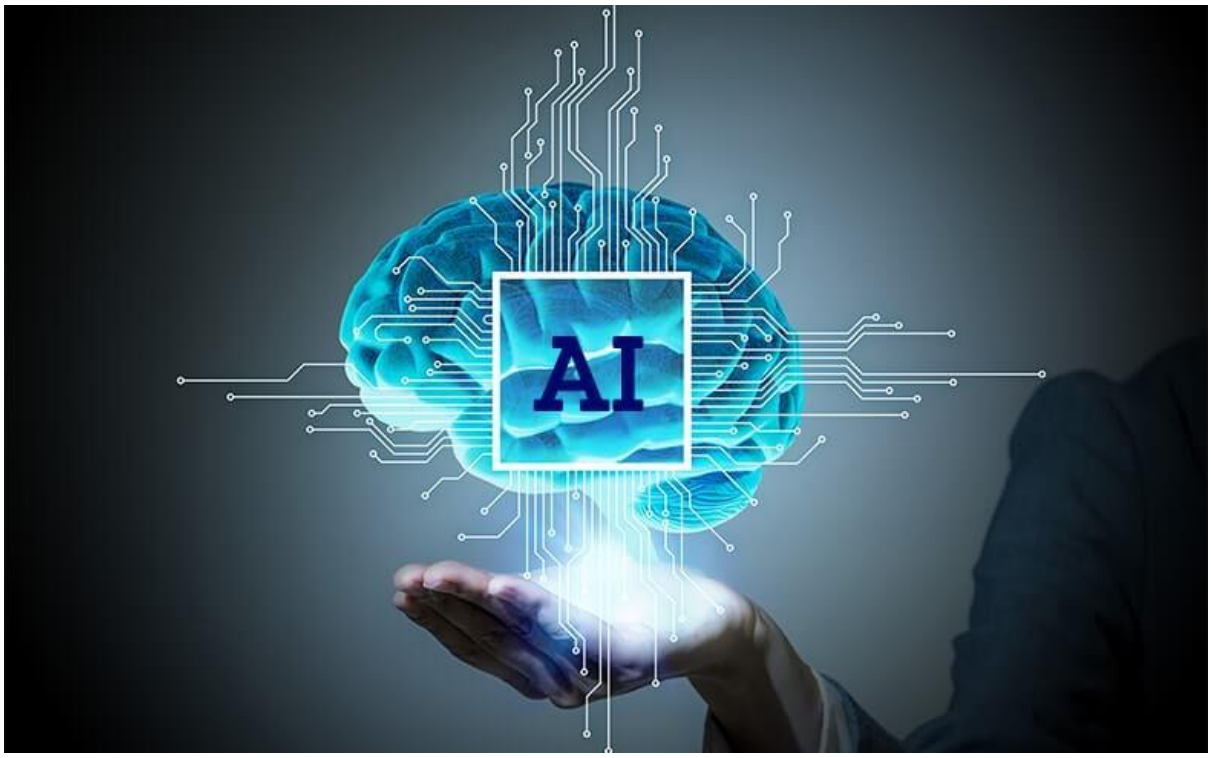
**Πανεπιστήμιο Πελοποννήσου**

**Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών**

*Θέμα : Τεχνικές οπτικής αναγνώρισης και μηχανικής μάθησης για ηλεκτρονική μετάδοση παρτίδων σκάκι.*




Φοιτητής: Κωστίνας Αθανάσιος – ΑΜ: 2153

Επιβλέπων Καθηγητής: Χριστοδούλου Σωτήρης

Πάτρα, 2021

**Ευχαριστίες**

Θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους που συνέβαλλαν ενεργά στη διεκπεραίωση αυτής της διπλωματικής. Τον επιβλέποντα καθηγητή μου κ. Χριστοδούλου Σωτήρη για την εμπιστοσύνη που μου έδειξε και την στήριξή του καθ' όλη τη διάρκεια της συνεργασίας μας την υποστήριξή και τις συμβουλές του σε όλα τα στάδια της διπλωματικής .Το θερμότερο Ευχαριστώ το οφείλω στους γονείς μου Ταξιάρχη και Φωτεινή και την αδελφή μου Ηλέκτρα στους οποίους στηρίχθηκα τόσο στην εκπόνηση αυτής της διπλωματικής όσο και στις σπουδές μου. Τέλος, είμαι ευγνώμων σε συγγενείς, φίλους και γνωστούς για την όποια συνεισφορά τους τόσο στην προσωπική όσο και στην επαγγελματική μου πορεία.

# Περιεχόμενα


## Περίληψη

Στην παρούσα διατριβή θα γίνει μία αφήγησή σε τεχνάσματα υπολογιστικής όρασης (Computer Vision) όπως και των δυνατοτήτων που προσφέρει στην επίλυση καθημερινών προβλημάτων. Συγκεκριμένα θα αναπτυχθούν οι έννοιες της εκπαίδευσης (training) μέσω των τεχνητών νευρωνικών δικτύων (ΤΝΔ) όπου θα δοθεί ο παραγόμενος κώδικας γραμμένος σε γλώσσα προγραμματισμού Python και οι τεχνολογίες που χρησιμοποιήθηκαν στην οπτική αναγνώριση σκακιέρας και σκακιστικών κομματιών με στόχο την μετάδοση παρτίδων σκάκι. Ο αλγόριθμος που θα χρησιμοποιηθεί για την διαδικασία οπτικής αναγνώρισης είναι ο YOLO.



## Λέξεις Κλειδιά

Τεχνητή νοημοσύνη, Μηχανική μάθηση, Βαθιά μηχανική μάθηση, Υπολογιστική τεχνητή νοημοσύνη, Μετάδοση γνώσης, Νευρωνικά δίκτυα, Συνελικτικά νευρωνικά δίκτυα, Νευρωνικά δίκτυα εντοπισμού αντικειμένων


## Abstract

In this dissertation there will be a description of computer vision (tricks) as well as the possibilities it offers in solving everyday problems. To be more specific, the concepts of training through artificial neural networks (TNN´s) will be developed, where the generated code will be written in Python programming language, as well as the technologies used in the visual recognition of chessboard and chess pieces with the aim of live transmission of chess pieces will be given. The algorithm we will use for the optical recognition process is YOLO.



## Key Words

Artificial Intelligence, Machine Learning, Deep Machine Learning, Computational Artificial Intelligence, Knowledge Transmission, Neural Networks, Coherent Neural Networks, Object Locating Neural Networks

# Κεφάλαιο 1ο

## 1.1 Εισαγωγή

Η μεγάλη, εκθετική και εντυπωσιακή ανάπτυξη της τεχνολογίας και η συνεχής ώθηση της επιστήμης στα όρια της προσφέρει την τεράστια δυνατότητα να δημιουργήσει ο άνθρωπος δρομολογητές που δρουν κινούνται αλλά και σκέφτονται σαν πραγματικοί άνθρωποι. Μελετώντας της βιολογικές λειτουργίες του ανθρωπίνου εγκέφαλου προσομοιώσαμε την λειτουργία του σε τεχνικά νευρονικά δίκτια (ΤΝΔ) που είναι ικανά να εκπαιδεύονται και να περνούν αποφάσεις μέσω ένα σύνολο δεδομένων που θα τους δοθούν.

Η παραπάνω διαδικασία είναι γνωστή ως τεχνητή νοημοσύνη. Η δυνατότητα αυτή έχει ξεκινήσει ως θεωρία μερικές δεκαετίες πριν, αλλά τα τελευταία χρόνια με τη συνεχή αύξηση της υπολογιστικής ισχύος, τείνει να γίνει πράξη και να εδραιωθεί στην καθημερινή ζωή.

Η Τεχνητή Νοημοσύνη είναι ο κλάδος/τομέας της επιστήμης της πληροφορικής, που ασχολείται με την σχεδίαση και κατασκευή ευφυών συστημάτων, δηλαδή συστημάτων που διαθέτουν χαρακτηριστικά που σχετίζονται με την ανθρώπινη νοημοσύνη και συμπεριφορά. Ο πιο ισχυρός υπολογιστής που γνωρίζει ο άνθρωπος μέχρι στιγμής είναι ο ίδιος ο εγκέφαλός του (όπως βέβαια και οι εγκέφαλοι άλλων ζώων). Αυτός για να φτάσει στο σημείο να χαίρει της τωρινής υπολογιστικής του ικανότητας χρειάστηκε εκατομμύρια χρόνια βιολογικής εξέλιξης. Ακόμα και με αυτή την υπολογιστική ικανότητα που διαθέτει συνεχίζει να χρησιμοποιεί τη τεχνική μετάδοσης γνώσης, παρά τα χρόνια εξέλιξής του. Αυτό σύμφωνα με τη δαρβινική θεωρία [1] τη φυσικής επιλογής, δείχνει πως ήταν μία βελτιστοποίηση στη λειτουργία του (δεν ισχυρίζεται η τελειότητα αυτού) η οποία δοκιμάστηκε και επέζησε μέχρι στιγμής.

Ως εκ τούτου, ο σχεδιασμός αλγορίθμων και μοντέλων που εκπαιδεύονται μέσω της μετάδοσης γνώσης είναι μία αρκετά δοκιμασμένη επιλογή και μία αρκετά δικαιολογημένη απόφαση. Ωστόσο, έχοντας πάλι ως παράδειγμα τον εγκέφαλο, τα νευρωνικά δίκτυα εντοπισμού αντικειμένων θα έπρεπε να απαιτούν λιγότερες παραμέτρους και να επεξεργάζονται την είσοδό τους σε πραγματικό χρόνο (τουλάχιστον 24 καρέ /δευτερόλεπτο). Η τελευταία αυτή απαίτηση σε συνδυασμό με την προηγούμενη οδηγούν στη διαμόρφωση του προβλήματος που περιγράφεται παρακάτω.

Επιπλέον, η μετάδοση γνώσης είναι ένα από τα εργαλεία που είναι στενά συνδεδεμένο με την έννοια της Γενικής Τεχνητής νοημοσύνης (Artificial General Intelligence). Αυτό συμβαίνει γιατί μοντελοποιείται ένα πολύ σημαντικό κομμάτι της διαδικασίας του εγκεφάλου: η ίδια η μάθηση.

Ένας σημαντικός τομέας που ωθεί τα σύγχρονα επιτεύγματα στη τεχνητή νοημοσύνη είναι η εμφάνιση και εξέλιξη του κλάδου της Βαθιάς Μηχανικής Μάθησης (Deep learning - DL).Η χρήση τεχνικών βαθιάς μάθησης στην επίλυση προβλημάτων Μηχανικής Όρασης, έχει κατορθώσει να αντιμετωπίσει περίπλοκα προβλήματα τα οποία μέχρι και πριν από λίγα χρόνια θεωρείτο ακατόρθωτο να λυθούν. Ακόμα και η χρήση της DL σε ενσωματωμένα συστήματα με μικρή μνήμη και απαιτήσεις ταχύτατης επεξεργασίας δίνει καλύτερα αποτελέσματα από ότι άλλοι αλγόριθμοι. Σήμερα, το γενικότερο πρόβλημα της ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες χρησιμοποιεί εκτεταμένα Νευρωνικά Δίκτυα Συνέλιξης (Convolutional Neural Networks - CNNs). Η εκπαίδευση και η επανεκπαίδευση αυτών έχει επισημανθεί ότι ωφελείται αρκετά από τη χρήση τεχνικών μετάδοση γνώσης [2]. Σε αυτό παίζουν ρόλο και άλλοι οικονομοτεχνικοί λόγοι όπως το μέγεθος των διαθέσιμων δεδομένων. Σε συνδυασμό όλων των παραπάνω η εργασία καλείται να δώσει κάποιες παρατηρήσεις προς την επίλυση του παρακάτω προβλήματος.

## 1.2 Περιγραφή του Προβλήματος

Η απλοποίηση επαναλαμβανομένων εργασιών είναι κοινή συμπεριφορά τον ανθρώπων επειδή συνήθως προτιμάνε να επικεντρωθούν σε προκλήσεις που απαιτούνε υψηλή δημιουργικότητα. Ως εκ τούτου, μηχανές χρησιμοποιούνται ευρέως για την αυτοματοποίηση τέτοιον εργασιών. Ένα πρόβλημα που θα μπορούσε να αυτοματοποιηθεί συμφώνα με το παραπάνω είναι ψηφιοποιήσει του φυσικού κόσμου. Η ψηφιοποιήσει είναι μια διαδικασία που συχνά είναι πολύ χρονοβόρα δεν απατάει εξειδικευμένη γνώση και εξακολουθεί να εκτελείται ή να υποστηρίζεται ουσιαστικά από τους ανθρώπους. Με τις εξελίξεις αλγόριθμόν οπτικής αναγνώρισης και της τεχνητής νοημοσύνης μείωσε την ανάγκη σε ανθρώπινη βοήθεια σε αλγορίθμους τέτοιους [1].

## 1.3 Σκοπός-Συνεισφορά της Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία κάνει μια ιστορική αναδρομή στις τεχνικές οπτικής αναγνώρισης όπως και μελετά τις μεθόδους που γίνονται εφικτές αυτές. Βασικός σκοπός είναι η ανάπτυξη νευρωνικών δικτύων συνέλιξης (CNN) για εφαρμογές ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων (object recognition and localization - object detection), χρησιμοποιώντας τον state of the art αλγόριθμο YOLO. Σκοπός είναι η οπτική αναγνώριση σκακιέρας και σκακιστικών κομματιών με στόχο την ζωντανή μετάδοση παρτίδων σκάκι, το να μετατρέψουμε μια φυσική σκακιέρα σε μια ψηφιακή.

Βασικός σκοπός όμως είναι η ανάπτυξη νευρωνικών δικτύων συνέλιξης (CNN) για εφαρμογές ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων (object recognition and localization - object detection) σε εικόνες χρησιμοποιώντας τον state of the art (υπερσύγχρονες μεθόδους) αλγόριθμο YOLO.

## 1.4 Διάρθρωση της αναφοράς

Η διάρθρωση της παρούσας διπλωματικής εργασίας είναι η εξής:

• Κεφάλαιο 2  Ορισμοί ΤεχνητήςΤεχνητής νοημοσύνης Βαθιάς μάθησης ιστορική αναδρομή

• Κεφάλαιο 3 Υπολογιστική Όραση (Computer Vision) Τεχνικές και αλγόριθμοι οπτικής αναγνώρισης

• Κεφάλαιο 4 ο αλγόριθμος YOLO

• Κεφάλαιο 5 Αλγόριθμος

• Κεφάλαιο 6 Αναφέρονται τα προβλήματα που προέκυψαν, συμπεράσματα  και προτείνονται θέματα για μελλοντική μελέτη, αλλαγές και επεκτάσεις.

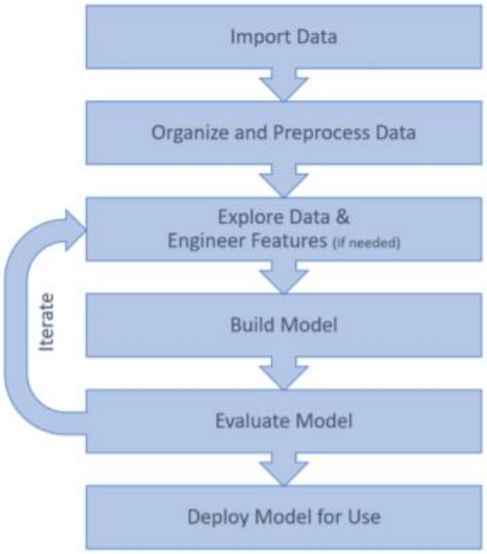



***Figure 3:*** *Overview of the ML Classification Workflow [1]*

*Εικόνα 1.1. Σχηματική απεικόνιση της λειτουργίας ML*

# Κεφαλαίο 2ο

**Εισαγωγή**

Η τεχνολογική έξαρση τον τελευταίων δεκαετιών, προσφέρει την τεράστια δυνατότητα να δημιουργήσει ο άνθρωπος ένα αυτόνομο να πράττει, αλλά ακόμα πιο σημαντικό, να σκέφτεται, δημιούργημα, όμοιο με τον άνθρωπο, αλλά δημιουργημένο με κώδικα σε γλώσσα προγραμματισμού. Το παραπάνω είναι γνωστό σαν τεχνητή νοημοσύνη. Η δυνατότητα αυτή έχει ξεκινήσει ως θεωρία μερικές δεκαετίες πριν, αλλά τα τελευταία χρόνια με τη συνεχή αύξηση της υπολογιστικής ισχύος, τείνει να γίνει πράξη και να εδραιωθεί στην καθημερινή ζωή. Οι τομείς στους οποίους εφαρμόζεται η τεχνητή νοημοσύνη είναι πάρα πολλοί, καθώς εφαρμόζονται σε ολοένα περισσότερες πτυχές και εφαρμογές της καθημερινότητας, τόσο στις επιχειρήσεις όσο και στον ιδιωτικό βίο. Η τεχνητή νοημοσύνη επηρεάζει, ανατρέπει και διαμορφώνει τη ζωή, εισάγοντας νέα δεδομένα στην αντιμετώπιση τετριμμένων και μη προβλημάτων, που άλλοτε η λύση τους, φαινόταν πολλές φορές αδύνατη.

Ο όρος Τεχνητή νοημοσύνη γνωστή ως AI εμφανίστηκε για πρώτη φορά το 1956 σε ένα μικρό συνέδριο στο Dartmouth College, New Hampshire . Από τότε, πολλές επιστήμες σε πολλούς τομείς όπως οι επιστήμες υπολογιστών ή ακόμη και στη φιλοσοφία εξακολουθούν να προσπαθούν να προσδιορίσουν το τι είναι η τεχνίτη νοημοσύνη.

## 2.1 Ορισμός τεχνητής νοημοσύνης

Τις τελευταίες δεκαετίες έχουν εμφανιστεί διάφοροι ορισμών της τεχνητής νοημοσύνης (AI) . ο John McCarthy προσφέρει τον ακόλουθο ορισμό σε αυτό το έγγραφο το 2004 [24] "*Τεχνίτη νοημοσύνη είναι η επιστήμη και η μηχανική στο να φτιάχνεις ευφυείς μηχανές, ειδικά ευφυή υπολογιστικά προγράμματα. Σχετίζετε με την χρήσης υπολογιστών για την κατανόηση της ανθρώπινης νοημοσύνης, αλλά η τεχνητή νοημοσύνη δεν χρειάζεται να περιοριστεί σε μεθόδους που είναι βιολογικά παρατηρήσιμες*".

Ωστόσο, πριν από αυτόν τον ορισμό, η γέννηση του ορού τεχνίτη νοημοσύνη πρωτοεμφανίστηκε σε ένα σεμινάριο από τον Alan Turing, οπού και εκδόθηκε το 1950. Σε αυτό το έγγραφο [3] ο Turing που συχνά αναφέρονταν ως πατέρας της επιστήμη των υπολογιστών, θετή την ερώτηση στο αν μπορούν οι μηχανές να σκεφτούν. Έτσι, κάνει μια

δοκιμή, γνωστό τώρα ως "Turing Test", όπου ένας ανθρώπινος ανακριτής θα προσπαθούσε να διακρίνει μεταξύ ενός υπολογιστή και μιας ανθρώπινης συγγραφείς κειμένου. Έκτοτε η δοκιμή αυτή έχει συζητηθεί αρκετά αλλά παραμένει ένα βασικό κομμάτι της ιστορίας την τεχνητής νοημοσύνης.

