

Ιόνιο Πανεπιστήμιο

Τμήμα Πληροφορικής



Πτυχιακή Εργασία

Εφαρμογή μηχανικής μάθησης για την αναγνώριση στοιχείων-
τοπίου από φωτογραφικό υλικό

Γιώργος Ηλιόπουλος(Π2016006)

Επιβλέπων: Μιχαήλ Στεφανιδάκης

Επιβλέπων

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής

Τριμελής Επιτροπή

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής

Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής

Κάτια-Λυδία Κερμανίδου, Αναπληρώτρια Καθηγήτρια,
Τμήμα Πληροφορικής

Περίληψη

Η παρούσα εργασία αφορά στη δημιουργία μιας εφαρμογής τεχνητής νοημοσύνης που σχετίζεται με την αναγνώριση των αντικειμένων και των προσώπων σε μια φωτογραφία, το λεγόμενο image specification. Το image specification θεωρείται εφαρμογή μηχανικής μάθησης, καθώς αφενός είναι απαραίτητη η ύπαρξη ενός μοντέλου, το οποίο θα μπορεί να αναλύει την φωτογραφία που ανεβάζει ο χρήστης, αφετέρου το ίδιο μοντέλο εκπαιδεύεται με σκοπό να επιστρέφει το επιθυμητό αποτέλεσμα. Ένα τέτοιο εργαλείο βοηθάει μία εφαρμογή να προωθήσει διαφημίσεις με βάση το περιεχόμενο που κοινοποιεί ένας χρήστης. Αν, δηλαδή, ένας χρήστης ανεβάσει 10 φωτογραφίες και στις 8 φαίνεται, επί παραδείγματι, να πίνει καφέ, τότε στο προφίλ του θα μπορούσε να εμφανιστεί κάποια διαφήμιση καφέ. Με αυτό τον τρόπο μια εταιρία επιτυγχάνει και τη διαφήμιση των προϊόντων της αλλά και τη συλλογή στατιστικών στοιχείων που θα μπορούσαν πιθανόν να φανούν χρήσιμα στο μέλλον.

Η δομή της παρακάτω εφαρμογής υπαγορεύτηκε από τα εξής τρία τμήματα: 1) front-end, 2) back-end, 3) database. Το πρόγραμμα είναι γραμμένο σε python μια και είναι η πιο γνωστή γλώσσα σε ό,τι έχει να κάνει με μηχανική μάθηση ενώ το front-end δημιουργεί η βιβλιοθήκη Flask. Όσον αφορά στην επικοινωνία με τη βάση δεδομένων (database), χρησιμοποιήθηκε η SQLAlchemy, που είναι επίσης βιβλιοθήκη της Flask.

Abstract

The present work concerns the creation of an application related to artificial intelligence, that aims to the identification of objects and persons from a photograph, i.e. image specification. Image specification is also considered an application of machine learning, employing a model that analyzes the photos that the user will take. This model must be trained so as to output the desired result. Such a tool could help an application to promote ads based on photos the user uploads: if for example a user uploads 10 photos and in 8 of them seems to be drinking coffee, then in his profile can appear an ad about coffee. In this way, a company can advertise products and extract measurements that could be used for further research.

The project's structure was implemented in three modules: 1) front-end, 2) back-end, 3) database. The application is written in python, since it is the most popular language when it comes to machine learning and Flask was used to drive the front-end. As for communicating with the database, SQLAlchemy which is a Flask library was used.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την παρέα μου που με βοήθησε για την πραγμάτωση της πτυχιακής μου εργασίας, παρέχοντάς μου συμβουλές και τρόπους αντιμετώπισης προβλημάτων στο κομμάτι της εφαρμογής. Επίσης, θα ήθελα να ευχαριστήσω την κοπέλα μου που με βοήθησε στη σύνταξη και διόρθωση της πτυχιακής αλλά και μία φίλη που είχε αναλάβει παρόμοιο ρόλο. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την συμπαράσταση και την συνολική βοήθεια και στήριξη που δέχτηκα όλα αυτά τα χρόνια, σε όλη τη διάρκεια της ακαδημαϊκής μου πορείας.

