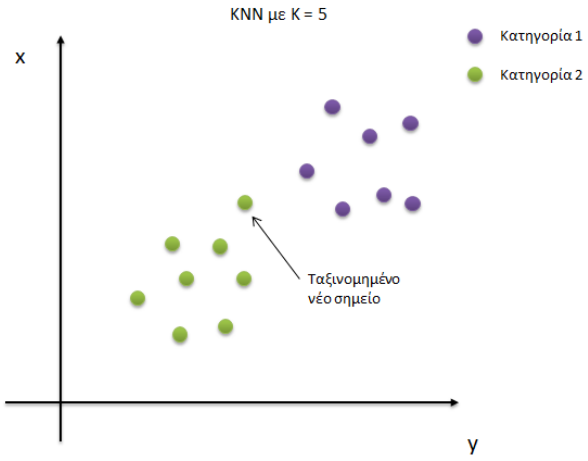
Εικόνα 8: Διαγραμματική απεικόνιση δεδομένων KNN



Εικόνα 9: Προσθήκη νέου σημείου για κατηγοριοποίηση



Εικόνα 10: Κοντινότερα γειτονικά σημεία



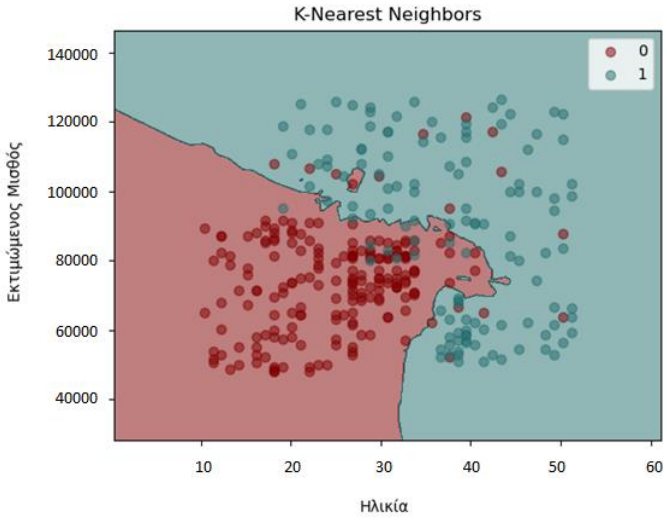
Εικόνα 11: Κατηγοριοποίηση του σημείου

Παράδειγμα

Έστω πως έχουμε ένα σύνολο δεδομένων που περιλαμβάνει πληροφορίες για χρήστες μέσων κοινωνικής δικτύωσης και αν αγόρασαν ένα διαφημιζόμενο προϊόν ή όχι.

Age	EstimatedSalary	Purchased
19	19000	0
35	20000	0
26	43000	0
27	57000	0
19	76000	0
27	58000	0
27	84000	0
32	150000	1
25	33000	0
35	65000	0
26	80000	0
26	52000	0
20	86000	0
32	18000	0
18	82000	0
29	80000	0
47	25000	1
45	26000	1
46	28000	1
48	29000	1
45	22000	1
47	49000	1

Εικόνα 12: Σύνολο Δεδομένων KNN



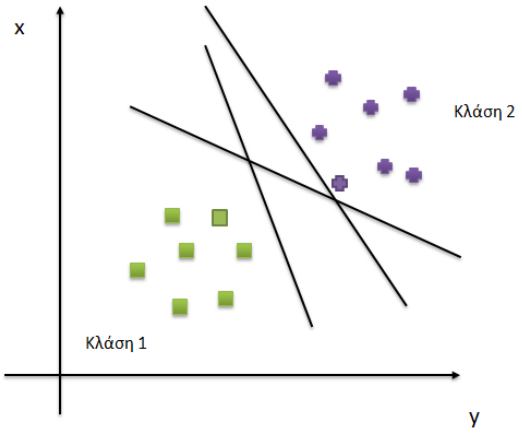
Εικόνα 13: Γραφική Απεικόνιση KNN

3.2.3 Support Vector Machine

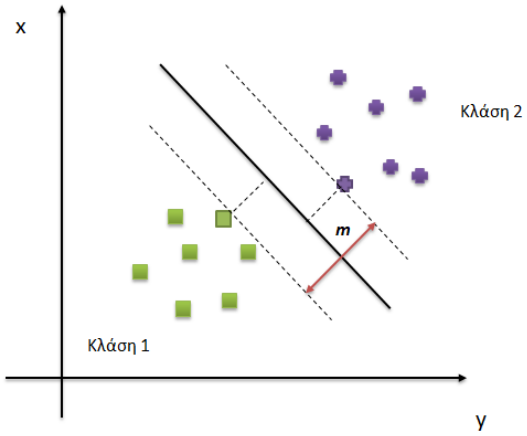
Ο αλγόριθμος Support Vector Machine (SVM) είναι ένας από τους πιο σημαντικούς αλγόριθμους κατηγοριοποίησης. Αναπαριστώντας τα δεδομένα μας διαγραμματικά παρατηρούμε ότι υπάρχουν πολλοί τρόποι να τα διαχωρίσουμε. Ο στόχος του αλγορίθμου είναι να δημιουργήσει ένα βέλτιστο διαχωριστικό υπερεπίπεδο. Η κατηγοριοποίηση των δεδομένων προκύπτει από την εύρεση αυτού του υπερεπιπέδου που διαχωρίζει τα δεδομένα δημιουργώντας το μέγιστο περιθώριο (m) ανάμεσα στα θετικά και αρνητικά παραδείγματα. Κάθε SVM είναι δυαδικός ταξινομητής, δηλαδή μπορεί να κατηγοριοποιήσει τα δεδομένα σε δύο κλάσεις.

Ο αλγόριθμος του SVM είναι ένας αλγόριθμος βελτιστοποίησης. Ξεκινάει δημιουργώντας με μια ευθεία γραμμή (hyperplane) και δύο παράλληλες εκατέρωθεν σε αυτήν. Το περιθώριο που πρέπει να μεγιστοποιήσουμε δημιουργείται από το άθροισμα των αποστάσεων των σημείων που βρίσκονται πάνω σε αυτές τις παράλληλες με την ευθεία γραμμή που διαχωρίζει τις κλάσεις. Τα σημεία που δημιουργούν αυτές τις δύο ευθείες ονομάζονται support vectors.

Για την μεγιστοποίηση του περιθωρίου ο αλγόριθμος ξεκινάει με μία ευθεία και τις παράλληλες σε αυτή. Για έναν μεγάλο αριθμό επαναλήψεων διαλέγουμε έναν αριθμό κοντά στο 1 που ονομάζεται expanding factor. Στη συνέχεια, διαλέγουμε ένα τυχαίο σημείο και αν είναι σωστά ταξινομημένο δεν κάνουμε τίποτα αλλιώς μετακινούμε τη γραμμή πιο κοντά στο σημείο. Κάθε φορά οι ευθείες απομακρύνονται περισσότερο μέχρι που καταλήγουμε στην τελική ευθεία.



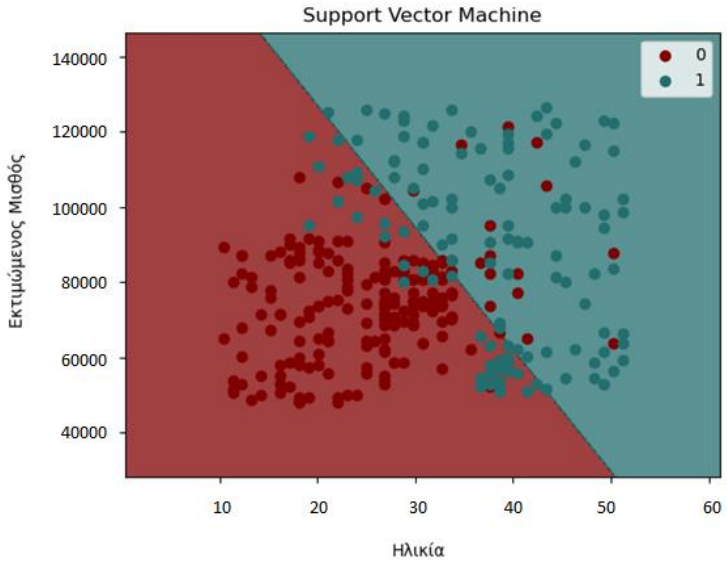
Εικόνα 14: Διάφορες ευθείες διαχωρισμού επιπέδου



Εικόνα 15: Υπερεπίπεδο SVM

Παράδειγμα

Χρησιμοποιούμε το ίδιο σύνολο δεδομένων με το προηγούμενο παράδειγμα.



Εικόνα 16: Γραφική Απεικόνιση SVM

3.3 Συσταδοποιηση

Η συσταδοποίηση είναι όμοια με την κατηγοριοποίηση απλά διαφέρει η βάση τους. Στην περίπτωση της συσταδοποίησης δεν γνωρίζουμε τι θέλουμε να προβλέψουμε, αλλά θέλουμε να δημιουργήσουμε και να αναγνωρίσουμε κάποια πρότυπα. Δηλαδή, δημιουργούνται ομάδες δεδομένων από το σύνολο που ακολουθούν παρόμοιες συμπεριφορές.

3.3.1 K-means

Ο αλγόριθμος k-means είναι ένας από τους απλούστερους και περισσότερο χρησιμοποιούμενους αλγορίθμους για την ομαδοποίηση των δεδομένων. Είναι ένα εργαλείο για την δημιουργία κατηγοριών στο σύνολο δεδομένων. Ο k-means ακολουθεί μια εύκολη διαδικασία για την ομαδοποίηση με δεδομένο ότι ο αριθμός των ομάδων (clusters) πρέπει να είναι γνωστός εκ των προτέρων.

Τα βήματα που ακολουθεί ο αλγόριθμος είναι τα εξής:

1. Ο αλγόριθμος παίρνει σαν είσοδο το πλήθος K των συστάδων(clusters).
2. Επιλέγει K τυχαία σημεία (centroids), όχι απαραίτητα από το dataset.
3. Αναθέτει κάθε σημείο από τα δεδομένα μας στο κοντινότερο centroid και έτσι σχηματίζονται τα πρώτα clusters. Ο υπολογισμός της απόστασης των σημείων από το σημείο centroid γίνεται με χρήση της Ευκλείδειας απόστασης και το κάθε σημείο αντιστοιχίζεται στην ανάλογη ομάδα.
4. Επαναϋπολογίζει και τοποθετεί καινούργια κέντρα στην κάθε συστάδα.
5. Επαναθέτει το κάθε σημείο στη νέα κοντινότερη συστάδα. Αν η επανάθεση γίνει επιστρέφει στο βήμα 4, αλλιώς στο τέλος και το μοντέλο είναι έτοιμο.

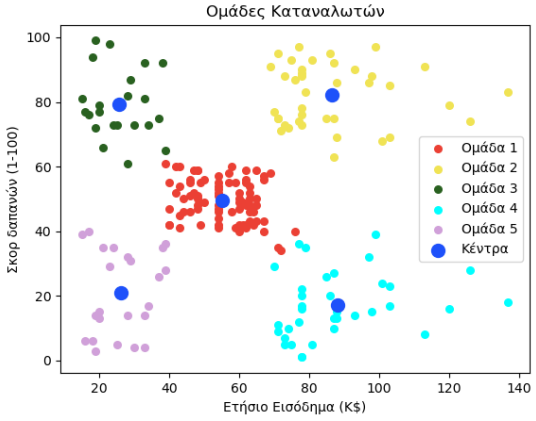
Μια παγίδα που μπορεί να εντοπίσουμε στον αλγόριθμο k-means είναι στην επιλογή των αρχικών κεντρικών σημείων. Αυτό έχει ως αποτέλεσμα να μην βρίσκει πάντα τη βέλτιστη λύση. Μια καλή πρακτική είναι η επαναληπτική εκτέλεση του αλγορίθμου μέχρι να μειωθεί η επίδραση αυτής της επιλογής.

Παράδειγμα

Το σύνολο δεδομένων που θα χρησιμοποιήσουμε, περιέχει πληροφορίες σχετικά με τους πελάτες ενός Εμπορικού Κέντρου. Οι πληροφορίες αφορούν το φύλο, την ηλικία, το ετήσιο εισόδημα και το σκορ κατανάλωσης, το οποίο είναι ένας αριθμός μεταξύ του 1 – 100 και δηλώνει πόσα χρήματα ξόδεψε ο κάθε πελάτης συγκριτικά μέσα σε αυτή την κλίμακα για μια συγκεκριμένη χρονική περίοδο. Σκοπός του παραδείγματος είναι να δημιουργήσουμε πρότυπα πελατών ανάλογα με το ετήσιο εισόδημα τους και το σκορ κατανάλωσης. Στη συσταδοποίηση δεν υπάρχει εξαρτημένη μεταβλητή εξαρχής, καθώς δεν ξέρουμε τι θα προβλέψουμε αλλά δημιουργείται για τον αλγόριθμο. Η επιλογή του αριθμού συστάδων γίνεται στην αρχή. Για την εύρεση του βέλτιστου αριθμού χρησιμοποιείται η μέθοδος Elbow. Στο παράδειγμά μας ο βέλτιστος αριθμός συστάδων είναι το 5.

Customer	Genre	Age	Annual Income	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35
18	Male	20	21	66
19	Male	52	23	29
20	Female	35	23	98
21	Male	35	24	35
22	Male	25	24	73
23	Female	46	25	5
24	Male	31	25	73
25	Female	54	28	14
26	Male	29	28	82
27	Female	45	28	32

Εικόνα 17: Σύνολο Δεδομένων Kmeans



Εικόνα 18: Γραφική Απεικόνιση Kmeans

4. Βαθια Μαθηση

Η Βαθιά Μάθηση (Deep Learning) είναι ένα εξειδικευμένο πεδίο της Μηχανικής Μάθησης που βασίζεται στην εκπαίδευση των Τεχνητών Νευρωνικών Δικτύων (Artificial Neural Networks (ANNs)) χρησιμοποιώντας ένα μεγάλο σύνολο δεδομένων , όπως εικόνες και κείμενα.

4.1 Τεχνητα νευρωνικα δικτυα

Τα Τεχνητά νευρωνικά δίκτυα ή ANN (Artificial Neural Networks) είναι μοντέλα επεξεργασίας πληροφοριών εμπνευσμένα από τον ανθρώπινο εγκέφαλο. Ο ανθρώπινος εγκέφαλος αποτελείται από δισεκατομμύρια νευρώνες που επικοινωνούν μεταξύ τους χρησιμοποιώντας ηλεκτρικά και χημικά σήματα και επιτρέπουν στους ανθρώπους να βλέπουν, να αισθάνονται και να αποφασίζουν. Τα ANN μιμούνται μαθηματικά τον ανθρώπινο εγκέφαλο και συνδέουν πολλαπλούς «τεχνητούς» νευρώνες με πολυστρωματικό τρόπο. Όσο περισσότερα κρυφά στρώματα προστίθενται στο δίκτυο, τόσο βαθύτερα γίνεται το δίκτυο.

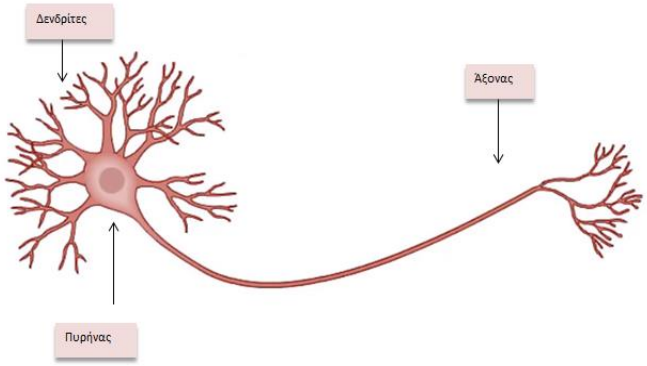
Το κύριο χαρακτηριστικό τους είναι ότι οι αρχές και λειτουργίες τους βασίζονται στο νευρικό σύστημα του ανθρώπου, αλλά η μελέτη και η χρήση τους έχει προχωρήσει πολύ πέρα από τους βιολογικούς οργανισμούς, και σήμερα τα νευρωνικά δίκτυα χρησιμοποιούνται για να λύσουν κάθε είδους προβλήματα με ηλεκτρονικό υπολογιστή. Η λειτουργία τους προσπαθεί να συνδυάσει τον τρόπο σκέψης του ανθρώπινου εγκεφάλου με τον αφηρημένο μαθηματικό τρόπο σκέψης. Η έμπνευση για κάθε μορφής νευρωνικό δίκτυο ξεκινά από την βιολογία. Οι ζώντες οργανισμοί, από τους πιο απλούς μέχρι τον άνθρωπο, έχουν ένα νευρικό σύστημα, το οποίο είναι υπεύθυνο για διεργασίες, όπως είναι η επαφή με τον εξωτερικό κόσμο, η μάθηση, η μνήμη, κλπ. Το νευρικό σύστημα των οργανισμών αποτελείται από πολλά νευρωνικά δίκτυα τα οποία είναι εξειδικευμένα στις διεργασίες αυτές. Η κεντρική μονάδα του νευρικού συστήματος είναι, οπωσδήποτε, ο εγκέφαλος, ο οποίος επίσης αποτελείται από νευρωνικά δίκτυα. Κάθε νευρωνικό δίκτυο αποτελείται από ένα μεγάλο αριθμό μονάδων, που λέγονται νευρώνες (neurons). Ο νευρώνας είναι η πιο μικρή ανεξάρτητη μονάδα του δικτύου. Οι νευρώνες επεξεργάζονται πληροφορίες, παίρνοντας και στέλνοντας ηλεκτρικά σήματα σε άλλους νευρώνες.

4.1.1 Ο νευρωνας

Ο νευρώνας είναι το βασικό δομικό στοιχείο ενός τεχνητού νευρωνικού δικτύου. Στην βιολογία νευρώνας είναι το κύτταρο που αποτελεί δομικό μέρος και λειτουργική μονάδα του νευρικού συστήματος. Τα μέρη που αποτελούν το νευρώνα είναι το κυτταρικό σώμα που περιλαμβάνει τον πυρήνα και μεγάλο αριθμό οργανιδίων, και μία ή περισσότερες αποφυάδες. Αυτές ονομάζονται δενδρίτες όταν συλλέγουν τα σήματα που στέλνονται στο κύτταρο, και άξονες όταν μεταδίδουν ώσεις από το κυτταρικό σώμα.

Το κρίσιμο ερώτημα είναι πως μπορεί να αναπαρασταθεί μια τέτοια δομή σε ένα μηχάνημα, καθώς ο κύριος λόγος της βαθιάς μάθησης είναι να δημιουργηθεί ένα σύστημα που μιμείται τον τρόπο που λειτουργεί ο ανθρώπινος εγκέφαλος, ένας από τους πιο δυνατούς μηχανισμούς εκμάθησης στον πλανήτη.

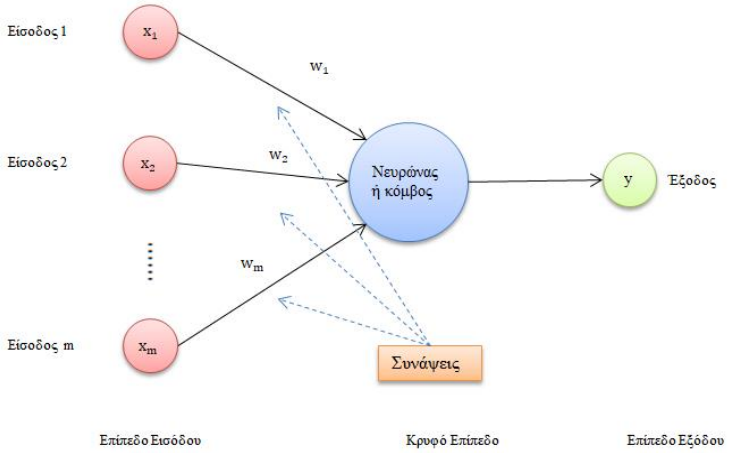
Το πρώτο βήμα αυτής της πρόκλησης, δηλαδή της δημιουργίας ενός τεχνητού νευρωνικού δικτύου είναι η αναπαράσταση του νευρώνα. Διαγραμματικά, ένας νευρώνας μοιάζει:



Εικόνα 19: Βιολογικός Νευρώνας (Image source : Wikipedia)

Αξιοσημείωτο είναι το γεγονός πως ένας νευρώνας πρακτικά δεν μπορεί να δουλέψει. Κάθε νευρώνας από μόνος του δεν είναι τόσο δυνατός, αλλά όταν δημιουργείται ένα ολόκληρο δίκτυο από νευρώνες μπορούν να συμβούν σπουδαία πράγματα. Οι νευρώνες ενώνονται με τους δενδρίτες και τον άξονα. Οι δενδρίτες κάνουν και αυτοί επαφή με άλλους νευρώνες, και δέχονται τα εισερχόμενα σήματα. Οι συνδέσεις μεταξύ των νευρώνων, με τους άξονες και τους δενδρίτες, γίνονται στις επαφές που ονομάζονται συνάψεις. Η σύναψη έχει πολύ περίπλοκη δομή και επιτελεί επίσης περίπλοκες διεργασίες κατά την μετάδοση του σήματος. Ο άξονας όπως είδαμε συνήθως έχει πάρα πολλές διακλαδώσεις και έτσι στέλνει πολλά σήματα σε διαφορετικά σημεία. Στα σημεία που εφάπτονται οι δενδρίτες δημιουργείται μία σύναψη.

Στο παρακάτω σχήμα βλέπουμε πως αναπαριστάται ένας νευρώνας, τον ονομάζουμε και κόμβο, σε μια μηχανή. Ο νευρώνας δέχεται κάποια εισερχόμενα σήματα και παράγει ένα εξερχόμενο σήμα, όπως αντίστοιχα οι δενδρίτες και οι άξονες. Σε αναλογία με τον ανθρώπινο εγκέφαλο το επίπεδο των εισερχόμενων σημάτων είναι όλες οι πληροφορίες που λαμβάνονται από τις ανθρώπινες αισθήσεις. Σε όρους μηχανικής μάθησης οι μεταβλητές εισόδου στην ουσία είναι οι ανεξάρτητες μεταβλητές που υπάρχουν σε ένα σύνολο δεδομένων. Στο συγκεκριμένο παράδειγμα έχουμε το επίπεδο εισόδου , το κρυφό επίπεδο και το επίπεδο εξόδου. Αυτή είναι η βασική δομή.



Εικόνα 20: Βασική δομή Τεχνητού νευρώνα

Παρατηρώντας ξεχωριστά τα παραπάνω στοιχεία, ξεκινώντας από το επίπεδο εισόδου, συμπεραίνουμε ότι αναφέρεται στις ανεξάρτητες μεταβλητές. Μια σημαντική παρατήρηση είναι πως όλες αυτές οι ανεξάρτητες μεταβλητές αφορούν μια μόνο παρατήρηση, δηλαδή μια γραμμή στη βάση δεδομένων μας. Οι συνάψεις συνοδεύονται από ένα βάρος wi. Τα βάρη έχουν κρίσιμο ρόλο στα νευρωνικά δίκτυα, καθώς προσδιορίζουν τον τρόπο με τον όποιο ένα νευρωνικό εκπαιδεύεται και διακρίνει ποιο σήμα είναι σημαντικό και ποιο λιγότερο σημαντικό. Μέσα στο νευρώνα, το πρώτο βήμα που γίνεται είναι ότι όλες αυτές οι τιμές πολλαπλασιασμένες με το βάρος τους προστίθενται. Έπειτα εφαρμόζεται η συνάρτηση ενεργοποίησης, την οποία θα μελετήσουμε αναλυτικά στην επόμενη παράγραφο. Περιεκτικά, πρόκειται για μια συνάρτηση, την οποία ο κάθε νευρώνας εφαρμόζει στο σταθμισμένο άθροισμα και από το αποτέλεσμα της κρίνεται αν το σήμα θα περάσει ή όχι. Η έξοδος που θα δώσει το νευρωνικό σύστημα μπορεί να είναι μια συνεχής μεταβλητή, μια δυαδική (true/false) είτε μια κατηγορική.

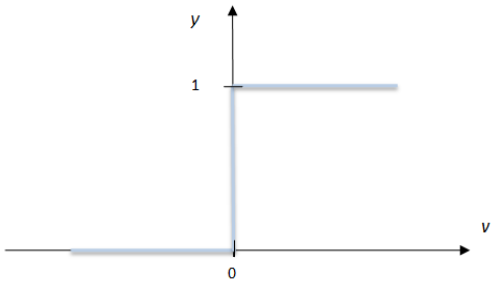
4.1.2 Συναρτηση ενεργοποιησης Τεχνητου νευρωνα

Το επόμενο βήμα είναι η συνάρτηση ενεργοποίησης που θα εφαρμοστεί πάνω στο νευρώνα και ορίζει την τιμή εξόδου του συγκεκριμένου κόμβου δοσμένου ενός συνόλου από εισόδους. Στη συνέχεια θα παρουσιάσουμε τέσσερις διαφορετικούς τύπους συναρτήσεων ενεργοποίησης που χρησιμοποιούνται συχνότερα.

- Συνάρτηση κατωφλιού – Threshold Function

Ο άξονας x περιλαμβάνει το άθροισμα των τιμών εισόδου πολλαπλασιασμένες με τα βάρη, $v = \sum_{i=1}^{m} w_i x_i$, και στον άξονα y είναι οι τιμές από 0 έως 1. Ο υπολογισμός της τιμής εξόδου γίνεται με έναν απλό κανόνα: εάν το σταθμισμένο άθροισμα $\sum_{i=1}^{m} w_i x_i$ είναι μικρότερο ή μεγαλύτερο από κάποια τιμή κατωφλίου *θ* (threshold):

$$\varphi(v) = \begin{cases} 0, v < \theta \\ 1, v \geq \theta \end{cases}$$



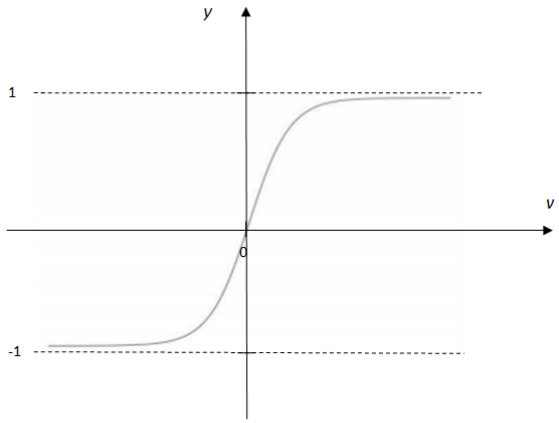
Εικόνα 21: Γραφική απεικόνιση συνάρτησης κατωφλιού

Ουσιαστικά πρόκειται για μια δυαδική συνάρτηση. Λειτουργεί ακριβώς με το παραπάνω απλό μαθηματικό μοντέλο, δηλαδή βάσει της βαρύτητας κάθε στοιχείου που έχει διαθέσιμο και ανάλογα την τιμή του κατωφλιού που έχουμε ορίσει λαμβάνει τις αποφάσεις. Αν το σταθμισμένο άθροισμα είναι αρνητικό τότε η συνάρτηση γίνεται ίση με το μηδέν(false), αλλιώς αν είναι θετικό τότε η συνάρτηση γίνεται 1(true).