Στη συνέχεια, οι Stuart Russell και Peter Norvig προχώρησαν στη δημοσίευση, με τίτλο Artificial Intelligence: A Modern Approach [7], καθιστώντας ένα από τα κορυφαία εγχειρίδια στη μελέτη της τεχνητής νοημοσύνης. Σε αυτό, ερευνούν τέσσερις πιθανούς στόχους ή ορισμούς της τεχνητής νοημοσύνης, που διαφοροποιεί τα συστήματα υπολογιστών με βάση τον ορθολογισμό και τη σκέψη, έναντι της δράσης:

**Ανθρώπινη προσέγγιση:**

-Συστήματα που σκέφτονται σαν τους ανθρώπους

-Συστήματα που δρουν σαν άνθρωποι

**Ιδανική προσέγγιση**:

Συστήματα που σκέφτονται ορθολογικά

Συστήματα που λειτουργούν ορθολογικά

Στην απλούστερη μορφή του, η τεχνητή νοημοσύνη είναι ένας τομέας, ο οποίος συνδυάζει την επιστήμη των υπολογιστών και τα στιβαρά σύνολα δεδομένων, για την επίλυση προβλημάτων. Περιλαμβάνει επίσης κάποια υπο-πεδία της μηχανικής μάθησης (Machine Learning)και της βαθιάς μάθησης (Deep Learning), τα οποία αναφέρονται συχνά σε συνδυασμό με την τεχνητή νοημοσύνη. Αυτοί οι κλάδοι αποτελούνται από αλγόριθμους AI που επιδιώκουν τη δημιουργία ειδικών συστημάτων που κάνουν προβλέψεις ή ταξινομήσεις με βάση δεδομένα εισόδου χρησιμοποιούνται στην αντίληψη μέσω της όρασης, την μάθηση, την εξαγωγή συμπερασμάτων, την κατανόηση της φυσικής γλώσσας κτλ”

## 2.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning) αποτελεί ένα παρακλάδι της Τεχνητής Νοημοσύνης. Το όνομα αποδόθηκε από τον Arthur Samuel το 1959 [5] και βασίζεται στην

καινοτόμα προσέγγιση με βάση την οποία οι υπολογιστές αντί να προγραμματίζονται ντετερμινιστικά για τον τρόπο εκτέλεσης εντολών, να μπορούν από μόνοι τους να διδαχθούν πως θα πρέπει να δράσουν. Είναι άρρηκτα συνδεδεμένη με την επιστήμη της Υπολογιστικής Στατιστικής (Computational statistics) καθώς η διαδικασία εξέλιξής της συχνά βασίζεται σε μαθηματικές βελτιστοποιήσεις. Ο τυπικός ορισμός δόθηκε από τον Tom M. Mitchell και είναι ο εξής: ”Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία E ως προς μια κλάση εργασιών T και ένα μέτρο επίδοσης P, αν η επίδοσή του σε εργασίες της κλάσης T, όπως αποτιμάται από το μέτρο P, βελτιώνεται με την εμπειρία E” [6]. Στόχος της αποτελεί η διεύρυνση του παραδοσιακού προγραμματισμού στη βάση μιας λογικής όπου ένα πρόγραμμα μηχανικής μάθησης να μπορεί αξιοποιώντας τους διαθέσιμους αλγορίθμους να προσαρμόζεται στο εκάστοτε πρόβλημα. Παραδείγματα μηχανικής μάθησης που ήδη έχουμε πρακτικές εφαρμογές είναι η αναγνώριση ομιλίας και γραφικού χαρακτήρα αλλά και η αυτόματη οδήγηση αυτοκινήτων. Η μηχανική μάθηση ταξινομούνται σε 3 βασικές κατηγορίες ανάλογα με την φύση των δεδομένων εισόδου και την ανατροφοδότηση σε ένα σύστημα εκμάθησης [7]:

• Επιτηρούμενη μάθηση (Supervised learning): Το υπολογιστικό σύστημα δέχεται τα δεδομένα εισόδου μαζί με τις ετικέτες του, δηλαδή τα επιθυμητά χαρακτηριστικά, και ο στόχος είναι να μάθει έναν γενικό κανόνα προκειμένου να αντιστοιχίσει τις εισόδους με τα αποτελέσματα.

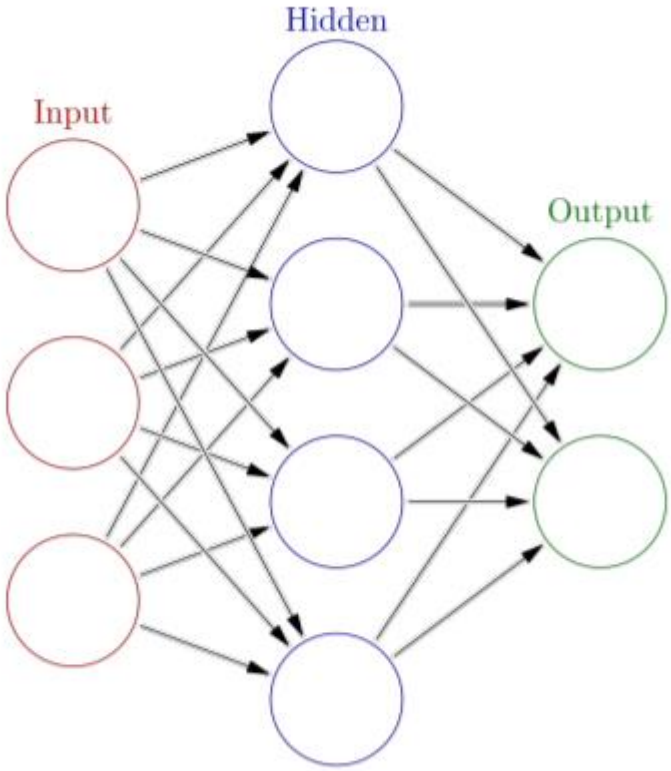
• Μη επιτηρούμενη μάθηση (Unsupervised learning): Στο σύστημα δεν παρέχονται μαζί με τα δεδομένα εισόδου ετικέτες, δηλαδή επιπρόσθετες πληροφορίες, και καλείται να βρει τη δομή των δεδομένων. Η μη επιτηρούμενη μάθηση μπορεί να χρησιμοποιηθεί για την ανακάλυψη κρυμμένων μοτίβων σε δεδομένα ή ως μέσο για την εύρεση κάποιου χαρακτηριστικού.

• Ενισχυτική μάθηση (Reinforcement learning): Ουσιαστικά πρόκειται για συνδυασμό των προηγούμενων δύο, καθώς το σύστημα λαμβάνει δεδομένα εισόδου χωρίς ετικέτες και προσπαθεί να εξάγει χαρακτηριστικά. Σε περίπτωση που το κάνει με επιτυχία ”επιβραβεύεται” ενώ αντίθετα δέχεται ”τιμωρία”. Στόχος είναι η μεγιστοποίηση της ανταμοιβής.

Στην παρούσα εργασία θα ασχοληθούμε μόνο με συστήματα επιτηρούμενης μάθησης, εξ ου θα γίνει και ανάλυση μόνο αυτού του πεδίου.

## 2.3 Τεχνητά Νευρωνικά Δίκτυα και Βαθιά Μάθηση

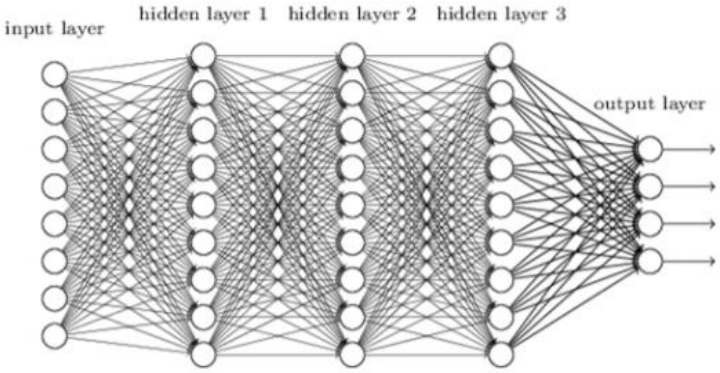
Το τεχνητό νευρωνικό δίκτυον (Artificial Neural Network) είναι ένα δίκτυο από απλούς υπολογιστικούς κόμβους (νευρώνες), διασυνδεδεμένους μεταξύ τους. Κάθε νευρώνας δέχεται ένα σύνολο εισόδων από διαφορετικές πηγές και παράγει μία έξοδο. Πολλές αλγοριθμικές στρατηγικές που αφορούν τη μηχανική μάθηση μπορούν να μοντελοποιηθούν με τη χρήση νευρώνων. Οι νευρώνες κατανέμονται σε διάφορα επίπεδα, τα εσωτερικά ονομάζονται και κρυφά (hidden), επιτελώντας ξεχωριστές λειτουργίες. Η απρόσεκτη αύξηση επιπέδων μπορεί μεν να επιφέρει μεγαλύτερη ακρίβεια αλλά παράλληλα να αυξήσει και την πολυπλοκότητα σε τέτοιο βαθμό που να καθιστά το σύστημα μη αποδοτικό στην πράξη.




*Εικόνα 2.1. Απεικόνιση απλού νευρωνικού δικτύου*

Η Βαθιά Μάθηση (Deep Learning) αποτελεί εξέλιξη των απλών νευρωνικών δικτύων. Ο όρος εισήχθη από την Rina Dechter το 1986 [9] στην μηχανική μάθηση και από τον Igor Aizenberg το 2000 στα τεχνητά νευρωνικά δίκτυα [8]. Σε αντίθεση με τα απλά νευρωνικά δίκτυα, στη βαθιά 19 μάθηση έχουμε πολύ περισσότερα κρυφά επίπεδα που αποσκοπούν στην εξαγωγή συμπερασμάτων με μεγαλύτερη ακρίβεια καθώς και εξαγωγή χαρακτηριστικών υψηλού

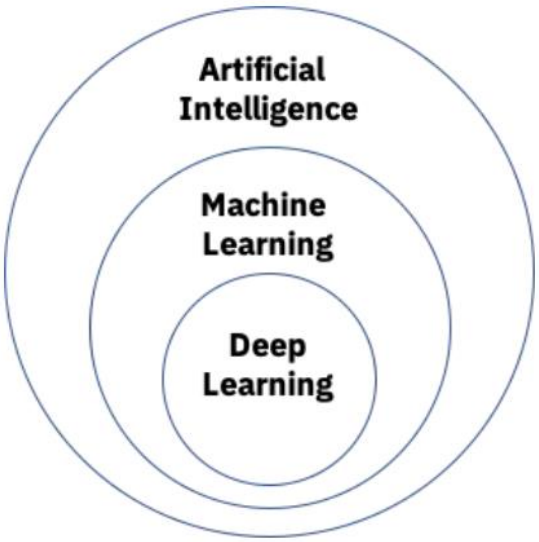
επιπέδου. Τα δίκτυα συνήθως λαμβάνουν πολυδιάστατα δεδομένα και σε συνδυασμό με την ύπαρξη πολλών επιπέδων, η επεξεργασία αυτών καθίσταται εξαιρετικά χρονοβόρα. Όμως, η ανάπτυξη της τεχνολογίας έχει φέρει ευεργετικές επιπτώσεις στο ζήτημα αυτό με την δημιουργία πολυπύρηνων αρχιτεκτονικών και την αξιοποίηση των νέων μονάδων γραφικής επεξεργασίας (GPUs) που πλέον χρησιμοποιούνται κατά κόρον, μειώνοντας δραστικά το χρόνο εκπαίδευσης τέτοιων πολύπλοκων δικτύων.




*Εικόνα 2.2. Βαθύ νευρωνικό δίκτυο*

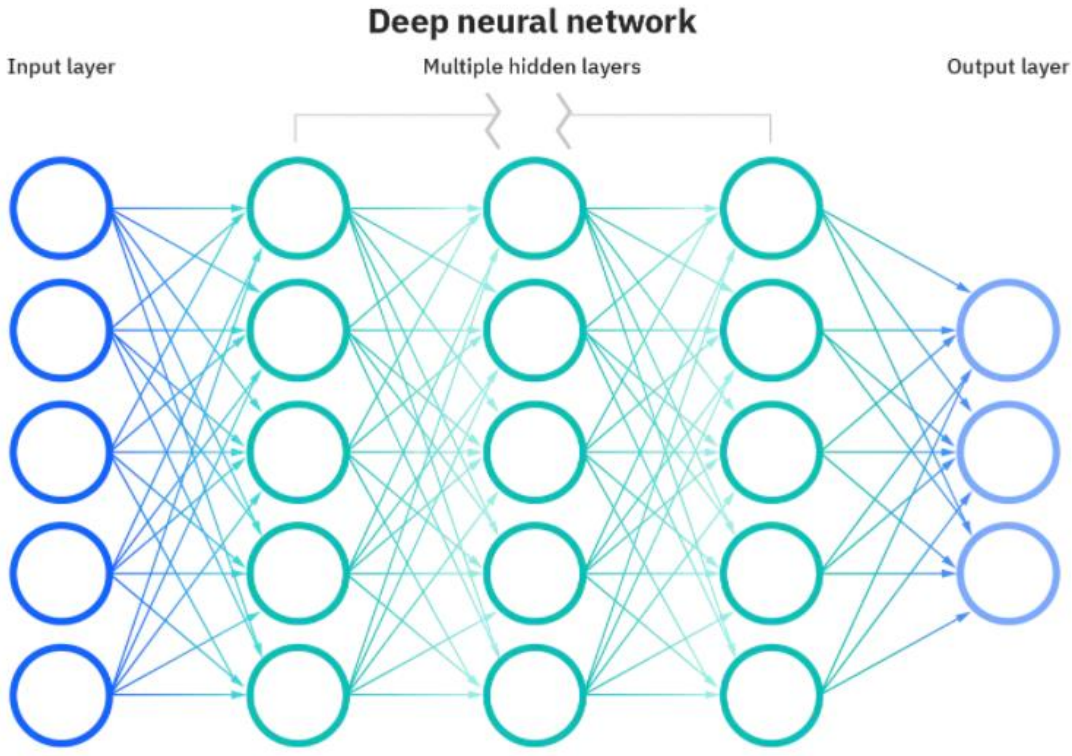
## 2.4 Διαφορά βαθιάς μαθήσεις και μηχανικής

Δεδομένου ότι η βαθιά μάθηση και η μηχανική μάθηση τείνουν να χρησιμοποιούνται εναλλακτικά, αξίζει να σημειωθούν οι διαφοροποιήσεις μεταξύ των δύο. Όπως αναφέρθηκε παραπάνω, τόσο η βαθιά μάθηση όσο και η μηχανική μάθηση είναι υπο-τομείς της τεχνητής νοημοσύνης και η βαθιά μάθηση είναι στην πραγματικότητα ένας υπο-τομέας της μηχανικής μάθησης.




*Εικόνα 2.3. Χάσμα AI μηχανικής μαθήσεις και βαθιάς μαθήσεις*

Η βαθιά μάθηση αποτελείται από νευρωνικά δίκτυα. Το "Deep" στη βαθιά μάθηση αναφέρεται σε ένα νευρωνικό δίκτυο που αποτελείται από περισσότερα από τρία επίπεδα, το οποίο θα περιλαμβάνει τις εισόδους και την έξοδο. Αυτό μπορεί να θεωρηθεί αλγόριθμος βαθιάς μάθησης. Αυτό αντιπροσωπεύεται γενικά χρησιμοποιώντας το ακόλουθο διάγραμμα:



*Εικόνα 2.4. Βαθιά νευρωνικά δίκτυα*

Ο τρόπος με τον οποίο η βαθιά μάθηση και η μηχανική μάθηση διαφέρουν είναι ο τρόπος με τον οποίο μαθαίνει κάθε αλγόριθμος. Η βαθιά μάθηση αυτοματοποιεί μεγάλο μέρος της διαδικασίας εξαγωγής χαρακτηριστικών (feature extraction) της διαδικασίας, εξαλείφοντας μέρος της χειροκίνητης ανθρώπινης παρέμβασης που απαιτείται και επιτρέποντας τη χρήση μεγαλύτερου όγκου δεδομένων. Μπορείτε να σκεφτείτε τη βαθιά μάθηση ως «επεκτάσιμη μηχανική μάθηση» όπως σημείωσε ο Lex Fridman [10]. Η κλασική, ή "μη βαθιά", μηχανική μάθηση εξαρτάται περισσότερο από την ανθρώπινη παρέμβαση για μάθηση.

Η αναφερόμενη ως "βαθιά" μηχανική εκμάθηση μπορεί να αξιοποιήσει τα σύνολα δεδομένων με ετικέτα, επίσης γνωστά ως εποπτευόμενη μάθηση (supervised learning), για να ενημερώσει τον αλγόριθμό του, αλλά δεν απαιτεί απαραίτητα ένα σύνολο δεδομένων με ετικέτα (labeled). Μπορεί να απορροφήσει μη δομημένα δεδομένα στην αρχική του μορφή (π.χ. κείμενο, εικόνες) και μπορεί να καθορίσει αυτόματα την ιεραρχία χαρακτηριστικών που διακρίνουν διαφορετικές κατηγορίες δεδομένων μεταξύ τους. Σε αντίθεση με τη μηχανική μάθηση, δεν απαιτείται ανθρώπινη παρέμβαση για την επεξεργασία δεδομένων, επιτρέποντάς μας να κλιμακώσουμε τη μηχανική μάθηση με πιο ενδιαφέροντες τρόπους.

## 2.5 Σύντομη Ιστορία τεχνητής νοημοσύνης

Οι πρώτες ανάδρομες στην τεχνίτη νοημοσύνη ξεκινούν με μύθους από την αρχαιά Ελλάδας , από τον μπρούτζινο αυτοματοποιημένο γίγαντα Ταλο προστάτη της Κρήτης και της Ευρώπης και αργότερα από τον Αριστοτέλη που ανέπτυξε τον συλλογισμό που θεωρήθηκε κλειδί για την ανθρωπότητα προς κατανόηση την ανθρώπινης νοημοσύνης. Παρόλο που οι ρίζες του ορού της τεχνητής νοημοσύνη είναι παλιές και βαθιά ρυαζόμενες στο παρελθόν, η τεχνίτη νοημοσύνη όπως την ξέρουμε σήμερα υπάρχει για λιγότερο από έναν αιώνα. Παρακάτω θα δούμε μερικές στιγμές κλειδί της ιστορίας της τεχνητής νοημοσύνης τα τελευταία χρονιά

- 1950: Ο Άλαν Τούρινγκ γνωστός για την αποκρυπτογράφηση του κωδικού Enigma στο Β παγκόσμιο πόλεμό ,εκδίδει το Computing Machinery and Intelligence, προτείνετε να απάντηση στο ερώτημα “Μπορούν οι μηχανές να σκεφτούν;” Ο Άλαν εισήγαγε το

τεστ Τούρινγκ για να προσδιορίσει εάν ένας υπολογιστής μπορεί να αποδείξει την ίδια νοημοσύνη (ή τα αποτελέσματα της ίδιας νοημοσύνης) με τον άνθρωπο. Η αξία του τεστ Τούρινγκ έχει συζητηθεί έκτοτε. [1]

- 1956 : Ο Τζον Μακάρθι εισήγαγε τον όρο «τεχνητή νοημοσύνη» στο πρώτο συνέδριο AI στο Dartmouth College. (Ο McCarthy θα συνέχιζε να εφευρίσκει τη γλώσσα προγραμματισμού Lisp.) Αργότερα εκείνο το έτος, ο Allen Newell, ο J.C. Shaw και ο Herbert Simon δημιουργούν το Logic Theorist οπου ηταν το πρώτο λογισμικό τεχνητής νοημοσύνης - AI που έτρεξε ποτέ.

