Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών

Τεχνικές ταυτοποίησης οντοτήτων σε πραγματικό χρόνο μέσω όρασης υπολογιστών

Διπλωματική Εργασία
του
Γιαννιτσόπουλου Εμμανουήλ

Επιβλέπουσα:
Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π

Αθήνα, Ιούλιος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών

Τεχνικές ταυτοποίησης οντοτήτων σε πραγματικό χρόνο μέσω όρασης υπολογιστών

Διπλωματική Εργασία

του

Γιαννιτσόπουλου Εμμανουήλ

Επιβλέπουσα:
Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π

Εγκρίθηκε από τριμελή επιτροπή την .....

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....
Θεοδωρα Βαρβαρίγου	Εμμανουήλ Βαρβαρίγος	Συμεών Παπαβασιλείου

Αθήνα, 13 Ιουλίου 2020

.....

Γιαννιτσόπουλος Εμμανουήλ

Περίληψη

Η ανάπτυξη των υπολογιστικών συστημάτων τα τελευταία χρόνια έχει οδηγήσει στην δυνατότητα υλοποίησης μηχανικής εκπαίδευσης στην επίλυση προβλημάτων. Ένας από τους βασικούς τομείς όπου μπορεί να δημιουργηθεί αλγόριθμος μέσω εκπαίδευσης είναι η αναγνώριση προτύπων σε εικόνες. Έτσι έχουν αναπτυχθεί πολλές προσεγγίσεις και αρχιτεκτονικές πάνω στην δημιουργία νευρωνικών δικτύων προς εκπαίδευση ώστε να επιτελείται η αναγνώριση προτύπων. Οι εφαρμογές που μπορούν να χρησιμοποιηθούν τέτοιες τεχνολογίες αναγνώρισης προτύπων είναι πάρα πολλές.

Στην παρούσα εργασία επιλέγεται να υλοποιηθεί σύστημα αναγνώρισης δεκαδικών ψηφίων και εν συνεχεία εύρεση του κωδικού αθλητών που φαίνονται σε βίντεο αγώνων δρόμου. Αρχικά αναλύονται σε θεωρητικό επίπεδο οι περισσότερες από τις σύγχρονες τεχνολογίες αναγνώρισης αντικειμένων σε εικόνα, και λαμβάνοντας υπόψη κάποια κριτήρια, επιλέγεται η τεχνολογία retinanet για την υλοποίηση της λύσης του προβλήματος που ερευνάται. Στην συνέχεια αναφέρονται οι προδιαγραφές του συστήματος, περιγράφεται η προετοιμασία και η διαδικασία της εκπαίδευσης του νευρωνικού δικτύου, και αναλύεται η εφαρμογή του αλγορίθμου που δημιουργήθηκε πάνω σε εικόνες αθλητών. Τέλος επεκτείνεται ο αλγόριθμος αυτός ώστε να εντοπίζονται αθλητές σε βίντεο.

Λέξεις κλειδιά: αναγνώριση προτύπων, μηχανική μάθηση, δεκαδικά ψηφία, εικόνα, βίντεο, αθλητές.

Abstract

The growth of computer systems the last decade has given the possibility to create machine learning algorithms and neural networks. One of the main subjects that take advantage of these technologies is the object detection in images and videos. Many architectures and algorithms have been created recently and they can be used in a wide variety of applications

In this thesis a model of detecting decimal digits in images is created, and is used to recognise the full numbers of athletes in videos o street racing events. Firstly we analyse most of the contemporary object detection technologies, and after taking into consideration some parameters we choose the retinanet technology to implement the solution of our project. Afterwards, we mention the requirments of the projects system, we describe the preparation and the procedure af the machine learning of the neural network, and analyse the algorithm that was created after executing it in athlete's photos. Finally we extend this algorithm so it recognises athletes in videos.

Keywords: object detection, digit recognition, machine learning, retinanet, decimal digits, image, video, athletes.

Ευχαριστίες

Ευχαριστώ την καθηγήτρια Θεοδώρα Βαρβαρίγου που μου έδωσε την δυνατότητα ανάπτυξης ενός θέματος μηχανικής εκμάθησης και αναγνώρισης προτύπων για την διπλωματική εργασία μου.

Ευχαριστώ επίσης την διοργάνωση αγώνων δρόμου Poseidon Athens Half Marathon που μας παρείχε πλήθος δεδομένων εικόνας και βίντεο από διοργανώσεις προηγούμενων ετών, και ήταν πάρα πολύ σημαντικά στην εκπόνηση της εργασίας.

Τέλος θέλω να ευχαριστώ τα παιδιά του εργαστηρίου Distributed Knowledge and Media Systems για την συνεργασία και την βοήθεια τους. Ειδικότερα ευχαριστώ τον Τάσο Νικολακόπουλο και τον Βρεττό Μούλο που κάτω από την καθοδήγηση τους επετεύχθη ο σκοπός.

Περιεχόμενα

1 Εισαγωγή

Η ανάπτυξη των υπολογιστικών συστημάτων την τελευταία δεκαετία είναι ραγδαία, τόσο πού κατάφερε να δώσει την δυνατότητα υλοποίησης θεωρητικών μοντέλων, και στατιστικών διαδικασιών όπως είναι τα νευρωνικά δίκτυα (neural networks) και μέσα από αυτό να έρθει η ανάπτυξη του machine learning. Έχουν δημιουργηθεί αλγόριθμοι, που τροφοδοτούμενοι με μεγάλο αριθμό κατάλληλων δεδομένων, μπορούν να ¨εκπαιδευτούν' στο να επιτελούν συγκεκριμένες διαδικασίες. Έτσι πλέον, αυτόματες διαδικασίες αναγνώρισης αντικειμένων σε φωτογραφίες είναι εύκολες και γρήγορες στην υλοποίηση.

Καθώς ο τομέας της αναγνώρισης προτύπων σε φωτογραφίες, και κατ επέκταση σε βίντεο βρίσκεται σε άνθηση, υπάρχουν πολλές προσεγγίσεις αλγορίθμων σε θέματα αρχιτεκτονικής και εκπαίδευσης των δικτύων αυτών, οι οποίες προσφέρουν διαφορετικά θετικά χαρακτηριστικά σε σχέση με τις υπόλοιπες και βελτιώνονται συνεχώς. Για να μπορέσει να οριστεί η απόδοση όλων αυτών των προσεγγίσεων έχουν δημιουργηθεί διάφορα πρότυπα αναγνώρισης και δεδομένων (Pascal VOC, COCO, ...) όπου οδήγησαν στην δημιουργία προ-εκπαιδευμένων μοντέλων για την αναγνώριση αντικειμένων, πάνω στα οποία πατάνε οι οι δημιουργοί των αλγορίθμων αναγνώρισης, και έτσι παρέχουν έτοιμες υλοποιήσεις τις οποίες οι χρήστες μπορούν να χρησιμοποιήσουν στις εφαρμογές τους. Αυτά τα pretrained μοντέλα δίνουν την δυνατότητα αναγνώρισης συνήθως μέχρι 80 κοινών αντικειμένων όπως άνθρωποι, αυτοκίνητα, ζώα, φαγητά και άλλα κοινά αντικείμενα.

Βασικό κομμάτι της δημιουργίας αυτών των τεχνολογιών, εκτός του γρήγορου και αξιόπιστου εντοπισμού αντικειμένων, είναι η βέλτιστη διαδικασία εκπαίδευσης και η μείωση του χρόνου της διαδικασίας αυτής. Εκτός των προεκπαιδευμένων μοντέλων από τους δημιουργούς κάθε τεχνολογίας που περιγράφηκαν προηγουμένως, δίνεται η δυνατότητα εκπαίδευσης σε πιο στοχευμένα αντικείμενα και αντίστοιχη κατάλληλη αξιοποίηση σε εφαρμογές. Μία τέτοια στοχευμένη εφαρμογή υλοποιήθηκε στην παρούσα εργασία.

Σημαντικό ρόλο στην σύγχρονη κουλτούρα έχουν οι αθλητικές διοργανώσεις, τόσο σε επαγγελματικό επίπεδο όσο και σε ερασιτεχνικό. Ένας τομέας που είναι ιδιαίτερα δημοφιλής τα τελευταία χρόνια είναι οι αγώνες δρόμου μεγάλων και μικρών αποστάσεων. Συμμετέχουν εκατοντάδες άνθρωποι, από επαγγελματίες δρομείς μέχρι γονείς με τα παιδιά τους. Ο μεγάλος αριθμός συμμετεχόντων, κυρίως κοινών ανθρώπων, δίνει την δυνατότητα ανάπτυξης συ-

στημάτων για καλύτερη καταγραφή των επιδόσεων, την ψηφιακή καταγραφή εικόνας και βίντεο όλων των συμμετεχόντων κατά την διάρκεια του αγώνα, και την αξιοποίηση όλων αυτών των πραγμάτων με διάφορους τρόπους μετά το τέλος της διοργάνωσης. Διαπιστώνουμε ότι μια διαδικασία που θα καταφέρει να αυτοματοποιήσει πολλές από τις διαδικασίες είναι η αυτόματη αναγνώριση του κάθε αθλητή σε φωτογραφίες και βίντεο από τον αγώνα. Η αναγνώριση αυτή γίνεται μέσω του κωδικού που έχουν οι αθλητές στα ταμπελάκια που φοράνε πάνω στις μπλούζες τους. Επομένως δημιουργείται η ανάγκη ύπαρξης ενός αλγορίθμου που να εντοπίζει και να ξεχωρίζει τα 10 ψηφία του δεκαδικού συστήματος αριθμών, και με κάποιον τρόπο να δημιουργεί τον συνολικό αριθμό του κάθε κωδικού.

Το πρώτο επίπεδο του προβλήματος, η αναγνώριση των 10 ψηφίων από το 0 μέχρι το 9, θα γίνει επιλέγοντας μία από τις υπάρχουσες τεχνολογίες αναγνώρισης προτύπων, και εκπαιδεύοντας την σε ένα dataset φωτογραφιών που περιέχουν ψηφία, κατάλληλα επεξεργασμένων ώστε να είναι ικανά στην εκπαίδευση του νευρωνικού δικτύου. Τα κριτήρια επιλογής του κατάλληλου αλγορίθμου θα μπορούσαμε να τα συνοψίσουμε σε ότι έχει να κάνει με την απόδοση της αναγνώρισης σε μικρά αντικείμενα σε μία εικόνα, όπως επίσης και στην ασύγχρονη επεξεργασία, η οποία μας απαλλάσσει από περιορισμούς και θυσία απόδοσης εύρεσης για μεγάλη ταχύτητα επεξεργασίας αντίστοιχης με τον frame rate ενός βίντεο. Η ασύγχρονη επεξεργασία είναι μία υπαρκτή επιλογή στο πρόβλημα μας καθώς στους αγώνες μεγάλων αποστάσεων, και ειδικότερα στις διοργανώσεις που απευθύνονται σε μη επαγγελματίες αθλητές, δεν υπάρχει τηλεοπτική κάλυψη ούτε υπάρχει η ανάγκη εξαγωγής πληροφοριών του είδους που μελετάμε την ώρα του αγώνα. Η διαδικασία της εκπαίδευσης θα μπορούσε να παίξει κ αυτή ρόλο στην επιλογή της τεχνολογίας που θα χρησιμοποιηθεί, καθώς προσφέρονται διαφορετικοί χρόνοι εκπαίδευσης και διαφορετικά ελάχιστα system requirements για να είναι δυνατή η εκπαίδευση ειδικά σε ένα απαιτητικό dataset, όπως αυτό που χρειαζόμαστε εδώ, αλλά οι τεχνολογίες που καταλήγουμε από το πρώτο επίπεδο διαχωρισμού, έχουν παραπλήσια εκπαίδευση.

2 Object Detection

Η αναγνώριση προτύπων σε εικόνα, είναι μια τεχνολογία computer vision και image processing που βασίζεται σε θεωρίες βιολογίας και νευροεπιστημών που ξεκίνησαν ήδη από το 1960, για το πως αντιλαμβάνεται ο εγκέφαλος σχήματα, γραμμές, γωνίες κλπ, και άρχισε να αναπτύσσεται μετά το 2000. Η βελτίωση και αύξηση της υπολογιστικής ισχύος έκανε δυνατή την υλοποίηση αλγορίθμων οι οποίοι μπορούν να βγάλουν αποτελέσματα σε πολύ μικρό χρόνο, και πλέον σε πραγματικό χρόνο, ακόμη και με χρήση πολύ μικρών και ενσωματωμένων συστημάτων (πχ Raspberry Pi). Σημαντική ώθηση στην τεχνολογική αυτή κατεύθυνση έδωσε η δυνατότητα υλοποίησης στατιστικών μεθόδων και δημιουργία patterns και αλληλεπιδράσεις στοιχείων (Machine Learning), καθώς και εξέλιξη της τεχνολογίας αυτής (Deep Learning), η οποία χρησιμοποίηση περισσότερα επίπεδα(layers) στον αλγόριθμο, και δημιούργησε την έννοια του Νευρωνικού δικτύου.

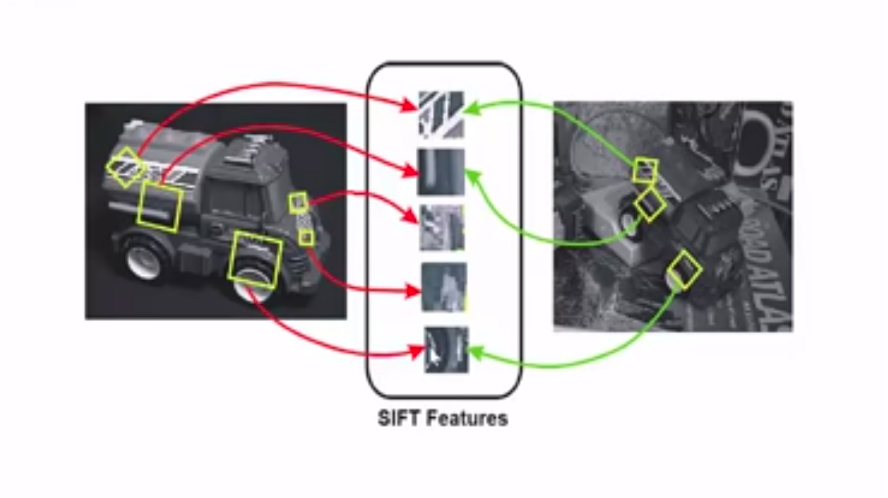
Οι βάσεις για τα CNN (Convolutional Neural Networks), δόθηκαν από τον Kunihiko Fukushima το 1979, και το νευρωνικό δίκτυο που πρότεινε (Neocognitron), το οποίο έβρισκε μοτίβα σε εικόνες χωρίς να επηρεάζεται από μετακίνηση θέσης αυτών. Σημαντική ήταν η συνεισφορά του Yann LeCun που το 1989 εξέλιξε την δουλειά του Fukushima, εφαρμόζοντας αντίστροφη εκμάθηση στον αλγόριθμο, και δημιούργησε το πρώτο ουσιαστικά convnet, του οποίου στοιχεία χρησιμοποιούνται ακόμη και σήμερα στα CNNs.

2.1 Object Recognition from Local Scale-Invariant Features (SIFT)

Η τεχνική SIFT είναι βασισμένη στον προσδιορισμό χαρακτηριστικών (features) μιας εικόνας ώστε να πετύχει αναγνώριση αντικειμένου. Σε σύγκριση με τις μέχρι τότε μεθόδους που χρησιμοποιούνταν, όπως η 3D απεικόνιση χαρακτηριστικών, που είχε πρωτοαναφέρει ο David Marr το 1982, η τεχνική SIFT είναι ανεξάρτητη από αλλαγές στο scale της εικόνας, περιστροφής, και αλλαγών στον φωτισμό.