- Σιγμοειδής συνάρτηση

Η σιγμοειδής συνάρτηση είναι πολύ χρήσιμη ειδικά για το επίπεδο εξόδου όταν προσπαθούμε να προβλέψουμε πιθανότητες. Ο τύπος της σιγμοειδούς συνάρτησης είναι:

$\varphi(v) = \frac{1}{1+e^{-v}}$, όπου $v = \sum_{i=1}^{m} w_i x_i$ και η γραφική της απεικόνιση φαίνεται στο σχήμα.

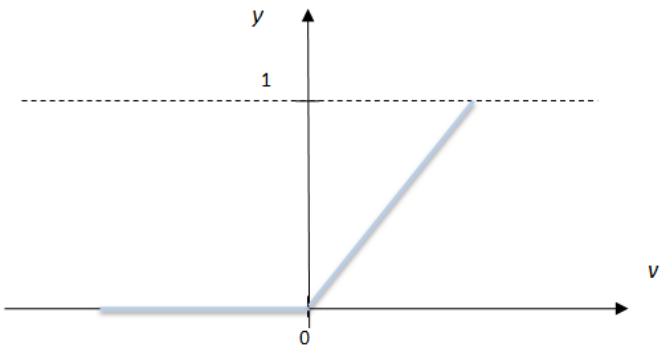


Εικόνα 22: Γραφική απεικόνιση σιγμοειδούς συνάρτησης

Η συνάρτηση αυτή είναι πιο ομαλή σε σχέση με την συνάρτηση κατωφλιού, καθώς η καμπύλη της δεν έχει κόμβους και γωνίες.

- Ανορθωμένη Γραμμική Συνάρτηση Ράμπας (ReLU- Rectifier Linear Unit)

Η συνάρτηση ράμπας έχει την εξής μορφή φ(v) = max (0,v), όπου v = $\sum_{i=1}^{m} w_i x_i$.Η συγκεκριμένη συνάρτηση είναι αρκετά δημοφιλής για βαθιά νευρωνικά δίκτυα . Προτιμάται από τις υπόλοιπες συναρτήσεις διότι έχει την δυνατότητα να εκπαιδεύσει ένα δίκτυο αρκετά πιο γρήγορα, δίνοντας ακριβή αποτελέσματα. Ωστόσο, ένα σημαντικό μειονέκτημα της είναι πως κάποιες φορές μπορεί να οδηγήσει ορισμένους νευρώνες του δικτύου σε κάποιες τιμές βαρών, οι οποίες τους αποτρέπουν να ενεργοποιηθούν. Έτσι αυτοί οι νευρώνες νεκρώνουν, σταματάνε δηλαδή να εκπαιδεύονται. Η καμπύλη της συνάρτησης ReLU είναι η ακόλουθη:



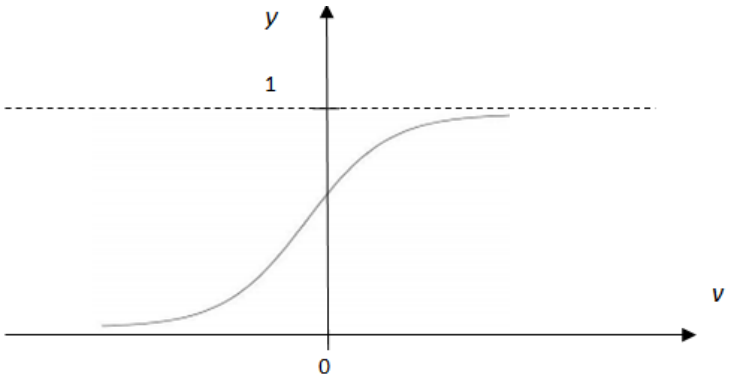
Εικόνα 23: Γραφική απεικόνιση συνάρτησης ReLU

- Η Υπερβολική Εφαπτομένη (Hyperbolic Tangent)

Είναι αρκετά όμοια με τη σιγμοειδή συνάρτηση αλλά η συνάρτηση υπερβολικής εφαπτομένης πηγαίνει και κάτω από το 0. Η συνάρτηση αυτή μοιάζει και διαγραμματικά με τη σιγμοειδή συνάρτηση με τη διαφορά ότι παίρνει τιμές στο διάστημα (−1,1). Είναι προτιμότερη από

την σιγμοειδή λόγω του γεγονότος ότι οι τιμές εξόδου είναι κεντραρισμένες στο μηδέν. Ο τύπος της συνάρτησης είναι :

$\varphi_{tanh(v)} = \frac{\sinh(v)}{\cosh(v)} = \frac{e^{v}-e^{-v}}{e^{v}+e^{-v}}$, όπου $v = \sum_{i=1}^{m} w_i x_i$ και η γραφική της απεικόνιση φαίνεται στο σχήμα.



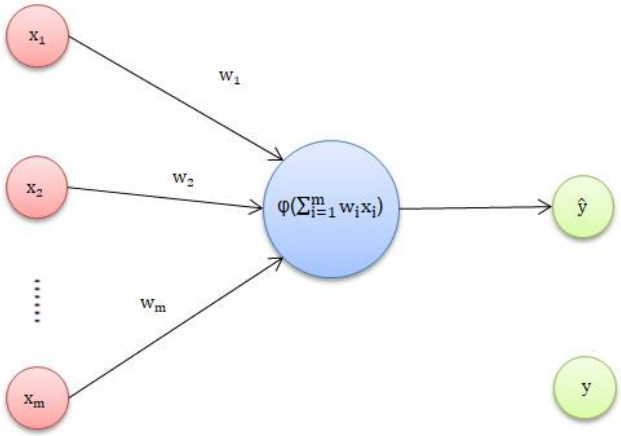
Εικόνα 24: Γραφική απεικόνιση συνάρτησης Υπερβολικής εφαπτομένης

4.1.3 Διαδικασια εκπαιδευσης

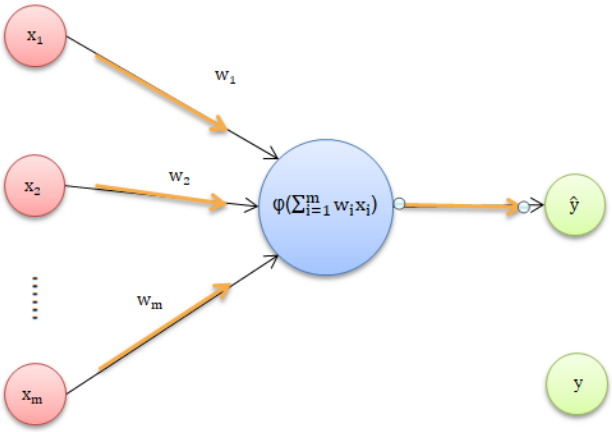
Ο στόχος είναι η δημιουργία ενός τεχνητού νευρωνικού δικτύου το οποίο μπορεί να μάθει μόνο του ώστε να λύνει συγκεκριμένα προβλήματα ή να εκτελεί μόνο του κάποιες διεργασίες, όπως αναγνώριση αντικειμένων ή φωνής. Για να επιτευχθεί αυτό πρέπει να δημιουργήσουμε ένα δίκτυο που θα εκπαιδευτεί κατάλληλα για την διεργασία που προορίζεται. Το δίκτυο τροφοδοτείται με πολλά δεδομένα τα οποία είναι ήδη κατηγοριοποιημένα, δηλαδή του δίνονται κάποια πρότυπα όπου για το καθένα αντιστοιχεί στη σωστή απάντηση. Για παράδειγμα αν θέλουμε το δίκτυο μας να μπορεί να αναγνωρίσει έναν σκύλο ή μια γάτα, του δίνουμε πάρα πολλές εικόνες με σκύλους και γάτες που είναι ήδη κατηγοριοποιημένες και το δίκτυο πρέπει να μάθει πώς μοιάζει ένας σκύλος και πώς μια γάτα. Έτσι το δίκτυο εκπαιδεύεται και όταν του δοθεί μια νέα εικόνα θα μπορεί να αναγνωρίσει αν είναι σκύλος ή γάτα. Επομένως, δίνουμε στο δίκτυο ως εισόδους κάποια πρότυπα για τα οποία ξέρουμε ποια πρέπει να είναι η έξοδος. Γνωρίζουμε ποιος είναι ο στόχος, τι πρέπει να δίνει το δίκτυο ως απάντηση στα πρότυπα που του παρουσιάζουμε. Το δίκτυο χρησιμοποιεί την κατάλληλη συνάρτηση ενεργοποίησης για να μεταδίδει το σήμα σε όλη τη δομή του, από την είσοδο ως την έξοδο. Κατά την εκπαίδευση αλλάζουν οι τιμές των βαρών των συνάψεων. Η αλλαγή των τιμών εξαρτάται από την μέθοδο που χρησιμοποιούμε.

Το δίκτυο με τα δεδομένα αυτά τροποποιεί την εσωτερική του δομή, ώστε να μπορεί να κάνει την ίδια αντιστοιχία που του δώσαμε εμείς. Ενώ αρχικά ξεκινάει με τυχαίες τιμές στα βάρη w, κατά την διάρκεια της εκπαίδευσης μεταβάλλει τις τιμές αυτές μέχρι να εκπαιδευθεί πλήρως. Ακολούθως, αφού βρει την σωστή εσωτερική δομή του, τότε θα μπορεί να αναγνωρίζει και άλλα ανάλογα πρότυπα τα οποία δεν τα έχει δει προηγουμένως, δηλαδή δεν έχει εκπαιδευθεί στα πρότυπα των προβλημάτων αυτών.

Στην εικόνα βλέπουμε την βασική δομή του νευρωνικού δικτύου με ένα επίπεδο (single layer feedforward neural network ή perceptron). Η ιδέα του perceptron επινοήθηκε πρώτη φορά το 1957 από τον Frank Rosenblat και η συνολική ιδέα του ήταν να δημιουργήσει κάτι το οποίο θα μαθαίνει και θα προσαρμόζεται μόνο του. Ονομάζουμε την προβλεπόμενη τιμή που θα έχουμε ως αποτέλεσμα $\hat{y}$, και y ονομάζουμε το πραγματικό αποτέλεσμα που πρέπει αν παίρνουμε ως έξοδο.



Εικόνα 25: Βασική δομή του νευρωνικού δικτύου με ένα επίπεδο



Εικόνα 26: Αλγόριθμος Feedforward

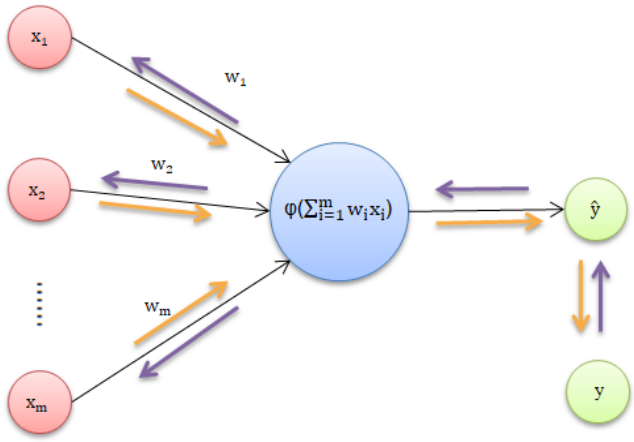
Έστω πως έχουμε κάποιες τιμές εισόδου που δέχεται το perceptron και έπειτα εφαρμόζεται η συνάρτηση ενεργοποίησης και παίρνουμε τις τιμές εξόδου. Δημιουργώντας ένα απλό διάγραμμα για την αναπαράσταση της προβλεπόμενης τιμής εξόδου και της πραγματικής, τις συγκρίνουμε.

- Συνάρτηση κόστους

Επόμενο βήμα είναι ο υπολογισμός της συνάρτησης κόστους. Στην ουσία η συνάρτηση κόστους είναι αυτή που προσδιορίζει ποιο είναι το σφάλμα της πρόβλεψης. Ο στόχος μας είναι να ελαχιστοποιήσουμε την συνάρτηση κόστους και να μπορέσουμε να φτάσουμε όσο πιο κοντά στην πραγματική τιμή. Υπάρχουν πολλές συναρτήσεις κόστους η πιο δημοφιλής είναι η συνάρτηση ελαχίστων τετραγώνων:

$$C= \frac{1}{2} (\hat{y} - y)^2$$

Μετά τον υπολογισμό της συνάρτησης κόστους επιστρέφεται στο δίκτυο αυτή η πληροφορία και τα βάρη στις συνάψεις ανανεώνονται, αλλάζοντας τα κατά λίγο. Η διαδικασία αυτή επαναλαμβάνεται μέχρι η συνάρτηση κόστους να ελαχιστοποιηθεί, δηλαδή να φτάσουμε κοντύτερα στην πραγματική τιμή.



Εικόνα 27: Αλγόριθμος οπισθοδιάδοσης

Στο σχήμα, όλη η διαδικασία αυτή αφορά μόνο μια γραμμή παρατηρήσεων. Όταν μελετούμε πολλές παρατηρήσεις, δηλαδή ένα ολόκληρο σύνολο δεδομένων ή μέρος αυτού, ο υπολογισμός αφορά όλα τα δεδομένα. Το χρονικό διάστημα που χρειάζεται για να εκπαιδευτεί το νευρωνικό δίκτυο σε όλες τις παρατηρήσεις ονομάζεται epoch. Για κάθε epoch εφαρμόζεται η συνάρτηση ελαχίστων τετραγώνων σε όλες τις παρατηρήσεις και στη συνέχεια υπολογίζεται το άθροισμά τους. Δηλαδή:

$$C = \sum \frac{1}{2}(\hat{y} - y)^2$$

Όλα τα perceptrons μαζί αποτελούν το νευρωνικό δίκτυο άρα τα βάρη και η ανανέωσή τους είναι ίδια για όλες τις παρατηρήσεις του συνόλου δεδομένων . Η διαδικασία αυτή ονομάζεται αλγόριθμος οπισθοδιάδοσης (backpropagation).

- Μέθοδος επικλινούς καθόδου (gradient descent)

Ο gradient descent είναι μια τεχνική που χρησιμοποιείται για την ελαχιστοποίηση του σφάλματος, δηλαδή της συνάρτησης κόστους, στα τεχνητά νευρωνικά δίκτυα. Η καμπύλη σφάλματος δεν είναι ποτέ κοίλη. Αυτό σημαίνει πως ο αλγόριθμος μπορεί να βρεθεί και να «εγκλωβιστεί» σε ένα τοπικό ελάχιστο και όχι στο ολικό ελάχιστο που είναι και το ζητούμενο.

Ο αλγόριθμος ξεκινάει από μια αρχική τιμή βαρών τυχαία και σε κάθε επανάληψη υπολογίζεται η κλίση και ενημερώνονται τα βάρη.

Αυτό το πρόβλημα συνήθως επιλύεται με τη χρήση στατιστικών μεθόδων εκπαίδευσης ή την επανεκκίνηση της διαδικασίας εκμάθησης με νέα αρχικοποίηση. Μία παράμετρος που επηρεάζει την ταχύτητα εκμάθησης είναι το βήμα καθόδου ή αλλιώς ο ρυθμός μάθησης (learning rate) ο οποίος καθορίζει το μέγεθος του βήματος προς την κατεύθυνση της μείωσης της συνάρτησης. Όταν η τιμή του ρυθμού μάθησης είναι μικρή συνεπάγεται ομαλή κάθοδο προς το τοπικό ελάχιστο, αλλά απαιτούνται περισσότερες επαναλήψεις.

Αντιθέτως, όταν η τιμή του ρυθμού μάθησης είναι μεγάλη, τα βάρη των συνάψεων αλλάζουν σε μεγάλο βαθμό και υπάρχει ο κίνδυνος αυξημένης πιθανότητα εμφάνισης ταλαντώσεων γύρω από το σημείο ελαχίστου, δηλαδή να μην συμβεί επί της ουσίας 'κάθοδος' στην καμπύλη.

- Μέθοδος Στοχαστικής Επικλινούς Καθόδου (Stochastic Gradient Descent - SGD)

Η μέθοδος stochastic gradient descent, είναι μία παραλλαγή της τεχνικής gradient descent. Λειτουργεί με τον ίδιο τρόπο με σκοπό την ελαχιστοποίηση του σφάλματος, με τη διαφορά πως η όλη διαδικασία πρόβλεψης του σφάλματος γίνεται πολύ γρηγορότερα. Στη βασική μορφή της μεθόδου SGD, κάτι τέτοιο επιτυγχάνεται με τη χρήση μόλις ενός δείγματος εκπαίδευσης για την κάθε επιμέρους πρόβλεψη της κλίσης.

4.2 Συνελικτικα νευρωνικα δικτυα

Τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Network - CNN) είναι ένα είδος νευρωνικών δικτύων. Στον τομέα της αναγνώρισης και κατηγοριοποίησης εικόνων έχει καθιερωθεί η χρήση τους. Οι εικόνες μπορεί να είναι ασπρόμαυρες ή έγχρωμες και αποτελούνται από pixel. Κάθε εικόνα περιέχει έναν αριθμό από pixel, που υπολογίζεται απλά πολλαπλασιάζοντας κάθε της διάσταση. Οι έγχρωμες εικόνες έχουν 3 διαστάσεις: πλάτος, ύψος και βάθος. Ως βάθος ορίζονται τα 3 επίπεδα χρωμάτων Κόκκινο-Πράσινο-Μπλε (RGB). Στην περίπτωση που είναι ασπρόμαυρες υπάρχει μόνο ένα επίπεδο. Μια ασπρόμαυρη εικόνα αναπαρίσταται από έναν πίνακα δύο διαστάσεων όπου κάθε ένα pixel αντιστοιχεί σε έναν αριθμό μεταξύ 0 και 255 που δηλώνει την ένταση του χρώματος, δηλαδή το άσπρο. Αντίστοιχα οι έγχρωμες είναι στην πραγματικότητα ένας τρισδιάστατος πίνακας , όπου κάθε διάσταση αντιστοιχεί σε ένα από τα τρία χρώματα δηλωμένα με έναν αριθμό από 0 έως 255 που δείχνει την ένταση του χρώματος. Επομένως κάθε pixel έχει 3 τιμές από 1 έως 255 που συνθέτουν το συνολικό χρώμα.

Τα νευρωνικά δίκτυα που έχουμε δει μέχρι στιγμής θα αποσυνέθεταν την μορφή των εικόνων αυτών από τρισδιάστατη και θα δημιουργούσαν ένα διάνυσμα μίας διάστασης παραθέτοντας γραμμικά τα στοιχεία από όλες τις διαστάσεις. Η σημαντική εξέλιξη των συνελικτικών δικτύων είναι ότι μπορούν να δεχθούν ως είσοδο έναν πολυδιάστατο πίνακα δεδομένων. Το γεγονός αυτό δίνει τη δυνατότητα να διατηρηθεί η τρισδιάστατη μορφή των έγχρωμων εικόνων και ως αποτέλεσμα έχει την αποδοτικότερη ανάλυση τους.

Ένα συνελικτικό επίπεδο (convolutional layer) είναι ουσιαστικά ένα σύνολο από νευρώνες που εκτελούν συνέλιξη των φίλτρων που έχουν προκαθοριστεί, με την εικόνα-διάνυσμα που δέχονται στην είσοδο. Κάθε επίπεδο μπορεί να περιλαμβάνει νευρώνες που εκτελούν συνέλιξη, διαδικασίες pooling, εισαγωγή μη γραμμικότητας ή ακόμη και κανονικοποίηση, ενώ έχει διακριτές εισόδους και εξόδους. Οι διαστάσεις των φίλτρων που περιλαμβάνουν, ο αριθμός τους και το βάθος τους (αριθμός καναλιών) μπορεί να διαφέρει σημαντικά ανάλογα με το πρόβλημα.

Η λειτουργία των CNN συνοψίζεται στα εξής τέσσερα βήματα:

1. Συνέλιξη και εφαρμογή μη γραμμικότητας
   (Convolution operation – ReLU layer)
2. Συγκέντρωση
   (Pooling)
3. Κανονικοποίηση
   (Flattening)
4. Πλήρης Σύνδεση
   (Full Connection)

4.2.1 Επιπεδο Συνελιξης

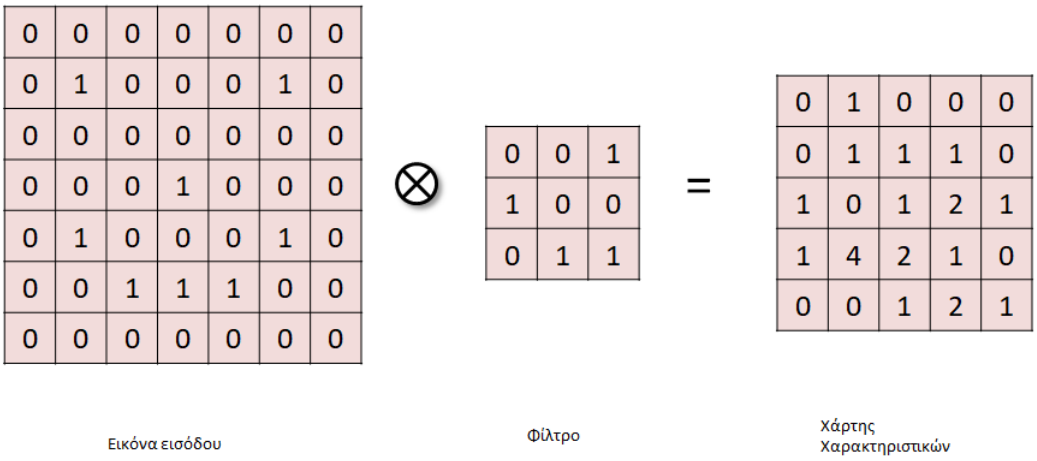
Το πρώτο βήμα και η βάση του μοντέλου CNN είναι το επίπεδο συνέλιξης. Χρησιμοποιείται ένα σύνολο από φίλτρα τα οποία μπορούν να εντοπίσουν την ύπαρξη κάποιων χαρακτηριστικών ή μοτίβων (κλίσεις, ακμές κλπ.) που παρουσιάζονται στην αρχική εικόνα που δίνεται ως είσοδος.

Το συνελικτικό δίκτυο επεξεργάζεται κάθε φορά τετράγωνα κομμάτια από pixel (patches) περνώντας τους ένα φίλτρο. Αυτό το φίλτρο (feature detector) είναι επίσης ένα τετράγωνο πλέγμα μικρότερο από την ίδια την εικόνα και ίσο σε μέγεθος με το κάθε κομμάτι .

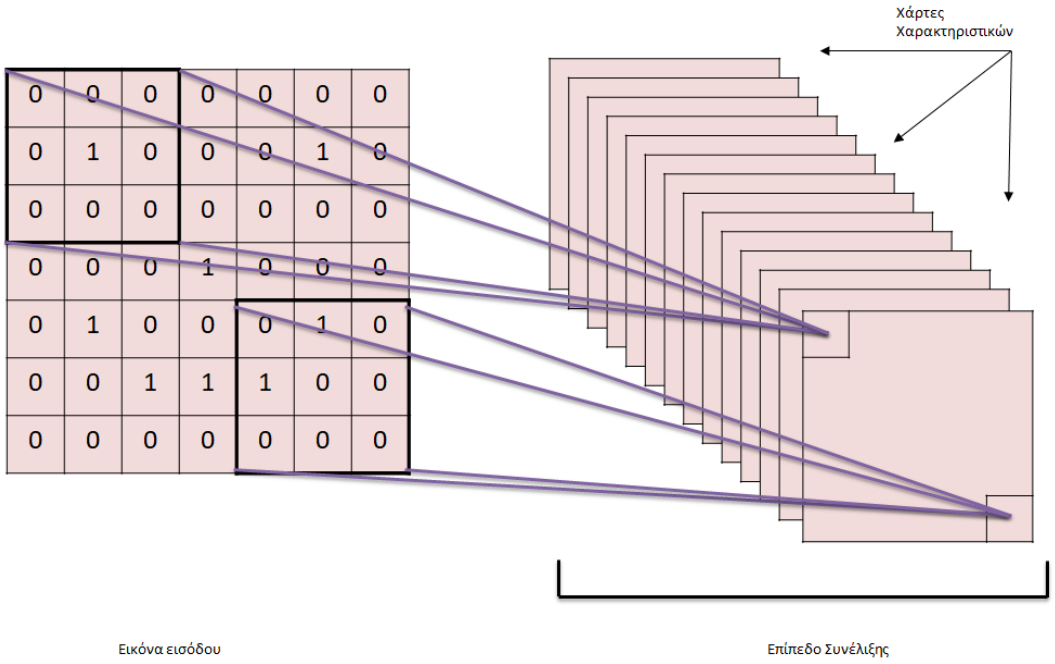
Σκοπός της συνέλιξης είναι η μείωση του μεγέθους της εικόνας για την ευκολότερη και γρηγορότερη επεξεργασία της, δεδομένου ότι μια ολόκληρη φωτογραφία έχει πολύ μεγάλο όγκο δεδομένων. Ένα αξιόλογο ερώτημα θα ήταν αν μέσα από αυτή τη διαδικασία χάνουμε δεδομένα. Η απάντηση είναι ότι προφανώς χάνονται κάποιες πληροφορίες εφόσον στον τελικό πίνακα υπάρχουν και λιγότερες τιμές. Ταυτόχρονα όμως ο σκοπός του φίλτρου είναι να εντοπίσει συγκεκριμένα χαρακτηριστικά σε συγκεκριμένα σημεία της εικόνας που είναι καίριας σημασίας. Αυτό συμβαίνει και στην πραγματική ζωή. Δεν παρατηρούμε κάθε μικρό κομμάτι της εικόνας αλλά τα συγκεκριμένα χαρακτηριστικά που θα μας βοηθήσουν να αντιληφθούμε τι είναι αυτό που κοιτάζουμε.

Όπως είπαμε το κάθε φίλτρο εφαρμόζεται κατά πλάτος και κατά ύψος της εικόνας εισόδου, και ένα εσωτερικό γινόμενο υπολογίζεται για να δώσει το τελικό αποτέλεσμα, έναν χάρτη χαρακτηριστικών (feature map).

