Πανεπιστήμιο Πειραιώς – Τμήμα Ψηφιακών Συστημάτων

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφοριακά Συστήματα και Υπηρεσίες: Μεγάλα Δεδομένα και Αναλυτική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	YOLOv4 και YOLOv4-tiny για ανίχνευση μάσκας σε πραγματικό χρόνο για κινητές συσκευές Performance Evaluation of YOLOv4 and YOLOv4-tiny for Real-Time Mask Detection on Mobile Devices
Ονοματεπώνυμο Φοιτητή	Ντζελέπη Αικατερίνη
Πατρώνυμο	Βασίλειος
Αριθμός Μητρώου	me1937
Επιβλέπων	Μιχαήλ Φιλιππάκης, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης: Φεβρουάριος 2022

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Μιχαήλ Φιλιππάκης	Δημοσθένης Κυριαζής	Μαρία Χαλκίδη
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Αναπληρώτρια Καθηγήτρια

Στην Οικογένειά μου,

Ευχαριστίες

Για την υλοποίηση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Φιλιππάκη Μιχαήλ, που χωρίς την καθοδήγηση και τις συμβουλές του δεν θα ήταν εφικτή η διεκπεραίωση της εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου για την υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια των ακαδημαϊκών μου ετών.

Περίληψη

Η ιογενής έξαρση του COVID-19 που ξεκίνησε το 2019, άλλαξε ριζικά την καθημερινότητά μας, με αρνητικές επιπτώσεις στις απλές, καθημερινές συνήθειες των πολιτών. Σε πολλές χώρες σε όλο τον κόσμο, η χρήση μάσκας είναι απαραίτητη ως μέτρο προστασίας κατά του COVID-19. Κάθε υπηρεσία, οργανισμός, διάφορα καταστήματα, σχολεία, πανεπιστήμια, νοσοκομεία, εταιρείες και πολλά άλλα μέρη, τα οποία επισκέπτονται εκατοντάδες άτομα καθημερινά, καθιστούν απαραίτητη τη χρήση μάσκας για την είσοδο σε αυτά. Το γεγονός αυτό απαιτεί τον έλεγχο των ατόμων κατά την είσοδό τους στους αντίστοιχους χώρους για να διαπιστωθεί αν φοράνε μάσκα κατά την είσοδό τους στον χώρο. Σε αυτήν την έρευνα συγκρίναμε την απόδοση του αλγορίθμου YOLOv4 και του αλγορίθμου YOLOv4-tiny σε εικόνες, βίντεο και βίντεο σε πραγματικό χρόνο. Στο επόμενο βήμα θα εφαρμόσουμε το μοντέλο YOLOv4 TFlite και YOLOv4-tiny TFlite για εφαρμογές για κινητά χρησιμοποιώντας την πλατφόρμα Android Studio. Στο προτεινόμενο σύνολο δεδομένων, ο αλγόριθμος YOLOv4 πέτυχε 92.91% mAP και η εκπαίδευση του μοντέλου χρειάστηκε περίπου 2 ώρες για 1000 επαναλήψεις. Από την άλλη πλευρά, ο YOLOv4-tiny πέτυχε 74.75% mAP και η εκπαίδευση του μοντέλου χρειάστηκε λιγότερο από μισή ώρα για 1000 επαναλήψεις. Για περαιτέρω βελτίωση μετατρέπουμε τους αλγορίθμους YOLOv4 και YOLOv4-tiny σε YOLOv4 TFlite και YOLOv4-tiny TFlite αντίστοιχα. Μετά από αυτό το βήμα, συγκρίνουμε την απόδοση του μοντέλου YOLOv4 TFlite και YOLOv4-tiny TFlite σε φορητή συσκευή. Το YOLOv4 TFlite πέτυχε ακρίβεια 96.92% σε βίντεο σε πραγματικό χρόνο στα 5017 ms και YOLOv4-tiny πέτυχε ακρίβεια 74.72% σε βίντεο σε πραγματικό χρόνο στα 491 ms.

Abstract

The viral outbreak of COVID-19 that started in the year 2019, radically changed our everyday life, with a detrimental impact on the simple, daily habits of citizens. In many countries around the world, the usage of mask is necessary as a protection measure against covid-19. Every service, organization, various stores, schools, universities, hospitals, companies, and many other places, which are attended by hundreds of people every day, make the use of a mask necessary to enter them. This fact requires the control of the persons when they enter the respective spaces to determine if they are wearing a mask when entering the area. In this research we compared performance on YOLOv4 and the Tiny-YOLOv4 algorithm on images, video, and real time video. In the next step we will implement the YOLOv4 TFlite and Tiny YOLOv4 TFlite model for mobile applications using the Android Studio platform. On the proposed dataset YOLOv4 achieved 92.91% mAP and training took around 2 hours for 1000 iterations. On the other hand, YOLOv4-tiny achieved 74.75% mAP and training took less than half an hour for 1000 iterations. For further improvement we convert YOLOv4 and YOLOv4-tiny to YOLOv4 TFlite and YOLOv4-tiny TFlite respectively. After this step we compare YOLOv4 TFlite and YOLOv4-tiny TFlite model performance on mobile device. YOLOv4 TFlite achieved 96.92% accuracy on real time video at 5017ms and YOLOv4-tiny 74.72% accuracy on real time video at 491ms.

Πίνακας Περιεχομένων

Ευχαριστίες ... 3
Περίληψη ... 4
Abstract ... 5
Πίνακας Εικόνων ... 8
1.Εισαγωγή ... 10
2.Μηχανική Μάθηση ... 12
3. Κατηγορίες Μηχανικής Μάθησης ... 13
3.1 ΠΑΛΙΝΔΡΟΜΗΣΗ ... 14
3.1.1. Απλή Γραμμική Παλινδρόμηση ... 14
3.1.2 Πολλαπλή Γραμμική Παλινδρόμηση ... 15
3.1.3 Πολυωνυμική Παλινδρόμηση ... 16
3.2 ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ ... 17
3.2.1 Λογιστική Παλινδρόμηση (Logistic Regression) ... 17
3.2.2 K-Nearest Neighbors (K-NN) ... 18
3.2.3 Support Vector Machine(SVM) ... 19
3.2.4 K-means ... 20
4. Deep Learning ... 22
4.1 Artificial Neuron Networks ... 22
4.2 Τεχνητός Νευρώνας ... 22
4.2.1 Συνάρτηση Ενεργοποίησης ... 23
4.2.2 Συνάρτηση Κόστους ... 27
4.2.3 Βελτιστοποιητές στην Βαθιά Μάθηση ... 28
Mini-Batch Gradient Descent ... 30
5.Συνελικτικά Νευρωνικά Δίκτυα (CNN) ... 31
5.1 Επίπεδο Συνέλιξης ... 32
5.2 Συνάρτηση ενεργοποίησης(Μη γραμμικότητα) ... 33
5.3 Επίπεδο Συγκεντρωσης (Pooling) ... 34
5.4 Κανονικοποίηση (Normalization Layer) ... 35
5.5 Πλήρως Συνδεδεμένο Δίκτυο (Fully Connected Layer) ... 35
5.6 Παράδειγμα ... 36
6. Object Detection ... 38
6.1 YOLO αλγόριθμος ... 39
6.1.1 Σημαντικότητα YOLO ... 39
6.1.2 YOLO Architecture ... 40
6.1.3 Περιορισμοί YOLO ... 40

6.1.4 Λειτουργία YOLO 40
6.2 YOLOV4 44
6.3 YOLOv4-tiny 49
7.Υλοποίηση ανίχνευσης αντικειμένων 51
7.1 Real-time ανίχνευση με Mobile App με χρήση Android Studio 75
Συμπεράσματα 79
Βιβλιογραφία 80
Appendix 82

Πίνακας Εικόνων

Εικόνα 1: Κατηγορίες Μηχανικής Μάθησης 14
Εικόνα 2: Δεδομένα Απλής Γραμμικής Παλινδρόμησης 15
Εικόνα 3: Ευθεία Απλής Γραμμικής Παλινδρόμησης 15
Εικόνα 4: Γραφική Αναπαράσταση Πολυωνυμικής Παλινδρόμησης 16
Εικόνα 5: Γραφική Αναπαράσταση Λογιστικής Παλινδρόμησης 17
Εικόνα 6:Διαγραμματική Απεικόνιση Δεδομένων πριν την Εφαρμογή K-NN 18
Εικόνα 7:Διαγραμματική Απεικόνιση Δεδομένων μετά την Εφαρμογή K-NN 19
Εικόνα 8: Γραφική Απεικόνιση SVM 20
Εικόνα 9: Γραφική Απεικόνιση K-means 21
Εικόνα 10: Βιολογικός Νευρώνας 23
Εικόνα 11: Γραφική Αναπαράσταση Συνάρτησης Κατωφλιού 24
Εικόνα 12: Γραφική Αναπαράσταση Σιγμοειδής Συνάρτησης 25
Εικόνα 13: Γραφική Αναπαράσταση Υπερβολικής Εφαπτομένης 25
Εικόνα 14: Γραφική Αναπαράσταση Συνάρτησης Rectified Linear Unit-ReLU 26
Εικόνα 15: Γραφική Αναπαράσταση Συνάρτησης Leaky Relu Activation Function 26
Εικόνα 16: Παράδειγμα Gradient Descent 28
Εικόνα 17: Γραφική Απεικόνιση Gradient Descent 29
Εικόνα 18: Αρχιτεκτονική Συνελικτικού Δικτύου 31
Εικόνα 19: Μετατροπή Εικόνων σε Πίνακες 32
Εικόνα 20: Φίλτρα Συνέλιξης 33
Εικόνα 21: Relu operation 34
Εικόνα 22: Μέθοδοι Συνέλιξης 35
Εικόνα 23: Fully connected layer 36
Εικόνα 24: Ακτινογραφία χωρίς πνευμονία(NORMAL:0) 37
Εικόνα 25: Ακτινογραφία με πνευμονία(PNEUMONIA:1) 37
Εικόνα 26: Αρχιτεκτονική YOLO 40
Εικόνα 27: Αλγόριθμος YOLO 41
Εικόνα 28: YOLO Residual Blocks 42
Εικόνα 29: YOLO Bounding Box 42
Εικόνα 30: YOLO Intersection Over Union (IOU) 43
Εικόνα 31: Αρχιτεκτονική YOLOv4 44
Εικόνα 32: Σύγκριση Διάφορων Αλγορίθμων με YOLOv4 45
Εικόνα 33: CSPDarknet53 46
Εικόνα 34: SPP (Spatial Pyramid Pooling Layer) 47
Εικόνα 35: Ενσωμάτωση SPP στον YOLOv4 47
Εικόνα 36: Λειτουργία PANet 48
Εικόνα 37: PANet 48
Εικόνα 38: Τροποποίηση PAN 49
Εικόνα 39: Απεικόνιση Συγκέντρωσης Χαρακτηριστικών PAN 49
Εικόνα 40: Σύγκριση YOLOv4-tiny με διάφορους αλγορίθμους σε FPS 50
Εικόνα 41: Αρχείο obj.names 52
Εικόνα 42: Αρχείο yolo.cfg 55
Εικόνα 43: Αρχείο process.py 56
Εικόνα 44: Γράφημα Average loss και mAP YOLOv4 57
Εικόνα 45: Γράφημα Average loss και mAP YOLOv4-tiny 58
Εικόνα 46: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4 60

Εικόνα 47: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny 61
Εικόνα 48: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4..... 62
Εικόνα 49: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny 63
Εικόνα 50: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4..... 64
Εικόνα 51: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny 65
Εικόνα 52: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4..... 66
Εικόνα 53: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny 67
Εικόνα 54: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4 68
Εικόνα 55: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4-tiny..... 69
Εικόνα 56: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4 69
Εικόνα 57: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4-tiny..... 70
Εικόνα 58: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4 70
Εικόνα 59: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4-tiny..... 71
Εικόνα 60: Αποτελέσματα ανιχνεύσεων σε real-time βίντεο YOLOv4 (with mask)..... 72
Εικόνα 61: Αποτελέσματα ανιχνεύσεων σε real-time βίντεο YOLOv4-tiny (with mask) 73
Εικόνα 62:Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4 (without mask)..... 73
Εικόνα 63: Αποτελέσματα ανιχνεύσεων σε real-time βίντεο YOLOv4-tiny (without mask)... 74
Εικόνα 64: Πλατφόρμα Android Studio..... 75
Εικόνα 65: Δημιουργία κατάλληλου περιβάλλοντος σε gpu 76
Εικόνα 66: Μετατροπή σε TensorFlow 76
Εικόνα 67: Αποθήκευση TensorFlow μοντέλου 77
Εικόνα 68: Μετατροπή σε tflite..... 77
Εικόνα 69: Quantize float16 77
Εικόνα 70: Real time ανίχνευση σε Android συσκευή YOLOv4 77
Εικόνα 71: Εικόνα 60: Real time ανίχνευση σε Android συσκευή YOLOv4-tiny 78

1.Εισαγωγή

Η επιδημία COVID-19 εξαπλώθηκε παγκοσμίως από τα τέλη του 2019 και απειλεί την ανθρωπότητα παρόλο που οι εκστρατείες εμβολιασμού συνεχίζονται. Φημολογείται ότι ο COVID-19 ξεκίνησε από νυχτερίδες στη Γουχάν της Κίνας τον Νοέμβριο του 2019 και εξαπλώθηκε δραματικά γρήγορα σε 114 χώρες σε όλο τον κόσμο . Η διαφορά μεταξύ του COVID-19 και ενός άλλου κορονοϊού από το παρελθόν είναι ότι μπορεί να μεταδοθεί από τον αέρα που αναπνέουν οι άνθρωποι καθημερινά και μπορεί να μολύνει ανθρώπους ενώ επικοινωνούν απλά μέσα σε πολύ σύντομο χρονικό διάστημα. Επίσης, ο COVID-19 θα παραμείνει λανθάνον στο σώμα του μολυσμένου ατόμου και τα συμπτώματα θα εμφανιστούν έως και 14 ημέρες αργότερα. Κατά τη διάρκεια αυτής της περιόδου, ο ασθενής δεν θα εμφανίσει κανένα σύμπτωμα, πράγμα που σημαίνει ότι οι ασυμπτωματικοί ασθενείς δεν μπορούν να εντοπιστούν και η κυβέρνηση δεν θα μπορούσε να τους απομονώσει εγκαίρως για να αποφύγει την περαιτέρω εξάπλωση του ιού. Επιπλέον, τα άτομα με υποκείμενα νοσήματα και άνω των 50 διατρέχουν μεγαλύτερο κίνδυνο, επειδή για αυτούς οι πιθανότητες του COVID-19 να επηρεάσουν το αναπνευστικό σύστημα που προκαλεί οξεία πνευμονική νόσο και μετατρέπεται σε θανατηφόρο νόσημα είναι υψηλότερες. Ακόμη, ορισμένες χώρες ανακοίνωσαν περιορισμούς για τον COVID-19, όπως εθνικό lockdown, απαγόρευση κυκλοφορίας, ταξιδιωτικούς περιορισμούς, κλείσιμο δημόσιων χώρων, φυσική απόσταση και κλείσιμο των συνόρων. Αυτοί οι περιορισμοί παρουσίασαν οξείες προκλήσεις στις αναπτυσσόμενες χώρες, όπου οι εξασθενημένες υποδομές, τα υπερβολικά εκτεταμένα συστήματα υγείας, η ανεπαρκής χρηματοδότηση και η περιορισμένη επιτήρηση της δημόσιας υγείας έβαλαν σε κίνδυνο την πιθανή αποτελεσματικότητά τους. Η πανδημία έχει επιβάλει οικονομική πίεση που έχει οδηγήσει σε κοινωνικές και πολιτικές προκλήσεις. Αντίστοιχα, οι περισσότερες χώρες έχουν αυξήσει τα lockdown, αλλά οι κυβερνήσεις έχουν υποχρεώσει τους πολίτες να φορούν μάσκες σε δημόσιους χώρους, όπως σε καταστήματα, σχολεία, πανεπιστήμια και χώρους εργασίας, παρά την αύξηση των ποσοστών εμβολιασμού. Οι στατιστικές έχουν δείξει ότι η χρήση μάσκας προσώπου μειώνει σημαντικά τη μεταφορά του ιού. Οι περιορισμοί μεγάλης κλίμακας δεν είναι εύκολο να εφαρμοστούν, να τηρηθούν και στη συνέχεια δύσκολο να εφαρμοστούν και να διατηρηθούν, οι οποίοι οδηγούν σε ατελή δημόσια συμμόρφωση, ειδικά εάν υπάρχει σημαντικός αντίκτυπος στους κοινωνικούς και πολιτικούς κανόνες, την οικονομία και την ψυχολογική ευημερία του πληγέντος πληθυσμού. Εμφανώς, η προσοχή στρέφεται πλέον προς τον εμβολιασμό των πληθυσμών μετά την επιτυχή ανάπτυξη των εμβολίων. Ωστόσο, οι αναδυόμενες παραλλαγές του COVID-19, η τακτική μετακίνηση άτυπων εμπόρων και η πώληση πλαστών πιστοποιητικών εμβολιασμού εξακολουθούν να απειλούν την πρόοδο που έχει σημειωθεί προς τον περιορισμό του ιού σε ορισμένες χώρες.

Για το λόγο αυτό, ο Παγκόσμιος Οργανισμός Υγείας έχει προτείνει την υποχρεωτική χρήση ιατρικής μάσκας για τον περιορισμό της εξάπλωσης του ιού. Όπως λέγεται, η πρόληψη είναι καλύτερη από τη θεραπεία, έτσι η χρήση μάσκας κατά τη διάρκεια των κοινωνικών καθημερινών συναναστροφών είναι πια αναγκαία. Τώρα η χρήση μάσκας είναι πλέον απαραίτητη για την είσοδο σε διάφορα μέρη όπως το σχολείο, το

σούπερ μάρκετ, τα εμπορικά κέντρα, τους κινηματογράφους, σε πολλά γραφεία και εταιρείες, νοσοκομεία, γυμναστήρια κ.λπ. Αν κάποιος δεν φοράει μάσκα, δεν επιτρέπεται να εισέλθει σε οποιανδήποτε εσωτερικό χώρο. Επίσης, σε πολλές χώρες όπου τα κρούσματα κορονοϊού αυξάνονται δραματικά είναι απαραίτητη η χρήση μάσκας ακόμα και στο δρόμο.

Ένα νέο πρόβλημα εμφανίστηκε καθώς οι μάσκες γίνονται αναγκαιότητα για την πρόληψη του COVID-19. Πολλά μέρη με πολύ κόσμο, όπως σούπερ μάρκετ, δημόσιες συγκοινωνίες, αεροδρόμια, σχολεία, εμπορικά κέντρα πρέπει να επιβλέπουν εάν όλοι για να μπουν σε αυτά φορούν μάσκα ή όχι.

Σήμερα, η τεχνολογία ανίχνευσης αντικειμένων μας δίνει τη δυνατότητα να την ενσωματώσουμε σε κάμερες υπολογιστή, κινητών και σε άλλες συσκευές για να πραγματοποιήσουμε την ανίχνευση μάσκας προσώπου, έτσι ώστε να επιτυγχάνεται η αυτόματη ανίχνευση μάσκας χωρίς επαφή. Ειδικά οι αλγόριθμοι βαθιάς εκμάθησης είναι οι πιο δημοφιλείς στο πεδίο ανίχνευσης αντικειμένων. Αυτή τη στιγμή το YOLOv4 (You Only Look Once) είναι ένα από τα πιο δημοφιλή μοντέλα ανίχνευσης αντικειμένων, επειδή συνδυάζει υψηλή ακρίβεια και ταχύτητα. Το YOLOv4 τρέχει δύο φορές πιο γρήγορα από ένα πρόσφατο μοντέλο ανίχνευσης αντικειμένων, το EfficientDet, με παρόμοια ακρίβεια. Επίσης, το YOLOv4-tiny σχεδιάστηκε για να αυξάνει την ταχύτητα και να μειώνει το υπολογιστικό κόστος όσο είναι δυνατόν.

Στην παρούσα διπλωματική εργασία παρουσιάζεται ένας από τους πιο δημοφιλείς αλγόριθμους ανίχνευσης αντικειμένων, YOLOv4 και στην έκδοση YOLOv4-tiny, η οποία είναι ιδανική για ανίχνευση μάσκας σε πραγματικό χρόνο. Στο πρόβλημα της ανίχνευσης μάσκας η ταχύτητα του μοντέλου είναι το πιο σημαντικό μετά την ακρίβεια της πρόβλεψης. Επιπλέον, το YOLOv4-tiny είναι ένα πολύ ελαφρύ μοντέλο που με τη βοήθεια της Tensorflow τα βάρη YOLOv4 και YOLOv4-tiny μπορούν να μετατραπούν σε tflite και να γίνουν πολύ ελαφρύτερα. Με αυτήν την τεχνική θα αναπτύξουμε ένα μοντέλο ανίχνευσης μάσκας που μπορεί να χρησιμοποιηθεί σε ενσωματωμένες συσκευές ή σε κάμερες παρακολούθησης.

2.Μηχανική Μάθηση

Μηχανική Μάθηση (Machine Learning) και Τεχνητή Νοημοσύνη (Artificial Intelligent).

Η Μηχανική Μάθηση αποτελεί έναν τομέα της επιστήμης των υπολογιστών που έχει ως βάση τα υπολογιστικά μαθηματικά και την επιστήμη της στατιστικής. Οι αλγόριθμοι της μηχανικής μάθησης δίνουν τη δυνατότητα να μάθουν από τα δεδομένα, ακόμα και να βελτιωθούν χωρίς να είναι πλήρως προγραμματισμένοι και αυτοματοποιημένοι.

Η βασική ιδέα της Μηχανικής Μάθησης στηρίζεται στη δημιουργία αλγορίθμων που μπορούν να λάβουν κάποια δεδομένα ως είσοδο και χρησιμοποιώντας στατιστική ανάλυση να προβλέπουν μια πιθανή έξοδο, ενώ παράλληλα ενημερώνουν τις εξόδους τους καθώς εισάγονται νέα δεδομένα.

3. Κατηγορίες Μηχανικής Μάθησης

Στο σημείο αυτό είναι σημαντικό να αναφέρουμε ότι υπάρχουν τρία είδη μηχανικής μάθησης:

1. Επιβλεπόμενη Μάθηση (Supervised Learning)
2. Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)
3. Semi-supervised Learning
4. Ενισχυτική Μάθηση (Reinforcement Learning)

- Επιβλεπόμενη Μάθηση (Supervised Learning)

Στη φάση της εκπαίδευσης ο αλγόριθμος τροφοδοτείται με δεδομένα με ετικέτες (label data), τα οποία διδάσκουν ποια έξοδος σχετίζεται με κάθε συγκεκριμένη τιμή εισόδου. Στη συνέχεια το εκπαιδευμένο μοντέλο υλοποιείται με δεδομένα για τεστ (test data). Τα συγκεκριμένα δεδομένα έχουν ετικέτες (labels), αλλά οι ετικέτες δεν έχουν αποκαλυφθεί στον αλγόριθμο. Ο στόχος των δεδομένων δοκιμής είναι να μετρηθεί η ακρίβεια που θα εκτελεστεί ο αλγόριθμος σε δεδομένα χωρίς ετικέτες (labels).

Υπάρχουν δύο επιπλέον κατηγορίες στις οποίες μπορούμε να διαχωρίσουμε την επιβλεπόμενη μάθηση, την παλινδρόμηση (regression) και την ταξινόμηση (classification). Οι πιο συνήθεις αλγόριθμοι εποπτευόμενης μάθησης είναι οι παρακάτω, οι οποίο αναλύονται και στην συνέχεια της παρούσας εργασίας:

- ✓ Naïve Bayes
- ✓ Nearest Neighbors
- ✓ Decision Trees
- ✓ Support Vector Machine
- ✓ Neural Networks

- Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

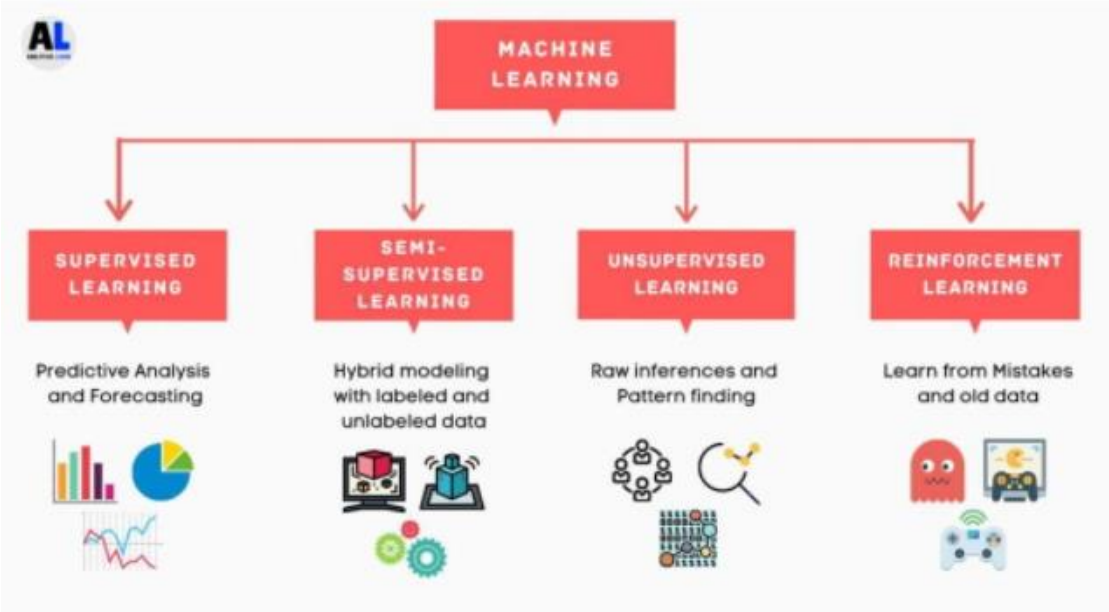
Η μη επιβλεπόμενη μάθηση έχει το πλεονέκτημα να είναι σε θέση να εργαστεί με δεδομένα χωρίς ετικέτα, το οποίο σημαίνει πως δεν απαιτείται η ανθρώπινη επέμβαση έτσι ώστε τα δεδομένα να γίνονται ευανάγνωστα. Αυτοί οι αλγόριθμοι έχουν την δυνατότητα να ανακαλύπτουν κρυφές δομές. Οι κοινές συσχετίσεις μεταξύ κάποιων μοτίβων των δεδομένων γίνονται αντιληπτές από τον αλγόριθμο με αφηρημένο τρόπο χωρίς την παρέμβαση του ανθρώπου. Η δυνατότητα δημιουργίας αυτών των κρυφών δομών είναι που καθιστά τους αλγόριθμους χωρίς επίβλεψη πιο ευέλικτους. Η πιο συνηθισμένη χρήση της μάθησης χωρίς επίβλεψη είναι στα προβλήματα ομαδοποίησης (clustering), με τους πιο πολυσυζητημένους αλγόριθμους ναι είναι ο k-means και η ιεραρχική ομαδοποίηση (hierarchical clustering).

- Semi-Supervised Learning

Σε τέτοιου είδους προβλήματα ο αλγόριθμος τροφοδοτείται με μικρό αριθμό δεδομένων που εμπεριέχουν ετικέτα και με μεγάλο αριθμό δεδομένων χωρίς ετικέτα.

- Reinforcement Learning

Το σύστημα προσπαθεί να μάθει αλληλοεπιδρώντας με το περιβάλλον του και εκπαιδεύεται σε μηχανισμούς επιβράβευσης και τιμωρίας. Πρόκειται για τη λήψη της καλύτερης δυνατής απόφασης ή ενέργειας με στόχο την απόκτηση μέγιστων ανταμοιβών και ελάχιστων τιμωριών.



Εικόνα 1: Κατηγορίες Μηχανικής Μάθησης

3.1 ΠΑΛΙΝΔΡΟΜΗΣΗ

Η στατιστική μεθοδολογία που χρησιμοποιεί τη σχέση μεταξύ δύο ή περισσοτέρων ποσοτικών μεταβλητών έτσι ώστε η μία να μπορεί να προβλεφθεί από την άλλη ονομάζεται Ανάλυση Παλινδρόμησης (Regression Analysis). Τον όρο «Παλινδρόμηση» χρησιμοποίησε πρώτος ο F.Galton το 1900 όταν μελετώντας τη σχέση μεταξύ ύψους των γονιών και αυτού των παιδιών τους παρατήρησε ένα είδος επαναφοράς (παλινδρόμησης) του ύψους των παιδιών στο ύψος των γονιών τους.