Κάθε αντικείμενο μιας εικόνας έχει κάποια κύρια χαρακτηριστικά, όπως γωνίες, ακμές και καμπύλες, τα οποία εξάγονται σε μορφή διανυσμάτων(vectors). Σε πρώτο επίπεδο, βρίσκει σημαντικά σημεία στον scale χώρο ψάχνοντας μέγιστα και ελάχιστα σε διαφορική-Γκαουσιανή συνάρτηση. Κάθε εικόνα παράγει

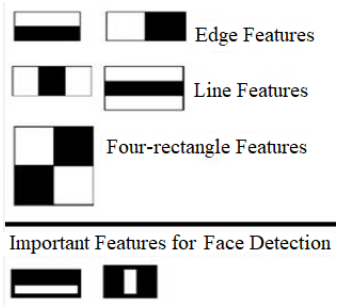
1000 SIFT keys. Η σύγκριση των σημείων αυτών κάνει την σύνδεση με το πιθανό αντικείμενο, χρησιμοποιώντας πίνακας μεταμόρφωσης Hough (Hough transform table). Αν συμφωνούν πάνω από 3 keys, τότε είναι δυνατή απόδειξη ύπαρξης του αντικειμένου. [1]



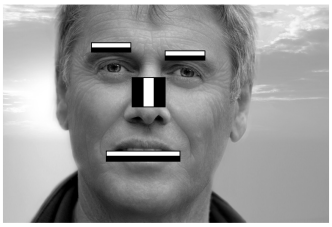
Σχήμα 1: Εξαγωγή χαρακτηριστικών SIFT

2.2 Robust Real-Time Detection (Viola-Jones framework)

Ο αλγόριθμος που δημιούργησαν οι Viola και Jones το 2001 , είναι αλγόριθμος αναγνώρισης προσώπων κυρίως. Σε πρώτο επίπεδο επιλέγονται στοιχεία Haar (Haar features). Το ανθρώπινο πρόσωπο έχει κάποια κοινά χαρακτηριστικά σε όλες τις φωτογραφίες, όπως το ότι η οριζόντια περιοχή των ματιών είναι πιο σκούρα από τα μάγουλα, ή ότι η κάθετη γραμμή της μύτης είναι πιο φωτεινή από τα μάτια. Επομένως τα στοιχειά Haar, που ταιριάζουν στην γεωμετρία των χαρακτηριστικών που επιλέξαμε, και θα εξεταστούν ως υπό-παράθυρα πάνω στην εικόνα, φαίνονται στην παρακάτω φωτογραφία.



(i) Features



(ii) Features example

Σχήμα 2: Χαρακτηριστικά Haar

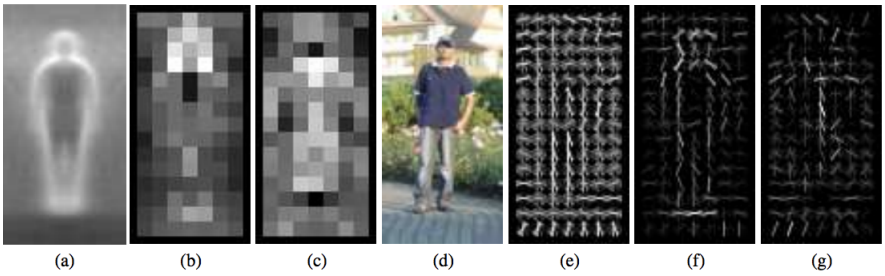
Για γρηγορότερους υπολογισμούς σε επίπεδο pixel και παραθύρων ενδιαφέροντος, ο αλγόριθμος εισήγαγε την χρήση του Integral Image. Για να μειώσουμε τον αριθμό των πράξεων ανάμεσα σε pixels, δημιουργείτε ένας νέος πίνακας, όπου κάθε στοιχείο, αντί να περιλαμβάνει την τιμή του pixel, έχει το άθροισμα επιλεγμένης δομής (πχ προηγούμενες θέσεις πινάκα, γραμμής ή στήλης)

Έκανε επίσης χρήση του αλγόριθμου AdaBoost, ο οποίος επιλέγει λιγότερα και πιο σημαντικά χαρακτηριστικά (features) που θα εξεταστούν. Έτσι έχουμε ένα πιο αποδοτικό αλγόριθμο εκμάθησης (learning algorithm).Τέλος, κατάφερε να εξετάζει περιοχές της εικόνας με μεγαλύτερο ενδιαφέρον, αφαιρώντας σε 1^o επίπεδο (layer) τα υπο-παράθυρα με περισσότερες πιθανότητες να μην περιέχουν θετικό αποτέλεσμα, σε 2^o επίπεδο να αφαιρούνται τα πιο δύσκολα, και να συνεχίζει αντίστοιχα. [2]

2.3 Histograms of Oriented Gradients for Human Detection (HOG)

Ο HOG είναι ένας descriptor ο οποίος θύμιζε σε αρκετά σημεία ιστόγραμμα ακμών(edge orientation histograms), SIFT descriptors και περιεχόμενα σχήματος (shape contexts), αλλά μπορεί να υπολογιστεί σε πυκνό πλέγμα (grid) ομοιόμορφων τοποθετημένων κελιών, και χρησιμοποιεί υπερκάλυψη για ομαλοποίηση της αντίθεσης(contrast) κατά τόπους, για να πετύχει καλύτερη απόδοση. Χρησιμοποίησαν γραμμικό SVM(Support Vector Machine).

Η βασική ιδέα είναι το σχήμα και η απεικόνιση ενός αντικειμένου μπορεί εύκολα να χαρακτηριστεί μέσω μιας διανομής τοπικών διαφορών έντασης (local intensity gradients) ή κατευθύνσεις ακμών. Αυτές ομαλοποιούνται σε σχέση με τις τιμές γειτονικών τους στοιχείων (blocks). Η τεχνική αυτή είναι ανεξάρτητη σε τοπικές γεωμετρικές και φωτομετρικές αλλαγές. Συγκεκριμένα για την αναγνώριση ανθρώπων, οι διαδικασίες της ΄παχιάς΄ χωρικής δειγματοληψίας, της καλής προσανατολισμένης δειγματοληψίας και της ισχυρής φωτομετρικής ομαλοποίησης είναι η καλύτερη στρατηγική καθώς επιτρέπει την μετατόπιση και η αλλαγή σχήματος του σώματος και των άκρων. [3]



Σχήμα 3: Τοπικά διανύσματα έντασης στο HOG

2.4 R-CNN

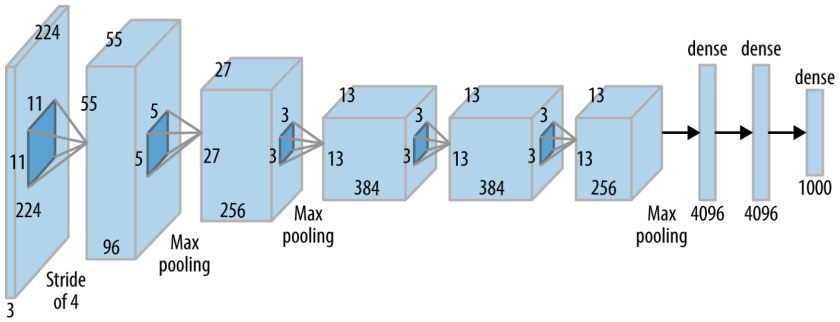
2.4.1 R-CNN

Η προσέγγιση της μεθόδου R-CNN υπήρξε καθοριστική στην ανάπτυξη και την βελτίωση της αποδοτικότητα στην αναγνώριση αντικειμένων σε εικόνες. Πρότεινε τον συνδυασμό μικρότερων κομματιών της εικόνας (regional proposals) με την αναπτυσσόμενη τεχνολογία των νευρωνικών δικτύων στην πληροφορική, δηλαδή τα CNN (Convolutional Neural Networks).

Η μέθοδος επιλογής των παραθύρων που θα χωριστεί η εικόνα δεν σχετίζεται με την τεχνολογία R-CNN, επομένως μπορεί να επιλεχθεί οποιαδήποτε. Στον αλγόριθμο που προτάθηκε χρησιμοποιήθηκε η επιλεκτική αναζήτηση (selective search) , και δημιουργούνται περίπου 2000 διαφορετικές περιοχές, αρκετά λιγότερες σε σύγκριση με την μέθοδο των κινούμενων παραθύρων σε όλη την εικόνα (sliding window).

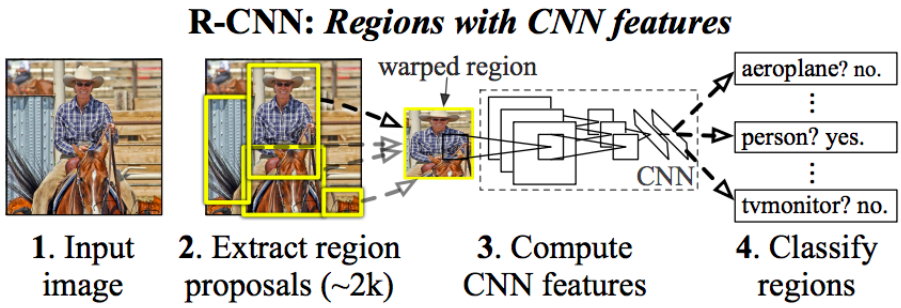
Το επόμενο βήμα είναι να πάρουμε αυτά τα κομμάτια, και χρησιμοποιώντας το νευρωνικό δίκτυο να αναπαρασταθούν με ένα διάνυσμα πολύ μικρότερου μεγέθους.

Σαν feature extractor χρησιμοποιήθηκε ο AlexNet, βασικό μειονέκτημα του οποίου είναι ότι δεν μπορεί να εκπαιδευτεί όλο το σύστημα μαζί. Το πρόβλημα αυτό θα λυθεί σε μελλοντικούς αλγόριθμους. Μετά την εκπαίδευση αφαιρείται το τελευταίο layer, και έτσι τα χαρακτηριστικά που εξάγονται είναι διάστασης 4096. Επίσης πρέπει τα regional proposals να είναι μεγέθους (227, 227, 3), επομένως πρέπει να γίνεται κατάλληλη μετατροπή σε αυτά που είναι μικρότερα ή μεγαλύτερα.



Σχήμα 4: Εξαγωγή χαρακτηριστικών AlexNet (RCNN)

Στη συνέχεια, μέσω ενός SVM classificator (Support-Vector-Machine) γίνεται κατηγοριοποίηση σε κλάσεις των διανυσμάτων αναλόγως με το τι αντιπροσωπεύουν. Χρησιμοποιούμε έναν SVM για κάθε αντικείμενο που θέλουμε να εντοπίσουμε. Η έξοδος είναι ένα confidence score. Τέλος, χρησιμοποιώντας την μέθοδο greedy non-maximum suppression επιστρέφουμε το αποτέλεσμα στην αρχική εικόνα. Σε περίπτωση επικάλυψης τμημάτων, κρατάμε αυτό με την μεγαλύτερη τιμή. [4]



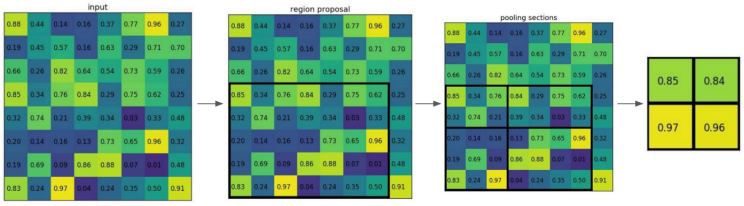
Σχήμα 5: Σύστημα αναγνώρισης προτύπων RCNN

2.4.2 Fast R-CNN

Τα βασικά αρνητικά της RCNN τεχνολογίας που περιγράφηκε προηγουμένως, είναι ότι:

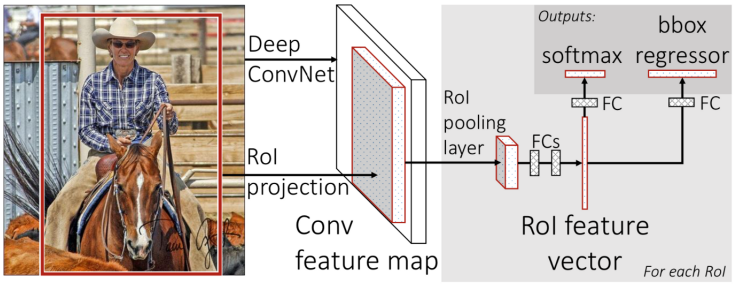
- 1. Είναι αργή καθώς χρειάζεται υπολογισμός ενός χάρτη χαρακτηριστικών (feature map) για κάθε regional proposal.
- 2. Δύσκολο στην εκπαίδευση καθώς τα 3 διαφορετικά επίπεδα που το απαρτίζουν χρειάζονται ξεχωριστή εκπαίδευση.
- 3. Έχει μεγάλες απαιτήσεις μνήμης

Για την βελτιστοποίηση των παραπάνω θεμάτων, η ομάδα που είχε δημιουργήσει τον RCNN πρότεινε και την βελτιωμένη εκδοσή του, τη Fast-RCNN. Σε αυτή, η εικόνα επεξεργάζεται ολόκληρη από το νευρωνικό δίκτυο, αποτέλεσμα της οποίας είναι ένας feature map. Σε αυτόν τον πίνακα αντιστοιχίζονται τα region proposals και προκύπτει ο region proposal feature map. Αυτά τα κομμάτια προσαρμόζονται σε κατάλληλο μέγεθος με την βοήθεια του pooling layer RoI (Region of Interest). Κάθε κομμάτι χωρίζεται σε όσα μέρη επιθυμούμε να έχουμε στο τέλος, και από τα καινούρια κομμάτια που έχουν δημιουργηθεί κρατάμε την μέγιστη τιμή από τα περιεχόμενα στοιχεία. Έτσι προκύπτει ένα διάνυσμα χαρακτηριστικών (feature vector) σταθερού μεγέθους.



Σχήμα 6: Proposal Region Map + RoI pooling (Fast R-CNN)

Αυτό το διάνυσμα είναι η είσοδος του τελευταίου επιπέδου του ανιχνευτή (detector), που αποτελείται από 2 μέρη, το softmax classification layer όπου αποφασίζεται σε ποια κλάση ανήκει το αντικείμενο που αντιπροσωπεύει το διάνυσμα, και το Bounding Box Regressor το οποίο δημιουργεί τις συντεταγμένες του περιγράμματος του αναγνωρισμένου αντικειμένου. Ο Fast-RCNN εκπαιδεύει 9 φορές γρηγορότερα ένα VGG16 νευρωνικό δίκτυο, είναι 213 φορές πιο γρήγορο στην εξαγωγή συμπεράσματος και έχει υψηλότερη μέση ακρίβεια εντοπισμού (mAP). [5]



Σχήμα 7: Αρχιτεκτονική του Fast R-CNN

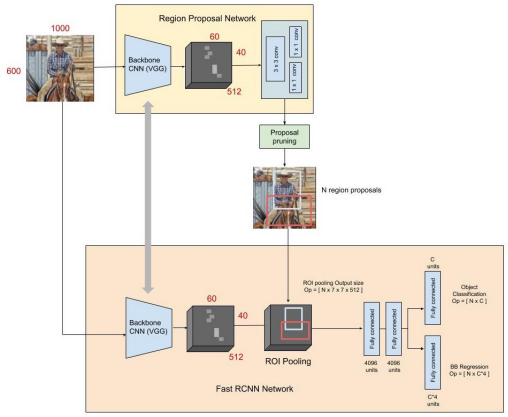
2.4.3 Faster R-CNN

Ο Faster R-CNN είναι μια βελτιωμένη εκδοχή του Fast R-CNN και επιτυγχάνει την επιτάχυνση του region proposal από 2 δευτερόλεπτα ανά εικόνα, σε 10 χιλιοστά του δευτερολέπτου. Επίσης έδωσε την δυνατότητα διαμοιρασμού layers με επόμενα επίπεδα της επεξεργασίας που οδήγησε σε γενική βελτίωση της αναπαράστασης των χαρακτηριστικών.

Το βασικότερο στοιχείο που άλλαξε ήταν η εισαγωγή ενός νευρωνικού δικτύου στην διαδικασία της τοπικής επιλογής (regional proposal). Το δίκτυο αυτό ονομάστηκε Region Proposal Network (RPN), και αυτό που κάνει είναι να δημιουργεί ένας τρισδιάστατο χάρτη χαρακτηριστικών feature map,και κάθε πίνακας αυτού αποτελεί ένα σημείο της εικόνας όπως φαίνεται παρακάτω.

Σχήμα 8: RPN anchors

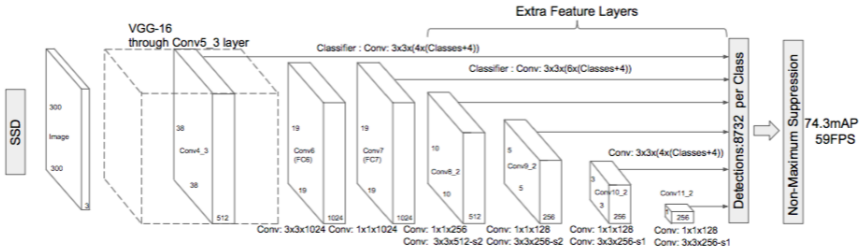
Η αρχιτεκτονική του Faster R-CNN αποτελείται από τον RPN σαν regional proposal αλγόριθμο, και ένα Fast R-CNN σαν δίκτυο αναγνώρισης.