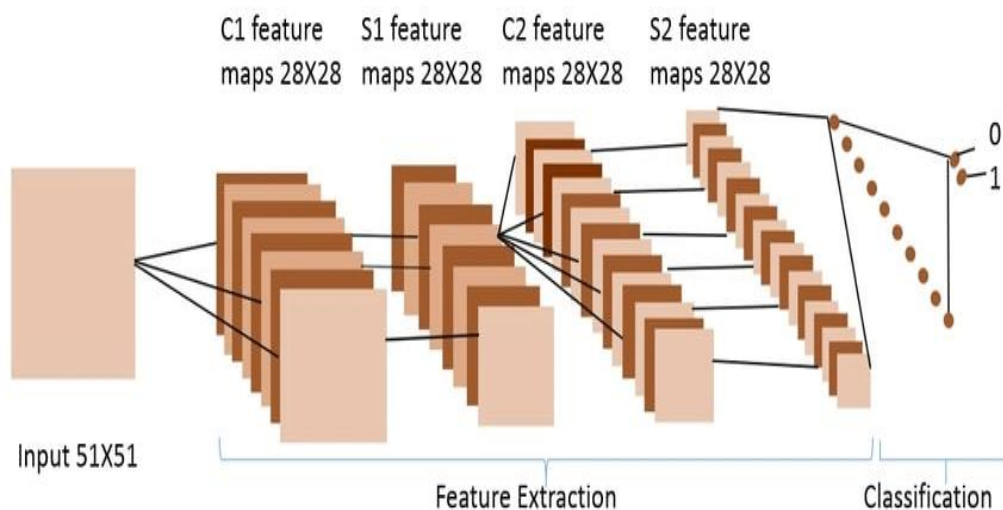
Περιεχόμενα

1. Εισαγωγή	7
2. Θεωρία	10
2.1 Εισαγωγή στα νευρωνικά δίκτυα	10
2.1.1 Τύπος Μάθησης	12
2.2 Μοντέλο και κλάση	14
2.2.1 Convolutional Neural Network	15
Receptive Fields	16
Weights	17
Bias	17
Pooling layers	18
Classification Layer	19
2.3 Image Processing Layer	19
2.4 Αλγόριθμοι/ Detectors	20
2.4.1 Fast R-CNN	20
2.4.2 YOLOv3	21
2.4.3 YOLOv3 Vs Fast R-CNN	22
2.5 Βιβλιοθήκες και εργαλεία	27
2.5.1 Machine Learning Libraries	28
2.5.2 Front-End	29
2.5.3 Back-End	29
2.5.4 Database	30
3. Αρχιτεκτονική εφαρμογής	27
3.1 Αρχιτεκτονική προγράμματος	Error! Bookmark not defined.
3.1.1 Back-end Authentication	31
3.1.2 Front-end	33
3.1.3 Database Schema	35
3.2 Επεξήγηση κώδικα	37
3.2.1 Feature extraction	38
3.2.2 Initialize	41
3.2.3 HTTP Requests	43
3.3 Δημοσίευση Εφαρμογής (Deployment)	45
4. Συμπεράσματα	49
4.1 Μελλοντικές Επεκτάσεις	51
5. Βιβλιογραφία - Δικτυογραφία	51

1. Εισαγωγή

Οι φωτογραφίες που υπάρχουν αναρτημένες στο διαδίκτυο περιέχουν, πολλές φορές, πληθώρα πληροφορίας και, στην πλειοψηφία τους, είναι ένα υποσύνολο κάποιων ήδη συντελεσμένων μετρήσεων. Παραδείγματα τέτοιων μετρήσεων μπορεί να είναι: αστρονομικές παρατηρήσεις, χάρτες

από δορυφόρους, μικροσυστοιχίες DNA, κυριάρχηση ρομποτικής όρασης και πολλών διαστάσεων φωτογραφίες όπως 3-D μαγνητική τομογραφία. Για την εξαγωγή τέτοιων πολύπλοκων δεδομένων χρειάζονται εργαλεία, τα οποία να είναι εύκολα στη χρήση τους, χωρίς περιορισμούς και ικανά να αποδώσουν το επιθυμητό αποτέλεσμα ([Stefan van der Walt, Johannes L. Schonberger, Juan Nunez-Iglesias, Francois Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors, 2014](#)).



Εικόνα 1.1: Από ένα input φωτογραφίας, το μοντέλο χωρίζει την φωτογραφία σε πολλά στρώματα για την καλύτερη αναγνώριση των αντικειμένων

Στόχος αυτού του προγράμματος της εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας είναι η αναγνώριση κάποιων πιο απλών και καθημερινών αντικειμένων και προσώπων από φωτογραφικό υλικό. Χρήση τέτοιου εργαλείου κάνουν και άλλες, πιο γνωστές, εφαρμογές για διαφορετικό σκοπό. Η κατασκευή του βασίζεται σε ένα μοντέλο μηχανικής μάθησης ή αλλιώς Machine Learning.

Ο όρος machine learning ή αλλιώς ML, εφευρέθηκε από τον Arthur Samuel το 1959 θέλοντας να λύσει ένα παιχνίδι σκάκι μέσω μιας μηχανής. Ο όρος αναφέρεται σε ένα υπολογιστικό πρόγραμμα το οποίο μπορεί να μάθει να παράγει συμπεριφορές οι οποίες δεν είναι απαραίτητα προγραμματισμένες από τον προγραμματιστή του. Οι συμπεριφορές που μαθαίνει είναι βάση τριών παραγόντων: πόσα δεδομένα έχει λάβει το πρόγραμμα, μία μέτρηση που επαληθεύει τα σφάλματα ή μια εκτίμηση του πόσο διαφέρει από την

ιδανική συμπεριφορά και ένας μηχανισμός ανατροφοδότησης που χρησιμοποιεί ποσοτικά σφάλματα για να καθοδηγήσει το πρόγραμμα έτσι ώστε να παράξει μία καλύτερη συμπεριφορά ([Ameet V Joshi ,2020](#)).

Αυτό που, ουσιαστικά, κάνει η εφαρμογή είναι ότι ζητά από τον χρήστη να εισάγει μία φωτογραφία μέσα στο μοντέλο και κατόπιν, με διαδικασίες και υπολογισμούς που θα αναλυθούν παρακάτω, το πρόγραμμα αναγνωρίζει τα πρόσωπα και αντικείμενα της εν λόγω φωτογραφίας και τα προβάλλει στην εφαρμογή.

Το κείμενο που ακολουθεί είναι χωρισμένο σε ακόμα δύο κεφάλαια: το ένα εξηγεί την θεωρία της τεχνητής νοημοσύνης και γενικότερα του νευρωνικού δικτύου, καθώς και κάποια μοντέλα. Το άλλο κεφάλαιο αφορά την αρχιτεκτονική της εφαρμογής, δηλαδή τα modules και τις βιβλιοθήκες που χρησιμοποιήθηκαν για να στηθεί η εφαρμογή.