3.1.1. Απλή Γραμμική Παλινδρόμηση

Η γραμμική παλινδρόμηση ποσοτικοποιεί τη σχέση μεταξύ μίας ή περισσοτέρων μεταβλητών και μια μεταβλητής αποτελέσματος. Η γραμμική παλινδρόμηση χρησιμοποιείται συνήθως για προγνωστική ανάλυση και μοντελοποίηση. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για τον ποσοτικό προσδιορισμό των σχετικών επιπτώσεων της ηλικίας, του φύλου και της διατροφής (οι μεταβλητές πρόβλεψης) στο ύψος (μεταβλητή έκβασης).

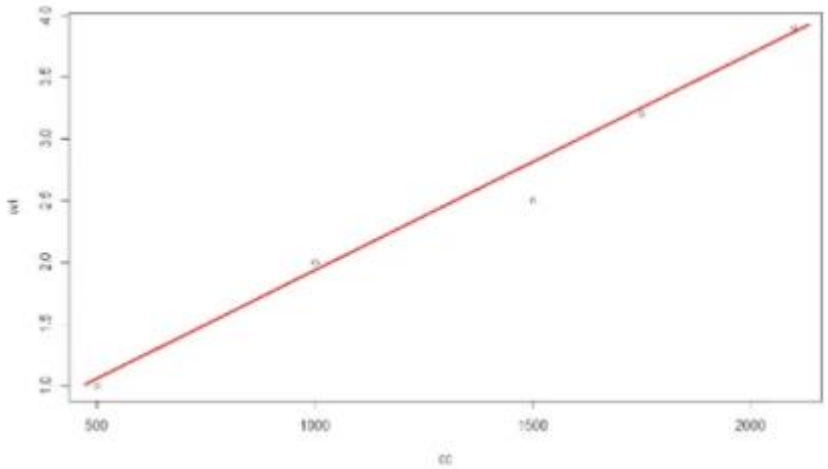
$$Y = \beta_0 + \beta_1 + X + \varepsilon$$

- Y είναι η προβλεπόμενη τιμή της εξαρτημένης μεταβλητής (y) για οποιαδήποτε δεδομένη τιμή της ανεξάρτητης μεταβλητής (x).
- β_0 είναι η τομή, η προβλεπόμενη τιμή του y όταν το x είναι 0.
- β_1 είναι ο συντελεστής παλινδρόμησης – πόσο περιμένουμε να αλλάξει το y καθώς το x αυξάνεται.
- X είναι η ανεξάρτητη μεταβλητή (η μεταβλητή που αναμένουμε επηρεάζει το y)
- ε είναι το σφάλμα της εκτίμησης ή πόση διακύμανση υπάρχει στην εκτίμησή μας για τον συντελεστή παλινδρόμησης.

Ας υποθέσουμε ότι θέλουμε να προβλέψουμε την αύξηση βάρους με βάση της θερμίδες που καταναλώνονται με βάση τα παρακάτω δεδομένα του πίνακα:

Calories Consumed	Weight Gain
500	1
1000	2
1500	2.5
1750	3.2
2100	3.9

Εικόνα 2: Δεδομένα Απλής Γραμμικής Παλινδρόμησης



Εικόνα 3: Ευθεία Απλής Γραμμικής Παλινδρόμησης

3.1.2 Πολλαπλή Γραμμική Παλινδρόμηση

Η πολλαπλή γραμμική παλινδρόμηση επεκτείνει την απλή γραμμική παλινδρόμηση έτσι ώστε να περιλαμβάνει περισσότερες από μία επεξηγηματικές μεταβλητές και

στις δύο περιπτώσεις εξακολουθούμε να χρησιμοποιούμε τον όρο γραμμικός επειδή υποθέτουμε ότι η μεταβλητή απόκρισης σχετίζεται άμεσα με έναν γραμμικό συνδυασμό των επεξηγηματικών μεταβλητών.

Η μαθηματική εξίσωση για την πολλαπλή γραμμική παλινδρόμηση έχει την ίδια μορφή με την απλή λογιστική παλινδρόμηση αλλά περιλαμβάνει περισσότερους όρους. Έστω ένα δείγμα πολλαπλών δεδομένων x πλήθους n $x_1, x_2, x_3, \dots, x_n$ και ένα σύνολο συντελεστών $\beta_1, \beta_2, \beta_3, \dots, \beta_n$ τότε η έξοδος y θα είναι :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_n x_n$$

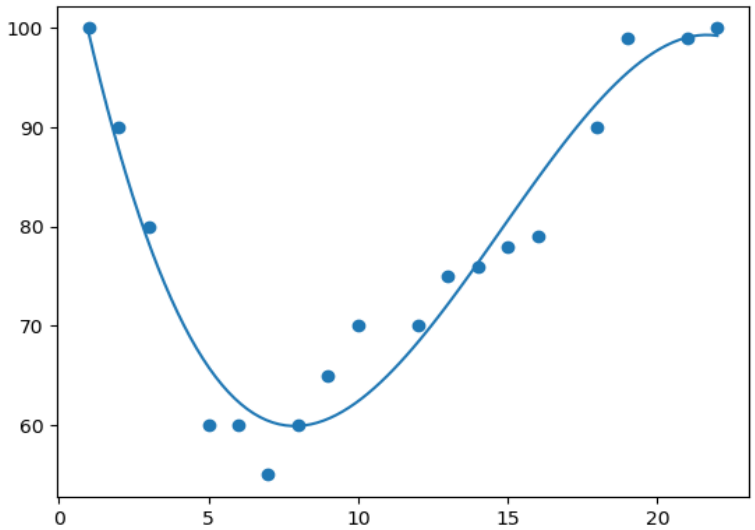
3.1.3 Πολυωνυμική Παλινδρόμηση

Η πολυωνυμική παλινδρόμηση είναι ένα είδος γραμμικής παλινδρόμησης, όπου μόνο λόγω της μη γραμμικής σχέσης μεταξύ εξαρτημένων και ανεξάρτητων μεταβλητών προσθέτουμε ορισμένους πολυωνυμικούς όρους στη γραμμική παλινδρόμηση για να τη μετατρέψουμε σε πολυωνυμική παλινδρόμηση.

Ας υποθέσουμε ότι έχουμε το X ως ανεξάρτητα δεδομένα και το Y ως εξαρτημένα δεδομένα. Πριν την τροφοδοσία των δεδομένων στο στάδιο προ επεξεργασίας των δεδομένων μετατρέπουμε τις μεταβλητές εισόδου σε πολυωνυμικούς όρους χρησιμοποιώντας κάποιο βαθμό.

Για παράδειγμα ας υποθέσουμε πως η τιμή εισόδου είναι το 35 και ο βαθμός ενός πολυωνύμου είναι το 2. Έτσι θα υπολογίσω το 35^0,το 35^1και το 35^2, βοηθώντας με αυτόν τον τρόπο στη μη γραμμική ερμηνεία των δεδομένων. Η εξίσωση της πολυωνυμικής παλινδρόμησης παρουσιάζεται παρακάτω:

$$Y = \beta_0 + \beta_1 x_{1+} \beta_2 x_1^2 + \cdots + \beta_n x_1^n$$



Εικόνα 4: Γραφική Αναπαράσταση Πολυωνυμικής Παλινδρόμησης

3.2 ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ

3.2.1 Λογιστική Παλινδρόμηση (Logistic Regression)

Η Λογιστική Παλινδρόμηση είναι μια στατιστική μέθοδος για να προβλέπει δυαδικές κλάσεις (0 ή 1). Για παράδειγμα μπορεί να χρησιμοποιηθεί σε προβλήματα αναγνώρισης πνευμονίας ή όχι (η πνευμονία αντιστοιχεί σε 1 ενώ όχι πνευμονία αντιστοιχεί σε 0), δηλαδή υπολογίζει την πιθανότητα εμφάνισης ενός συμβάντος.

Πρόκειται για μια ειδική περίπτωση γραμμικής παλινδρόνησης, όπου η μεταβλητή στόχος είναι κατηγορηματικής φύσης (0 ή 1). Η Λογιστική Παλινδρόμηση προβλέπει την πιθανότητα εμφάνισης ενός δυαδικού συμβάντος χρησιμοποιώντας τη σιγμοειδή συνάρτηση στην συνάρτηση της γραμμικής παλινδρόμησης. Παρακάτω ακολουθεί η εξίσωση της γραμμικής παλινδρόμησης:

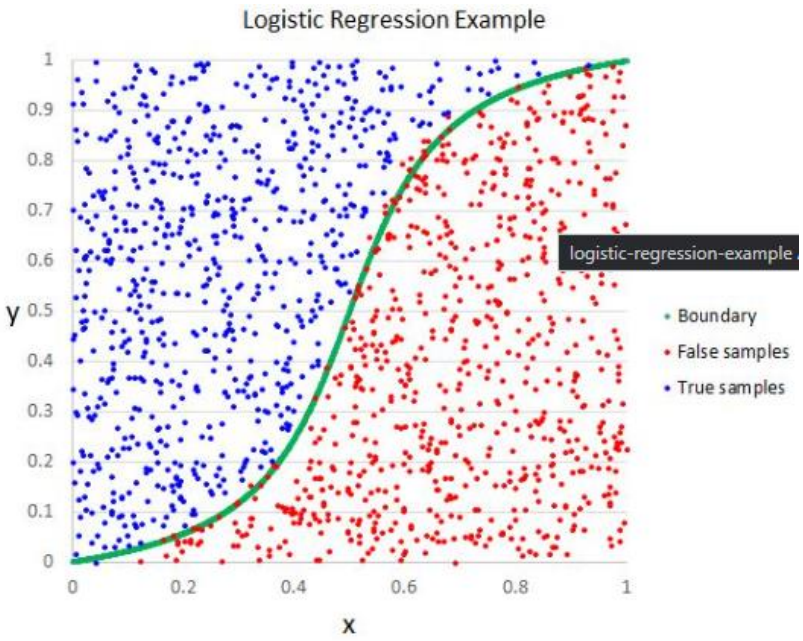
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_n x_n$$

Σιγμοειδής Συνάρτηση:

$$p = 1 + 1/e^{-y}$$

Εφαρμόζοντας τη σιγμοειδή συνάρτηση στη συνάρτηση της γραμμικής παλινδρόμησης προκύπτει η συνάρτηση της λογιστικής παλινδρόμησης όπως φαίνεται παρακάτω:

$$p = 1 + 1/e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_n x_n)}$$



Εικόνα 5: Γραφική Αναπαράσταση Λογιστικής Παλινδρόμησης

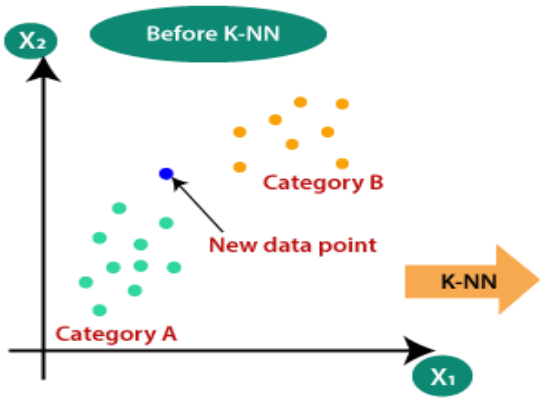
3.2.2 K-Nearest Neighbors (K-NN)

Ο K-Nearest Neighbors (K-NN) είναι ένας αλγόριθμος επιβλεπόμενης Μηχανικής Μάθησης που χρησιμοποιείται κυρίως για προβλήματα κατηγοριοποίησης (classification).

Βήματα που ακολουθούνται στην εφαρμογή του K-NN αλγόριθμου:

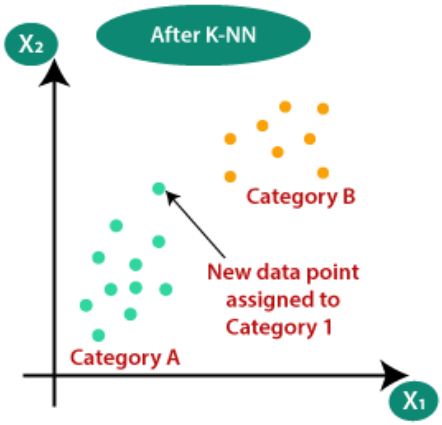
1. Θεωρούμε έναν ακέραιο αριθμό για το k, ο οποίος υποδηλώνει τον αριθμό των πλησιέστερων γειτόνων από το σημείο το οποίο θέλουμε να κατηγοριοποιήσουμε.
2. Για το σημείο των δεδομένων μας που θέλουμε να κατηγοριοποιήσουμε υπολογίζουμε τον αριθμό των γειτόνων (που αντιστοιχεί στον αριθμό του k), που βρίσκονται στην ελάχιστη απόσταση από το σημείο, σύμφωνα με την ευκλείδεια απόσταση
3. Το σημείο που θέλουμε να κατηγοριοποιήσουμε θα ανήκει στην κλάση στην οποία ανήκουν και οι περισσότεροι πλησιέστεροι γείτονές του.

Δεδομένα πριν την εφαρμογή του K-NN παρουσιάζονται παρακάτω:



Εικόνα 6:Διαγραμματική Απεικόνιση Δεδομένων πριν την Εφαρμογή K-NN

Στην συνέχεια παρουσιάζονται τα ίδια δεδομένα ύστερα από την εφαρμογή του K-NN:



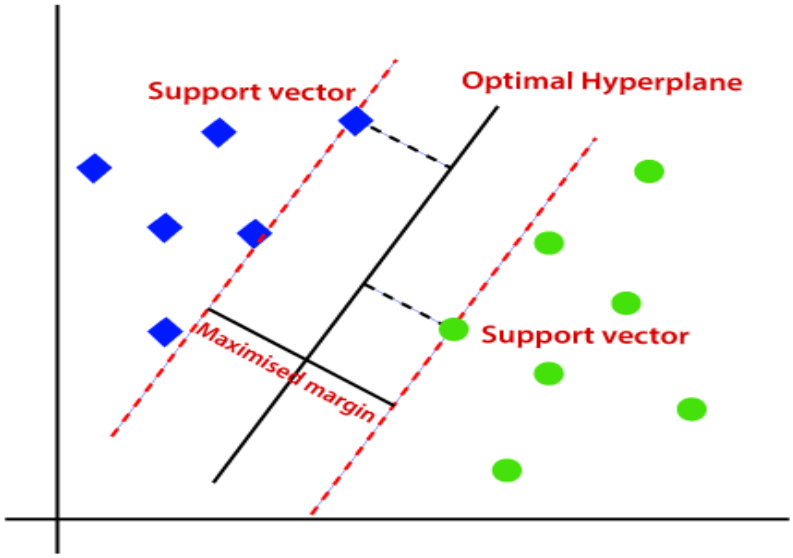
Εικόνα 7:Διαγραμματική Απεικόνιση Δεδομένων μετά την Εφαρμογή K-NN

3.2.3 Support Vector Machine(SVM)

Ο Support Vector Machine (SVM) είναι ένας αλγόριθμος επιβλεπόμενης Μηχανικής Μάθησης που μπορεί να χρησιμοποιηθεί για προβλήματα κατηγοριοποίησης αλλά και παλινδρόμησης. Ωστόσο, πιο συχνά χρησιμοποιείται σε προβλήματα κατηγοριοποίησης. Στον SVM αλγόριθμο κάθε αντικείμενο των δεδομένων σχεδιάζεται σαν ένα ξεχωριστό σημείο σε ένα χώρο n-διαστάσεων (όπου n είναι ο αριθμός των χαρακτηριστικών που διαθέτουμε), με την τιμή κάθε χαρακτηριστικού να είναι η τιμή μιας συγκεκριμένης συντεταγμένης.

Ο στόχος του αλγόριθμου SVM είναι να βρει την ευθεία γραμμή (hyperplane) που ταξινομεί ευδιάκριτα τα δεδομένα στο N-διαστάσεων χώρο. Για να διαχωριστούν σωστά τα δεδομένα, υπάρχουν πολλές ευθείες (hyperplanes) που είναι κατάλληλες για επιλογή. Ο στόχος είναι να βρεθεί η γραμμή που έχει τα μεγαλύτερα περιθώρια, δηλαδή που έχει τη μεγαλύτερη απόσταση από τα σημεία των δεδομένων της κάθε κλάσης αντίστοιχα.

Support Vectors είναι τα σημεία των δεδομένων που είναι πιο κοντά στην ευθεία γραμμή. Χρησιμοποιώντας τους Support Vectors, μεγιστοποιούμε τα περιθώρια του κατηγοριοποιητή. Διαγράφοντας τους Support Vectors αλλάζουμε την θέση της ευθείας γραμμής (hyperplane). Εν κατακλείδι, οι Support Vectors είναι τα σημεία εκείνα που βοηθούν στην δημιουργία του αλγορίθμου Support Vector Machine (SVM).



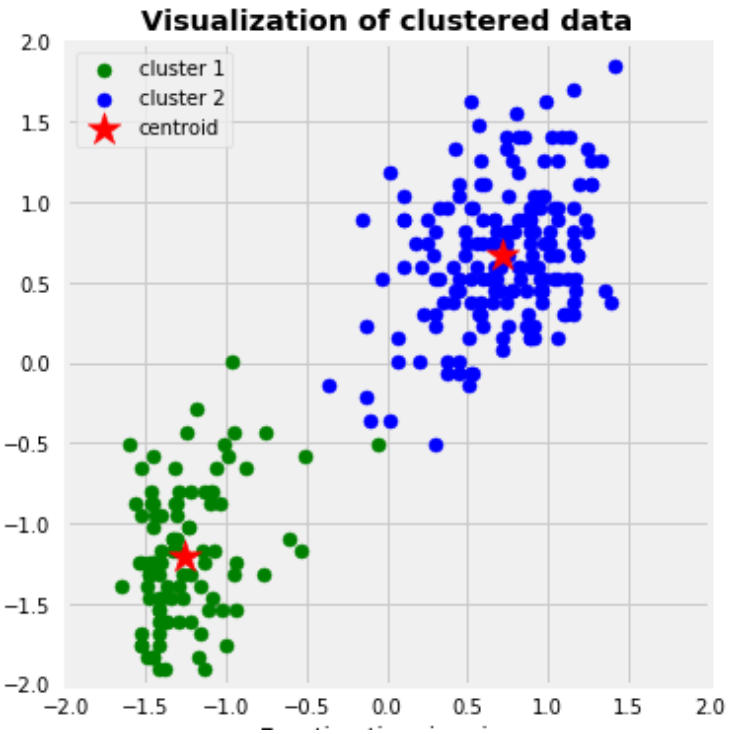
Εικόνα 8: Γραφική Απεικόνιση SVM

3.2.4 K-means

Ο K-Means είναι ένας από τους απλούστερους αλγορίθμους μη επιβλεπόμενης μάθησης, που λύνει το γνωστό πρόβλημα της συσταδοποίησης. Είναι ένας επαναληπτικός αλγόριθμος που προσπαθεί να χωρίσει το σύνολο των δεδομένων σε k προκαθορισμένες συστάδες (clusters), όπου κάθε σημείο δεδομένων ανήκει σε μία μόνο συστάδα (cluster). Προσπαθεί να κάνει τα σημεία των δεδομένων εντός της συστάδας (cluster) όσο το δυνατόν πιο όμοια.

Ο αλγόριθμος k-means εκτελεί τα παρακάτω βήματα:

1. Για να διεξαχθεί ο αλγόριθμος k-means είναι αναγκαίο να διευκρινιστεί ο αριθμός των συστάδων (clusters) που συμβολίζεται από το γράμμα k.
2. Στη συνέχεια επιλέγονται τυχαία k σημεία δεδομένων και κάθε ένα από αυτά τα σημεία δεδομένων αντιστοιχίζεται σε μία συστάδα (cluster). Με πιο απλά λόγια, τα δεδομένα ταξινομούνται με βάση των αριθμό των σημείων δεδομένων.
3. Στο σημείο αυτό υπολογίζονται τα κεντροειδή (centroids) της κάθε συστάδας (cluster).
4. Τέλος επαναλαμβάνονται τα παραπάνω έως ότου βρεθεί το βέλτιστο κέντρο, το οποίο είναι η εκχώρηση σημείων δεδομένων στις συστάδες και δεν αλλάζει πλέον.



Εικόνα 9: Γραφική Απεικόνιση K-means

4. Deep Learning

Η Βαθιά Μάθηση είναι ένα υποσύνολο της Μηχανικής Μάθησης, το οποίο είναι ουσιαστικά ένα νευρωνικό δίκτυο με τρία ή περισσότερα επίπεδα. Αυτά τα νευρωνικά δίκτυα προσπαθούν να προσομοιώσουν τη συμπεριφορά του ανθρώπινου εγκεφάλου επιτρέποντάς του να μάθει από μεγάλες ποσότητες δεδομένων. Ενώ ένα νευρωνικό δίκτυο με ένα μόνο στρώμα μπορεί ακόμα να κάνει κατά προσέγγιση προβλέψεις, πρόσθετα κρυφά στρώματα μπορούν να βοηθήσουν στην βελτιστοποίηση και τη βελτίωση της ακρίβειας.

Επιπλέον, η Βαθιά Μάθηση οδηγεί πολλές εφαρμογές και υπηρεσίες τεχνητής νοημοσύνης που βελτιώνουν την αυτοματοποίηση, εκτελώντας αναλυτικές και φυσικές εργασίες χωρίς ανθρώπινη παρέμβαση. Η τεχνολογία Βαθιάς Μάθησης βρίσκεται πίσω από προϊόντα και υπηρεσίες της καθημερινής ζωής στην σύγχρονη εποχή, όπως ψηφιακούς βοηθούς, τηλεχειριστήρια τηλεόρασης με δυνατότητα φωνής και ανίχνευση απάτης με πιστωτικές κάρτες καθώς και από αναδυόμενες τεχνολογίες, όπως τα αυτοκίνητα αυτόνομης οδήγησης.

4.1 Artificial Neuron Networks

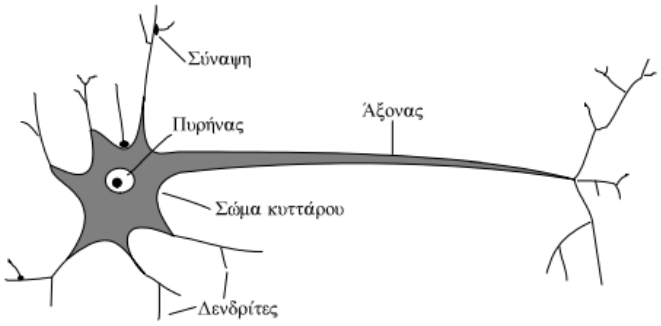
Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ), θεωρείται μία από τις πιο επιτυχημένες τεχνολογίες τα τελευταία 20 χρόνια που έχει χρησιμοποιηθεί ευρέως σε διάφορους τομείς σε μεγάλη ποικιλία σύγχρονων εφαρμογών. Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ) είναι ένα μαθηματικό μοντέλο, το οποίο προσπαθεί να προσομοιώσει τη δομή και τη λειτουργικότητα των βιολογικών νευρωνικών δικτύων του ανθρώπινου εγκεφάλου. Το Τεχνητό Νευρωνικό Δίκτυο (ΤΝΔ) αποτελείται από έναν αριθμό τεχνητών νευρώνων, όπου κάθε ένας από αυτούς τους νευρώνες συνδέεται με τον άλλο μέσω βαρών ή συνοπτικών βαρών. Η μαθησιακή ικανότητα ενός τεχνητού νευρώνα επιτυγχάνεται στην προσαρμογή των βαρών σύμφωνα με τον επιλεγμένο αλγόριθμο εκμάθησης.

4.2 Τεχνητός Νευρώνας

Ο Τεχνητός Νευρώνας είναι ένα βασικό δομικό στοιχείο κάθε Τεχνητού Νευρωνικού Δικτύου, όπου ο σχεδιασμός και οι λειτουργίες του προέρχονται από την παρατήρηση και την μελέτη ενός βιολογικού νευρώνα του ανθρώπινου εγκεφάλου.

Οι βιολογικοί νευρώνες δέχονται σήματα μέσω των δενδριτών, το σώμα επεξεργάζεται την πληροφορία, η οποία διοχετεύεται μέσω των αξόνων.

Παρακάτω απεικονίζεται ένας βιολογικός νευρώνας, ο οποίο αποτελείται από τους δενδρίτες, το σώμα κυττάρου, τον πυρήνα, την σύναψη και τον άξονα.



Εικόνα 10: Βιολογικός Νευρώνας

Από την άλλη πλευρά ο τεχνητός νευρώνας είναι ένα μαθηματικό μοντέλο που λειτουργεί με τον τρόπο που περιγράφεται παρακάτω. Αρχικά ένας τεχνητός νευρώνας δέχεται κάποια σήματα εισόδου $x_1, x_2, x_3, \dots, x_n$ (όπως οι δενδρίτες), όπου κάθε ένα από αυτά τα σήματα μεταβάλλεται από μία αντίστοιχη τιμή βάρους w_i(όπως οι συνάψεις). Η τιμή αυτή του βάρους μπορεί να είναι θετική ή και αρνητική ανάλογα με το αν η λειτουργία της σύναψης επιτυγχάνεται ή επιβραδύνεται αντίστοιχα.

Το σώμα του τεχνητού νευρώνα αρχικά αθροίζει όλα τα σήματα εισόδου που έχουν μεταβληθεί σύμφωνα με τα βάρη τους καθώς και προσθέτει και την πόλωση (bias). Στη συνέχεια το άθροισμα αυτό το φιλτράρει το δεύτερο μέρος του σώματος του νευρώνα, όπου είναι μια μαθηματική συνάρτηση, η συνάρτηση ενεργοποίησης όπου διαμορφώνει την τελική τιμή της εξόδου y.

$$y = F(\sum_{i=0}^{n} x_i * w_i + b)$$

4.2.1 Συνάρτηση Ενεργοποίησης

Μια συνάρτηση ενεργοποίησης αποφασίζει πότε ένας νευρώνας πρέπει να ενεργοποιηθεί και πότε όχι. Γεγονός το οποίο σημαίνει ότι η συνάρτηση ενεργοποίησης θα αποφασίζει πότε η είσοδος του νευρώνα είναι σημαντική ή όχι στη διαδικασία πρόβλεψης χρησιμοποιώντας απλούστερες μαθηματικές πράξεις.

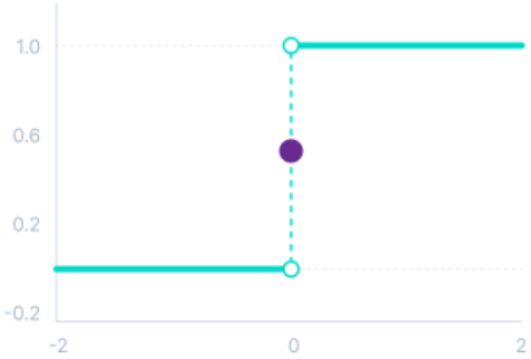
Η αναγκαιότητα της ύπαρξης συναρτήσεων ενεργοποίησης στα Τεχνητά Νευρωνικά Δίκτυα είναι για να προσθέτουν την μη-γραμμικότητα σε αυτά. Ας υποθέσουμε ότι έχουμε ένα Τεχνητό Νευρωνικό Δίκτυο χωρίς τη συνάρτηση ενεργοποίησης. Στην περίπτωση αυτή κάθε νευρώνας που αποτελεί το Τεχνητό Νευρωνικό Δίκτυο θα χρησιμοποιεί μόνο ένα γραμμικό μετασχηματισμό στις εισόδους χρησιμοποιώντας τα βάρη (weights) και τις πολώσεις (biases). Κατά τον τρόπο αυτό όλα τα στρώματα του Τενητού Νευρωνικού Δικτύου θα συμπεριφέρονται με τον ίδιο τρόπο, καθώς η σύνθεση δύο γραμμικών συναρτήσεων αποτελεί το ίδιο με μία ίδια γραμμική

συνάρτηση, οπότε δεν θα έχει σημασία πόσα κρυμμένα επίπεδα έχει το Τεχνητό Νευρωνικό Δίκτυο. Από την μία πλευρά, το νευρωνικό δίκτυο γίνεται πιο απλό, αλλά από την άλλη πλευρά η εκμάθηση οποιασδήποτε περίπλοκης εργασίας γίνεται αδύνατη και το μοντέλο μας καταλήγει να γίνεται ένα μοντέλο γραμμικής παλινδρόμησης.