- 1967: Ο Frank Rosenblatt χτίζει το Mark 1 Perceptron, τον πρώτο υπολογιστή που βασίζεται σε ένα νευρωνικό δίκτυο που «εκπαιδεύτηκε» με την διαδικασία πειραματισμού και σφάλματος. Μόλις ένα χρόνο αργότερα, οι Marvin Minsky και Seymour Papert δημοσιεύουν ένα βιβλίο με τίτλο Perceptrons, το οποίο γίνεται ορόσημο για τα νευρονικά δίκτυα.

- 1980: Νευρονικά δίκτυα που χρησιμοποιούν τον αλγόριθμο οπισθοδρόμησης για να εκπαιδεύσουν τον εαυτό τους χρησιμοποιούνται ευρέως σε εφαρμογές AI.

- 1997: Το Deep Blue της IBM νικάει τότε τον παγκόσμιο πρωταθλητή σκακιού Garry Kasparov, σε έναν αγώνα σκακιού.

- 2015: Ο υπερυπολογιστής του Baidu Minwa χρησιμοποιεί ένα ειδικό είδος βαθύ νευρωνικού δικτύου που ονομάζεται convolutional neural network για τον αναγνώριση και την κατηγοριοποίηση εικόνων με υψηλότερο ποσοστό ακρίβειας από τον μέσο άνθρωπο.

- 2016: Το πρόγραμμα AlphaGo του DeepMind, που υποστηρίζεται από ένα βαθύ νευρωνικό δίκτυο, νικά τον Lee Sodol, τον παγκόσμιο πρωταθλητή Go player, σε έναν αγώνα πέντε παιχνιδιών. Η νίκη είναι σημαντική δεδομένου του τεράστιου αριθμού

πιθανών κινήσεων καθώς το παιχνίδι εξελίσσεται (πάνω από 14,5 τρισεκατομμύρια μετά από μόλις τέσσερις κινήσεις!). Αργότερα, η Google αγόρασε το DeepMind έναντι του ποσού των 400 εκατομμυρίων δολαρίων.

*Εικόνα 2.5. Kasparov εναντίον του υπολογιστή Deep Blue*

# Κεφάλαιο 3ο

## 3.1 Μηχανική όραση

Η μηχανική όραση είναι ένα πεδίο τεχνητής νοημοσύνης (AI) που επιτρέπει στους υπολογιστές και τα συστήματα να αντλήσουν σημαντικές πληροφορίες από ψηφιακές εικόνες, βίντεο και άλλες οπτικές εισόδους. Συγκριτικά εάν η τεχνητή νοημοσύνη επιτρέπει στους υπολογιστές να σκέφτονται, η μηχανική όραση τους επιτρέπει να βλέπουν, να παρατηρούν και να κατανοούν.

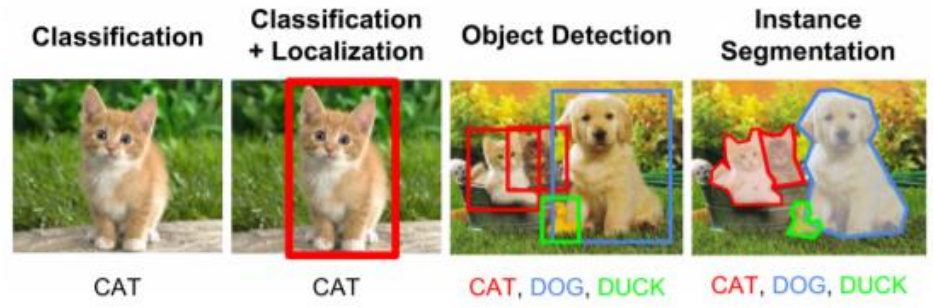
Η μηχανική όραση [12] είναι ένα διεπιστημονικό πεδίο στο οποίο γίνεται προσπάθεια οι υπολογιστές να αποκτήσουν τη δυνατότητα κατανόησης αντικειμένων σε ψηφιακές εικόνες ή βίντεο. Επιχειρείται, λοιπόν, η αλγοριθμική αναπαραγωγή της αίσθησης της όρασης στο μέγεθος του ανθρώπινου επιπέδου, τουλάχιστον, σε μηχανικά συστήματα. Η εισαγωγή που περιεγράφηκε στα προηγούμενα κεφάλαια έθεσε τις βάσεις για την επεξήγηση της μηχανικής όρασης, των τεχνικών που εφαρμόζονται και τις δυνατότητες που προσφέρει η ανάπτυξη της τεχνολογίας αυτής. Η μηχανική όραση έχει πολύ μεγάλη σημασία σε ένα ευρύ φάσμα του ανθρώπινου πολιτισμού. Χρησιμοποιείται στην ψυχαγωγία όπως στις κονσόλες παιχνιδιών, στην ασφάλεια, με την αναγνώριση επικίνδυνων αντικειμένων ή συμπεριφορών και στην πρόληψη ατυχημάτων στο οδικό δίκτυο ή την ανάπτυξη των αυτόνομων οχημάτων. Έχει μεγάλη χρησιμότητα και στην ιατρική, όπου οι ακτινογραφίες και τομογραφίες αναλύονται σε βάθος και με λεπτομέρεια που πιθανόν δεν μπορεί να αντιληφθεί το ανθρώπινο μάτι, οδηγώντας στην έγκαιρη διάγνωση επιβλαβών ασθενειών. Ένας ακόμα τομέας είναι η βιομηχανία. Η μηχανική μάθηση χρησιμοποιείται στην ανίχνευση βεβλημένων αντικειμένων κατά τη διαδικασία παραγωγής τους, στην βαθμονόμηση των βιομηχανικών ρομπότ και στην απομάκρυνση ανεπιθύμητων ουσιών και υλικών από τα τρόφιμα της γεωργικής γραμμής παραγωγής. Τέλος, έχει μεγάλη εφαρμογή και στο στρατιωτικό τομέα με τα συστήματα μή επανδρωμένων αεροσκαφών, για την αναγνώριση αγνώστου εδάφους, εχθρικών οχημάτων και προσωπικού και άλλα στοιχεία εχθρικού ενδιαφέροντος. Η ανάπτυξη της τεχνολογίας αυτής συνεχώς εξελίσσεται, καθώς βασίζεται τόσο στο υλικό όσο και στο λογισμικό των συστημάτων που χρησιμοποιούνται. Συνεχής έρευνα γίνεται τόσο στην ανάπτυξη καμερών μεγαλύτερης ανάλυσης, τη χρησιμοποίηση ισχυρότερου υλικού για την ταχύτερη υλοποίηση του λογισμικού

αλλά και βελτίωσης των αλγορίθμων και μεθόδων αναγνώρισης των αντικειμένων ή ανάπτυξη νέων και πιο αποδοτικών.

## 3.2 Ανίχνευση Αντικειμένων

Η Ανίχνευση Αντικειμένων (Object Detection) [13] αποτελεί ένα βασικό πρόβλημα στην Όραση Υπολογιστή (Computer Vision) και στην Επεξεργασία Εικόνας (Image Processing) Ως στόχο έχει την απομόνωση στιγμιότυπων μιας εικόνας ή την ταξινόμησή εικόνων σε σημασιολογικές κλάσεις και την περαιτέρω επεξεργασία τους. Παραπλήσιος όρος είναι και η Κατάτμηση Εικόνας (Image Segmentation) [Chen80] όπου κατά την επεξεργασία η εικόνα διαιρείται σε μικρότερες περιοχές ενδιαφέροντος (regions of interest), το οποίο δε θα αποτελέσει αντικείμενο αυτής της διπλωματικής. Τόσο η ανίχνευση αντικειμένων όσο και η τμηματοποίηση εικόνας αποτελούν βασικά στάδια για την ερμηνεία τους και την εξόρυξη πληροφοριών από αυτές.

Αν και μια εικόνα μπορεί να χωριστεί σε λιγότερες ή περισσότερες περιοχές ενδιαφέροντος και να εφαρμοστεί αναλυτική ανίχνευση αντικειμένων, όπου το κάθε επιμέρους αντικείμενο θα πρέπει να χαρακτηρίζεται από τυπική ομοιογένεια η οποία μπορεί να ποικίλλει ανάλογα με τα δεδομένα εισόδου και τη φύση του προβλήματος, ενώ αντίστοιχα οι γειτονικές περιοχές να εμφανίζουν αισθητές και ευδιάκριτες διαφορές.




*Εικόνα 3.1. Παράδειγμα Ανίχνευσης Αντικειμένων και Κατάτμησης Εικόνας*

Σήμερα αυτές οι τεχνικές χρησιμοποιούνται σε μια σειρά από πρακτικές εφαρμογές. Πολύ συνοπτικά αναφέρουμε μερικές:

- Ανίχνευση και αναγνώριση προσώπου
- Αναγνώριση δακτυλικών αποτυπωμάτων
- Συστήματα ελέγχου οδικής κυκλοφορίας

- Διάγνωση ιατρικών ασθενειών

Η εξέχουσα σημασία αυτών των ζητημάτων για τη μηχανική μάθηση επαληθεύεται από την συνεχώς αυξανόμενη προβολή και χρηματοδότηση διαγωνισμών όπως ο ImageNet Large Scale Visual Recognition Competition (ILSVRC) που διεξάγεται σε ετήσια βάση και έχει ως αντικείμενο τα προβλήματα που θέσαμε.

## 3.3 Μοντέλα Ανίχνευσης Αντικειμένων

**Προκλήσεις αλγόριθμων μηχανικής όρασης**

1. Σύνολο αντικείμενων.

Η αναγνώριση αντικείμενων είναι η προσπάθεια ενός αλγορίθμου τεχνητής νοημοσύνης να εντοπίσει και να ταξινομήσει ένα σύνολο αντικείμενων που προέρχονται από μια εικόνα. Το σημαντικό σημείο είναι το μέγεθος του σύνολο. Το μέγεθος του συνόλου των αντικείμενων που θέλουμε να γίνει η αναγνώριση ποικίλουν από εικόνα σε εικόνα. Συνεπώς το πρόβλημα που συσχετίζεται με αυτό είναι ότι στα μοντέλα μηχανικής μαθήσεις συνηθίζεται να εκπροσωπούνται τα δεδομένα σε διανύσματα σταθερού μεγέθους. Μιας και το μέγεθος των αντικειμένων στην εικόνα μας είναι άγνωστα εκ τον προτέρων, δεν μπορούμε να ξέρουμε τον ακριβή αριθμό τον εξόδων του νευρωτικού μας μοντέλου γεγονός που ενδεχομένως μας οδηγεί να κάνουμε κάποια προεργασία το οποίο μας κάνει το τελικό μοντέλο πιο πολύπλοκο.

2. Διαστάσεις αντικείμενων.

Τα αντικείμενα στην εικόνα μπορούν να ποικίλουν σε μέγεθος και φάσματος διαστάσεων. Κάποια αντικείμενα μπορεί να καλύπτουν το μεγαλύτερο μέγεθος της εικόνας ενώ αλλά αντικείμενα που θέλουμε να ανιχνεύσουμε να αποτελούνται από ένα μικρό σύνολο pixels (να είναι πολύ μικρά σε μέγεθος). Ακόμη, το ίδιο αντικείμενο μπορεί να έχει διαφορετικά μεγέθη σε διαφορετικές εικόνες. Αυτή η ποικιλία και ιδιομορφία των αντικείμενων τα κάνουν πιο δύσκολα στην ανίχνευση τους.

3. Εντοπισμός θέσης.

Όταν κάνεις αναγνώριση πολλαπλών αντικείμενων προϋποθέτει την αναγνώριση και ταξινόμηση του εκάστου αντικείμενου, όπως και τον εντοπισμό της θέσης του μέσα στην εικόνα. Για την επίλυση του παραπάνω ζητήματος οι ερευνητές δοκίμασαν ένα πιο σύνθετο τρόπο υπολογισμού του σφάλματος που τιμωρούσε τόσο τα σφάλματα πάνω σε λάθη ταξινόμησης όσο και σε λάθη θέσης. Αυτή η δυαδική διαφοροποίηση στην προσέγγιση υπολογισμού του σφάλματος είχε ως αποτέλεσμα το μοντέλο να υπολειτουργεί τόσο στην ταξινόμηση τόσο και στην εύρεση θέσεως.

4. Περιορισμένο σύνολο σχολιασμένων δεδομένων.

Το πεπεισμένο σύνολο σχολιασμένων δεδομένων που είναι διαθέσιμα προς το παρόν για αναγνώριση αντικείμενων είναι άλλο ένα εμπόδιο στην διαδικασία. Τα σύνολα δεδομένων για την αναγνώριση αντικείμενων συνήθως περιέχουν σχολιασμένα παραδείγματα που μπορεί να φτάσουν από δύο μέχρι και δεκάδες χιλιάδες κλάσεις. Γεγονός που κάνει την συλλογή των ετικετών μαζί με τα κουτιά οριοθέτησης για κάθε κλάση να είναι μια πολύ κουραστική εργασία για επίλυση.

5. Ταχύτητα για αναγνώριση σε πραγματικό χρόνο.

Οι αλγόριθμοι ανίχνευσης αντικειμένων πρέπει να είναι όχι μόνο ακριβείς στην πρόβλεψη της κλάσης των αντικείμενων και των θέσεων τους, αλλά πρέπει επίσης να είναι απίστευτα γρήγοροι σε όλα αυτά για να ανταποκριθούν στις ανάγκες των απαιτήσεων επεξεργασίας βίντεο σε πραγματικό χρόνο. Συνήθως, ένα βίντεο βιντεοσκοπείται με ταχύτητα περίπου 24 FPS και για τη δημιουργία ενός αλγορίθμου που μπορεί να επιτύχει αυτό το ρυθμό καρέ είναι πολύ δύσκολο έργο.

**Διάφορες προσεγγίσεις για επίλυση των παραπάνω προβλημάτων.**

Εξετάσαμε ορισμένες από τις κύριες προκλήσεις στον τομέα της Ανίχνευσης αντικειμένων, τι είναι και πώς επηρεάζουν τη διαδικασία. Τώρα θα ρίξουμε μια ματιά σε ορισμένα μοντέλα που έχουν προσπαθήσει να λύσουν αυτές τις προκλήσεις πριν παρουσιάσουμε το καλύτερο από αυτά - τον αλγόριθμο YOLO.

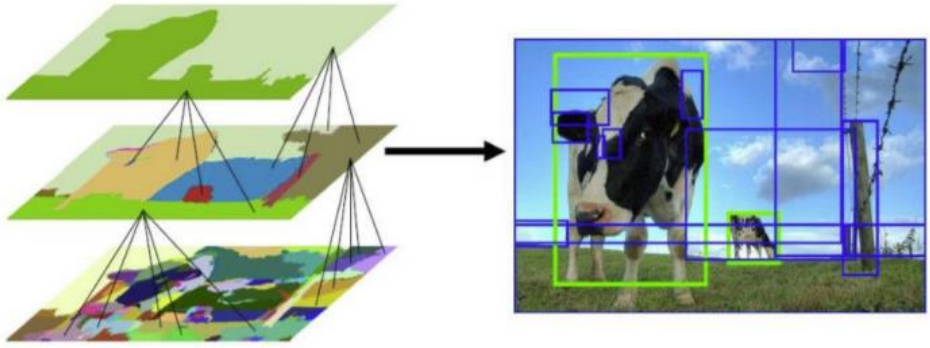
**R - CNN**

Οι χρησιμοποιούμενοι αλγόριθμοι είναι οι R - CNN, Fast R - CNN, Faster R - CNN, YOLO[12]. Ο R – CNN (Region CNN)[13] χρησιμοποιείται επιλεκτική αναζήτηση για να εξαχθούν μόνο 2000 περιοχές από την εικόνα και ονομάζονται προτάσεις περιοχής (region proposal). Επομένως, αντί να γίνει ταξινόμηση ενός τεράστιου αριθμού περιοχών, ο αλγόριθμος εργάζεται με 2000 περιοχές. Αυτές επιλέγονται χρησιμοποιώντας τον αλγόριθμο επιλεκτικής αναζήτησης ως εξής [14]:

1. Επειδή στις εικόνες υπάρχουν διαφορετικά χρώματα, υφές και περιοχές, ξεκινώντας από το χαμηλό επίπεδο προς το υψηλότερο, επιλέγονται πολύ μικρές περιοχές της εικόνας.

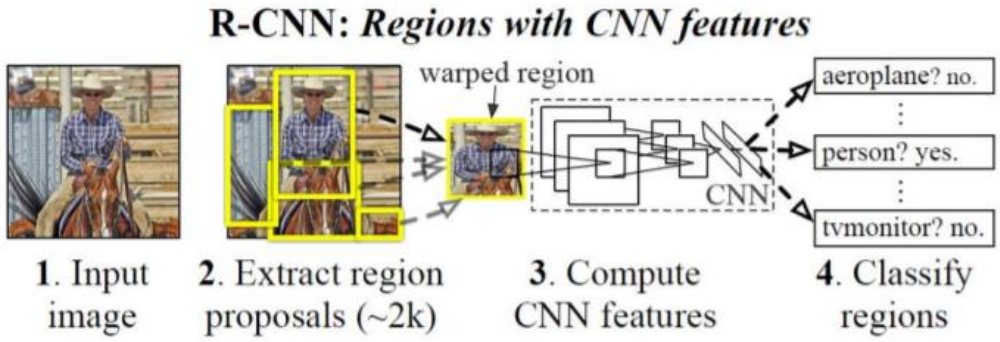
2. Με επαναληπτική διαδικασία, ενώνονται οι γειτονικές περιοχές που έχουν παρόμοιο χρώμα και υφή, δημιουργώντας μεγαλύτερες περιοχές

3. Δημιουργούνται τελικά 2000 περιοχές προς επεξεργασία.



*Εικόνα 3.2. Αλγόριθμος Επιλεκτικής Αναζήτησης*

Εν συνεχεία, όπως περιγράψαμε και πιο πάνω, οι περιοχές αυτές εισάγονται στο ΣΝΔ και τελικά ταξινομούνται στην έξοδο. Η Εικόνα 3.3 παρουσιάζει συνοπτικά το R – CNN.

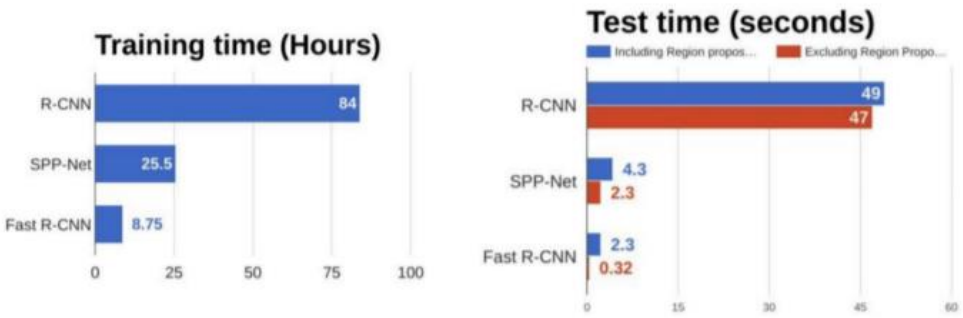



*Εικόνα 3.3. Ο Αλγόριθμος R – CNN*

Προβλήματα που σχετίζονται με τον αλγόριθμο είναι ο χρόνος εκπαίδευσης, ο οποίος είναι μεγάλος, καθώς χρειάζεται η ταξινόμηση 2000 περιοχών ανά εικόνα, δεν μπορεί να εφαρμοστεί σε πραγματικό χρόνο, αφού κάθε εικόνα χρειάζεται 47 δευτερόλεπτα για να ταξινομηθεί και τέλος, επειδή ο αλγόριθμος είναι στατικός, που σημαίνει ότι δεν γίνεται κάποια εκπαίδευση, μπορεί να οδηγηθεί σε λάθος προτάσεις περιοχών.