2. Θεωρία

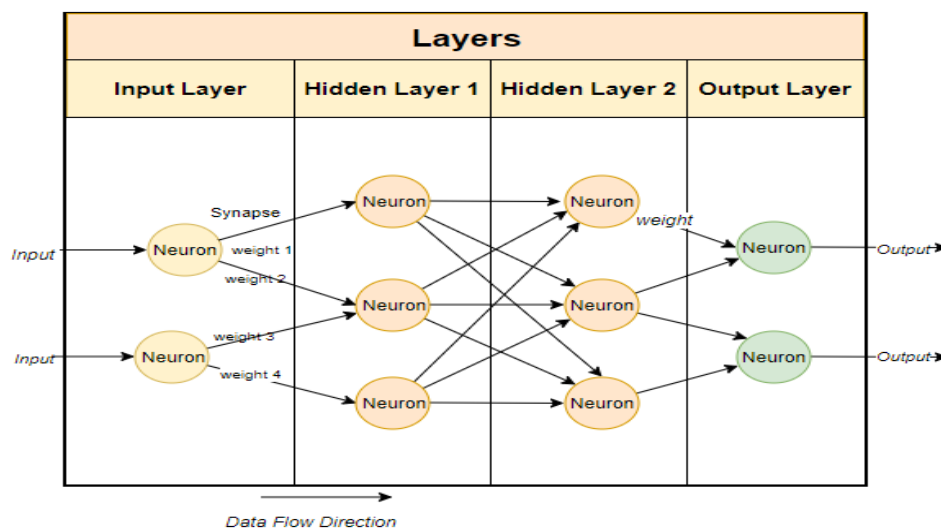
2.1 Εισαγωγή στα νευρωνικά δίκτυα

Ο ανθρώπινος εγκέφαλος είναι ένα πολύ καλό παράδειγμα νευρωνικού δικτύου, το οποίο μπορεί να φέρει εις πέρας πλήθος γνωστικών και αντιληπτικών ικανοτήτων. Πάνω από 15 χρόνια τώρα, ο άνθρωπος διαπίστωσε ότι πολλά υπολογιστικά μοντέλα που χρησιμοποιούνται σήμερα, εμπνεύστηκαν τη δομή τους και τις λειτουργίες τους από τις αντίστοιχες του προαναφερθέντος βιολογικού νευρωνικού δικτύου. Βάσει λοιπόν των παραπάνω διαπιστώσεων, προέκυψε η ιδέα ότι τα βιολογικά νευρωνικά δίκτυα μπορούν, με τη συνδρομή των ανάλογων τεχνολογικών μέσων, να δώσουν λύσεις σε θεμελιώδη ζητήματα. Όλο αυτό το πεδίο ονομάζεται **Artificial Neural Network** ή αλλιώς **Τεχνητό Νευρωνικό Δίκτυο** ([Artificial Neural Network](#)).

Η τεχνητή νοημοσύνη είχε πρώτα εμφανιστεί με τη δημιουργία του πρώτου υπολογιστή, τον οποίο τον περιέγραψε ο Alan Turing, λέγοντας: “Αν υπάρχει ένας/μία υπολογιστής/μηχανή πίσω από μία κουρτίνα και ένας άνθρωπος αλληλεπιδρά με αυτό και ο άνθρωπος νιώσει ότι αλληλεπιδρά με κάποιον άλλον άνθρωπο, τότε ο υπολογιστής είναι τεχνητή νοημοσύνη”. Τεχνητή νοημοσύνη, σύμφωνα με τον Turing, δεν είναι η κατασκευή μιας μηχανής ή ενός προγράμματος που λύνει γρηγορότερα κάποια προβλήματα αλλά ένα πρόγραμμα ή μια μηχανή που να μπορεί να μιμείται τις συμπεριφορές ενός ανθρώπου. Βέβαια η σύγχρονη οπτική του AI, είναι να ανατίθενται περισσότερα tasks σε μηχανές και προγράμματα όπως για παράδειγμα, κατανόηση της ανθρώπινης γλώσσας, λύση δύσκολων προβλημάτων που έχουν να κάνουν με έναν μεγάλο όγκο δεδομένων σε μικρό χρονικό διάστημα κλπ. Υπάρχουν δύο είδη τεχνητής νοημοσύνης που εξετάζονται από την άποψη της ανθρώπινης συμπεριφοράς. Στην πρώτη, όταν οι μηχανές είναι ικανές να επικοινωνήσουν με τον άνθρωπο αλλά δεν έχουν κίνηση, και η δεύτερη είναι αυτή που βρίσκεται σε φυσική αλληλεπίδραση με την ικανότητα της ανθρώπινης κίνησης, πράγμα το οποίο παραπέμπει στη ρομποτική ([Ameet V Joshi ,2020](#)).

Οι βασικές δομές ενός δικτύου ονομάζονται **τεχνητοί νευρώνες** ή απλώς **νευρώνες** ή **κόμβοι**. Τη βασική δομή, από την άλλη, ενός δικτύου ή, αλλιώς, μοντέλου σε ένα απλό μαθηματικό επίπεδο συνιστούν οι συνδέσεις(**weights**), οι οποίες αφενός διαμορφώνουν την επίδραση των σημάτων εισόδου και αφετέρου το μη γραμμικό χαρακτηριστικό που εκτίθεται από τους νευρώνες με συναρτήσεις μεταφοράς.

Στη συνέχεια, υπολογίζεται το άθροισμα των συνδέσμων από το σήμα εισόδου, το οποίο με τη σειρά του μετασχηματίζεται από τις συναρτήσεις μεταφοράς. Η ικανότητά του μοντέλου να μαθαίνει επιτυγχάνεται, προσαρμόζοντας τα weights σύμφωνα με τον αλγόριθμο.