4.2.1.1 Συνάρτηση του Κατωφλιού(Binary Step Function)

Η Συνάρτηση Ενεργοποίησης του Κατωφλιού βασίζεται κυρίως στην τιμή του κατωφλιού, η οποία είναι και υπεύθυνη για το αν ο νευρώνας πρέπει να ενεργοποιηθεί ή να παραμείνει ανενεργός. Η είσοδος $\sum_{i=0}^{n} x_i * w_i$, με την οποία τροφοδοτείται η συνάρτηση ενεργοποίησης, συγκρίνεται με την τιμή ενός συγκεκριμένου κατωφλιού (threshold). Αν η τιμή της εισόδου είναι μεγαλύτερη ή ίση από την τιμή του κατωφλιού τότε ο νευρώνας ενεργοποιείται, σε κάθε άλλη περίπτωση ο νευρώνας δεν ενεργοποιείται, που σημαίνει ότι η έξοδος δεν περνά στο επόμενο κρυφό επίπεδο (hidden layer). Χρησιμοποιείται συνήθως όταν τα δεδομένα χωρίζονται σε δύο κλάσεις, που σημαίνει πως δεν θα ήταν ιδανική για προβλήματα όπου αποτελούνται από πολλές κλάσεις. Παρακάτω παρουσιάζεται και ο μαθηματικός τύπος καθώς και η γραφική παράσταση της συνάρτησης:

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$$

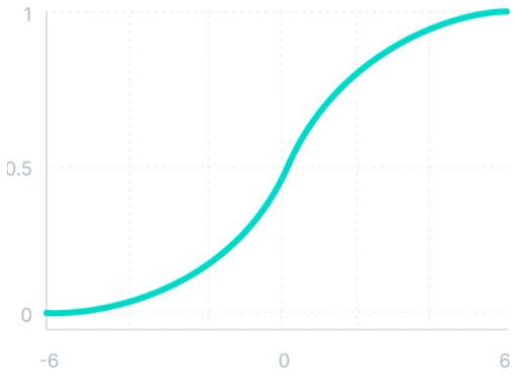


Εικόνα 11: Γραφική Αναπαράσταση Συνάρτησης Κατωφλιού

4.2.1.2 Σιγμοειδής Συνάρτηση Ενεργοποίησης

Η Σιγμοειδής Συνάρτηση Ενεργοποίησης δέχεται οποιαδήποτε είσοδο και έχει ως έξοδο τιμές από 0 έως 1. Όσο μεγαλύτερη η τιμή της εισόδου τόσο πιο κοντά στο 1 θα είναι η τιμή της εξόδου, αντίστοιχα όσο μικρότερη είναι η τιμή της εισόδου τόσο πιο κοντά στο 0 θα είναι η τιμή της εξόδου. Συνήθως χρησιμοποιείται για μοντέλα που χρειάζεται να προβλέψουμε την πιθανότητα σαν έξοδο, μιας και οι τιμές μιας πιθανότητας κυμαίνονται από 0 έως 1, η σιγμοειδής συνάρτηση ενεργοποίησης αποτελεί ιδανική λύση για τέτοιου είδους προβλήματα προσφέροντας το ελάχιστο εύρος τιμών και όσο το δυνατόν μεγαλύτερη ακρίβεια.

$$f(x) = \frac{1}{1 + e^{-x}}$$

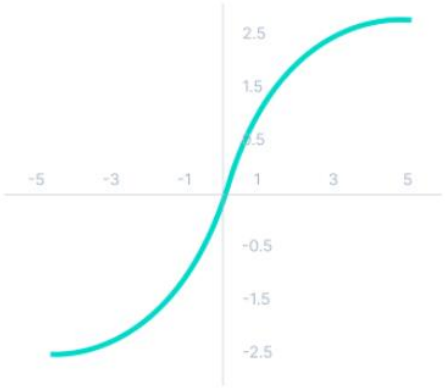


Εικόνα 12: Γραφική Αναπαράσταση Σιγμοειδής Συνάρτησης

4.2.1.3 Υπερβολική Εφαπτομένη

Η Συνάρτηση Ενεργοποίησης Υπερβολικής Εφαπτομένης είναι όμοια με τη Σιγμοειδή Συνάρτηση Ενεργοποίησης, έχουν και οι δύο σιγμοειδές σχήμα όπως άλλωστε και η ονομασία τους. Στο σημείο αυτό είναι σημαντικό να αναφερθεί η διαφορά τους όπου είναι ότι η Συνάρτηση Υπερβολικής Εφαπτομένης έχει σαν έξοδο τιμές που κυμαίνονται από το -1 έως το 1. Επιπλέον, δίνει τη δυνατότητα χαρτογράφησης των τιμών εξόδων ως έντονα αρνητικών, ουδέτερων ή έντονα θετικών. Η μαθηματικής συνάρτηση δίνεται παρακάτω:

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$



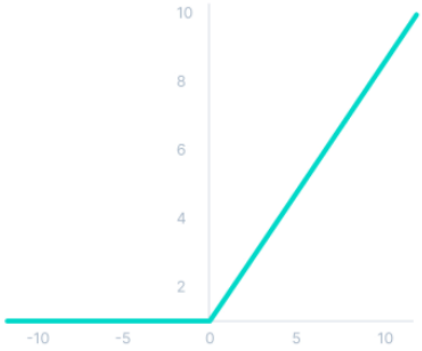
Εικόνα 13: Γραφική Αναπαράσταση Υπερβολικής Εφαπτομένης

4.2.1.4 Rectified Linear Unit-ReLU

Η Συνάρτηση Ενεργοποίησης ReLU είναι από της πιο δημοφιλής συναρτήσεις ενεργοποίησης αυτή τη στιγμή στα Τεχνητά Νευρωνικά Δίκτυα και οι τιμές της κυμαίνονται από 0 έως το άπειρο. Η συνάρτηση ReLU μετατρέπει όλες τις αρνητικές

τιμές σε 0. Το αρνητικό της μετατροπής όλων των αρνητικών τιμών σε 0 άμεσα είναι ότι μειώνεται η ικανότητα του μοντέλου να κάνει fit στα δεδομένα ή να εκπαιδευτεί από αυτά.

$$f(x) = \max(0, x)\,, \text{όπου } x = \sum_{i=0}^{n} x_i * w_i$$

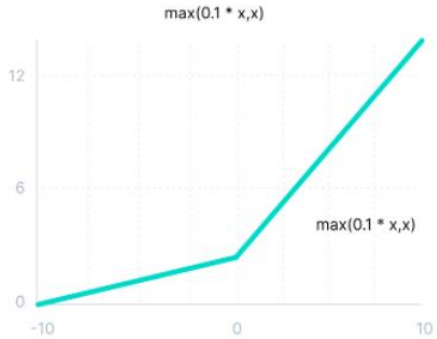


Εικόνα 14: Γραφική Αναπαράσταση Συνάρτησης Rectified Linear Unit-ReLU

4.2.1.5 Leaky ReLU Activation Function

Η Συνάρτηση Ενεργοποίησης Leaky ReLU Activation Function επιδιορθώνει στην ουσία το μειονέκτημα της Συνάρτησης Ενεργοποίησης ReLU χωρίς να μετατρέπει τις αρνητικές εισόδους σε 0 άμεσα, αλλά μετατρέποντάς τις τιμές εισόδου σε τιμές πολύ κοντά στο 0, λύνοντας έτσι το κυριότερο πρόβλημα της Συνάρτησης Ενεργοποίησης ReLU.

$$f(x) = \max(0.1x, x)\,, \text{όπου } x = \sum_{i=0}^{n} x_i * w_i$$



Εικόνα 15: Γραφική Αναπαράσταση Συνάρτησης Leaky Relu Activation Function

4.2.1.6 Softmax

Η Συνάρτηση ενεργοποίησης Softmax λειτουργεί όπως και η Σιγμοειδής Συνάρτηση ενεργοποίησης, δηλαδή χρησιμοποιείται για μοντέλα που θέλουμε να προβλέψουμε την πιθανότητα της κάθε κλάσης. Η τιμές των εξόδων της κυμαίνονται από 0 έως 1 αλλά το άθροισμά τους ισούται πάντα με 1. Όταν τα προβλήματα είναι δυαδικά προτείνεται η χρήση της σιγμοειδούς συνάρτησης ενεργοποίησης , ενώ όταν τα προβλήματα περιέχουν παραπάνω από δύο κλάσεις προτείνεται η χρήση της συνάρτησης ενεργοποίησης Softmax. Παρακάτω ορίζεται μαθηματική της συνάρτηση:

$$softmax(z_i) = \frac{\exp{(z_i)}}{\sum_j \exp{(z_j)}}$$

4.2.2 Συνάρτηση Κόστους

Η συνάρτηση κόστους είναι ένα από τα σημαντικά στοιχεία των Τεχνητών Νευρωνικών Δικτύων. Το κόστος ή η απώλεια είναι ένας σφάλμα της πρόβλεψης των Νευρωνικών Δικτύων και η μέθοδος για τον υπολογισμό του συγκεκριμένου κόστους ή απώλειας ονομάζεται Συνάρτηση Κόστους (Loss Function). Για να μάθει μόνο του ένα Νευρωνικό Δίκτυο αλλάζει, δηλαδή μειώνεται η Συνάρτηση Κόστους, ώστε να μπορέσει να φτάσει όσο πιο κοντά γίνεται στην πραγματική τιμή. Υπάρχουν ποικίλες Συναρτήσεις Κόστους ώστε να υπολογιστεί το σφάλμα με βάση την προβλεπόμενη και πραγματική τιμή ανάλογα με το πρόβλημα που πρόκειται να αντιμετωπιστεί. Σε γενικά πλαίσια οι Συναρτήσεις Κόστους θα μπορούσαν να διαχωριστούν σε δυο μεγάλες κατηγορίες τις Συναρτήσεις Κόστους Ταξινόμησης και της Συναρτήσεις Κόστους Παλινδρόμησης.

Από τις πιο γνωστές Συναρτήσεις Κόστους Ταξινόμησης θεωρείται η Cross Entropy Loss Function. Κάθε προβλεπόμενη πιθανότητα κλάσης συγκρίνεται με την πραγματική επιθυμητή απόδοση κλάσης 0 ή 1 και υπολογίζεται το σφάλμα ή η απώλεια που «τιμωρεί» την πιθανότητα με βάση το πόσο απέχει από την πραγματική αναμενόμενη τιμή. Η «ποινή» είναι λογαριθμικής φύσεως και δίνει μεγάλο αποτέλεσμα για διαφορές κοντά στο 1 και μικρό αποτέλεσμα για διαφορές που τείνουν στο 0. Η μαθηματική εξίσωση της Cross Entropy Loss Function ορίζεται ως εξής:

$$-\sum_{i=1}^{n} t_i \log(p_i)$$

- n ο αριθμός των κλάσεων
- t_i η πραγματική πιθανότητα
- p_i η προβλεπόμενη από το μοντέλο πιθανότητα

Το μέσο τετραγωνικό σφάλμα (MSE) είναι η πιο συχνή σε χρήση συνάρτηση κόστους που χρησιμοποιείται για προβλήματα παλινδρόμησης. Το μέσο τετραγωνικό σφάλμα υπολογίζεται ως ο μέσος όρος των προβλεπόμενων και των πραγματικών τιμών.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (y - \hat{y}_i)^2$$

Όπου $\hat{y}$, η προβλεπόμενη τιμή.

4.2.3 Βελτιστοποιητές στην Βαθιά Μάθηση

Για ένα πρόβλημα Βαθιάς Μάθησης ορίζεται πρώτα μια συνάρτηση απώλειας (loss function). Ύστερα από την επιλογή της συνάρτησης απώλειας (loss function), μπορεί να οριστεί ένας αλγόριθμος ή μία μέθοδος βελτιστοποίησης με στόχο να ελαχιστοποιήσει την απώλεια που έχει δημιουργηθεί. Οι βελτιστοιητές είναι μαθηματικές συναρτήσεις που εξαρτώνται από της μαθησιακές παραμέτρους του μοντέλου π.χ. τα βάρη (weights). Τέλος οι βελτιστοποιητές βοηθούν ώστε να γνωρίζουμε τον τρόπο με τον οποίο αλλάζουν τα βάρη και ο ρυθμός εκμάθησης του νευρωνικού δικτύου, έτσι ώστε να μειωθούν οι απώλειες. Στη συνέχεια παρουσιάζονται κάποιες από τι δημοφιλέστερες τεχνικές βελτιστοποίησης των Τεχνητών Νευρωνικών Δικτύων.

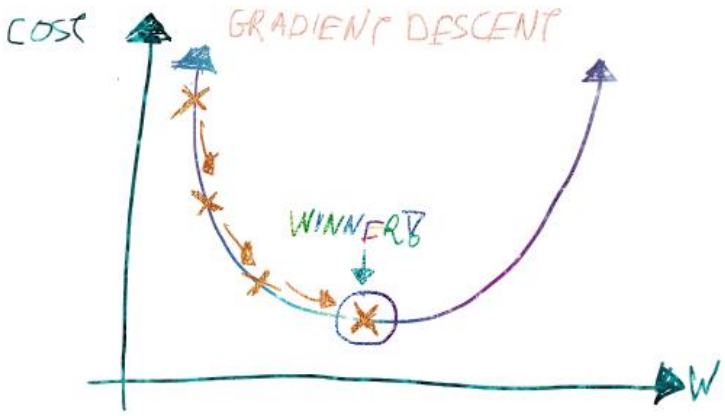
4.2.3.1 Gradient descent

Ο αλγόριθμος της Επικλινούς Καθόδου είναι ένας αλγόριθμος βελτιστοποίησης που χρησιμοποιείται για την ελαχιστοποίηση κάποιας συνάρτησης (σε αυτή την περίπτωση της συνάρτησης κόστους), με επαναληπτική κίνηση προς την κατεύθυνση της «απότομης κατάβασης», όπως ορίζεται από το αρνητικό της κλίσης. Στη Μηχανική Μάθηση χρησιμοποιούμε Gradient Descent για να ενημερώσουμε τις παραμέτρους του μοντέλου μας. Οι παράμετροι αυτοί αντιστοιχούν στα coefficients για την Γραμμική Παλινδρόμηση (Linear Regression) και στα βάρη (weights) για τα Νευρωνικά Δίκτυα.

Ας υποθέσουμε ότι βρισκόμαστε στην κορυφή ενός βουνού και στόχος μας είναι να μετακινηθούμε από την υψηλότερη κορυφή του βουνού (υψηλό κόστος) κάτω στο επίπεδο της λίμνης (χαμηλό κόστος). Τα βέλη αντιπροσωπεύουν την κατεύθυνση της πιο απότομης καθόδου (αρνητική κλίση) από οποιοδήποτε σημείο που μπορεί να δοθεί. Η κατεύθυνση αυτή που μειώνει τη συνάρτηση κόστους όσο το δυνατόν γρηγορότερα.

Εικόνα 16: Παράδειγμα Gradient Descent

Ξεκινώντας από την κορυφή του βουνού, κάνουμε το πρώτο μας βήμα κατηφορικά προς την κατεύθυνση που ορίζει η αρνητική κλίση . Στη συνέχεια ορίζουμε εκ νέου την αρνητική κλίση (έχοντας περάσει στις συντεταγμένες του νέου μας σημείου) και κάνουμε κι άλλο ένα βήμα προς την κατεύθυνση που ορίζει. Συνεχίζουμε αυτή τη διαδικασία επαναληπτικά μέχρι να φτάσουμε στο κάτω μέρος του γραφήματος ή σε κάποιο άλλο σημείο από το οποίο δεν θα μπορούμε να κινηθούμε κατηφορικά πλέον, δηλαδή ένα τοπικό ελάχιστο.



Εικόνα 17: Γραφική Απεικόνιση Gradient Descent

4.2.3.2 Batch Gradient Descent

Ο Batch Gradient Descent αθροίζει το σφάλμα για κάθε σημείο σε ένα σετ εκπαίδευσης, ενημερώνοντας το μοντέλο αφού πρώτα αξιολογηθούν όλα τα παραδείγματα εκπαίδευσης. Η διαδικασία αυτή αναφέρεται ως και μια εποχή (epoch) εκπαίδευσης.

Αν και αυτή η ομαδοποίηση παρέχει υπολογιστική απόδοση, μπορεί να έχει μεγάλο χρόνο επεξεργασίας για μεγάλα σύνολα δεδομένων εκπαίδευσης, καθώς χρειάζεται να αποθηκεύει όλα τα δεδομένα στη μνήμη. Το Batch Gradient Descent συνήθως παράγει μια σταθερή κλίση σφάλματος και σύγκλιση, αλλά μερικές φορές αυτό το σημείο σύγκλισης δεν είναι το ιδανικότερο, βρίσκοντας το τοπικό ελάχιστο έναντι του κανονικού.

4.2.3.3 Stochastic Gradient Descent

Στο Stochastic gradient descent (SGD) εκτελείται μια περίοδος εκπαίδευσης (training epoch) για κάθε παράδειγμα ξεχωριστά μέσα σε όλο το σύνολο των δεδομένων και οι παράμετροι κάθε παραδείγματος εκπαίδευσης ενημερώνονται μία κάθε φορά. Γνωρίζοντας ότι πρέπει να κρατηθεί μόνο ένα παράδειγμα εκπαίδευσης, γίνεται πιο εύκολη η αποθήκευσή του στη μνήμη. Ενώ αυτές οι συχνές ενημερώσεις προσφέρουν πολύ περισσότερες λεπτομέρειες και πολύ καλύτερη απόδοση ταχυτήτων, υπάρχει

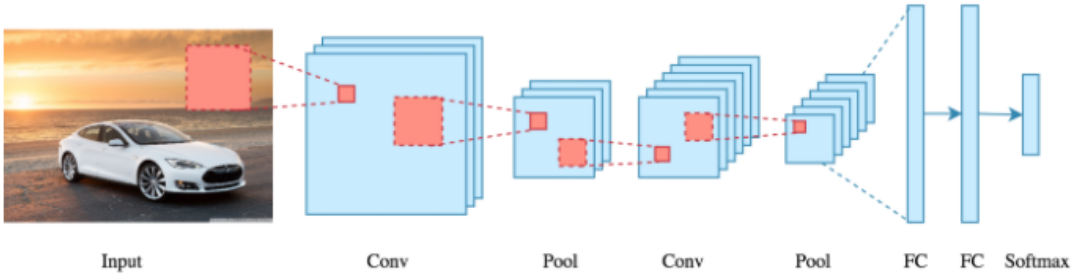
το ενδεχόμενο να οδηγήσουν σε απώλειες στην υπολογιστική απόδοση σε σύγκριση με την υπολογιστική απόδοση του Batch Gradient Descent. Τέλος οι συχνές ενημερώσεις του Stochastic gradient descent (SGD) μπορεί να οδηγήσουν σε θορυβώδεις κλίσεις (noisy gradients), αλλά αυτό μπορεί και να φανεί χρήσιμο για την απόδραση του τοπικού ελαχίστου και την εύρεση του καθολικού.

Mini-Batch Gradient Descent

Ο mini-batch gradient descent αλγόριθμος είναι ένας συνδυασμός εννοιών του Gradient Descent αλγορίθμου και του Batch Gradient Descent αλγορίθμου. Η λογική του mini-batch gradient descent είναι να διαχωρίζει το σύνολο δεδομένων εκπαίδευσης σε μικρά μεγέθη (mini batches) και εκτελεί ενημερώσεις σε κάθε ένα από αυτά ξεχωριστά. Η συγκεκριμένη προσέγγιση επιτυγχάνει μια ισορροπία μεταξύ της υπολογιστικής αποτελεσματικότητας του Batch Gradient Descent και της ταχύτητας του Stochastic gradient descent (SGD).

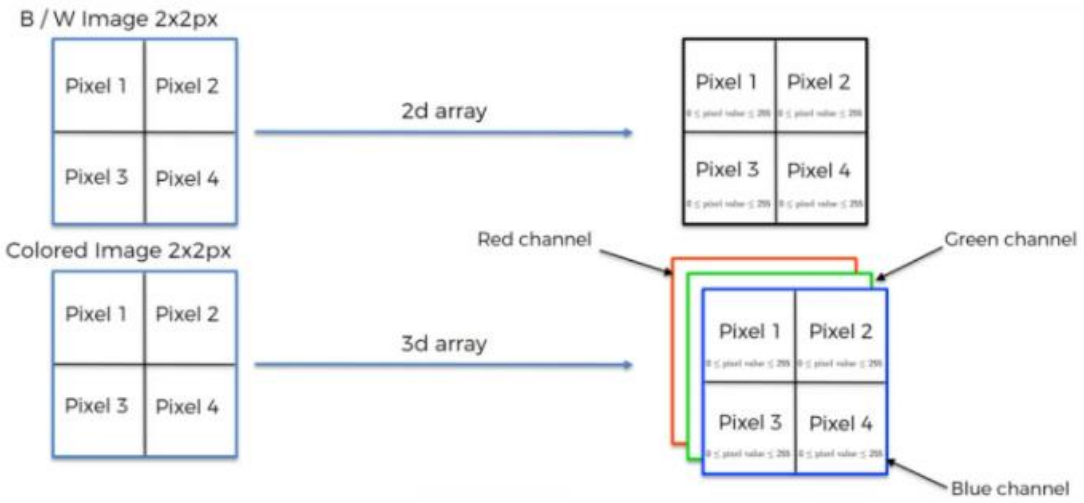
5.Συνελικτικά Νευρωνικά Δίκτυα (CNN)

Μέχρι τώρα γνωρίζαμε για τη Μηχανική Μάθηση, τον κλάδο της επιστήμης των υπολογιστών όπου μελετούν και αναλύουν τους τύπους των αλγορίθμων χρησιμοποιώντας ελαστικά μαθηματικά μοντέλα που μπορούν να μάθουν από τα εισερχόμενα δεδομένα και να κάνουν προγνώσεις πάνω σε άγνωστα δεδομένα. Όμως τώρα γνωρίζουμε τη Βαθιά Μηχανική Μάθηση (Deep Learning) η οποία αποτελεί ένα πεδίο της Μηχανικής Μάθησης, το οποίο εμπνέεται από τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks). Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) μοιάζουν σε μεγάλο βαθμό με τα απλά Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks) που έχουν μαθησιακά βάρη (weights and biases). Τα Συνελικτικά Νευρωνικά Δίκτυα (CNN) χρησιμοποιούνται ευρέως στη σημερινή εποχή σε εφαρμογές για την αναγνώριση εικόνων, την ταξινόμηση εικόνων, την ανίχνευση αντικειμένων, την αναγνώριση προσώπων κ.λ.π.



Εικόνα 18: Αρχιτεκτονική Συνελικτικού Δικτύου

Όπως όλοι γνωρίζουμε μια εικόνα είναι ένας πίνακας από εικονοστοιχεία (pixel). Στην περίπτωση που έχουμε μία ασπρόμαυρη εικόνα, τότε έχουμε ένα δισδιάστατο πίνακα. Όμως στην περίπτωση όπου η εικόνα μας δεν είναι ασπρόμαυρη αλλά έγχρωμη, τότε θα έχουμε ένα τρισδιάστατο πίνακα, που σημαίνει ότι υπάρχει μια επιπλέον παράμετρος στην περίπτωση αυτή, το βάθος, όπου στην ουσία είναι το κανάλι RGB, το οποίο αποτελείται από τα 3 επίπεδα των χρωμάτων Κόκκινο-Πράσινο- Μπλε. Όπως βλέπουμε παρακάτω (Εικόνα 19) μια ασπρόμαυρη εικόνα ή φωτογραφία αποτελείται μόνο από ένα επίπεδο, ενώ οι έγχρωμες εικόνες αποτελούνται από τρία επίπεδα όπου το κάθε ένα αντιστοιχεί στα ένα από τα χρώματα Κόκκινο-Πράσινο-Μπλε. Οι τιμές των εικονοστοιχείων (pixels) κυμαίνονται από 0 έως και 255.



Εικόνα 19: Μετατροπή Εικόνων σε Πίνακες

Η αρχιτεκτονική των Συνελικτικών Νευρωνικών Δικτύων (CNN) αποτελείται από έναν αριθμό από επίπεδα. Κάθε ένα από αυτά τα επίπεδα παρουσιάζονται παρακάτω αναλυτικά:

5.1 Επίπεδο Συνέλιξης

Η Συνέλιξη είναι μία μαθηματική πράξη δύο συναρτήσεων (f και g) που στοχεύει στην κατασκευή μίας τρίτης συνάρτησης, η οποία αναπαριστά πώς το σχήμα της μίας συνάρτησης τροποποιείται από την άλλη. Η μαθηματική πράξη της συνέλιξης απεικονίζεται παρακάτω:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d_\tau$$

Η συνέλιξη είναι το πρώτο επίπεδο που εξάγει χαρακτηριστικά από μια εικόνα εισόδου. Το Συνελικτικό επίπεδο είναι αυτό που διατηρεί τη σχέση μεταξύ των εικονοστοιχείων μαθαίνοντας τα χαρακτηριστικά της εικόνας, χρησιμοποιώντας μικρά τετράγωνα πάνω στις εικόνες που χρησιμοποιούνται σαν δεδομένα εισόδου. Είναι μια μαθηματική πράξη που έχει δύο εισόδους, όπως έναν πίνακα της εικόνας (image matrix) και ένα φίλτρο (filter) ή πυρήνα (kernel). Η Συνέλιξη μιας εικόνας με διαφορετικά φίλτρα μπορεί να εκτελέσει λειτουργίες όπως την ανίχνευση διάφορων γωνιών (edge detection), θόλωση (blur) και ευκρίνεια (sharpen) με την εφαρμογή φίλτρων. Το παρακάτω παράδειγμα δείχνει διάφορες εικόνες συνέλιξης μετά την εφαρμογή διαφορετικών τύπων φίλτρων.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Εικόνα 20: Φίλτρα Συνέλιξης

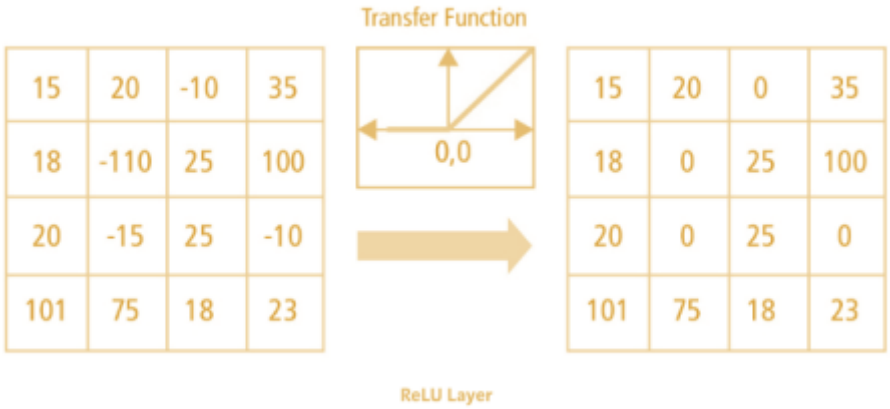
5.2 Συνάρτηση ενεργοποίησης(Μη γραμμικότητα)

Η διαδικασία αντιστοίχισης της εισόδου στην έξοδο είναι μία από τις βασικότερες λειτουργίες όλων των Συναρτήσεων ενεργοποίησης σε όλους τους τύπους Νευρωνικών Δικτύων. Για τον υπολογισμό της τιμής εισόδου για τον κάθε νευρώνα χρησιμοποιούνται τα ίδια βάρη (weights) και η ίδια κλίση (bias), αν αυτή υπάρχει. Η συνάρτηση ενεργοποίησης είναι αυτή που αποφασίζει εάν θα τροφοδοτήσει ή όχι έναν νευρώνα με μία αναφορά σε μία συγκεκριμένη είσοδο, δημιουργώντας την αντίστοιχη έξοδο.