Σχήμα 9: Αρχιτεκτονική του Faster R-CNN

2.5 SSD (Single Shot Detector)

Η βασική ιδέα είναι να χρησιμοποιήσατε ένα δίκτυο, χωρίς να χρειάζονται region proposals, αλλά αντί αυτού ο εντοπισμός της θέσης του αντικειμένου localization και η τοποθέτηση σε κλάση classification του αντικειμένου να γίνεται σε ένα πέρασμα του δικτύου. Η αρχιτεκτονική του SSD βασίζεται σε αυτή του νευρωνικού δικτύου VGG-16, καθώς αυτό έχει πολύ καλή ποιότητα στην κατηγοριοποίηση εικόνων. Η διαφοροποίηση έγκειται στο ότι αντί για εξ ολοκλήρου συνδεδεμένα επίπεδα layers δικτύου, προστέθηκαν εδώ κάποια layers που επιτρέπουν την εξαγωγή χαρακτηριστικών σε πολλαπλές διαστάσεις, και σιγά σιγά να μειώνουν το μέγεθος της κάθε εισόδου.



Σχήμα 10: Αρχιτεκτονική του SSD

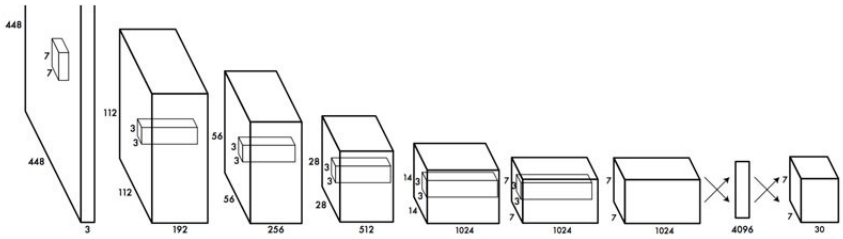
Ενδιαφέρον παρουσιάζει ο τρόπος που ο SSD διαχειρίζεται το πρόβλημα των bounding boxes. Ο αλγόριθμος συνδέει ένα σετ από σταθερά bounding boxes διαφορετικών διαστάσεων και κλίμακας σε κάθε πίνακα χαρακτηριστικών feature map, κ έτσι στο πέρασμα εις βάθος της αρχιτεκτονικής εντοπίζονται αντικείμενα από μικρή κλίμακα (0.1 - 0.2) και πηγαίνοντας προς τα επίπεδα δεξιά φτάνει ως 0.9.

Η αποδοτικότητα του SSD, μετρήθηκε σε υψηλή ακρίβεια αναγνώρισης με ταχύτητα 59 FPS και 74.3% mAP, σε σύγκριση με τον Fast R-CNN που τρέχει σε 7FPS με 73.2% mAP και τον YOLO που τρέχει σε 45FPS με 63.4% mAP0 [6]

2.6 YOLO (You Only Look Once)

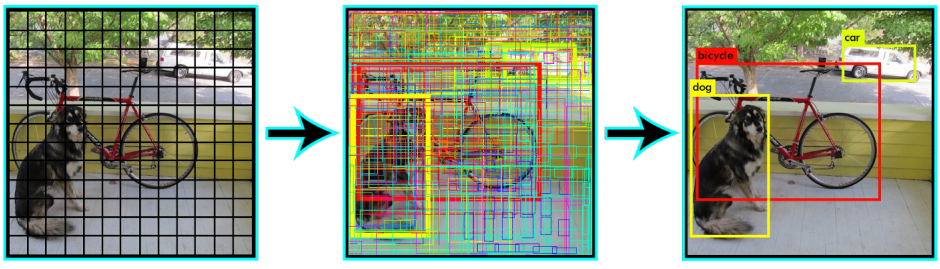
2.6.1 YOLO (Version 1)

Ο αλγόριθμος YOLO, έδωσε μία νέα οπτική στην προσέγγιση της αναγνώρισης αντικειμένων σε εικόνα. Η βασική ιδέα είναι πως ένα μόνο νευρωνικό δίκτυο CNN προβλέπει ταυτόχρονα, κατευθείαν από τα pixel της εικόνας τις συντεταγμένες του κουτιού οριοθέτησης bounding box, αλλά και τις πιθανότητες να περιλαμβάνεται μια από τις κλάσεις αναζήτησης μέσα σε αυτό το κουτί. Η απόδοση που επιτεύχθηκε ήταν 155 FPS με σχεδόν διπλάσιο mAP σε σύγκριση με άλλες παρόμοιες τεχνολογίες.



Σχήμα 11: Αρχιτεκτονική του YOLO

Το σύστημα που δημιουργήθηκε, χωρίζει την εικόνα σε ένα πλέγμα grid μεγέθους SxS. Κάθε κουτί προβλέπει B bounting boxes, και confidence scores, επομένως 5 τιμές, x, y, w, h, score. Επίσης υπολογίζει την πιθανότητα ύπαρξης κάθε διαφορετικής κλάσης που ορίζεται στο μοντέλο που ακολουθείται. Για Pascal VOC που έχουμε 20 διαφορετικές κλάσεις, C)=20. Επομένως το μέγεθος του τανυστή που δημιουργείται είναι [SxSx(B*5+C)].



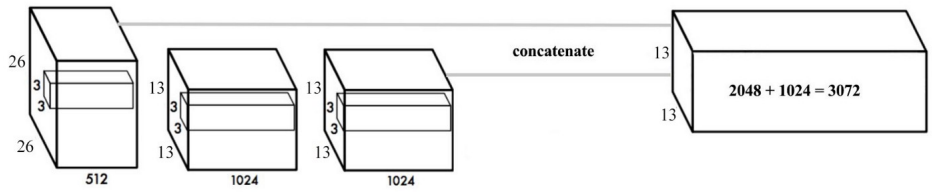
Σχήμα 12: Διαδικασία δημιουργίας πλέγματος (grid) στον YOLO

Το μεγάλο πρόβλημα του YOLO είναι ότι έχει πολλά λάθη τοπικού προσδιορισμού, όπως και εντοπισμού πολλών ίδιων αντικειμένων, όταν αυτά βρίσκονται κοντά το ένα με το άλλο. [7]

2.6.2 YOLO 9000 (Version 2)

Στην $2^η$ εκδοχή του YOLO βασικός σκοπός ήταν η επίτευξη ακόμη πιο γρήγορη και αξιόπιστη αναγνώριση, ενός ακόμη μεγαλύτερου σετ αντικειμένων. Αυτό δίνει λύση στο είδος των αλγορίθμων που κάνουν και εντοπισμό αντικειμένου σε μία εικόνα, αλλά και προσδιορισμού της κλάσης του αντικειμένου, πράγμα το οποίο μέχρι τότε γινόταν για μικρό αριθμό αντικειμένων.

Οι βελτιστοποίηση στο θέμα της ακρίβειας πρόβλεψης επετεύχθη κατ΄ αρχήν με ομαλοποίηση του μεγέθους δέσμης batch normalization στα layers του νευρωνικού δικτύου. Αυτό οδήγησε σε μία αύξηση του mAP κατά 2%. Στην νέα έκδοση χρησιμοποιούνται μεγαλύτερης ανάλυσης classifiers κατά την εκπαίδευση του δικτύου, και αντί για εκπαίδευση με εικόνες ανάλυσης 224x224 ακολουθούμενες από εικόνες 448x448, έχουμε εκπαίδευση με εικόνες 224x224 και μετά επαναδιαμόρφωση του ίδιου classifier με εικόνες 448x448, αλλά λιγότερες επαναλήψεις epochs. Αλλάχτηκε επίσης το τελευταίο μέρος του δικτύου, όπου γινόταν πρόβλεψη του boundary box, και αντικαταστάθηκε από νέο, ώστε να μην γίνονται αυθαίρετες προβλέψεις, αλλά να ικανοποιούν την λογική της χωρικής τοποθέτησης του κάθε διαφορετικού αντικειμένου στην καθημερινή ζωή, που είναι σχεδόν πάντα ίδιο, για παράδειγμα τα αυτοκίνητα έχουν παρόμοιο σχήμα, οι πεζοί έχουν αναλογία απεικόνισης (aspect ratio) 0.41. Με αντίστοιχη προσέγγιση, υπάρχει ομαδοποίηση των αντικείμενων σε clusters κατά την εκπαίδευση ανάλογα με τις ανάγκες της εφαρμογής καθώς παρατηρούνται πιο πιθανά παρατηρούμενα αντικείμενα. Όσον αφορά την αναγνώριση αντικειμένων διαφορετικού μεγέθους, σε αντίθεση με την μέθοδο που χρησιμοποιεί ο SSD αλγόριθμος, ο οποίος εντοπίζει τα διαφορετικού μεγέθους αντικείμενα από διαφορετικά layers του πίνακα χαρακτηριστικών, ο YOLO ακολουθεί μια διαδικασία που λέγεται passthrough, η οποία κάνει reshape σε κάποιο layer, το εφαρμόζει στο αρχικό, και μέσα από αυτό που προέκυψε κάνει την πρόβλεψη.



Σχήμα 13: Χαρακτηριστικά (features) Fine-Grained στον YOLOv2

Η αρχιτεκτονική που χρησιμοποιήθηκε ήταν η Darknet-19, η οποία αποτελείται από 19 επίπεδα layers συν 11 επίπεδα για αναγνώριση αντικειμένου, επομένως συνολικά 30 επιπέδων. Παρ΄ όλα αυτά συνεχίζει να έχει πρόβλημα στην αναγνώριση μικρών αντικειμένων.

Συγκεντρωτικά, η βελτιστοποίηση της ακρίβειας στον προσδιορισμό των αντικειμένων σε σχέση με τον YOLO, για κάθε μία βελτιστοποίηση που εφαρμόστηκε, φαίνεται στο παρακάτω διάγραμμα.

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

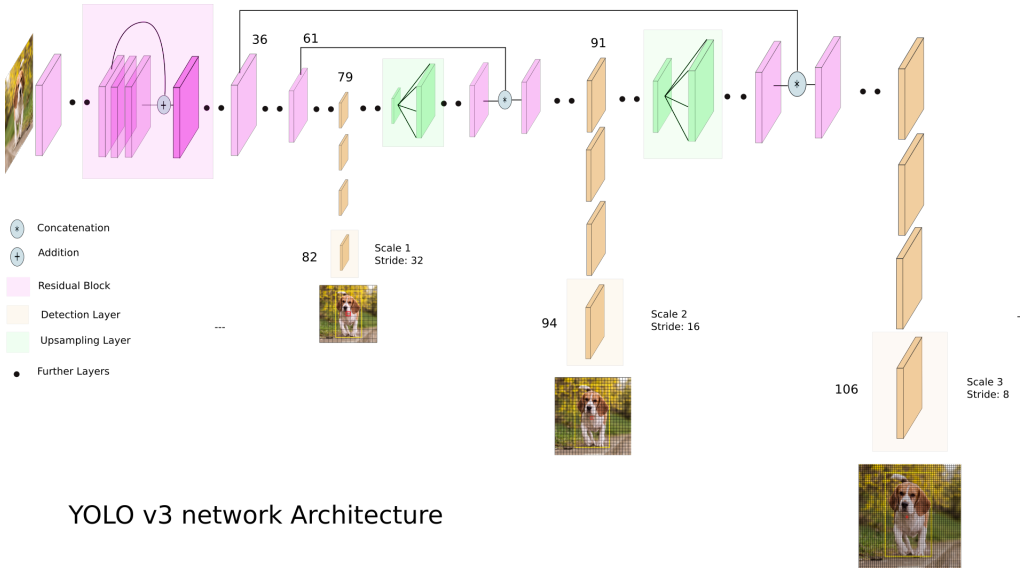
Σχήμα 14: Σύγκριση ακρίβειας σε εκδοχές τουYOLO

Τέλος η μεγάλη βελτίωση στο να μπορεί ο αλγόριθμος να κατηγοριοποιεί 9000 διαφορετικές κλάσεις αντικειμένων, επετεύχθη χρησιμοποιώντας ιεραρχική κατηγοριοποίηση με 9418 κόμβους του WorldTree, και συνδυάζοντας δείγματα από τις βάσεις δεδομένων και εικόνων (datasets) COCO και ImageNet [8]

2.6.3 YOLOv3

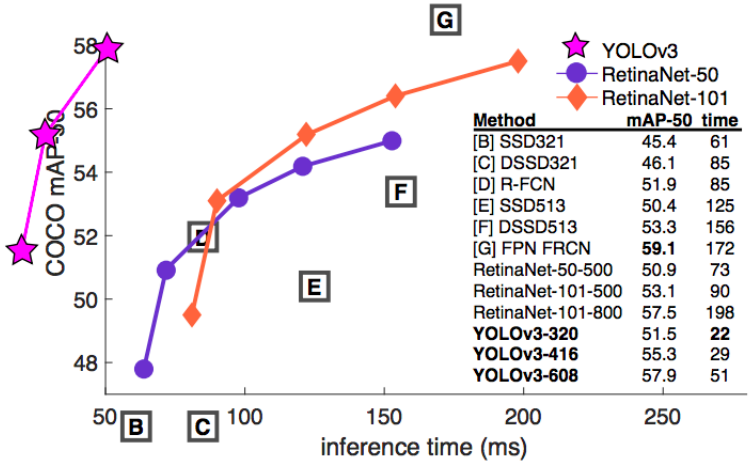
Η τρίτη έκδοση του αλγόριθμου YOLO προσπαθεί να βελτιώσει ακόμη περισσότερο τα προβλήματα της ακρίβειας στην αναγνώριση που συνέχιζαν να υπάρχουν στην προηγούμενη έκδοση του, χωρίς όμως να μειωθεί σημαντικά η ταχύτητα υλοποίησης του.

Πρώτη σημαντική διαφοροποίηση είναι ότι χρησιμοποιείται μία τροποποιημένη αρχιτεκτονική CNN Darknet με 53 layers, και άλλα 53 πάνω στα προηγούμενα για αναγνώριση αντικειμένου, δίνοντας συνολικά 106 επίπεδα δικτύου. Το πιο σημαντικό χαρακτηριστικό αυτού του δικτύου είναι η αναγνώριση σε 3 διαφορετικές κλίμακες scales, το οποίο επιτυγχάνεται εφαρμόζοντας έναν 1x1 kernel σε έναν χάρτη χαρακτηριστικών feature map τριών διαφορετικών μεγεθών, σε 3 διαφορετικά σημεία του δικτύου.



Σχήμα 15: Αρχιτεκτονική δικτύου του YOLOv3

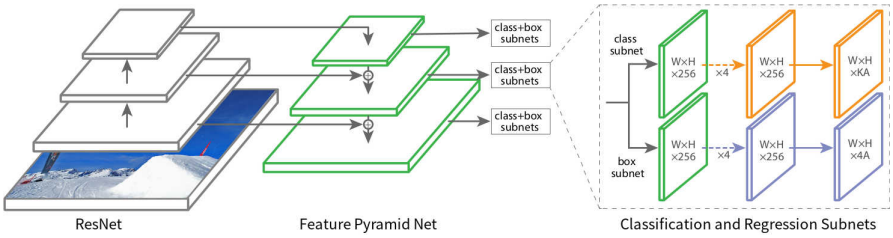
Για να βελτιωθεί η αναγνώριση μικρών αντικειμένων τα αυξημένης δειγματοληψίας επίπεδα συνενώθηκαν με τα προηγούμενα. Επίσης χρησιμοποιούνται 9 anchor boxes, 3 για κάθε κλίμακα (μικρή, μεσαία, μεγάλη). Επίσης προβλέπει 10.647 bounding boxes, πολύ περισσότερα σε σχέση με την προηγούμενη έκδοση, που υπολόγιζε μόλις 845. [9]



Σχήμα 16: Σύγκριση YOLOv3 Vs RetinaNet σε Coco 50 Benchmark

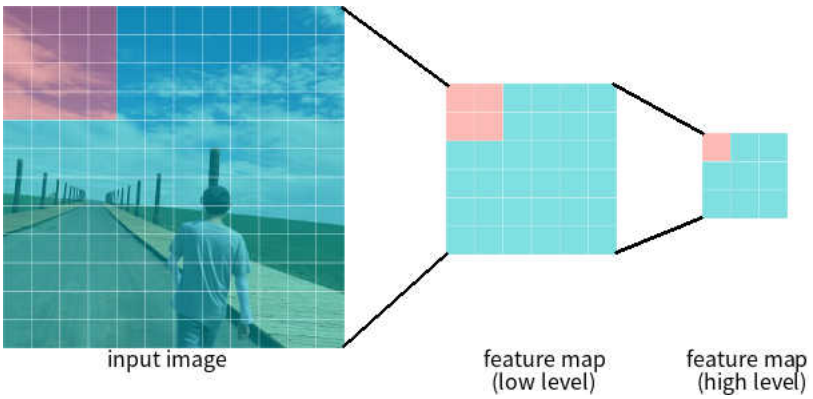
2.7 RetinaNet

Η τεχνολογία αναγνώρισης αντικειμένων RetinaNet είναι ένα δίκτυο το οποίο περιλαμβάνει 3 υποδίκτυα μέσα του. Το Feature Pyramid Net αποτελεί την ραχοκοκκαλιά του δικτύου, και είναι υπεύθυνο για τον νευρωνικό υπολογισμό των πινάκων χαρακτηριστικών feature maps όλης της εικόνας. Υπάρχει ένα υποδίκτυο υπεύθυνο για στην κατηγοριοποίηση σε κλάσεις των εξόδων του προηγούμενου δικτύου, και ένα για το προσδιορισμό των κουτιών (bounding boxes).