**Fast R – CNN**

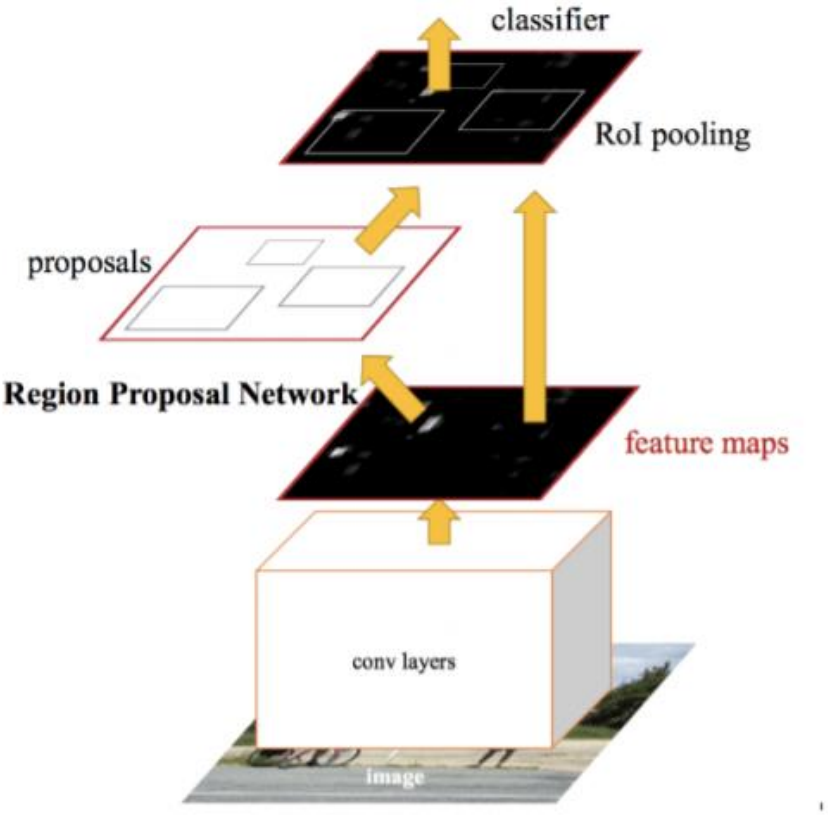
Fast R – CNN. Ο αλγόριθμος Fast R – CNN [16] διορθώνει τα προβλήματα (Εικόνα 26) με το να εισάγει την εικόνα στο ΣΝΔ ώστε να δημιουργηθεί ένας χάρτης χαρακτηριστικών (feature map), αντί να χρησιμοποιήσει ως είσοδο τις προτάσεις περιοχών. Από το χάρτη χαρακτηριστικών, εντοπίζονται οι προτάσεις περιοχών και χρησιμοποιούνται ως είσοδος σε συγκεντρωτικά επίπεδα και στη συνέχεια στο πλήρες συνδεδεμένο επίπεδο, κατά τα γνωστά. Ο Fast R-CNN είναι ταχύτερος από τον R-CNN διότι δεν τροφοδοτούνται κάθε φορά 2000 περιοχές στο ΣΝΔ, αλλά η συνέλιξη γίνεται μία φορά ανά εικόνα κατά την οποία εξάγεται ο χάρτης περιοχής




*Εικόνα 3.4. Σύγκριση R - CNN & Fast R – CNN*

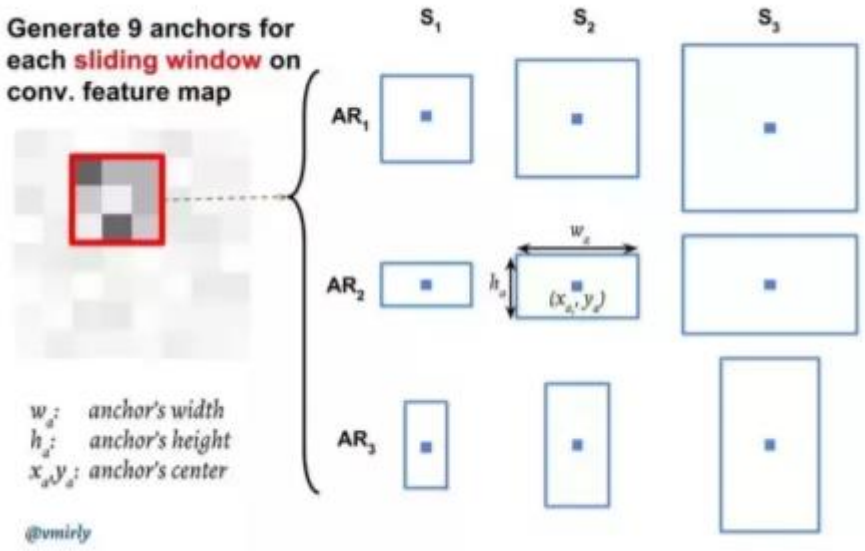
**Faster R – CNN**

Η κύρια διαφορά είναι ότι το R-CNN εισάγει τις προτάσεις περιοχής για ανίχνευση στο CNN σε επίπεδο pixel, ενώ η Fast R-CNN εισάγει τις προτάσεις περιοχής σε επίπεδο χάρτη χαρακτηριστικών. Το Faster R-CNN [16] χρησιμοποιεί την επιλεκτική αναζήτηση. Με το να χρησιμοποιηθεί ένα μικρό συνελικτικό δίκτυο που ονομάζεται Δίκτυο Προτάσεων Περιοχών (Regional Proposal Network - RPN), η ταχύτητα αυξάνεται ακόμα περισσότερο. Για να χειριστείτε τις παραλλαγές του λόγου διαστάσεων (aspect ratio) και της κλίμακας αντικειμένων (scale of objects), το Faster R-CNN χρησιμοποιεί τα κιβώτια αγκύρωσης (anchor boxes). Σε κάθε θέση, χρησιμοποιούνται 3 είδη κιβωτίων αγκύρωσης για κλίμακες 128x128, 256x256 και 512x512. Ομοίως, για λόγο διαστάσεων, χρησιμοποιεί τρεις αναλογίες διαστάσεων 1: 1, 2: 1 και 1: 2. Έτσι, συνολικά σε κάθε θέση, υπάρχουν 9 πλαίσια στα οποία το RPN προβλέπει την πιθανότητα να είναι ένα φόντο ή να βρίσκεται μπροστά. Στη συνέχεια, εφαρμόζεται επαναληπτικά ο ορισμός κιβωτίων οριοθέτησης για να βελτιωθούν αυτά σε κάθε θέση. Επομένως, το RPN εξάγει πλαίσια οριοθέτησης διαφόρων μεγεθών με τις αντίστοιχες πιθανότητες κάθε τάξης. Τα ποικίλα μεγέθη των πλαισίων οριοθέτησης μπορούν να γίνουν είσοδος στο υπόλοιπο δίκτυο το οποίο είναι παρόμοιο με το Fast-RCNN. Έτσι, το δίκτυο RPN πρέπει να ελέγξει εκ των προτέρων ποια τοποθεσία περιέχει αντικείμενο. Και οι αντίστοιχες θέσεις και τα κιβώτια οριοθέτησης θα περάσουν στο δίκτυο ανίχνευσης για την ανίχνευση της κλάσης αντικειμένου και την επιστροφή του πλαισίου οριοθέτησης αυτού του αντικειμένου. Στην Εικόνα 28 παρουσιάζεται ο αλγόριθμος Faster R – CNN.




*Εικόνα 3.5. Η διαδικασία που ακολουθείται στο Faster R-CNN.*

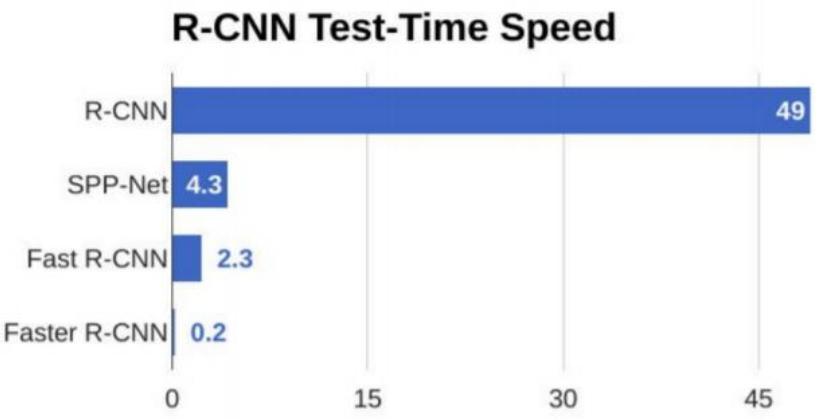
Ο χάρτης χαρακτηριστικών τροφοδοτείται στο υποδίκτυο RPN και έπειτα γίνεται η τελική αξιολόγηση.




*Εικόνα 3.6. Χρήση anchor boxes*

Στο σχήμα που αναπαριστάνει η εικόνα 3.6 βλέπουμε ότι η προκαθορισμένη επιλογή είναι η χρήση εννέα anchor boxes, τριών σχημάτων σε διαφορετικές διαστάσεις τα οποία εξετάζονται σε κάθε θέση του χάρτη χαρακτηριστικών. Ο αριθμός αυτός μπορεί να αλλάξει ανάλογα με τις ανάγκες και τις απαιτήσεις του κάθε προβλήματος.

Όπως φαίνεται στην Εικόνες 3.7 , ο Faster R - CNN είναι 10 φορές ταχύτερος από το Fast R – CNN. Αυτός είναι ο λόγος για τον οποίο το Faster-RCNN υπήρξε ένας από τους πιο ακριβείς αλγόριθμους ανίχνευσης αντικειμένων, καθώς, επίσης, μπορεί να χρησιμοποιηθεί για την ανίχνευση αντικειμένων σε πραγματικό χρόνο



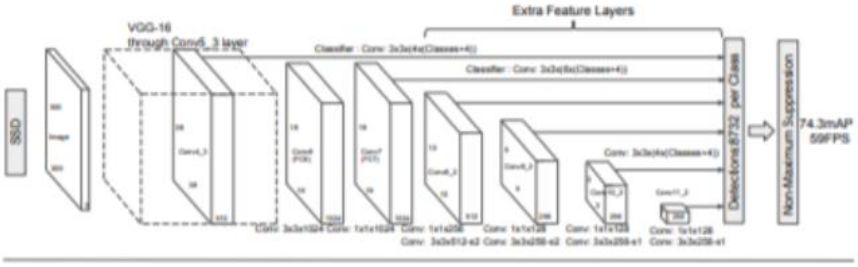

**Εικόνα 28. Σύγκριση Faster R - CNN με άλλους Αλγόριθμους**

| | R-CNN | Fast R-CNN | Faster R-CNN |
|---|---|---|---|
| Test Time per Image | 50 Seconds | 2 Seconds | 0.2 Seconds |
| Speed Up | 1x | 25x | 250x |

*Εικόνα 3.7. Σύγκριση Faster R - CNN με άλλους Αλγόριθμους*

**2. Single Shot MultiBox Detector (SSD)**

Το Single Shot MultiBox Detector (SSD) λειτουργεί με την μέθοδο free-forward convolution layer. Αυτή η μέθοδος επιτρέπει την εξαγωγή δεδομένων που έχουν συλλεγεί, θέτοντας κάποια όρια και σημειώνοντας μια βαθμολογία για την κλάση των αντικειμένων μέσα σε αυτά τα όρια. Ταυτόχρονα χρησιμοποιεί μια Μη-Μέγιστη (Non-Max Suppression) καταστολή για τη λήψη των τελικών αποφάσεων.

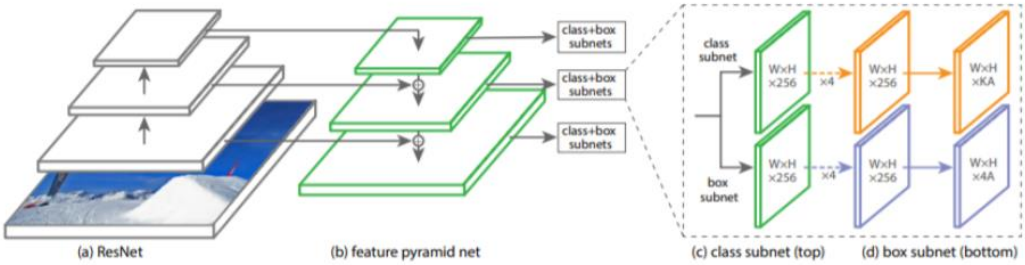



*Εικόνα 3.8. Η αρχιτεκτονική του αισθητήρα SSD Multibox*

Η αρχιτεκτονική του SSD είναι αρκετά απλή. Τα αρχικά επίπεδα στο μοντέλο είναι τα τυπικά επίπεδα ConvNet που χρησιμοποιούνται και για την ταξινόμηση εικόνας. Αποτελούν το βασικό δίκτυο στο οποίο στη συνέχεια προστίθενται μερικά βοηθητικά επίπεδα για να παράγουν τις ανιχνεύσεις έχοντας κατά νου τους χάρτες χαρακτηριστικών πολλαπλών κλιμάκων, τα προεπιλεγμένα πλαίσια και την αναλογία των διαστάσεων.

### 3. Retina-Net

Το Retina-Net αποτελεί ένα ενιαίο, ενοποιημένο δίκτυο που αποτελείται από ένα δίκτυο κορμού και δύο δευτερεύοντα δίκτυα για συγκεκριμένες εργασίες. Είναι ένα μοντέλο ανίχνευσης που έχει αποδειχθεί ότι λειτουργεί καλά με μικρά και πυκνά αντικείμενα. Η ραχοκοκαλιά είναι υπεύθυνη για τον υπολογισμό ενός χάρτη δυνατοτήτων μετατροπής σε μια ολόκληρη εικόνα εισόδου, δημιουργώντας ένα δίκτυο αυτο-αυτοσυγκέντρωσης. Το πρώτο υποδίκτυο πραγματοποιεί την ταξινόμηση στην έξοδο κορμών και το δεύτερο υποδίκτυο εκτελεί την παλινδρόμηση οριοθέτησης πλαισίου.




*Εικόνα 3.9. Η αρχιτεκτονική του Retina-Net*

Χρησιμοποιεί την λειτουργία Feature Pyramid Network (FPN) πάνω από μια feedforward ResNet αρχιτεκτονική προκειμένου να δημιουργήσει μια πλούσια πυραμίδα πολλών επιπέδων, συνελικτικών χαρακτηριστικών που στη συνέχεια τροφοδοτεί τα δύο υποδίκτυα όπου το ένα ταξινομεί τα κουτιά anchor και το άλλο εκτελεί παλινδρόμηση από το κουτί anchor στα κουτιά εδάφους-αλήθειας (ground-truth anchor boxes).

# Κεφάλαιο 4ο

## 4.1. Τι είναι ο αλγόριθμος YOLO

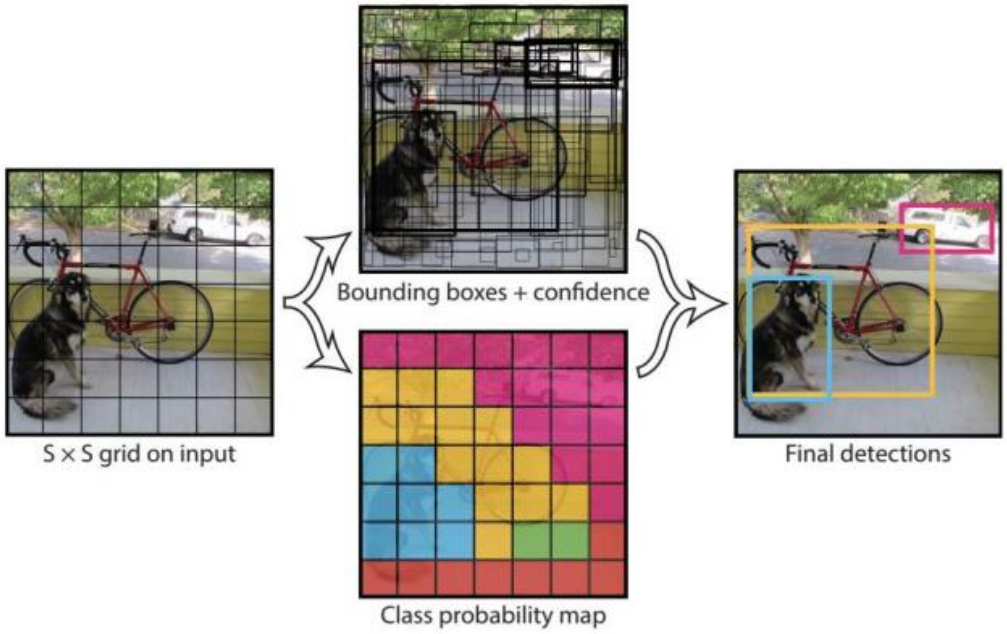
Η ανίχνευση αντικείμενων (Object detiction) αποτελεί ένα από τα πιο κλασικά προβλήματα στο πεδίο της μηχανικής όρασης. Στόχος του είναι η ανίχνευση αντικείμενου μέσα σε μια εικόνα και η ταξινόμηση του σε διάφορες κλάσεις. Ο εντοπισμός θέσης αντικείμενων πάνω σε μια εικόνα (localazation) είναι πιο συνθέτη διεργασία από μια απλή ταξινόμηση αντικείμενων, είναι αδύνατο να επιλυθεί με έναν από τους αλγορίθμους που είδαμε πιο πάνω όπου δεν μπορούν χειριστούν φωτογραφίες που περιέχουν παραπάνω από ένα αντικείμενο.

Πρόκειται για μια εργασία μηχανικής όρασης του υπολογιστή που περιλαμβάνει την πρόβλεψη ενός ή περισσότερων αντικειμένων, μαζί με τις κλάσεις και τα πλαίσια οριοθέτησης τους. Το YOLO είναι ένας ανιχνευτής αντικειμένων τελευταίας τεχνολογίας που μπορεί να εκτελέσει ανίχνευση αντικειμένων σε πραγματικό χρόνο με μεγάλη ακρίβεια. Πρωτοεμφανίστηκε στον κλάδο την μηχανικής όρασης το 2015 σε ένα επιστημονικό άρθρο του Joseph Redmon [17] και αμέσως απέκτησε την προσοχή από πολλούς συνάδερφους του στον κλάδο της τεχνητής νοημοσύνη.