Τα μη γραμμικά επίπεδα ενεργοποίησης χρησιμοποιούνται μετά από όλα τα επίπεδα με βάρη (weights), για το λόγο αυτό και αποκαλούνται επίπεδα εκμάθησης όπως τα επίπεδα FC και τα συνελικτικά επίπεδα, στην αρχιτεκτονική του CNN. Αυτή η μη γραμμική απόδοση των επιπέδων ενεργοποίησης σημαίνει ότι η αντιστοίχιση της εισόδου στην έξοδο θα είναι μη γραμμική. Επίσης, αυτά τα επίπεδα δίνουν την

δυνατότητα στο CNN να μαθαίνει εξαιρετικά περίπλοκα πράγματα. Επιπλέον, η συνάρτηση ενεργοποίησης πρέπει να κατέχει τη δυνατότητα διαφοροποίησης, χαρακτηριστικό το οποίο είναι εξαιρετικά σημαντικό, καθώς επιτρέπει τη χρήση της αντίστροφης διάδοσης (οπισθοδιάδοσης) σφαλμάτων για την εκπαίδευση του δικτύου. Η πιο διαδεδομένη συνάρτηση ενεργοποίησης που χρησιμοποιείται συνήθως στα CNN είναι η συνάρτηση ενεργοποίησης Relu. Η Relu συνάρτηση ενεργοποίησης μετατρέπει όλες τις τιμές της εισόδου σε θετικούς αριθμούς. Το χαμηλό υπολογιστικό φορτίο είναι το βασικό πλεονέκτημα της συνάρτησης Relu έναντι των υπολοίπων συναρτήσεων ενεργοποίησης. Κάποιες φορές ενδέχεται να προκύψουν μερικά σημαντικά ζητήματα κατά την χρήση της συνάρτησης Relu. Η μαθηματική της συνάρτηση παρουσιάζεται παρακάτω:

$$f(x)_{Relu} = \max(0, x)$$

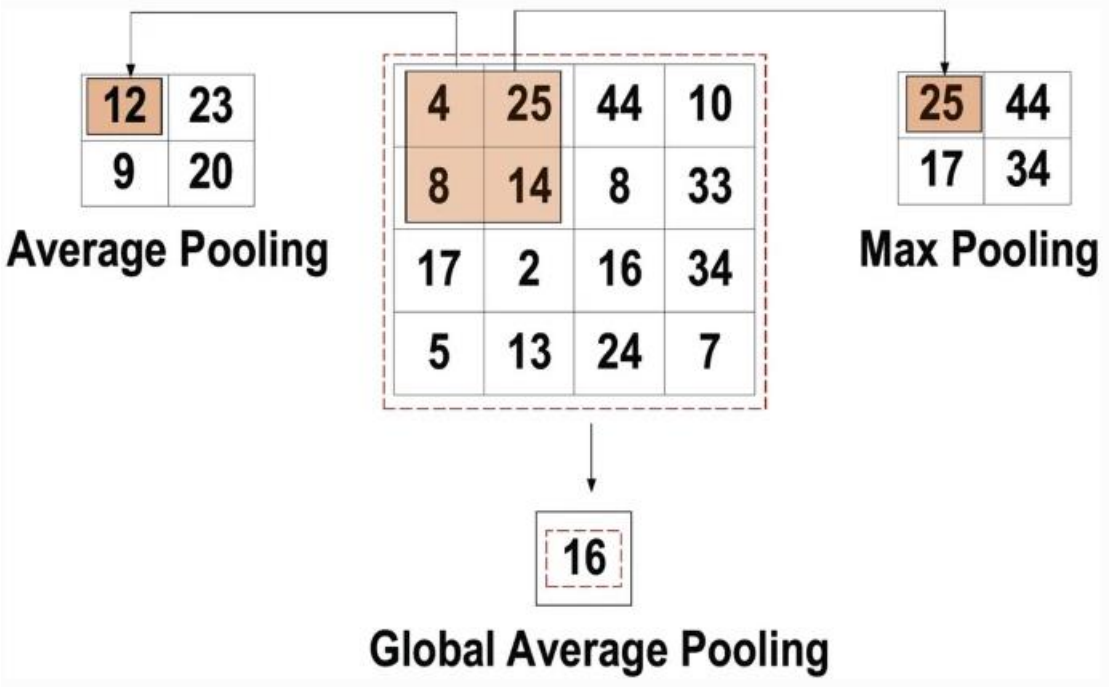


Εικόνα 21: Relu operation

5.3 Επίπεδο Συγκεντρωσης (Pooling)

Τα επίπεδα συγκέντρωσης προστίθενται μεταξύ δύο επιπέδων συνέλιξης με μοναδικό σκοπό τη μείωση του χωρικού μεγέθους της αναπαράστασης της εικόνας. Η κύρια εργασία του επιπέδου συγκέντρωσης είναι η υπό δειγματοληψία του χάρτη των χαρακτηριστικών (feature maps). Αυτοί οι χάρτες δημιουργούνται ακολουθώντας τις λειτουργίες της συνέλιξης. Με άλλα λόγια, αυτή η προσέγγιση συρρικνώνει τους χάρτες χαρακτηριστικών μεγάλου μεγέθους για να δημιουργήσει μικρότερους χάρτες χαρακτηριστικών. Παράλληλα, διατηρεί την πλειοψηφία των κυρίαρχων πληροφοριών (ή χαρακτηριστικών) σε κάθε βήμα του σταδίου συγκέντρωσης. Υπάρχουν διάφοροι τύποι επιπέδων συγκέντρωσης, όπως η Συγκέντρωση των Δέντρων (Tree Pooling), η Περιφραγμένη Συγκέντρωση (Gated Pooling), η Μέση Συγκέντρωση (Average Pooling), η Ελάχιστη Συγκέντρωση (Min Pooling), η Μέγιστη

Συγκέντρωση (Max Pooling), η Συνολική Μέση Συγκέντρωση (Global Average Pooling ή και GAP) και η Συνολική Μέγιστη Συγκέντρωση (Global Max Pooling). Οι πιο γνωστές και οι πιο διαδομένες στη χρήση μέθοδοι συγκέντρωσης είναι η Ελάχιστη Συγκέντρωση (Min Pooling), η Μέγιστη Συγκέντρωση (Max Pooling) και η Συνολική Μέση Συγκέντρωση (Global Average Pooling ή και GAP). Παρακάτω απεικονίζονται οι τρεις αυτές μέθοδοι συγκέντρωσης.



Εικόνα 22: Μέθοδοι Συνέλιξης

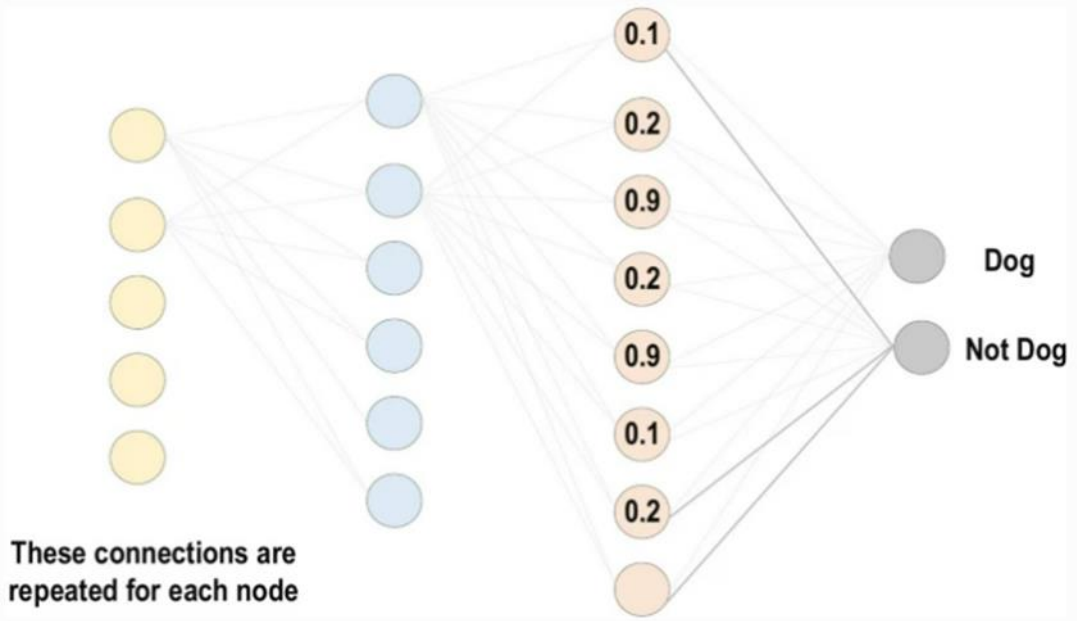
5.4 Κανονικοποίηση (Normalization Layer)

Τα επίπεδα κανονικοποίησης όπως λέει και το όνομά τους, κανονικοποιούν την έξοδο των προηγούμενων επιπέδων. Προστίθεται μεταξύ των επιπέδων συνέλιξης και συγκέντρωσης, επιτρέποντας σε κάθε επίπεδου του δικτύου να μαθαίνει περισσότερα ανεξάρτητα και να αποφεύγει την υπερβολική προσαρμογή του μοντέλου (overfitting).

5.5 Πλήρως Συνδεδεμένο Δίκτυο (Fully Connected Layer)

Πλήρως Συνδεδεμένο Δίκτυο (Fully Connected Layer). Συνήθως αυτό το επίπεδο βρίσκεται στο τέλος κάθε αρχιτεκτονικής CNN. Μέσα σε αυτό το επίπεδο, κάθε νευρώνας συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου, τη λεγόμενη πλήρως συνδεδεμένη προσέγγιση (Fully Connected). Χρησιμοποιείται και ως ταξινομητής των CNN. Ακολουθεί τη βασική μέθοδο του συμβατικού νευρωνικού δικτύου πολλαπλών επιπέδων perceptron (multiple-layer perceptron neural network), καθώς είναι ένας τύπος τροφοδοσίας των Τεχνητών Νευρωνικών Δικτύων (ANN). Η είσοδος του επιπέδου FC προέρχεται από το τελευταίο επίπεδο συγκέντρωσης ή συνέλιξης. Αυτή η είσοδος έχει τη μορφή ενός διανύσματος, το οποίο δημιουργείται από τα feature maps μετά την κανονικοποίηση (flattering). Η

έξοδος του επιπέδου FC αντιπροσωπεύει την τελική έξοδο CNN, όπως απεικονίζεται στην παρακάτω εικόνα.



Εικόνα 23: Fully connected layer

5.6 Παράδειγμα

Στο παράδειγμα αυτό θα γίνει η εκπαίδευση ενός συνελικτικού δικτύου, έτσι ώστε να ανιχνεύει σε μία ακτινογραφία αν ο ασθενή έχει νοσήσει από πνευμονία ή όχι. Για την παρούσα εργασία έχουμε χρησιμοποιήσει το σύνολο δεδομένων με ακτινογραφίες θώρακος του Kaggle, chest-xray-pneumonia. Το συγκεκριμένο dataset είναι οργανωμένο σε 3 φακέλους τους train, test και val και ο κάθε ένας από αυτούς εμπεριέχει από άλλους δύο υποφακέλους τον NORMAL και τον PNEUMONIA. Ο φάκελος NORMAL που βρίσκεται στον train, test και val εμπεριέχει ακτινογραφίες θώρακα όπου ο ασθενής είναι απόλυτα υγιής και δεν έχει προσβληθεί από την νόσο της πνευμονίας, ενώ στον φάκελο PNEUMONIA που επίσης περιλαμβάνεται στους φακέλους train, test και val ισχύει το ακριβώς αντίθετο, εμπεριέχει ακτινογραφίες ασθενών που έχουν προσβληθεί από πνευμονία.

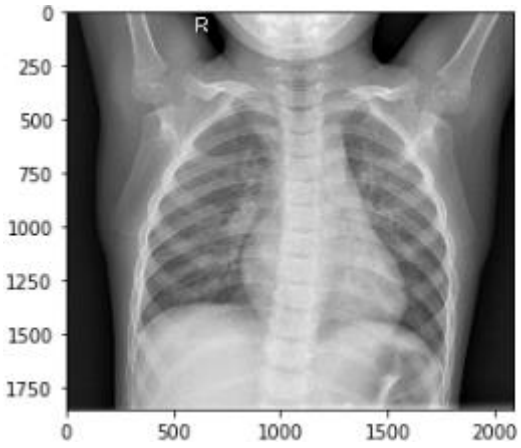
Η ανίχνευση πνευμονίας με σάρωση ακτινογραφιών θώρακα είναι ένα απλό πρόβλημα δυαδικής ταξινόμησης (binary classification). Δηλαδή είτε θα ανιχνεύσουμε την πνευμονία σε μια ακτινογραφία είτε όχι. Έτσι λοιπόν θα αντιστοιχίσουμε τις ακτινογραφίες των ασθενών χωρίς πνευμονία με τον αριθμό 0 και τις ακτινογραφίες των ασθενών με πνευμονία με τον αριθμό 1.

Ο φάκελος NORMAL που εμπεριέχεται στον φάκελο train περιλαμβάνεται από 1341 ακτινογραφίες όπου ο ασθενής είναι υγιής. Ενώ ο φάκελος PNEUMONIA που

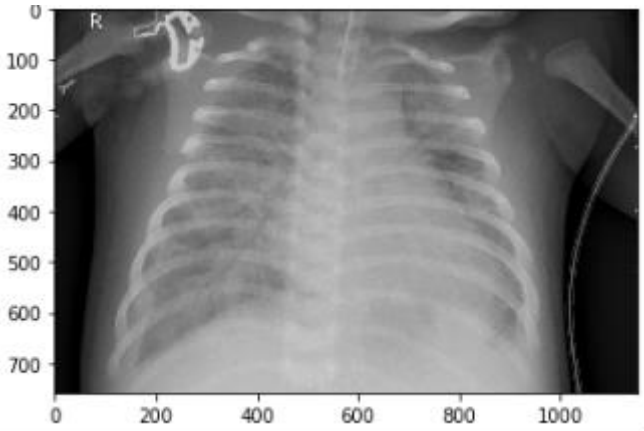
εμπεριέχεται στον φάκελο train περιλαμβάνεται από 3875 ακτινογραφίες όπου ο ασθενής νοσεί από πνευμονία.

Ο φάκελος NORMAL που εμπεριέχεται στον φάκελο val περιλαμβάνεται από 8 ακτινογραφίες όπου ο ασθενής είναι υγιής και ο φάκελος PNEUMONIA που εμπεριέχεται στον φάκελο val περιλαμβάνεται από 8 ακτινογραφίες όπου ο ασθενής νοσεί από πνευμονία.

Ο φάκελος NORMAL που εμπεριέχεται στον φάκελο test περιλαμβάνεται από 234 ακτινογραφίες όπου ο ασθενής είναι υγιής και ο φάκελος PNEUMONIA που εμπεριέχεται στον φάκελο test περιλαμβάνεται από 390 ακτινογραφίες όπου ο ασθενής νοσεί από πνευμονία.



Εικόνα 24: Ακτινογραφία χωρίς πνευμονία(NORMAL:0)



Εικόνα 25: Ακτινογραφία με πνευμονία(PNEUMONIA:1)

6. Object Detection

Η ανίχνευση αντικειμένων είναι μία τεχνική όρασης του υπολογιστή που λειτουργεί για την αναγνώριση και τον εντοπισμό αντικειμένων μέσα σε μία εικόνα ή μέσα σε ένα βίντεο. Πιο αναλυτικά, η ανίχνευση αντικειμένων σχεδιάζει οριοθετημένα πλαίσια (bounding boxes) γύρω από τα αντικείμενα που πρέπει να ανιχνευθούν, τα οποία μας επιτρέπουν να εντοπίσουμε που βρίσκονται τα εν λόγω αντικείμενα μέσα στην εικόνα ή στο βίντεο. Πολλές φορές η ανίχνευση αντικειμένων συνήθως συγχέεται με την αναγνώριση εικόνας, οπότε στο σημείο αυτό θα ήταν σημαντικό να διευκρινιστούν οι διακρίσεις μεταξύ αυτών των δύο εννοιών. Η αναγνώριση εικόνας εκχωρεί κάθε φορά μια ετικέτα σε μία εικόνα, δηλαδή μια φωτογραφία ενός σκύλου λαμβάνει την ετικέτα «σκύλος». Από την άλλη πλευρά, η ανίχνευση αντικειμένων σχεδιάζει ένα κουτί γύρω από κάθε σκύλο και χαρακτηρίζει το κουτί ως «σκύλος». Επίσης, το μοντέλο προβλέπει που βρίσκεται κάθε αντικείμενο μέσα στην εικόνα ή στο βίντεο και ποια ετικέτα πρέπει να εφαρμοστεί. Με τον τρόπο που αναφέρθηκε προηγουμένως, η ανίχνευση αντικειμένου (object detection) παρέχει αρκετά περισσότερες πληροφορίες για μια εικόνα σε σχέση με την αναγνώριση εικόνας (image recognition).

Σε γενικές γραμμές, η ανίχνευση αντικειμένων μπορεί να χωριστεί σε δύο κατηγορίες. Η πρώτη κατηγορία της ανίχνευσης αντικειμένων είναι αυτή που βασίζεται στη Μηχανική Μάθηση (Machine Learning) και η άλλη κατηγορία είναι αυτή που βασίζεται στη Βαθιά Μάθηση (Deep Learning).

Στις πιο διαδεδομένες προσεγγίσεις που βασίζονται στη Μηχανική Μάθηση (Machine Learning), οι τεχνικές όρασης υπολογιστή χρησιμοποιούνται για την εξέταση διαφόρων χαρακτηριστικών μιας εικόνας, όπως το έγχρωμο ιστόγραμμα η τις πτυχές μιας εικόνας, για τον εντοπισμό ομάδων από εικονοστοιχεία (pixels) που μπορεί να ανήκουν σε ένα αντικείμενο. Τα χαρακτηριστικά αυτά στη συνέχεια τροφοδοτούνται σε ένα μοντέλο παλινδρόμησης που προβλέπει τη θέση του αντικειμένου μαζί με την ετικέτα του.

Από την άλλη πλευρά, οι προσεγγίσεις που βασίζονται σε Βαθιά Μάθηση (Deep Learning), χρησιμοποιούν Συνελικτικά Νευρωνικά Δίκτυα (CNN) για την εκτέλεση ανίχνευσης αντικειμένων από άκρο σε άκρο, χωρίς επίβλεψη.

Επειδή οι μέθοδοι της Βαθιάς Μάθησης (Deep Learning) έχουν γίνει οι πιο σύγχρονες και διαδεδομένες προσεγγίσεις για την ανίχνευση αντικειμένων στη σύγχρονη εποχή, αυτές είναι και οι τεχνικές στις οποίες θα επικεντρωθούμε για τους σκοπούς της παρούσας εργασίας.

Όπως αναφέρθηκε και προηγουμένως η ανίχνευση αντικειμένων (Object Detection) είναι συνδεδεμένη με άλλες παρόμοιες τεχνικές όρασης του υπολογιστή, όπως η αναγνώριση εικόνας (image recognition) και η κατάτμηση εικόνας (image segmentation), καθώς μας βοηθά να κατανοήσουμε και να αναλύσουμε σκηνές σε εικόνες και βίντεο.

Υπάρχουν όμως σημαντικές διαφορές αφού η αναγνώριση εικόνας εξάγει μόνο μία ετικέτα κλάσης για ένα αναγνωρισμένο αντικείμενο και η τμηματοποίηση εικόνας δημιουργεί μια κατανόηση σε επίπεδο εικονοστοιχείων (pixels) των στοιχείων μιας σκηνής. Αυτό που διαχωρίζει τον εντοπισμό αντικειμένων από αυτές τις δύο εργασίες είναι η μοναδική του ικανότητα να εντοπίζει αντικείμενα μέσα σε μια εικόνα ή ένα βίντεο. Γεγονός το οποίο μας επιτρέπει να μετράμε και στη συνέχεια να παρακολουθούμε αυτά τα αντικείμενα και κατά προέκταση να αντλούμε πληροφορίες για αυτά.

Δεδομένων αυτών των βασικών διακρίσεων και των μοναδικών δυνατοτήτων ανίχνευσης αντικειμένων, μπορούμε να δούμε πως μπορεί να εφαρμοστεί με διάφορους τρόπους, σε διάφορους τομείς τις καθημερινής ζωής στην σύγχρονη εποχή:

- Καταμέτρηση πλήθους (Crow Counting)
- Αυτό-οδηγούμενα αυτοκίνητα (Self-driving cars)
- Παρακολούθηση βίντεο (Video Surveillance)
- Ανίχνευση προσώπων (Face Detection)
- Ανίχνευση ανωμαλιών (Anomaly Detection)

6.1 YOLO αλγόριθμος

Ο YOLO, όπου είναι η συντομογραφία του όρου (You Only Look Once), είναι ένας αλγόριθμος που ανιχνεύει και αναγνωρίζει διάφορα αντικείμενα σε μία ή και πολλές εικόνες σε πραγματικό χρόνο. Ο αλγόριθμος YOLO έχει υλοποιηθεί από τον δημιουργό του Joseph Redmond που παρουσιάστηκε και δημοσιεύτηκε στο συνέδριο IEEE/CVF για την οπτική του υπολογιστή (Computer Vision) και την αναγνώριση προτύπων (Pattern Recognition) (CVPR) ως έγγραφο συνεδρίου, κερδίζοντας το OpenCv People’s Choice Award.

Ο αλγόριθμος YOLO χρησιμοποιεί συνελικτικά νευρωνικά δίκτυα (CNN) για την ανίχνευση αντικειμένων σε πραγματικό χρόνο. Όπως υποδηλώνει και η ονομασία του, ο αλγόριθμος απαιτεί μόνο μια φορά την είσοδο μιας εικόνας μέσω ενός νευρωνικού δικτύου για την ανίχνευση των αντικειμένων. Έτσι η πρόβλεψη που γίνεται σε ολόκληρη την εικόνα πραγματοποιείται με μόνο μία εκτέλεση του συγκεκριμένου αλγορίθμου. Το συνελικτικό δίκτυο (CNN) χρησιμοποιείται για την πρόβλεψη διάφορων πιθανοτήτων κλάσεων και πλαισίων (bounding boxes) ταυτόχρονα.

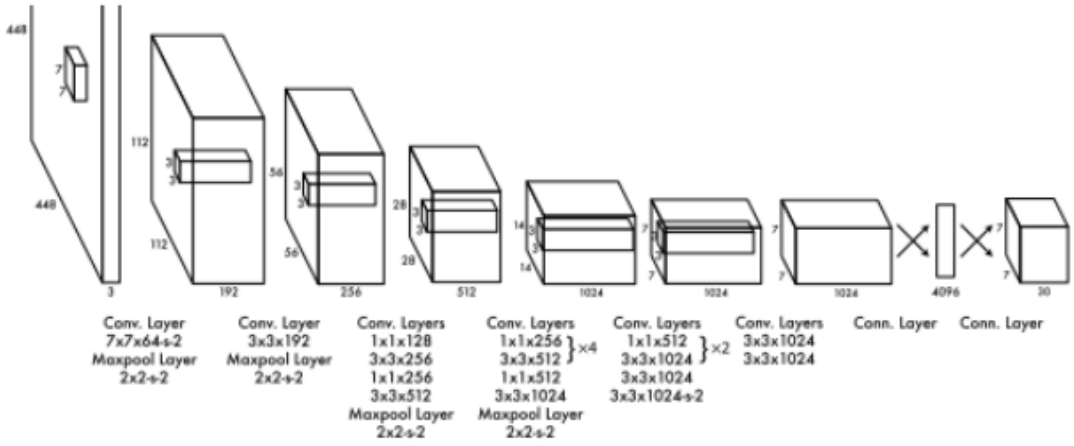
6.1.1 Σημαντικότητα YOLO

Ο αλγόριθμος YOLO κατατάσσεται στους πιο σημαντικούς αλγορίθμους στην αναγνώριση αντικειμένων λόγω της εξαιρετικής του ταχύτητας, αφού βελτιώνει την ταχύτητα της αναγνώρισης αναγνωρίζοντας αντικείμενα σε πραγματικό χρόνο. Ένας επιπλέον λόγος που καθιστά τον αλγόριθμο YOLO σημαντικό είναι η υψηλή του ακρίβεια, μιας και προσφέρει υψηλά επίπεδα ακρίβειας με ελάχιστα σφάλματα (minimal background errors). Τέλος ο αλγόριθμος παρουσιάζει εξαιρετικές

μαθησιακές δυνατότητες που του επιτρέπουν να μάθει τις αναπαραστάσεις αντικειμένων και να τις εφαρμόσει ύστερα στην ανίχνευση αντικειμένων (object detection).

6.1.2 YOLO Architecture

Εμπνευσμένος από την αρχιτεκτονική του GoogleNet, η αρχιτεκτονική του YOLO συνολικά αποτελείται από 24 Συνελικτικά Νευρωνικά Δίκτυα (CNN), 4 max pooling layers και 2 πλήρως συνδεδεμένα δίκτυα (fully connected layers) στο τέλος. Η συγκεκριμένη αρχιτεκτονική δέχεται σαν είσοδο μια εικόνα και αλλάζει το μέγεθός της σε 448*448, διατηρώντας την ίδια αναλογία διαστάσεων και εκτελώντας συμπλήρωση (padding). Επιπλέον, η αρχιτεκτονική του YOLO χρησιμοποιεί ως συνάρτηση ενεργοποίησής τη Leaky Relu σε ολόκληρη την αρχιτεκτονική εκτός από το επίπεδο όπου χρησιμοποιεί γραμμική συνάρτηση ενεργοποίησης.



Εικόνα 26: Αρχιτεκτονική YOLO

6.1.3 Περιορισμοί YOLO

Παρόλο που ο αλγόριθμος YOLO θεωρείται ένας από τους αποτελεσματικότερους αλγορίθμους στην αναγνώριση αντικειμένων σε εικόνες, παρουσιάζει διάφορους περιορισμούς.

Ένας από αυτούς τους περιορισμούς είναι ότι ο αλγόριθμος YOLO χαρακτηρίζεται από χαμηλότερη ακρίβεια σε σύγκριση με πιο αργούς αλγορίθμους ανίχνευσης αντικειμένων, όπως για παράδειγμα τον αλγόριθμο Fast R-CNN.

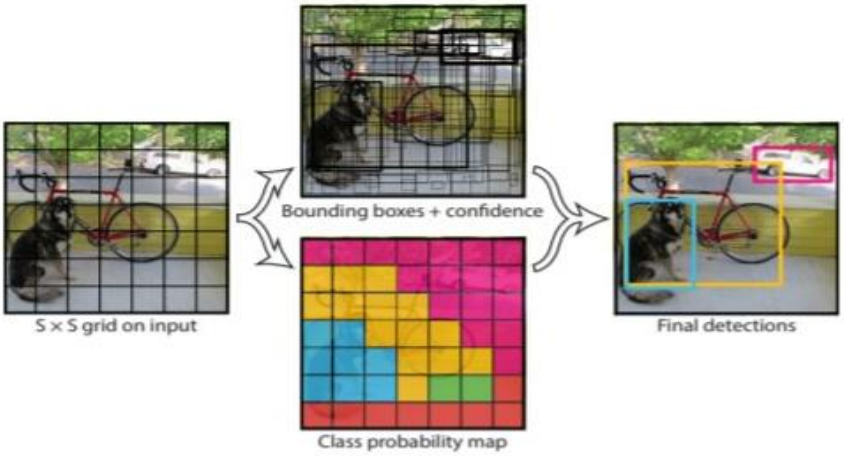
6.1.4 Λειτουργία YOLO

Για παράδειγμα, παρακάτω (Εικόνα 27) παρατηρείται πως το αυτοκίνητο περιβάλλεται από το ροζ πλαίσιο, το ποδήλατο από το κίτρινο πλαίσιο και τέλος ο σκύλος έχει επισημανθεί χρησιμοποιώντας μπλε πλαίσιο οριοθέτησης.