Εικόνα 2.1: Απεικόνιση ενός μοντέλου

Η βασική δομή ενός νευρωνικού δικτύου απαρτίζεται από 3 τύπους κόμβων: είσοδος(input), κρυφός(hidden), έξοδος(output). Στα feed-forward δίκτυα, η ροή του σήματος είναι από το input στο output, αυστηρά σε feed-forward κατεύθυνση. Η επεξεργασία των δεδομένων μπορεί να χρειαστεί πολλαπλά στρώματα αλλά όχι συνδέσεις ανατροφοδότησης. Σε πολλές περιπτώσεις, οι τιμές ενεργοποίησης (συναρτήσεις μεταφοράς) των μονάδων υφίστανται μια διαδικασία χαλάρωσης, έτσι ώστε το δίκτυο να εξελιχθεί σε μια σταθερή κατάσταση στην οποία οι ενεργοποιήσεις δεν θα αλλάζουν. Σε άλλες εφαρμογές, οι αλλαγές των τιμών ενεργοποίησης για τους output νευρώνες είναι σημαντικές, καθώς η δυναμική συμπεριφορά αποτελεί το αποτέλεσμα του δικτύου. Υπάρχουν πολλές αρχιτεκτονικές

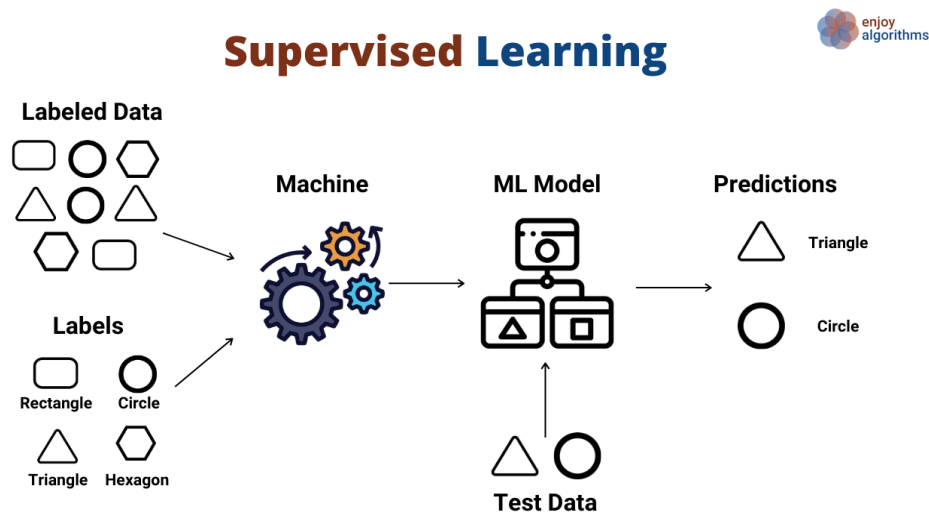
νευρωνικών δικτύων (δίκτυο Elman, χάρτες προσαρμοστικού συντονισμού, ανταγωνιστικά δίκτυα κλπ), με βάση αυτά που ζητάει κάθε εφαρμογή ([Artificial Neural Network](#)).

Ένα νευρωνικό δίκτυο πρέπει να διαμορφωθεί έτσι ώστε το σέτ των inputs να παράξει το επιθυμητό σέτ των outputs. Υπάρχουν πολλές μέθοδοι για να οριστεί η δύναμη της σύνδεσης. Ένα τρόπος είναι να οριστούν τα weights διαφορετικά, χρησιμοποιώντας εκ των προτέρων γνώση, και ένας άλλος τρόπος είναι να εκπαιδευτεί το νευρωνικό δίκτυο μαθαίνοντας τα μοτίβα και αφήνοντάς το να διαμορφώσει τα weights σύμφωνα με αυτά που έχει μάθει. Η κατάσταση της μάθησης στα νευρωνικά δίκτυα μπορούν να ταξινομηθούν σε τρία διαφορετικά είδη γνωστά και ως **τύποι μάθησης** ([Artificial Neural Network](#)).

2.1.1 Τύπος Μάθησης

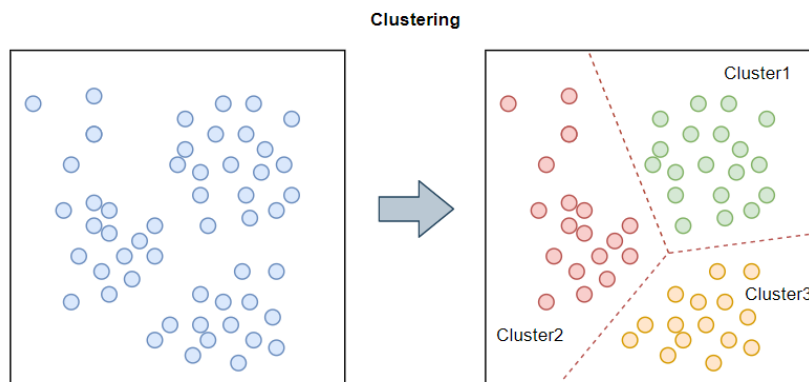
Το Machine Learning είναι ένα περιβάλλον που κληρονομεί ιδέες από διάφορα άλλα πεδία της επιστήμης, όπως η τεχνητή νοημοσύνη.