Μέχρι τώρα, είδαμε μερικές πολύ διάσημες και με καλές επιδόσεις αρχιτεκτονικές για την ανίχνευση αντικειμένων. Όλοι αυτοί οι αλγόριθμοι έλυσαν ορισμένα προβλήματα που αναφέρονται στις προκλήσεις αλγόριθμων μηχανικής όρασης, αλλά αποτυγχάνουν στο πιο σημαντικό - Ταχύτητα για ανίχνευση αντικειμένων σε πραγματικό χρόνο. Ο αλγόριθμος YOLO δίνει πολύ καλύτερη απόδοση σε όλες τις παραμέτρους που συζητήσαμε μαζί με ένα υψηλό fps για χρήση σε πραγματικό χρόνο. Ο αλγόριθμος YOLO είναι ένας αλγόριθμος που βασίζεται στην παλινδρόμηση, αντί να επιλέγει και να τμηματοποιεί μια εικόνα και να την ταξινομεί επιμέρους, αναγνωρίζει κλάσεις και πλαίσια οριοθέτησης για ολόκληρη την εικόνα σε μία εκτέλεση του Αλγόριθμου.

Ο YOLO, πρόκειται για ένα αλγόριθμο μονού σταδίου που κάνει ταυτόχρονα και την επιλογή περιοχών και την αξιολόγησή τους. Για να το επιτύχει αυτό, απορρίπτει την προσέγγιση του κυλιόμενου παραθύρου που ακολουθεί μεταξύ άλλων και ο Faster R-CNN. Σε αντίθεση με αυτόν, χωρίζει την εικόνα σε ένα πλέγμα SxS, όπου το S καθορίζεται ανάλογα με τις απαιτήσεις μας και τις ιδιαιτερότητες του συνόλου δεδομένων, και σε κάθε μέρος του

πλέγματος, αν υπάρχει αντικείμενο με βάση τα δεδομένα, προχωρά η ανάλυση για την επιλογή των σωστών περιοχών.




*Εικόνα 5.1. Ο Αλγόριθμος YOLO*

Παρατηρείται ότι κατά το χρόνο εκτέλεσης ο αλγόριθμος διατρέχει την εικόνα στο ΣΝΔ μόνο μία φορά. Ως εκ τούτου, ο YOLO είναι τάξεις μεγέθους ταχύτερος (45 καρέ ανά δευτερόλεπτο) από τους άλλους αλγόριθμους ανίχνευσης αντικειμένων. Μια άλλη βασική διαφορά είναι ότι το YOLO βλέπει την πλήρη εικόνα ταυτόχρονα σε αντίθεση με την εξέταση μόνο προτάσεων περιοχής που δημιουργήθηκαν στις προηγούμενες μεθόδους.

Με αυτόν τον τρόπο περιορίζεται δραστικά ο χρόνος εκπαίδευσης αφού περιοχές χωρίς αντικείμενα που συνήθως κυριαρχούν, δεν συμβάλλουν στην επεξεργασία. Τα αντικείμενα όμως δεν βρίσκονται μόνο σε μια περιοχή του πλέγματος. Ο αλγόριθμος θεωρεί πως μια περιοχή είναι αρμόδια για την ανίχνευση ενός αντικειμένου, αν το κέντρο του αντικειμένου με βάση τα δεδομένα, βρίσκεται στην περιοχή αυτή. Για να γίνει καλύτερα κατανοητός ο τρόπος λειτουργίας του αλγορίθμου, παρατίθεται η μορφή της πληροφορίας που κρατείται για κάθε περιοχή.

| | |
|---|---|
| Y | $p_c$ |
| | $b_x$ |
| | $b_y$ |
| | $b_h$ |
| | $b_w$ |
| | $c_1$ |
| | $c_2$ |

Θεωρώντας ότι έχουμε 2 κλάσεις, η πληροφορία κωδικοποιείται ως εξής. Ο ηλεκτρονικός υπολογιστής (pc) συμβολίζει την πιθανότητα ύπαρξης αντικειμένου, οι επόμενες 4 μεταβλητές τα όρια της προβλεπόμενης περιοχής και οι μεταβλητές c1 και c2 παίρνουν τιμή μεταξύ 0 και 1 και συμβολίζουν την πιθανότητα μιας κλάσης να αντιπροσωπεύεται στο αντικείμενο. Η μεταβλητή pc αποτελεί γινόμενο της πραγματικής δυαδικής τιμής ύπαρξης αντικειμένου και του δείκτη IoU που υπολογίζεται με την προβλεπόμενη περιοχή. Η αρχική έκδοση του YOLO παρά τα ενθαρρυντικά και προπαντός γρήγορα αποτελέσματα, δυσκολευόταν στην ανίχνευση αντικειμένων που εμφανίζονταν πολλάκις σε κάποιο μπλοκ καθώς ο προηγούμενος μηχανισμός δεν προέβλεπε αυτήν την ύπαρξη, ενώ υπήρχε σχετικά μεγάλος αριθμός σφαλμάτων που προερχόταν από τον σωστό καθορισμό των bounding boxes. Αυτή οι περιορισμοί αίρονται στην επόμενη έκδοση του αλγορίθμου [19] όπου περιέχει μια σειρά από βελτιώσεις. Εκτός από την χρήση batch normalization αντί για dropout για την αποφυγή overfitting αλλά και συνελικτικών επιπέδων στη θέση των χρονοβόρων πλήρων συνδεδεμένων, βασική βελτίωση έφερε η χρήση κάποιας μορφής anchor boxes. Αντί για την κλασική μορφή του διανύσματος που παρουσιάσαμε πριν σε κάθε μέρος του πλέγματος, πλέον σε κάθε επιμέρους κομμάτι εξετάζονται όλα τα διαθέσιμα anchor boxes και αποθηκεύονται βαθμοί εμπιστοσύνης για το καθένα. Με αυτόν τον τρόπο βελτιώνεται η απόδοση ενώ γίνεται εφικτή και η ανίχνευση παραπάνω αντικείμενων στο πλέγμα αφού κάθε anchor box θα μπορεί να εξειδικεύεται σε διαφορετικό αντικείμενο.

Τα anchor boxes έχουν την δυνατότητα να προσαρμόζονται και αυτά στα εκάστοτε αντικείμενα μέσα από την διαδικασία εκμάθησης. Με τυχαία αρχικοποίηση όμως χάνεται πολύτιμος χρόνος και είναι πιθανό τα αποτελέσματα να μην είναι τα αναμενόμενα μέσα σε ένα περιορισμένο χρονικό διάστημα εκπαίδευσης. Για αυτό το λόγο προτείνεται να τρέξει πρώτα ένας αλγόριθμος συσταδοποίησης (k-means) στο σύνολο δεδομένων ώστε να

χρησιμοποιηθούν τα anchor boxes με τον μεγαλύτερο δείκτη IoU. Πειράματα έδειξαν ότι το δίκτυο αποδίδει πολύ καλύτερα με αυτήν την προσέγγιση. Επειδή υπάρχει η πιθανότητα ένα αντικείμενο να φέρει υψηλή πιθανότητα σε διαφορετικά anchor boxes, εφαρμόζεται και εδώ μη μέγιστη καταστολή (non-maximum suppression - NMS) για τον περιορισμό τους.

Επειδή όπως έχουμε τονίσει, η ανίχνευση αντικειμένων δεν είναι ένα απλό πρόβλημα ταξινόμησης, η συνάρτηση σφάλματος, όπως και στον Faster R-CNN, θα πρέπει να αξιολογεί τόσο την σωστή επιλογή κλάσης όσο και τα προβλεπόμενα όρια του αντικειμένου. Πιο συγκεκριμένα, η συνάρτηση σφάλματος στον YOLO έχει 3 επιμέρους συστατικά. Το πρώτο μέρος έχει να κάνει με την πιθανότητα ύπαρξης αντικειμένου μιας κλάσης:

$$L_{prob} = \lambda_{class} \sum_{i}^{S^2} \sum_{j}^{B} \sum_{c}^{C} 1_{ij}(p_{ij}(c) - p^*_{ij}(c))^2$$

Με C συμβολίζουμε των σύνολο των κλάσεων, με S2 το πλέγμα, με B των αριθμό των anchor boxes και pij (c) και p ∗ ij (c) είναι η πιθανότητα να ανήκει στην κλάση και η πραγματική τιμή αντίστοιχα. Ο όρος 1ij ενεργοποιείται και γίνεται 1 όταν υπάρχει αντικείμενο στο κελί του πλέγματος και το box που είναι υπεύθυνο έχει IoU > 0.5 και υψηλότερο IoU, διαφορετικά είναι 0. Το δεύτερο μέρος λαμβάνει υπόψη τις διαστάσεις των περιοχών:

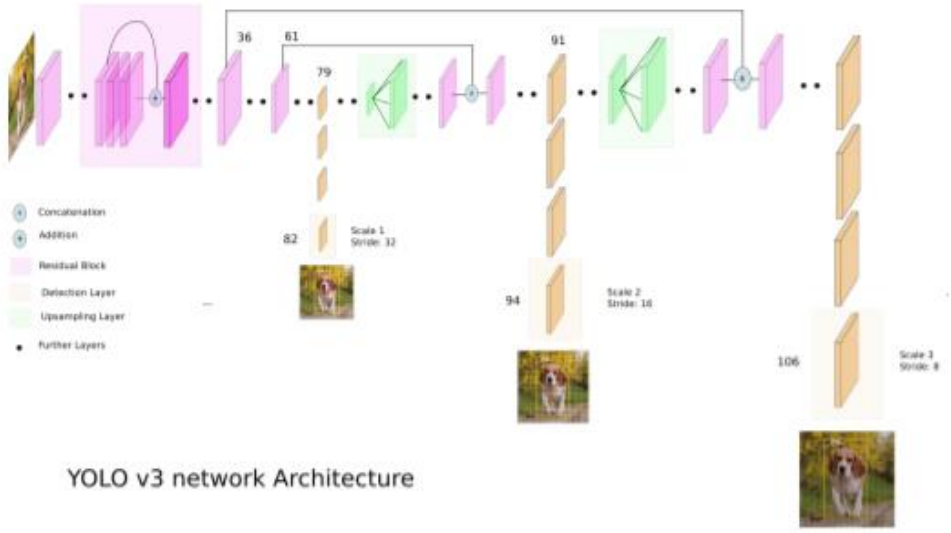
$$L_{box}(r) = \lambda_{coord} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{respObj} \sum_{r \in (x,y,w,h)} (pred_{ij}^r - truth_{ij}^r)^2$$
$$+\lambda_{coord} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{noRespObj} \sum_{r \in (x,y,w,h)} (pred_{ij}^r - anchor_{ij}^r)^2$$

Με r συμβολίζονται οι διαστάσεις της περιοχής, με pred οι προβλέψεις, truth τα πραγματικά δεδομένα και anchor οι διαστάσεις του εκάστοτε anchor box. Παρόμοια με πριν, ενεργοποιούνται οι όροι 1ij , ο πρώτος όταν το anchor box είναι αρμόδιο για τον εντοπισμό του αντικειμένου (IoU > 0.5 και υψηλότερο IoU) και ο δεύτερος όταν δεν είναι αρμόδιο. Σε πλήρη αναλογία υπολογίζεται και ο τελευταίος όρος:

$$L_{conf}(r) = \lambda_{obj}^{conf} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{respObj} (conf_{ij}^{pred} - IoU(pred_{ij}, truth_{ij}^{r})^2$$

$$+\lambda_{noobj}^{conf} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{noRespObj} (conf_{ij}^{pred} - 0)^2$$

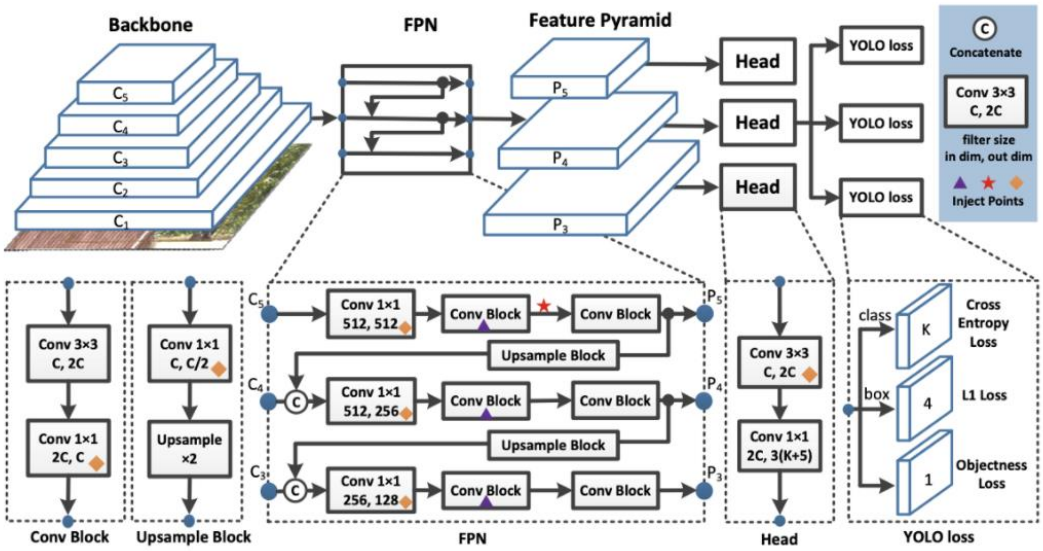
Όλοι οι όροι λ αποτελούν υπερπαραμέτρους και στόχο έχουν την καλύτερη ισορροπία του συστήματος. Σε αντίθεση με άλλους αλγορίθμους, ως βοηθητικό δίκτυο για την ανάλυση χαρακτηριστικών δεν χρησιμοποιήθηκε κάποιο από τα γνωστά. Στα πλαίσια αυτής της διπλωματικής εργασίας θα αξιοποιήσουμε 3 διαφορετικά δίκτυα που δημιουργήθηκαν για τον YOLO αποκλειστικά. Μια μικρή έκδοση του YOLOv2 με συνελικτικά 9 επίπεδα (tiny-YOLOv2), το full-YOLOv2 με 23 συνελικτικά επίπεδα και η τελευταία έκδοση (full-YOLOv3) με 76 συνελικτικά επίπεδα και συνολικά 106. Παρά τον μεγάλο αριθμό των επιπέδων, ο αλγόριθμος YOLO πετυχαίνει εφάμιλλα αποτελέσματα και καλύτερα σε σχέση με τον Faster R-CNN σε πολύ μικρότερο χρόνο όμως, καθιστώντας τον αλγόριθμο ικανό για ανίχνευση real-time.




*Σχήμα 5.2 Η πλήρης αρχιτεκτονική του YOLOv3*

Η εικόνα μαρτυράει ότι η διαδικασία ανίχνευσης ολοκληρώνεται σε 3 φάσεις. Κάθε μία από τις φάσεις χρησιμοποιεί το 1/3 των anchor boxes και διεξάγει ανίχνευση για διαφορετικά μεγέθη αντικειμένων. Πρώτα γίνεται η ανίχνευση αντικειμένων μεγάλων διαστάσεων έπειτα από 82 επίπεδα και έχοντας τους μικρότερους χάρτες χαρακτηριστικών. Στην συνέχεια με χρήση αντίστροφης δειγματοληψίας, οι χάρτες χαρακτηριστικών μεγαλώνουν και στις

επόμενες 2 φάσεις γίνεται η ανίχνευση για αντικείμενα μεσαίων και μικρών διαστάσεων αντίστοιχα. Παρατηρούμε ότι στα πρώτα βήματα, έχουμε προσθέσεις χαρτών για καλύτερη εκμάθηση, χαρακτηριστικό το οποίο στα πλαίσια αυτής της διπλωματικής εργασίας θα επεκτείνουμε, ενώ μετά από κάθε αντίστροφη δειγματοληψία έχουμε συγκόλληση χαρτών από προηγούμενα επίπεδα, προσθέτοντας μεγαλύτερη συνοχή στο δίκτυο.




## 4.2. Καινοτομίες και Χρήση Νέου Μοντέλου

Στην παράγραφο που ακολουθεί θα αναλυθούν οι καινοτομίες που εφαρμόστηκαν στον υπάρχων αλγόριθμο YOLO και που οδήγησαν υπό ορισμένες συνθήκες σε βελτίωση των αρχικών αποτελεσμάτων.

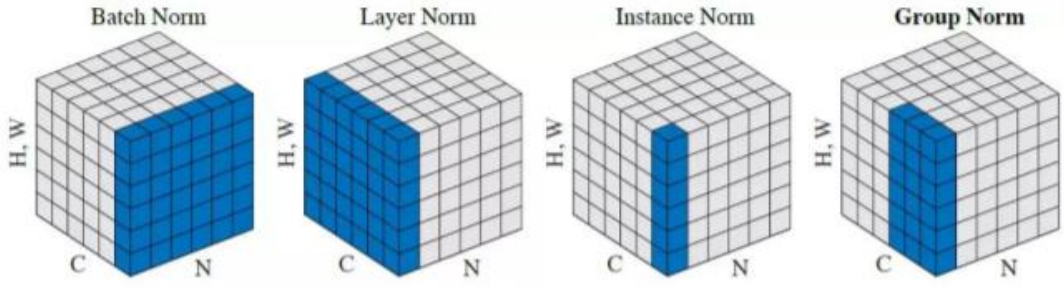
**Κανονικοποίηση παρτίδων (Batch -Normalization)**

Η κανονικοποίηση παρτίδων (Batch Normalization - BN), όπως αναφέραμε και σε προηγούμενο κεφάλαιο, αποτελεί βασικό κομμάτι πλέον στα νευρωνικά δίκτυα και ειδικότερα στην βαθιά μάθηση. Το εκπαιδευόμενο δίκτυο όμως αποδίδει καλύτερα όταν ο αριθμός των εικόνων στις παρτίδες (batch size) είναι σχετικά μεγάλος, συνήθως 16 ή 32. Η διατήρηση τόσο υψηλού αριθμού εικόνων στις παρτίδες αποτελεί τροχοπέδη για προβλήματα που καταναλώνουν πολλή μνήμη. Ειδικά στην ανίχνευση αντικειμένων, που θα ασχοληθούμε,

όπως και στην κατάτμηση εικόνας, το μεγάλο υπολογιστικό κόστος περιορίζει το μέγεθος παρτίδας σε μικρούς αριθμούς, μειώνοντας την απόδοση του δικτύου.

Υπό το φως αυτών των ζητημάτων, μόλις το 2018 αναδείχθηκε μια νέα τεχνική κανονικοποίησης που έρχεται να δώσει λύσεις στα προβλήματα που αναφέραμε. Η Κανονικοποίηση Ομάδας (Group Normalization - GN) [20] χρησιμοποιεί διαφορετική προσέγγιση από την BN, μην εκτελώντας κανονικοποίηση στην διάσταση των παρτίδων. Αν και προηγουμένως, είχαν περιγραφεί τεχνικές που ακολουθούν την ίδια τακτική με την GN, όπως η Layer Normalization και η Instance Normalization, καμία από αυτές δεν επιτύγχανε καλύτερα αποτελέσματα στην ανάλυση εικόνας.