Το YOLO σημαίνει You Only Look Once. Είναι ένα υπερσύγχρονο σύστημα ανίχνευσης αντικειμένων σε πραγματικό χρόνο, το οποίο μπορεί να αναγνωρίσει πολλά

αντικείμενα σε ένα μόνο πλαίσιο. Επιπλέον, οι πιο εξελιγμένες εκδόσεις του YOLO είναι ο YOLOv2, ο YOLOv3 και ο YOLOv4. Ο YOLO χρησιμοποιεί μια εντελώς διαφορετική προσέγγιση από άλλα προηγούμενα συστήματα ανίχνευσης αντικειμένων. Εφαρμόζει μόνο ένα νευρωνικό δίκτυο σε ολόκληρη την εικόνα. Το δίκτυο αυτό, όπως αναφέρθηκε και παραπάνω διαιρεί την εικόνα σε περιοχές και προβλέπει οριοθετημένα πλαίσια και πιθανότητες για κάθε μία από αυτές τις περιοχές. Αυτά τα οριοθετημένα πλαίσια αποκτούν βάρη (weights), δηλαδή σταθμίζονται, ανάλογα με τις προβλεπόμενες πιθανότητες. Η βασική ιδέα του YOLO επίσης απεικονίζεται στην εικόνα παρακάτω, όπου η εικόνα εισόδου διαιρείται σε ένα πλέγμα S x S και κάθε κελί του πλέγματος αυτού είναι υπεύθυνο για την πρόβλεψη του αντικειμένου που βρίσκεται μέσα σε αυτό το κελί. Επιπλέον, κάθε κελί πλέγματος προβλέπει B οριοθετημένα πλαίσια (bounding boxes) και βαθμολογίες εμπιστοσύνης για τα πλαίσια αυτά. Αυτές οι βαθμολογίες εμπιστοσύνης αντικατοπτρίζουν πόσο σίγουρο είναι το μοντέλο, για το αν το πλαίσιο περιέχει ένα αντικείμενο και επίσης πόσο ακριβές είναι για αυτό που προβλέπει.

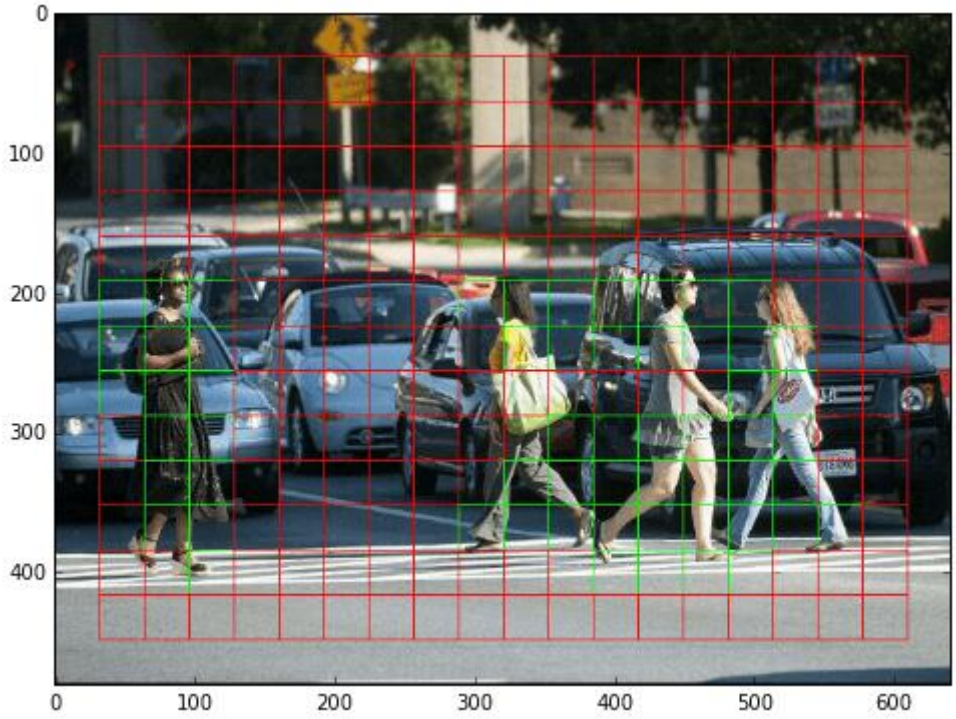
Επίσης κάνει προβλέψεις με μία μόνο αξιολόγηση δικτύου, σε αντίθεση με άλλους αλγορίθμους όπως ο R-CNN που απαιτεί χιλιάδες αξιολογήσεις δικτύου για μία μόνο εικόνα. Το γεγονός αυτό καθιστά τον αλγόριθμο YOLO εξαιρετικά γρήγορο, περισσότερο από 1000 φορές ταχύτερο από τον R-CNN και 100 φορές ταχύτερο από τον Fast R-CNN. Ο αλγόριθμος YOLO επιτρέπει την εκπαίδευση από άκρο σε άκρο και ταχύτητες σε πραγματικό χρόνο διατηρώντας παράλληλα υψηλή μέση ακρίβεια.



Εικόνα 27: Αλγόριθμος YOLO

6.1.4.1 Residual blocks

Η λειτουργία του αλγορίθμου YOLO βασίζεται στην διαίρεση της εικόνας σε N κελιά καθένα από αυτά έχει ίδια διάσταση SxS και είναι υπεύθυνο για την αναγνώριση και τον εντοπισμό της θέσης του αντικειμένου.

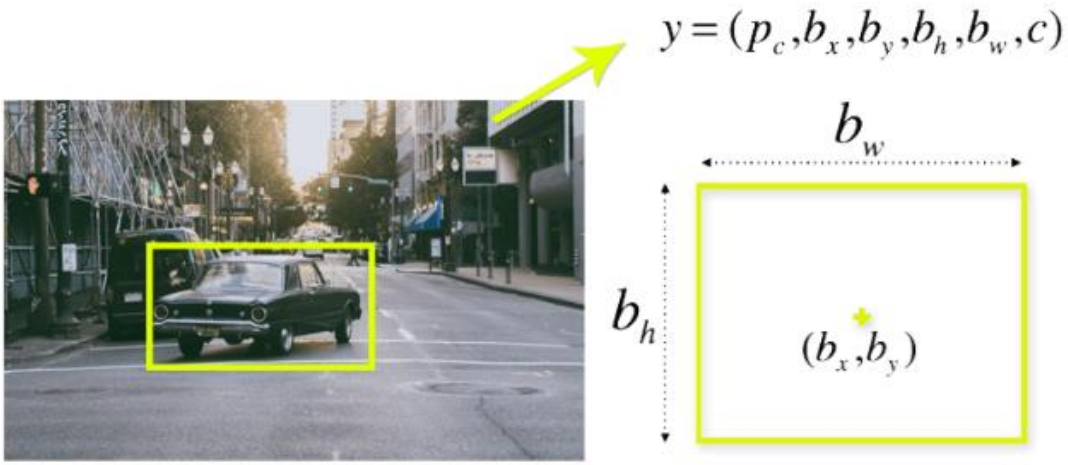


Εικόνα 28: YOLO Residual Blocks

6.1.4.2 Bounding BOX

Κάθε bounding box αποτελείται από τα εξής χαρακτηριστικά:

1. Πλάτος (bw)
2. Ύψος (bh)
3. Κλάση αντικειμένου (c)
4. Κέντρο του πλαισίου (bx,by)

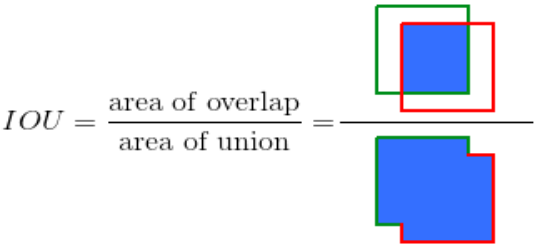


Εικόνα 29: YOLO Bounding Box

6.1.4.3 Intersection Over Union

Intersection Over Union είναι ένα φαινόμενο στην αναγνώριση αντικειμένων, το οποίο περιγράφει πως τα boxes επικαλύπτουν το ένα το άλλο.

Κάθε grid cell είναι υπεύθυνο για την πρόβλεψη των bounding boxes και για τα confidence scores τους. The IOU είναι ίσο με 1 εάν το προβλεπόμενο bounding box είναι ακριβώς το ίδιο με το πραγματικό πλαίσιο (box). Με τον μηχανισμό αυτό μειώνονται τα bounding boxes που δεν είναι ίδια με τα πραγματικά. Ο τύπος για τον υπολογισμό του Intersection Over Union (IOU) έχει ως εξής:



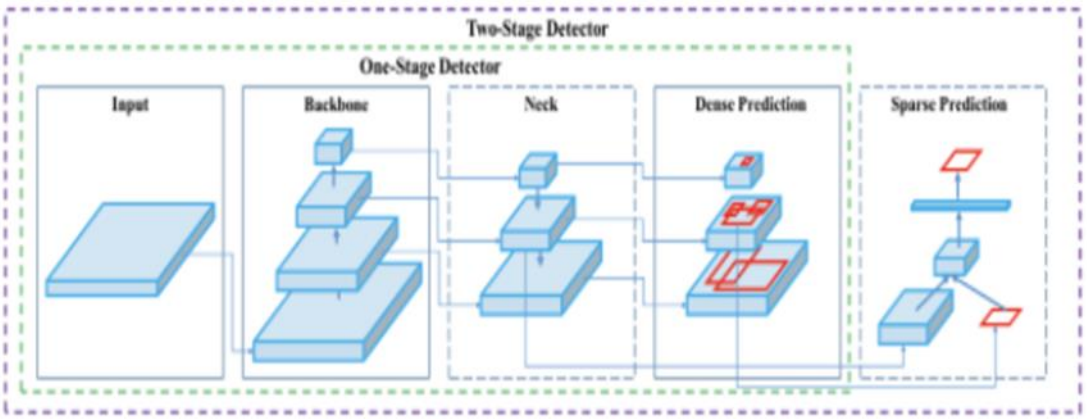
Εικόνα 30: YOLO Intersection Over Union (IOU)

Αρχικά, η εικόνα χωρίζεται σε κελιά πλέγματος (grid cells), κάθε κελί προβλέπει τα όρια των οριοθετημένων κουτιών (bounding boxes) και παρέχει τις βαθμολογίες εμπιστοσύνης τους. Τα κελιά προβλέπουν τις πιθανότητες κλάσης για να καθορίσουν την κλάση του κάθε αντικειμένου.

Η Intersection Over Union (IOU) διασφαλίζει ότι τα προβλεπόμενα οριοθετημένα πλαίσια είναι ίσα με τα πραγματικά πλαίσια των αντικειμένων. Το φαινόμενο αυτό εξαλείφει τα περιττά πλαίσια οριοθέτησης που δεν πληρούν τα χαρακτηριστικά των αντικειμένων (όπως ύψος και πλάτος). Η τελική ανίχνευση θα αποτελείται από μοναδικά πλαίσια οριοθέτησης που ταιριάζουν ιδανικά στα αντικείμενα.

6.2 YOLOV4

Ο Yolov4 βελτιώνει το Yolov3 σε AP και FPS κατά 10% και 12% αντίστοιχα. Υπάρχει ένας μεγάλος αριθμός χαρακτηριστικών που βελτιώνουν την ακρίβεια των Συνελικτικών Νευρωνικών Δικτύων. Επιλέχθηκαν ορισμένα καθολικά χαρακτηριστικά που ισχύουν για την πλειονότητα των μοντέλων, διεργασιών, και συνόλων δεδομένων και επαληθεύτηκε η επιρροή τους στην εκπαίδευση, το οποίο ονομάζουμε και Bag-of-Freebies και Bag-of-Specials μεθόδους στην ανίχνευση αντικειμένων. Τα Bag-of-Freebies είναι μέθοδοι που αυξάνουν μόνο το κόστος εκπαίδευσης και κάνουν τον ανιχνευτή αντικειμένων να έχει καλύτερη ακρίβεια χωρίς όμως να αυξάνει το κόστος των συμπερασμάτων. Η επαύξηση των δεδομένων (Data Augmentation) είναι ένα τέτοιο παράδειγμα Bag-of-Freebies, που αυξάνει την μεταβλητότητα των εικόνων εισόδου, γεγονός το οποίο με τη σειρά του κάνει το μοντέλο ανίχνευσης ανθεκτικό στις εικόνες που λαμβάνονται από διαφορετικά περιβάλλοντα. Τα Bag-of-Specials, είναι μέθοδοι που αυξάνουν μόνο το κόστος συμπερασμάτων κατά ένα μικρό ποσό, αλλά βελτιώνουν σημαντικά την ακρίβεια της ανίχνευσης. Αυτά τα μοντέλα ενισχύουν ορισμένα χαρακτηριστικά σε ένα μοντέλο.

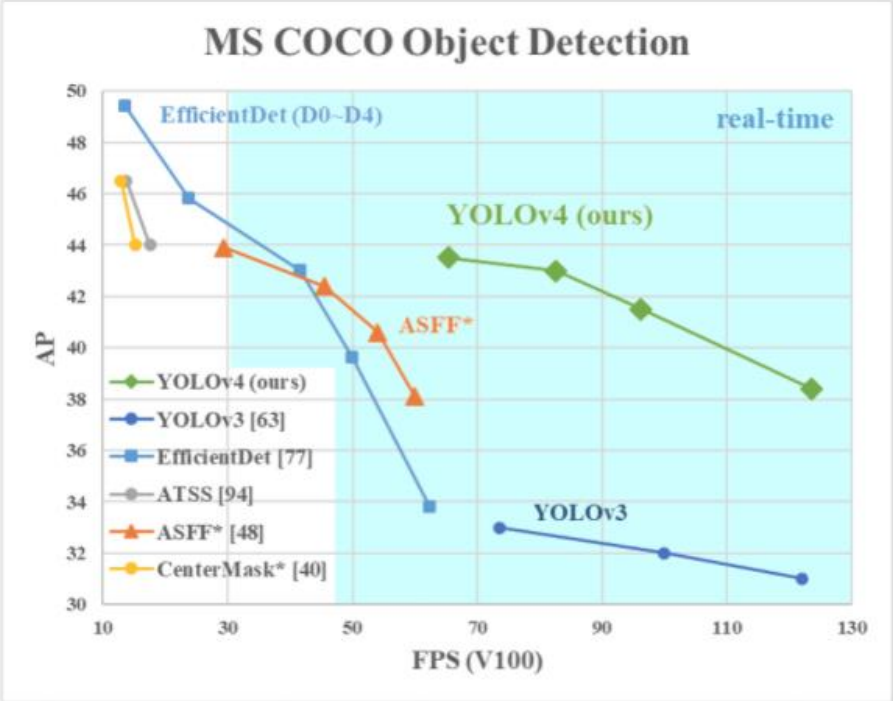


Εικόνα 31: Αρχιτεκτονική YOLOv4

Ο αλγόριθμος YOLOv4 χρησιμοποιεί πολλές νέες δυνατότητες και συνδυάζει ορισμένες από αυτές, για να επιτύχει ένα αποτέλεσμα αιχμής με 43,5% AP (65,7% AP50) για το σύνολο δεδομένων MS COCO με ταχύτητα σε πραγματικό χρόνο ~65 FPS στο Tesla V100. Παρακάτω αναφέρονται τα νέα χαρακτηριστικά που χρησιμοποιεί:

- Weighted-Residual-Connections (WRC)
- Cross-Stage-Partial-connections (CSP)
- Cross mini-Batch Normalization (CmBN)
- Self-adversarial-training (SAT)
- Mish activation
- Mosaic data augmentation
- DropBlock regularization

- Complete Intersection over Union loss (CIoU loss)



Εικόνα 32: Σύγκριση Διάφορων Αλγορίθμων με YOLOv4

Ο αλγόριθμος YOLOv4 είναι ένας αλγόριθμος ανίχνευσης αντικειμένων που αποτελεί στην ουσία την εξέλιξη του μοντέλου YOLOv3. Η έκδοση YOLOv4 δημιουργήθηκε από τους Alexey Bochkovskiy, Chien-Yao Wang και Hong-Yuan Mark Liao. Είναι δύο φορές πιο γρήγορη από τον EfficientDet σε σύγκριση επιδόσεων. Επιπλέον, το AP (Average Precision) και το FPS (Frames Per Second) έχουν αυξηθεί κατά 10% και 12% αντίστοιχα σε σύγκριση με το YOLOv3. Η αρχιτεκτονική του αποτελείται από CSPDarknet53 σαν ραχοκοκαλιά (backbone), spatial pyramid pooling, PANet path-aggregation neck και YOLOv3 head, το κάθε ένα από αυτά παρουσιάζεται αναλυτικά στη συνέχεια του κεφαλαίου.

- Backbone CSPDarknet53

Ο αλγόριθμος YOLOv4 χρησιμοποιεί συνδέσεις CSP με το Darknet53 ως τον κορμό ή ραχοκοκαλία (backbone) στην εξαγωγή χαρακτηριστικών.

Το μοντέλο CSPDarknet53 διαθέτει πολύ μεγαλύτερη ακρίβεια στην ανίχνευση των αντικειμένων σε σύγκριση με την ανίχνευση αντικειμένων που βασίζεται στους σχεδιασμούς του ResNet, ακόμη κι αν αυτοί έχουν καλύτερη απόδοση ταξινόμησης. Όμως η ακρίβεια ταξινόμησης του CSPDarknet53 μπορεί να βελτιωθεί και με άλλων ειδών τεχνικές. Επομένως η τελική επιλογή για το YOLOv4 είναι το CSPDarknet.

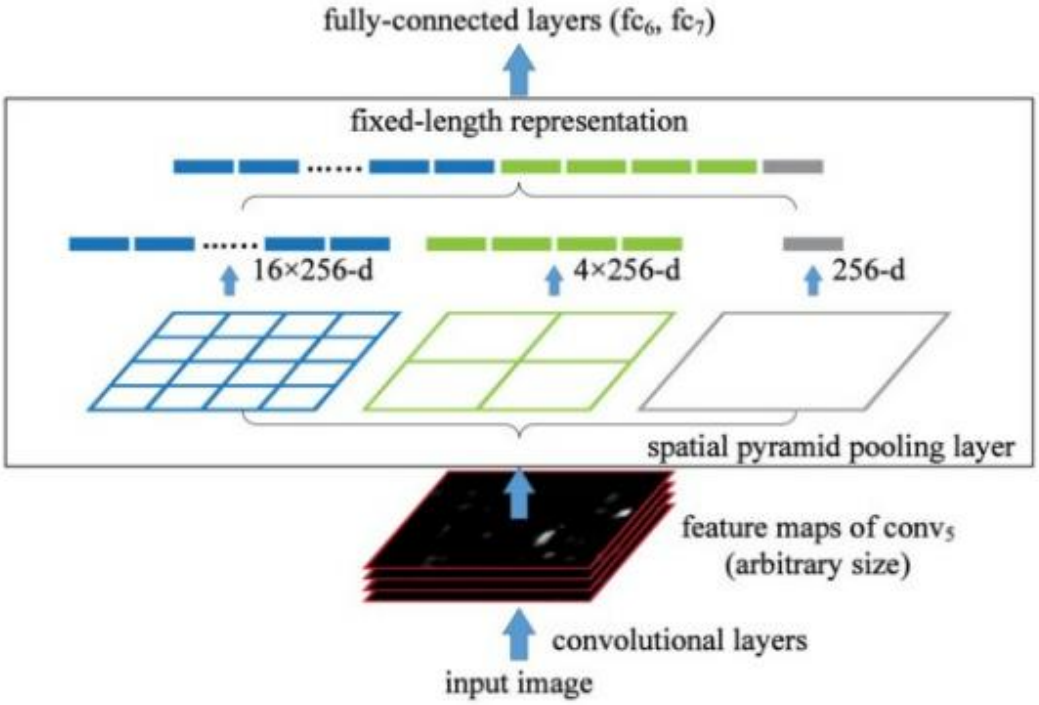
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 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 33: CSPDarknet53

- SPP (Spatial Pyramid Pooling Layer)

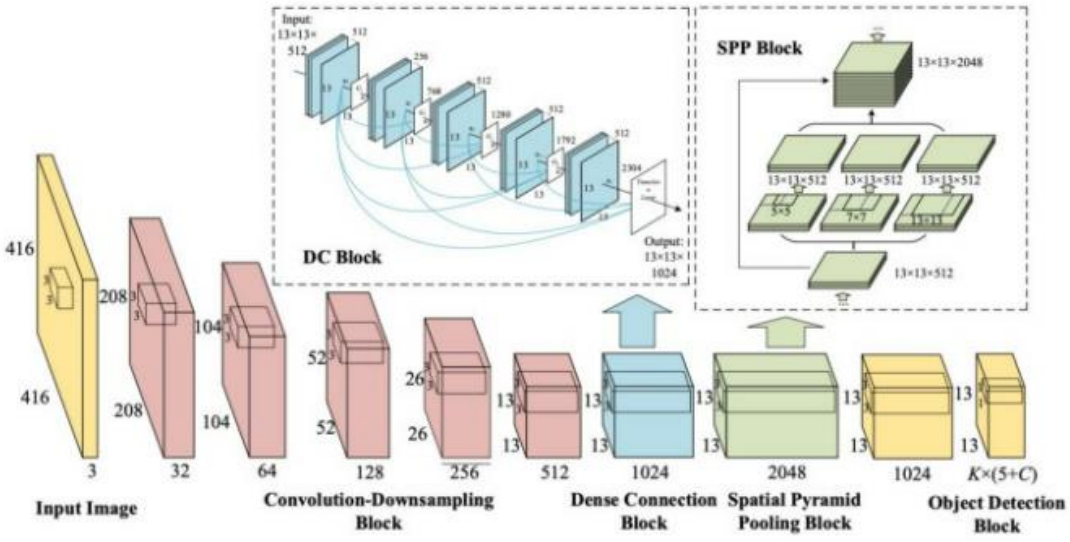
Το SPP εφαρμόζει μια ελαφρώς διαφορετική στρατηγική στην ανίχνευση αντικειμένων διαφορετικής κλίμακας. Αντικαθιστά το τελευταίο στρώμα συγκέντρωσης (μετά το τελευταίο συνελικτικό στρώμα) με ένα στρώμα συγκέντρωσης χωρικής πυραμίδας. Οι χάρτες χαρακτηριστικών χωρίζονται χωρικά σε θέσεις m*m με το m να ισούται με 1,2 και 4 αντίστοιχα. Στη συνέχεια εφαρμόζεται ένα μέγιστο pool σε κάθε κάδο για κάθε κανάλι. Αυτό σχηματίζει μια αναπαράσταση σταθερού μήκους που μπορεί να αναλυθεί περαιτέρω με FC-layers.

Πολλά μοντέλα που βασίζονται στα Συνελικτικά Νευρωνικά Δίκτυα (CNN) περιέχουν επίπεδα FC (FC-layers), κατά συνέπεια δέχονται εικόνες εισόδου μόνο συγκεκριμένων διαστάσεων. Αντιθέτως, το SPP δέχεται εικόνες διαφορετικών μεγεθών. Ωστόσο υπάρχουν τεχνολογίες, όπως τα δίκτυα πλήρους συνέλιξης (FCN), που δεν περιέχουν επίπεδα FC και δέχονται εικόνες διαφορετικών διαστάσεων. Ο συγκεκριμένος τύπος σχεδίασης είναι ιδιαίτερα χρήσιμος για την τμηματοποίηση εικόνων μιας και οι χωρικές πληροφορίες είναι ιδιαίτερα σημαντικές. Επομένως για τον YOLO, η μετατροπή 2-D χαρτών χαρακτηριστικών σε ένα σταθερού μεγέθους 1-D διάνυσμα δεν είναι απαραίτητα επιθυμητή.



Εικόνα 34: SPP (Spatial Pyramid Pooling Layer)

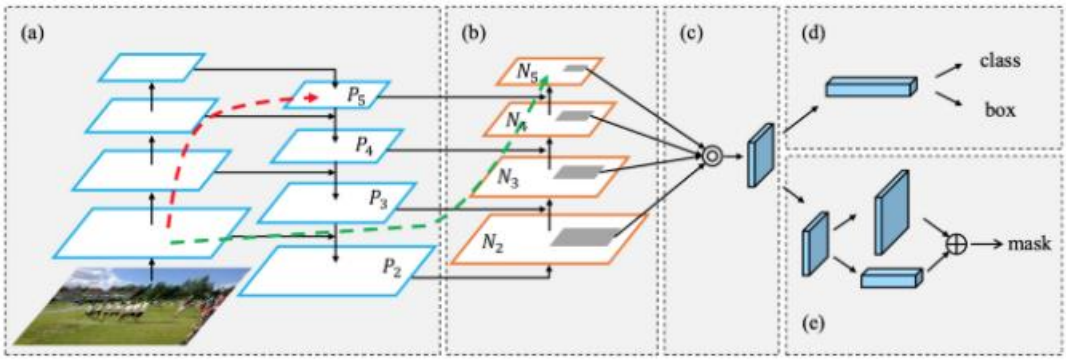
Στο YOLO, το SPP τροποποιείται για να διατηρήσει τη χωρική διάσταση εξόδου. Ένα μέγιστο pool εφαρμόζεται πάνω σε ένα sliding kernel, μεγέθους 1*1, 5*5, 9*9, 13*13 και η χωρική διάσταση διατηρείται. Οι χάρτες χαρακτηριστικών από διαφορετικά μεγέθη πυρήνα (kernel size) στη συνέχεια συνενώνονται μεταξύ τους ως έξοδος. Το παρακάτω διάγραμμα δείχνει πως το SPP ενσωματώνεται στο YOLO.



Εικόνα 35: Ενσωμάτωση SPP στον YOLOv4

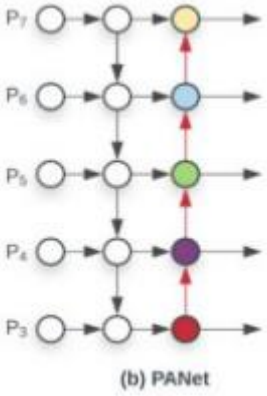
- Path Aggregation Network (PAN)

Το διάγραμμα παρακάτω (Εικόνα 36) απεικονίζει την λειτουργία του Path Aggregation Network (PAN). Μία διαδρομή από κάτω προς τα επάνω (b) επαυξάνεται για να διευκολύνει τη διάδοση πληροφοριών χαμηλού επιπέδου στην κορυφή. Στο FPN, οι εντοπισμένες χωρικές πληροφορίες ταξιδεύουν προς τα πάνω όπως δείχνει το κόκκινο βέλος. Αν και δεν είναι απόλυτα κατανοητό στο διάγραμμα η κόκκινη διαδρομή περνά από παραπάνω από 100 στρώματα. Με την εφαρμογή του PAN εισάγεται μια νέα πιο σύντομη διαδρομή αυτή με το πράσινο χρώμα, η οποία χρειάζεται μόνο περίπου 10 στρώσεις για να μεταβεί στο επόμενο επίπεδο N_5.



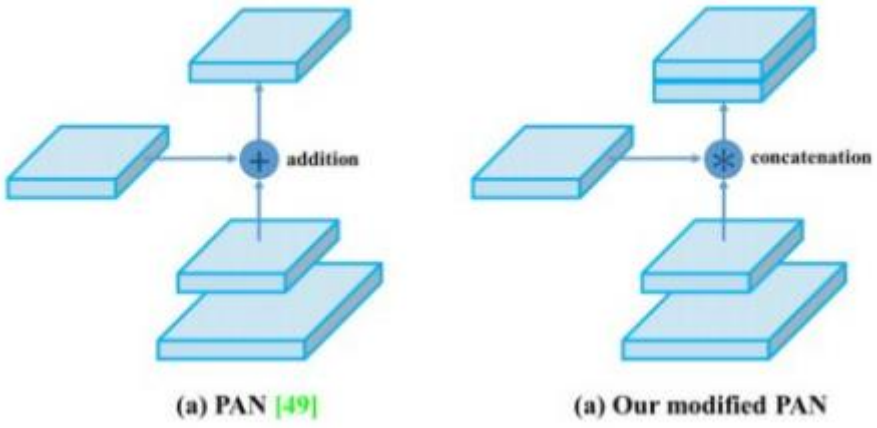
Εικόνα 36: Λειτουργία PANet

Επίσης μπορούμε να απεικονίσουμε το σχέδιο του neck στο παρακάτω διάγραμμα:



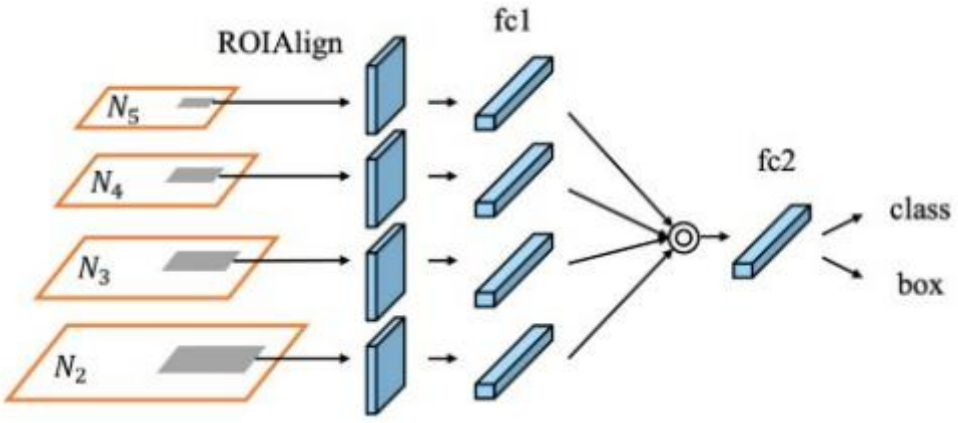
Εικόνα 37: PANet

Ωστόσο, αντί να προσθέτουμε γειτονικά επίπεδα, οι χάρτες χαρακτηριστικών στον YOLOv4 αλγόριθμο συνδέονται και μεταξύ τους.