Σχήμα 17: Αρχιτεκτονική του RetinaNet

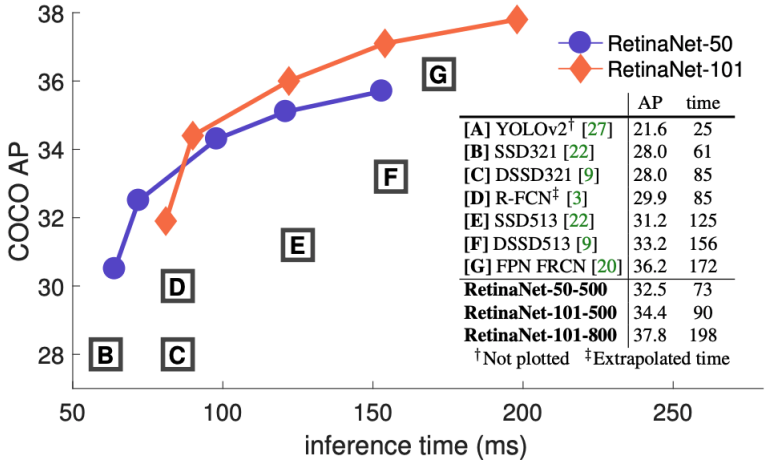
Βασική διαφοροποίηση στην προσέγγιση προσδιορισμού των feature maps σε σχέση με προϋπάρχουσες τεχνολογίες, οι οποίες είτε χρησιμοποιούν το τελευταίο επίπεδο layer του feature map για το prediction, είτε χρησιμοποιούν όλα τα επίπεδα για το prediction, όπως ο SSD, εδώ ο FFP προτείνει feature maps, σε πολλά επίπεδα, από τα οποία τα υψηλά βοηθούν στον εντοπισμό μεγάλων αντικειμένων, και τα χαμηλά σε μικρά αντικείμενα.



Σχήμα 18: Πίνακες χαρακτηριστικών στον retinanet

Το υποδίκτυο για την κατηγοριοποίηση αποτελείται από 4 3x3 convolutional layers, με 256 φίλτρα ακολουθούμενα απο RELU ενεργοποιητές, και μετά από ένα ακόμη ίδιο layer με KxA φίλτρα.Σχεδόν ίδιο είναι και το regression υποδίκτυο.

Ο αλγόριθμος αυτός κατάφερε να είναι πάρα πολύ αποδοτικός, ειδικά για μικρά αντικείμενα στην εικόνα, όπου άλλοι αλγόριθμοι παρουσίαζαν προβληματική απόδοση, όπως ο YOLO, και κατάφερε να το πετύχει αυτό χωρίς να θυσιάσει σημαντικό μέρος από την ταχύτητα. [10]



Σχήμα 19: Σύγκριση του RetinaNet με άλλες τεχνολογίες

Σημαντική καινοτομία του αλγόριθμου, ήταν η νέα προσέγγιση στην διαδικασία της εκπαίδευσης, όπου οι ενός επιπέδου detectors υστερούσαν σε χρόνο και αποδοτικότητα εκπαίδευσης σε με αυτούς των 2 σταδίων. Προτάθηκε η focal loss τεχνική η οποία επανασχηματίζει την cross entropy loss έτσι ώστε εύκολα σημεία της εικόνας, όπως είναι το φόντο όπου η σιγουριά του αλγόριθμου είναι αρκετά ισχυρή ότι δεν περιέχει αντικείμενο ενδιαφέροντος, να συνεισφέρουν λιγότερο στην συνολική απώλεια, και η εκπαίδευση να επικεντρώνεται στα σημαντικά σημεία της εικόνας.

2.8 Σύγκριση των μεθόδων object detection

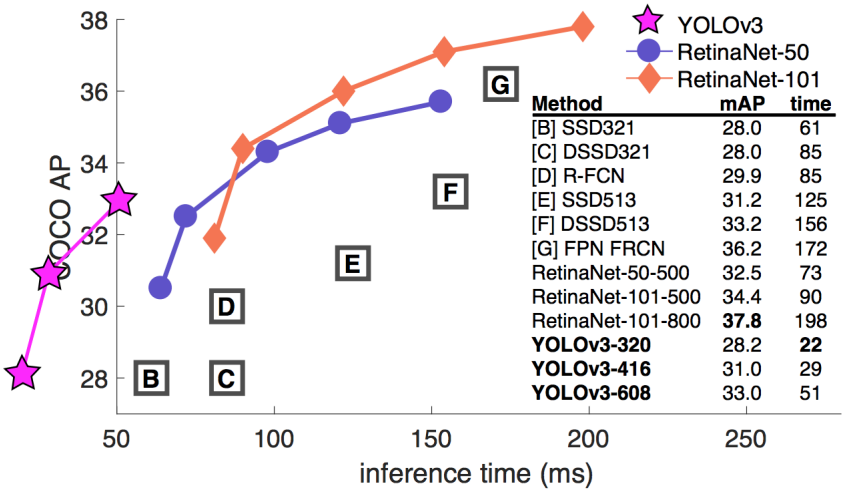
Μία πρώτη προσέγγιση στην σύγκριση των βασικών αλγορίθμων αναγνώρισης αντικειμένων σε εικόνες που περιγράφηκαν στις προηγούμενες ενότητες είναι οι συγκριτικοί πίνακες που παρουσιάζονται στα ίδια τα papers των δημιουργών. Εκεί έχουν γίνει μετρήσεις στην απόδοση του εκάστοτε αλγόριθμου, και σύγκριση με τις τιμές που είχαν παρουσιαστεί για τις υπόλοιπες τεχνολογίες. Παρακάτω παρουσιάζονται κάποιοι από αυτούς τους πίνακες. Επιλέχθηκαν πίνακες από πιο σύγχρονες τεχνολογίες καθώς περιλαμβάνουν περισσότερες συγκρίσεις με τους προηγούμενους αλγόριθμους.

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

Σχήμα 20: Σύγκριση τεχνολογιών στοYOLOv2 paper με PascalVOC 2012 τεστ

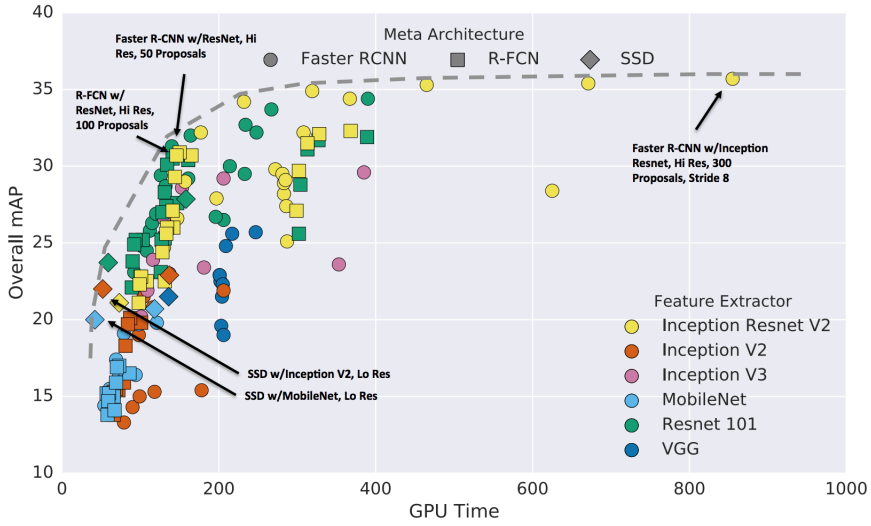
Ο πίνακας αυτός περιλαμβάνει τιμές AP(Average Precision) σε γενικές αναζητήσεις, αλλά και στοχευμένες ανά κατηγορία αντικειμένου. Είναι φανερό πως οι αλγόριθμοι Yolov2, ResNet, SSD έχουν παραπλήσια απόδοση, αρκετά καλύτερα από τους YOLO, Fast R-CNN, Faster R-CNN. Σε διαφορετικά αντικείμενα παρατηρούμε ότι υπάρχουν διακυμάνσεις στην απόδοση, χωρίς βέβαια να αλλάζει η ομαδοποίηση που παρατηρήθηκε στον γενικό μέσο όρο. [8]

Στο παρακάτω διάγραμμα που παρουσιάστηκε στον αλγόριθμο YOLOv3 βλέπουμε σχετικά χαμηλές αποδόσεις σε mAP (mean Average Precision) για όλα τα μοντέλα πάνω σε COCO dataset. Ενδιαφέρον παρουσιάζει η σύγκριση που γίνεται στην χρονική απόκριση της εκτέλεσης τους. Σαν συμπέρασμα ως προς τις βέλτιστες τεχνολογίες, παρατηρείται σαφής ανωτερότητα του RetinaNet αλλα με τεράστια χρονική υστέρηση σε σχέση με τις τεχνολογίες YOLO. Οι SSD, FPN FRCN, R-FCN έχουν παραπλήσια απόδοση με τους YOLO, αλλά με περίπου 4 φορές μεγαλύτερη διάρκεια εκτέλεσης. [9]



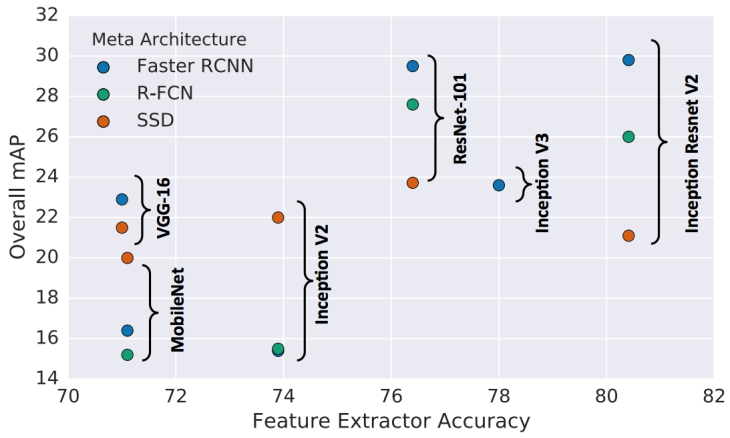
Σχήμα 21: Σύγκριση τεχνολογιών στοYOLOv3 paper με COCO τεστ

Γίνεται φανερό πως δεν μπορεί να υπάρξει εύκολα απάντηση στο ερώτημα ποιος detector είναι πιο γρήγορος, και ίσως να μην έχει και νόημα μία τέτοια ερώτηση. Η κρίσιμη ερώτηση είναι ποια διαμόρφωση συστήματος μας δίνει την καλύτερη ισορροπία ανάμεσα σε αποδοτικότητα και ταχύτητα εκτέλεσης για την εκάστοτε εφαρμογή που χρησιμοποιείται. Μία έρευνα τη Google Research κάνει μια τέτοιας προσέγγισης σύγκριση ανάμεσα στους SSD, R-CNN και R-FCN detectors. Παρακάτω παρουσιάζονται κάποια χαρακτηριστικά διαγράμματα της έρευνας αυτής.



Σχήμα 22: Σύγκριση ταχύτητας με ακρίβεια σε SSD, R-CNN and R-FCN detectors

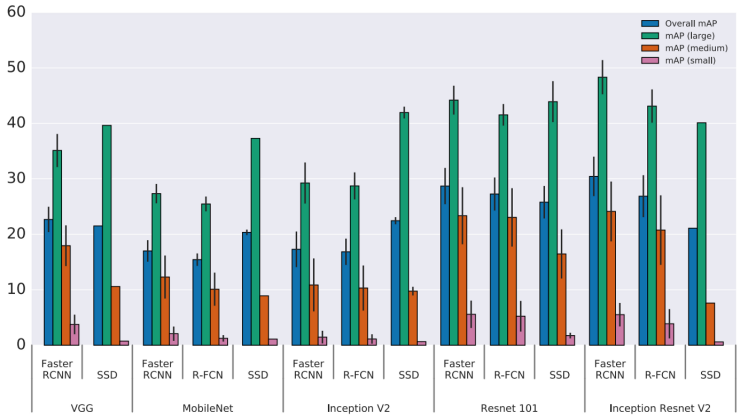
Σε γενικές γραμμές ο Faster R-CNN έχει καλύτερη ακρίβεια, και οι SSD, R-FCN είναι πιο γρήγοροι.



Σχήμα 23: Feature Accuracy in SSD, R-CNN and R-FCN detectors

Μετά την έρευνα που έγινε στο πως η εξαγωγή του πίνακα χαρακτηριστικών της εικόνας feauture extractor επηρεάζει την ακρίβεια ανίχνευσης. Οι Faster

R-CNN και R-FCN αξιοποιούν καλύτερα τον feature extractor. Παρακάτω παρουσιάζονται 2 τελευταίοι ενδιαφέροντες πίνακες που δίνουν αρκετά στοιχεία για τους αλγόριθμους που αναλύονται στην έρευνα. [11]



Σχήμα 24: Σύγκριση για διαφορετικού μεγέθους αντικείμενα σε SSD, R-CNN and R-FCN detectors

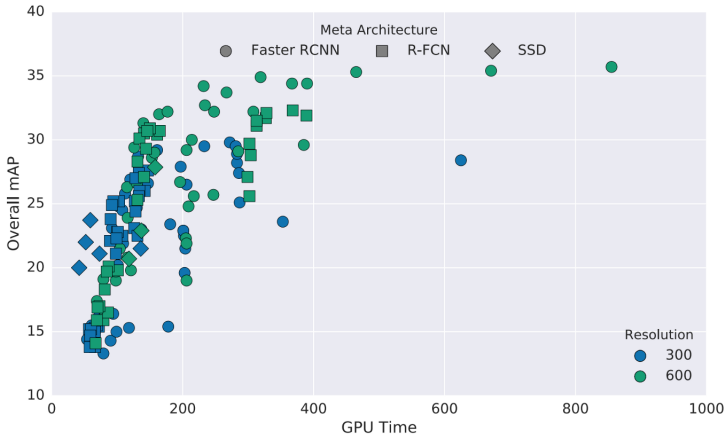


Figure 5: Effect of image resolution.

Σχήμα 25: Σύγκριση για διαφορετικής ανάλυσης εικόνες σε SSD, R-CNN and R-FCN detectors

3 Σχετικές εφαρμογές

Η ανάπτυξη πολυμεσικών εφαρμογών τα τελευταία χρόνια, όπως και η βελτιστοποίηση του hardware υπολογιστικών συστημάτων είναι ραγδαία. Η χρήση και οι δυνατότητες που προσφέρουν τα κινητά τηλέφωνα όσον αφορά την χρήση της φωτογραφικής μηχανής, έχουν δημιουργήσει μεγάλες δυνατότητες ανάπτυξης εφαρμογών τόσο επεξεργασίας εικόνας και βίντεο, όσο και αλγορίθμων εντοπισμού προσώπου και αντικειμένων. Οι χρήσεις των αποτελεσμάτων που εξάγονται μέσω τέτοιων τεχνικών είναι πολλές, και παρακάτω θα αναφερθούν κάποιες βασικές που αναπτύχθηκαν από μεγάλες τεχνολογικές εταιρίες.