Για την δημιουργία του πρώτου επιπέδου συνέλιξης, διαφορετικά φίλτρα τα οποία εντοπίζουν διαφορετικά χαρακτηριστικά περιστρέφονται στην εικόνα εισόδου και ένα σύνολο από χάρτες χαρακτηριστικών προκύπτει ως έξοδος, το οποίο περνά στο επόμενο επίπεδο του ΣΝΔ, όπως φαίνεται και παρακάτω:



Εικόνα 28: Εφαρμογή Φίλτρου - Εξαγωγή Χάρτη Χαρακτηριστικών



Εικόνα 29: Πρώτο επίπεδο συνέλιξης

Μη γραμμικότητα – ReLU

Συνεχίζοντας στο επόμενο βήμα, θα εφαρμόσουμε πάνω στο επίπεδο συνέλιξης την συνάρτηση ενεργοποίησης . Η συνάρτηση αυτή ορίζει την απόφαση που θα παρθεί για το αν θα πυροδοτηθεί ένας νευρώνας ή όχι. Υπάρχουν πολλές συναρτήσεις ενεργοποίησης, αλλά η πιο συνήθης για τα συνελικτικά δίκτυα, είναι η συνάρτηση ανόρθωσης - συνάρτηση ReLU (rectified linear unit) με σκοπό να αυξήσουμε τη μη γραμμικότητα στην εικόνα μας ή στο δίκτυο μας, καθώς οι εικόνες είναι μη γραμμικές ιδιαίτερα στην αναγνώριση διαφορετικών αντικειμένων είτε αυτά είναι το ένα δίπλα στο άλλο είτε απλά βρίσκονται στο φόντο. Για τον υπολογισμό της τιμής του κάθε νευρώνα, χρησιμοποιούνται τα ίδια βάρη και κλίση. Για το λόγο αυτό, ορίζονται ως κοινά βάρη και κλίση (shared weights and bias). Η επιλογή της συγκεκριμένης συνάρτησης γίνεται διότι, όπως φαίνεται και στην γραφική της παράσταση που είδαμε σε προηγούμενο κεφάλαιο, οι αρνητικές τιμές μετατρέπονται σε μηδέν. Αυτό σημαίνει πως ο συγκεκριμένος νευρώνας δεν ενεργοποιείται άρα , συνολικά, δεν ενεργοποιούνται ταυτόχρονα όλοι οι νευρώνες. Το συνολικό σύστημα γίνεται πιο αποδοτικό αφού η ReLU συγκλίνει γρηγορότερα από τη σιγμοειδή και την συνάρτηση εφαπτομένης κατά έξι φορές.

4.2.2 Επιπεδο Συγκεντρωσης (Pooling)

Μετά το στρώμα συνέλιξης ακολουθεί το στρώμα pooling. Το στρώμα αυτό έχει ως στόχο την μείωση της διάστασης του μοντέλου και κατ΄ επέκταση τη βελτίωση της απόδοσης του. Η διαδικασία που ακολουθεί το επίπεδο συγκέντρωσης είναι η εξής:

Έστω ότι έχουμε ένα νευρωνικό δίκτυο το οποίο θέλουμε να αναγνωρίζει κάποιο ζώο. Σε κάθε εικόνα παρατηρούμε αρκετές διαφορές στο ζώο ως προς τη στάση, το μέγεθος, την όψη, την κατεύθυνση που κοιτάει, την θέση του στην εικόνα ακόμα και τον φωτισμό. Το νευρωνικό δίκτυο πρέπει να είναι σε θέση να αναγνωρίζει την κατηγορία του ζώου ανεξάρτητα από αυτές τις

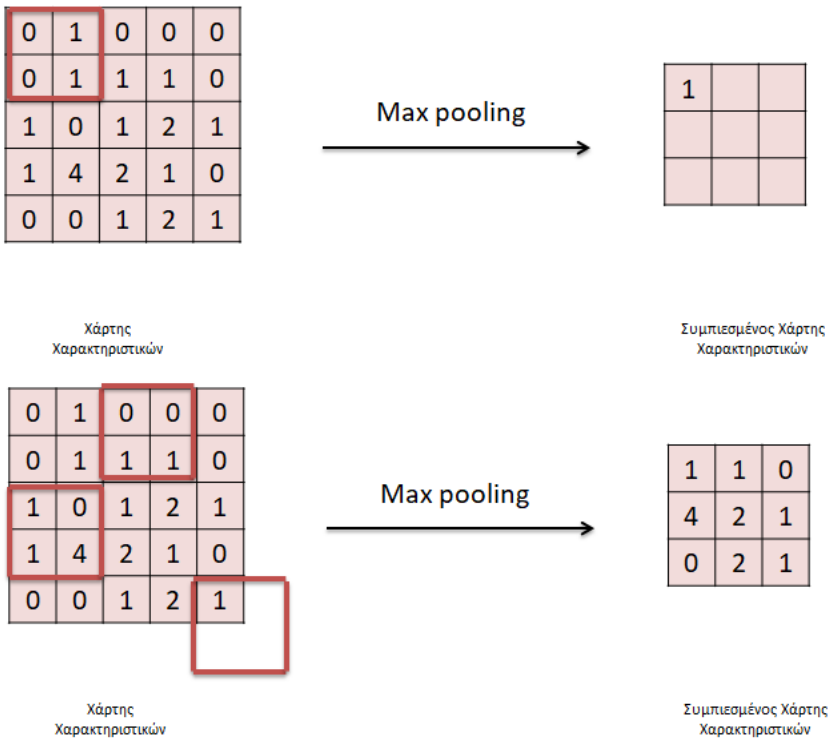
διαφορές, βλέποντας το σύνολο της εικόνας και έχοντας την ευελιξία να βρίσκει τα σημαντικά χαρακτηριστικά.

Για τον λόγο αυτό χρησιμοποιείται η τεχνική pooling. Το pooling εφαρμόζεται στο χάρτη των χαρακτηριστικών που έχουμε από το επίπεδο συνέλιξης. Σε αυτό το επίπεδο, υπάρχουν αρκετές λειτουργίες που μπορούν να εφαρμοστούν. Οι πιο συνηθισμένες είναι η Μέση Συγκέντρωση (Average Pooling) και η Μέγιστη Συγκέντρωση (Max Pooling). Πρακτικά η δεύτερη χρησιμοποιείται περισσότερο και για αυτό το λόγο, στην παρούσα εργασία θα εστιάσουμε σε αυτή.

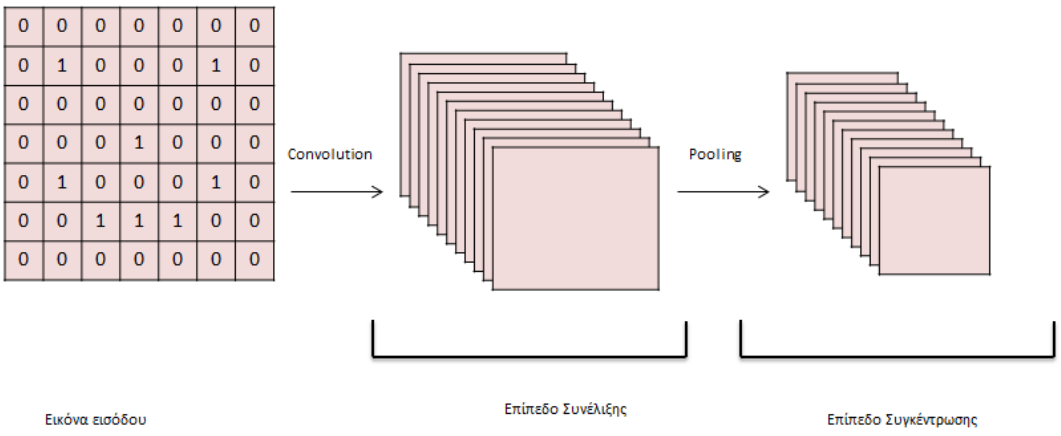
Max pooling :

Για κάθε πλέγμα 2x2 pixel επιλέγουμε το κελί με τη μεγαλύτερη τιμή και το τοποθετούμε στον συμπιεσμένο χάρτη χαρακτηριστικών (pooled feature map) - φίλτρο όπως φαίνεται στην εικόνα. Για κάθε άλμα (stride) κρατείται η μεγαλύτερη τιμή παραμέτρου και οι υπόλοιπες δεν χρησιμοποιούνται.

Με την τεχνική αυτή επιτυγχάνουμε να διατηρήσουμε τα σημαντικά χαρακτηριστικά της εικόνας, όπως είδαμε από το επίπεδο συνέλιξης τα βασικά χαρακτηριστικά είναι εκεί που εντοπίζονται οι μεγαλύτεροι αριθμοί. κρατώντας μόνο το 25% της πληροφορίας και ως αποτέλεσμα έχουμε και τη μείωση της εισόδου των παραμέτρων του νευρωνικού δικτύου. Γίνεται δηλαδή μια διαδικασία δειγματοληψίας του δικτύου.



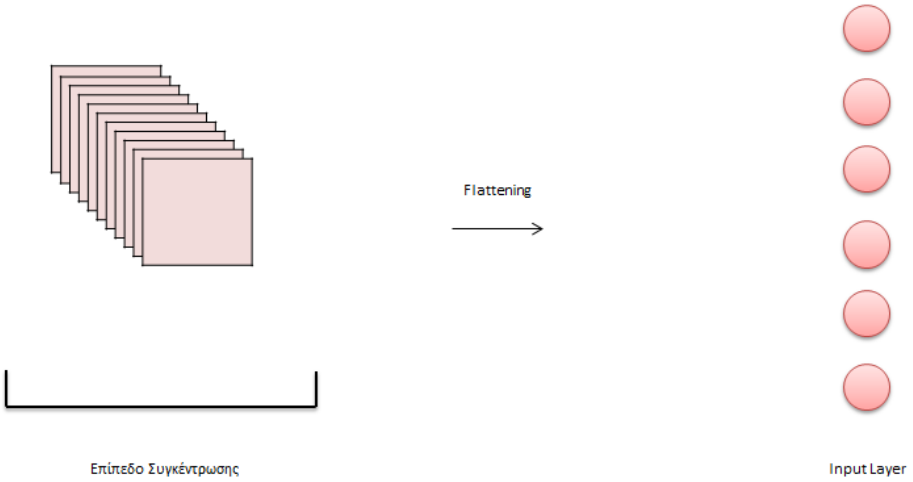
Εικόνα 30: Εφαρμογή τεχνικής Max Pooling



Εικόνα 31: Αποτελέσματα στο επίπεδο συγκέντρωσης

4.2.3 Κανονικοποιηση (Flattening)

Σε αυτό το βήμα θέλουμε να διαμορφώσουμε κατάλληλα τους πίνακες που δημιουργήσαμε στο προηγούμενο επίπεδο ώστε να δώσουμε στο νευρωνικό δίκτυο σαν είσοδο τα δεδομένα σε σωστή μορφή(vector).



Εικόνα 32: Κανονικοποίηση δεδομένων για εισαγωγή στο νευρωνικό

4.2.4 Πλήρως Συνδεδεμενο Επιπεδο

Μετά από αρκετά στρώματα συνέλιξης και συγκέντρωσης, το CNN συνήθως τελειώνει με αρκετά πλήρως συνδεδεμένα στρώματα fully connected layers. Η πολυδιάστατη συστοιχία (tensor) που έχουμε στην έξοδο αυτών των στρωμάτων μετατρέπεται σε διάνυσμα και στη συνέχεια προσθέτουμε αρκετά στρώματα perceptron. Το επίπεδο αυτό έχει ως στόχο την ταξινόμηση της εικόνας εισόδου σε μια από τις κλάσεις που υπάρχουν με βάση το σύνολο δεδομένων που χρησιμοποιείται.

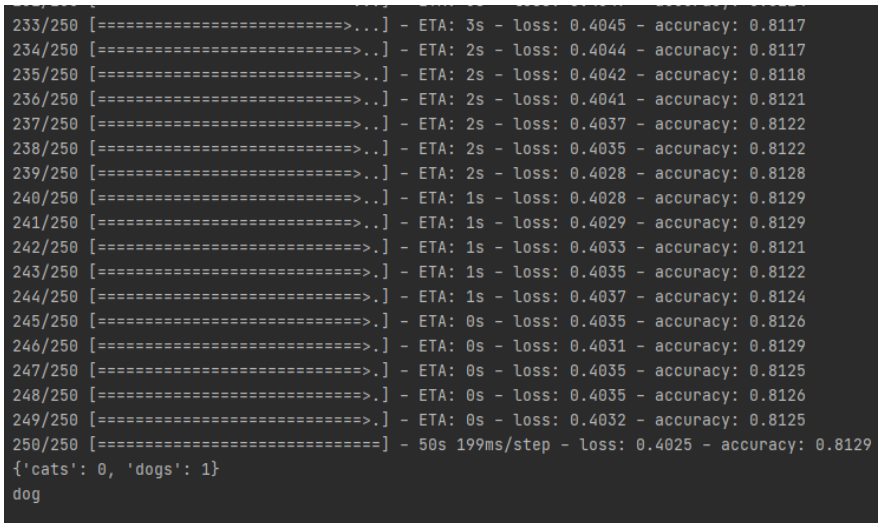
4.2.5 Παραδειγμα

Στο παράδειγμα αυτό θα χτίσουμε και θα εκπαιδεύσουμε ένα συνελικτικό νευρωνικό δίκτυο, το οποίο θα αναγνωρίζει σε μια εικόνα αν περιέχει σκύλο ή γάτα. Το σύνολο δεδομένων που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου περιέχει 4000 εικόνες σκύλων και 4000 εικόνες γατών και για την αξιολόγηση περιέχει 1000 εικόνες σκύλων και 1000 εικόνες γατών.

Εκπαιδεύουμε το μοντέλο μας με την διαδικασία που περιγράψαμε προσαρμόζοντας τις παραμέτρους του αλγορίθμου στο σύνολο δεδομένων μας. Οι παράμετροι αυτές αφορούν την προεπεξεργασία των εικόνων του συνόλου για την βέλτιστη απόδοση του αλγορίθμου, την αρχιτεκτονική του νευρωνικού δικτύου, τον αριθμό των φίλτρων για την εξαγωγή των χαρτών χαρακτηριστικών και των συμπιεσμένων χαρτών, την συνάρτηση ενεργοποίησης που χρησιμοποιείται σε κάθε επίπεδο, τις μεθόδους αξιολόγησης του μοντέλου καθώς και τον αριθμό των φορών που θα εκπαιδεύεται το μοντέλο .

Αφού ολοκληρώσουμε τη διαδικασία της εκπαίδευσης, το μοντέλο μας θα μπορεί να αναγνωρίζει σε μια εικόνα αν το ζώο που απεικονίζεται είναι σκύλος ή γάτα. Δίνοντας, επομένως σαν είσοδο την Εικόνα 33, το μοντέλο μας θα εξάγει το συμπέρασμα πως αυτό που απεικονίζει η εικόνα είναι ένας σκύλος.

Εικόνα 33: Εικόνα εισόδου στο νευρωνικό δίκτυο



Εικόνα 34: Αποτελέσματα μετά την εκπαίδευση

Συνοψίζοντας, η διαδικασία των συνελικτικών νευρωνικών δικτύων ξεκινά με την εικόνα εισόδου, όπου σε αυτή εφαρμόζονται πολλαπλά φίλτρα με σκοπό την ανίχνευση χαρακτηριστικών για την δημιουργία χαρτών χαρακτηριστικών. Αυτό το βήμα γίνεται στο επίπεδο συνέλιξης. Σε αυτό το επίπεδο εφαρμόζεται και η συνάρτηση ReLU με σκοπό την αφαίρεση οποιασδήποτε γραμμικότητας και την αύξηση μη γραμμικότητας στην εικόνα. Στη συνέχεια, το επίπεδο συγκέντρωσης παράγει τον συμπιεσμένο χάρτη χαρακτηριστικών. Κύριος σκοπός αυτού του βήματος είναι να διαβεβαιώσει ότι έχουμε κάποιες χωρικές σταθερές στις εικόνες , ώστε σε περίπτωση που αυτές δεν είναι ιδανικές και ευδιάκριτες, το δίκτυο να μπορεί ακόμα να εντοπίσει τα σημαντικά χαρακτηριστικά. Επίσης, μειώνει σε σημαντικό βαθμό το μέγεθος της εικόνας. Έπειτα, τα pixels από τους πίνακες με τις συμπιεσμένες εικόνες τοποθετούνται σε διανύσματα, τα όποια παίρνει ως είσοδο το νευρωνικό δίκτυο. Τέλος στο τελευταίο βήμα, στο πλήρες συνδεδεμένο νευρωνικό δίκτυο, όλα τα χαρακτηριστικά προχωρούν και αποφασίζεται τελικά σε ποια κλάση ανήκει η εικόνα.

5. Εντοπισμος Αντικειμενων

Ο Εντοπισμός Αντικειμένων είναι ένα από το πιο σημαντικά προβλήματα της Μηχανικής Όρασης. Επιπλέον, η μηχανική όραση είναι ένα πεδίο της τεχνητής νοημοσύνης και σκοπός της είναι η αναπαραγωγή της αίσθησης της όρασης μέσα από αλγόριθμους. Στην ουσία πρόκειται για μια προσπάθεια των μηχανών να μιμούνται το ανθρώπινο μάτι. Ο τυπικός τρόπος για να επιτευχθεί αυτό είναι η εφαρμογή μετασχηματισμών σε μια εικόνα για τον εντοπισμό κάποιων χαρακτηριστικών, τα οποία στην πορεία συγκρίνονται με μια προκαθορισμένη λίστα χαρακτηριστικών που θέλουμε να εντοπίσουμε.

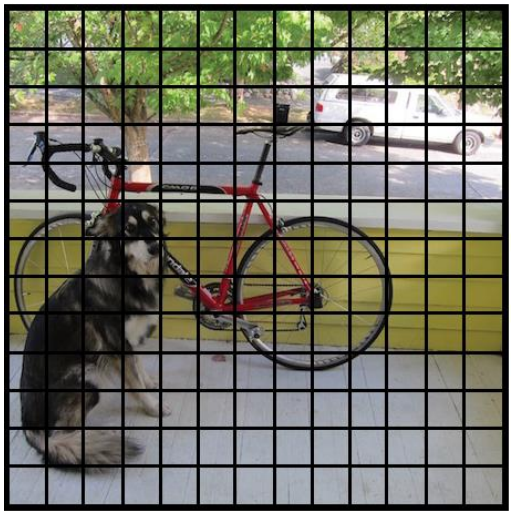
Ο εντοπισμός αντικειμένων είναι μια διαδικασία που συνδέεται με την υπολογιστική όραση και την επεξεργασία της εικόνας. Ο εντοπισμός ενός αντικειμένου προκύπτει έπειτα από μια σειρά ενεργειών. Στην ουσία, η ανίχνευση των αντικειμένων γίνεται μέσω της ταξινόμησης τους. Δεδομένου ότι υπάρχουν συγκεκριμένες κλάσεις αντικειμένων, οι πληροφορίες σχετικά με το είδος του αντικειμένου που παρέχονται βοηθούν στην ταξινόμηση του στην αντίστοιχη κλάση, αφού η κάθε κλάση έχει συγκεκριμένα χαρακτηριστικά.

5.1 Γενικη περιγραφη του Yolo

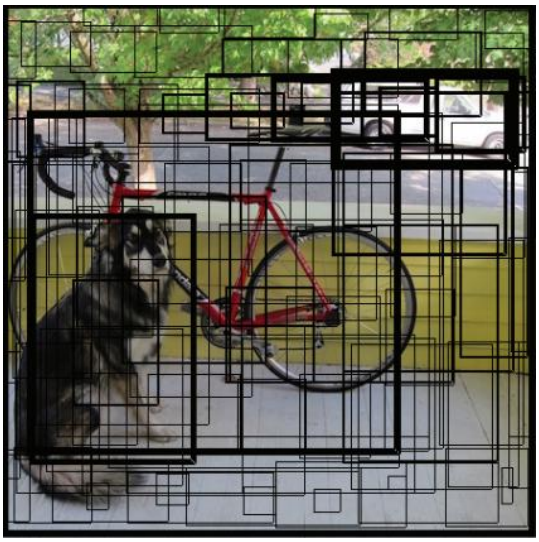
Το YOLO είναι ένας αλγόριθμος για την ανίχνευση πολλαπλών αντικειμένων (multiple object detection) που δημιουργήθηκε από τον Joseph Redmon. Ο αλγόριθμος αυτός ονομάστηκε έτσι από το αρχικά της πρότασης «You Only Look Once». Το όνομα αυτό χαρακτηρίζει και την λειτουργία του. Είναι ένας αλγόριθμος ενός σταδίου που κάνει ταυτόχρονα και την επιλογή περιοχών και την αξιολόγησή τους. Στο μοντέλο αυτό, το δίκτυο δέχεται την εικόνα εισόδου μόνο μια φορά χωρίς να την επεξεργαστεί ξανά κατά την εκτέλεση του τόσο όσον αφορά το στάδιο του εντοπισμού όσο και το στάδιο της εκπαίδευσης. Πρόκειται για ένα σύστημα εντοπισμού αντικειμένων που βασίζεται στα Συνελικτικά Νευρωνικά Δίκτυα (CNN). Ο αλγόριθμος YOLO μπορεί να εντοπίζει πολλαπλά αντικείμενα σε μια εικόνα, δηλαδή μπορεί να προβλέπει την κλάση του αντικειμένου καθώς και να αναγνωρίζει την τοποθεσία του. Αποτελεί μια λύση στο πρόβλημα εντοπισμού αντικειμένων σε πραγματικό χρόνο με βασική έμφαση στην κατά το δυνατόν μεγαλύτερη ακρίβεια αποτελεσμάτων. Το μοντέλο αυτό ενδείκνυται για την ταχύτητα και την ακρίβεια που προσφέρει. Το YOLO διαφοροποιείται από τις άλλες μεθόδους καθώς αντιμετωπίζει το πρόβλημα ως πρόβλημα βελτιστοποίησης. Για τον εντοπισμό και την αναγνώριση του αντικειμένου ο αλγόριθμος χρησιμοποιεί ένα και μοναδικό δίκτυο CNN, το οποίο σκανάρει την εικόνα για να εντοπίσει τα αντικείμενα που έχει εκπαιδευτεί να αναγνωρίζει.

5.1.1 YOLOv3

Το YOLOv3 δημοσιεύτηκε το 2018. Είναι μια βελτιωμένη έκδοση του YOLO και YOLOv2. Το νευρωνικό, για κάθε εικόνα εισόδου, χωρίζει την εικόνα σε ένα πλέγμα κελιών ίσου μεγέθους S x S (Εικόνα 35). Η αναγνώριση της κλάσης του αντικειμένου αλλά και του περιβλήματος (bounding box) του γίνεται την ίδια χρονική στιγμή για κάθε τοποθεσία του πλέγματος. Το περίβλημα είναι ένα παραλληλόγραμμο το οποίο περικλείει το αντικείμενο. Εκτός από το περίβλημα των αντικειμένων προβλέπει και για το κάθε ένα από αυτά ένα σκορ εμπιστοσύνης. Το σκορ εμπιστοσύνης δείχνει την πιθανότητα το περίβλημα να περιέχει ένα αντικείμενο και με πόση ακρίβεια έχει προβλέψει τα όρια του αντικειμένου.



Εικόνα 35: Εικόνα χωρισμένη σε πλέγμα κελιών S x S
(Πηγή: original paper Yolo)

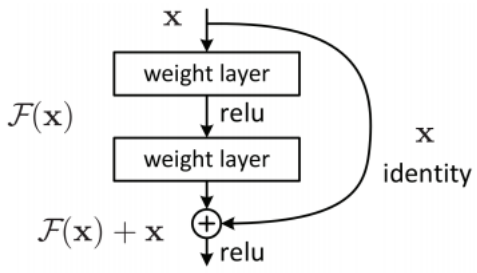


Εικόνα 36: Εικόνα με πιθανές προβλέψεις περιβλημάτων
(Πηγή: original paper Yolo)

Αρχιτεκτονική -Περιγραφή δομής των επιπέδων

Το YOLOv3 χρησιμοποιεί συνελικτικά επίπεδα. Το νευρωνικό δίκτυο φάνηκε ότι όσο πιο βαθιά προχωρούσε, ξεχνούσε κάποια χαρακτηριστικά που είχαν ήδη εκπαιδευτεί στα πρώτα στάδια. Αυτό είχε ως αποτέλεσμα την μείωση της απόδοσης. Για τον λόγο αυτόν, δημιουργήθηκαν τα Residual Networks(ResNes). Η δομή αυτή χρησιμοποιεί την έννοια της ανατροφοδότησης καθώς ανά 2 ή 3 συνελικτικά επίπεδα, το αποτέλεσμα ανατροφοδοτείται με αυτό που έλαβε σαν είσοδο στο πρώτο επίπεδο. Στην τρίτη έκδοση του, το YOLO αποτελείται από 53 συνελικτικά επίπεδα. Η δομή αυτή λέγεται και Darknet-53.Ένα τέτοιο δίκτυο Darknet-53 που αποτελεί το backbone δίκτυο για το YOLOv3.

Για την ανίχνευση αντικειμένων συνδέονται ακόμη 53 επίπεδα κι έτσι παρουσιάζεται μια αρχιτεκτονική 106 επιπέδων. Η συνάρτηση ενεργοποίησης είναι η Leaky ReLU. Δεν υπάρχουν επίπεδα που γίνεται το pooling όπως είδαμε στα CNN αλλά υπάρχουν πρόσθετα συνελικτικά επίπεδα με stride(άλμα) = 2(stride είναι ο αριθμός των pixels που μετακινούνται στον πίνακα εισόδου) για την μείωση του μεγέθους των πινάκων χαρακτηριστικών. Αυτή η τεχνική αποτρέπει την απώλεια των μικρών χαρακτηριστικών και έχει ως αποτέλεσμα την ανίχνευση ακόμη και πολύ μικρών αντικειμένων.