Το βασικό χαρακτηριστικό του machine learning είναι τα διάφορα πεδία μάθησης που μπορεί να διαθέτει. Θα πρέπει να σημειωθεί εδώ ότι υπάρχουν διαφορετικοί τύποι μάθησης. Ένας από αυτούς ονομάζεται **Supervised Learning** και το κύριο χαρακτηριστικό του είναι πως περιγράφει μια κλάση ενός προβλήματος, ενώ χρησιμοποιεί το μοντέλο, για να μάθει ένα μοτίβο μεταξύ των παραδειγμάτων που θα μπουν στο input layer και του αποτελέσματος. Το μοντέλο είναι προσαρμοσμένο σε δεδομένα (training data) που αποτελούνται από εισόδους και εξόδους και χρησιμοποιούνται για να κάνουν προβλέψεις σε test data. Στην περίπτωση αυτή, παρέχονται μόνο τα inputs ενώ τα outputs του μοντέλου αναλαμβάνουν να συγκριθούν με τον στόχο, προκειμένου να αναδειχθούν τελικά οι ικανότητες του τελευταίου ([Ameet V Joshi ,2020](#)).



Εικόνα 2.2: Παράδειγμα ενός μοντέλου που χρησιμοποιεί supervised Learning

Ένας ακόμα, γνωστός τύπος μάθησης είναι το **Unsupervised Learning**. Σε αυτό τον τύπο τα labelled data δεν είναι διαθέσιμα, σε αντίθεση με το supervised. Ένα κλασικό παράδειγμα τέτοιας μάθησης είναι το clustering. Παραδείγματος χάρη, έχουμε μετρήσεις από 4 διαφορετικά χαρακτηριστικά αυτοκινήτων μέγεθος, βάρος, ύψος, τιμή και έχουμε και 3 διαφορετικούς τύπους αυτοκινήτων αγωνιστικό, επιβατικό και φορτηγό. Για κάθε έναν από αυτούς τους τύπους έχουμε 30 διαφορετικά αυτοκίνητα. Στην περίπτωση του supervised learning τα δεδομένα αυτά θα χρησιμοποιούνταν για training data όπου θα είχαμε 4 inputs (μέγεθος, βάρος, ύψος, τιμή) και ένα output (τύπος αυτοκινήτου). Στην περίπτωση όμως του unsupervised learning δεν έχουμε τα ακριβή ονόματα των αυτοκινήτων για κάθε μία μέτρηση, υπάρχει απλά ένα σετ μετρήσεων. Οπότε στο παράδειγμα που αναφέρθηκε, αυτό που μπορεί να γίνει είναι αντί να έχουμε τους τύπους των αμαξιών (αγωνιστικό, επιβατικό και φορτηγό) να έχουμε τρεις clusters, αμάξι-1, αμάξι-2 και αμάξι-3, έτσι ώστε όταν θα δοθεί μία καινούρια μέτρηση να μπορέσουμε να εκτιμήσουμε πόσο κοντά βρίσκεται σε κάποιο cluster



Εικόνα 2.3: Παράδειγμα μοντέλου που χρησιμοποιεί unsupervised learning

Τέλος, υπάρχει και το *Reinforcement Learning* το οποίο είναι μια ειδική κατηγορία στις μεθόδους μάθησης και αντιμετωπίζεται πολύ διαφορετικά από τα supervised και unsupervised. Η μέθοδος αυτή ανατροφοδοτείται από το περιβάλλον, αλλά δεν θεωρείται ως unsupervised. Επίσης, δεν έχει labelled δείγματα τα οποία είναι διαθέσιμα για training και ως εκ τούτου δεν μπορεί να θεωρηθεί ως supervised. Σε αυτού του είδους τη μέθοδο, το σύστημα αλληλεπιδρά συνεχόμενα με το περιβάλλον προσπαθώντας να παράξει την καλύτερη συμπεριφορά και να λάβει ανατροφοδότηση από το περιβάλλον ([Ameet V Joshi, 2020](#)).



Εικόνα 2.4: Παράδειγμα μοντέλου που χρησιμοποιεί reinforced learning

2.2 Μοντέλο και κλάση

Αυτή η ενότητα θα ασχοληθεί με το μοντέλο αλλά και την κλάση που χρησιμοποιήθηκε για την διεξαγωγή της εφαρμογής. Η κλάση αυτή

ονομάζεται **CNN(Convolutional Neural Network)** και είναι υπεύθυνη για την αναγνώριση των αντικειμένων και των προσώπων, στη συγκεκριμένη περίπτωση, από την φωτογραφία.

2.2.1 Convolutional Neural Network

Τα CNN, όπως αναφερθήκαμε και παραπάνω, είναι μία κλάση αναγνώρισης που εφαρμόζεται για να αναλύσει εικόνες. Τα CNN είναι μία από τις κλάσεις των deep learning neural networks. Αντιπροσωπεύουν μία τεράστια βελτίωση στην αναγνώριση φωτογραφιών. Συνήθως χρησιμοποιούνται στην ανάλυση φωτογραφιών και βρίσκονται “πίσω” από την αναγνώριση φωτογραφιών. Βασίζονται πάνω στο Image Classification το οποίο δέχεται κάποια inputs (όπως μια φωτογραφία) και κάποια outputs (όπως είναι μία κλάση “σκύλος”) ή μία πιθανότητα ότι τα inputs είναι μια κλάση. Η αρχιτεκτονική στην αναγνώριση φωτογραφιών είναι:

Input->Convolutional->ReLU->Convolutional->ReLU->Pooling->ReLU->Convolutional->ReLU->Pooling->Fully Connected