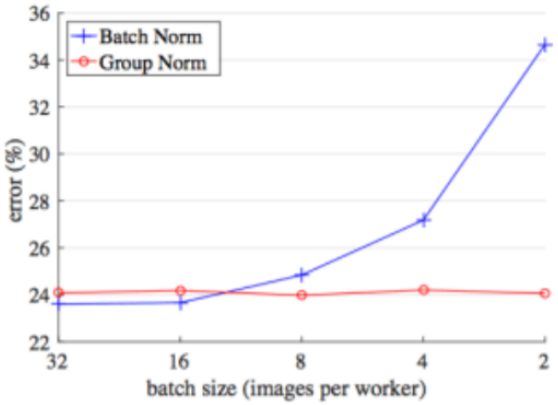
Στην ανάλυση εικόνων 2 διαστάσεων, έπειτα από κάθε συνελικτικό επίπεδο του δικτύου ο όγκος πληροφορίας αποτελείται από 4 διαστάσεις NxCxHxW. Οι όροι H και W αντιπροσωπεύουν τους άξονες των διαστάσεων της εικόνας, ο όρος N τον άξονα των παρτίδων ενώ ο C των καναλιών του επιπέδου. Η BN εκτελεί κανονικοποίηση στον άξονα του N, ενώ η Layer Normalization στον άξονα του C. Αντίθετα, η καινοτομία της GN ενέχει στο γεγονός ότι χωρίζει την διάσταση του C σε ομάδες, συνήθως 32, και εκτελεί κανονικοποίηση σε κάθε ομάδα χωριστά. Με αυτήν την τακτική, η απόδοση του συστήματος αποδεσμεύεται σε μεγάλο βαθμό από τον αριθμό των παρτίδων, διευρύνοντας τους ορίζοντες ακόμα και για υπολογιστικά συστήματα που υστερούν σε hardware.




*Εικόνα 5.3. Διαφορετικές προσεγγίσεις των κυρίαρχων τεχνικών κανονικοποίησης*

Πειραματικά αποτελέσματα στο πεδίο της ταξινόμησης εικόνων έδειξαν ότι η GN υστερεί ελαφρώς σε περιπτώσεις όπου ο αριθμός των παρτίδων είναι 16 ή 32 ενώ αντίθετα υπερτερεί όταν ο αριθμός μικραίνει. Το δίκτυο στο οποίο δοκιμάστηκε είναι το ResNet-50 και έχει

αρκετές ομοιότητες με το δίκτυο στο οποίο πειραματιστήκαμε στα πλαίσια του αλγορίθμου YOLO για ανίχνευση αντικειμένων.



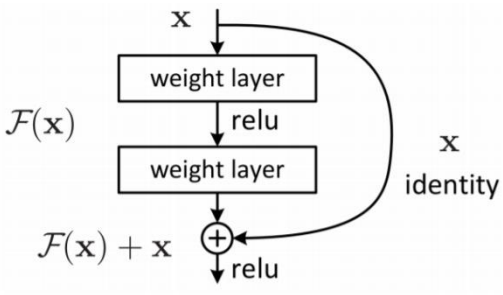

*Εικόνα 5.4. Σύγκριση αποτελεσμάτων διαφορετικών κανονικοποιήσεων*

Στην εικόνα 5.4 βλέπουμε την σύγκριση αποτελεσμάτων για διαφορετικές κανονικοποιήσεις στον τομέα της ταξινόμησης εικόνων. Παρατηρούμε ότι το ποσοστό λάθους με χρήση group normalization μένει σχεδόν ανεπηρέαστο από των αριθμό παρτίδων ενώ με χρήση batch normalization η αύξηση των λαθών είναι αλματώδης.

**Residual Networks of Residual Networks**

Η μεγάλη ανάπτυξη της ισχύς των υπολογιστικών συστημάτων έκανε εφικτή, όπως προείπαμε, την χρήση βαθιών δικτύων που παλιότερα ήταν εξαιρετικά χρονοβόρα. Η χρήση όμως πολύ βαθιών δικτύων έφερε τους ερευνητές αντιμέτωπους με μια νέα πρόκληση. Τα δίκτυα παρατηρήθηκαν ότι είχαν την τάση να «ξεχνάνε» τα χαρακτηριστικά στα οποία θα έπρεπε θεωρητικά να είχαν εκπαιδευτεί στα αρχικά στάδια και ως εκ τούτου η αύξηση των επιπέδων δεν έφερε την προσδοκώμενη αύξηση στην απόδοση. Η απάντηση της επιστημονικής κοινότητας ήταν τα Residual Networks (ResNets) [He16] στα οποία εισήχθη η έννοια της ανατροφοδότησης με την είσοδο.

Πιο συγκεκριμένα, στα ResNets ανά 2 ή ανά 3 συνελικτικά επίπεδα, το αποτέλεσμα ανατροφοδοτείται με αυτό που έλαβε σαν είσοδο το πρώτο από την ομάδα επιπέδων. Με αυτήν την προσέγγιση, τα συστήματα λάβανε μεγάλη ώθηση και αξιοποίησαν στο έπακρο τις δυνατότητες της βαθιάς μάθησης. Ένα τέτοιο δίκτυο είναι και το Darknet-53 που αποτελεί το backbone δίκτυο για το YOLOv3.




*Σχήμα 5.5. Εικονική απεικόνιση ενός residual block.*

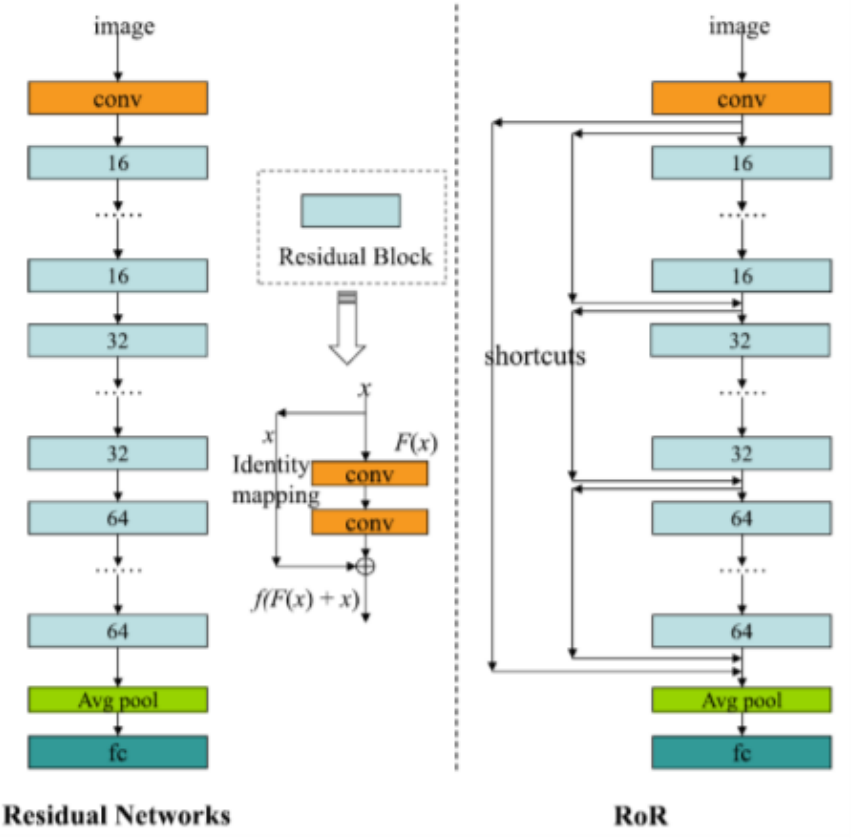
Στο Darknet-53 που αποτελεί τα 74 πρώτα επίπεδα του συνολικού δικτύου των 106 επιπέδων του YOLOv3, κάθε 2 ή 3 συνελικτικά επίπεδα έχουμε ανατροφοδότηση. Με αυτήν την αρχιτεκτονική το δίκτυο περιέχει 21 residual blocks.

| | 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 5.6. Αναλυτική περιγραφή του Darknet-53*

Σε συνέχεια αυτή της λογικής προτάθηκε η χρήση των λεγόμενων Residual Networks of Residual Networks (RoR) [Zhan18] που επεκτείνουν την ιδέα των απλών ResNets.

Αναλυτικότερα, εκτός από τον χωρισμό του δικτύου σε ομάδες 2 ή 3 επιπέδων που κάνουν τα ResNets, δημιουργούνται και ευρύτερες ομάδες επιπέδων, οι οποίες έχουν τις ίδιες διαστάσεις όγκου πληροφορίας στις οποίες επεκτείνεται η ανατροφοδότηση. Με αυτήν την καινοτομία, οι δυνατότητες των δικτύων επεκτείνονται καθώς ο εκάστοτε όγκος πληροφορίας συνυπολογίζεται από περισσότερα δεδομένα, περιορίζοντας ταυτόχρονα και τις ενδεχόμενες αστοχίες του.




*Εικόνα 5.7. Σύγκριση των αρχιτεκτονικών των απλών ResNets με τα RoR*

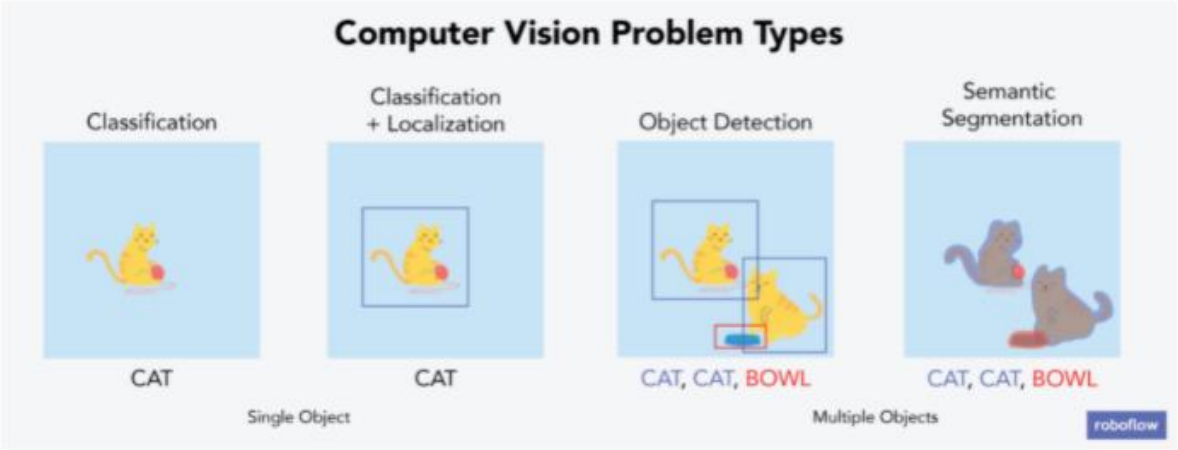
# Κεφάλαιο 5ο

## 5.1. Προλογισμός - Στόχος αλγορίθμου

Το σκάκι είναι ένα διασκεδαστικό παιχνίδι εξυπνάδας και στρατηγικής. Η βελτίωση της ικανότητάς σας στο παιχνίδι απαιτεί μελέτη και κατανόηση το που είχες κάνει αξιοσημείωτα λάθη κατά την εξέλιξη του παιχνιδιού και ποιες διαφορετικές κινίσεις θα μπορούσε να είχε κάνει ένας ανώτερος παίχτης στις ίδιες καταστάσεις.

Έτσι, το να έχεις ένα σύστημα που να αναγνωρίζει την κατάσταση του παιχνιδιού και να καταγράφει κάθε κίνηση θα ήταν πολύτιμο. Αυτό απαιτεί όχι μόνο τον προσδιορισμό τις κλάσης του συγκεκριμένου πιονιού, αλλά και την θέση του πάνω στην σκακιέρα.

Οποιοδήποτε πρόβλημα μηχανικής εκμάθησης ξεκινά με μια καλά διαμορφωμένη διατύπωσή του προβλήματος, την περισυλλογή δεδομένων μας την εκπαίδευση του μοντέλου και την επικύρωση της ορθότητας των αποτελεσμάτων του. Στόχος μας λοιπόν, είναι η οπτική αναγνώριση σκακιστικών κομματιών από ένα σύνολο δεδομένων με σκοπό την αναγνώριση και την ταξινόμηση σε μια από τις 12 διαφορετικές κλάσης που θα περιέχει το σύνολο των δεδομένων μας (όσα δηλαδή και το σύνολο σκακιστικών ποινών και των χρωμάτων τους), αλλά και ο εντοπισμός της ακριβής θέση τον σκακιστικών κομματιών πάνω στην σκακιέρα.




*Εικόνα 6.1. Οι τύποι οπτικών προβλημάτων του υπολογιστή*

## 5.2. Αρχιτεκτονική επίλυσης αλγορίθμου.

1. **Προσέγγιση πρώτη.**

Πρακτικά ο αλγόριθμος μπορεί να χωριστεί σε 2 μέρη, στην αναγνώριση σκακιστικών ποινών και την αναγνώριση των θέσεων τους πάνω στην σκακιέρα. Πρώτες σκέψεις επίλυσης του προβλήματος ήταν το να διαιρέσω μια εικόνα σκακιέρας με τα πιόνια της σε 64 μικρά τετραγωνικά τμήματα. Το κάθε ένα από αυτά τα τμήματα θα ισοδυναμεί με μια από τις πιθανές θέσεις του σκακιστικού ταμπλό και θα κάνουμε την υπόθεση ότι έχει το πολύ 1 η κανένα πιόνι πάνω του, όπως και ακριβώς επιτρέπεται στο σκάκι. Έχοντας τα τμήματα θα περάσουμε στην εκπαίδευση του μοντέλου μας. Στο σκάκι υπάρχουν 6 διαφορετικά πιόνια σε 2 διαφορετικά χρώματα, αρά στο σύνολο έχουμε 12 κλάσης. Έτσι, θα εκπαιδεύσουμε το νευρονικό μας μοντέλο με τον αλγόριθμο αναγνώρισης CNN. Ο αλγόριθμος θα περάσει τα τμήματα που φτιάξαμε από το μοντέλο μας με σκοπό την αναγνώριση και την ταξινόμηση των αποτελεσμάτων. Αυτή η προσέγγιση μας δίνει το πλεονέκτημα να γνωρίζουμε την θέσεις του πιονιού, αλλά απαιτεί μεγάλη υπολογιστική δύναμη μιας και πρέπει να τρέξει το μοντέλο 64 φορές. Γεγονός που υπολογιστικά μας δυσκολεύει να κάνουμε την διαδικασία την αναγνώρισης σε άμεσο χρόνο.

2. **Προσέγγιση δεύτερη.**

Το πρόβλημα τις υπολογιστικής πολυπλοκότητας το λύνει ο αλγόριθμος YOLO . Ο αλγόριθμος αυτός όπως είδαμε και νωρίτερα ευημερεί στην αναγνώριση και ταξινόμηση πολλαπλών αντικείμενων από μια εικόνα, και αυτό με ένα πέρασμα του. Ο YOLO θα επιστρέψει ένα σύνολο από κουτιά οριοθέτησης του κάθε πιονιού που αναγνώρισε στην εικόνα, σε μια λίστα από συντεταγμένες τον χ,y αξόνων τους. Έπειτα, θα στήσουμε την πηγή που μας δίνει τις εικόνες σε σταθερό σημείο, τέτοιο, ώστε να βλέπει ολόκληρη την σκακιέρα. Προϋποθέτοντας όμως ότι οι γωνία και η θέση του τρίποδα και τις σκακιέρας δεν μεταβάλλονται. Γνωρίζοντας τις συντεταγμένες των πιονών και έχοντας σταθερή σκακιέρα μπορούμε να γνωρίζουμε γεωμετρικά που βρίσκονται και τα 64 τετράγωνα της σκακιέρας διαιρώντας την εικόνα σε 64 ίσα τετράγωνα. Με αυτό τον τρόπο κάνοντας μόνο μια σάρωσή την εικόνα ο αλγόριθμος θα

μπορέσει να αναγνωρίσει τα πιόνια αλλά και να γνωρίζει την θέση τους. Αυτή η προσέγγιση λύνει το υπολογιστικό πρόβλημα που είχε η πρώτη προσέγγιση.

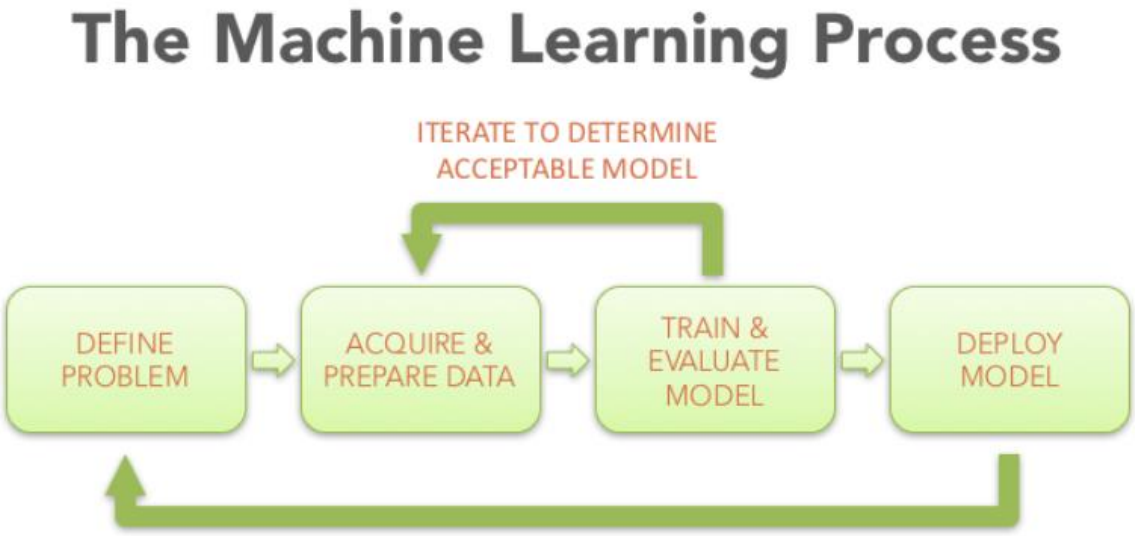
**Έκδοση αλγορίθμου**

Αφού βρήκαμε τον τρόπο που θα σχεδιάσουμε τον αλγόριθμο θα πρέπει να βρούμε και την έκδοση του αλγορίθμου YOLO που θα κάλυπτε καλυτέρα τις ανάγκες μας. Αναμεσά στις 2 τελευταίες έκδοσης του αλγορίθμου παρατηρήσαμε ότι ο YOLO 4 έχει μεγαλύτερο ακρίβεια από τον YOLO 5, που από την άλλη είναι πιο γρήγορος σε καταστάσεις πραγματικού χρόνου. Η ιδιομορφία του σακακιού βασίζετε σε αργές σκεπτόμενες κινήσεις που προϋποθέτουν κάποιο χρονικό διάστημα να πραγματοποιηθούν. Ακόμη και στα ταχύρρυθμα παιχνίδια η παίχτες δεν χρειάζονται εξωπραγματικές ταχύτητες (φύσιν αδύνατον) για να χρειαζόμαστε έναν αλγόριθμο που να αγγίζει ταχύτητες πραγματικού χρόνου. Αρά μπορούμε ευκολά να θυσιάσουμε λίγη ταχύτητα για περισσότερη ακρίβεια. Αυτός είναι ο λόγος που στην συγκεκριμένη περίπτωση η τέταρτη έκδοση του αλγορίθμου YOLO είναι πιο ιδανική.

**Περιβάλλον**

Σε συνδυασμό με τον παραπάνω αλγόριθμο, θα χρησιμοποιηθεί η γλώσσα προγραμματισμού Python. Η Python είναι μία «ώριμη» γλώσσα, περιέχει έναν μεγάλο αριθμό βιβλιοθηκών και χρησιμοποιείται κατά κόρον στη μηχανική μάθηση. Το PyTorch είναι μια πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση, ανεπτυγμένο από την εταιρεία Facebook. Έχει ένα ολοκληρωμένο, ευέλικτο οικοσύστημα εργαλείων, βιβλιοθηκών και με την συνεισφορά προγραμματιστών εκτός εταιρείας, επιτρέπει στους ερευνητές να αναπτύξουν περαιτέρω τη μηχανική μάθηση και στους προγραμματιστές να δημιουργήσουν εύκολα εφαρμογές.