Εικόνα 37: Residual Blocks

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

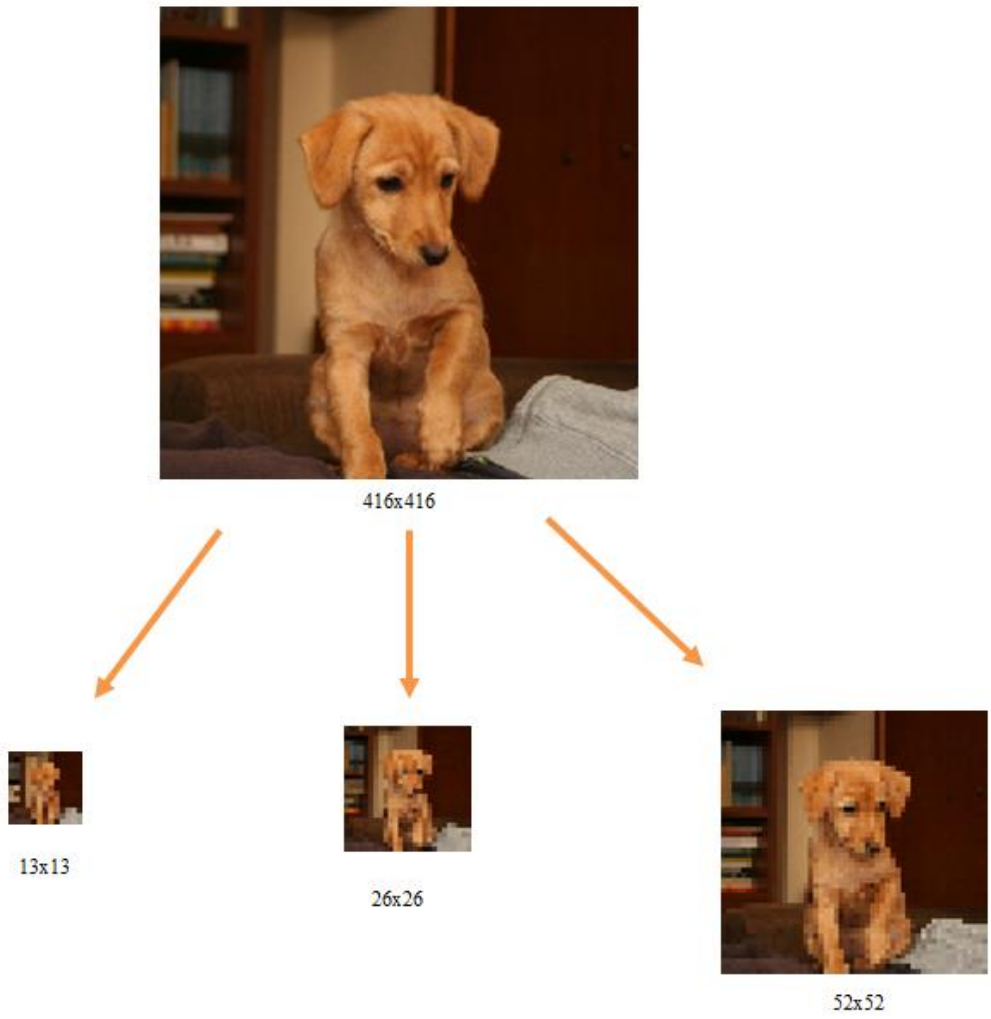
Εικόνα 38: Αρχιτεκτονική Darknet-53

Είσοδος

Η είσοδος που δέχεται το δίκτυο είναι n εικόνες χωρισμένες σε παρτίδες (batches). Δεν υπάρχει προκαθορισμένο μέγεθος εικόνων που δέχεται το μοντέλο. Το YOLO αλλάζει και θέτει το μέγεθος των εικόνων 416 x 416 ή οποιονδήποτε άλλον αριθμό πολλαπλάσιο του 32. Το σχήμα της κάθε παρτίδας είναι (n, 416,416,3), όπου n ο αριθμός των εικόνων , οι επόμενοι αριθμοί δηλώνουν το πλάτος και το ύψος της κάθε εικόνας και ο τελευταίο είναι ο αριθμός των καναλιών χρωμάτων, δηλαδή το RGB. Στην παρούσα εργασία χρησιμοποιείται ο αριθμός 416.

Ανίχνευση αντικειμένων σε τρεις κλίμακες

Το YOLO ολοκληρώνει τον εντοπισμό των αντικειμένων σε 3 διαφορετικές κλίμακες σε 3 διαφορετικά σημεία του δικτύου. Τα σημεία αυτά είναι στα επίπεδα 82, 94 και 106. Σε κάθε επίπεδο η κλίμακα προσδιορίζεται από τον αριθμό των strides. Στο επίπεδο 82 το stride ορίζεται 32, στο 94 ορίζεται 16 και στο 106 ορίζεται 8. Έτσι, αν το μέγεθος της εικόνας έχει οριστεί ως 416 x 416 τότε το μέγεθος των εικόνων που προκύπτουν θα είναι 13x13, 26x26 και 52x52 αντίστοιχα.



Εικόνα 39: Κλίμακες αντικειμένων

Η κάθε κλίμακα εικόνα εξειδικεύεται στον εντοπισμό αντικειμένων διαφορετικού μεγέθους. Στην κλίμακα 13x13 εντοπίζονται αντικείμενα μεγάλου μεγέθους, στην κλίμακα 26x26 μεσαίου μεγέθους και στην κλίμακα 52x52 τα πιο μικρά αντικείμενα.

Για την παραγωγή αποτελέσματος και την μείωση του μεγέθους (down sampling), εφαρμόζονται στα τρία διαφορετικά επίπεδα των ανιχνεύσεων detection kernels μεγέθους 1 x 1 σε κάθε κλίμακα. Έτσι λοιπόν ,εφαρμόζονται 1x1 συνελίξεις (convolutions) με σκοπό να μειωθεί η κλίμακα των εικόνων (down sampling). Ως αποτέλεσμα θα προκύψουν χάρτες χαρακτηριστικών με τις ίδιες διαστάσεις. Η αρχιτεκτονική του kernel προσδιορίζεται από την εξίσωση B * (5 + C) , όπου B ο αριθμός των περιβλημάτων που μπορεί να προβλέψει το κάθε κελί του πίνακα

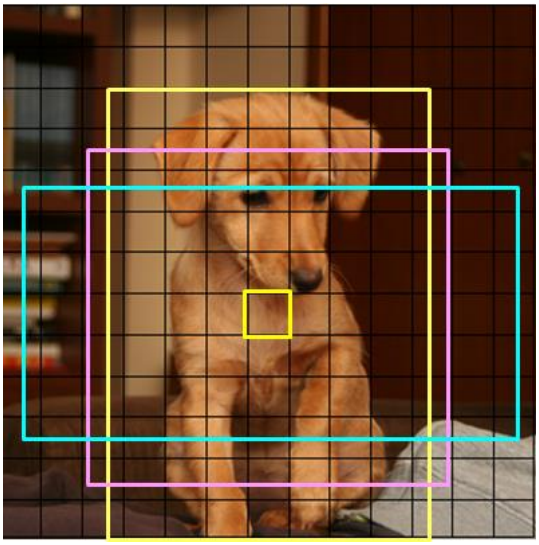
χαρακτηριστικών. Το YOLO προβλέπει 3 περιβλήματα για κάθε κελί. Για κάθε ένα από αυτά υπάρχει μια πληροφορία που το χαρακτηρίζει.

Αυτή η πληροφορία είναι ένας πίνακας με τα εξής 5 γνωρίσματα (x, y, h, w, c)

- (x, y): το κέντρο του bounding box
- w = πλάτος ορθογωνίου / πλάτος εικόνας
- h = ύψος ορθογωνίου / ύψος εικόνας
- c πιθανότητες PrClassi |Object, i = 1, .., C , όπου C ο αριθμός των κλάσεων

Το YOLOv3 εκπαιδεύεται στο dataset COCO το οποίο περιέχει 80 κλάσεις αντικειμένων και θα το εξετάσουμε παρακάτω. Επομένως, στην παραπάνω εξίσωση το B = 3 και το C = 80. Για κάθε bounding box υπάρχουν αθροιστικά 85 γνωρίσματα και συνολικά για όλα τα σημεία του δικτύου 3 x 85 γνωρίσματα. Επομένως η αρχιτεκτονική των χαρτών χαρακτηριστικών που παράγονται από τα kernels 1x1 στα 3 σημεία του δικτύου είναι (13,13,255),(26,26,255) και (52,52,255).

Όπως αναφέραμε υπάρχουν 3 περιβλήματα για κάθε κελί του χάρτη χαρακτηριστικών. Κάθε κελί προβλέπει ένα αντικείμενο για κάθε περίβλημα που σχηματίζει και το περίβλημα είναι υπεύθυνο για την πρόβλεψη του αντικειμένου, υπολογίζοντας τόσο τη θέση του όσο και την διάσταση του. Το YOLO αναγνωρίζει το κελί που βρίσκεται στο κέντρο του αντικειμένου. Στην πράξη, το κελί αυτό είναι που κάνει την τελική πρόβλεψη του αντικειμένου.



Εικόνα 40: Περιβλήματα για το κεντρικό κελί της εικόνας

Στην παραπάνω εικόνα το κελί στο οποίο βρίσκεται το κέντρο του αντικειμένου είναι το κίτρινο. Κατά την εκπαίδευση του YOLOv3 υπάρχει ένα πραγματικό περίβλημα (ground truth bounding box) που είναι υπεύθυνο για τον εντοπισμό ενός αντικειμένου και στο οποίο πρέπει να προσδιοριστούν τα κελιά που εμπεριέχονται σε αυτό.

Anchor Boxes

Για την πρόβλεψη του πραγματικού περιβλήματος, το YOLO χρησιμοποιεί κάποια προκαθορισμένα περιβλήματα που ονομάζονται anchor boxes. Στην τρίτη έκδοση του YOLO χρησιμοποιούνται συνολικά 9 anchor boxes, 3 για κάθε κλίμακα. Ο υπολογισμός τους γίνεται με την βοήθεια του αλγορίθμου συσταδοποίησης k-means που μελετήθηκε παραπάνω.

Objectness score και class confidence

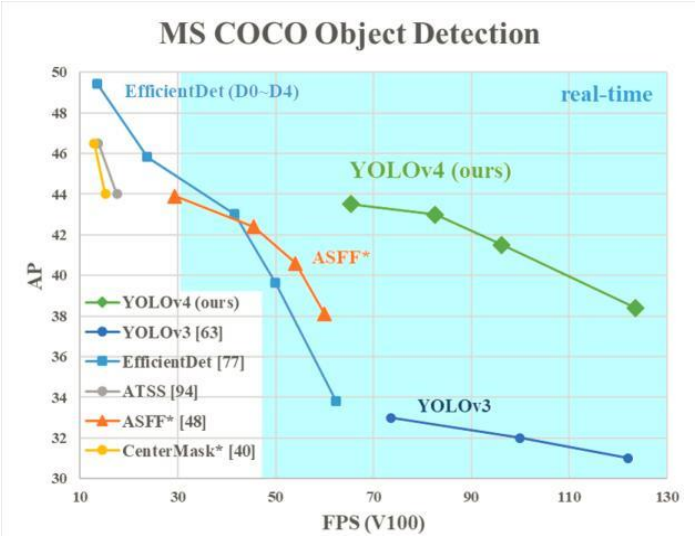
Δυο σημαντικά μεγέθη που παίζουν πολύ σημαντικό ρόλο στην τελική ανίχνευση του αντικειμένου είναι το «objectness» σκορ και το «class confidence». Το Objectness σκορ ορίζεται ως η πιθανότητα ένα συγκεκριμένο κελί να αποτελεί το κεντρικό κελί που είναι υπεύθυνο για την πρόβλεψη ενός αντικειμένου και του αντίστοιχου bounding box που το περικλείει. Το μέγεθος class confidence δηλώνει τις πιθανότητες να ανήκει το συγκεκριμένο αντικείμενο στην κάθε μια κλάση. Εάν δεν υπάρχει αντικείμενο σε αυτό το κελί θα πρέπει το σκορ να είναι μηδέν. Επομένως το objectness score μελετά αν το bounding box περιέχει αντικείμενο ενώ το class confidence ποια είναι η πιθανότητα να ανήκει σε συγκεκριμένη κλάση.

Συνοψίζοντας, λοιπόν, τα βασικά σημεία του αλγορίθμου είναι τα εξής:

- χωρίζει την εικόνα σε πλέγμα SxS
- το κελί που περιέχει το κέντρο του αντικειμένου είναι υπεύθυνο για την ανίχνευσή του
- κάθε κελί προβλέπει τα περιβλήματα και το σκορ εμπιστοσύνης τους
- σκορ εμπιστοσύνης αντιπροσωπεύει την βεβαιότητα ότι το κουτί βλέπει το πραγματικό αντικείμενο
- κάθε περίβλημα ακολουθείται από 5 γνωρίσματα
- ο υπολογισμός της κλάσης εμπιστοσύνης αντιπροσωπεύει την τομή της ένωσης (IOU) μεταξύ του προβλεπόμενου περιβλήματος και του πραγματικού.

5.1.2 YOLOv4

Όπως μελετήσαμε παραπάνω το YOLOv3 είναι ένας πολύ δημοφιλής αλγόριθμος και αποτελεσματικός. Μια νέα έκδοση του YOLO, η τέταρτη που δημοσιεύθηκε το 2020, φαίνεται να δείχνει μια σημαντική βελτίωση ως προς την ταχύτητα και την απόδοση σε σχέση με την Τρίτη έκδοση. Συγκεκριμένα, η μέση απόδοση βελτιώνεται κατά 10% και κατά 12% ο αριθμός των FPS. Η τέταρτη έκδοση αναπτύχθηκε από τους Alexey Bochkovskiy, Chien-Yao Wang και Hong-Yuan Mark Liao και όχι από τους αρχικούς δημιουργούς του YOLO. Ο στόχος του YOLOv4 είναι ένας ανιχνευτής αντικειμένων με γρήγορη λειτουργικότητα και αρκετά βελτιωμένος για τον παράλληλο υπολογισμό.



Εικόνα 41: Σύγκριση απόδοσης διάφορων detector σε σχέση με το YOLOv4 (Πηγή: original paper YOLOv4)

Το μοντέλο ανίχνευσης αποτελείται από πολλές συνιστώσες. Αυτές είναι:

- Input : για την εισαγωγή εικόνων
- Backbone: αφορά το δίκτυο που επεξεργάζεται την εικόνα και δημιουργεί τον χάρτη χαρακτηριστικών. Τα δίκτυα μπορεί να είναι VGG16, ResNet50, Darknet53, ResNext50 . Στη συνιστώσα αυτή γίνεται κυρίως η εξαγωγή των χαρακτηριστικών
- Neck και Head : είναι μικρότερα σύνολα του backbone που βοηθούν στην πιο αποδοτική ικανότητα να διακρίνει τα χαρακτηριστικά και για τον χειρισμό των προβλέψεων. Το Neck εξυπηρετεί κυρίως στην ένωση των χαρακτηριστικών.

Το head block είναι για την χωροθέτηση των περιβλημάτων και την κατηγοριοποίηση του περιεχομένου τους. Στην τέταρτη έκδοση του YOLO χρησιμοποιείται η ίδια διαδικασία με αυτή της τρίτης . Το δίκτυο ανιχνεύει τις συντεταγμένες του περιβλήματος και το σκορ εμπιστοσύνης για κάθε κλάση. Η τεχνική της πρόβλεψης βασίζεται στα anchor boxes που αναφέραμε παραπάνω.

Επιπλέον, η τελευταία έκδοση καθιστά ευκολότερη την εκπαίδευση του νευρωνικού δικτύου σε μια μόνο GPU και χρησιμοποιεί συγκεκριμένες τεχνικές που βοήθησαν στην ενίσχυση της ακρίβειας του μοντέλου τόσο στην διαδικασία της εκπαίδευσης όσο και στην περίοδο μετά την επεξεργασία. Αυτές οι τεχνικές χωρίζονται σε:

- Bag of Frebies (BoF) : Οι τεχνικές BoF εφαρμόζονται στο backbone. Πρόκειται για ένα σύνολο τεχνικών που βοηθάει στην εκπαίδευση χωρίς να προσθέτει χρόνο στην διαδικασία εξαγωγής των αποτελεσμάτων
- Bag of Specials (BoS) : Οι τεχνικές BoS αποτελούν ένα άλλο είδος τεχνικών. Σε αντίθεση με τις BoF, αλλάζουν την αρχιτεκτονική του δικτύου και κάποιες φορές αυξάνουν το κόστος της διαδικασία εξαγωγής των αποτελεσμάτων.

Στο YOLOv4, μετά από τις δοκιμές και πειραματική επαλήθευση του CSPResNeXt50 και του CSPDarknet53 σαν δομή δικτύου για το backbone, επιλέχθηκε το δίκτυο CSPDarkNet53 ως καταλληλότερο για το μοντέλο ανίχνευσης.

5.1.3 TINY-YOLO

Ο αλγόριθμος tiny-YOLO είναι μια πιο ελαφριά, συμπιεσμένη έκδοση του YOLO. Η έκδοση αυτή βασίζεται πάνω στην ολοκληρωμένη έκδοση. Απλοποιώντας την δομή του δικτύου και μειώνοντας κάποιες παραμέτρους γίνεται εφικτό να υλοποιηθεί σε συσκευές με μικρότερη υπολογιστική ισχύ. Το βασικό του πλεονέκτημα είναι η ταχύτητα του. Το tiny-YOLO είναι μικρότερο σε μέγεθος από την κανονική έκδοση του YOLO κάνοντάς το γρηγορότερο σε σημαντικό βαθμό. Είναι λογικό να σκεφτεί κανείς πως αυτή η αύξηση στην ταχύτητα που προσφέρει θα έχει κάποιο αντίτιμο. Πράγματι, το tiny-YOLO προσφέρει μικρότερη ακρίβεια στις προβλέψεις του για την αναγνώριση των αντικειμένων σε σχέση με το YOLO . Ωστόσο, φαίνεται πως η τελευταία έκδοση του tiny-YOLO έχει μειώσει κατά πολύ αυτή την απόκλιση καθιστώντας τον αλγόριθμο tiny-YOLOv4 κατάλληλο για ακριβείς και γρήγορους υπολογισμούς σε πραγματικό χρόνο ακόμα και από ενσωματωμένες συσκευές.

Το tiny-YOLO αποτελείται από πολύ λιγότερα επίπεδα συνέλιξης και επίπεδα συγκέντρωσης, γεγονός που το κάνει να απαιτεί συνολικά πολύ λιγότερα layers από το YOLOv3 και YOLOv4 αντίστοιχα. Η βασική διαφορά τους είναι ότι το YOLO είναι σχεδιασμένο για να ανιχνεύει σε 3 διαφορετικές κλίμακες αντικειμένων, ενώ το tiny-YOLO ανιχνεύει σε 2 κλίμακες. Γενικότερα, εκτός αυτής της διαφοροποίησης, ο τρόπος λειτουργίας τους είναι ο ίδιος.

5.2 Υλοποιηση ανιχνευσης αντικειμενων

5.2.1 Περιβαλλον - Γλωσσα - βιβλιοθηκες

Python

Ο αλγόριθμος υλοποιήθηκε σε γλώσσα προγραμματισμού Python. Η Python είναι από τις πιο δημοφιλής γλώσσες υψηλού επιπέδου στη Μηχανική Μάθηση καθώς είναι ευέλικτη γλώσσα , υποστηρίζει διάφορες χρήσεις και ένα μεγάλο σύνολο βιβλιοθηκών.

Anaconda

Το πρώτο βήμα είναι η δημιουργία ενός περιβάλλοντος κατάλληλου για την υλοποίηση αλγορίθμων Μηχανικής Μάθησης. Το Conda είναι ένα εργαλείο που βοηθάει την σωστή διαχείριση των εξαρτήσεων(dependencies) της Python δημιουργώντας εικονικά περιβάλλοντα (virtual environments). Το εικονικό περιβάλλον δίνει τη δυνατότητα να δουλεύουν ταυτόχρονα και αποδοτικά διαφορετικά πρότζεκτ της Python με χρήση διαφορετικών πακέτων και διαφορετικών εκδόσεων. Το Miniconda , το οποίο χρησιμοποιείται για τις ανάγκες της παρούσας διπλωματικής, αποτελεί μια μικρότερη έκδοση του Anaconda, μια πιο εκτενής έκδοση με πολλά στοιχεία (components) που δεν είναι απαραίτητα. Με το Miniconda , λοιπόν, δίνεται η δυνατότητα εικονικού περιβάλλοντος που μπορεί να φιλοξενεί ταυτόχρονα και άλλα framework Μηχανικής Μάθησης, όπως tensorflow και keras.

OpenCV

Το OpenCv είναι μια βιβλιοθήκη με διάφορες προγραμματιστικές διαδικασίες και αλγόριθμους μηχανικής μάθησης με κύριο στόχο την υλοποίηση υπολογιστικής όρασης σε πραγματικό χρόνο. Αναπτύχθηκε στο κέντρο ερευνών της Intel στο Νίζνι Νοβγκόροντ της Ρωσίας το 1999. Είναι μια βιβλιοθήκη πολλαπλών πλατφορμών γραμμένη σε C/C++, ανοιχτού κώδικα δωρεάν στην χρήση. Υποστηρίζει μεγάλο εύρος γλωσσών προγραμματισμού, όπως Python, Java, MATLAB κλπ. και περιέχει πάνω από 500 συναρτήσεις με υποστήριξη αλγορίθμων για τεχνητή όραση και επεξεργασία εικόνας. Οι αλγόριθμοι αυτοί βοηθούν στον εντοπισμό αντικειμένων, την αναγνώριση προσώπων, την παρακολούθηση κινήσεων και κινούμενων αντικειμένων, ταξινόμηση ανθρώπινης κίνησης, σε εφαρμογές επαυξημένης πραγματικότητας και πολλά άλλα.

Βιβλιοθήκες που χρησιμοποιήθηκαν για τα παραδείγματα :

- Tensorflow: είναι μια μαθηματική βιβλιοθήκη νευρωνικών δικτύων ανοιχτού κώδικα ανεπτυγμένη από την ομάδα της Google Brain. Με την χρήση του tensorflow καλύπτεται η ανάγκη δημιουργίας ενός νευρωνικού δικτύου από την αρχή. Επομένως, εφόσον υπάρχει η βάση του δικτύου, καθιστά ευκολότερη την εκπαίδευση των custom data.
- NumPy: είναι μια πολύ δημοφιλής βιβλιοθήκη της python η οποία υποστηρίζει την επεξεργασία πινάκων πολλών διαστάσεων με τη βοήθεια μιας μεγάλης συλλογής μαθηματικών συναρτήσεων υψηλού επιπέδου.
- Scikit-learn: είναι μια πολύ χρήσιμη βιβλιοθήκη για τους κλασικούς αλγορίθμους μηχανικής μάθησης. Υποστηρίζει τους περισσότερους από τους αλγόριθμους επιβλεπόμενης και μη επιβλεπόμενης μάθησης.
- Pandas είναι μια βιβλιοθήκη γνωστή για την ανάλυση δεδομένων. Χρησιμοποιείται κυρίως για την προετοιμασία των δεδομένων πριν από την εκπαίδευσή τους.

- Matpoltlib: βιβλιοθήκη για την αναπαράσταση των δεδομένων. Αναπαριστά δεδομένα σε γράφους και διαγράμματα δύο διαστάσεων-2D.

5.2.2 Αλγοριθμος

Στην προηγούμενη ενότητα μελετήθηκε ο τρόπος λειτουργίας του αλγορίθμου YOLOv3 και YOLOv4. Σε αυτό το κεφάλαιο θα υλοποιηθεί ο αλγόριθμος. Οι εκδόσεις που χρησιμοποιούμε είναι η τρίτη και η τέταρτη, με ικανοποιητική αύξηση ταχύτητας εκπαίδευσης, αναγνώρισης και ακρίβειας. Θα παρουσιάσουμε τα αποτελέσματα και στις δύο περιπτώσεις . Ο αλγόριθμος είναι ο ίδιος και για τις δύο εκδόσεις και αλλάζουν μόνο τα βάρη και τα αρχεία διαμόρφωσης (.cfg).

Τα αρχεία που χρειάζονται για την εφαρμογή του αλγορίθμου για κάθε έκδοση είναι :

coco.names: αρχείο με το ονόματα των 80 κλάσεων του συνόλου δεδομένων (Εικόνα 42).

yolo.cfg: περιέχουν τις ρυθμίσεις των παραμέτρων για τις δύο αυτές διαδικασίες (Εικόνα 43).

- batches : ο αριθμός των εικόνων που επεξεργάζονται σε κάθε επανάληψη
- max_batches : ο συνολικός αριθμός των επαναλήψεων
- learning rate : η παράμετρος εκμάθησης
- angle : τυχαία γωνία περιστροφής εικόνων
- saturation : τυχαίος κορεσμός χρωμάτων εικόνας
- exposure : τυχαία έκθεση χρωμάτων εικόνας
- hue : τυχαία απόχρωση χρωμάτων εικόνας

Δομή των συνελικτικών νευρωνικών δικτύων που περιέχει :

- αριθμό φίλτρων
- τον τύπο της συνάρτησης ενεργοποίησης
- το μέγεθος των φίλτρων
- το stride(άλμα)

Τέλος, περιέχει τα τρία τελευταία επίπεδα YOLO που περιγράφουν την αρχιτεκτονική του.

- αριθμός κλάσεων
- αριθμός φίλτρων – υπολογίζεται από τον τύπο (classes + coordinates + 1) * mask: ο αριθμός των καναλιών (RGB)

yolo.weights: το αρχείο αυτό περιέχει όλα τα εκπαιδευμένα βάρη που προκύπτουν από την εκπαίδευση του αλγορίθμου.