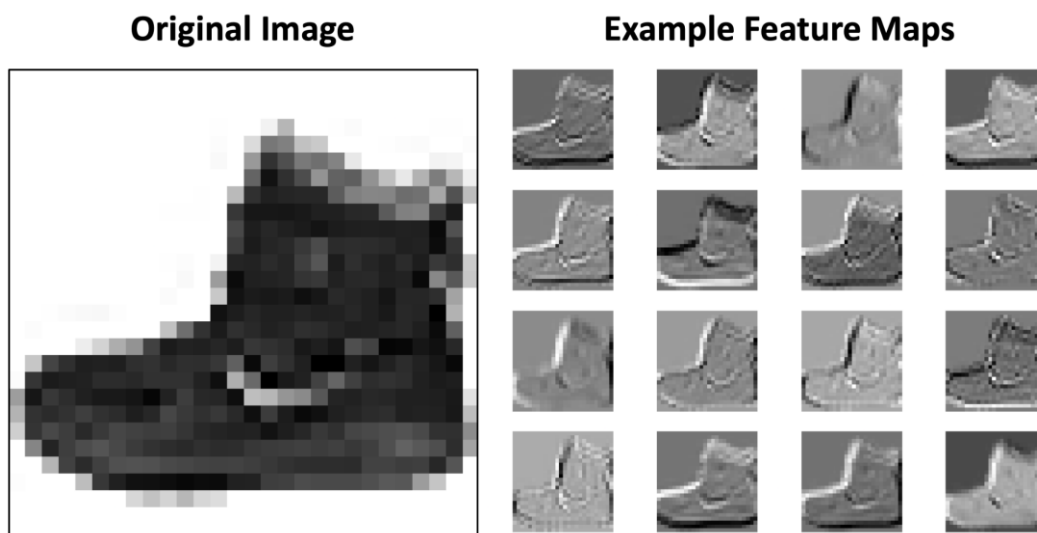
(Οι όροι εξηγούνται στη συνέχεια)

Αυτό που συμβαίνει είναι ότι το CNN μαζεύει όσο το δυνατόν περισσότερα δεδομένα γίνεται για το input και έπειτα χρησιμοποιεί ένα 2D Convolutional Layer. Αυτό σημαίνει ότι αυτού του είδους τα δίκτυα είναι ιδανικά για την επεξεργασία 2D φωτογραφιών. Σε σύγκριση με άλλους αλγόριθμους αναγνώρισης φωτογραφιών, το CNN χρησιμοποιεί ελάχιστη επεξεργασία και μπορεί να μάθει τα φίλτρα της φωτογραφίας, σε αντίθεση με τους άλλους αλγορίθμους που πρέπει να γίνουν χειροποίητα.

Η διαδικασία που ακολουθεί η αρχιτεκτονική για να αναγνωρίσει τα αντικείμενα της φωτογραφίας είναι η εξής:

1. δέχεται σαν input μια φωτογραφία
2. εφαρμόζει πολλαπλά φίλτρα έτσι ώστε να δημιουργήσει τον χάρτη των χαρακτηριστικών (feature map) της φωτογραφίας

3. εφαρμόζει την συνάρτηση ReLU για να αυξήσει την μη-γραμμικότητα
4. εφαρμόζει το pooling layer σε κάθε χάρτη χαρακτηριστικών
5. κάνει flatten τα pooled images σε ένα μονοδιάστατο πίνακα
6. εισάγει αυτόν τον πίνακα μέσα στο μοντέλο
7. επεξεργάζεται τα χαρακτηριστικά μέσα από το δίκτυο. Το τελευταίο layer περιέχει το “ψήφισμα” για τις κλάσεις.
8. κάνει train μπρος και πίσω το μοντέλο για πολλές επαναλήψεις, μέχρις ότου βρει το επιθυμητό αποτέλεσμα ([The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification, 2019](#)).



Εικόνα 2.5: Παράδειγμα φωτογραφίας και το feature map της

- **Receptive Fields**

Στο δίκτυο, κάθε νευρώνας δέχεται ένα input από κάποιον αριθμό τοποθεσίας του προηγούμενου επιπέδου (layer). Σε ένα convolutional layer, κάθε νευρώνας δέχεται ένα input από μία και μοναδική περιορισμένη περιοχή του προηγούμενου layer που ονομάζεται **receptive field (δεκτικό πεδίο)**. Σε ένα πλήρως συνδεδεμένο δίκτυο, το δεκτικό πεδίο είναι ολόκληρο το προηγούμενο layer. Για αυτό, και κάθε convolutional layer, κάθε νευρώνα παίρνει ένα input από μία μεγαλύτερη περιοχή απ' ότι του προηγούμενου layer. Αυτό οφείλεται στην εφαρμογή του convolutional layer, το οποίο παίρνει υπόψη την τιμή ενός pixel, καθώς και τα γύρω του pixels. Όταν χρησιμοποιεί διεσταλμένα layers, ο αριθμός των pixels σε

ένα δεκτικό πεδίο παραμένει σταθερός, αλλά το πεδίο είναι πιο αραιοκατοικημένο καθώς οι διαστάσεις μεγαλώνουν όταν συνδυάζεται η επίδραση πολλών layers ([Convolutional Neural Network](#)).

- **Weights**

Κάθε νευρώνας σε ένα δίκτυο υπολογίζει μια τιμή εξόδου εφαρμόζοντας μια συγκεκριμένη συνάρτηση στις τιμές εισόδου που λαμβάνει από το δεκτικό πεδίο προηγούμενου επιπέδου. Η συνάρτηση που εφαρμόζεται στις τιμές εισόδου καθορίζεται από ένα διάνυσμα από **weights** και **bias**. Για να καταφέρει να μάθει το μοντέλο, χρειάζεται επαναληπτική προσαρμογή των weights και bias.