3.1 Object detection από την Amazon (Sagemaker/Rekognition)

Η Amazon, στο πλαίσιο της πλατφόρμας AWS (Amazon Web Services) που έχει δημιουργήσει για να παρέχει σε πελάτες και developers εργαλεία cloud computing, έχει δημιουργήσει έναν αλγόριθμο αναγνώρισης και κατηγοριοποίησης εικόνων. Ονομάζεται Sagemaker Δέχεται σαν είσοδο μία εικόνα, εντοπίζει τα αντικείμενα μέσα σε αυτή, προσδιορίζει την κατηγορία του κάθε αντικειμένου, και το bounding box που το περικλείει. Χρησιμοποιεί την SSD πλατφόρμα και υποστηρίζει δύο νευρωνικά δίκτυα, το VGG και το ResNet. Η χρήση γίνεται σε λειτουργικό και εύχρηστο γραφικό περιβάλλον μέσω ενός browser. Δίνει επίσης την δυνατότητα στους χρήστες να εκπαιδεύσουν το μοντέλο με δικά τους dataset. [12]

Έχει αναπτύξει επίσης μία cloud based εφαρμογή όρασης υπολογιστών computer vision, η οποία δίνει την δυνατότητα στον χρήστη να αξιοποιήσει υπηρεσίες αναγνώρισης προσώπων,αναγνώριση celebrities, εντοπισμό της διαδρομής που ακολούθησε κάποιος άνθρωπος, αναγνώριση κειμένου και εντοπισμό ακατάλληλου οπτικού περιεχομένου. Όσον αφορά το αναγνώριση προσώπων, δίνει την δυνατότητα στους χρήστες να μπορούν να εκπαιδεύσουν το μοντέλο σύμφωνα με τις δικές τους ανάγκες. [13]

3.2 Object detection από την Facebook (DETR/Detector)

Μια ακόμη μεγάλη εταιρία που έχει ασχοληθεί με τεχνολογίες object detection, είναι η Facebook. Κάτω από την γενική ανάπτυξη Artificial Inteligent (AI) εφαρμογών, έχει δημιουργήσει τον Detr, έναν end-to-end object detector. Χρησιμοποιώντας την βιβλιοθήκη PyTorch για machine learning συνδυάζει την τεχνολογία Faster R-CNN με το μοντέλο δικτύου ResNet-50 και πετυχαίνει επιδόσεις αναγνώρισης αντικειμένων 42 AP σε COCO dataset. [14]

Έχει δημιουργήσει επίσης ένα artificial inteligence σύστημα που συνδυάζει και ενσωματώνει διάφορες τεχνολογίες object detection (Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN) καθώς και διάφορες αρχιτεκτονικές νευρονικών δικτύων (ResNeXt(50,101,152), (ResNet(50,101,152), Feature Pyramid Networks, VGG16. Έτσι δίνει την δυνατότητα σε ευέλικτη και γρήγορη δημιουργία ερευνητικών εφαρμογών. [15]

4 Παρουσίαση Λύσης

4.1 Βιβλιοθήκες

Κατά την διάρκεια του πειραματισμού με διάφορες πλατφόρμες αναγνώρισης, φάνηκε πόσο βασικό είναι να υπάρχουν διαθέσιμες οι κατάλληλες βιβλιοθήκες που χρησιμοποιούνται κατά την εκπαίδευση και την αναγνώριση. Μικρές αλλαγές στην έκδοση κάποιας βιβλιοθήκης μπορούν να οδηγήσουν σε αδυναμία εκτέλεσης διάφορων συστημάτων της πλατφόρμας. Για τους παραπάνω λόγους κρίνω σκόπιμο να αναφέρω ακριβώς πώς στήθηκε ο υπολογιστής ώστε να επιτευχθεί ο στόχος.

Η βάση της πλατφόρμας του retinanet βασίζεται πάνω στο openCV και στην στο Keras. Το OpenCV (Computer Vision) είναι βασικό εργαλείο επεξεργασίας εικόνας και βίντεο, δίνοντας τεράστιες δυνατότητες παραμετροποίησης με εύκολο τρόπο στον προγραμματιστή, όπως επίσης και δυνατότητα συνεργασίας των project με το CUDA API της Nvidia το οποίο κάνει εφικτή την χρήση υπολογιστικών πόρων από την κάρτα γραφικών GPU, αυξάνοντας κατά πολύ την ταχύτητα υλοποίησης απαιτητικών εφαρμογών. Το Keras είναι ένα deep learning API το οποίο δίνει την δυνατότητα να δημιουργηθεί ένα νευρωνικό δίκτυο, με όλα τα υπομέρη του (convolutional layers, pooling, filters), καθώς και τρόπους ενεργοποίησης του, αλλά και εργαλείων διαχείρισης. Και οι 2 αυτές βιβλιοθήκες είναι γραμμένες και για γλώσσα προγραμματισμού Python, επομένως κρίθηκε αναγκαία η οργάνωση των βιβλιοθηκών με το εργαλείο διαχείρισης πακέτων για R και Python που ονομάζεται Anaconda

Το Anaconda είναι μία πλατφόρμα μέσω της οποίας είναι εύκολο να δομηθεί και να διαχειριστεί το σύνολο των πακέτων που χρειάζονται σε ένα project. Είναι πολύ χρήσιμο να δημιουργούνται περιβάλλοντα βιβλιοθηκών ξεχωριστά για τις ανάγκες κάθε project, καθώς οι εκδόσεις και τα updates των βιβλιοθηκών μπορεί να οδηγήσουν σε σφάλματα στην εκτέλεση του κώδικα. Για τον λόγο αυτό δημιουργήσαμε ένα anaconda environment, που εκτός των βασικών libraries που περνάει αυτόματα σε όλα τα περιβάλλοντα που δημιουργούνται για την Python 3.6, προσθέσαμε τα παρακάτω:

- numpy 1.18.4
- tensorflow 2.1.0

- matplotlib 3.2.1
- tensorflow-gpu 2.1.0
- pillow 7.1.2
- scipy 1.3.1

Η εγκατάσταση των παραπάνω πακέτων οδήγησε σε αυτόματη ενεργοποίηση και άλλων libraries απαραίτητων στην συνολική λειτουργία, από τα οποία θα αναφέρω τα πιο σημαντικά για την βελτιστοποίηση της απόδοσης του retinanet τόσο κατά την εκτέλεση αλγορίθμων αναγνώρισης, όσο και εκπαίδευσης του μοντέλου μας.

- cudnn 7.6.5
- cudatoolkits 10.1.243
- keras-applications 1.0.8

4.2 Εκπαίδευση Μοντέλου

4.2.1 Μορφή Δεδομένων για Εκπαίδευση

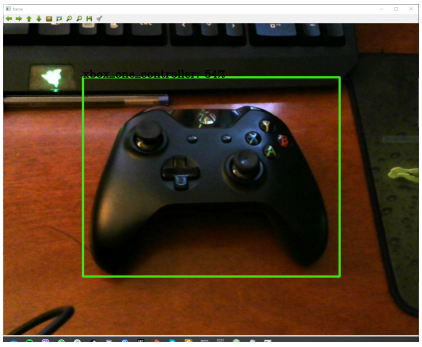
Η πλατφόρμα retinanet έχει δημιουργηθεί όχι μόνο για να εκτελεί object detection σε εικόνες χρησιμοποιώντας υπάρχοντα μοντέλα αναγνώρισης αντικειμένων, αλλά και για να μπορεί να εκπαιδευτεί με καινούργια dataset. Τα δεδομένα αυτά πρέπει να τηρούν συγκεκριμένους κανόνες παρουσίασης των πληροφοριών που φέρουν. Υπάρχουν 2 βασική τρόποι δόμησης των dataset, και ο retinanet μας δίνει την δυνατότητα χρήσης και των 2. Αυτά είναι ο COCO(Common Objects in Context) και ο Pascal VOC (Pascal Visual Object Classes).

Επιλέξαμε να χρησιμοποιήσουμε το Pascal VOC τρόπο δόμησης του dataset, καθώς οι πληροφορίες των εικόνων εκπαίδευσης ήταν πιο εύκολο να μετατραπούν σε αυτή την μορφή. Η μόνη διαφορά ανάμεσα στα 2 πρότυπα έχει να κάνει με την μορφή του αρχείου που περιέχει τις πληροφορίες για τα αντικείμενα που αναγνωρίζουμε ότι βρίσκονται στην κάθε εικόνα. Στην περίπτωση του COCO έχουμε ένα αρχείο JSON που περιέχει τις πληροφορίες για τα bounding

boxes όλων των φωτογραφιών (Annotations), και διαφέρει στον τρόπο που ορίζονται τα bounding boxes (x-top left, y-top left, width, height).[16] Στο PascalVOC οι πληροφορίες για τις εικόνες υπάρχουν σε μορφή αρχείου xml, και αντιστοιχεί σε κάθε εικόνα και ένα αρχείο. Επίσης η δομή των bounding boxes είναι (x-top left, y-top left,x-bottom right, y-bottom right) [17].

4.2.2 Προεργασία που οδήγησε στην επιλογή του dataset

Η εκπαίδευση από το μηδέν στην πλατφόρμα του retinanet και την δημιουργία ενός νέου μοντέλου για την αναγνώριση ενός αντικειμένου είναι μια διαδικασία που περιλαμβάνει πολύωρη προετοιμασία των δεδομένων που τροφοδοτούν το προς εκπαίδευση νευρωνικό δίκτυο, καθώς και πολύωρη τροφοδότηση του δικτύου. Μετά από πειραματισμούς στην δημιουργία κατάλληλου dataset για την αναγνώριση ενός τηλεχειριστηρίου κονσόλας XBOX χρησιμοποιώντας 300 εικόνες με χειροκίνητο προσδιορισμό των bounding boxes χρησιμοποιώντας το εργαλείο label image tool διαπιστώθηκε ότι χρειάζονται πάνω από 6 ώρες για την προετοιμασία των xml αρχείων για τα annotation files, και η απόδοση του μοντέλου που δημιουργήθηκε ήταν ελάχιστα ικανοποιητική, καθώς υπό γωνίες η σιγουριά του αλγόριθμου για την ύπαρξη του αντικειμένου έπεφτε κάτω από το 10% και πολλές φορές απλώς αλλάζοντας την γωνία θέασης δεν το έβλεπε καθόλου. Σε ευθεία προβολή η σιγουριά έφτανε το 60%. Παρακάτω φαίνονται 2 screenshot κατά την υλοποίηση αναγνώρισης.



(i) 0 degrees angle - 54% precision

(ii) Another point of view - 18% precision

Σχήμα 26: Υλοποίηση εκπαίδευσης για 1 αντικείμενο με 300 φωτογραφίες

Έγινε φανερό πως για την εκπαίδευση ενός νευρωνικού δικτύου χρειάζονται πολλές περισσότερες εικόνες, διαφορετικών οπτικών γωνιών, μεγέθους αντικειμένου και χρωμάτων. Μετά από έρευνα σε δημοσιεύσεις αντίστοιχων προβληματισμών στο διαδίκτυο, διαπιστώθηκε ότι για να γίνει η εκπαίδευση με ικανοποιητικό τρόπο χρειάζονται 2000-3000 εικόνες. Επομένως για εκπαίδευση ενός δικτύου που θα μπορεί να αναγνωρίζει τα 10 ψηφία του δεκαδικού συστήματος, χρειάζονται 20000-30000 φωτογραφίες συνολικά, κατάλληλης ποιότητας και οπτικής προσέγγισης (χρώματα, φόντο, μέγεθος, οπτική γωνία) με το πρόβλημα που θα επιλύουν. στην περίπτωσή μας την αναγνώριση κωδικού αθλητών δρόμου. Η χειροκίνητη εύρεση και επεξεργασία για τον ορισμό των bounding boxes ώστε να εξαχθεί το απόλυτο αποτέλεσμα δεν είναι εφικτό να αποτελέσει μέρος μίας εργασίας 1 ατόμου. Επιλέχθηκε η χρήση έτοιμου dataset, που μπορεί να δείξει της δυνατότητες της τεχνολογίας αυτής πάνω στο συγκεκριμένο πρόβλημα.

4.2.3 SVHN dataset

Μετά από έρευνα πάνω σε έτοιμα dataset για την εκπαίδευση νευρωνικών δικτύων και δημιουργία μοντέλων αναγνώρισης, διαπιστώθηκε η ύπαρξη 2 συλλογών εικόνων που περιέχουν ψηφία αριθμών. Το πρώτο dataset, που δημιουργήθηκε για τον σκοπό αυτό είναι το The MNIST database of handwritten digits, το οποίο περιέχει 60000 εικόνες διαφορετικών χειρόγραφων δεκαδικών ψηφίων για εκπαίδευση, και 10000 για τεστάρισμα. [18]

Βασισμένο στην λογική και στην νοοτροπία του MNIST είναι το dataset που επιλέξαμε να χρησιμοποιήσουμε. Ονομάζεται SVHN και είναι βασισμένο σε εικόνες από αριθμούς σπιτιών απο το Google Street View. Είναι διαθέσιμα συνολικά 3 πακέτα εικόνων, 73257 εικόνες για εκπαίδευση, 26032 για τεστάρισμα και άλλες 531131 λιγότερο δύσκολες εικόνες για επιπλέον χρήση. Οι εικόνες είναι σε μορφή αρχείου png. Για τον προσδιορισμό των bounding boxes κάθε set εικόνων συνοδεύεται από ένα αρχείο digitStruct.mat που περιέχει όσα στοιχεία είναι και το σύνολο των εικόνων. Κάθε στοιχείο περιέχει το όνομα της εικόνας που του αντιστοιχεί και τα στοιχεία των bounding boxes (θέση,μέγεθος και ετικέτα αριθμού) που περιέχονται στην αντίστοιχη εικόνα. [19]

Σχήμα 27: Παραδείγματα εικόνων από SVHN dataset

4.2.4 Επεξεργασία dataset για εκπαίδευση σε PascalVoc

Ο αλγόριθμος εκπαίδευσης του keras-retinanet, αναλόγως το πρότυπο dataset που θα επιλεχθεί να χρησιμοποιηθεί, ορίζει κάποιες προδιαγραφές για τις εικόνες και τα annotation αρχεία που περιέχουν τα δεδομένα για τα bounding boxes. Επίσης ορίζει και συγκεκριμένο τρόπο δόμησης των φακέλων που περιέχουν τα παραπάνω δεδομένα. Παρακάτω παρουσιάζονται οι αλλαγές που έπρεπε να γίνουν στο SVHN dataset που επιλέξαμε να χρησιμοποιήσουμε ώστε να έρθει σε κατάλληλη μορφή για το PascalVOC πρότυπο.

Χρησιμοποιούμε 33402 εικόνες, ώστε να έχουμε 3000 εικόνες τουλάχιστον που να περιέχουν κάθε διαφορετικό στοιχείο. Η εκπαίδευση σε PascalVOC στον retinanet επιβάλει οι εικόνες να είναι στο μορφή αρχείου jpg. Επομένως έπρεπε να γίνει μετατροπή από png σε jpg κρατώντας το όνομα του αρχείου ίδιο.

```
from PIL import Image
import glob
import os

read_path = "./VOCdevkit/Images/*.png"
write_path = "./VOCdevkit/JPEGImages"
for file in glob.glob(read_path):    # run through the entire
    file
     im1 = Image.open(file)
     basename = os.path.basename(file)    # keep the name of
    the current file
```

```
    splitext = os.path.splitext(basename)[0] + ".jpg"
    outfile = "%s/%s" % (write_path, splitext)
    im1.save(outfile)
```

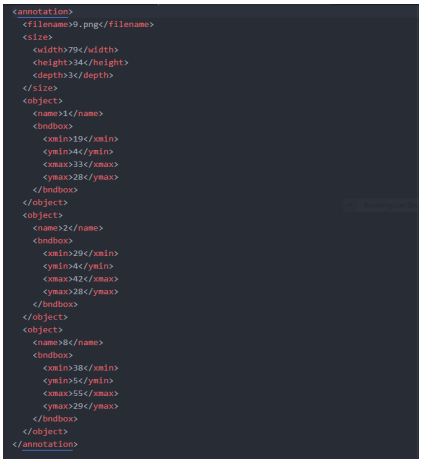
Listing 1: Convert png to jpg

Ένα άλλο βασικό στοιχείο που πρέπει να τηρεί αυστηρά τις προδιαγραφές του αλγόριθμου εκπαίδευσης είναι το format των annotation files. Για κάθε εικόνα, σε έναν άλλο φάκελο έχουμε ένα xml αρχείο στο οποίο περιεγράφονται τα bounding boxes. Το αρχείο digitStruct.mat που περιγράφηκε παραπάνω δόθηκε έτοιμο σε μορφή xml, όμως σαν όνομα αρχείου που αντιστοιχεί είχε το αρχικό .png, ενώ εμείς τα έχουμε αλλάξει σε .jpg. Επίσης έλειπαν κάποιες πληροφορίες για τα bounding boxes που έπρεπε να συμπληρωθούν, έστω με μηδενικές τιμές.

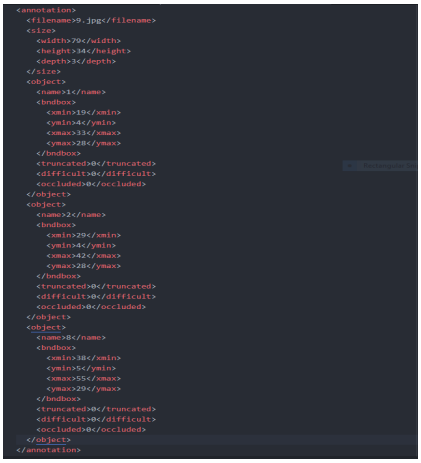
```
import xml.etree.ElementTree as ET
import glob
import os

read_path = "./Annotations/*.xml"
write_path = "./VOCdevkit/Annotations2"
for file in glob.glob(read_path):
    tree = ET.parse(file)
    root = tree.getroot()

    # get the png filename and change it to jpg
    old_name = root.find("filename").text
    new_name = old_name.split(".",1)[0] + ".jpg"
    root.find("filename").text = new_name

    # create elements that are missing for Pascal VOC
    annotation format
    for object in root.findall("object"):
        truncated = ET.SubElement(object, "truncated")
        truncated.text = "0"
        difficult = ET.SubElement(object, "difficult")
        difficult.text = "0"
        occluded = ET.SubElement(object, "occluded")
        occluded.text = "0"

    basename = os.path.basename(file)
    outfile = "%s/%s" % (write_path,basename)
    tree.write(outfile)
```

Listing 2: Create xml annotation files

Έτσι προκύπτουν οι αλλαγές που φαίνονται παρακάτω. Μόνο αν τα annotation files είναι σε αυτή την μορφή μπορεί να τρέξει ο αλγόριθμος εκπαίδευσης.