Αρχιτεκτονική YOLO (Εικόνα 44):

- Το YOLOv3 έχει 107 layers, και τα εκπαιδευμένα βάρη του είναι 234.9 MB.
- Το YOLOv3-tiny έχει 24 layers, και τα εκπαιδευμένα βάρη του είναι 33.1 MB
- Το YOLOv4 έχει 162 layers, και τα εκπαιδευμένα βάρη του είναι 245.7 MB.
- Το YOLOv4-tiny έχει 38 layers, και τα εκπαιδευμένα βάρη του είναι 23.1 MB.

```
1  person        21 elephant      41 wine glass    61 diningtable
2  bicycle       22 bear          42 cup           62 toilet
3  car           23 zebra         43 fork          63 tvmonitor
4  motorbike     24 giraffe       44 knife         64 laptop
5  aeroplane     25 backpack      45 spoon         65 mouse
6  bus           26 umbrella      46 bowl          66 remote
7  train         27 handbag       47 banana        67 keyboard
8  truck         28 tie           48 apple         68 cell phone
9  boat          29 suitcase      49 sandwich      69 microwave
10 traffic light 30 frisbee       50 orange        70 oven
11 fire hydrant  31 skis          51 broccoli      71 toaster
12 stop sign     32 snowboard     52 carrot        72 sink
13 parking meter 33 sports ball   53 hot dog       73 refrigerator
14 bench         34 kite          54 pizza         74 book
15 bird          35 baseball bat  55 donut         75 clock
16 cat           36 baseball glove 56 cake         76 vase
17 dog           37 skateboard    57 chair         77 scissors
18 horse         38 surfboard     58 sofa          78 teddy bear
19 sheep         39 tennis racket 59 pottedplant   79 hair drier
20 cow           40 bottle        60 bed           80 toothbrush
```

Εικόνα 42: Αρχείο coco.names

```
[net]
# Testing
# batch=1
# subdivisions=1
# Training
batch=64
subdivisions=16
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky
```

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=2
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=16
size=3
stride=1
pad=1
activation=leaky
```

```
[net]
batch=64
subdivisions=8
# Training
#width=512
#height=512
width=608
height=608
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.0013
burn_in=1000
max_batches = 500500
policy=steps
steps=400000,450000
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 8

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=mish
```

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=1
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000

max_batches = 2000200
policy=steps
steps=1600000,1800000
scales=.1,.1


#weights_reject_freq=1001
#ema_alpha=0.9998
#equidistant_point=1000
#num_sigmas_reject_badlabe
#badlabels_rejection_perce

[convolutional]
batch_normalize=1
filters=32
size=3
stride=2
pad=1
activation=leaky
```

Εικόνα 43: αρχεία .cfg για YOLOv3, YOLOv3-tiny, YOLOv4 και YOLOv4-tiny

```
[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

```
[yolo]
mask = 0,1,2
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=80
num=6
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

```
[yolo]
mask = 6,7,8
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
scale_x_y = 1.05
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5
```

```
[yolo]
mask = 1,2,3
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=80
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
resize=1.5
nms_kind=greedynms
beta_nms=0.6
```

Εικόνα 44: Αρχιτεκτονική του YOLOv3, YOLOv3-tiny, YOLOv4 και YOLOv4-tiny

Θα μελετήσουμε τρεις περιπτώσεις για την ανίχνευση αντικειμένων με το YOLOv3 και το YOLOv4:

1. Ανίχνευση αντικειμένων σε εικόνα.
2. Ανίχνευση αντικειμένων σε βίντεο.
3. Ανίχνευση αντικειμένων σε βίντεο πραγματικού χρόνου

Όταν θέλουμε να ανιχνεύσουμε αντικείμενα σε μια απλή εικόνα χρειάζεται να διαβάσουμε την εικόνα να την δώσουμε στον αλγόριθμο σαν είσοδο και να πάρουμε την επεξεργασμένη εικόνα. Στην περίπτωση του βίντεο, πρέπει να διαβάσουμε κάθε frame ξεχωριστά, να το δώσουμε ως είσοδο στον αλγόριθμο, να πάρουμε το αποτέλεσμα από αυτό το frame και να επαναλάβουμε την διαδικασία για κάθε ένα frame μέχρι το τέλος του βίντεο. Παράλληλα, όλα τα frames που έχουν ήδη επεξεργαστεί συγκεντρώνονται και αποθηκεύονται. Στο τέλος το αποτέλεσμα είναι ένα νέο βίντεο με τα ανιχνευμένα αντικείμενα. Σχεδόν η ίδια διαδικασία συμβαίνει και με το βίντεο πραγματικού χρόνου. Η διαφορά είναι πως σε αυτή την περίπτωση διαβάζουμε το κάθε frame απευθείας από την κάμερα και όχι από κάποιο αρχείο. Η υπόλοιπη διαδικασία παραμένει ίδια.

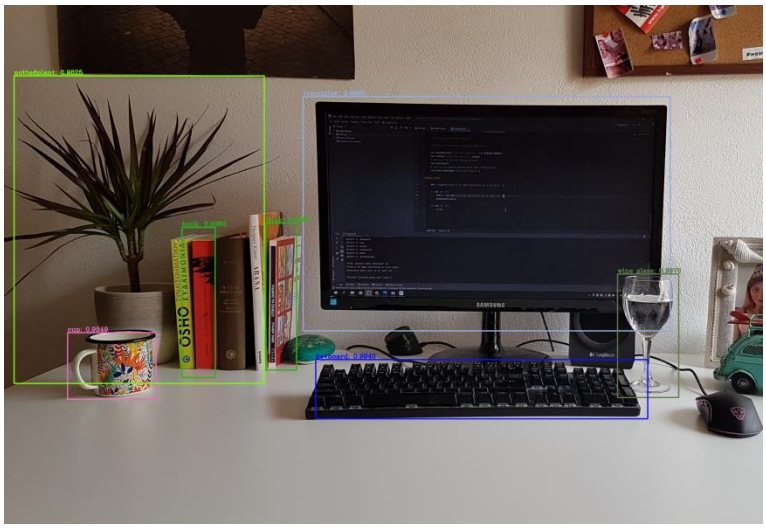
Ο αλγόριθμος στα περισσότερα σημεία παραμένει ίδιος και αλλάζει στο σημείο που δέχεται τα διαφορετικά input, καθώς και στον τρόπο επεξεργασίας τους. Σε αυτό το στάδιο της εργασίας το dataset που χρησιμοποιούμε για την σύγκριση των δύο εκδόσεων θα είναι το αρχικό dataset του YOLO το COCO.

- Εικόνα

Βήματα Αλγορίθμου:

1. Διάβασμα της RGB εικόνας και αποθήκευση των διαστάσεων της.
2. Μετατροπή της εικόνας σε blob (Binary Large Object: συλλογή από δυαδικά δεδομένα και αποθηκευμένα ως μια οντότητα) που θα χρησιμοποιηθεί ως είσοδος στο δίκτυο.
3. Φόρτωση του δικτύου YOLOv3 ή YOLOv4. Σε αυτό το βήμα θέτουμε δύο σημαντικές παραμέτρους. Η μια είναι η ελάχιστη πιθανότητα (minimum probability) και η άλλη το κατώφλι (threshold). Η πρώτη θέτει την ελάχιστη πιθανότητα που πρέπει να έχει μια πρόβλεψη. Στην περίπτωση που είναι μικρότερη από αυτή που έχει τεθεί το αντικείμενο αφαιρείται από την λίστα των ανιχνεύσεων. Η δεύτερη είναι η ελάχιστη τιμή για το φιλτράρισμα των περιβλημάτων σε κάθε αντικείμενο.
4. Forward pass. Το δίκτυο λαμβάνει ως είσοδο την εικόνα σε μορφή blob και γίνεται η διαδικασία προώθησης διάδοσης που γίνεται στα νευρωνικά δίκτυα.
5. Bounding Boxes. Αφού ληφθεί το αποτέλεσμα από το προηγούμενο βήμα, συγκεντρώνουμε όλα τα αντικείμενα που ανιχνεύτηκαν, τα αντίστοιχα περιβλήματα αυτών και τα σκορ εμπιστοσύνης. Στο στάδιο αυτό γίνεται και η αφαίρεση των αδύναμων προβλέψεων με βάση τον αριθμό ελάχιστης πιθανότητας.
6. Non-maximum suppression. Πολλές φορές για κάθε ένα από τα αντικείμενα που ανιχνεύονται δημιουργούνται περισσότερα από ένα επικαλυπτόμενα περιβλήματα. Για την επιλογή του καταλληλότερου περιβλήματος χρησιμοποιείται η τεχνική Non-maximum suppression. Η τεχνική αυτή φιλτράρει τα περιβλήματα που δεν είναι αναγκαία με την βοήθεια των δύο παραμέτρων που τέθηκαν στο βήμα 3.
7. Σχεδιασμός των περιβλημάτων με τις αντίστοιχες ετικέτες.

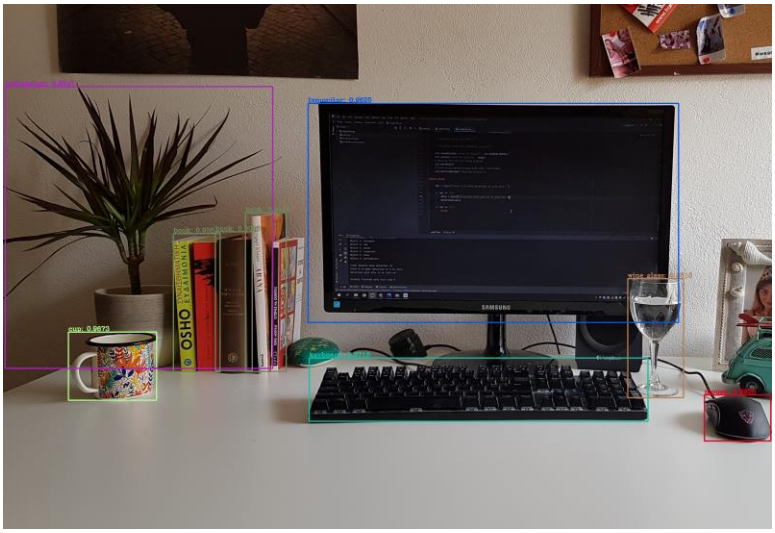
Το τελικό αποτέλεσμα θα είναι μια εικόνα σαν την αρχική με τα αντικείμενα που ανιχνεύτηκαν, τα περιβλήματά τους και την ετικέτα.



```
Detection with yolo v3 or yolo v4? v3
The Detection took 0.59370 seconds
Object 1: tvmonitor
with confidence 0.9995
Object 2: wine glass
with confidence 0.9976
Object 3: keyboard
with confidence 0.9940
Object 4: cup
with confidence 0.9849
Object 5: pottedplant
with confidence 0.9825
Object 6: book
with confidence 0.8857
Object 7: book
with confidence 0.8362

Total objects been detected: 15
```

Εικόνα 45: Αποτελέσματα ανιχνεύσεων με YOLOv3



```
Detection with yolo v3 or yolo v4? v4
The Detection took 0.64878 seconds
Object 1: mouse
with confidence 0.9925
Object 2: wine glass
with confidence 0.9818
Object 3: keyboard
with confidence 0.9718
Object 4: cup
with confidence 0.9673
Object 5: tvmonitor
with confidence 0.9625
Object 6: pottedplant
with confidence 0.9547
Object 7: book
with confidence 0.9351
Object 8: book
with confidence 0.9093
Object 9: book
with confidence 0.8406

Total objects been detected: 32
```

Εικόνα 46: Αποτελέσματα ανιχνεύσεων με YOLOv4

- Βίντεο

Σε αυτή την περίπτωση, η ανίχνευση αντικειμένων γίνεται σε ένα βίντεο που είτε έχει οριστεί εξ αρχής είτε βιντεοσκοπείται εκείνη τη στιγμή. Σαν είσοδο , λοιπόν ο αλγόριθμος θα δέχεται ένα βίντεο. Ένα βίντεο αποτελείται από πολλά στιγμιότυπα. Επομένως θα χρησιμοποιηθεί ο ίδιος αλγόριθμος με την διαφορά ότι όλη η διαδικασία θα επαναλαμβάνεται για κάθε ένα στιγμιότυπο ξεχωριστά μέχρι να ολοκληρωθεί το βίντεο. Το κάθε στιγμιότυπο για το οποίο θα εφαρμόζεται ο αλγόριθμος θα αποθηκεύεται σε ένα νέο βίντεο που θα δημιουργείται κατά την εκτέλεσή του. Τα βήματα που ακολουθούνται είναι τα εξής:

1. Διάβασμα του βίντεο ή λήψη νέου βίντεο.

2. Φόρτωση του δικτύου YOLOv3 ή YOLOv4 (ίδια διαδικασία με την περίπτωση απλής εικόνας).

Διάβασμα κάθε στιγμιότυπου επαναληπτικά και εφαρμογή των βημάτων:

3. Μετατροπή του στιγμιότυπου σε blob.
4. Forward pass.
5. Bounding Boxes.
6. Non-maximum suppression.
7. Σχεδιασμός των περιβλημάτων με τις αντίστοιχες ετικέτες.
8. Εγγραφή των επεξεργασμένων στιγμιότυπων σε νέο βίντεο.

Το τελικό αποτέλεσμα θα είναι ένα νέο βίντεο σαν το αρχικό με τα αντικείμενα που ανιχνεύτηκαν, τα περιβλήματά τους και την ετικέτα.

- Real Time

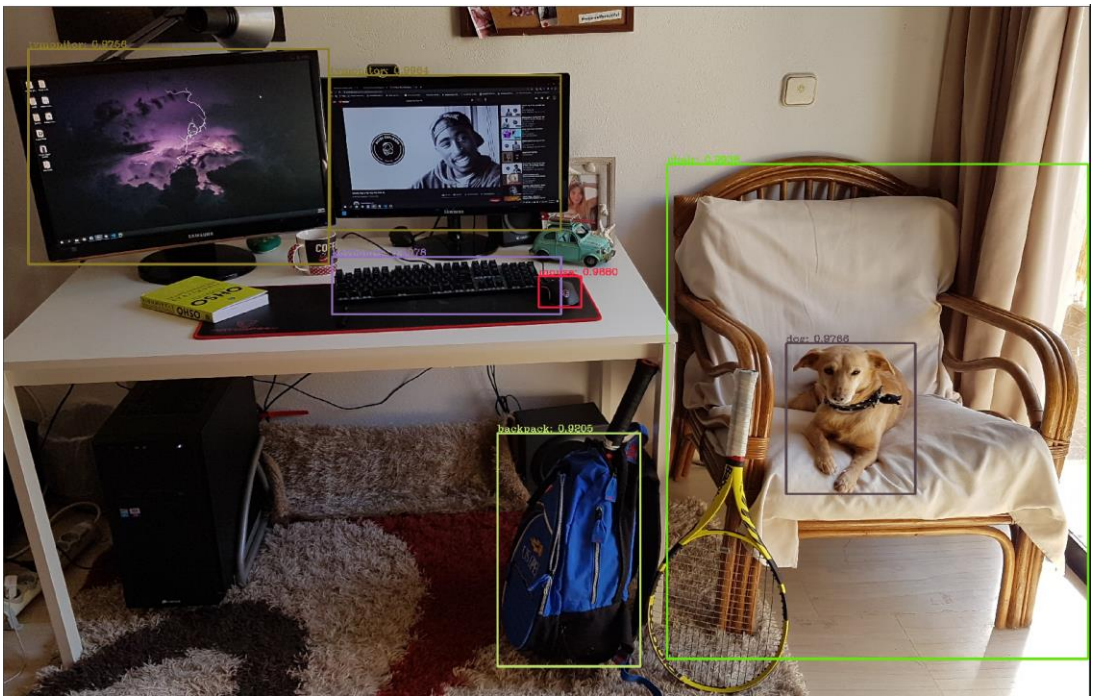
Η τελευταία περίπτωση που θα μελετήσουμε και θα μας απασχολήσει και στην επόμενη ενότητα είναι η ανίχνευση αντικειμένων σε βίντεο πραγματικού χρόνου. Μια κάμερα συνδεδεμένη στο μηχάνημα μας παράγει στιγμιότυπα σε πραγματικό χρόνο με τα οποία τροφοδοτείται ο αλγόριθμος . Στον ίδιο χρόνο τα στιγμιότυπα αυτά φαίνονται επεξεργασμένα με σχεδιασμένα τα περιβλήματα, τις ετικέτες και το σκορ εμπιστοσύνης του κάθε αντικειμένου που ανιχνεύεται. Οι διαφορές στον αλγόριθμό σε σχέση με την προηγούμενη περίπτωση είναι στο βήμα 1 όπου αντί να διαβάσει κάποιο βίντεο, λαμβάνεται ως είσοδος ένα stream βίντεο και στο βήμα 8 όπου το βίντεο δεν εγγράφεται αλλά προβάλλεται σε πραγματικό χρόνο την ίδια χρονική στιγμή.

Αποτελέσματα σε διάφορες εικόνες και στιγμιότυπα από βίντεο και σύγκριση αυτών:

Στην πρώτη περίπτωση που η είσοδος στον αλγόριθμο είναι μια εικόνα, παρατηρούμε πως οι εκδόσεις YOLOv3 και YOLOv4 έχουν μεγάλη ακρίβεια στις προβλέψεις τους και υπάρχει πολύ μικρή διαφορά στα τελικά αποτελέσματα. Από την άλλη πλευρά, φαίνεται πως τα αποτελέσματα της τέταρτης έκδοσης του tiny-YOLO έχουν μεγαλύτερη ακρίβεια βάση της πιθανότητας να είναι ορθή η πρόβλεψη και μπορεί να εντοπίζει περισσότερα αντικείμενα σε όλες τις περιπτώσεις.

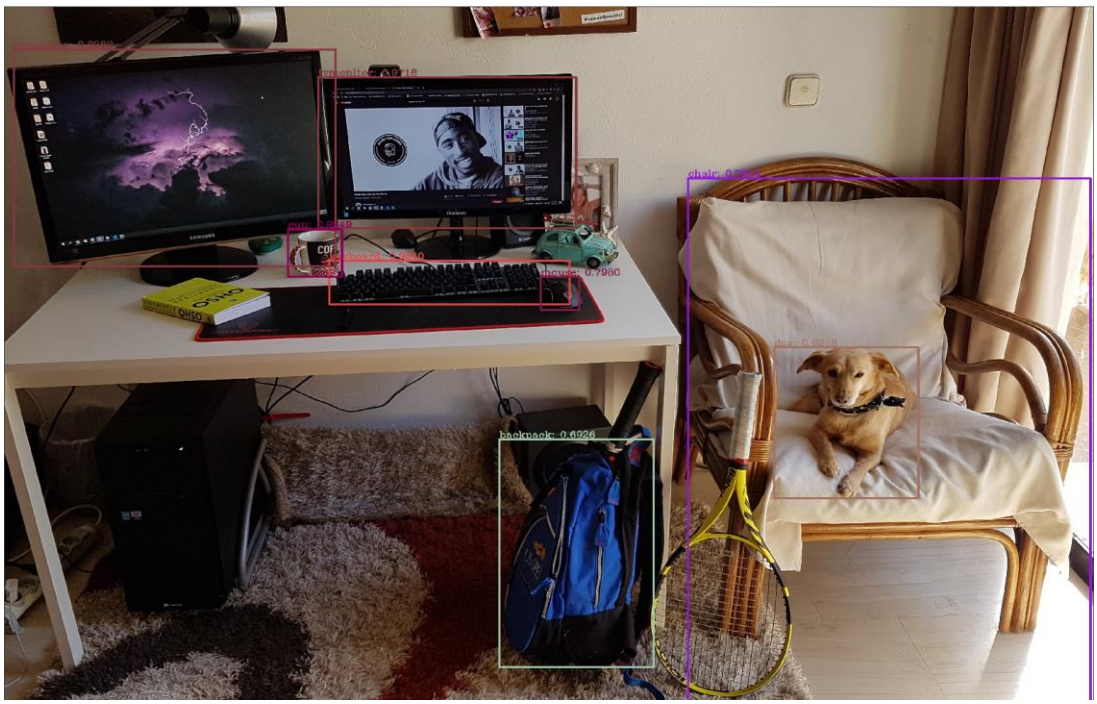
Input: Image Desktop	YOLOv3	YOLOv4	Tiny-YOLOv3	Tiny-YOLOv4
Accuracy	High	High	Low	Low
Objects Detected	7	8	3	4

YOLOv3:



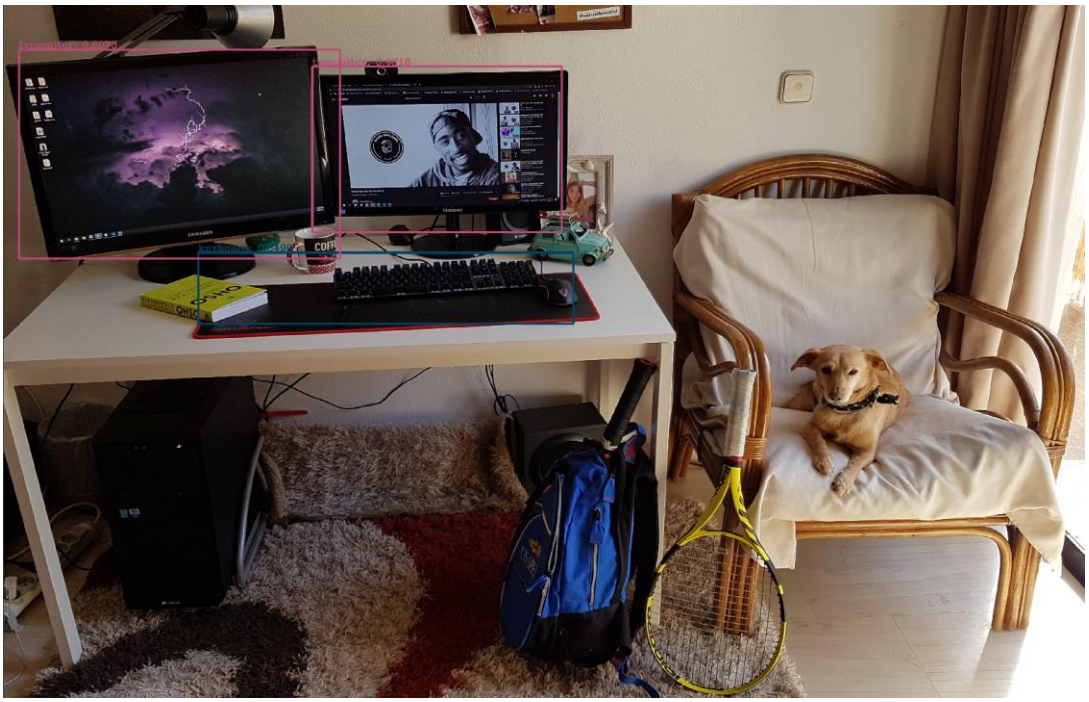
Εικόνα 47: Αποτελέσματα σε εικόνα YOLOv3

YOLOv4:



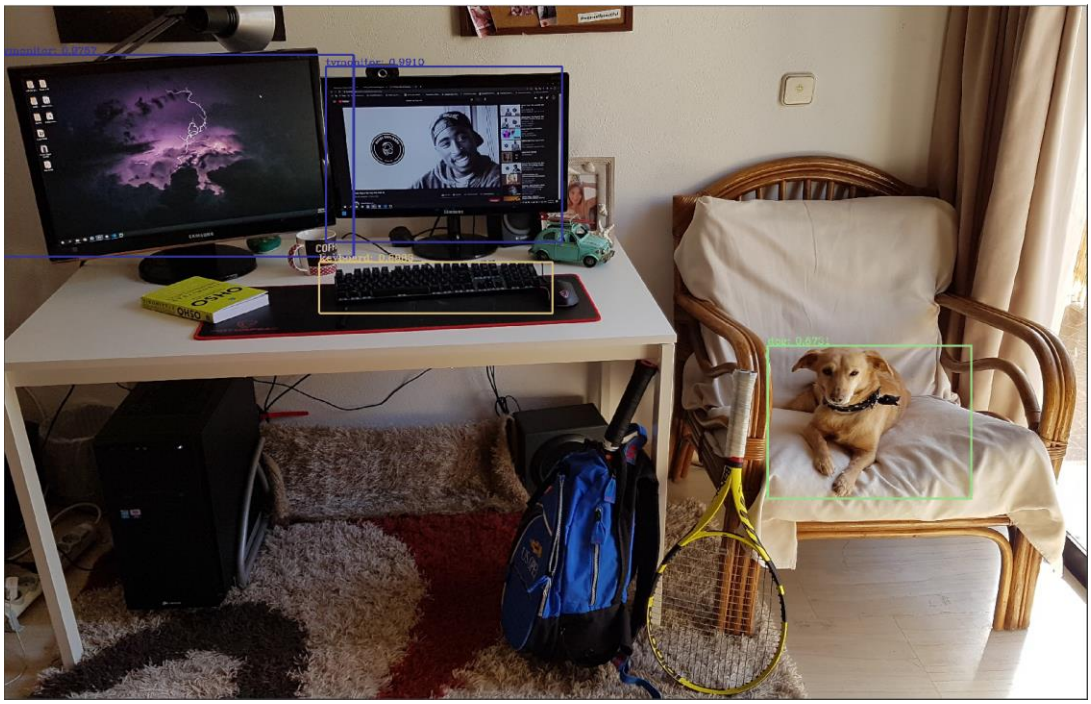
Εικόνα 48: Αποτελέσματα σε εικόνα YOLOv4

Tiny-YOLOv3:



Εικόνα 49: Αποτελέσματα σε εικόνα tiny-YOLOv3

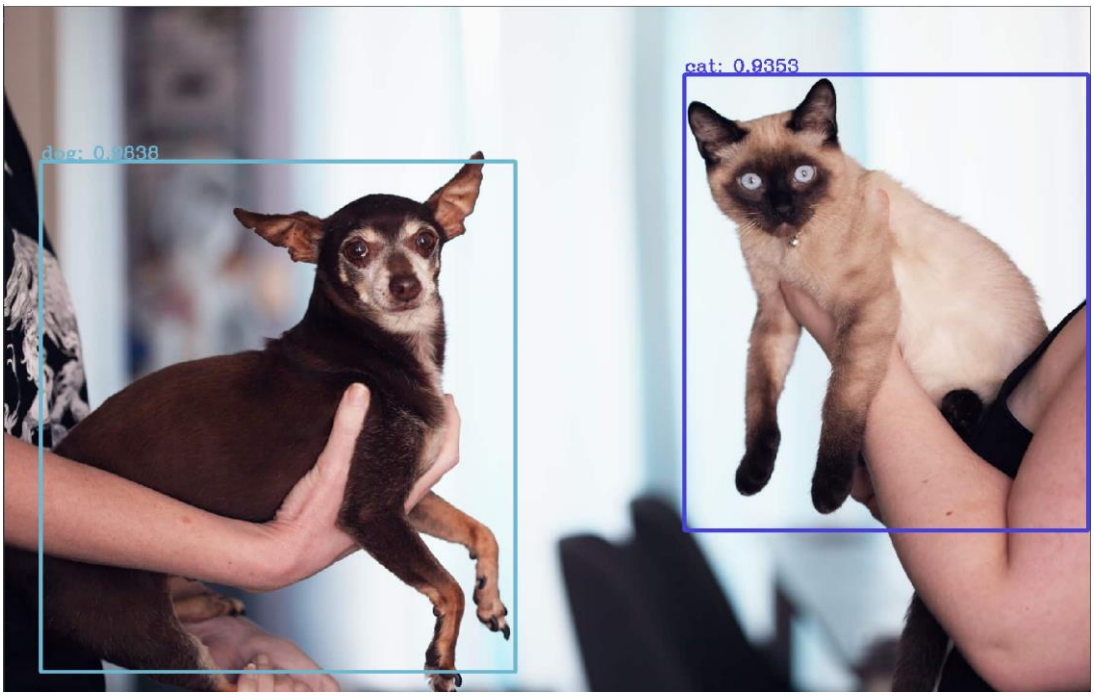
Tiny- YOLOv4:



Εικόνα 50: Αποτελέσματα σε εικόνα tiny-YOLOv4

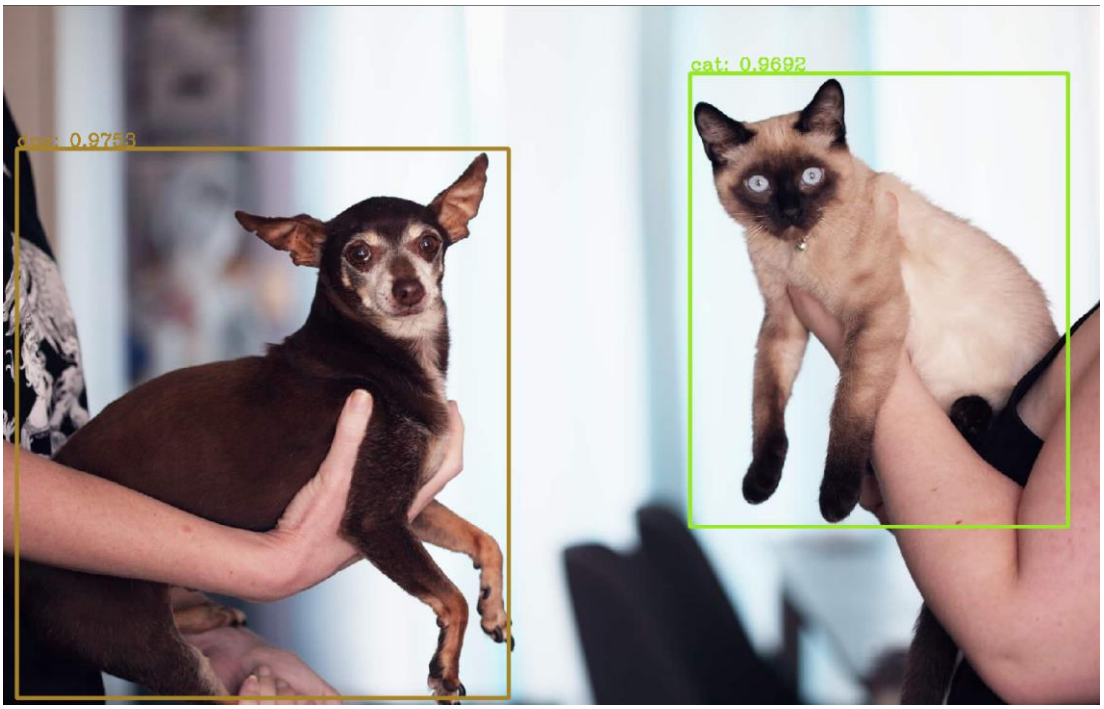
Input: Image Dog & Cat	YOLOv3	YOLOv4	Tiny-YOLOv3	Tiny-YOLOv4
Accuracy	High	High	Low	Low
Objects Detected	2	2	1	2

YOLOv3:



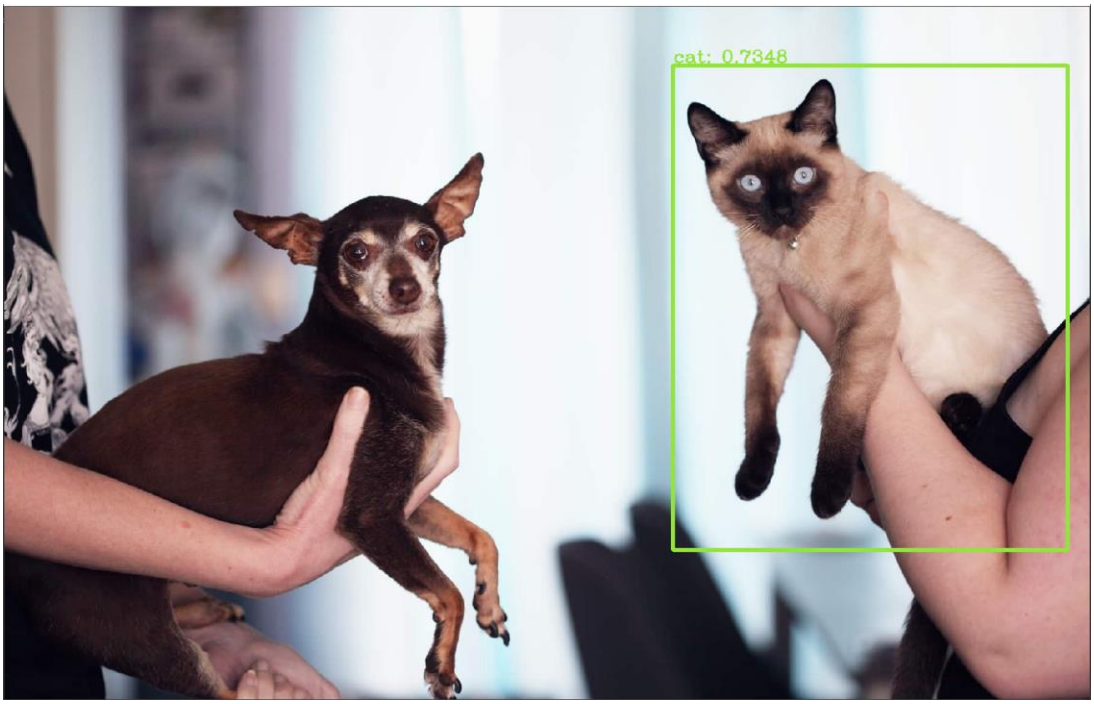
Εικόνα 51: Αποτελέσματα σε εικόνα YOLOv3

YOLOv4:



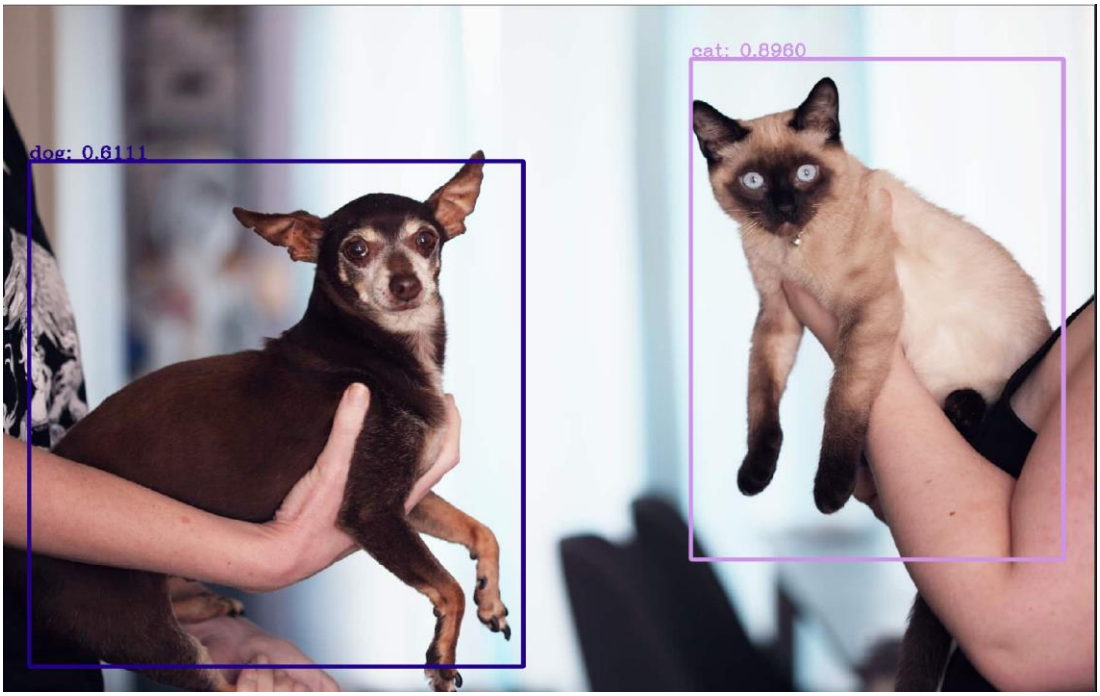
Εικόνα 52: Αποτελέσματα σε εικόνα YOLOv4

Tiny-YOLOv3:



Εικόνα 53: Αποτελέσματα σε εικόνα tiny-YOLOv3

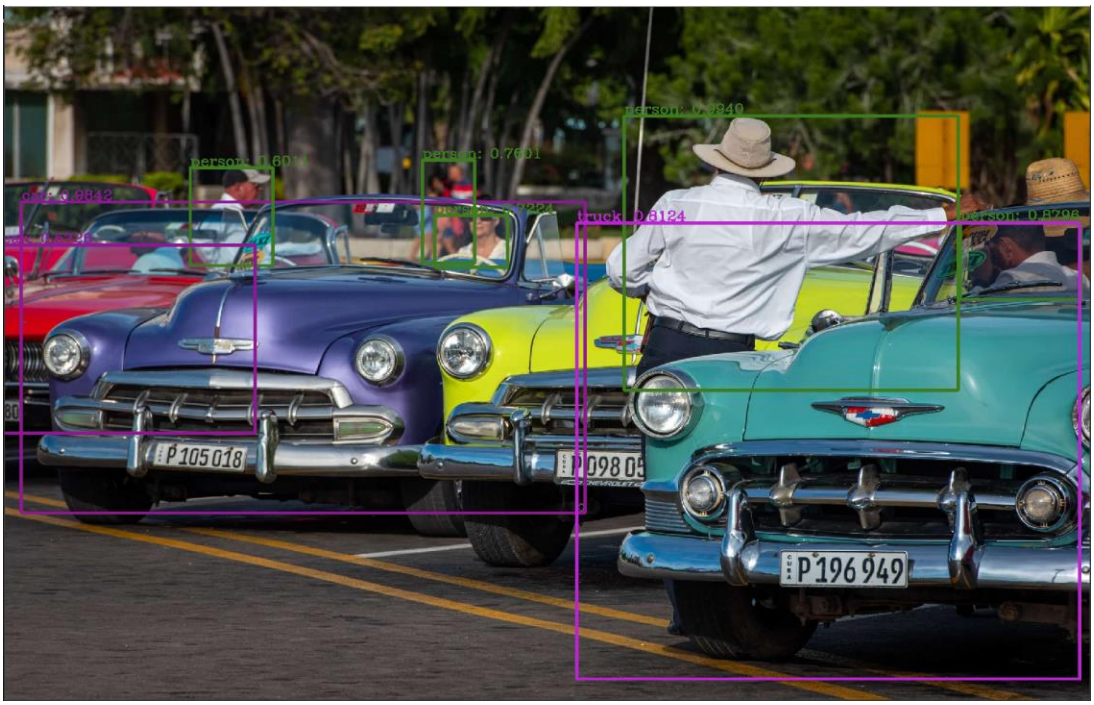
Tiny-YOLOv4:



Εικόνα 54: Αποτελέσματα σε εικόνα tiny-yolov4

Input: Image Cars & People	YOLOv3	YOLOv4	Tiny-YOLOv3	Tiny-YOLOv4
Accuracy	High	High	Low	Low
Objects Detected	8	7	2	8

YOLOv3:



Εικόνα 55: Αποτελέσματα σε εικόνα YOLOv3

YOLOv4:

Εικόνα 56: Αποτελέσματα σε εικόνα YOLOv4

Tiny-YOLOv3:

Εικόνα 57: Αποτελέσματα σε εικόνα tiny-YOLOv3

Tiny-YOLOv4:

Εικόνα 58: Αποτελέσματα σε εικόνα tiny-YOLOv4

Στην δεύτερη περίπτωση που η είσοδος είναι βίντεο, θα συγκρίνουμε ίδια στιγμιότυπα από το βίντεο που χρησιμοποιήθηκε . Σε αυτή την κατηγορία παρατηρούμε πόσο μεγάλη είναι η διαφορά στον χρόνο που χρειάζεται για την ανίχνευση στην ολόκληρη έκδοση του YOLO σε σχέση με την tiny-YOLO αρχιτεκτονική.

Input: Video Traffic Frames No.: 373	YOLOv3	YOLOv4	Tiny-YOLOv3	Tiny-YOLOv4
Accuracy	High	High	Low	Low
Time	1.5 FPS	1.3 FPS	18.4 FPS	16.5 FPS
Speed	Slow	Slow	Fast	Fast

YOLOv3:

Εικόνα 59: Αποτελέσματα σε video YOLOv3

YOLOv4:

Εικόνα 60: Αποτελέσματα σε video YOLOv4

Tiny-YOLOv3:

Εικόνα 61: Αποτελέσματα σε video tiny-YOLOv3

Tiny-YOLOv4:

Εικόνα 62: Αποτελέσματα σε video tiny-YOLOv4

Input: Video **House** **Frames No.: 284**	**YOLOv3**	**YOLOv4**	**Tiny-YOLOv3**	**Tiny-YOLOv4**
Accuracy	High	High	Low	Low
Time	1.6 FPS	1.3 FPS	22.6 FPS	19.4 FPS
Speed	Slow	Slow	Fast	Fast

YOLOv3:

Εικόνα 63: Αποτελέσματα σε video YOLOv3

YOLOv4:

Εικόνα 64: Αποτελέσματα σε video YOLOv4

Tiny-YOLOv3:

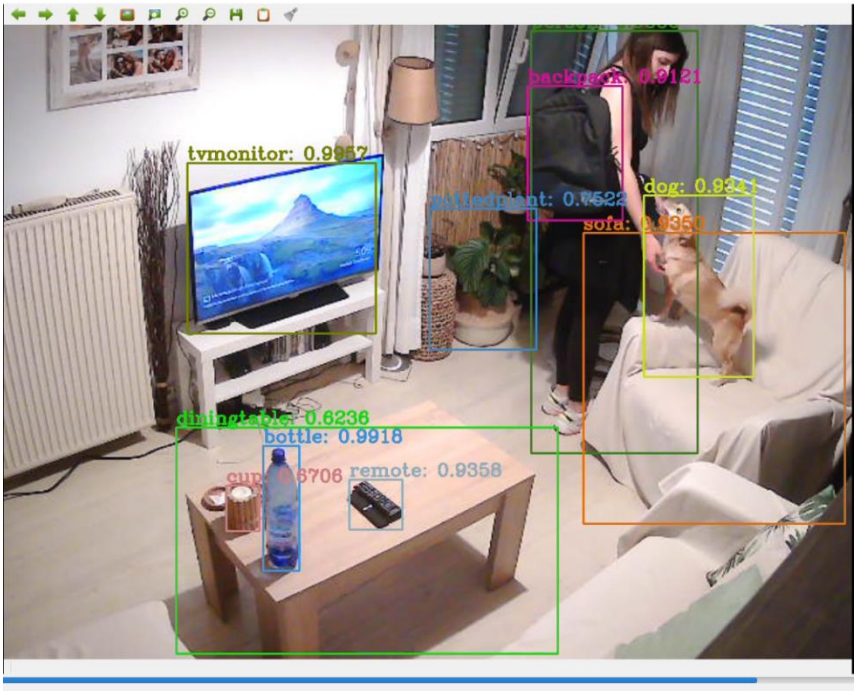
Εικόνα 65: Αποτελέσματα σε video tiny-YOLOv3

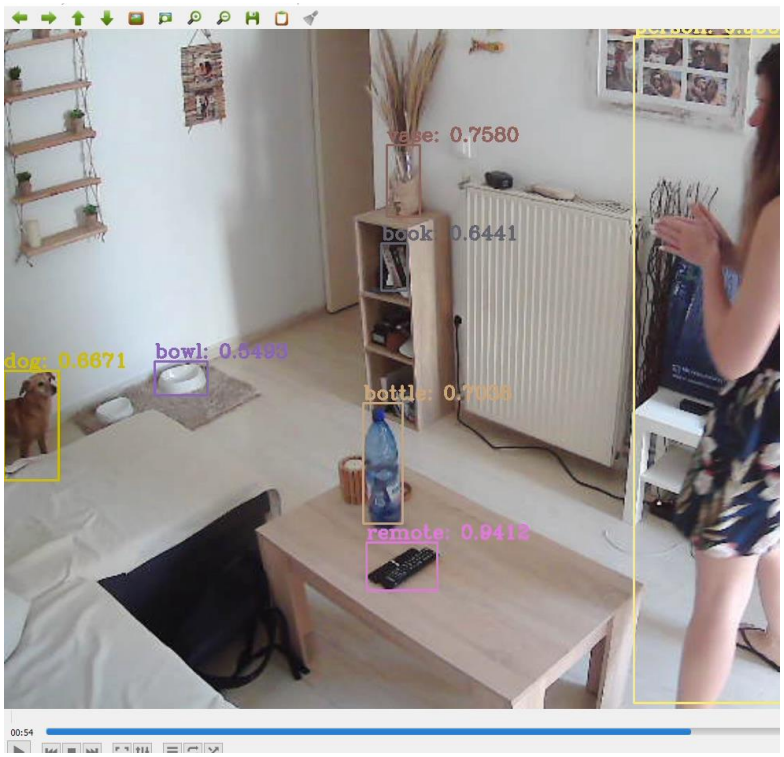
Tiny-YOLOv4:

Εικόνα 66: Αποτελέσματα σε video tiny-YOLOv4

Τελευταία περίπτωση είναι αυτή της ανίχνευσης αντικειμένων σε πραγματικό χρόνο. Για προφανείς λόγους τα στιγμιότυπα προς σύγκριση δεν μπορούν να είναι ακριβώς τα ίδια.

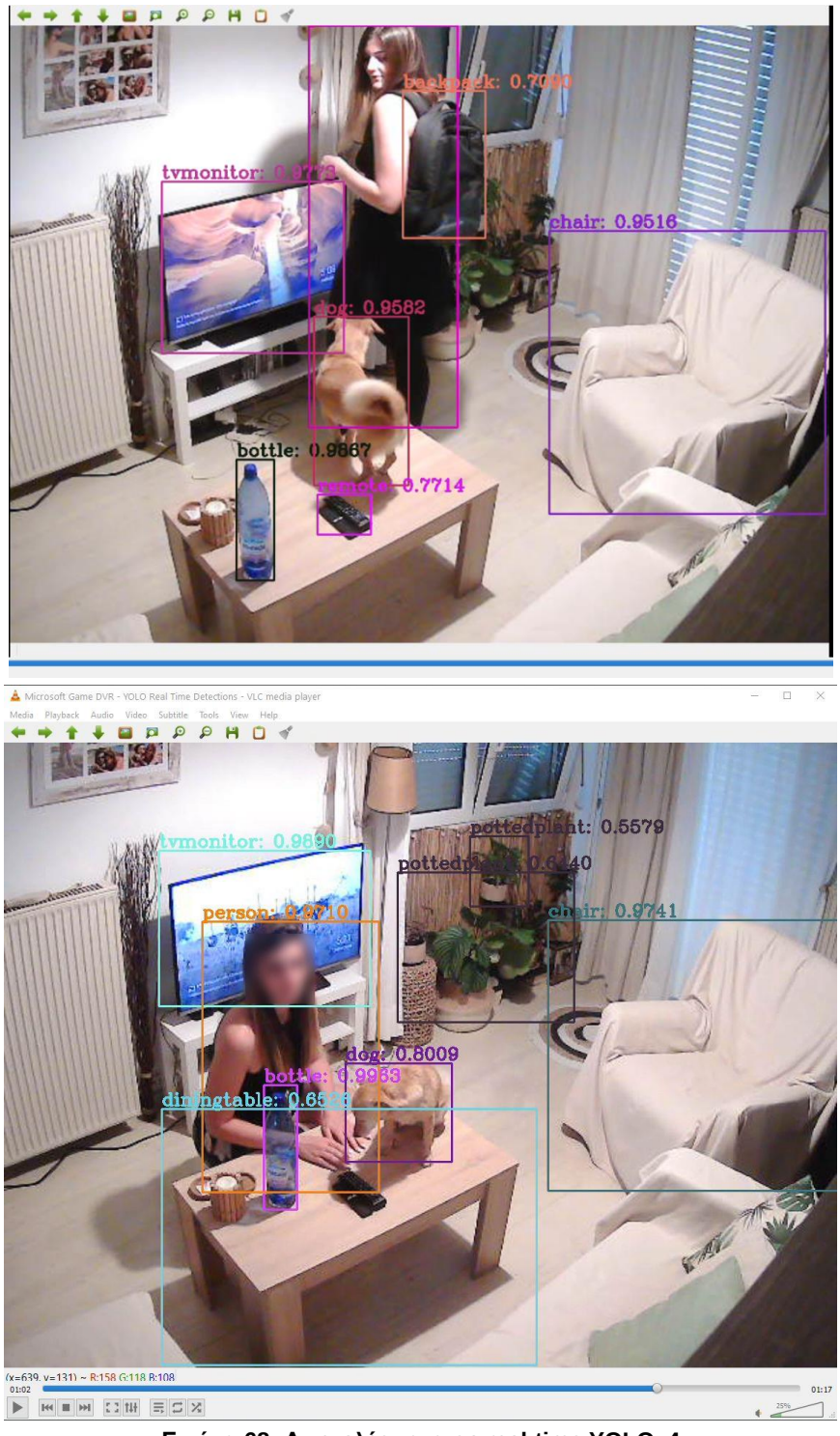
YOLOv3:





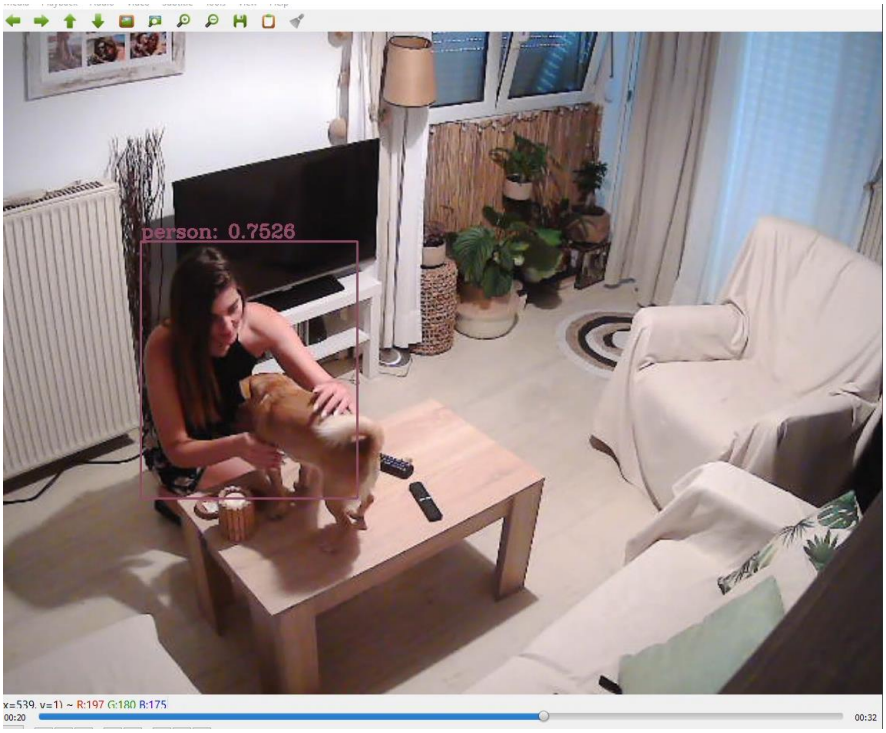
Εικόνα 67:Αποτελέσματα σε real-time video YOLOv3

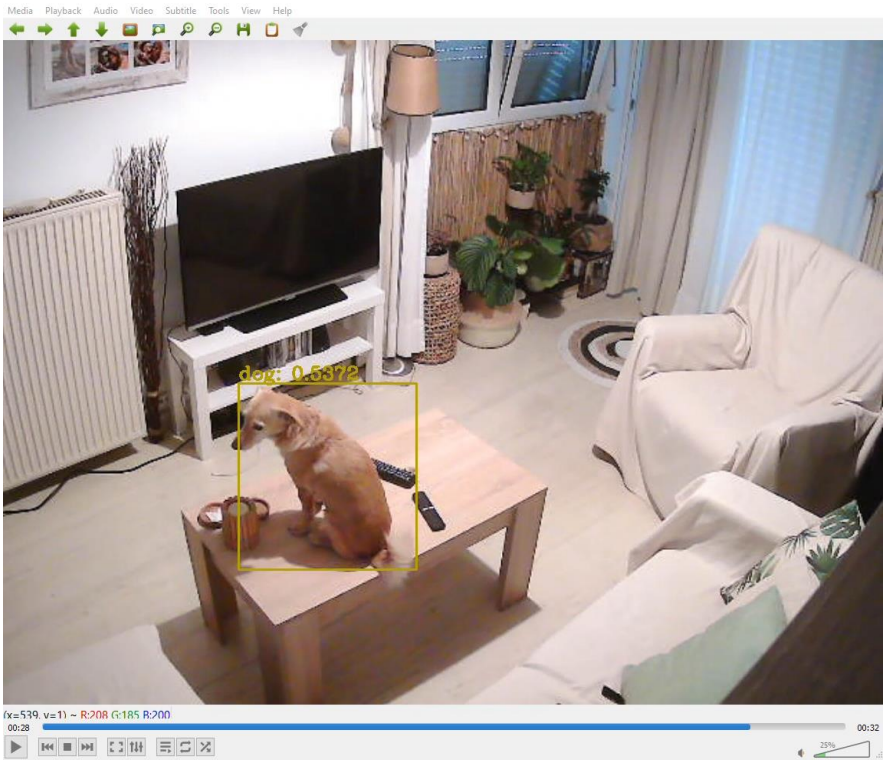
YOLOv4:



Εικόνα 68: Αποτελέσματα σε real-time YOLOv4

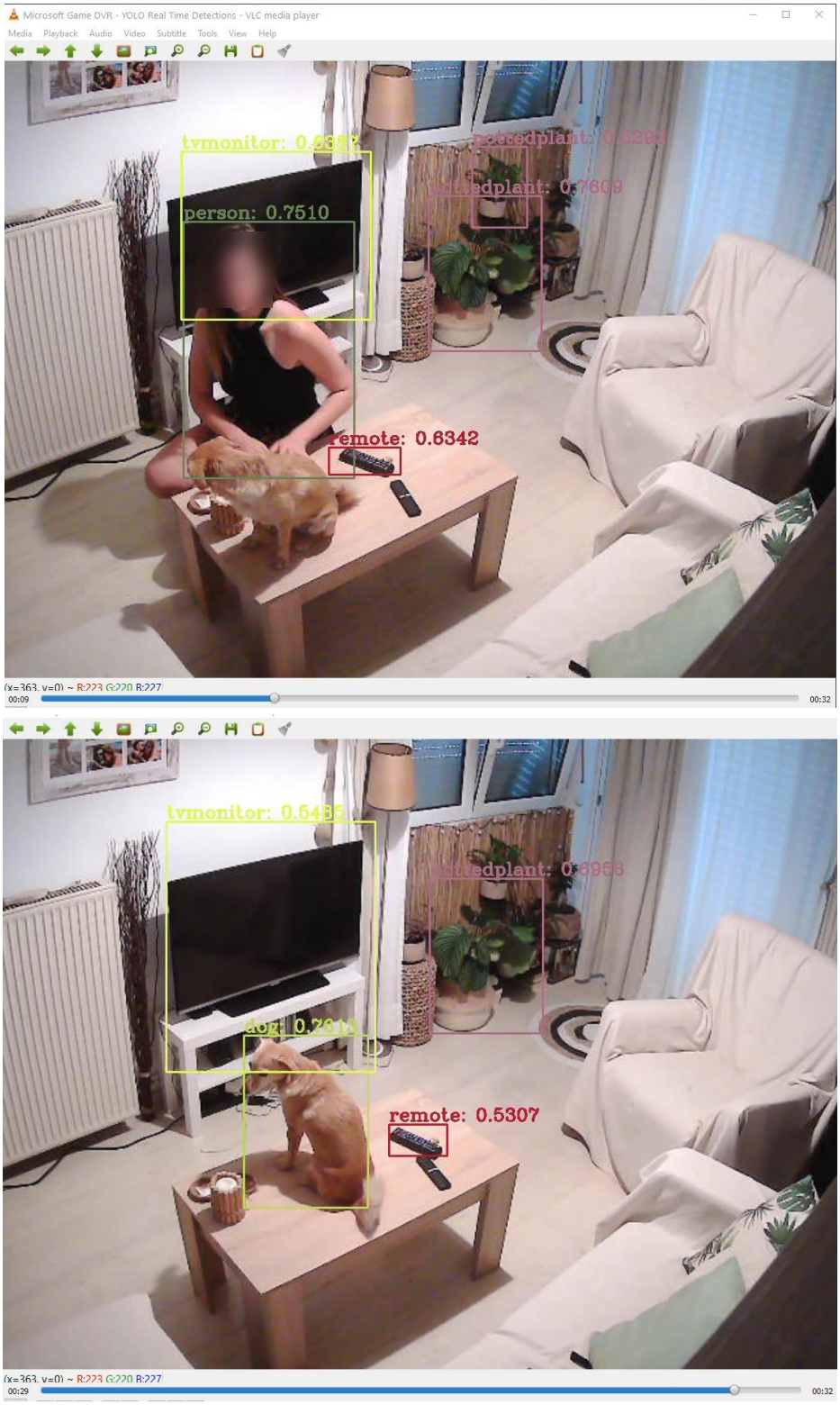
Tiny-YOLOv3:





Εικόνα 69: Αποτελέσματα σε real-time video tiny-YOLOv3

Tiny-YOLOv4:



Εικόνα 70: Αποτελέσματα σε real-time video tiny-YOLOv4

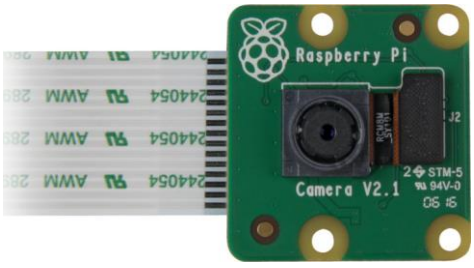
5.3 Real Time ανιχνευση με Raspberry pi

Ένα συνεχώς αναπτυσσόμενο πεδίο είναι το Edge AI. Λόγω της ευχρηστίας, της διαθεσιμότητας και της υψηλής απόδοσης είναι όλο και συχνότερη η εφαρμογή αλγορίθμων βαθειάς μηχανικής μάθησης σε συσκευές χαμηλού κόστους, όπως το Raspberry pi.