Τα διανύσματα των weights και bias ονομάζονται και αλλιώς φίλτρα και αντιπροσωπεύουν συγκεκριμένα χαρακτηριστικά της εισόδου. Ένα διακριτικό χαρακτηριστικό των CNN είναι ότι πολλοί νευρώνες μοιράζονται το ίδιο φίλτρο. Αυτό μειώνει το αποτύπωμα της μνήμης επειδή ένα μοναδικό bias και διάνυσμα από weights χρησιμοποιούνται σε όλο το δεκτικό πεδίο που μοιράζεται τα φίλτρα, σε αντίθεση με το δεκτικό πεδίο που έχει το δικό του bias και το δικό του διάνυσμα από weights ([Convolutional Neural Network](#)).

- **Bias**

Το διάνυσμα ενός bias είναι ένα επιπλέον σεντ από weights μέσα σε ένα δίκτυο που δεν χρειάζεται input, και αντιστοιχεί σε ένα output του δικτύου όταν έχει την τιμή 0.

Τα bias εκπροσωπούν ένα επιπλέον νευρώνα που περιέχει το καθένα ένα pre-output layer και αποθηκεύει την τιμή 1 για κάθε ενέργεια. Τα bias δεν είναι συνδεδεμένα σε κανένα προηγούμενο layer μέσα στο δίκτυο, οπότε δεν εκπροσωπούν καμία μορφή δραστηριότητας, αλλά συμπεριφέρονται όπως και τα weights.

Χωρίς κάποιο bias node, κανένα layer δεν θα μπορούσε να παράγει κάποιο output για το επόμενο layer που διαφέρει από το 0 αν η χαρακτηριστική του τιμή ήταν 0 ([Bias Vector](#)).

- **Pooling layers**

Στόχος των **Pooling Layers** είναι να μειώνουν τις διαστάσεις των δεδομένων συνδυάζοντας τις εξόδους των **neuron clusters(σύμπλεγμα νευρώνων)** σε έναν νευρώνα του επόμενου στρώματος. Το local pooling συνδυάζει μικρά συμπλέγματα, σε μέγεθος ενός πλακιδίου και διάστασης 2x2, συνήθως. Το global pooling δρα σε όλους τους νευρώνες του συστήματος. Υπάρχουν 2 συνηθισμένοι τύποι που χρησιμοποιούνται πιο συχνά: το max και το average. Το max pooling χρησιμοποιεί την μέγιστη τιμή για κάθε local cluster των νευρώνων του συστήματος, ενώ το average pooling παίρνει την μέση τιμή ([Convolutional Neural Network](#)).

- **Convolutional Layer**

Τα convolutional layers συγκεντρώνουν τα inputs και δίνουν το αποτέλεσμα στο επόμενο layer. Κάθε νευρώνας του convolutional layer επεξεργάζεται την πληροφορία μόνο για το δεκτικό του πεδίο. Παρόλα αυτά, το πλήρες συνδεδεμένο νευρωνικό δίκτυο μπορεί να χρησιμοποιηθεί για να μάθει χαρακτηριστικά και να μπορεί να αναγνωρίσει τα δεδομένα. Αυτή η αρχιτεκτονική δεν είναι πολύ πρακτική για μεγάλου όγκου δεδομένα όπως για παράδειγμα μια μεγαλύτερη ανάλυση σε μία φωτογραφία. Θα χρειαστεί ένας μεγάλος αριθμός από νευρώνες, καθ'όλη την διάρκεια του input αυτής της μεγάλης φωτογραφίας, καθώς κάθε pixel της φωτογραφίας περιλαμβάνει και από ένα χαρακτηριστικό([Convolutional Neural Network](#)).

- **ReLU Layer**

Το ReLU είναι μία συντομογραφία του **Rectified Linear Unit**, το οποίο εφαρμόζει την συνάρτηση ενεργοποίησης $f(x) = \max(0, x)$. Μπορεί και αφαιρεί τις αρνητικές τιμές από τον χάρτη ενεργοποίησης θέτοντάς τα όλα 0([Convolutional Neural Network](#)).

- **Fully Connected Layer**

Έπειτα από πολλά pooling layers αλλά και convolutional layers, το τελικό classification γίνεται μέσω των connected layers. Οι νευρώνες μέσα σε αυτό

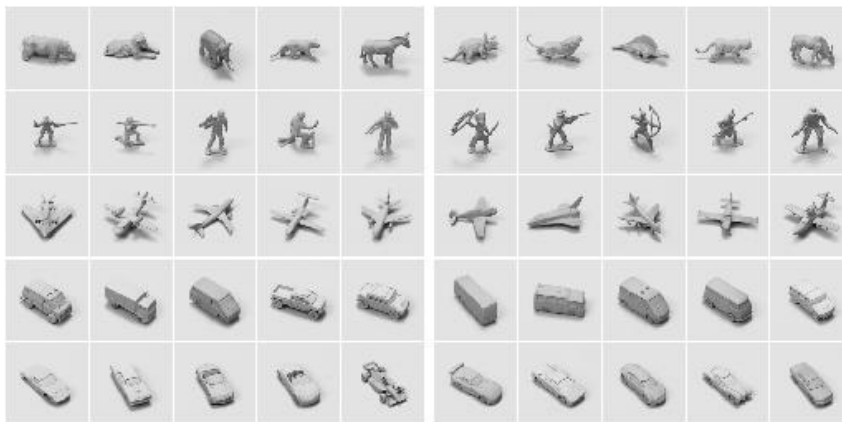
το layer έχουν συνδέσεις σε όλες τις ενεργοποιήσεις των προηγούμενων layers([Convolutional Neural Network](#)).