(i) Xml annotation as given

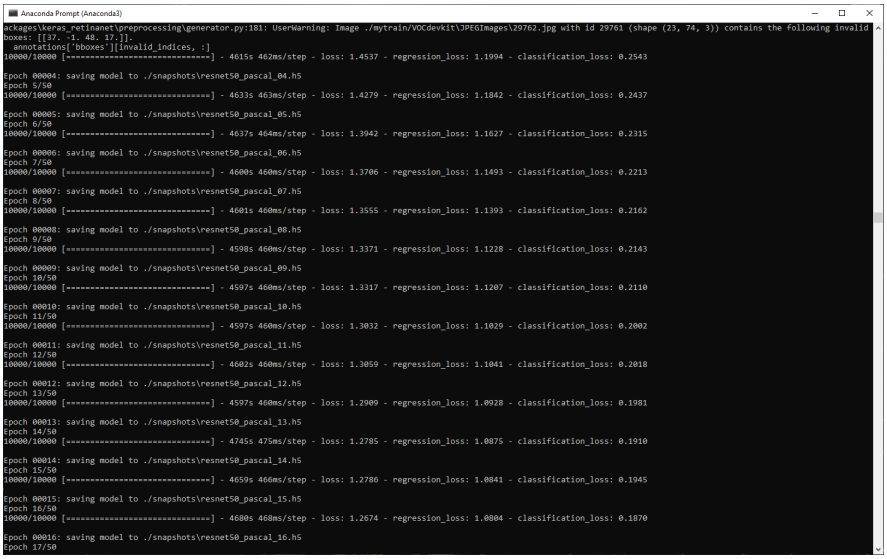


(ii) Xml annotation modified

Σχήμα 28: Τροποποιήσεις στα annotation xml files

4.2.5 Εκπαίδευση Μοντέλου

Έχοντας έτοιμα όλα τα δεδομένα που χρειάζονται για την εκπαίδευση, οι δημιουργοί του retinanet σου δίνουν την δυνατότητα εκπαίδευσης με μία απλή εντολή χρησιμοποιώντας τις προκαθορισμένες επιλογές επαναλήψεων, βημάτων κλπ. Οι 3 σημαντικότεροι παράμετροι που θα αναφερθούν εδώ, είναι το batch size, δηλαδή το πόσα δεδομένα προς εκπαίδευση τρέχουν στο νευρωνικό δίκτυο πριν ελεγχθεί η απόδοση στα ίδια δεδομένα, και τροποποιηθεί το δίκτυο για βελτίωση. Είναι το epochs, που είναι ο αριθμός το επαναλήψεων πάνω σε ολόκληρο το dataset. Τέλος έχουμε και το steps, που είναι ο αριθμός των επαναλήψεων που γίνονται μέσα σε ένα epoch. Η μόνη διαφοροποίηση από τις αρχικές επιλογές που αλλάξαμε ήταν το batch size το οποίο από την default τιμή του που είναι 1, το κάναμε 4. Τα steps ήταν 10000 και τα epochs 50. Παρακάτω παρουσιάζεται ένα screenshot από την console των windows που έτρεξε η όλο το project.



Σχήμα 29: Εκπαίδευση μοντέλου σε Windows 10 console

Οι προδιαγραφές του υπολογιστή που έτρεξε ο αλγόριθμος εκπαίδευσης ήταν ένας επεξεργαστης Inter i5-3330(3GHz), 16Gb Ram και μία κάρτα γραφικών Nvidia GTX 1060 (6BG Memory. Βασικό ρόλο στην επίτευξη της εκπαίδευσης όπως αναφέρθηκε και προηγουμένως παίζει η ισχύς της κάρτας γραφικών. Το σύστημά μας οριακά κατάφερε να αντεπεξέλθει στις προδιαγραφές που ορίσαμε. Χρειάστηκαν περίπου 65 ώρες και η κάρτα γραφικών να δουλεύει στο 90% για να ολοκληρωθεί η διαδικασία. Σε δοκιμή που έγινε εκτελώντας τον αλγόριθμο εκπαίδευσης με την default τιμή ένα για το batch size, η χρήση της κάρτας γραφικών μειώθηκε στο 40%. Το μέγιστο δυνατό batch size κατά την εκπαίδευση επηρεάζει θετικά την ποιότητα του μοντέλου που δημιουργείται, και μειώνει το error gradient κατά την διάρκεια της εκπαίδευσης. Γίνεται επομένως φανερό πως με πειραματισμούς σε ισχυρά μηχανήματα μπορούμε να πετύχουμε καλύτερα αποτελέσματα σε λιγότερο χρόνο πάνω ακριβώς στο ίδιο dataset.

4.3 Υλοποίηση digit detection

Οι δημιουργοί του keras-retinanet δίνουν έτοιμο τον κώδικα για βασική λειτουργία αναγνώρισης αντικειμένων σε μία εικόνα. Με κάποιες τροποποιήσεις μπορούμε να επεκτείνουμε τον κώδικα ώστε να διαβάζει όσες εικόνες θέλουμε

και να αποθηκεύει το αποτέλεσμα με ενσωματωμένα τα bounding boxes πάνω στις εικόνες. Παρακάτω φαίνεται πως λειτουργεί ο αλγόριθμος αναγνώρισης δεκαδικών ψηφίων σε διάφορες εικόνες αθλητών, διαφορετικού μεγέθους, ανάλυσης και περιεχομένου.

(i) Ένας αθλητής με κωδικό 4 ψηφίων

(ii) Ένας αθλητής με κωδικό 4 ψηφίων

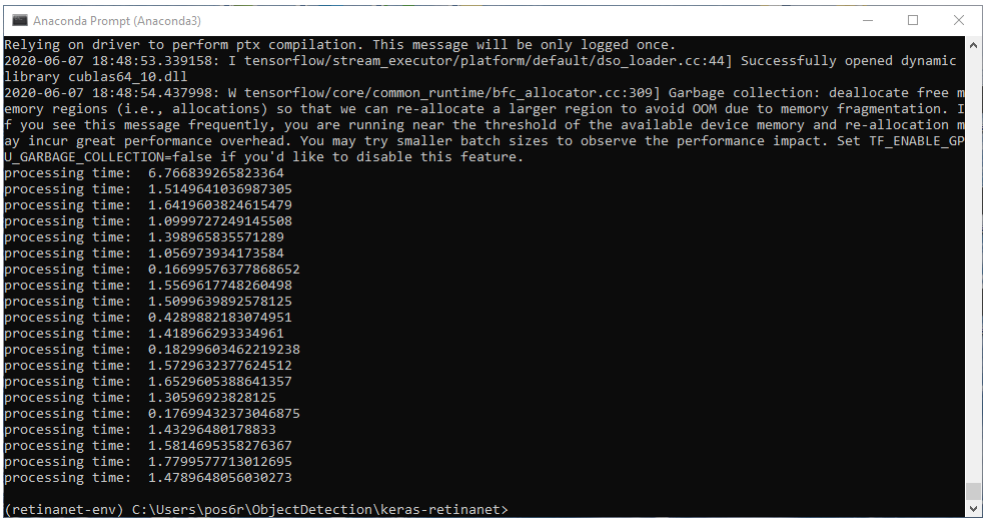
(iii) Αθλητές με κωδικούς 4 ψηφίων

(iv) Ένας αθλητής με κωδικό 3 ψηφίων

Σχήμα 30: Παραδείγματα αναγνώρισης ψηφίων σε μπλούζες αθλητών

Κατά την αναγνώριση των φωτογραφιών ο αλγόριθμος μας επιστρέφει και

τον χρόνο που χρειάζεται η κάθε φωτογραφία. Στην επόμενη εικόνα φαίνονται οι χρόνοι για ένα σύνολο 19 φωτογραφιών. Παρατηρούμε έναν μεγάλο χρόνο επεξεργασίας 6.7 δευτερολέπτων για την πρώτη φωτογραφία, μετά παρατηρούμε χρόνους γύρω στα 1.5 δευτερόλεπτα, συν-πλην 0.2 δευτερόλεπτα, και 3 χρόνους σχεδόν μηδενικής διάρκειας. Ο μεγάλος πρώτος χρόνος μπορεί να εξηγηθεί γιατί προσμετράται και όλη η προετοιμασία του μοντέλου για να αρχίσει η διαδικασία. Οι σχεδόν μηδενικοί χρόνοι αφορούν εικόνες μεγάλης ανάλυσης όπου ο αλγόριθμος δεν εκτελείται και δεν βρίσκεται κανένα αποτέλεσμα.



Σχήμα 31: Χρόνοι επεξεργασίας σε σετ εικόνων σεWindows 10 console

4.4 Αναγνώριση κωδικών αθλητών

Ο αλγόριθμος keras-retinanet που περιγράφηκε παραπάνω αναγνωρίζει ποια δεκαδικά ψηφία βρίσκονται σε μια εικόνα, και προσδιορίζει και την ακριβή θέση τους, δημιουργώντας bounding boxes γύρω τους. Σκοπός μας, εκτός από την εκπαίδευση ενός νευρωνικού δικτύου και την δημιουργία ενός μοντέλου που θα επιτελεί αυτή την διεργασία, είναι και η υλοποίηση ενός αυτόματου συστήματος που δεχόμενο φωτογραφίες ή βίντεο από κάποια αθλητική διοργάνωση αγώνων δρόμου, να εντοπίζει ποιοι αθλητές βρίσκονται εκεί. Επομένως δημιουργήσαμε έναν αλγόριθμο που μέσω γεωμετρικών διεργασιών, και μεθόδων διαχωρισμού χρήσιμων και άχρηστων ψηφίων της εικόνας, να ομαδοποιεί τα ψηφία ανά τον

αριθμό που έχει ο κωδικός των αθλητών, και να παρουσιάζει ποιοι βρίσκονται μέσα σε κάθε εικόνα, ή frame ενός βίντεο.

```
1
2           # create boxes and label lists of items of interest
3           # box has the coordinates, and label the label of the
     detection
4           myboxes.append(box)
5           mylabels.append(label)
6           myscores.append(score)
7
8       #up_left, down_left, up_right, down_right = x1, y1, x2,
    y2
9       x1, y1, x2, y2 = ([] for i in range(4))
10      for i in range(len(myboxes)):
11          x1.append(myboxes[i][0])
12
13      # sort the boxes of digits, from left to right, and the
    same way the labels
14      x1_sorted = sorted(x1)
15      mylabels_sorted = [x for _,x in sorted(zip(x1,mylabels))]
16      myscores_sorted = [x for _,x in sorted(zip(x1,myscores))]
17
18      # if 2 prediction in the same area, we keep the one with
    the higher probability(score)
19      i = 0
20      count = 0   # we use count so when an element is deleted
    we keep the correct size of the lists
21      length = len(x1_sorted)
22      while( i < (length-1-count)):
23          if (x1_sorted[i+1] - x1_sorted[i] < 10):
24              for j in range(len(x1_sorted)-1-i):     # loop
    for as many close elementsto keep the "strongest one"
25                  if (x1_sorted[i+1] - x1_sorted[i] > 10):
    break
26                  if myscores_sorted[i] < myscores_sorted[i+1]:
27                      del myscores_sorted[i]
28                      del mylabels_sorted[i]
29                      del x1_sorted[i]
30                  else:
31                      del myscores_sorted[i+1]
32                      del mylabels_sorted[i+1]
33                      del x1_sorted[i+1]
34                  count +=1
35          i += 1
```

```
36
37      digits = 3
38      # delete random findings that are not paired in groups of
        the size we want
39      # we must change the ...and (count < x) to our needs. x =
        number of digits - 1.
40      mylabels_sorted2 = []
41      count = 0
42      for i in range(len(x1_sorted)-1):
43              if (x1_sorted[i+1] - x1_sorted[i] > 80):
44                  if count == digits - 1:
45                      for j in range(i - count, i + 1):
46                          mylabels_sorted2.append(
    mylabels_sorted[j])
47                  count = 0
48              else:
49                  count += 1
50                  if (i == len(x1_sorted) - 2) and (count ==
    digits - 1): # after the final comparison we do this so we
     save the last digits if we need them
51                      for j in range(i - count + 1, i + 2):
52                          mylabels_sorted2.append(
    mylabels_sorted[j])
53
54
55      # if no digits recognised in excpected size don t procced
        the following cause size bugs appear
56      if len(mylabels_sorted2)>=digits:
57          # create a list of the numbers in the image
58          numbers = []
59          newnumber = 0
60          for i in range(0, len(mylabels_sorted2)-1, digits):
61              for j in range(digits):
62                  newnumber += mylabels_sorted2[j+i] * pow(10,
    digits-1-j)
63              numbers.append(newnumber)
64              newnumber = 0
65
66          # print the result to the console
67          print("this image has the following athletes: ", end
    = "")
68          for number in numbers:
69              print(number, end = " ")
70
```

```
71      else: print("no athlete recognized")
```

Listing 3: Find athletes in photo

Ο αλγόριθμος αναγνώρισης αντικειμένων μας επιστρέφει 3 πίνακες labels, boxes και scores, μέσα στου οποίους βρίσκονται οι πληροφορίες για κάθε bounding box που βρέθηκε, και συνδέονται μεταξύ τους μέσω της θέσης που βρίσκεται κάθε στοιχείο. Έτσι, δημιουργούμε νέους πίνακες με βάση αυτούς που μας δίνει ο αλγόριθμος, ώστε μέσω αναδιατάξεων, διαγραφή στοιχείων και συγκρίσεις να προκύψει το επιθυμητό αποτέλεσμα. Η θεμελιώδης ιδέα για την υλοποίηση του στόχου είναι πώς κάθε bounding box και επομένως το ψηφίο που περιέχει, για να ανήκουν σε έναν αθλητή θα πρέπει να βρίσκονται στο ίδιο ύψος της εικόνας Όσον αφορά τον οριζόντιο άξονα πρέπει να έχουν κάποια συγκεκριμένη απόσταση μεταξύ τους, επομένως πειραματικά για κάθε διαφορετικού είδους ταμπελάκι, και οπτική γωνία εικόνας ορίζεται ένα threshold για να θεωρείται ότι δύο ψηφία ανήκουν στο ίδιο κωδικό. Τα βασικά σημεία του παραπάνω αλγόριθμου παρουσιάζονται παρακάτω.

- Δημιουργούμε 6 πίνακες με περιεχόμενα τα labels, scores, και 4 πίνακες με τα στοιχεία θέσης των bounding boxes. Θα χρησιμοποιήσουμε μόνο αυτό με την πληροφορία για το πόσο απέχει η αριστερά πάνω γωνία από την πρώτη στήλη των pixels στα αριστερά.

- Κάνουμε sort τον πίνακα x1, και αντίστοιχα στους άλλους ώστε να έχουμε στις ίδιες θέσεις των πινάκων στοιχεία για το ίδιο ψηφίο.

- Αν σε κάποιο ψηφίο ο αλγόριθμος έχει κάνει λάθος και έχει προτείνει παραπάνω από μια προβλέψεις, κρατάμε αυτή με το μεγαλύτερο score. Σε κάθε αλλαγή που κάνουμε ενημερώνουμε και τους υπόλοιπους πίνακες.

- Εντοπίζει και αφαιρεί τυχαία λανθασμένα ψηφία, είτε ψηφία από αθλητές που έχουν βρεθεί λιγότερα ψηφία από το προκαθορισμένο.

- Έχει προκύψει ένας πίνακας (mylabels_sorted2) που περιέχει τα ψηφία σε ν-αδες (όπου ν είναι το πλήθος των ψηφίων που έχει το ταμπελάκι κάθε αθλητή). Κάθε ψηφίο κατέχει μια θέση στον πίνακα. Είναι εύκολο να εξαχθεί ο δεκαδικός αριθμός κάθε αναγνωρισμένου αθλητή κάνοντας πράξεις σύμφωνες με το ορισμό του δεκαδικού συστήματος. Τα αποτελέσματα αποθηκεύονται στον τελικό πίνακα numbers, και από εκεί και πέρα μπορούμε να προχωρήσουμε σε περαιτέρω ενέργειες

Η εφαρμογή του κώδικα αναγνώρισης αθλητών σε φωτογραφίες δίνει το παρακάτω αποτέλεσμα για διαφορετικές εικόνες και πλήθος ψηφίων.