Raspberry pi

Το Raspberry Pi δημιουργήθηκε από το "Raspberry Pi Foundation" με σκοπό τη προώθηση της διδασκαλίας της βασικής επιστήμης των υπολογιστών στα σχολεία. Είναι ένας υπολογιστής, με μέγεθος πιστωτικής κάρτας, αλλά πλήρως λειτουργικός. Το κόστος του είναι αρκετά μικρό. Το Raspberry έχει όλες τις λειτουργίες ενός υπολογιστή και μπορούμε να συνδέσουμε πάνω σε αυτό όλες τις γνωστές μας περιφερειακές συσκευές όπως οθόνη, κάμερα, πληκτρολόγιο, ποντίκι, ηχεία. Ο βασικός σκοπός ήταν η εκπαίδευση στα σχολεία, τόσο στο χειρισμό ενός υπολογιστή όσο και στην εκμάθηση προγραμματισμού. Χρησιμοποιείται για όλες τις απλές λειτουργίες ενός υπολογιστή όπως είναι η χρήση internet, χρήση διαφόρων εφαρμογών, αναπαραγωγή μουσικής και βίντεο κλπ. Μπορεί να χρησιμοποιηθεί σε εφαρμογές για το σπίτι, όπως: μετατροπή απλής τηλεόρασης σε smart TV, μετατροπή απλού εκτυπωτή σε ασύρματο, σύστημα παρακολούθησης χώρου, συσκευή αύξησης της εμβέλειας του WiFi, δημιουργία media streaming box. Χρησιμοποιείται επίσης σε ηλεκτρονικές εφαρμογές και κυρίως σε εφαρμογές IoT αφού πάνω του μπορούν να συνδεθούν διάφοροι αισθητήρες (θερμοκρασίας, υγρασίας, κίνησης, άλλες πλακέτες (breadboards, arduino), LCD οθόνες, καθώς και να χρησιμοποιηθεί για οδήγηση ρομποτικών βραχιόνων, κινητήρων κλπ.

Εικόνα 71: Raspberry pi 4



Εικόνα 72: Raspberry Pi Camera Module

Υλικό Hardware

Raspberry Pi 4 Model B 4GB

Σε αυτή την εργασία χρησιμοποιήθηκε το Raspberry Pi 4 Model B 4GB. Το μοντέλο αυτό είναι το τελευταίο προϊόν της δημοφιλούς σειράς υπολογιστών Raspberry Pi. Έχουν πραγματοποιηθεί μεγάλες αλλαγές στην αύξηση της ταχύτητας του επεξεργαστή, στην απόδοση των πολυμέσων, στην μνήμη RAM αλλά και τη συνδεσιμότητα του σε σχέση με το προηγούμενο μοντέλο διατηρώντας ταυτόχρονα την συμβατότητα προς τις παλιότερες εκδόσεις και παρόμοια κατανάλωση ενέργειας. Για τον τελικό χρήστη, το Raspberry Pi 4 Model B παρέχει επίδοση όπως ένα κλασικό σύστημα Η/Υ. Τα βασικά χαρακτηριστικά αυτού του προϊόντος περιλαμβάνουν, τετραπύρηνο επεξεργαστή υψηλής απόδοσης στα 64bit, , αποκωδικοποίηση βίντεο μέχρι και 4Kp60, μνήμης RAM 4GB.

Raspberry Pi 8MP Camera Module (RB-CAMERAV2)

Το Raspberry Pi Camera Module V2 είναι η επίσημη κάμερα από το Raspberry Pi Foundation. Διαθέτει εξαιρετικά υψηλής ποιότητας αισθητήρα εικόνας Sony IMX219 8 megapixel και φακό σταθερής εστίασης. Η μονάδα κάμερας V2 είναι ικανή για στατικές εικόνες 3280 x 2464 pixel και υποστηρίζει επίσης βίντεο 1080p30, 720p60 και 640x480p90. Η μονάδα συνδέεται με το Raspberry Pi μέσω καλωδίου 15 Pin Ribbon, στο 15-pin MIPI Camera Serial Interface (CSI), το οποίο σχεδιάστηκε ειδικά για τη διασύνδεση με κάμερες.

Usb camera

Για καλύτερη ανάλυση χρησιμοποιήθηκε και camera USB με ανάλυση FULL HD 1920x1080 1080p (30 καρέ ανά δευτερόλεπτο) , φακό 3,6M ,γωνία θέασης 85 ° και ρυθμό καρέ βίντεο: 30fps.

Software

Raspbian

Το λειτουργικό σύστημα που χρησιμοποιήθηκε είναι το Rasbian. Βασίζεται στην έκδοση Debian των Linux και είναι βελτιστοποιημένο για το hardware Raspberry pi με ειδικό σχεδιασμό για την καλύτερη δυνατή απόδοση. Σε αυτό το λειτουργικό πραγματοποιήθηκε η ανίχνευση αντικειμένων σε πραγματικό χρόνο με την έκδοση του tiny-YOLO.

VNC

Το VNC είναι μια τεχνολογία που προσφέρει απομακρυσμένη επιφάνεια εργασίας. Επιτρέπει την οπτική προβολή επιφάνειας εργασίας ενός υπολογιστή και τον έλεγχο αυτής από απόσταση μέσω μια σύνδεσης δικτύου. Η τεχνολογία αυτή είναι χρήσιμη στα δίκτυα οικιακών υπολογιστών, επιτρέποντας σε κάποιον να έχει πρόσβαση στους επιτραπέζιους υπολογιστές από άλλο μέρος του σπιτιού ή κατά τη διάρκεια ταξιδιού.

Επίσης όλος ο κώδικας είναι γραμμένος σε γλώσσα προγραμματισμού Python και οι βιβλιοθήκες αυτές που περιγράφηκαν σε προηγούμενες ενότητες.

Υλοποίηση στο Raspberry

Η ιδέα είναι να μπορέσουμε να πραγματοποιήσουμε ανίχνευση αντικειμένων σε πραγματικό χρόνο τοποθετώντας το Raspberry, συνδεδεμένο με μία κάμερα, σε οποιοδήποτε σημείο επιθυμούμε δεδομένου ότι υπάρχει σύνδεση με WiFi. Με την βοήθεια της τεχνολογίας VNC μπορούμε να ελέγχουμε απομακρυσμένα την επιφάνεια εργασίας του Raspberry και να εκτελούμε το πρόγραμμα για την ανίχνευση των αντικειμένων. Επιλέγουμε αυτόν τον τρόπο για να

αποφύγουμε τον επιπλέον εξοπλισμό για περιφερειακά του Raspberry, οθόνη, ποντίκι, πληκτρολόγιο, καθώς η συσκευή μας μπορεί να είναι τοποθετημένη σε δύσκολα προσβάσιμο σημείο.

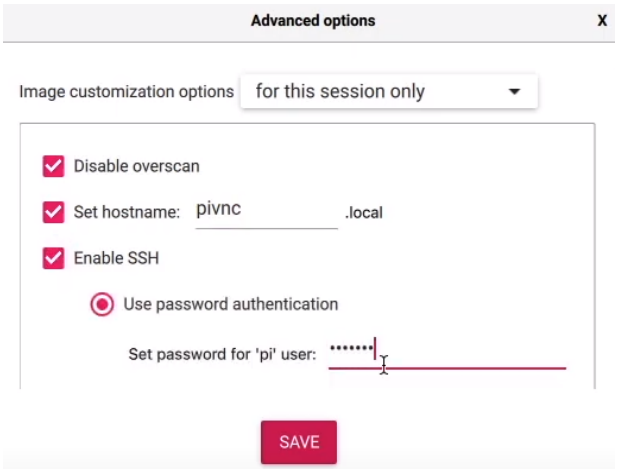
Για να μπορέσουμε να χρησιμοποιούμε πλήρως λειτουργικά το Raspberry χωρίς να έχει κανέναν εξοπλισμό, δηλαδή να είναι «headless», πρέπει να ακολουθήσουμε μια σειρά από ενέργειες. Κατά την εγκατάσταση του Raspbian θα δημιουργήσουμε μια σύνδεση δικτύου για το Raspberry pi. Αυτό θα επιτευχθεί με την δημιουργία ενός αρχείου wpa_supplicant.conf που θα προστεθεί στον κατάλογο με τα αρχεία εγκατάστασης του λειτουργικού. Το αρχείο αυτό θα περιέχει ρυθμίσεις σχετικά με το δίκτυο στο οποίο θα συνδεθεί το Raspberry (Εικόνα 69). Με αυτόν τον τρόπο έχουμε επιτύχει άμεση σύνδεση του Raspberry στο δίκτυο που επιθυμούμε με την συμπλήρωση των πεδίων «ssid» και «psk».

```
country=GB
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="NETWORK-NAME"
    psk="NETWORK-PASSWORD"
    key_mgmt=WPA-PSK
}
```

Εικόνα 73: Αρχείο wpa_supplicant.conf

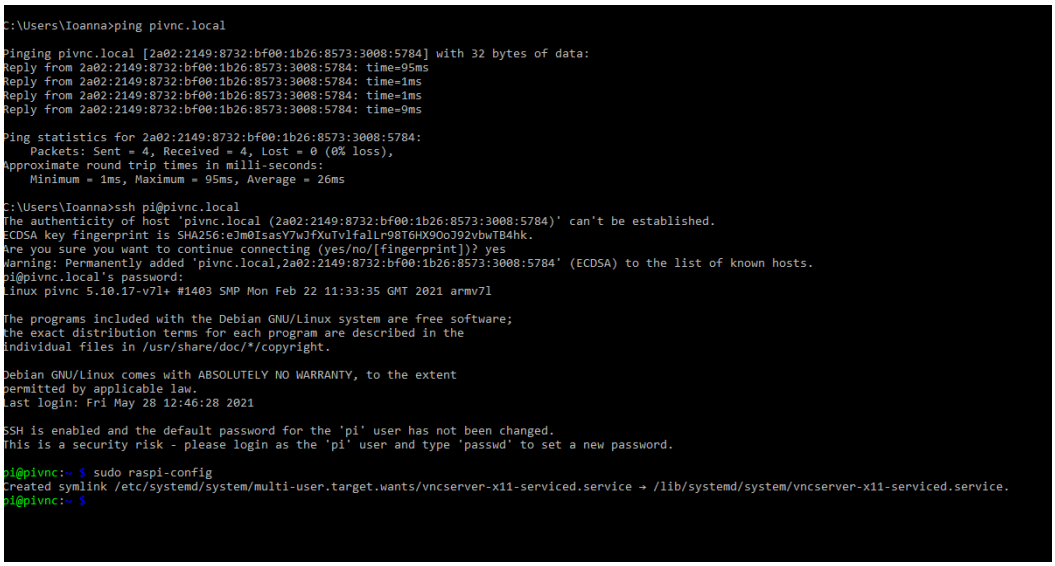
Κατά τη διάρκεια της εγκατάστασης του λειτουργικού συστήματος πρέπει να ρυθμίσουμε και κάποιες άλλες παραμέτρους, όπως το όνομα που θα έχει η συσκευή στο δίκτυο μας και την ενεργοποίηση του ssh για να μπορέσουμε να έχουμε πρόσβαση στη συσκευή μέσω του τερματικού του υπολογιστή μας.



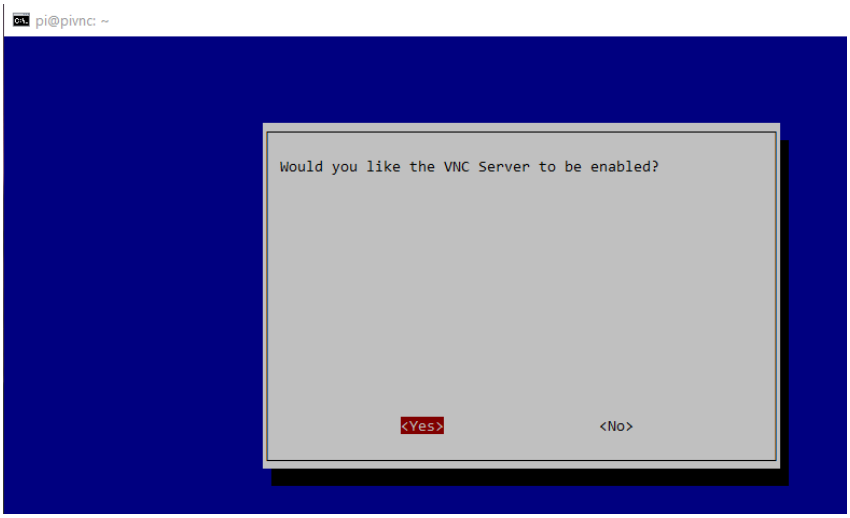
Εικόνα 74: Ρυθμίσεις κατά την εγκατάσταση του λειτουργικού

Για να ελέγξουμε ότι η σύνδεση του Raspberry στο δίκτυο μας είναι επιτυχημένη εκτελούμε τις εντολές που φαίνονται στην Εικόνα 71 και αποκτούμε απομακρυσμένη πρόσβαση στη συσκευή

ώστε να μπορέσουμε να δώσουμε την άδεια να χρησιμοποιηθεί το λογισμικό VNC (Εικόνα 72) και να μπορέσουμε να αποκτήσουμε πλήρη έλεγχο της συσκευής όχι μόνο μέσω του τερματικού της αλλά και από την επιφάνεια εργασίας της συσκευής.

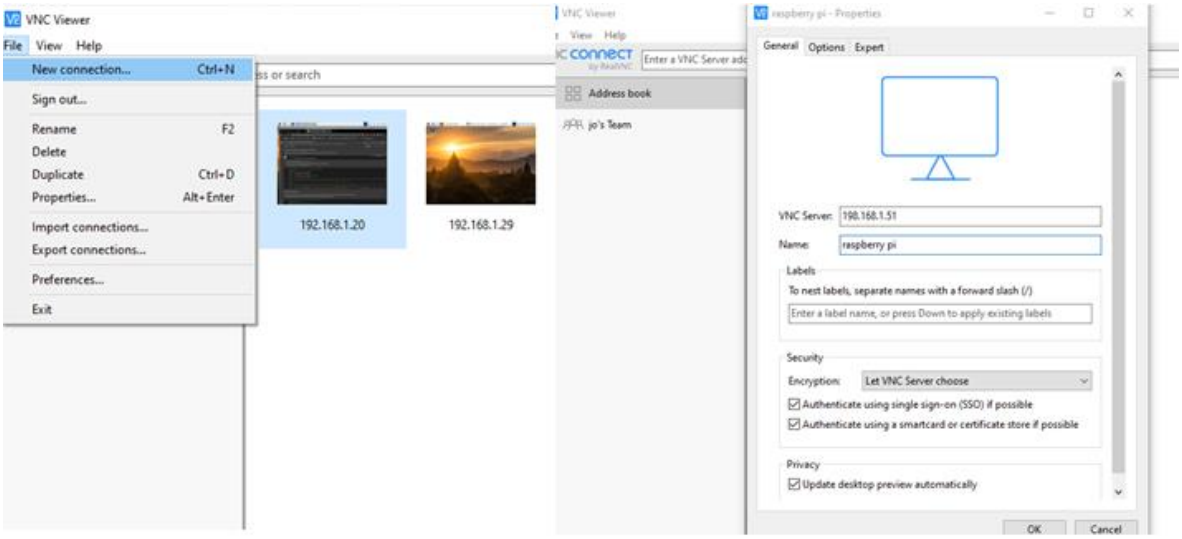


Εικόνα 75: Έλεγχος επιτυχημένης σύνδεσης του Raspberry στο δίκτυο



Εικόνα 76: Ενεργοποίηση του VNC με απομακρυσμένη σύνδεση μέσω ssh

Αφού ολοκληρωθεί η σύνδεση στο δίκτυο το επόμενο βήμα είναι να ξεκινήσουμε μια καινούρια σύνδεση μέσω του VNC από τον υπολογιστή.



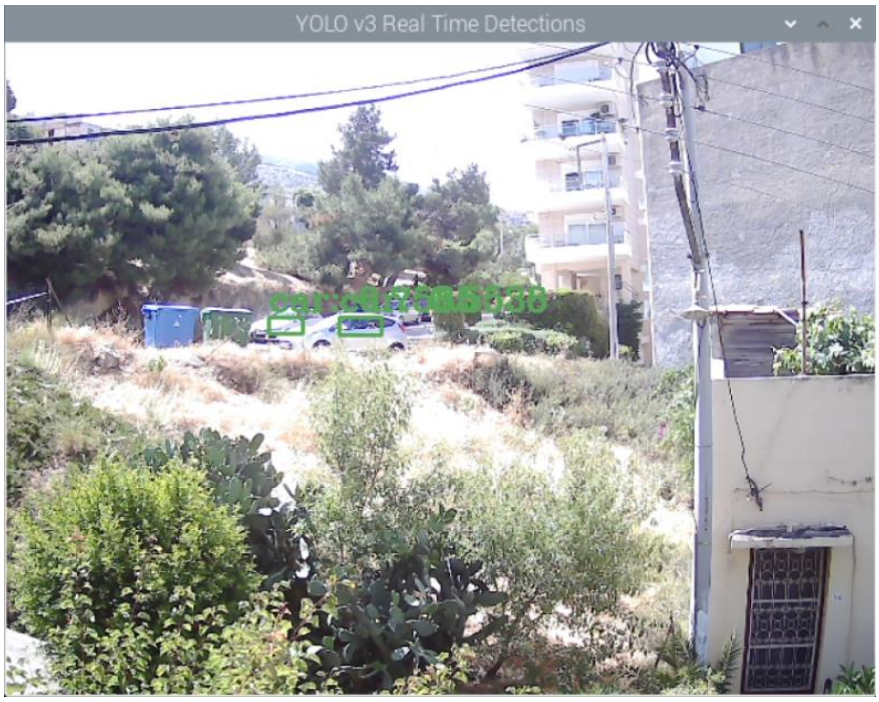
Εικόνα 77: Σύνδεση στο VNC

C

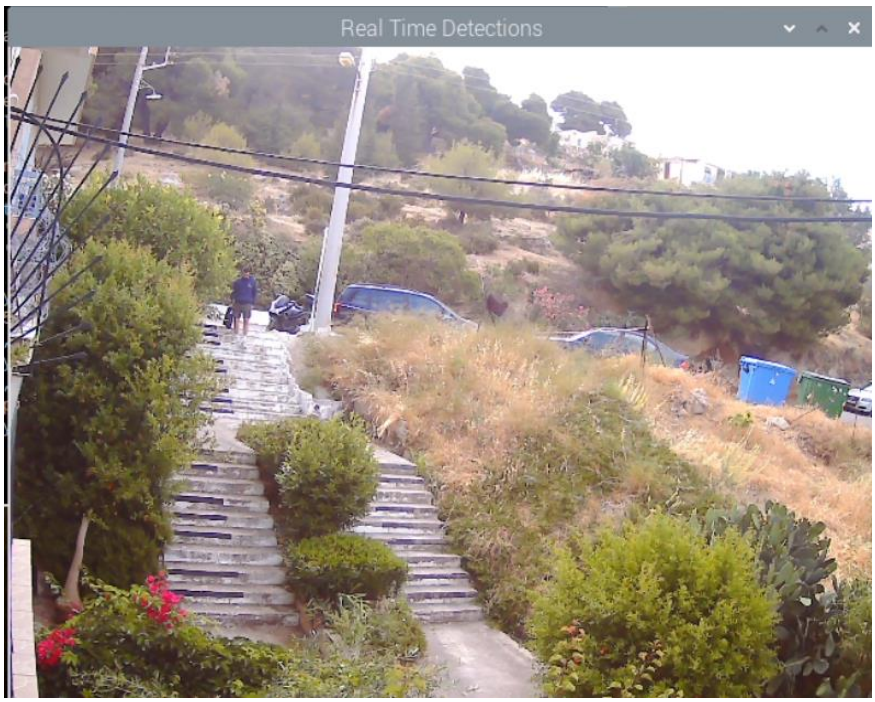
Εικόνα 78: Σύνδεση στο VNC

Όπως αναφέραμε στην προηγούμενη παράγραφο, η έκδοση του YOLO που είναι καταλληλότερη για τον εντοπισμό των αντικειμένων σε πραγματικό χρόνο, είναι η tiny-YOLO. Ειδικά στην συσκευή Raspberry, λόγω υπολογιστικής ισχύς η ολόκληρη έκδοση του YOLO δεν μπορεί να υποστηριχθεί. Στις δοκιμές που θα γίνουν, θα εξετάσουμε τις δύο τελευταίες εκδόσεις του tiny-YOLO, version3 και version4, και θα συγκρίνουμε τα αποτελέσματα.