Εικόνα 38: Τροποποίηση PAN

Τέλος στο FPN, τα αντικείμενα εντοπίζονται χωριστά και ανεξάρτητα σε διαφορετικά επίπεδα κλίμακας. Όμως έτσι μπορεί να παράγει διπλές προβλέψεις και να μην χρησιμοποιεί πληροφορίες από άλλους χάρτες χαρακτηριστικών. Το PAN συγχωνεύει πρώτα τις πληροφορίες από όλα τα επίπεδα χρησιμοποιώντας max operation.



Εικόνα 39: Απεικόνιση Συγκέντρωσης Χαρακτηριστικών PAN

6.3 YOLOv4-tiny

Το YOLOv4-tiny είναι η συμπιεσμένη έκδοση του YOLOv4. Βασίζεται σε μεγάλο βαθμό στον αλγόριθμο YOLOv4 και στοχεύει στην απλοποίηση του δικτύου και στην μείωση των παραμέτρων, έτσι ώστε να καταστεί εφικτή η ανάπτυξη σε κινητές συσκευές και ενσωματωμένες κατασκευές.

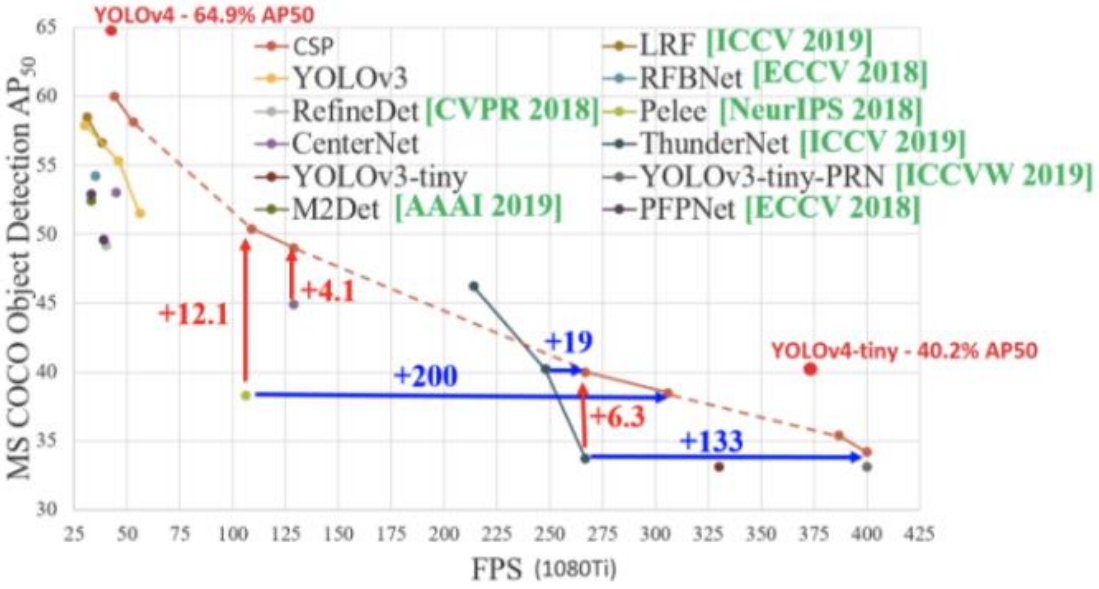
Μπορούμε να χρησιμοποιήσουμε τον YOLOv4-tiny για ταχύτερη εκπαίδευση και ταχύτερη ανίχνευση. Αποτελείται μόνο από δύο κεφαλές YOLO σε αντίθεση σε αντίθεση με τον YOLOv4 που διαθέτει τρεις και έχει εκπαιδευτεί από 29 προ εκπαιδευμένα συνελικτικά στρώματα σε αντίθεση με τον YOLOv4 που έχει εκπαιδευτεί από 137 προ εκπαιδευμένα συνελικτικά στρώματα.

Το FPS (Frames Per Seconds) στο YOLOv4-tiny είναι περίπου 8 φορές μεγαλύτερο από τον YOLOv4. Ωστόσο η ακρίβεια για τον YOLOv4-tiny είναι τα 2/3 του YOLOv4 όταν δοκιμάστηκε στο σύνολο δεδομένων MS COCO.

Επιπλέον, το μοντέλο YOLOv4-tiny επιτυγχάνει 22,0% AP (42,0% AP50) με ταχύτητα 443 FPS σε RTX 2080Ti, ενώ χρησιμοποιώντας TensorRT, με batch size = 4 και ακρίβεια FP16, το YOLOv4-tiny επιτυγχάνει 1774 FPS.

Στο σημείο αυτό είναι σημαντικό να αναφέρουμε πως σε ανίχνευση αντικειμένων σε πραγματικό χρόνο, το YOLOv4-tiny αποτελεί καλύτερη επιλογή σε σύγκριση με το YOLOv4, καθώς ο ταχύτερος χρόνος συμπερασμάτων είναι πιο σημαντικός από την ακρίβεια ή την ακρίβεια κατά την εργασία με περιβάλλον ανίχνευσης αντικειμένων σε πραγματικό χρόνο.

Το YOLOv4-Tiny έχει συγκριτικά ανταγωνιστικά αποτελέσματα με το YOLOv4 δεδομένης της μείωσης μεγέθους. Επιτυγχάνει 40 mAP @.5 στο σύνολο δεδομένων MS COCO. Τα αποτελέσματα απεικονίζονται στο παρακάτω διάγραμμα.



Εικόνα 40: Σύγκριση YOLOv4-tiny με διάφορους αλγορίθμους σε FPS

7.Υλοποίηση ανίχνευσης αντικειμένων

Python

Στην παρούσα διπλωματική εργασία η υλοποίηση του αλγορίθμου ανίχνευσης αντικειμένων πραγματοποιήθηκε στην γλώσσα προγραμματισμού Python, μια εξαιρετικά δημοφιλής γλώσσα προγραμματισμού για την Μηχανική Μάθηση χάρη στην ευελιξία της και στο πλήθος των βιβλιοθηκών από τις οποίες απαρτίζεται.

Anaconda

Το Anaconda είναι ένα εργαλείο που διανέμει την γλώσσα προγραμματισμού Python προ εγκατεστημένη με πολλές χρήσιμες, αλλά και απαραίτητες βιβλιοθήκες της python για την επιστήμη των δεδομένων (data science) και της μηχανικής μάθησης. Το Anaconda είναι δημοφιλές επειδή διαθέτει πολλά από τα εργαλεία που χρησιμοποιούνται στην επιστήμη δεδομένων και τη μηχανική εκμάθηση με μία μόνο εγκατάσταση.

Darknet

Το Darknet είναι ένα framework νευρωνικών δικτύων, το οποίο γράφτηκε σε γλώσσα προγραμματισμού C και στην τεχνολογία CUDA. Το γεγονός αυτό, το καθιστά εξαιρετικά γρήγορο και παρέχει υπολογισμούς στην GPU, κάτι το οποίο θεωρείται απαραίτητο για προβλέψεις σε πραγματικό χρόνο.

MSVC (Microsoft Visual Studio)

Το MSVC (Microsoft Visual Studio) είναι ένα IDE που δημιουργήθηκε από την Microsoft και χρησιμοποιείται για διαφορετικούς τύπους ανάπτυξης λογισμικού, όπως προγράμματα υπολογιστών, ιστότοπους, εφαρμογές και υπηρεσίες ιστού και κινητών συσκευών. Το MSVC περιλαμβάνει εργαλεία ολοκλήρωσης, μεταγλωττιστές και άλλες δυνατότητες που καθιστούν ευκολότερη τη διαδικασία ανάπτυξης λογισμικού.

CMake

Το CMake είναι ένα εργαλείο ανοιχτού κώδικα με πολλαπλές πλατφόρμες, το οποίο χρησιμοποιεί αρχεία διαμόρφωσης ανεξάρτητα από τον μεταγλωττιστή και την πλατφόρμα για τη δημιουργία εγγενών αρχείων εργαλείων ειδικά σχεδιασμένα για τον μεταγλωττιστή και την πλατφόρμα όπου θα χρησιμοποιηθούν. Η επέκταση CMake Tools ενσωματώνει τον κώδικα του Visual Studio έτσι ώστε να διευκολύνει την διαμόρφωση, την δημιουργία και τον εντοπισμό σφαλμάτων του έργου που είναι υλοποιημένο σε γλώσσα προγραμματισμού C++.

CUDA

Το CUDA είναι μια παράλληλη πλατφόρμα υπολογιστών και μοντέλο προγραμματισμού που αναπτύχθηκε από την Nvidia για γενικούς υπολογισμούς στις δικές της GPU (μονάδες επεξεργασίας γραφικών ή κάρτες γραφικών). Επιπλέον, το CUDA δίνει τη δυνατότητα στους προγραμματιστές να επιταχύνουν την ένταση στις

εφαρμογές των υπολογιστών αξιοποιώντας την ισχύ των GPU για το παραλληλιζόμενο μέρος του υπολογισμού.

cuDNN

Η βιβλιοθήκη NVIDIA CUDA Deep Neural Network (cuDNN) είναι μια βιβλιοθήκη με επιτάχυνση GPU για βαθιά νευρωνικά δίκτυα. Το cuDNN δίνει τη δυνατότητα για εξαιρετικά συντονισμένες υλοποιήσεις, όπως η συνέλιξη προς τα εμπρός (forward convolution), η συνέλιξη προς τα πίσω (backward convolution), η συγκέντρωση (pooling), η κανονικοποίηση (normalization) και τα επίπεδα ενεργοποίησης (activation layers).

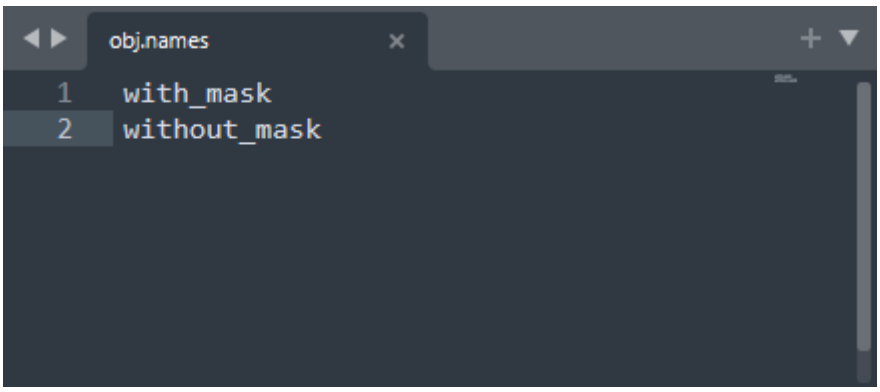
OpenCV

Το OpenCV είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα για υπολογιστική όραση, την μηχανική μάθηση και την επεξεργασία εικόνας και συμβάλλει σημαντικά στη λειτουργία των παραπάνω σε πραγματικό χρόνο, γεγονός αρκετά σημαντικό για τα σημερινά συστήματα. Χρησιμοποιώντας το OpenCV, δίνεται η δυνατότητα για την αναγνώριση αντικειμένων, προσώπων κ.τ.λ. Έχει σχεδιαστεί κυρίως για χρήση με γλώσσα προγραμματισμού Python, αλλά είναι επίσης διαθέσιμη και με τη γλώσσες προγραμματισμού C++ και Java.

Τα αρχεία που είναι απαραίτητα για την υλοποίηση του αλγορίθμου είναι τα παρακάτω:

obj.names

Περιλαμβάνει τα ονόματα των κλάσεων του συνόλου δεδομένων. Κάθε όνομα αντιστοιχεί σε μία γραμμή, with_mask (με μάσκα) και without_mask (χωρίς μάσκα).



Εικόνα 41: Αρχείο obj.names

yolo.cfg:

[net]

- ✓ *Batch:* ο αριθμός των εικόνων που λαμβάνει ο αλγόριθμος για κάθε επανάληψη. Για παράδειγμα batch = 64 σημαίνει ότι ο αλγόριθμος φορτώνει 64 εικόνες για μία επανάληψη.

- ✓ *Subdivisions:* Ο αριθμός των mini-batches στις οποίες ο αλγόριθμος χωρίζει τα batches. Για παράδειγμα subdivisions=64, σημαίνει ότι χωρίζει τα batches σε 64 mini-batches, έτσι ώστε 64/64=1 εικόνα ανά mini-batches και αυτή η μία εικόνα αποστέλλεται για επεξεργασία. Αυτή η διαδικασία θα εκτελεστεί 64 φορές μέχρι να ολοκληρωθεί το batch και έπειτα θα ξεκινήσει μία νέα επανάληψη με 64 νέες εικόνες.
- ✓ *Width:* Μέγεθος δικτύου (πλάτος). Επομένως το μέγεθος κάθε εικόνας θα αλλάξει στο μέγεθος του δικτύου κατά την εκπαίδευση και τον εντοπισμό.
- ✓ *Height:* Μέγεθος δικτύου (ύψος). Επομένως το μέγεθος κάθε εικόνας θα αλλάξει στο μέγεθος του δικτύου κατά την εκπαίδευση και τον εντοπισμό.
- ✓ *Channels:* Τα κανάλια αναφέρονται στο μέγεθος του καναλιού της εικόνας εισόδου, το οποίο είναι 3 για εικόνες RGB.
- ✓ *Momentum:* Συσσώρευση κίνησης, πόσο επηρεάζει το ιστορικό την περαιτέρω αλλαγή βαρών.
- ✓ *Decay:* Μια πιο αδύναμη ενημέρωση των βαρών για τυπικά χαρακτηριστικά, εξαλείφει τη δυσαναλογία στο σύνολο των δεδομένων.
- ✓ *Angle:* Περιστρέφει τυχαία τις εικόνες κατά τη διάρκεια της εκπαίδευσης.
- ✓ *Saturation:* Αλλάζει τυχαία τον κορεσμό των εικόνων κατά τη διάρκεια της εκπαίδευσης.
- ✓ *Exposure:* Αλλάζει τυχαία την φωτεινότητα των εικόνων κατά τη διάρκεια της εκπαίδευσης.
- ✓ *Hue:* Αλλάζει τυχαία την απόχρωση (χρώμα) των εικόνων κατά τη διάρκεια της εκπαίδευσης.
- ✓ *Learning_rate:* Αρχικό ποσοστό μάθησης για εκπαίδευση.
- ✓ *Burn_in:* Το αρχικό burn_in θα υποβληθεί σε επεξεργασία για τις πρώτες 1000 επαναλήψεις, current_learning rate = learning_rate * pow(iterations / burn_in, power) = 0.001 * pow(iterations/1000, 4), όπου power = 4 από προεπιλογή.
- ✓ max_batches: ο αριθμός όλων των επαναλήψεων.
- ✓ *Policy:* Χρήση των παραμέτρων steps ή scales για την προσαρμογή του ρυθμού εκμάθησης κατά τη διάρκεια της εκπαίδευσης. Από προεπιλογή η παράμετρος policy μπορεί να πάρει τις παρακάτω τιμές: sgdr, steps, step, sig, exp, poly.
- ✓ *Steps:* Προσαρμογή του ρυθμού εκμάθησης μετά από 4800 και 5400 batches, εάν batches= 4800,5400.
- ✓ *Scales:* Ύστερα από 4800 batches ο ρυθμός εκμάθησης πολλαπλασιάζεται με 0.1 και στη συνέχεια μετά από 5400 batches ο ρυθμός εκμάθησης πολλαπλασιάζεται με 0.1, εάν scales=.1,.1.

[convolutional]

- ✓ *batch_normalize:* Εάν πάρει την τιμή 1, τότε γίνεται χρήση του batch_normalize, εάν πάρει την τιμή 0 δεν χρησιμοποιείται.
- ✓ *filters:* Υποδηλώνει των αριθμό των φίλτρων του πυρήνα (kernel-filters).
- ✓ *size:* Υποδηλώνει το kernel_size των filters.

- ✓ *stride:* Υποδηλώνει το βήμα μετατόπισης (διασκελισμό) του φίλτρου του πυρήνα, έχει την τιμή 1 από προεπιλογή.
- ✓ *pad:* Στην περίπτωση όπου έχει την τιμή 1, τότε ισχύει: padding = size/2.
- ✓ *activation:* Η συνάρτηση ενεργοποίησης.

yolov4-custom.cfg

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

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

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish
```

Εικόνα 42: Αρχείο yolo.cfg

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

process.py: Το αρχείο process.py δημιουργεί τα αρχεία train.txt και text.txt, όπου το αρχείο train.txt συμπεριλαμβάνει το path με το 90% των εικόνων και το test.txt συμπεριλαμβάνει το path με το 10% των εικόνων.

process.py

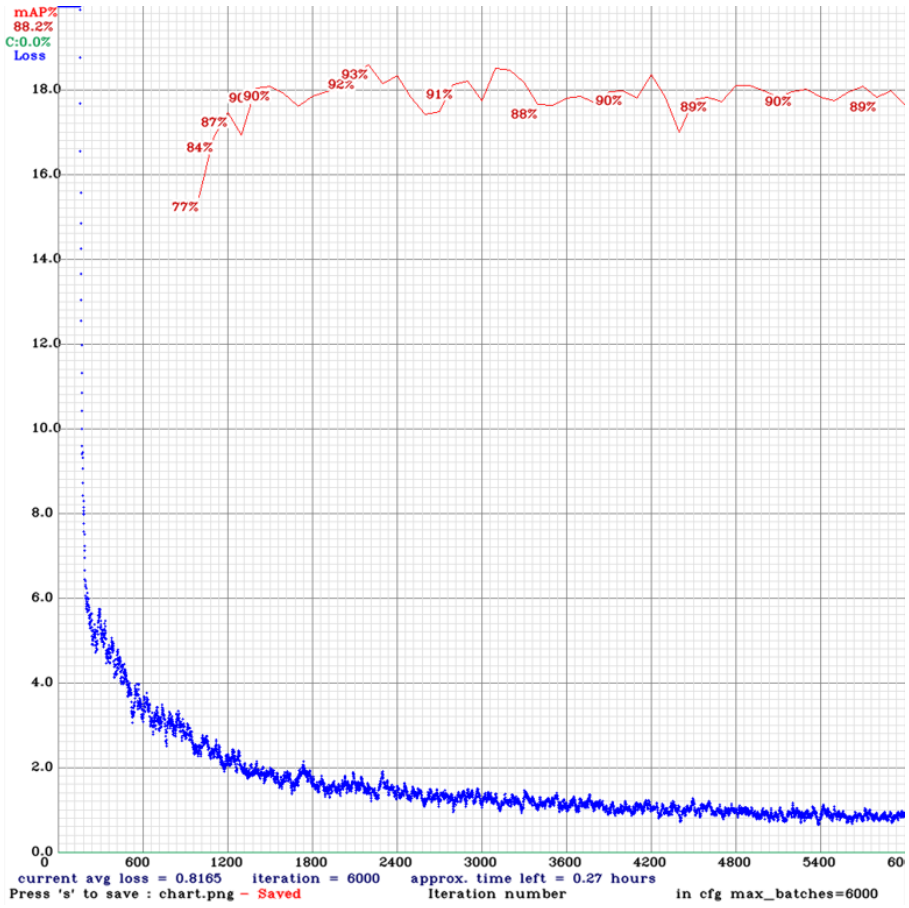
```
import glob, os

# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)

current_dir = 'data/obj'

# Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')

# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpg")):
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:
        counter = 1
        file_test.write("data/obj" + "/" + title + '.jpg' + "\n")
    else:
        file_train.write("data/obj" + "/" + title + '.jpg' + "\n")
        counter = counter + 1
```

Εικόνα 43: Αρχείο process.py

Η εκπαίδευση πραγματοποιήθηκε χρησιμοποιώντας το repository Darknet γραμμένο σε γλώσσα προγραμματισμού C. Το λειτουργικό σύστημα είναι τα Windows 10 Pro. Η CPU είναι AMD Ryzen 7 3700X 8-Core. Οι υπολογιστικοί πόροι είναι 64 GB RAM και GPU NVIDIA GeForce GTX 1660. Προκειμένου να γίνει πλήρης χρήση της GPU για την επιτάχυνση της εκπαίδευσης δικτύου, εγκαταστάθηκε CUDA 11.0 και το αντίστοιχο CUDNN στο σύστημα.

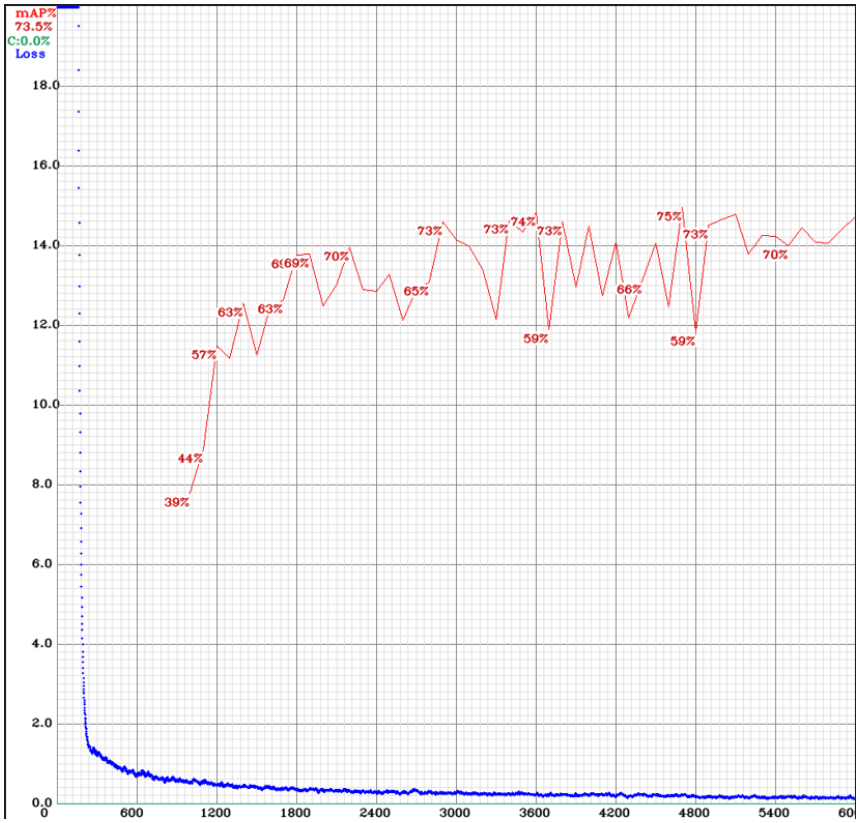
Μετά την εκπαίδευση του μοντέλου YOLOv4 δημιουργήθηκε ένα γράφημα όπως φαίνεται στην Εικόνα 44. Οι τιμές του Average loss και οι τιμές του mAP που χρησιμοποιούνται για την αξιολόγηση του μοντέλου αλλάζουν πολλές φορές κατά την εκπαίδευση του μοντέλου και αποδεικνύουν πόσο ακριβές είναι το μοντέλο. Το σύνολο δεδομένων ξεκίνησε από τις 0 επαναλήψεις και συνεχίστηκε μέχρι τις 6000 επαναλήψεις. Στις 1000 επαναλήψεις υπολογίστηκε το πρώτο mAP 77,24% και φαίνεται ότι το mAP ξεπέρασε το 90% σε περίπου 2000 επαναλήψεις. Κατά τη διάρκεια αυτής της εκπαίδευσης οι τιμές του Average loss και του mAP κυμαίνονται. Η χαμηλότερη τιμή του Average loss είναι περίπου 0,8 και υπολογίστηκε σε περίπου 2200 επαναλήψεις. Το καλύτερο και υψηλότερο mAP που υπολογίστηκε είναι

92,91%. Μετά το τέλος των 6000 επαναλήψεων, η τιμή του Average loss ήταν περίπου 0,8.



Εικόνα 44: Γράφημα Average loss και mAP YOLOv4

Μετά την εκπαίδευση του μοντέλου YOLOv4-tiny δημιουργήθηκε ένα γράφημα όπως φαίνεται στην Εικόνα 45. Οι τιμές του Average loss και οι τιμές του mAP που χρησιμοποιούνται για την αξιολόγηση του μοντέλου αλλάζει πολλές φορές κατά την εκπαίδευση του μοντέλου και αποδεικνύει πόσο ακριβές είναι το μοντέλο . Το σύνολο δεδομένων ξεκίνησε από τις 0 επαναλήψεις και συνεχίστηκε μέχρι τις 6000 επαναλήψεις. Στις 1000 επαναλήψεις υπολογίστηκε το πρώτο mAP 38,85% και φαίνεται ότι το mAP ξεπέρασε το 70% σε περίπου 3000 επαναλήψεις. Κατά τη διάρκεια αυτής της εκπαίδευσης οι τιμές του Average loss και οι τιμές του mAP κυμαίνονται. Η χαμηλότερη τιμή του Average loss είναι περίπου 1,3 και υπολογίστηκε σε περίπου 3600 επαναλήψεις. Το καλύτερο και υψηλότερο mAP που υπολογίστηκε είναι 74,75%. Μετά το τέλος των 6000 επαναλήψεων, η τιμή του Average loss ήταν περίπου 1,34.



Εικόνα 45: Γράφημα Average loss και mAP YOLOv4-tiny

- **<u>Εικόνα</u>**

1. Φόρτωση της εικόνας εισόδου (η εικόνα είναι σε μορφή RGB) και αποθήκευση όλων των διαστάσεών της.
2. Κατασκευή της εικόνας σε μορφή blob (Binary Large Object). Το blob είναι ένας πίνακας τεσσάρων διαστάσεων (4D NumPy array), που αποτελείται από την εικόνα τα κανάλια, το πλάτος και το ύψος και συλλέγει τα δεδομένα αυτά για την εικόνα σε δυαδική μορφή, έτσι ώστε να χρησιμοποιηθούν σε επόμενα βήματα.
3. Φόρτωση του αλγορίθμου YOLOv4 και ορισμός της παραμέτρου κατώφλι (threshold). Θα φιλτράρουμε δηλαδή τα αντικείμενα που δεν πληρούν το κατώφλι (threshold), το οποίο έχει όριο 0,5.
4. Στη συνέχεια συλλέγονται τα αντικείμενα που έχουν εντοπιστεί, στην συγκεκριμένη περίπτωση η χρήση ή όχι της μάσκας για την προστασία από τον κορονοϊό (COVID-19), μαζί με τα αντίστοιχα σκορ εμπιστοσύνης τους. Το επόμενο βήμα είναι να απορριφθούν οι αδύναμες προβλέψεις για τα συγκεκριμένα αντικείμενα με βάση την ελάχιστη πιθανότητα που έχει οριστεί.
5. Είναι γνωστό πως οι αλγόριθμοι ανίχνευσης αντικειμένων όπως και ο YOLOv4, δημιουργούν πολλαπλά πλαίσια οριοθέτησης (bounding boxes). Για κάθε αντικείμενο στην εικόνα πρέπει να έχουμε μόνο ένα πλαίσιο οριοθέτησης. Η

τεχνική Non-maximum suppression επιλέγει το καταλληλότερο πλαίσιο οριοθέτησης σε συνεργασία με την ελάχιστη πιθανότητα και το κατώφλι.

6. Τέλος σχηματίζονται οι κατάλληλες ετικέτες (labels) για τα αντίστοιχα πλαίσια οριοθέτησης (bounding boxes).

Τα τελικά αποτελέσματα για την ανίχνευση της χρήσης μάσκας για την προστασία από τον covid-19 με την βοήθεια των αλγορίθμων YOLOv4 και YOLOv4-tiny παρουσιάζονται παρακάτω.

YOLOv4:


```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 59.570
avg_outputs = 489910
 Allocate additional workspace_size = 18.88 MB
Loading weights from ../training/yolov4-custom_best.weights...
 seen 64, trained: 140 K-images (2 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
../../mask_test_images/image1.jpg: Predicted in 601.593000 milli-seconds.
with_mask: 89%
with_mask: 69%
```

Εικόνα 46: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4

Tiny-YOLOv4:


```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 6.789
avg_outputs = 299797
 Allocate additional workspace_size = 72.42 MB
Loading weights from ../training/yolov4-tiny-custom_best.weights...
 seen 64, trained: 300 K-images (4 Kilo-batches_64)
Done! Loaded 38 layers from weights-file
 Detection layer: 30 - type = 28
 Detection layer: 37 - type = 28
../../mask_test_images/image1.jpg: Predicted in 369.843000 milli-seconds.
with_mask: 86%
with_mask: 34%
```

Εικόνα 47: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny

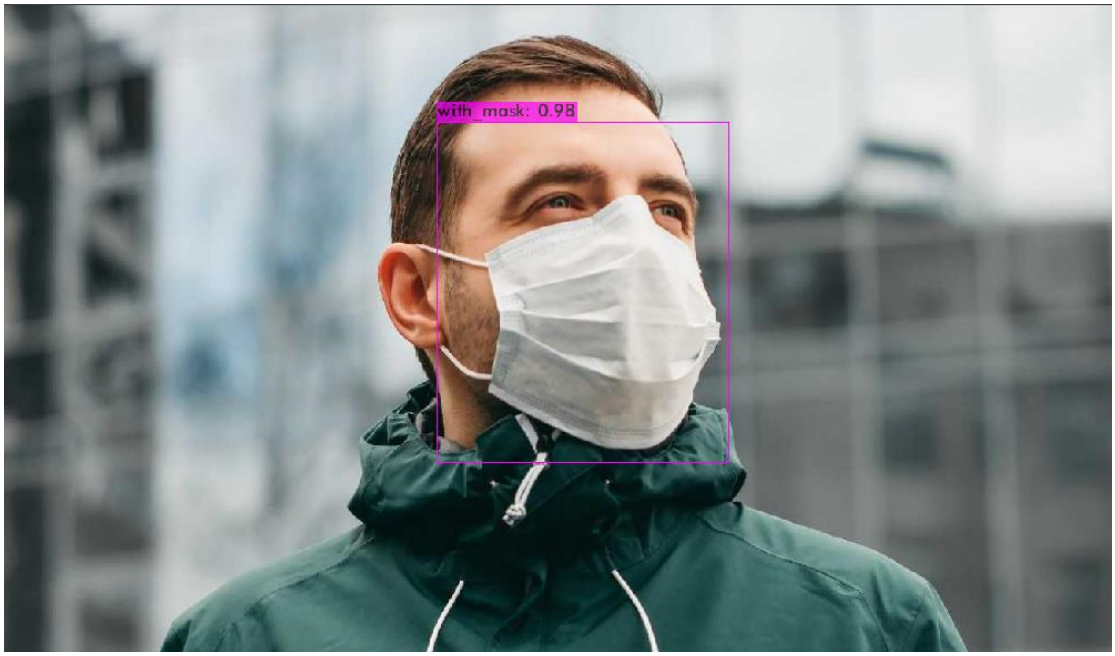
Input image: image1	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=89% with_mask=69%	with_mask=86% with_mask=34%
Objects Detected	2	2

YOLOv4:


```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 59.570
avg_outputs = 489910
 Allocate additional workspace_size = 18.88 MB
Loading weights from ../training/yolov4-custom_best.weights...
 seen 64, trained: 140 K-images (2 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
../../mask_test_images/maskdude.jpg: Predicted in 391.757000 milli-seconds.
with_mask: 95%
```

Εικόνα 48: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4

Tiny-YOLOv4:




```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 6.789
avg_outputs = 299797
 Allocate additional workspace_size = 72.42 MB
Loading weights from ../training/yolov4-tiny-custom_best.weights...
 seen 64, trained: 300 K-images (4 Kilo-batches_64)
Done! Loaded 38 layers from weights-file
 Detection layer: 30 - type = 28
 Detection layer: 37 - type = 28
../../mask_test_images/maskdude.jpg: Predicted in 365.955000 milli-seconds.
with_mask: 98%
```

Εικόνα 49: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny

Input image: maskdude	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=95%	with_mask=98%
Objects Detected	1	1

YOLOv4:


```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 59.570
avg_outputs = 489910
 Allocate additional workspace_size = 18.88 MB
Loading weights from ../training/yolov4-custom_best.weights...
 seen 64, trained: 140 K-images (2 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
../../mask_test_images/image3.jpg: Predicted in 428.940000 milli-seconds.
with_mask: 84%
```

Εικόνα 50: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4

Tiny-YOLOv4:

Εικόνα 51: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny

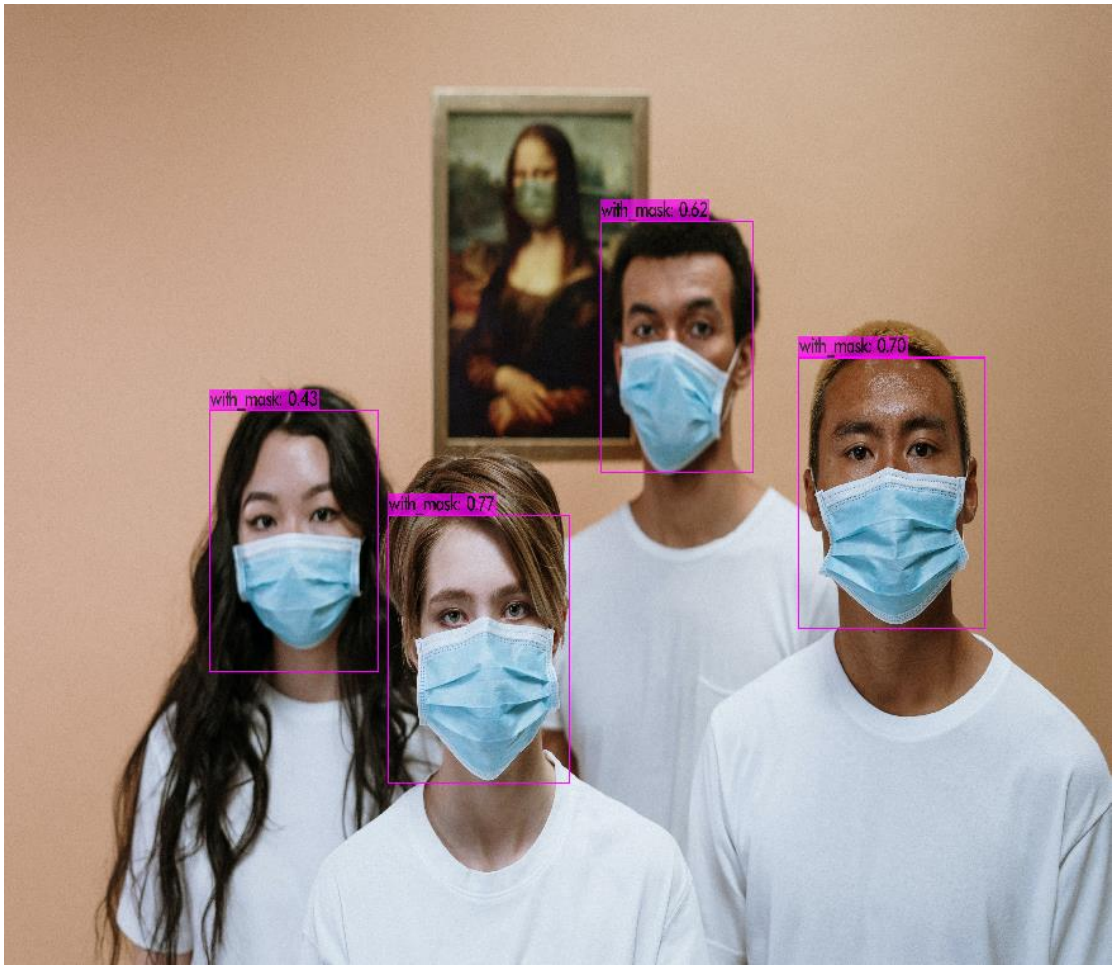
Input image: image3	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=84%	with_mask=76%
Objects Detected	1	1

YOLOv4:


```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 59.570
avg_outputs = 489910
 Allocate additional workspace_size = 18.88 MB
Loading weights from ../training/yolov4-custom_best.weights...
 seen 64, trained: 140 K-images (2 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
../../mask_test_images/image4.jpg: Predicted in 395.509000 milli-seconds.
with_mask: 94%
with_mask: 94%
with_mask: 44%
with_mask: 84%
with_mask: 95%
```

Εικόνα 52: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4

Tiny-YOLOv4:




```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 6.789
avg_outputs = 299797
 Allocate additional workspace_size = 72.42 MB
Loading weights from ../training/yolov4-tiny-custom_best.weights...
 seen 64, trained: 300 K-images (4 Kilo-batches_64)
Done! Loaded 38 layers from weights-file
 Detection layer: 30 - type = 28
 Detection layer: 37 - type = 28
../../mask_test_images/image4.jpg: Predicted in 362.348000 milli-seconds.
with_mask: 43%
with_mask: 77%
with_mask: 62%
with_mask: 70%
```

Εικόνα 53: Αποτελέσματα ανιχνεύσεων σε εικόνα YOLOv4-tiny

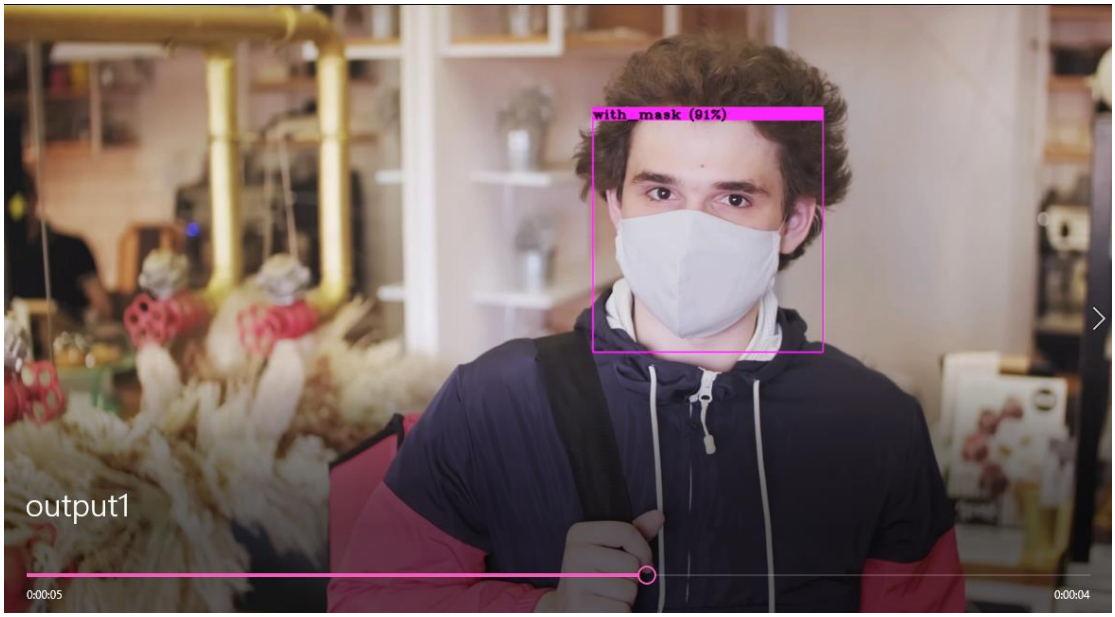
Input image: image4	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=94% with_mask=94% with_mask=44% with_mask=984% with_mask=95%	with_mask=43% with_mask=77% with_mask=62% with_mask=70%
Objects Detected	5	4

- **Βίντεο**

1. Εισαγωγή του βίντεο που θέλουμε να επεξεργαστούμε και να κάνουμε ανίχνευση της χρήσης μάσκας για την προστασία έναντι του covid-19.
2. Επιλογή του αλγορίθμου YOLOv4 και Tiny-YOLOv4.
3. Κατασκευή του στιγμιότυπου του βίντεο σε μορφή blob.
4. Εφαρμογή της διαδικασίας forward-pass.
5. Εφαρμογή των bounding boxes.
6. Εφαρμογή των Non-maximum suppression.
7. Σχηματισμός των κατάλληλων ετικετών (labels) για τα αντίστοιχα πλαίσια οριοθέτησης (bounding boxes).
8. Εγγραφή των νέων στιγμιότυπων σε νέο βίντεο και αποθήκευση σε μορφή .avi. Το νέο βίντεο θα είναι ίδιο με το πρωταρχικό με την προσθήκη ετικετών στα αντίστοιχα πλαίσια οριοθέτησης για τα αντικείμενα που ανιχνεύτηκαν.

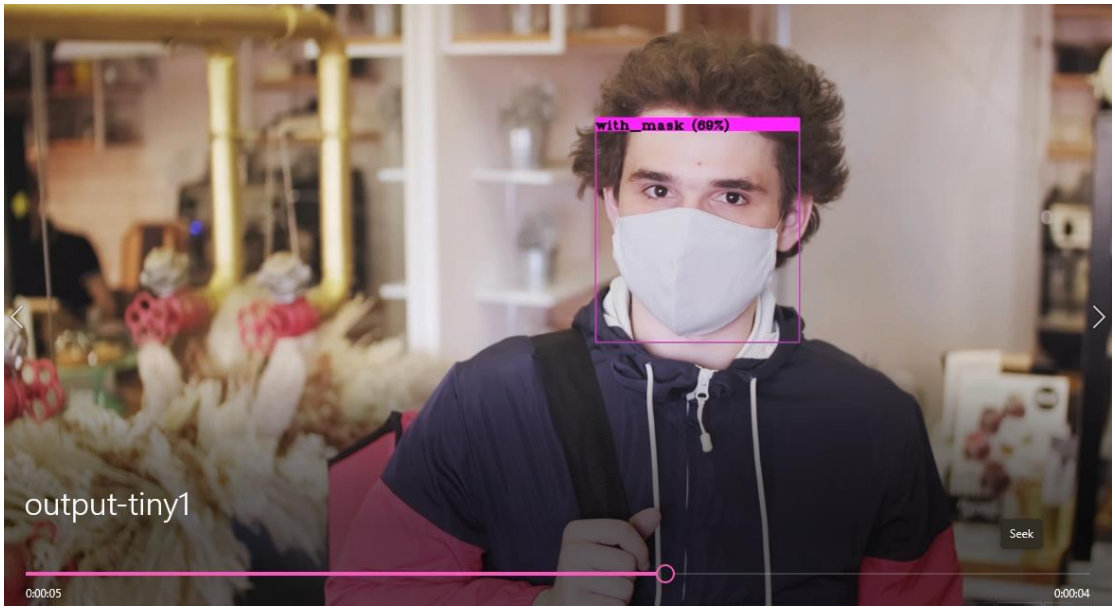
Στη συνέχεια θα εξετάσουμε και θα συγκρίνουμε την απόδοση των δύο αλγορίθμων YOLOv4 και YOLOv4-tiny. Η σύγκριση θα γίνει μεταξύ των ίδιων στιγμιότυπων για το κάθε βίντεο. Στην περίπτωση των βίντεο παρατηρείται πως ο YOLOv4-tiny μπορεί να παρουσιάζει μικρότερο ποσοστό ακρίβειας για τη χρήση ή όχι μάσκας, αλλά είναι πιο γρήγορος στη διαδικασία της ανίχνευσης. Τα αποτελέσματα της ανίχνευσης με YOLOv4 και YOLOv4-tiny σε στιγμιότυπα των βίντεο παρουσιάζονται παρακάτω.

YOLOv4:



Εικόνα 54: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4

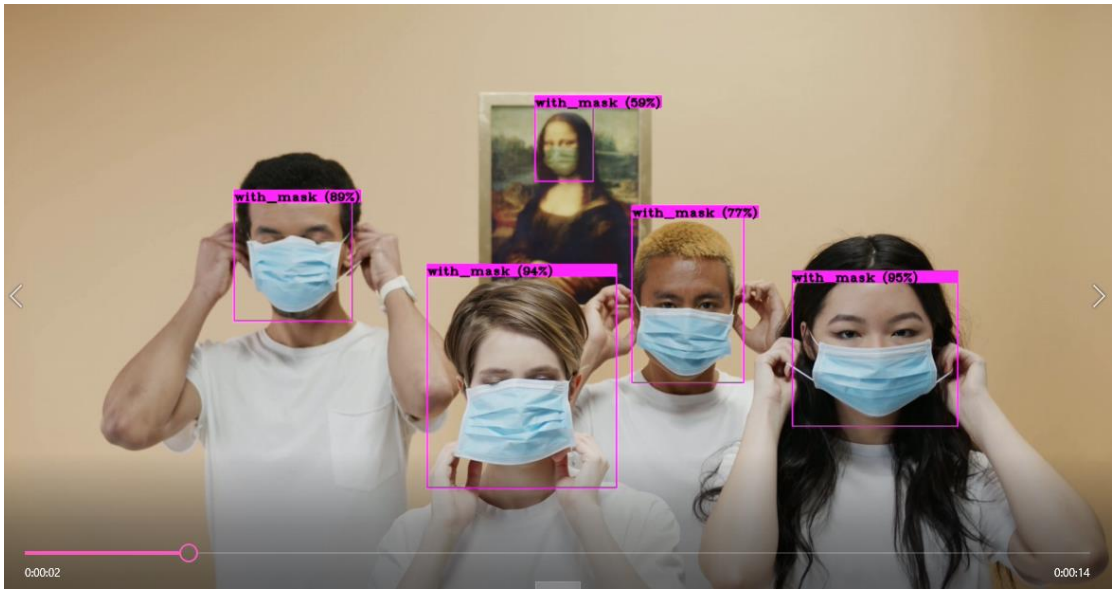
Tiny-YOLOv4:



Εικόνα 55: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4-tiny

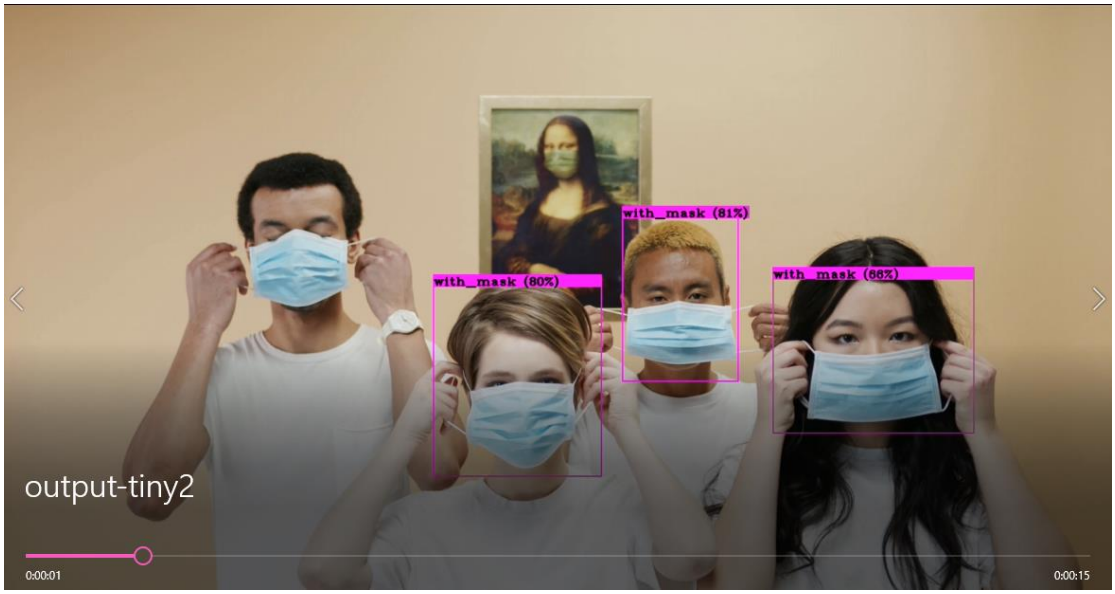
Input video: video1	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=91%	with_mask=69%
Objects Detected	1	1
Time	17.2	17

YOLOv4:



Εικόνα 56: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4

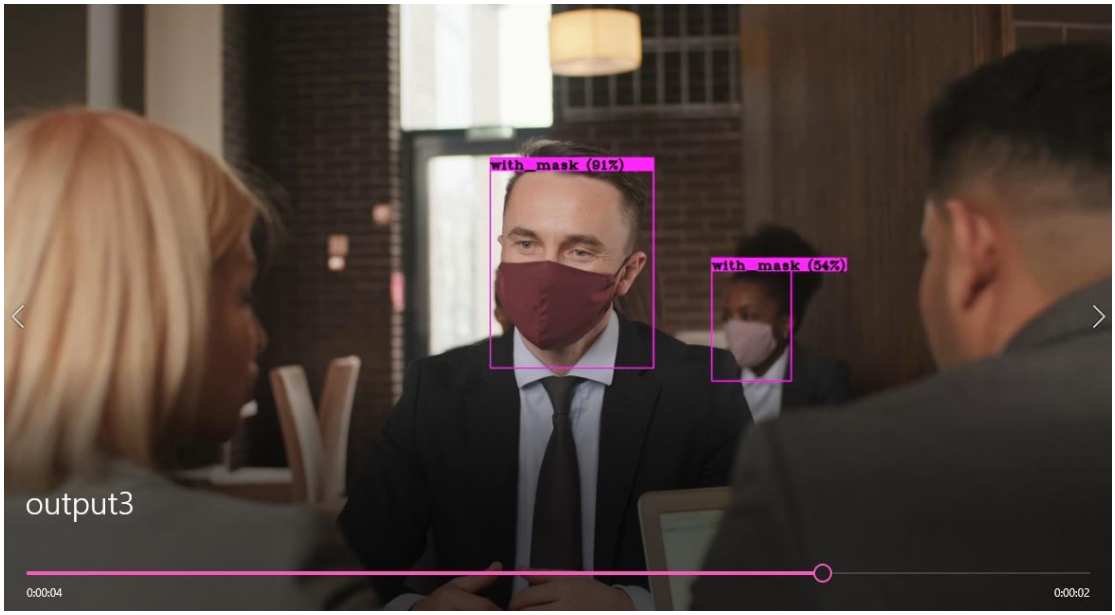
Tiny-YOLOv4:



Εικόνα 57: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4-tiny

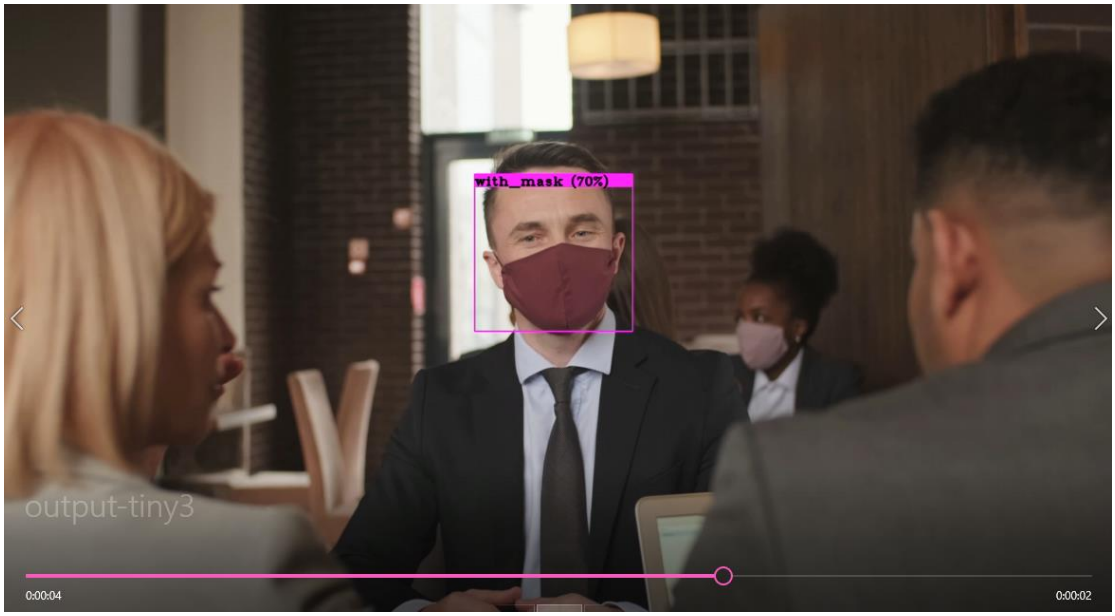
Input video: video2	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=89% with_mask=94% with_mask=59% with_mask=77% without_mask=95%	with_mask=80% with_mask=81% with_mask=66%
Objects Detected	5	3
Time	16.8	17.9

YOLOv4:



Εικόνα 58: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4

Tiny-YOLOv4:



Εικόνα 59: Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4-tiny

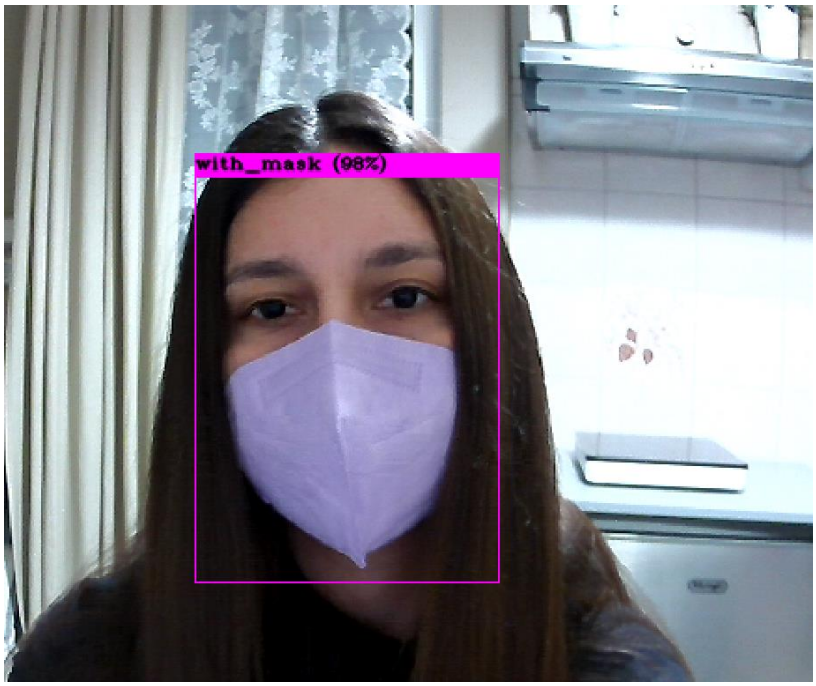
Input video: video3	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=91% with_mask=54%	with_mask=70%
Objects Detected	2	1
Time	16	17.6

- **Real-time video**

Η τελευταία κατηγορία που θα μας απασχολήσει είναι αυτή του Real-time video. Η κάμερα που είναι συνδεδεμένα στο σταθερό υπολογιστή ή στο λάπτοπ δημιουργεί στιγμιότυπα πραγματικού χρόνου σε βίντεο, τα οποία δίνει σαν είσοδο στον αλγόριθμο YOLOv4 και Tiny-YOLOv4. Έτσι φαίνεται στα στιγμιότυπα που λαμβάνουμε σε πραγματικό χρόνο η ακρίβεια του κάθε αντικειμένου μαζί με το αντίστοιχο πλαίσιο οριοθέτησης. Στην περίπτωση αυτή το βίντεο δεν αποθηκεύεται σε κάποια μορφή, αλλά βλέπουμε τα αποτελέσματα της ανίχνευσης σε βίντεο πραγματικού χρόνου.

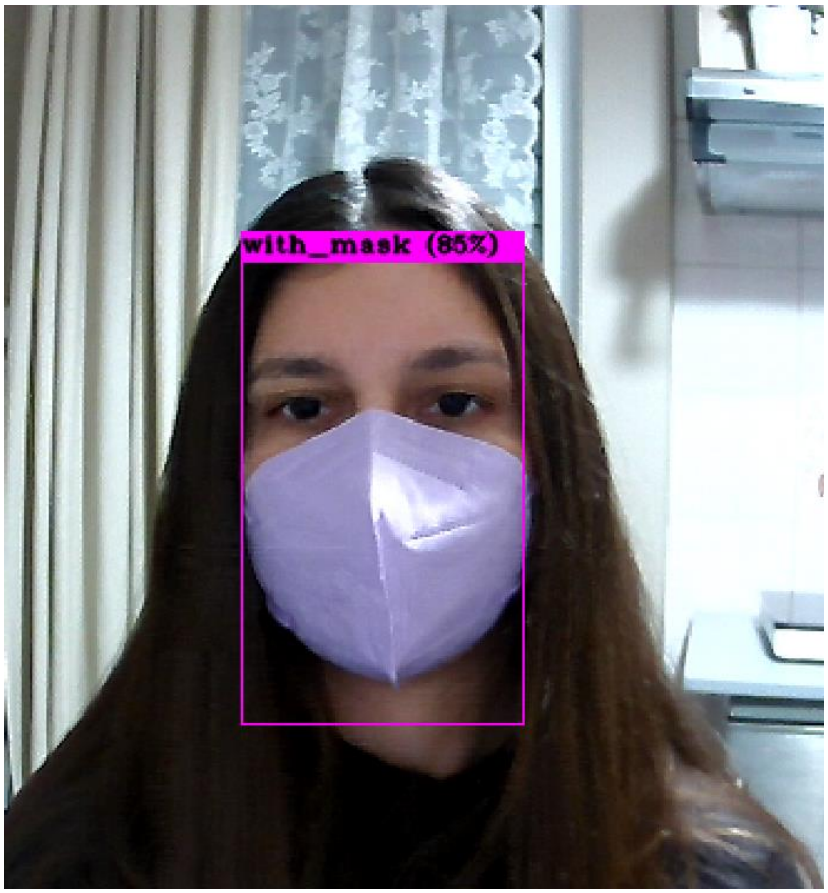
Τα αποτελέσματα της ανίχνευσης για την χρήση ή όχι μάσκας με την εφαρμογή των αλγορίθμων YOLOv4 και YOLOv4-tiny παρουσιάζονται στη συνέχεια. Για ευνόητους λόγους δεν είναι δυνατή η λήψη ίδιων στιγμιότυπων για την σύγκριση των YOLOv4 και YOLOv4-tiny, όπως στην περίπτωση των βίντεο.

Yolov4:



Εικόνα 60: Αποτελέσματα ανιχνεύσεων σε real-time βίντεο YOLOv4 (with mask)

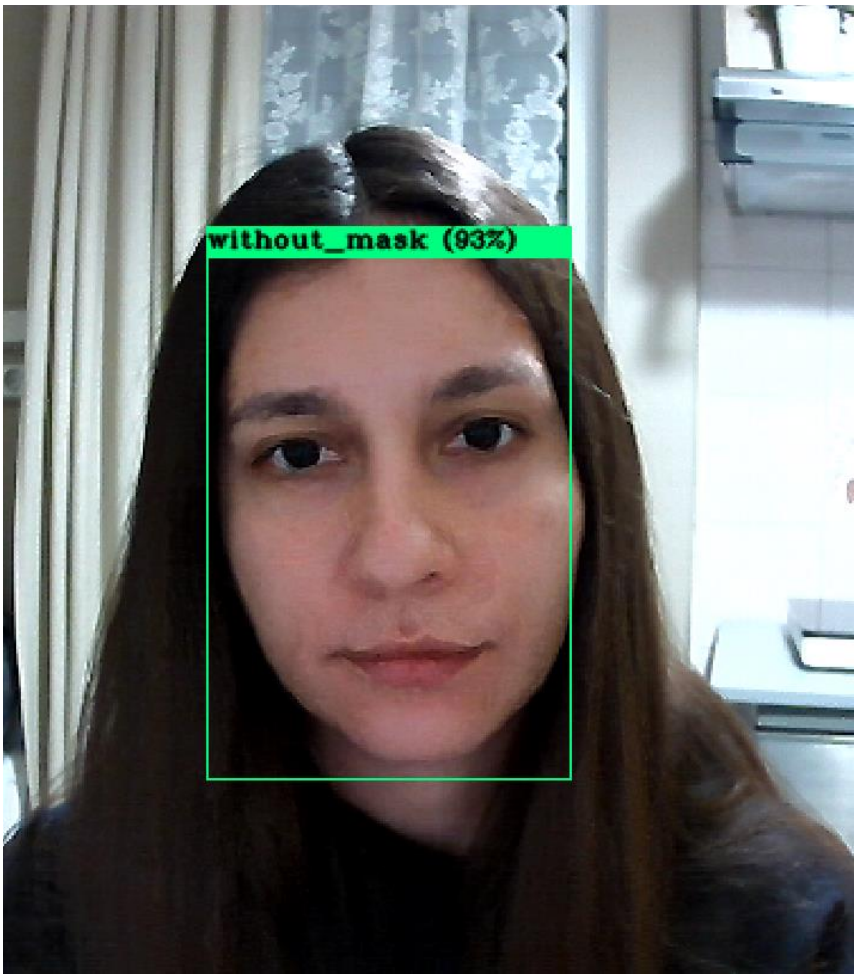
Tiny-yolov4:



Εικόνα 61: Αποτελέσματα ανιχνεύσεων σε real-time βίντεο YOLOv4-tiny (with mask)

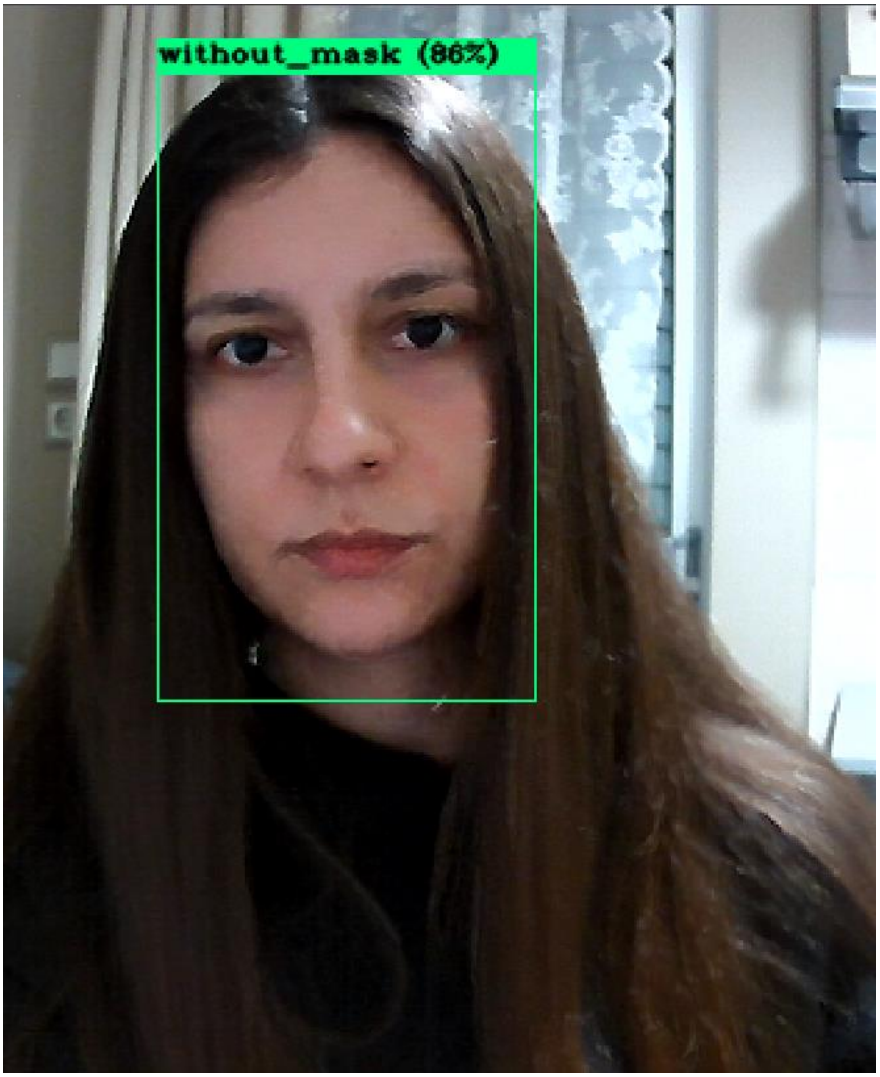
Input video: real time video	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=98%	with_mask=85%
Objects Detected	1	1
Time	6.3	6.5

Yolov4:



Εικόνα 62:Αποτελέσματα ανιχνεύσεων σε βίντεο YOLOv4 (without mask)

Tiny-yolov4:



Εικόνα 63: Αποτελέσματα ανιχνεύσεων σε real-time βίντεο YOLOv4-tiny (without mask)

Input video: real time video	YOLOv4	Tiny-YOLOv4
Accuracy	with_mask=93%	with_mask=93%
Objects Detected	1	1
Time	6.3	6.5

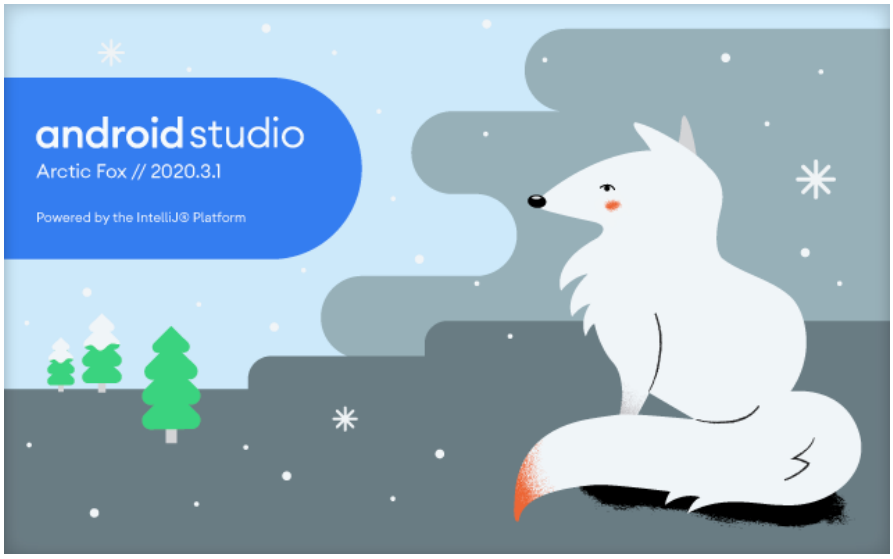
7.1 Real-time ανίχνευση με Mobile App με χρήση Android Studio

Αυτή η ενότητα της εργασίας αναφέρεται στη δοκιμή της απόδοσης του μοντέλου σε κινητές συσκευές. Αυτό μπορεί να επιτευχθεί με τη μετατροπή των μοντέλων ανίχνευσης αντικειμένων YOLOv4 και YOLOv4-tiny σε TensorFlow Lite (tflite) για την ανάπτυξή τους σε φορητές συσκευές. Η εισαγωγή των μοντέλων YOLOv4 TFlite και YOLOv4-tiny TFlite θα πραγματοποιηθεί με χρήση της πλατφόρμας Android Studio.

Android Studio

Το Android Studio είναι το επίσημα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την ανάπτυξη εφαρμογών Android, το οποίο κυκλοφόρησε στις 16 Μαΐου 2013, κατά τη διάρκεια της εκδήλωσης I/O 2013 της Google . Βασίζεται στο IntelliJ IDEA, ένα ολοκληρωμένο περιβάλλον ανάπτυξης Java για λογισμικό, και ενσωματώνει την επεξεργασία κώδικα και τα διάφορα εργαλεία προγραμματιστών.

Για να υποστηρίξει την ανάπτυξη εφαρμογών εντός του λειτουργικού συστήματος Android, το Android Studio χρησιμοποιεί ένα σύστημα κατασκευής που βασίζεται σε Gradle, έναν εξομοιωτή, τα πρότυπα κώδικα και την ενοποίηση Github. Κάθε έργο στο Android Studio έχει έναν ή περισσότερους τρόπους με πηγαίο κώδικα και αρχεία πόρων. Αυτές οι λειτουργίες περιλαμβάνουν λειτουργικές μονάδες εφαρμογών Android, λειτουργικές μονάδες βιβλιοθήκης και λειτουργικές μονάδες Google App Engine.



Εικόνα 64: Πλατφόρμα Android Studio

Tensorflow

Το TensorFlow παρέχει μια συλλογή ροών εργασίας για την ανάπτυξη και την εκπαίδευση μοντέλων χρησιμοποιώντας Python ή JavaScript και για εύκολη ανάπτυξη στο cloud, στο on-prem, σε διάφορα προγράμματα περιήγησης ή σε διάφορες κινητές συσκευές, ανεξάρτητα από τη γλώσσα που έχει χρησιμοποιηθεί. Το tf.data API δίνει τη δυνατότητα δημιουργίας πολύπλοκων αγωγών εισόδου από απλά, επαναχρησιμοποιήσιμα κομμάτια.

TensorFlow Lite

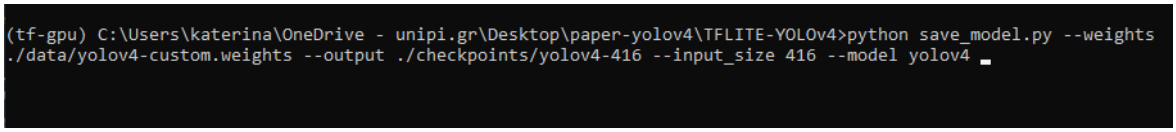
Το TensorFlow Lite είναι ένα πλαίσιο βαθιάς εκμάθησης ανοιχτού κώδικα που μετατρέπει ένα προ-εκπαιδευμένο μοντέλο στο TensorFlow σε μια ειδική μορφή που μπορεί να βελτιστοποιηθεί για ταχύτητα και αποθήκευση. Αυτή η ειδική μορφή είναι ιδανική για ανάπτυξη σε ενσωματωμένες συσκευές όπως κινητά που χρησιμοποιούν Android ή iOS.

Αφού το μοντέλο εκπαιδευτεί και μετατραπεί σε μορφή TensorFlow, θα μετατραπεί στην έκδοση TensorFlow Lite. Η έκδοση TensorFlow Lite συνδυάζει ικανοποιητική ακρίβεια και ταυτόχρονα καταλαμβάνει λιγότερο χώρο. Αυτές οι ιδιότητες καθιστούν το μοντέλο TF Lite κατάλληλο για ανάπτυξη σε κινητές συσκευές και ενσωματωμένες συσκευές.

Για να μπορέσει το μοντέλο να ενσωματωθεί σε κινητή συσκευή, χωρίς να χρειάζεται κανένας επιπλέον εξοπλισμός, θα πρέπει να πραγματοποιηθεί η εκτέλεση κάποιον συγκεκριμένων ενεργειών. Αρχικά θα πρέπει να δημιουργήσουμε το κατάλληλο περιβάλλον έτσι ώστε να εκμεταλλευτεί αναλόγως η GPU, όπως φαίνεται παρακάτω.

Εικόνα 65: Δημιουργία κατάλληλου περιβάλλοντος σε gpu

Το επόμενο βήμα είναι η μετατροπή του αρχείου που περιέχει τα βάρη του κάθε μοντέλο σε μορφή tensorflow. Αφού έχει γίνει η μετατροπή των βαρών για το μοντέλο YOLOv4 και YOLOv4-tiny αντίστοιχα σε tensorflow, θα αποθηκευτούν σε αυτή τη μορφή και ύστερα θα μετατραπούν στη μορφή tflite. Έπειτα, στη μορφή tflite των βαρών θα εφαρμοστεί κβαντοποίηση σε float16, έτσι ώστε να τα βάρη για τον YOLOv4 και YOLOv4-tiny να είναι στην κατάλληλη μορφή για να ενσωματωθούν στην κινητή συσκευή.



Εικόνα 66: Μετατροπή σε TensorFlow