(i) Τρεις αθλητής με κωδικούς 3 ψηφίων

```
U_GARBAGE_COLLECTION=false if you'd like to disable this feature.
processing time:  6.017846584320068
this image has the following athletes: 444 496
(retinanet-env) C:\Users\pos6r\ObjectDetection\keras-retinanet>
```

(ii) Έξοδος κονσόλας

Σχήμα 32: Παράδειγμα 1ο αναγνώρισης αθλητών

(i) Αθλητές με κωδικούς 2 ψηφίων

```
library cublas64_10.dll
processing time:  5.812851667404175
this image has the following athletes: 39 79 42
```

(ii) Έξοδος κονσόλας

Σχήμα 33: Παράδειγμα 2ο αναγνώρισης αθλητών

Στα παραδείγματα που παρουσιάζονται παραπάνω παρατηρούμε πως υπάρχουν κάποιες λάθος αναγνωρίσεις ψηφίων, όπως του αριστερά αθλητή στην πρώτη, όπου το 414 αναγνωρίζεται ως 444. Αυτό έχει να κάνει με την ομοιότητα των ψηφίων 1 και 4, που με πιο λεπτομερή εκπαίδευση με κατάλληλα για το πρόβλημα δεδομένα, θα μπορούσε να αποφευχθεί. Στην δεύτερη εικόνα παρατηρούμε πως στην δεύτερη αθλήτρια, το 19 αναγνωρίζεται ως 79. Πάλι υπάρχει ομοιότητα ανάμεσα στα ψηφία, αλλά εδώ δημιουργεί λάθος και η ύπαρξη του χεριού της αθλήτριας.

Ένας συμβιβασμός που έχει γίνει εξαιτίας της εκπαίδευσης, είναι ότι θέτουμε ένα πολύ χαμηλό όριο στην πιθανότητα σωστού digit detection (εδώ 0.3), και επομένως πολλά ψηφία που με καλύτερη απόδοση αναγνώρισης θα σβήνονταν, εδώ συνεχίζουν να υπάρχουν και να επηρεάζουν το τελικό αποτέλεσμα.

4.5 Εφαρμογή της αναγνώρισης ψηφίων και αθλητών σε βίντεο

Η εφαρμογή του αλγόριθμου αναγνώρισης δεκαδικών ψηφίων σε βίντεο δεν απέχει πολύ από αυτό που ήδη έχουμε δημιουργήσει. Ένα βίντεο αποτελείται

ουσιαστικά από πολλές εικόνες στη σειρά. Αν εφαρμόσουμε τον αλγόριθμο σε κάθε frame ενός βίντεο μπορούμε τα αναγνωρίσουμε την ύπαρξη ενός αντικειμένου στο σημείο αυτό. Μέσω της βιβλιοθήκης openCV δίνεται η δυνατότητα να σκανάρεται και να επεξεργάζεται το βίντεο frame by frame, και έτσι μπορούμε να εξάγουμε στοιχεία όπως πόσα frame έχει ολόκληρο το βίντεο, πόση ώρα διήρκεσε όλη η επεξεργασία και η αναγνώριση, ποιοι αθλητές βρίσκονται σε κάθε frame και πολλά άλλα. Επίσης δίνεται η δυνατότητα εξαγωγής ενός νέου βίντεο όπου φαίνονται όλα τα bounding boxes πάνω στο αρχικό βίντεο. Παρακάτω παρουσιάζονται τα βασικά σημεία διαφοροποίησης σε σχέση με τον κώδικα αναγνώρισης αθλητών σε φωτογραφίες που αναλύθηκε στο προηγούμενο κεφάλαιο.

```
# load retinanet model
model = models.load_model(model_path, backbone_name='resnet50')

labels_to_names = {0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9', 10: '10'}

#to use camera put 0 in video capture
capture = cv2.VideoCapture('./videos_first/Test1.m4v')
# capture.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
# capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)
# capture.set(cv2.CAP_PROP_FPS, 60)
total_frames = capture.get(cv2.CAP_PROP_FRAME_COUNT);

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'DIVX')
out = cv2.VideoWriter('output.avi',fourcc, 5.0 , (1920,1080))
     #in VideoWriter we put the exact size of our video (1920, 1080)

bar = IncrementalBar('Video Processing', max = total_frames)

dict_numbers = {}
while(True):
    stime = time.time()
    ret, draw = capture.read()
```

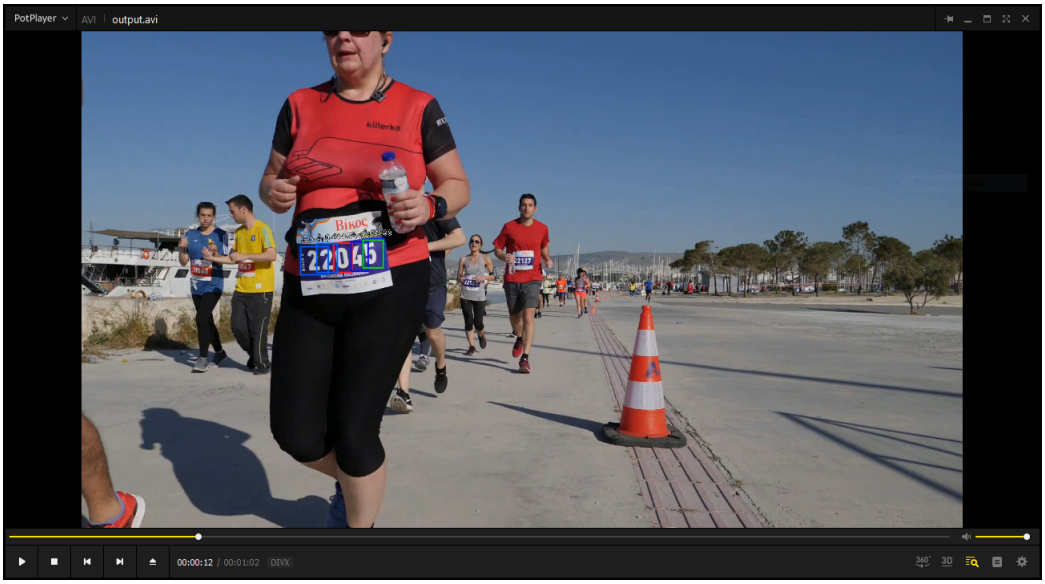
Listing 4: Open Video with openCV

```
bar.finish()
print("The number of frames in the video are: ", capture.get(cv2.CAP_PROP_FRAME_COUNT))
```

```
print("The athletes found are: ", dict_numbers)
capture.release()
out.release()
cv2.destroyAllWindows()

print("video finished")
```

Listing 5: Close video and show the result of athletes detection

Παρακάτω παρουσιάζονται παραδείγματα από διαφορετικά βίντεο αθλητών αγώνων δρόμου. Δείχνουμε ένα frame που περιέχει ολοκληρωμένα bounding boxes για κάποιον αθλητή, καθώς και τι έχουμε σαν έξοδο στην κονσόλα των windows.



(i) Στιγμιότυπο από βίντεο όπου έχουν βρεθεί κωδικοί αθλητών

```
library cublas64_10.dll
[KVideo Processing | ████████████████████████████████| 201/201
The number of frames in the video are:  201.0
The athletes found are:  {'44': [24043], '45': [22045], '51': [14045], '107': [22177], '108': [44127], '109': [14127],
123': [23777], '124': [73777]}
video finished
```

(ii) Έξοδος κονσόλας για (frames, athletes number)

Σχήμα 34: Παράδειγμα 1ο αναγνώρισης αθλητών σε βίντεο

Στην έξοδο της κονσόλας φαίνεται μία μπάρα που μας ενημερώνει σε ποιο frame βρίσκεται η επεξεργασία σε σχέση με τα συνολικά frames που περιέχει

το βίντεο, και αφού ολοκληρωθεί η αναγνώριση και η δημιουργία του βίντεο εξόδου, παρουσιάζονται τα frame που έχει γίνει επιτυχής αναγνώριση κωδικού αθλητή από άποψη πλήθους ψηφίων.

(i) Στιγμιότυπο από βίντεο όπου έχουν βρεθεί κωδικοί αθλητών

```
2020-06-13 00:58:34.947783: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic
library cublas64_10.dll
[KVideo Processing |                                        | 200/200
The number of frames in the video are:  200.0
The athletes found are:  {'1': [2844, 2911], '5': [2834, 9411], '7': [4144], '11': [2411], '13': [3342], '15': [2911, 17
44], '16': [2911], '17': [2111, 2234], '18': [8344, 8411, 4444], '19': [1244], '20': [1111], '23': [8312, 1234], '24': [
1234], '27': [8344], '28': [1234, 1117], '29': [2211], '32': [1734], '33': [2234], '34': [2811], '35': [2834, 1224], '36
': [4447], '38': [2411, 1374], '39': [1224], '41': [7044], '43': [1577], '44': [1244], '47': [2334, 1111], '49': [7941,
2211, 2533], '50': [1536], '51': [2834, 1244], '52': [2541], '53': [7711], '55': [1244], '56': [4911, 1224], '57': [2941
, 1677], '60': [2331, 2264], '61': [2234], '62': [2297], '64': [1227], '65': [2747], '66': [2377], '69': [7224], '72': [
2237], '74': [2337], '75': [2333], '77': [2251, 7224], '78': [2211, 4267], '79': [2831], '80': [1224], '81': [4441], '82
': [2331, 1224], '83': [2334], '84': [2243], '85': [2244], '86': [2241], '87': [2224], '89': [1224], '90': [2337], '92':
 [6471], '93': [4332, 2271, 2575], '94': [2277], '97': [3332], '99': [3333, 7577], '100': [2332], '101': [2432], '105':
[2111, 2244], '107': [4133, 1370], '108': [4137], '109': [2232], '110': [2332], '111': [1332], '116': [2224], '117': [12
24], '120': [7224], '123': [1322], '124': [2224], '125': [2141], '134': [4224], '142': [2241], '143': [9224], '146': [22
22], '149': [1222], '151': [2224], '156': [2224], '159': [1234], '168': [7244], '169': [7276], '172': [4274]}
video finished
[?25h
```

(ii) Έξοδος κονσόλας για (frames, athletes number)

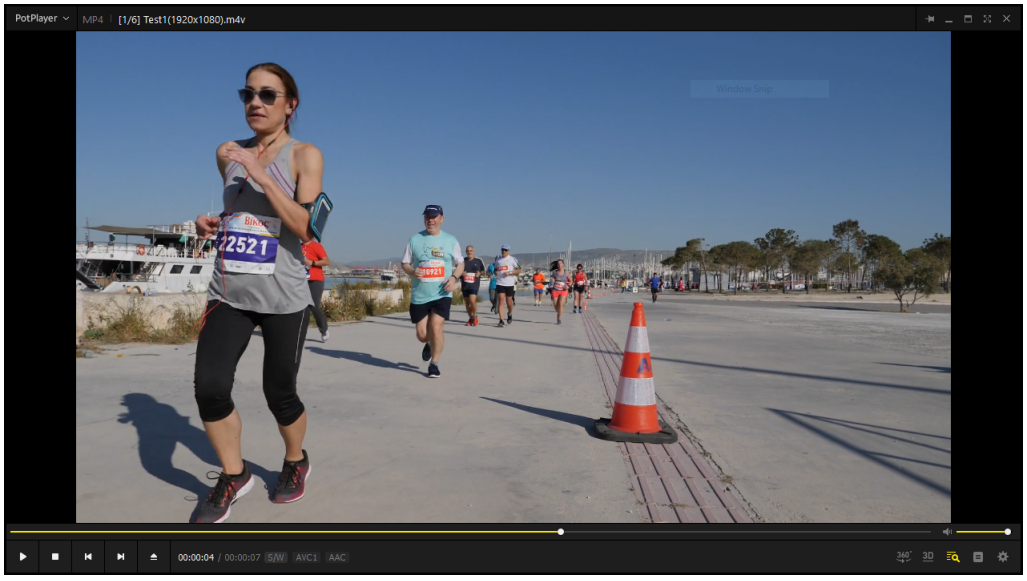
Σχήμα 35: Παράδειγμα 2ο αναγνώρισης αθλητών σε βίντεο

5 Αποτελέσματα αλγόριθμου εντοπισμού δεκαδικών ψηφίων

Θέλοντας να ελέγξουμε την απόδοση του αλγόριθμου αναγνώρισης δεκαδικών ψηφίων αλλά και την απόδοση του μοντέλου που εκπαιδεύσαμε, τροποποιήσαμε λίγο τον αλγόριθμο αναγνώρισης. Στην console εκτός των κωδικών των αθλητών που αναγνωρίστηκαν σε κάθε βίντεο, παίρνουμε στοιχεία για τα χαρακτηριστικά του βίντεο (ανάλυση, αριθμός frames, καθώς επίσης και τον χρόνο επεξεργασίας. Το σύστημα που χρησιμοποιήθηκε για τις μετρήσεις αυτές ήταν ίδιο με αυτό που χρησιμοποιήθηκε στην εκπαίδευση του μοντέλου αναγνώρισης. Αποτελείτο από επεξεργαστή Intel i5-3330(3GHz), μνήμη RAM 16GB και κάρτα γραφικών Nvidia GTX 1060 (6BG Memory).

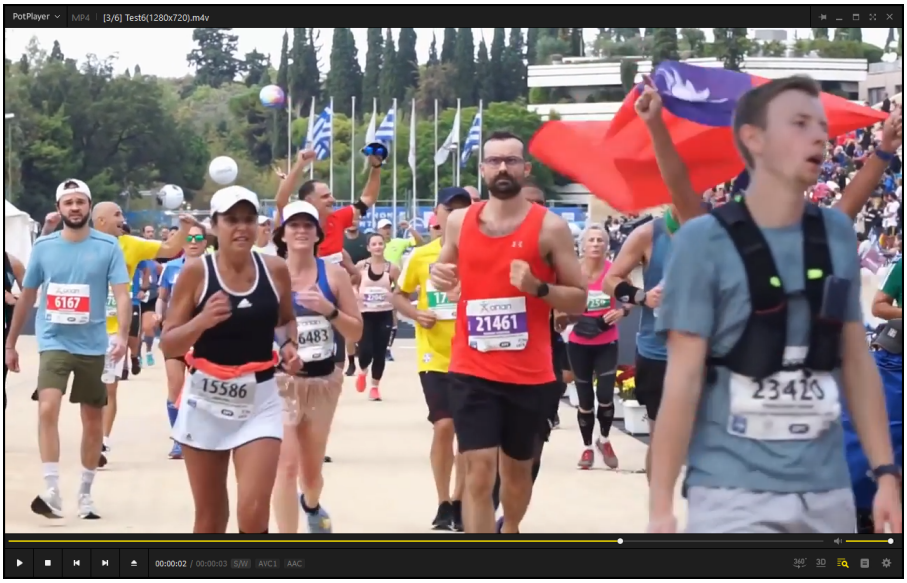
5.1 Βίντεο Μετρήσεων

Πριν παρουσιαστούν οι μετρήσεις και τα διαγράμματα που προκύπτουν από αυτές, θα παρουσιαστούν εδώ κάποια screenshot από τα βίντεο που χρησιμοποιούνται και αναφέρονται παρακάτω. Οι εικόνες αυτές δεν περιλαμβάνουν ενσωματωμένα τα bounding boxes.