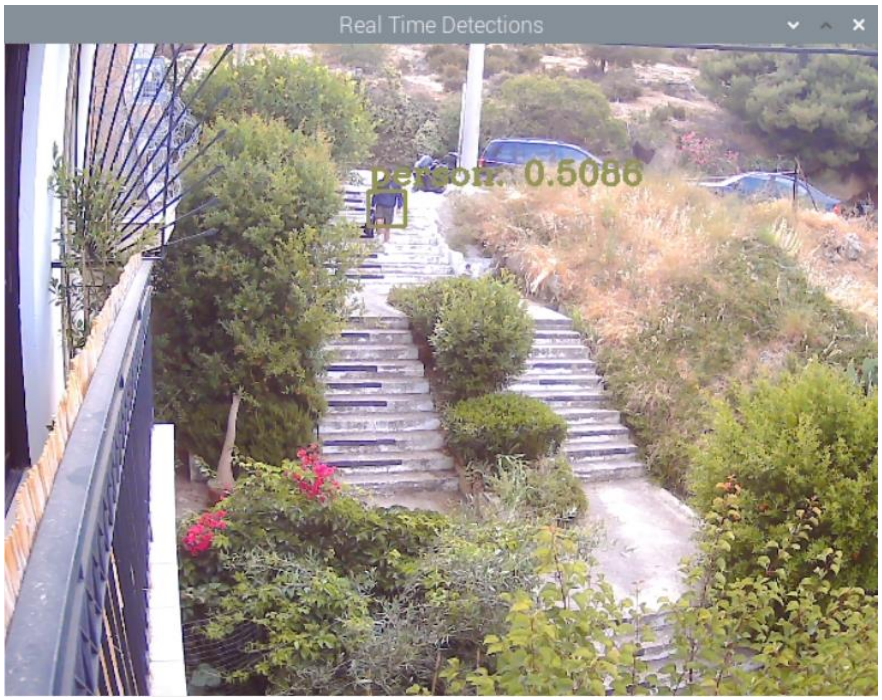
Tiny-YOLOv3:



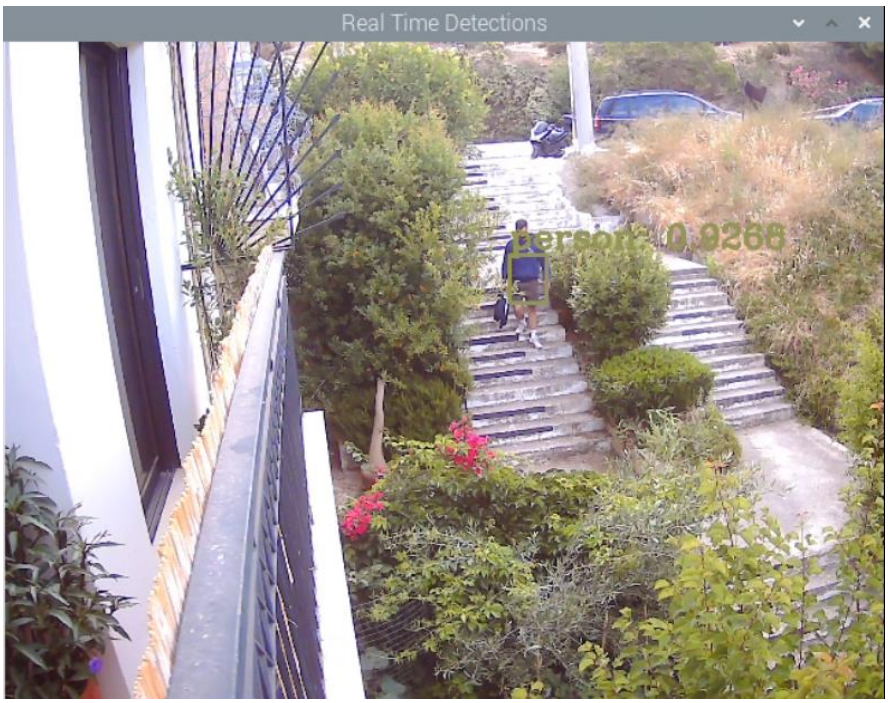
Εικόνα 79: Real-time ανίχνευση αυτοκινήτων με Raspberry



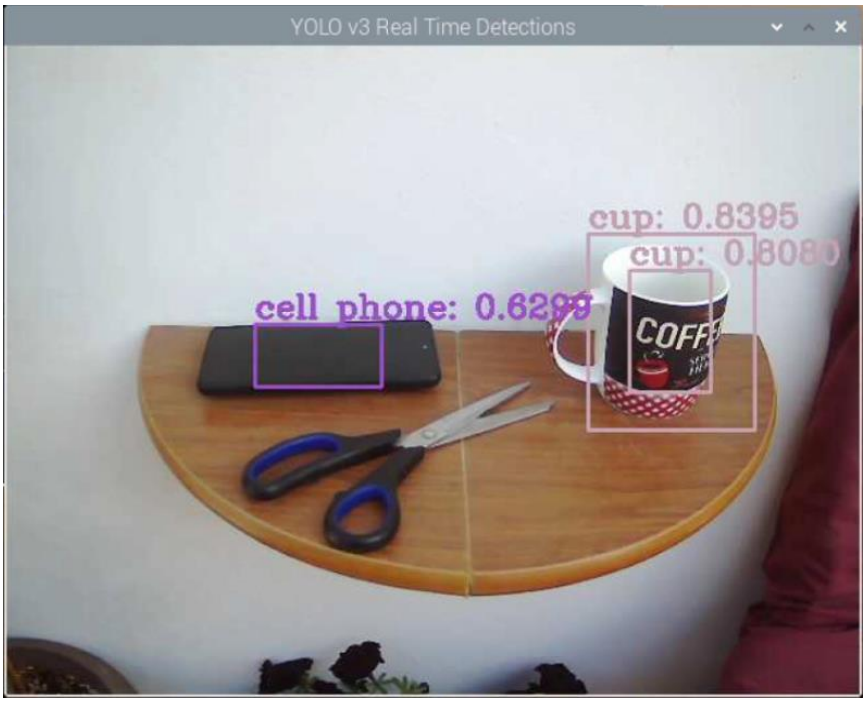
Εικόνα 80: Real-time ανίχνευση ατόμου με Raspberry



Εικόνα 81: Real-time ανίχνευση ατόμων με Raspberry



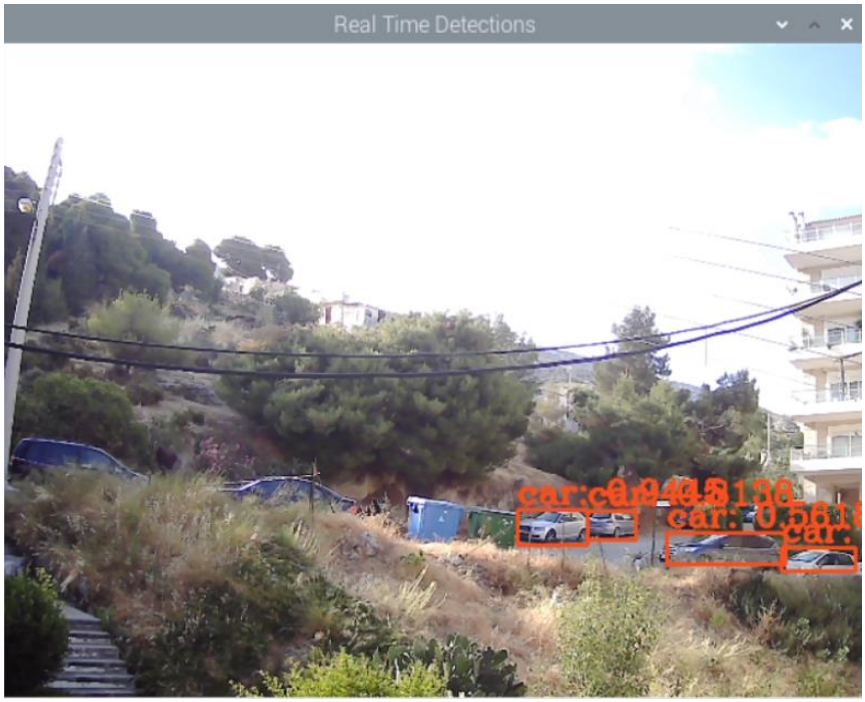
Εικόνα 82: Real-time ανίχνευση ατόμων με Raspberry



Εικόνα 83: Real-time ανίχνευση αντικειμένων με Raspberry

Εικόνα 84: Real-time ανίχνευση αντικειμένων με Raspberry

Tiny-YOLOv4:

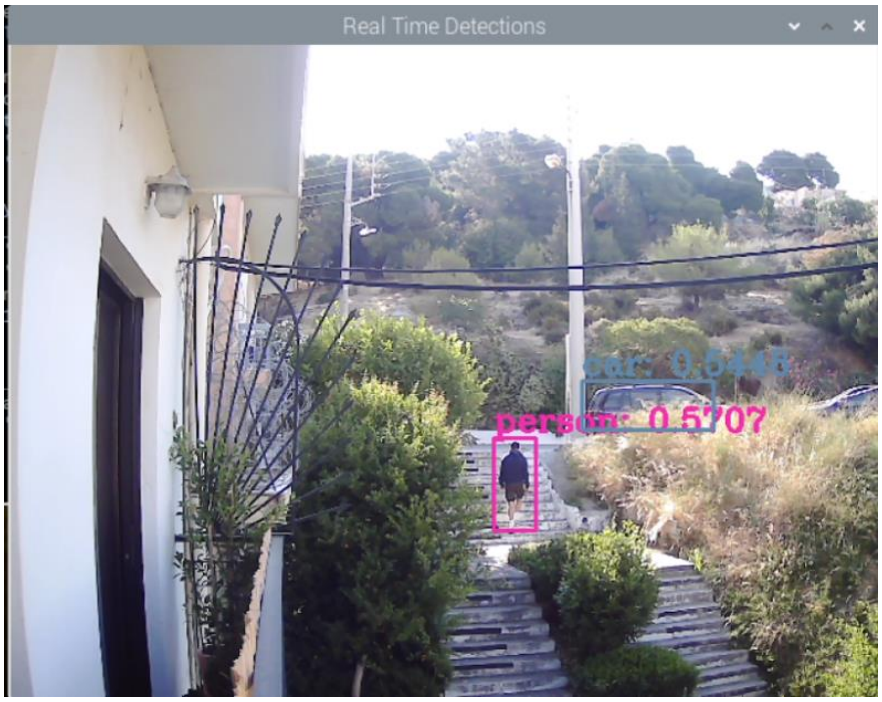


Εικόνα 85: Real-time ανίχνευση αυτοκινήτων με Raspberry

Εικόνα 86: Real-time ανίχνευση αντικειμένων με Raspberry

Εικόνα 87: Real-time ανίχνευση ατόμων με Raspberry

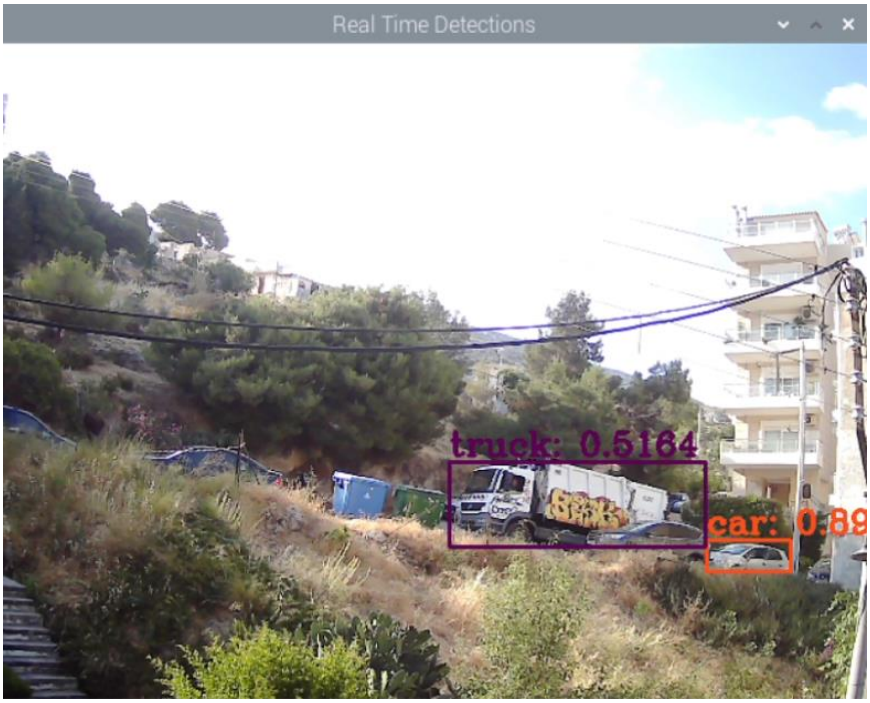
Εικόνα 88: Real-time ανίχνευση ατόμων με Raspberry



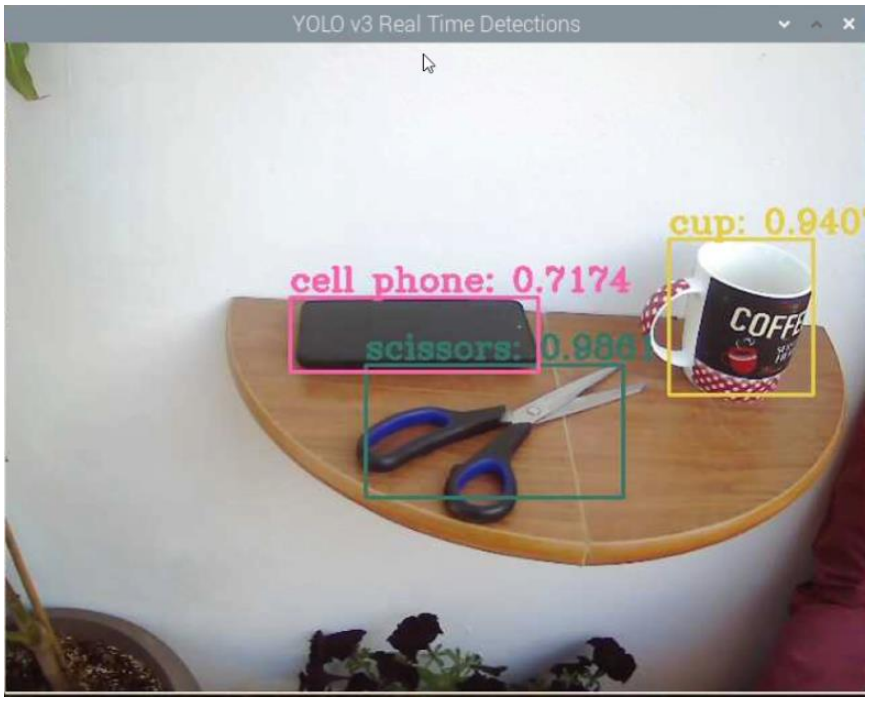
Εικόνα 89: Real-time ανίχνευση ατόμων με Raspberry

Εικόνα 90: Real-time ανίχνευση ατόμων με Raspberry

Εικόνα 91: Real-time ανίχνευση ατόμων με Raspberry



Εικόνα 92: Real-time ανίχνευση αυτοκινήτων με Raspberry



Εικόνα 93: Real-time ανίχνευση αντικειμένων με Raspberry

Από τα αποτελέσματα που προέκυψαν παρατηρούμε ότι η τέταρτη έκδοση του αλγόριθμου YOLO με την tiny αρχιτεκτονική, υπερτερεί κατά πολύ σε σχέση με την τρίτη. Είναι φανερό πως μπορεί να εντοπίζει περισσότερα αντικείμενα στο κάθε frame και με μεγαλύτερη ακρίβεια. Σε πολλές περιπτώσεις ο αλγόριθμος YOLOv3 δεν μπορεί να εντοπίσει καθόλου κάποια αντικείμενα ακόμα και αν αυτά είναι σε αρκετά κοντινή απόσταση. Από την άλλη, ο αλγόριθμος YOLOv4 εντοπίζει αντικείμενα σε μακρινή απόσταση με μεγάλη βεβαιότητα.

6. Συμπερασματα – Προτασεις για μελλοντικη ερευνα

Στην παρούσα διπλωματική παρουσιάστηκαν αλγόριθμοι Μηχανικής Μάθησης και Βαθιάς μάθησης τόσο σε θεωρητικό επίπεδο όσο και σε πρακτικό με τη γλώσσα Python. Στη συνέχεια, μελετήθηκαν οι αλγόριθμοι YOLOv3 και YOLOv4 στην πλήρη έκδοση και στην πιο ελαφριά έκδοση. Η πλήρης έκδοση χρησιμοποιήθηκε για την σύγκριση των αποδόσεων του YOLOv3 και YOLOv4 σε υπολογιστικό σύστημα, ενώ η ελαφριά έκδοση για την ανίχνευση των αντικειμένων σε πραγματικό χρόνο στη συσκευή Raspberry pi. Η υλοποίηση των αλγορίθμων έγινε σε γλώσσα προγραμματισμού Python με σκοπό την ανίχνευση αντικειμένων μέσα από ένα σύνολο δεδομένων 80 κλάσεων, το COCO Dataset.

Η ανίχνευση έγινε σε τρεις διαφορετικές κατηγορίες δεδομένων εισόδου. Στην πρώτη κατηγορία ο αλγόριθμος δέχτηκε σαν είσοδο μια εικόνα και σαν αποτέλεσμα έδωσε την ίδια εικόνα με εντοπισμένα τα αντικείμενα και ζωγραφισμένα περιβλήματα γύρω από αυτά. Στη δεύτερη κατηγορία η είσοδος ήταν ένα ολόκληρο βίντεο και το αποτέλεσμα ένα ίδιο βίντεο με τα εντοπισμένα αντικείμενα σε κάθε frame. Τέλος, η τρίτη κατηγορία δεχόταν σαν είσοδο ένα video stream και ταυτόχρονα εντόπιζε τα αντικείμενα στην εικόνα και σχεδίαζε τα περιβλήματα. Η τελική δοκιμή του αλγορίθμου πραγματοποιήθηκε στη συσκευή Raspberry χωρίς περιφερειακά, η οποία συνδέθηκε μέσω δικτύου στον υπολογιστή. Έτσι, έχοντας τη συσκευή σε οποιοδήποτε σημείο στην εμβέλεια του δικτύου και με απομακρυσμένη πρόσβαση, πραγματοποιήθηκε ο εντοπισμός των αντικειμένων. Αυτό που παρατηρήθηκε είναι πως η τέταρτη έκδοση του αλγόριθμου στην ελαφριά έκδοση του υπερτερεί σε σημαντικό βαθμό έναντι της τρίτης έκδοσης και ως προς την απόδοση αλλά και ως προς την ακρίβεια των αποτελεσμάτων.

Οι προτάσεις για μελλοντική έρευνα συνοψίζονται στις εξής:

- Δημιουργία custom dataset και εκπαίδευση του αλγορίθμου με αυτό ανάλογα με συγκεκριμένη μελέτη περίπτωσης για την real-time ανίχνευση αντικειμένων με το Raspberry Pi.
- Real-time ανίχνευση αντικειμένων με Raspberry Pi στην αρχική έκδοση του αλγορίθμου με τη βοήθεια του Cloud Computing.
- Αλλαγή παραμέτρων στην αρχιτεκτονική του YOLO με σκοπό την βελτίωση της απόδοσής του για συγκεκριμένη μελέτη περίπτωσης.
- Δημιουργία εφαρμογής για κινητό για την ανίχνευση αντικειμένων από την συσκευή.

7. Βιβλιογραφια

1. A. Bochkovskiy, C-Y Wang, Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection,2020

2. A. Rajkomar, S. Lingam, A. G. Taylor, M. Blum and J. Mongan, "High-throughput classification of radiographs using deep convolu- tional neural networks," Journal of digital imaging, vol. 30, no. 1, pp. 95-101, doi: 10.1007/s10278-016-9914-9, 2017.

3. A. M. Turing, "Computing machinery and intelligence," Mind, vol. 59, no. 236, pp.433-460, 1950

4. A. I. Schein and L. H. Ungar, "Active learning for logistic regression,", 2005

5. A Comprehensive Introduction to Different Types of Convolutions in Deep Learning [Internet) Medium. cited 16 February 2020] Available from: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-leaming -669281258215

6. C.– W.Hsu, C.C.Chang and C.-J.Lin, “A Practical Guide to Support Vector Classification,” 2003

7. C. Chen, C. Chen, E. Durand, F. Forbes and O. Francois, "Bayesian clustering algorithms ascertaining spatial population structure: A new computer program and a comparison study," Molecular Ecology Notes, vol. 7, no. 5, pp. 747-756, 2007.

8. C. Cortes and V. N. Vapnik, "Support-vector networks," 1995.

9. C. X. Ling and J. Du, "Active learning with direct query construction," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining,2008.

10. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguclov, D. Erhan, V.Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, 2015.

11. David Kriesel, A Brief Introduction to Neural Networks,2012

12. D.I.C. MacKay, “Information-based objective functions for active data selection," Neural Computation, vol. 4, no. 4, pp. 590-604, 1992.

13. D.W. Hosmer and S.Lemeshow, Applied Logistic Regression,1989

14. Deep Learning Methods and Applications Li Deng and Dong Yu https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/DeepLearning-NowPublishing-Vol7-SIG-039.pdf

15. Fukushima K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks. 1988 Jan 1, 1(2):119-30.

16. H. Drucker, C.J.C Burges,L. Kaufman, A.J.Smola and V.Vapnik, “Support Vector Regression Machines,” in Advances in Neural Information

17. Home - Keras Documentation Internet). Keras.io. 2020 [cited 16 February 2020).Available from: https://keras.io/

18. He K, Zhang X, Ren S. Sun J. Deep residual learning for image recognition In Proceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

19. J.Quinlan, “Induction of Decision Trees”, Machine Learning, vol.1, no.1, pp. 81-106,1986

20. J. Petterson and A. Gibson, Deep learning: A practitioner's approach, O'Reilly Media, Inc, 2017.

21. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. Διαθέσιμο στη διεύθυνση: https://pjreddie.com/darknet/yolo/

22. K. P. Murphy Machine Learning_A Probabilistic Perspective, 2012

23. Kaggle https://www.kaggle.com

24. Krizhevsky A, Hinton G. Convolutional deep belief networks on cifar-10. Unpublished manuscript. 2010 Aug;40(7):1-9.

25. L. Kaufman and P. J. Rousseeuw, Finding Groups in Data, 1990.

26. L. Breiman, "Bagging predictors," Machine Learning archive, 1996.

27. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W. Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989 Dec;1(4):541-51.

28. Lecun Y. LeNet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet. 2015:20:5.

29. LeCun, Y. a. Bengio and Yoshua, "Convolutional Networks for Images, Speech, and Time Series," in The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1998, p. 255-258.

30. Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, Orlando De Jesús : Neural Network Design 2nd Edtion

31. Mitchell T. Machine learning. New York: McGraw-Hill, 2013.

32. Nielsen M. Neural Networks and Deep Learning [Internet) Neuralnetworksanddeeplearning.com. 2020 [cited 16 February 2020]. Available from: http://neuralnetworksanddeeplearning.com/

33. OpenCV Available from: https://opencv.org/about/

34. P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, “Object Detection with Discriminatively Trained Part-Based Models,” IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 32, Issue: 9), pp. 1627-1645, 9 2010.

35. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back propagating errors, nature. 1986 Oct;323(6088):533-6.

36. Raspberry Pi web page: https://www.raspberrypi.org

37. S. J. Prince, Computer vision: models, learning, and inference. Cambridge University Press, 2012.

38. S. Geman, E. Bienenstock and R. Doursat, "Neural networks and the bias/variance dilemma," Neural Computation, vol. 4, no. 1, pp. 1-58, 1992.

39. S. Sharma, “Activation Functions: Neural Networks,” 6 9 2017.

40. Szeliski R. Computer vision: algorithms and applications. Springer Science & Business Media, 2010 Sep 30.

41. Sutton RS, Barto AG, Introduction to reinforcement learning. Cambridge: MIT press;1998 Mar 1.

42. T. G. Dietterich, "Ensemble Methods in Machine Learning," multiple classifier systems, pp. 1-15, 2000.

43. Turk M, Pentland A. Face recognition using cigenfaces. In Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition 1991 Jan 1 (pp. 586-587).

44. T. M. Michell, "Machine Learning," New York: McGraw-Hill, 2013.

45. Tan M, Le QV. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946. 2019 May 28.

46. Y. Freund, "Schapire: Experiments with a new boosting algorithm,", 1996.

47. Y. Freund, "Schapire RE: A decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, 1997.

48. Zhu X, Goldberg AB. Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning, 2009 Jun 8;3(1):1-30.

49. Zoph B, Le QV. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578. 2016 Nov 5.

8. Appendix

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

regression = LinearRegression()
regression.fit(X_train, y_train)

# Prediction
predictions = regression.predict(X_test)

# Visualisation
plt.scatter(X_train, y_train, color='magenta')
plt.plot(X_train, regression.predict(X_train), color='black')
plt.title("Salary-Experience")
plt.xlabel("Years of experience")
plt.ylabel("Salary")
plt.show()
```

Απλή γραμμική παλινδρόμηση

```
import numpy as np
import pandas as pd

dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encode data
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))

# Split into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train model
from sklearn.linear_model import LinearRegression
regr = LinearRegression()
regr.fit(X_train, y_train)

# Prediction
y_predictions = regr.predict(X_test)
np.set_printoptions(precision=2)

# Display predicted values and real values in 2 columns to compare
print(np.concatenate((y_predictions.reshape(len(y_predictions), 1), y_test.reshape(len(y_test), 1)), axis=1))
```

Πολλαπλή Γραμμική Παλινδρόμηση

```
data = dataset.copy()
data['Admitted'] = data['Admitted'].map({'Yes': 1, 'No': 0})

X = data['SAT']
y = data['Admitted']

plt.scatter(X, y, color='#236f6f')
plt.title("Admittance")
plt.xlabel('SAT')
plt.ylabel('Admitted')
plt.show()

x = sm.add_constant(X)
reg_lin = sm.OLS(y, x)
results_lin = reg_lin.fit()

plt.scatter(X, y, color='#236f6f')
y_hat = X * results_lin.params[1] + results_lin.params[0]

plt.plot(X, y_hat, lw=2.5, color='#800000')
plt.title("Admittance")
plt.xlabel('SAT')
plt.ylabel('Admitted')
plt.show()

reg_log = sm.Logit(y, x)
results_log = reg_log.fit()


def f(x, b0, b1):
    return np.array(np.exp(b0 + x * b1) / (1 + np.exp(b0 + x * b1)))


f_sorted = np.sort(f(X, results_log.params[0], results_log.params[1]))
x_sorted = np.sort(np.array(X))
```

Λογιστική Παλινδρόμηση

```
# libraries
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

# dataset
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values

# K-Means model on the dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=37)
y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 30, c = '#eb4034', label = 'Ομάδα 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 30, c = '#f0e354', label = 'Ομάδα 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 30, c = '#296120', label = 'Ομάδα 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 30, c = 'cyan', label = 'Ομάδα 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 30, c = '#d0a0d9', label = 'Ομάδα 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = '#1e51fa', label = 'Κέντρα')
plt.title('Ομάδες Καταναλωτών')
plt.xlabel('Ετήσιο Εισόδημα (K$)')
plt.ylabel('Σκορ δαπανών (1-100)')
plt.legend()
plt.show()
```

K-means

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, -1].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=0)

scaleX = StandardScaler()
X_train = scaleX.fit_transform(X_train)
X_test = scaleX.transform(X_test)

classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, Y_train)

# Predicting the test set results
predictions = classifier.predict(X_test)
print(predictions)

cm = confusion_matrix(Y_test, predictions)
print("The confusion matrix: ")
print(cm)

# Visualising the results
X_Set, Y_Set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start=X_Set[:, 0].min() - 1, stop=X_Set[:, 0].max() + 1, step=0.01),
                     np.arange(start=X_Set[:, 1].min() - 1, stop=X_Set[:, 1].max() + 1, step=0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75, cmap=ListedColormap(('#800000', '#236f6f')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(Y_Set)):
    plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j, 1],
                c=ListedColormap(('#800000', '#236f6f'))(i), label=j)
plt.title('Support Vector Machine')
plt.xlabel('Ηλικία')
plt.ylabel('Εκτιμώμενος Μισθός')
plt.legend()
plt.show()
```

Support Vector Machine

```
# Build and Train
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6, metric="minkowski",p=2) #constructor
knn.fit(X_train, y_train)

# prediction
prediction = knn.predict(scale.transform([[30,90000]]))
y_predictions = knn.predict(X_test)

print("The actual values vs The predicted values:")
print(np.concatenate((y_predictions.reshape(len(y_predictions), 1), y_test.reshape(len(y_test), 1)), axis=1))
print("The prediction for 30 years old and estimated salary 90000: ", prediction)

# confusion matrix classification metrics
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_predictions)
acs = accuracy_score(y_test, y_predictions)
print(cm, "\n", acs)

from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1, stop=X_set[:, 0].max() + 1, step=0.01),
                     np.arange(start=X_set[:, 1].min() - 1, stop=X_set[:, 1].max() + 1, step=0.01))
plt.contourf(X1, X2, knn.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.5, cmap=ListedColormap(('#800000', '#236f6f')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], alpha=0.5,
                c=ListedColormap(('#800000', '#236f6f'))(i), label=j)
plt.title('K-Nearest Neighbors')
plt.xlabel('Ηλικία')
plt.ylabel('Εκτιμώμενος Μισθός')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
```

K-Nearest Neighbor

```python
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
train_data = ImageDataGenerator(rescale=1./255,
                                shear_range=0.2,
                                zoom_range=0.2,
                                horizontal_flip=True)
training_set = train_data.flow_from_directory('dataset/training_set',
                                              target_size=(64, 64),
                                              batch_size=32,
                                              class_mode='binary')

test_datagen = ImageDataGenerator(rescale=1./255)
test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size=(64, 64),
                                            batch_size=32,
                                            class_mode='binary')

cnn = tf.keras.models.Sequential()

# Convolution
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[64, 64, 3]))

# Pooling
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# second convolutional layer
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Flattening
cnn.add(tf.keras.layers.Flatten())

# Full Connection
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

# Output Layer
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

```python
# Training the CNN

# Compiling
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Training the CNN on the Training set and evaluating it on the Test set
cnn.fit(x=training_set, validation_data=test_set, epochs=15)

# prediction
test_image = image.load_img('dataset/single_prediction/lizi3.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0) #batches
result = cnn.predict(test_image)
training_set.class_indices # define 1 = dog & 2 = cat
if result[0][0] == 1:
    prediction = 'dog'
else:
    prediction = 'cat'
print(prediction)
```

Convolutional Neural Networks