```
(tf-gpu) C:\Users\katerina\OneDrive - unipi.gr\Desktop\paper-yolov4\TFLITE-YOLOv4>python save_model.py --weights
./data/yolov4-custom.weights --output ./checkpoints/yolov4-416 --input_size 416 --model yolov4 --framework tflite
```

Εικόνα 67: Αποθήκευση TensorFlow μοντέλου

```
(tf-gpu) C:\Users\katerina\OneDrive - unipi.gr\Desktop\paper-yolov4\TFLITE-YOLOv4>python convert_tflite.py
--weights ./checkpoints/yolov4-416 --output ./checkpoints/yolov4-416.tflite
```

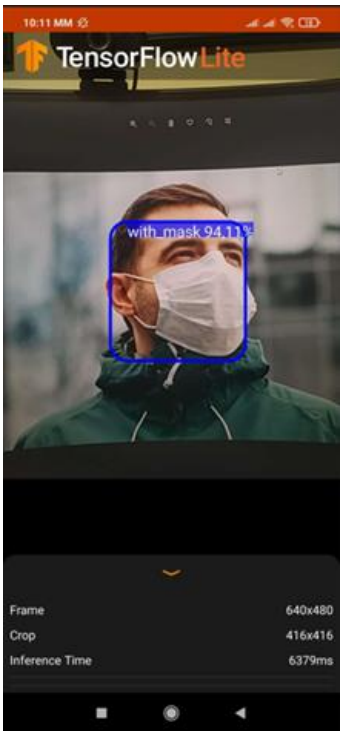
Εικόνα 68: Μετατροπή σε tflite

```
(tf-gpu) C:\Users\katerina\OneDrive - unipi.gr\Desktop\paper-yolov4\TFLITE-YOLOv4>python convert_tflite.py
--weights ./checkpoints/yolov4-416 --output ./checkpoints/yolov4-416-fp16.tflite --quantize_mode float16
```

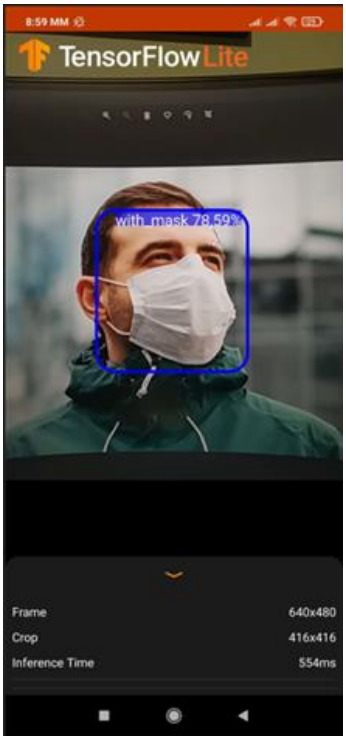
Εικόνα 69: Quantize float16

Τέλος τα βάρη του μοντέλου YOLOv4 και YOLOv4-tiny, θα εφαρμοστούν στην τελική μορφή τους στην αντίστοιχη εφαρμογή που έχει δημιουργηθεί και μέσω της πλατφόρμας Android Studio θα εγκατασταθούν στην κινητή συσκευή.

Όπως παρατηρήσαμε και στην προηγούμενη ενότητα η έκδοση YOLOv4-tiny είναι καταλληλότερη για την ανίχνευση των αντικειμένων σε πραγματικό χρόνο λόγω του χρόνου στον οποίο πραγματοποιείται ο εντοπισμός των αντικειμένων, ενώ παράλληλα η ακρίβεια διατηρείται σε ικανοποιητικά επίπεδα. Γεγονός το οποίο παρατηρείται και στις παρακάτω δοκιμές.



Εικόνα 70: Real time ανίχνευση σε Android συσκευή YOLOv4



Εικόνα 71: Εικόνα 60: Real time ανίχνευση σε Android συσκευή YOLOv4-tiny

Συμπεράσματα

Σε αυτήν την διπλωματική εργασία επιλέξαμε δύο πολύπλοκα, βασισμένα σε νευρωνικά δίκτυα μοντέλα βαθιάς μάθησης για το έργο της ανίχνευσης αντικειμένων. Για να εξειδικεύσουμε την εργασία στον εντοπισμό αντικειμένων που σχετίζονται με μέτρα προσωπικής υγιεινής, τα εκπαιδεύσαμε σε μάσκες προσώπου κ.λπ. Τα δύο επιλεγμένα μοντέλα είναι μια παραλλαγή της ίδιας οικογένειας μοντέλων με το YOLOv4 σε σύγκριση με το YOLOv4-tiny. Μετά την εκτέλεση πειραμάτων που πραγματοποιήθηκαν και σε κινητές συσκευές, τα πειραματικά αποτελέσματα έδειξαν πως και τα δύο μοντέλα και στις δύο περιπτώσεις παραμένουν αποτελεσματικά τόσο σε ακρίβεια, όσο και σε χρόνο. Αυτό σημαίνει ότι ο τύπος μοντέλου YOLOv4-tiny είναι πιο κατάλληλος για βίντεο σε πραγματικό χρόνο και ενσωματωμένες συσκευές λόγω του χρόνου για τον εντοπισμό αντικειμένων, αλλά το μοντέλο YOLOv4 εξακολουθεί να είναι καλό λαμβάνοντας υπόψη τα υψηλά επίπεδα ακρίβειας που προσφέρει στον εντοπισμό αντικειμένων.

Βιβλιογραφία

[1] A. I. B. P. a. T. Ahamed, "Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT," *Sensors,* p. 32, 2021.

[2] J. Y. a. W. Zhang, "Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4," *Sensors,* p. 21, 2021.

[3] A. K. A. S. M. K. Akhil Kumar, "A hybrid tiny YOLO v4-SPP module based improved face mask detection vision system," *Journal of Ambient Intelligence and Humanized Computing,* p. 14, 2021.

[4] S. I. N. Shraddha Sanjeev Pattanshetti, "Real-Time Object Detection with Pre-eminent Speed and Precision using YOLOv4," *International Journal of Research in Engineering, Science and Management,* vol. 4, no. 7, p. 6, 2021.

[5] M. R. Sarah Hraybi, "Examining YOLO for real-time face-mask detection," p. 6, 2021.

[6] Z. C. B. W. M. L. Xiaoyu Wang, "Application of Pruning Yolo-V4 with Center Loss in Mask Wearing Recognition for Gymnasiums and Sports Grounds of Colleges and Universities," *IEEE 6th International Conference on Computer and Communications,* p. 5, 2020.

[7] L. Z. S. L. Y. J. ZICONG JIANG, "Real-time object detection method for embedded devices," p. 11.

[8] H. L. G. Y. W. N. Y. L. X. T. B. R. Y. W. Yuxuan Cai, "YOLObile: Real-Time Object Detection on Mobile Devices via Compression-Compilation Co-Design," p. 10, 2020.

[9] M. J. Xinqi Fan, "RetinaFaceMask: A Single Stage Face Mask Detector for Assisting Control of the COVID-19 Pandemic," p. 6, 2021.

[10] S. S. S. G. F. B. A. A. S. M. Elliot Mbunge, "Application of deep learning and machine learning models to detect COVID-19 face masks - A review," *KeAi,* p. 11, 2021.

[11] H. L. N. Z. F. L. Peishu Wu, "FMD-Yolo: An efficient face mask detection method for COVID-19 prevention and control in public," *elsevier,* p. 10, 2022.

[12] J. P. F. X. J. Z. Q. L. X. H. ,. L. ,. W. Xu Li, "Fast and accurate green pepper detection in complex backgrounds via an improved Yolov4-tiny model," *elsevier,* p. 10, 2021.

[13] A. K. K. V. A. S. M. K. Akhil Kumar, "Scaling up face masks detection with YOLO on a novel dataset," *elsevier,* p. 15, 2021.

[14] R. J. a. A. M. a. R. A. a. P. K. a. J. H. b. Preeti Nagrath, "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," *elsevier,* p. 13, 2020.

[15] C.-Y. W. H.-Y. M. L. Alexey Bochkovskiy, "YOLOv4: Optimal Speed and Accuracy of Object Detection," p. 17, 2020.

[16] A. K. A. S. M. K. Akhil Kumar, "A hybrid tiny YOLO v4-SPP module based improved face mask detection vision system," *Journal of Ambient Intelligence and Humanized Computing ,* p. 14, 2021.

[17] S. N. D. G. D. D. P. B. T. D. Biparnak Roy, "MOXA: A Deep Learning Based Unmanned Approach For Real-Time Monitoring of People Wearing Medical Masks," *Springer,* p. 10, 2020.

[18] G. M. M. H. N. T. N. E. M. K. Mohamed Loey, "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection," *elsevier,* p. 8, 2020.

[19] A. H. R. S. S. Adban Akib Protik, "Real-time Personal Protective Equipment (PPE) Detection Using YOLOv4 and TensorFlow," p. 7, 2021.

[20] R. K. M. K. N. K. C. P. Sharjeel Anjum, "A Pull-Reporting Approach for Floor Opening Detection Using Deep-Learning on Embedded Devices," p. 9, 2021.

[21] P. G. P. S. a. J. P. Pooja Mahto, "REFINING YOLOV4 FOR VEHICLE DETECTION," *IJARET,* vol. 11, no. 5, pp. 409-419, 2020.

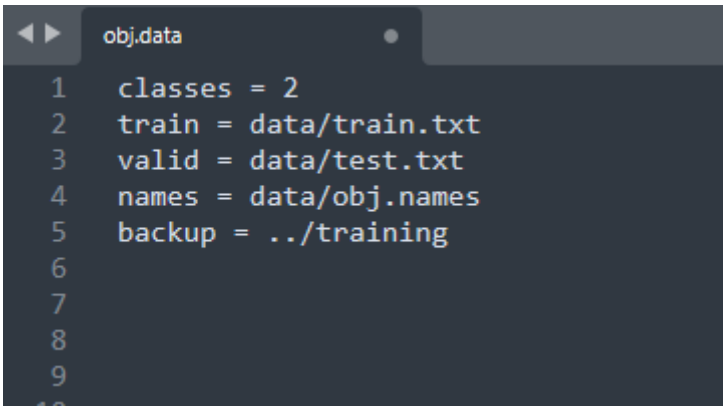
[22] R.-C. C. Y.-T. L. J. K. D. H. CHRISTINE DEWI, "Yolo V4 for Advanced Traffic Sign Recognition With Synthetic Training Data Generated by Various GAN," *IEEEAceess,* p. 15, 2021.

[23] J. S. J. J. Shu Liu† Lu Qi† Haifang Qin, "Path Aggregation Network for Instance Segmentation," p. 11, 2018.

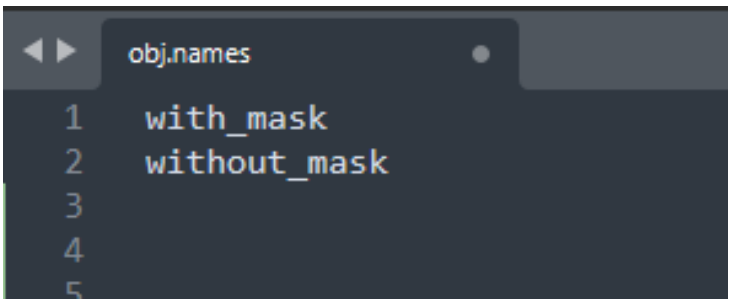
[24] 1. K. H. ,. M. X. L. ,. T. Z. ,. X. a. S. C. Zuopeng Zhao, "SAI-YOLO: A Lightweight Network for Real-Time Detection of Driver Mask-Wearing Specification on Resource-Constrained Devices," *Hindawi,* vol. 2021, p. 15, 2021.

[25] C. D. ,. Z. J. ,. M. G. a. Z. H. Han Wu, "SORT-YM: An Algorithm of Multi-Object Tracking with YOLOv4-Tiny and Motion Prediction," p. 20, 2021.

Appendix



obj.data αρχείο



obj.names αρχείο

process.py

```python
import glob, os

# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)

current_dir = 'data/obj'

# Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')

# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpg")):
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:
        counter = 1
        file_test.write("data/obj" + "/" + title + '.jpg' + "\n")
    else:
        file_train.write("data/obj" + "/" + title + '.jpg' + "\n")
        counter = counter + 1

```

process.py αρχείο

yolov4-custom.cfg

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

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

```

YOLOv4 cfg αρχείο

yolov4-custom.cfg

```
966
967   [yolo]
968   mask = 0,1,2
969   anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
970   classes=2
971   num=9
972   jitter=.3
973   ignore_thresh = .7
974   truth_thresh = 1
975   scale_x_y = 1.2
976   iou_thresh=0.213
977   cls_normalizer=1.0
978   iou_normalizer=0.07
979   iou_loss=ciou
980   nms_kind=greedynms
981   beta_nms=0.6
982   max_delta=5
983
```

YOLOv4 cfg αρχείο

yolov4-tiny-custom.cfg

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

learning_rate=0.00261
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1

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

YOLOv4-tiny cfg αρχείο

yolov4-tiny-custom.cfg

```
[yolo]
mask = 3,4,5
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=2
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
resize=1.5
nms_kind=greedynms
beta_nms=0.6
```

YOLOv4-tiny cfg αρχείο