Σχήμα 36: Test1

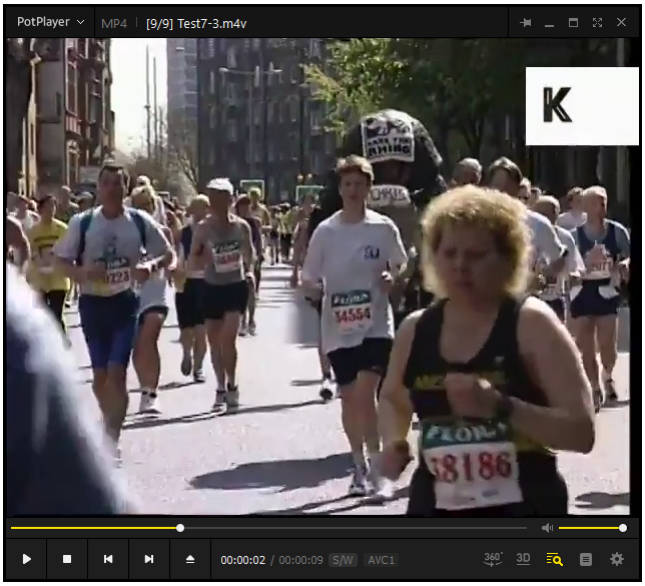
Σχήμα 37: Test5



Σχήμα 38: Test6

Σχήμα 39: Test7-1

Σχήμα 40: Test7-2



Σχήμα 41: Test7-3

5.2 Μετρήσεις

Για καλύτερη ακρίβεια στις μετρήσεις των χρόνων επεξεργασίας κάθε βίντεο επαναλάβαμε την αναγνώριση πολλαπλές φορές για κάθε βίντεο. Διαπιστώθηκε όπως αναμενόταν, ότι τα αποτελέσματα ήταν πάντα ίδια, και πώς οι χρόνοι επεξεργασίας ήταν πάρα πολύ κοντινοί. Οι χρόνοι που παρουσιάζονται στον Πίνακα 1 αποτελούν τον μέσο όρο της διάρκειας επεξεργασίας 3 διαφορετικών επαναλήψεων σε κάθε βίντεο δοκιμής. Οι στήλες που ονομάζονται ¨Καλή' και ¨Απόλυτη' περιέχουν το πλήθος των επιτυχών αναγνωρίσεων των κωδικών των αθλητών που υπάρχουν στα βίντεο. Κρίθηκε σκόπιμο να αναφερθεί και η όχι απόλυτα επιτυχής αναγνώριση.Σε κάποιες περιπτώσεις είδαμε να αποτυγχάνει μόνο για ένα ψηφίο του κωδικού η ολοκληρωμένη ταυτοποίηση. Οι παράγοντες που οδηγούν σε τέτοια σφάλματα θα αναλυθούν παρακάτω.

Στον Πίνακα 2 παρουσιάζονται επιπλέον στοιχεία από την διαδικασία των δοκιμών, που ουσιαστικά προκύπτουν από τα στοιχεία του Πίνακα 1, αλλά αποτελούν καλύτερα στοιχεία ώστε να ποσοτικοποιηθούν η απόδοση και η διάρκεια επεξεργασίας.

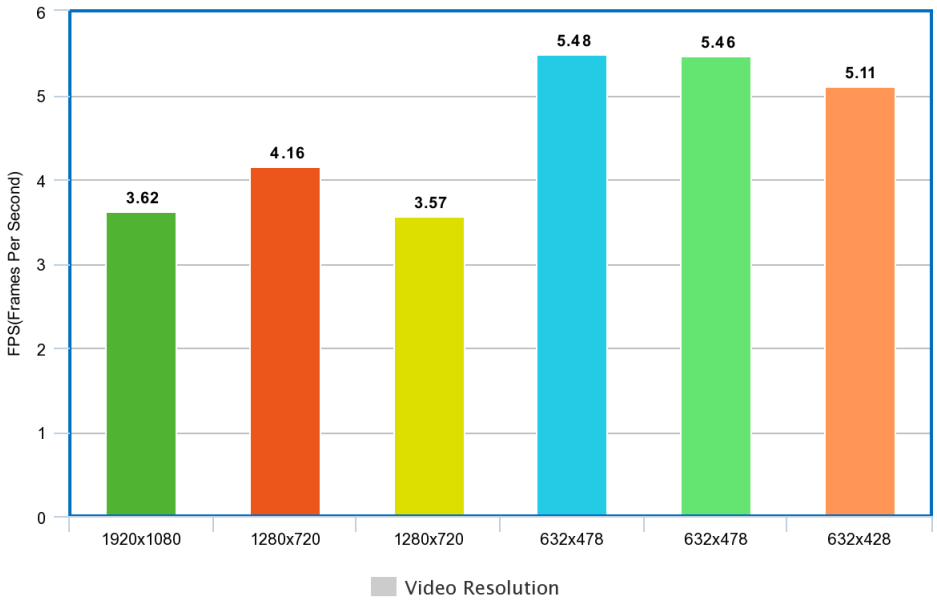
Όνομα Βίντεο	Ανάλυση	Frames	Μέσος Χρόνος (sec)	Αθλητές	Καλή	Απόλυτη
Test-1	1920 × 1080	201	55.46	4	2	1
Test-5	1280 × 720	200	48.11	10	5	2
Test-6	1280 × 720	75	20.98	4	1	1
Test-7.1	632 × 478	360	65.65	7	3	2
Test-7.2	632 × 478	313	57.31	5	2	1
Test-7.3	632 × 428	217	42.47	4	3	2

Πίνακας 1: Πίνακας Δοκιμών και Απόδοσης Αναγνώρισης Αθλητών σε Βίντεο

Όνομα Βίντεο	Frames ανά δευτερόλεπτο (FPS)	Ποσοστό Καλής Απόδοσης	Ποσοστό Απόλυτης Απόδοσης
Test-1	3.62	50%	25%
Test-5	4.16	50%	25%
Test-6	3.57	25%	25%
Test-7.1	5.48	42.9%	28.6%
Test-7.2	5.46	40%	20%
Test-7.3	5.11	75%	50%

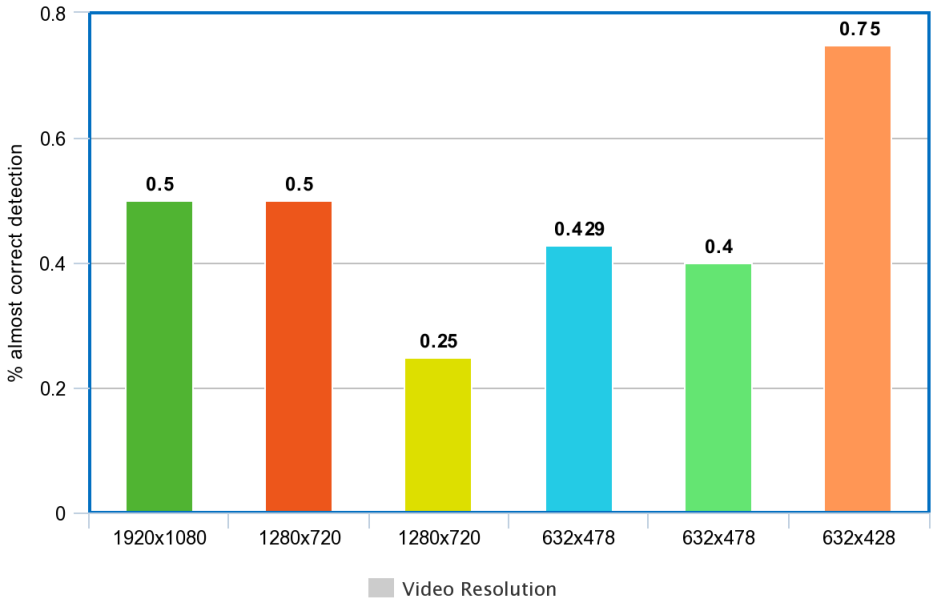
Πίνακας 2: Πίνακας Καλύτερης Ποσοτικοποίησης Απόδοσης και Διάρκειας Αναγνώρισης Αθλητών σε Βίντεο

Για καλύτερη κατανόηση των δεδομένων που παρουσιάζονται στους πίνακες 1 και 2, παραθέτουμε κάποια γραφήματα που εξάγονται από αυτούς.



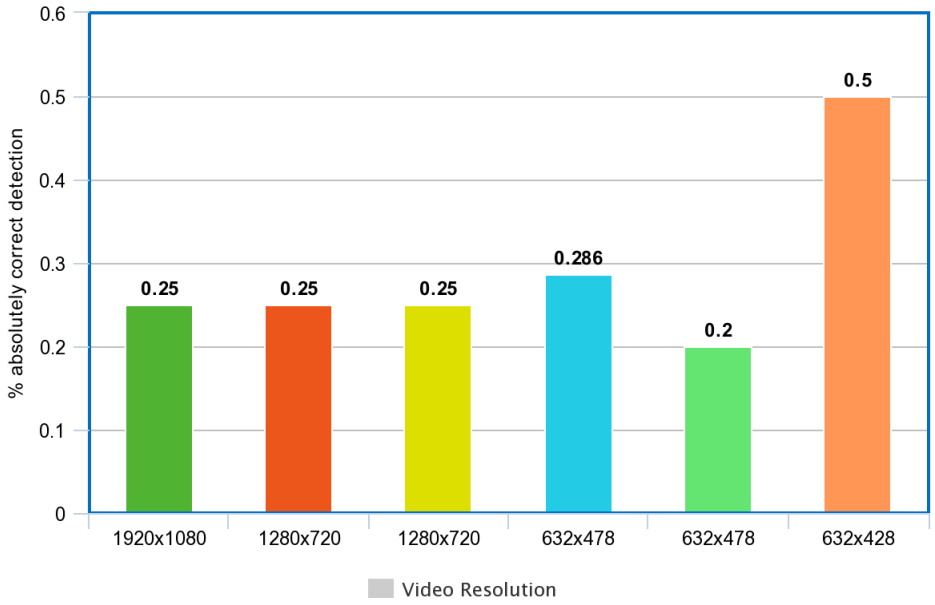
Σχήμα 42: Απόδοση ταχύτητας αλγόριθμου αναγνώρισης μετρούμενο σε FPS σε αναλογία με ανάλυση βίντεο

Επιλέχθηκε να εφαρμοστεί ο αλγόριθμο εντοπισμού σε βίντεο διαφορετικής ανάλυσης, αλλά και ποιότητας εικόνας, διοργάνωσης, και γενικώς διαφορετικών κριτηρίων εικόνας, ώστε να διαπιστώσουμε πως αντιδρά σε διαφορετικές συνθήκες. Από το παραπάνω διάγραμμα είναι φανερό πώς η ανάλυση του βίντεο παίζει σημαντικό ρόλο στην ταχύτητα επεξεργασίας. Αυτό είναι κάτι αναμενόμενο καθώς η μεγαλύτερη ανάλυση βίντεο σημαίνει περισσότερα pixels προς επεξεργασία.



Σχήμα 43: Απόδοση αλγόριθμου στην σχεδόν σωστή αναγνώριση ολόκληρου του κωδικού αθλητών σε βίντεο

Η απόδοση του αλγόριθμου δεν εξαρτάται από την ποιότητα της ανάλυσης του βίντεο όπως φαίνεται από το παραπάνω διάγραμμα. Το συμπέρασμα που μπορεί να εξαχθεί είναι πως η απόδοση του μοντέλου αναγνώρισης ψηφίων, αλλά και ο αλγόριθμος ομαδοποίησης αυτών ώστε να εξάγεται ο κωδικός των αθλητών, λειτουργεί με σχεδόν σωστή αναγνώριση περίπου 50% σε σύγκριση με τους κωδικούς που διακρίνει ένας θεατής του βίντεο. Η όχι απόλυτα σωστή απόδοση θεωρεί ως αποδεκτή την αναγνώριση με ένα ψηφίο λάθος στο κωδικό ενός αθλητή. η μελέτη αυτή έγινε ώστε να δούμε πώς λειτουργεί ο αλγόριθμος ακόμη και με δυσκολίες που προκύπτουν από ακατάλληλες γωνίες λήψεις, αλλά και φυσικά εμπόδια που μπορεί να κρύψουν κάποια ψηφία ανά στιγμές. Τέτοια εμπόδια μπορεί να είναι χέρια από τους αθλητές που όπως κινούνται κρύβουν ψηφία, είτε άλλοι αθλητές που μπαίνουν μέσα στο πλάνο.



Σχήμα 44: Απόδοση αλγόριθμου στην απόλυτα σωστή αναγνώριση ολόκληρου του κωδικού αθλητών σε βίντεο

Τέλος, εδώ παρουσιάζονται τα ποσοστά απόλυτα σωστής αναγνώρισης των κωδικών. Βλέπουμε πως και εδώ υπάρχει ανεξαρτησία με την ανάλυση του βίντεο. Επίσης η μέση απόδοση είναι περίπου 25%, και αυτό σε σύγκριση με το προηγούμενο πείραμα, αποτελεί το 50% της σχεδόν απόλυτης επιτυχούς αναγνώρισης που περιγράφηκε προηγουμένως.

6 Επίλογος-Επεκτάσεις-Εναλλακτικές

Η παρούσα διπλωματική εργασία σκοπό είχε να παρουσιάσει ένα μοντέλο, και έναν αλγόριθμο αναγνώρισης κωδικών αθλητών αποτελούμενων από δεκαδικά ψηφία. Αρχικά παρουσιάστηκαν οι σημαντικότερες τεχνολογίες αναγνώρισης αντικειμένων και προτύπων σε εικόνα. Στη συνέχεια, και αφού επιλέχθηκε η περισσότερο κατάλληλη τεχνολογία για το πρόβλημα μας αλλά και με βάση την δυνατότητα υλοποίησης σε οικιακό υπολογιστικό σύστημα, παρουσιάστηκε η διαδικασία επιλογής dataset και η εκπαίδευση πάνω σε αυτό του νευρωνικού δικτύου βασισμένο στην τεχνολογία retinanet. Τέλος αφού προσαρμόστηκε η εκτέλεση του αλγορίθμου πάνω σε βίντεο, αναπτύχθηκε αλγόριθμος ώστε να είναι εφικτό να εξαχθεί ο ν-ψήφιος κωδικός των αθλητών που βρίσκονται σε ένα βίντεο αγώνων δρόμου.

Οι βασικές κινήσεις που μπορούν να γίνουν μελλοντικά ώστε να βελτιωθεί η απόδοση αναγνώρισης κωδικών είναι οι παρακάτω.

- Δημιουργία ενός περισσότερο στοχευμένου στο πρόβλημα dataset εικόνων για την εκπαίδευση του μοντέλου. Βασικά στοιχεία διαφοροποίησης θα μπορούσαν να είναι η ανάλυση των εικόνων προς εκπαίδευση, αλλά και η οπτική και το είδος των ψηφίων που φαίνονται σε αυτές.

- Επιλογή και διερεύνηση άλλων αλγορίθμων όπως ο yolov3, οι οποίοι θα μπορούσαν να εκπαιδευτούν σε ισχυρότερα υπολογιστικά συστήματα, και στη συνέχεια να μπορούν να αποδώσουν σε πραγματικό χρόνο την αναγνώριση αθλητών σε βίντεο.

- Διερεύνηση κριτηρίων της εικόνας του βίντεο στο οποίο γίνεται η αναγνώριση. Η οπτική γωνία της κάμερας, καθώς και και προεπεξεργασία του βίντεο ώστε να λείπουν στοιχεία που μπορεί να επηρεάζουν την απόλυτα σωστή λειτουργία και αποδοτικότητα της αναγνώρισης.

- Τέλος, θα μπορούσαν να υπάρξουν προσεγγίσεις όπου ο αθλητής, ή το ταμπελάκι που περιέχει τον κωδικό της διοργάνωσης, θα αναγνωρίζεται ως ξεχωριστή οντότητα. Στη συνέχεια πάνω σε αυτά τα ξεχωριστά κομμάτια εικόνων, θα μπορούσε να εφαρμοστεί η αναγνώριση ψηφίων και η δημιουργία ολόκληρου του κωδικού

Κατάλογος Σχημάτων

Listings

Κατάλογος Πινάκων

Αναφορές

[1] David G. Lowe, 'Object Recognition from Local Scale-Invariant Features,' *Computer Science Department, University of British Columbia*, 1999.

[2] Paul Viola and Michael Jones, 'Robust Real-time Object Detection,' *Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling*, 2001.

[3] Navneet Dalal and Bill Triggs, 'Histograms of Oriented Gradients for Human Detection,' *International Conference on Computer Vision and Pattern Recognition (CVPR '05)*, 2005.

[4] Ross Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *UC Berkeley*, 2014.

[5] Ross Girshick, "Fast R-CNN," *ICCV 2015*, 2015.

[6] Wei Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," *ECCV (European Conference on Computer Vision) 2016*, 2016.

[7] Joseph Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," , 2015.

[8] Joseph Redmon and A. Farhadi, "YOLO 9000: Better, Faster, Stronger," , 2016.

[9] Joseph Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," , 2018.

[10] Tsung-Yi Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," , 2017.

[11] Jonathan Huang, V. Rathod, C. Sun, M. Zhu, and al, "Speed/accuracy trade-offs for modern convolutional object detectors," *arxiv.org*, 2017.

[12] Amazon, "Sagemaker." `https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection.html`, 2017.

[13] Amazon, "Rekognition." https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html, 2018.

[14] Facebook, "Detr." https://github.com/facebookresearch/detr, 2020.

[15] Facebook, "Detectron." https://github.com/facebookresearch/Detectron, 2018.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft COCO: Common Objects in Context*. Springer International Publishing, 2014.

[17] Everingham M., V. Gool, L. Williams, and C. et al, "the pascal visual object classes (voc) challenge," , 2010.

[18] Yann LeCun, C. Cortes, and C. J. Burges, "the mnist database of handwritten digits," .

[19] Yuval Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "reading digits in natural images with unsupervised feature learning," *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.