Ως περιβάλλον ανάπτυξης του αλγόριθμου χρησιμοποίησα το Google colab. Πρόκειται για ένα δωρεάν cloud εργαλείο της Google, διαθέσιμο κυρίως για την υποστήριξη ερευνητών σε project που απαιτούν μεγάλη υπολογιστική στήριξη σε GPU. [25].




*Εικόνα 6.2. Η διαδικασία machine learning*

**Συλλογή δεδομένων**

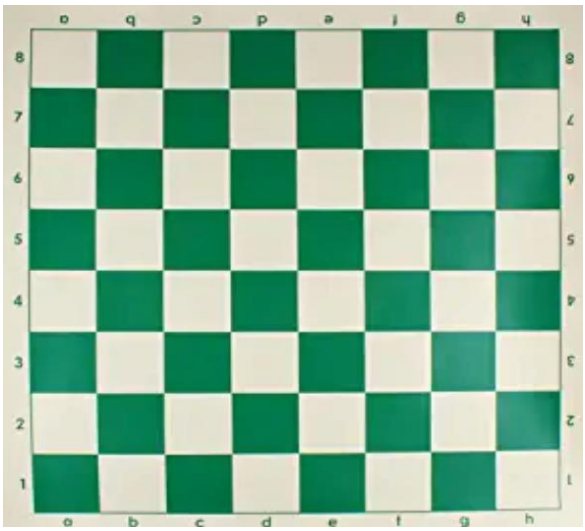
Αφού επιλέξαμε τον αλγόριθμο που θα χρησιμοποιήσουμε σειρά έχει η περισυλλογή των απαραίτητων δεδομένων. Προκειμένου να αναγνωρίσουμε σκακιστικά κομμάτια πρέπει να συλλέξουμε και να σχολιάσουμε (annotate) εικόνες από πιόνια πάνω σε μια σκακιέρα.

Προκλήσεις κατά την συλλογή δεδομένων.

**Γωνιά λήψης**

Έκανα μερικές υποθέσεις στη συλλογή δεδομένων μου. Πρώτον, όλες οι εικόνες μου τραβήχτηκαν από την ίδια γωνία. Έστησα ένα τρίποδο στο τραπέζι κοντά στη σκακιέρα μου όπου με το κινητό μου έβγαλα τις εικόνες. Στην συνέχεια θα πρέπει να μελετήσουμε και την γωνιά οπού θα βγουν οι φωτογραφίες. Οι πρώτες λήψεις που έκανα ήταν ακριβώς επάνω από την σκακιέρα. Δηλαδή οι γωνίες τις σκακιέρας και οι πλευρές εφάπτονταν πάνω από την σκακιέρα. Αυτό είχε το θετικό ότι μιας και είχαμε στατική θέση της σκακιέρας, μπορούσαμε και οποιαδήποτε στιγμή να γνωρίζουμε και την θέση που βρίσκετε το κάθε σκακιστικό πιόνι. Από την άλλη μεριά όμως, με τη λήψη εικόνας από την πάνοψη τις σκακιέρας ο αλγόριθμος δεν μπορούσε να διακρίνει τα χαρακτηριστικά που ήταν απαραίτητα για την αναγνώριση των

σκακιστικών πιονιών, και δεν μπορούσε να τα αναγνωρίσει και να τα ταξινομήσει με μεγάλη ακρίβεια.



*Εικόνα που βλέπει της σκακιέρα από πάνω.*

Έτσι για να πέτυχουμε μεγαλύτερη ακρίβεια στην ανάγνωση έστησα το τρίποδο σε νέα γωνιά που έβλεπε τα πιόνια από το πλάι τους. Αυτή η προσέγγιση είχε ως αποτέλεσμα ο αλγόριθμος να συλλέγει και να ταξινομεί τα πιόνια με μεγαλύτερη ακρίβεια. Τελικά, συγκέντρωσα 292 εικόνες, οπού στη συνέχεια υπήρξε τοποθέτηση σχολιασμών (labeling) πάνω σε αυτές.


*Εικόνα που βλέπει την σκακιέρα υπό γωνιά*

**Σχολιασμός ετικετών με πλαίσια οριοθετήσης.**

Η επισήμανση σχολιασμών στα δεδομένα μας είναι ένα ουσιαστικό βήμα σε έναν αλγόριθμο μηχανικής μάθησης. Σκουπίδια μέσα σκουπίδια έξω (Garbage In Garbage Out) είναι μια φράση που χρησιμοποιείται συνήθως στην κοινότητα μηχανικής μάθησης, που σημαίνει ότι η ποιότητα των δεδομένων εκπαίδευσης καθορίζει την ποιότητα του μοντέλου. Το ίδιο ισχύει για τους σχολιασμούς που χρησιμοποιούνται για την επισήμανση δεδομένων. Η ορθότητα του μοντέλου εξαρτάται από τις ετικέτες στις οποίες τροφοδοτούμε κατά τη φάση της εκπαίδευσης του. Η επισήμανση σχολιασμών δεδομένων είναι μια εργασία που απαιτεί πολλή χειροκίνητη εργασία. Υπάρχουν διάφορα εργαλεία δωρεάν και ανοιχτού κώδικα για επισήμανση δεδομένων. Το εργαλείο που θα χρησιμοποιήσουμε εμείς για τον σχολιασμό των εικόνων και αργότερα για την μεταμόρφωση των δεδομένων μας είναι το Roboflow. Μιας και τα σωστά δεδομένα παίζουν καθοριστικό ρολό στην απόδοση του αλγορίθμου, είναι πιθανό μετά τη διαδικασία της εκπαίδευσης για την ακεραιότητα του μοντέλου, ενδεχομένως να χρειαστεί να γίνουν αλλαγές σε αυτά. Αυτό που προσφέρει το Roboflow είναι ότι ανά πάσα στιγμή μπορείς τροποποιήσεις τα δεδομένα σου, και να τα ξανά εκδώσεις σε νέα έκδοση τους. Αλλά πλεονέκτημα που προσφέρει είναι:

1) Μας δίνει την δυνατότητα να σχολιάσουμε τις εικόνες μας τραβώντας πλαίσια οριοθέτησης σε ένα εύκολο σε χρήση User Interface

2) Να μπορούμε να διαχωρίσουμε τα σύνολα των δεδομένων μας σε εκπαίδευση/επικύρωση/δοκιμή, που αργότερα θα τα χρησιμοποιήσουμε στην διαδικασία της εκπαίδευσης του νευρονικού μας.

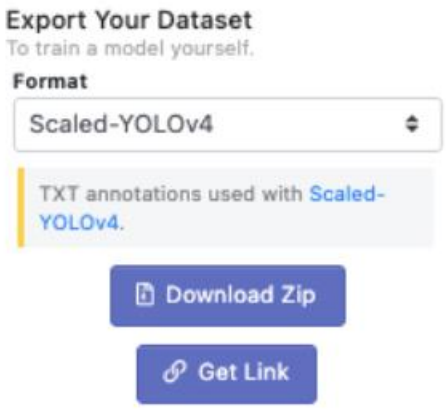
3) Προ επεξεργασία των δεδομένων όπως αλλαγή στην κλίμακα των εικόνων μας.

4) Μεταμόρφωση των δεδομένων μας, για αποφυγή για μεγαλύτερο όγκο δεδομένων και αποφυγή overfitting.

5) Εξαγωγή δεδομένων σε links σε διάφορες μορφές, με σκοπό την εύκολη ένταξη τους στον αλγόριθμο.[22]

*Διαδικασία σχολιασμού εικόνων*

Έχοντας έτοιμο το σύνολο δεδομένων το εξάγω σε μορφή COCO που είναι και ο προκαθορισμένος τρόπος αποθήκευσης εικόνων και σχολιασμών για τον αλγόριθμο YOLO.




Διαδικασία εξαγωγής δεδομένων από το Roboflow

**Αλγόριθμος**

Ο αλγόριθμος θα περιλαμβάνετε από τα εξής μέρη:

1. Στήνοντας το περιβάλλον με την εγκατάσταση τον απαραιτήτων βιβλιοθηκών.
2. Φόρτωση των δεδομένων από το URI που κάναμε εξαγωγή από το Roboflow.
3. Παραμετροποίηση παραμέτρων (Hyperparameter tunning).
4. Εκπαίδευση.

Έχοντας εγκαταστήσει τις απαραίτητες βιβλιοθήκες στο περιβάλλον του Google Colab βήμα πρώτο στον αλγόριθμο μας είναι η φόρτωση των δεδομένων μας από το URI που πήραμε από το Roboflow.

```
# REPLACE this link with your Roboflow dataset (export as YOLOv4 PyTorch format)
!curl -L "https://public.roboflow.com/ds/tx4ITnfMaD?key=cS1J4Kvnd6" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   896  100   896    0     0   1485      0 --:--:-- --:--:-- --:--:--  1485
100 7041k  100 7041k    0     0  7342k      0 --:--:-- --:--:-- --:--:-- 7342k
Archive:  roboflow.zip
 extracting: test/e4583d082076b2b549b3736ad1b193c9_jpg.rf.55af0c9be903e4dda4a002af87bdeaac.jpg
 extracting: test/5a35ba2ec3e0d0b2b12b1758a8ac29aa_jpg.rf.280f9940defacbb5d840aef65a9257e5.jpg
 extracting: test/33afc6085c4d5a8f2421c1adc5a1edbf_jpg.rf.642c5b5370e5900cae860045ade36211.jpg
 extracting: test/b9402881fa580d0eb8b9b98845417550_jpg.rf.238f51ed1096107324224ade76408bdb.jpg
 extracting: test/e0d38d159ad3a801d0304d7e275812cc_jpg.rf.0cd06a940ccc9894109d83792535e3eb.jpg
 extracting: test/2f6fb003bb89cd401322a535acb42f65_jpg.rf.49b342a7b1f6de3f0e328beaf094a945.jpg
 extracting: test/7a34d8620235048917b28bcfd3b5572b_jpg.rf.182f59c512dcb99ed53da97dfe2d3d85.jpg
 extracting: test/0b47311f426ff926578c9d738d683e76_jpg.rf.58093f799a6f56c30830617ca44745ca.jpg
 extracting: test/cfc306bf86176b92ffc1afbb98d7896f_jpg.rf.2de0d2da0025b5993598f47fb1d51d10.jpg
 extracting: test/a3863d0be6002c21b20ac88817b2c56f_jpg.rf.ba355f57305a2236db1a9cfb0ac79989.jpg
 extracting: test/ba6289667457a113c1a753a87c041c51_jpg.rf.31874cd941dfda8ac86b77fdc97f4931.jpg
 extracting: test/b4ff4132c8c85da97d8bf9a2a4ed3e3d_jpg.rf.445b4d70e01a3d742ecf0b7756e636da.jpg
 extracting: test/c5a012dfa72816098d23fc8baee67834_jpg.rf.8b0e56e40c7c2429ce0e56797eb55c88.jpg
```

Η παραμετροποίηση των παραμέτρων του μοντέλου μας κατά την διαδικασία της εκπαίδευσης είναι πολύ σημαντική,  μιας και ο κατάλληλος συνδυασμός τους είναι αυτός που θα μας φέρει τα πιο ακριβή αποτελέσματα.

Δοκιμαστήκαν διαφορά τεχνάσματα πριν οδηγηθούμε στις παραμέτρους που διαλέξαμε τώρα. Όπως η αυτόματη κλιμάκωση του ρυθμού μάθησης και διαφορετικά μεγέθη όγκου δεδομένου ανά epoch. Στο τέλος καταλήξαμε με ένα χαμηλό ρυθμό μάθησης κάτι που θα μας κάνει το μοντέλο τόσο ακριβές όσο και γρήγορο στην εκπαίδευση. Το darkent μας έχει αυτοματοποιήσει την διαδικασία της εκπαίδευσης και έτσι, με το python train και τις παραμέτρους μας ξεκινάμε την εκπαίδευση με τον αλγόριθμο YOLO, αποθηκεύοντας σε κάθε epoch τα βάρη του μοντέλου.

```
Epochs:          50
Batch size:      2
Subdivisions:    1
Learning rate:   0.001
Training size:   202
Validation size: 58
Checkpoints:     True
Device:          cuda
Images size:     608
Optimizer:       adam
Dataset classes: 13
Train label path:train.txt
```

```
start training
-b batch size (you should keep this low (2-4) for training to work properly)
-s number of subdivisions in the batch, this was more relevant for the darknet framework
-l learning rate
-g direct training to the GPU device
pretrained invoke the pretrained weights that we downloaded above
classes - number of classes
dir - where the training data is
epoch - how long to train for
!python train.py -b 2 -s 1 -l 0.001 -g 0 -pretrained ./yolov4.conv.137.pth -classes {num_classes} -dir ./train -epochs 50
```

```
Epoch 158/500
6/6 [==============================] - 16s 3s/step - loss: 73.7902 - val_loss: 68.7980
Epoch 159/500
6/6 [==============================] - 15s 3s/step - loss: 72.8749 - val_loss: 71.9348
Epoch 160/500
6/6 [==============================] - 16s 3s/step - loss: 74.6815 - val_loss: 63.4373
Epoch 161/500
6/6 [==============================] - 15s 3s/step - loss: 77.8056 - val_loss: 95.0817
Epoch 162/500
6/6 [==============================] - 15s 3s/step - loss: 77.7340 - val_loss: 68.5197
Epoch 163/500
6/6 [==============================] - 15s 2s/step - loss: 72.5670 - val_loss: 67.4765
Epoch 164/500
6/6 [==============================] - 15s 3s/step - loss: 71.9312 - val_loss: 75.5616
Epoch 165/500
6/6 [==============================] - 15s 2s/step - loss: 77.7279 - val_loss: 73.4924
Epoch 166/500
6/6 [==============================] - 15s 3s/step - loss: 81.1011 - val_loss: 78.7103
Epoch 167/500
6/6 [==============================] - 15s 2s/step - loss: 74.0910 - val_loss: 75.8251
Epoch 168/500
6/6 [==============================] - 15s 2s/step - loss: 78.8675 - val_loss: 62.8602
Epoch 169/500
6/6 [==============================] - 15s 2s/step - loss: 77.3417 - val_loss: 78.1505
Epoch 170/500
6/6 [==============================] - 15s 2s/step - loss: 69.7987 - val_loss: 65.9444
Epoch 171/500
6/6 [==============================] - 15s 2s/step - loss: 75.3233 - val_loss: 63.4790
```

Όσο πέφτει το validation loss τόσο το μοντέλο μας βελτιώνεται.

Η διαδικασία της εκπαίδευσης ήταν χρονοβόρα και χρειάστηκαν αρκετές ώρες για την ολοκλήρωση της. Έχοντας αποθηκεύσει τα βάρη μετά από κάθε epoch , θα χρησιμοποιήσουμε στο μοντέλο μας αυτό με τον μικρότερη τιμή λάθους, οπου και θα το αποθηκεύσουμε στο google cloud για μελλοντική χρήση.

```
[ ]  from google.colab import drive
     drive.mount('/content/gdrive')

     Mounted at /content/gdrive
```

Έχοντας φορτώσει το μοντέλο με το χαμηλότερο ποσοστό λάθους θα το δοκιμάσουμε και σε πραγματικά δεδομένα για να δούμε την ορθότητα των αποτελεσμάτων του, περνώντας μια τυχαία εικόνα από τα δεδομένα μας.

```
test_images = [f for f in os.listdir('test') if f.endswith('.jpg')]
import random
img_path = "test/" + random.choice(test_images);

n_classes = num_classes
weightfile = '/content/gdrive/MyDrive/ptixiakh/Yolov4_epoch46.pth'
imgfile = img_path
namesfile = 'test/_classes.txt'

model = Yolov4(n_classes=n_classes)

pretrained_dict = torch.load(weightfile, map_location=torch.device('cuda'))
model.load_state_dict(pretrained_dict)

use_cuda = 1
if use_cuda:
  model.cuda()

img = Image.open(imgfile).convert('RGB')
sized = img.resize((608, 608))
```

Ως αποτέλεσμα, πήραμε μια λίστα από τα πιόνια και την ορθότητα τον αποτελεσμάτων αυτών σύμφωνα με το μοντέλο όπως, και μια γραφική αναπαράσταση αυτών.

**white-queen:** 1.000000 **white-bishop:** 0.999688

**black-bishop:** 1.000000 **black-bishop:** 1.000000

**white-bishop:** 1.000000 **black-queen:** 1.000000

**white-rook:** 1.000000 **black-pawn:** 1.000000

**black-pawn:** 1.000000 **black-rook:** 0.999976

**black-pawn:** 1.000000 **black-pawn:** 1.000000

**white-king:** 1.000000 **black-pawn:** 1.000000

**white-pawn:** 1.000000 **black-pawn:** 1.000000

**white-pawn:** 1.000000 **white-pawn:** 1.000000

**white-pawn:** 1.000000w **hite-pawn:** 1.000000

**white-pawn:** 1.000000 **black-rook:** 0.999972

**black-king:** 0.999989 **black-knight:** 0.984503

**white-knight:** 0.999947 **black-knight:** 0.997152

**white-knight:** 0.999450

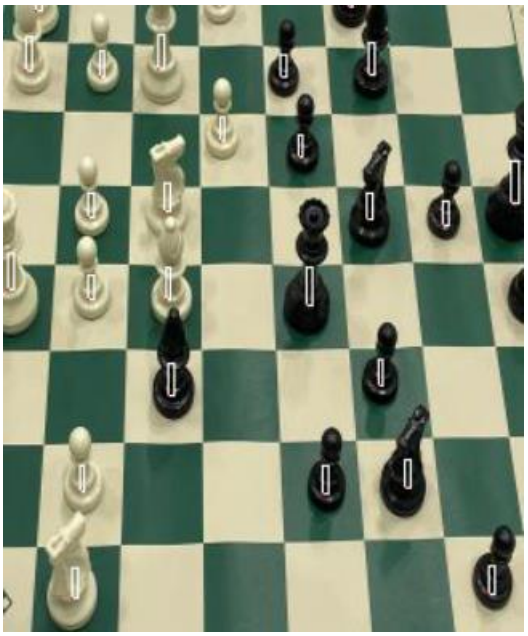

Παρατηρούμε με αυτό τον τρόπο ότι αναγνώρισε όλα τα πιόνια στην εικόνα και μάλιστα με μεγάλη ακρίβεια.

Έχοντας αναγνωρίσει τα πιόνια με σειρά ο αλγόριθμος έχει αναγνωρίσει και τις θέσεις του κάθε πιονιού στη σκακιέρα.

Με την ολοκλήρωση τις αναγνωρίσεις ο YOLO μας επιστρέφει ένα σύνολο από πλαίσια σηματοδότησης (Bounding Boxes), οπού μέσα έχουν τις συντεταγμένες του κάθε πιονιού στην εικόνα. Έτσι, και κάνοντας την υπόθεση ότι το κάθε πιόνι θα πατάει σε μια μόνο θέση πάνω στην σκακιέρα, παίρνω ένα χαμηλό κεντρικό σημείο αναμεσά στα πλαίσια σηματοδοτήσεις οπού είναι το σημείο που ακουμπάει το πιόνι πάνω στην σκακιέρα.

Συγκεκριμένα θα αποτελείται από 2 σημεία στον (X,Y) άξονα οπού και τα αναπαριστώ στην εικόνα παρακάτω με μια κόκκινη τελεία μέσα σε ένα άσπρο πλαίσιο. Το αποτέλεσμα που θα εξάγουμε από αυτό είναι ένας χάρτης με την όνομα του πιονιού σαν κλειδί και την θέση του στους άξονες X,Y σαν το ζεύγος του.

```
#Store values of chess pices locations  in a dict (PiceName, location)
#Hightlight the position it picks (Optional)
chessPicesLocationInImage = {}
chessPicesCordinatesPointsOnImage = {}
width = img.width
height = img.height
draw = ImageDraw.Draw(img)
for i in range(len(boxes)):
    box = boxes[i]
    x1 = (box[0] - box[2] /20.0) * width
    y1 = (box[1] - box[3] /2000.0) * height
    x2 = (box[0] + box[2] /20.0) * width A
    y2 = (box[1] + box[3] /4.0) * height
    rgb = (255, 255, 255)
    draw.rectangle([x1, y1, x2, y2], outline=rgb) #visualise selected l
ocation of pice
    draw.point((x1,y2),fill=233)
    if len(box) >= 7 and class_names:
            cls_conf = box[5]
            cls_id = box[6]
            classes = len(class_names)
            offset = cls_id * 123457 % classes
            chessPicesLocationInImage[class_names[cls_id]+str(i)] = [x1
,y1,x2,y2] #Here i add the location of Pice to map
            chessPicesCordinatesPointsOnImage[class_names[cls_id]+str(i
)] = [x1,y1] #Here i add the location of Pice to map
```



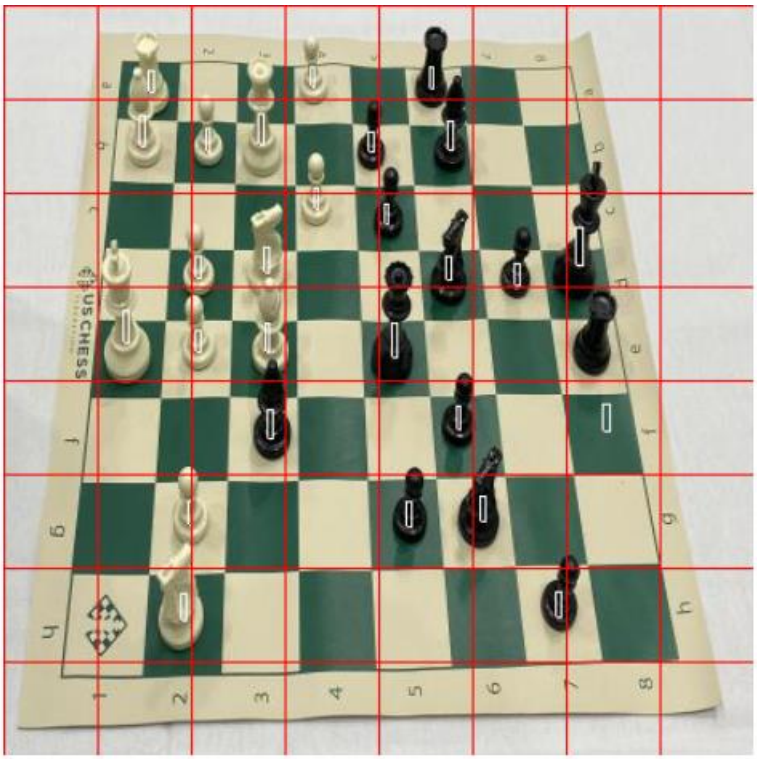
*Αναπαρασταση με κοκκινη τελεια των σημειων.*

```
{'black-bishop2': [146.9454379269951, 224.3893270305113],
 'black-bishop3': [246.55219490653587, 64.85793827003553],
 'black-king22': [317.8229667525542, 123.55869887480922],
 'black-knight23': [245.82793632494776, 139.70122974718873],
 'black-knight25': [264.55310027348366, 272.02402914378206],
 'black-pawn10': [251.25431950970702, 222.56518868921623],
 'black-pawn11': [306.69787984521764, 325.4231083753077],
 'black-pawn13': [211.49638627077402, 110.24075887144397],
 'black-pawn15': [283.9471578710957, 143.54382153996514],
 'black-pawn6': [223.8675334340648, 275.87594451088654],
 'black-pawn9': [202.79993259530318, 70.32451022369297],
 'black-queen4': [215.35184054249214, 176.07705425925946],
 'black-rook21': [236.0249449202889, 34.33132685818954],
 'black-rook8': [332.86887554494956, 221.43929037600756],
 'white-bishop1': [145.1640742665843, 176.87886159686664],
 'white-bishop5': [75.74216451958605, 61.87869512430931],
 'white-king12': [66.30530428635447, 169.4818387110798],
 'white-knight24': [144.93461941292412, 134.74781327600542],
 'white-knight26': [98.27235536449834, 326.77586654903075],
 'white-pawn14': [102.10718769776193, 275.4160660109237],
 'white-pawn16': [106.69842076740767, 180.26964186273122],
 'white-pawn17': [112.70952125536768, 68.8496903351263],
 'white-pawn18': [170.71426066975843, 34.715027989800035],
 'white-pawn19': [172.42917907363488, 101.58512969550334],
 'white-pawn20': [106.75302650803015, 139.52479125480903],
 'white-queen0': [141.61410700396488, 60.30475407815293],
 'white-rook7': [80.47609334870388, 36.927828081658014]}
```

*Χάρτης με όνομα του πιονιού και θέση σημείου*

Στην συνέχεια, για να βρούμε όλες τις πιθανές σκακιστικές θέσεις πάνω στην σκακιέρα της εικόνας, δημιουργήσουμε έναν ακόμη χάρτη οπού θα περιέχει τις συνταραγμένες όλων τον θέσεων αυτών. Θα περιέχει δηλαδή το όνομα τις θέσεις, αλλά και τα σημεία (X1,X2) (Y1,Y2) που βρίσκονται, έτσι ώστε να τα συσχετίσουμε με τις θέσεις των πιονιών και να βρούμε σε ποια θέση βρίσκετε το κάθε μας πιόνι.

```
#drwas all possible positions
#creates a map ,dict of all posible locations
letters = ['A','B','C','D','E','F','G','H']
chessBoardPicesLocationCordsDict = {}
chessBoardPicesLocationCordsDictForPychess = {}
step = height/8
rgb = (255, 0, 0)
y1=364
y2=416
x1=0
x2=0
count = 0
for x in range(0, 8):
  x1=0
  x2=52
  for y in range(0, 8):
    currentLetter = letters[y]
    chessBoardPicesLocationCordsDict[currentLetter,x+1] = [x1,y1,x2,y2]
 #make dict so it say ex (A.B)
    chessBoardPicesLocationCordsDictForPychess[count] = [x1,y1,x2,y2]
    draw.rectangle([x1, y1, x2, y2], outline=rgb)
    x1+=52
    x2+=52
    count+=1


  y1-=52
  y2-=52
```



*Όλες οι πιθανές θέσεις τις σκακιέρας.*

Έχοντας ένα χάρτη με την ακριβή θέση των πιονιών και έναν χάρτη με τις όλες τις πιθανές θέσεις της σκακιέρας, μένει μόνο να συσχετίσουμε αυτά τα δυο για να δούμε σε πιο κομμάτι τις σκακιέρας βρίσκετε το έκαστοτε πιόνι.

```
def findCorrespoindgLocation(piceXYLocation):
  for key,value in chessBoardPicesLocationCordsDict.items():
    if(value[0] < piceXYLocation[0] < value[2] and value[1] < piceXYLoc
ation[1] < value[3]):
     return  key
```

Τελευταίο βήμα του αλγορίθμου μας είναι να μετατρέψουμε τις εικόνες σε μορφή τέτοια που θα περιέχει όλες τις απαραίτητες πληροφορίες από το ταμπλό μας. Αυτή η μορφή είναι η FEN. Έτσι, θα μπορούμε να αποθηκεύουμε σε ένα String την κάθε κατάσταση τις σκακιέρας μας. Αυτό γίνεται ευκολά εφικτό με την βιβλιοθήκη pychess.

```
import chess

board = chess.Board()


def pieceColor(pieceName):
 if 'white' in pieceName:
   return True
 return False


def pieceValue(pieceName):
 if 'pawn' in pieceName:
   return 1
 if 'knight' in pieceName:
   return 2
 if 'bishop' in pieceName:
   return 3
 if 'rook' in pieceName:
   return 4
 if 'king' in pieceName:
   return 5
 if 'queen' in pieceName:
   return 6
```

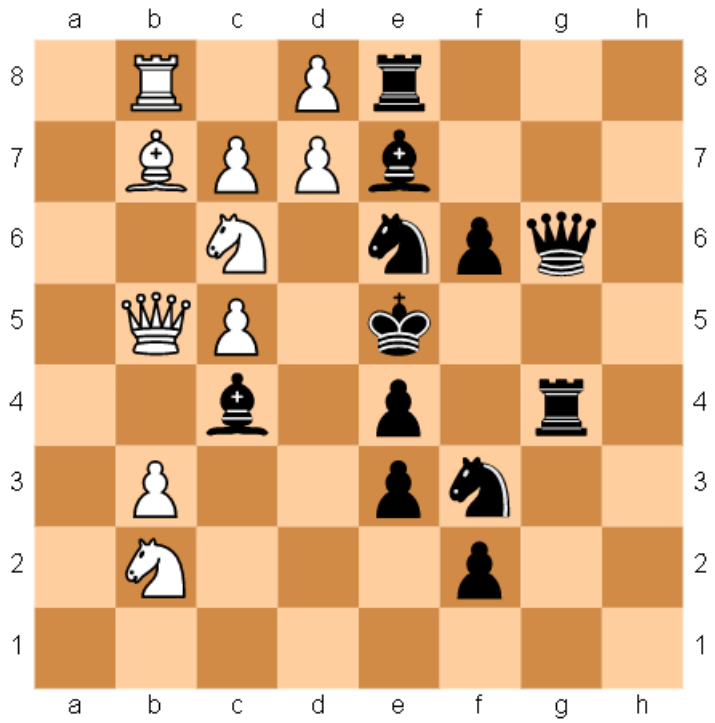
```
 return 1

for name,location in finalPicesLocationNamePychess.items():

  board.set_piece_at(location[0],chess.Piece(pieceValue(name),pieceColo
r(name)))

board.fen()
```

Αυτό που κάνουμε στο παραπάνω block κώδικα είναι ότι από το χάρτη που έχουμε δημιουργήσει πιο πάνω (finalPicesLocationNamePychess) περνούμε το κάθε πιόνι που έχουμε και με την board.set_piece το προσθέτουμε στην σκακιέρα μας.

Και έτσι με την εντολή board.fen() μπορούμε να πάρουμε την σκακιέρα τροποποιημένη σε μορφή FEN.

```
R2P1r2/BPK1pb2/3Pp2q/1PN2np1/QPB1k3/2b2p1r/1P2pn2/1N4p1 w - - 0 1
```

*Αναπαράσταση του τελικού ταμπλό σε μορφή FEN*



*Γραφική αναπαράσταση του τελικού ταμπλό*

**Συμπεράσματα και αξιολόγηση αλγορίθμου**

Για την δοκιμή του αλγορίθμου δημιούργησα έναν φάκελο με εικόνες από διαφορετικά και διάσπαρτα στιγμιότυπα από μια παρτίδα σκάκι. Ο αλγόριθμος αντλούσε τις εικόνες αυτές, τις πέρναγε μέσα από το μοντέλο και έπειτα τις έκανε μετατροπή σε μορφή FEN.

Για την δοκιμή αυτή έκανα την υπόθεση ότι όλες οι εικόνες είναι τιμές μορφής JPG και ότι το φίλτρο που θα αναγνώριζε την θέση και τα πιόνια θα ήταν ίδιου μεγέθους με την εικόνα δηλαδή 608x608.

Τα συμπεράσματα ήταν τα εξής που προέκυψαν ήταν τα ακόλουθα:

- Ο Αλγόριθμος κατάφερε να αναγνωρίσει τα πιόνια από τις σκακιέρα μας με ποσοστό 100%
- Σε μεμονωμένες περιπτώσεις δεν κατάφερε να βρει την ακριβής θέση του πιονιού
- Οι συνολική απόδοση του αλγορίθμου από το να περάσει την εικόνα από το μοντέλο έως και να την βγάλει σε μορφή Fen είναι περίπου 5 εικόνες / Sec

**Παρατηρήσεις**

Το γεγονός ότι βρίσκει μεγάλη ακρίβεια στα δεδομένα μας, δίνει την υποψία ότι αλγόριθμος τείνει να κάνει overfitting. Πράγματι, ο αλγόριθμος έχει εκπαιδευτεί σε ένα πολύ συγκεκριμένο τύπο συνόλου σκακιού που όμως αντιπροσωπεύει τα περισσότερα σύνολα, μιας και η βασική μορφή των πιονιών είναι αυτή που χρησιμοποιεί και ο αλγόριθμος.

Η τοποθέτηση του φίλτρου είναι πολύ σημαντικό κομμάτι μιας και μια μικρή απόκλιση μπορεί να αλλοιώσει το αποτέλεσμα του αλγορίθμου. Η σωστή τοποθέτηση του όμως μπορεί να δώσει ακριβή και ορθά αποτελέσματα.

# Επίλογος

Φτάνοντας στο τέλος της παρούσας διπλωματικής, χρησιμοποιώντας τον αλγόριθμο YOLO καταφέραμε να κατηγοριοποιήσουμε τόσο τις θέσεις αλλά και τις κλάσεις από σκακιστικά κομμάτια πάνω σε μια σκακιέρα. Το πλεονέκτημα που αποκομίσαμε είναι η μεγάλη ακρίβεια καθώς και η ταχύτητα του συγκριτικά με άλλους αλγορίθμους ταξινόμησης. Χρησιμοποιώντας τον συγκεκριμένο αλγόριθμο, μπορεί κάποιος να δημιουργήσει ένα σύνολο εικόνων από διάφορα στιγμιότυπα μιας παρτίδας σκάκι και να τα μετατρέψει σε ένα σύνολο αλφαριθμητικών δεδομένων που μπορούν να χρησιμοποιηθούν τόσο στην αναπαράσταση της σκακιέρας ηλεκτρονικά όσο και για εκπαιδευτική μάθηση. Ουσιαστικά αυτό που επιτύχαμε είναι ότι έγινε μια μηχανοποίηση και αυτοματοποίηση της ανθρώπινης όρασης μέσω μιας διαδικασίας απεικόνισης της καταγραφής όλων αυτών των πληροφοριών με χρήση τεχνητής νοημοσύνης.

**Μελλοντικές Παρατηρήσεις**

Οι εικόνες που εισέρχονται στο αλγόριθμο πηγάζουν από συγκεκριμένη θέση κάμερας τοποθετημένη πάνω στην σκακιέρα. Έτσι, είναι πιθανό πέρα από το happy scenario του να υπάρχουν μόνο πιόνια πάνω στην σκακιέρα, να υπάρχουν και διάφορες άλλες πιθανές καταστάσεις που ενδεχομένως να αλλοιώσουν το αποτέλεσμα της ροής του παιχνιδιού. Ένα πολύ σημαντικό παράδειγμα σε αυτό είναι η παρουσία χεριών την ώρα που μετακινούν τα πιόνια. Θα πρέπει δηλαδή να υπάρχει ένας επαληθευτής οπού πριν μπει η εικόνα στον αλγόριθμο να φιλτράρετε και να εντοπίζει καθώς και να αφαίρει μη έγκυρες εικόνες.

Κατά την διάρκεια του παιχνιδιού η κάμερα θα βγάζει συνεχόμενες εικόνες από το ταμπλό. Ο αλγόριθμος ενδέχεται να πάρει επαναλαμβανόμενα ιδίες εικόνες. Σημαντική επέκταση για τον αλγόριθμο θα ήταν η αναγνώριση των επαναλαμβανομένων θέσεων όπως και η αφαίρεση αυτών. Ενδεικτικά σε αυτό το βήμα θα μπορούμε να δούμε και την εγκυρότητα των κινήσεων που γίνονται βάση της προηγουμένης θέσης των πιονιών.

Μια πολύ σημαντική επέκταση είναι η ρύθμιση του μεγέθους του φίλτρου από τον χρήστη. Ο χρήστης να μπορεί δηλαδή να αλλάξει τις διαστάσεις του φίλτρου όσο και την γωνιά που θα παρατηρεί τα πιόνια έτσι ώστε να εξυπηρετεί τις ανάγκες του.

# Βιβλιογραφικές Πηγές


**Ελληνικές**

[11] Συλλογικό, Τεχνητή Νοημοσύνη (Δ' Έκδοση), Πανεπιστήμιο Μακεδονίας, 2011

[21] Αβούρης Ν., Κουκιάς Μ., Παλιούρας Β., Σγάρμπας Κ., Python – Εισαγωγή στους υπολογιστές 4η αναθεωρημένη έκδοση, Πανεπιστημιακές εκδόσεις Κρήτης, 2018

**Ξένες**

[1] Wasik S, Fratczak F, Krzyskow J, Wulnikowski J, Inferring Mathematical Equations Using Crowdsourcing. PLoS ONE, 2015.

[5] Arthur L Samuel, "Some studies in machine learning using the game of checkers", IBM Journal of research and development, vol. 3, no. 3, pp. 210–229, 1959.

[6] Tom Michael Mitchell, The discipline of machine learning, vol. 9, Carnegie Mellon University, School of Computer Science, Machine Learning Department Pittsburgh, PA, 2006.

[7] Stuart Russell, Peter Norvig and Artificial Intelligence, "A modern approach", Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, vol. 25, no. 27, pp. 79–80, 1995.

[8] Igor Aizenberg, Naum N Aizenberg and Joos PL Vandewalle, Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications, Springer Science & Business Media, 2013.

[9] Rina Dechter, Learning while searching in constraint-satisfaction problems, University of California, Computer Science Department, Cognitive Systems Laboratory, 1986.

[12] David A. Forsyth, Jean Ponce, Computer Vision: A Modern Approach (2nd Edition). Pearson, 2011.

[13 ] Yali Amit and Pedro Felzenszwalb, "Object Detection", Computer Vision: A Reference Guide, pp. 537–542, 2014.

[17] Joseph Redmon and Ali Farhadi, "YOLO9000: better, faster, stronger", arXiv preprint, 2017.

[19]Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi, "You only look once: Unified, real-time object detection", in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.

[20] Yuxin Wu and Kaiming He, "Group normalization", in Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19, 2018

[24] McCarthy J., Wright P., Technology as Experience. MIT Press, 2004

[25] Rojas I., Joya G., Catala A., Advances in Computational Intelligence: 15th International Work-Conference on Artificial Neural Intelligence, 2019

**Ηλεκτρονικές**

[2] https://homes.di.unimi.it/borghese/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf

[3] https://www.csee.umbc.edu/courses/471/papers/turing.pdf

[4] http://aima.cs.berkeley.edu

[10] https://deeplearning.mit.edu/

[14]https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detectionalgorithms-36d53571365e

[15]https://arxiv.org/abs/1311.2524

[16] https://arxiv.org/abs/1504.08083

[18] You Only Look Once: Unified, Real-Time Object Detection https://arxiv.org/abs/1506.02640

[22] https://roboflow.com/features

[23]https://towardsdatascience.com/how-to-train-scaled-yolov4-to-detect-custom-objects-13f9077ebc89