- **Classification Layer**

Τα μεγέθη του πυρήνα των φίλτρων και τα ορθογώνια max-pooling καθώς και οι παράγοντες παράλειψης, επιλέγονται έτσι ώστε είτε τα output maps ενός convolutional layer να κατέβουν ένα pixel ανά map, ή ένα πλήρως συνδεδεμένο επίπεδο να συνδέσει τις εξόδους του κορυφαίου convolutional layer σε μονοδιάστατο feature vector. Το ανώτερο στρώμα είναι πάντα πλήρως συνδεδεμένο, με μία μονάδα εξόδου ανά label ([Flexible, High Performance Convolutional Neural Networks for Image Classification](#)).

2.3 Image Processing Layer

Το **Image Processing Layer** είναι μία προαιρετική προεργασία στρώματος ενός προκαθορισμένου φίλτρου που έμεινε αναλλοίωτο καθ' όλη τη διάρκεια του training. Επομένως, στο δίκτυο μπορούν να παρέχονται πρόσθετες πληροφορίες εκτός από την ακατέργαστη εικόνα εισόδου, όπως ακμές και κλίσεις. Συγκεκριμένα, διαπιστώθηκε ότι ένα στρώμα εξαγωγής βοηθάει στην βελτίωση του ποσοστού αναγνώρισης του **NORB(NYU Object Recognition Benchmark)** ([Flexible, High Performance Convolutional Neural Network for Image Classification](#)).



Εικόνα 2.6: Η φωτογραφία δείχνει τα 25 αντικείμενα που είναι για training (στα αριστερά) και άλλα 25 διαφορετικά αντικείμενα για testing (στα δεξιά). Υπάρχουν 5 κατηγορίες: ζώα, ανθρώπινες φιγούρες, αεροπλάνα, αμάξια και φορητά

Το NORB είναι ένα dataset που περιέχει 50 φωτογραφίες χρωματιστών παιχνιδιών, 9 ανυψώσεις και 6 συνθήκες φωτισμού για ένα σύνολο από 194,400 φωτογραφίες. Η αναγνώριση του NORB εξάγει κατηγορίες από φωτισμό, και την παρουσία της αταξίας που είναι από τα μεγαλύτερα challenges του computer vision ([NORB: Generic Object Recognition in Images, 2003](#)).

2.4 Αλγόριθμοι/ Detectors

Με την πρόοδο του deep learning και ειδικότερα των CNN, σε εφαρμογές υπολογιστικής όρασης, η ακρίβεια της ταξινόμησης και της αναγνώρισης αντικειμένων έχει γνωρίσει εντυπωσιακή βελτίωση. Η εξέλιξη των GPU επίσης συνέβαλε σημαντικά στην υιοθέτηση του CNN για την ανάπτυξη της υπολογιστικής δύναμης, ξεπερνώντας τα προβλήματα της επεξεργασίας σε πραγματικό χρόνο μέσω παραλληλισμού. Επιπλέον οι τελευταίες τάσεις του cloud robotics έχουν επίσης ενεργοποιήσει την εκφόρτωση των βαριών υπολογισμών, όπως ανάλυση βίντεο στο cloud. Αυτό επιτρέπει την επεξεργασία των βίντεο σε πραγματικό χρόνο χρησιμοποιώντας προηγμένους αλγόριθμους στο πλαίσιο της επιτήρησης.

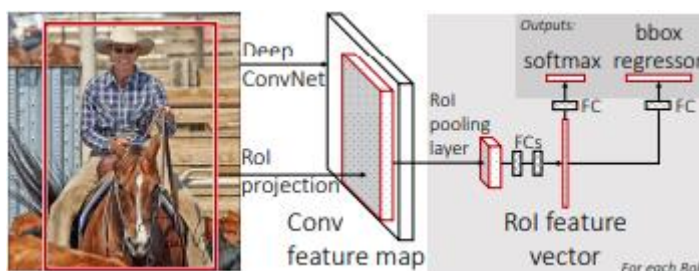
2.4.1 Fast R-CNN

Μία από τις πιο γρήγορες μεθόδους αναγνώρισης είναι το Fast R-CNN. Η κατασκευή του είναι βασισμένη σε προηγούμενους αλγόριθμους αναγνώρισης οι οποίοι είναι και αυτοί βασισμένοι πάνω στο CNN. Συγκριτικά με άλλα μοντέλα το Fast R-CNN χρησιμοποιεί αρκετές καινοτομίες για τη βελτίωση του training και του testing καθώς αυξάνει την ακρίβεια της αναγνώρισης. Είναι 9 φορές πιο γρήγορο από το R-CNN και 213 φορές στο test-time.

Το δίκτυο παίρνει σαν είσοδο μια φωτογραφία και μία λίστα από R αντικείμενα. Στο test-time, το R είναι συνήθως γύρω στο 2000, αν και εξετάζεται σε περιπτώσεις στις οποίες είναι μεγαλύτερο. Όταν

χρησιμοποιείται μία εικόνα πυραμίδα, κάθε Rol (Region of Interest) εκχωρείται στην κλίμακα έτσι ώστε το κλιμακωτό Rol να είναι πιο κοντά στα 224x224 εικονοστοιχεία της περιοχής.

Το Region of Interest(Rol) θεωρείται και αυτό ένα pooling layer, το οποίο χρησιμοποιεί το max pooling για να μετατρέψει τα χαρακτηριστικά μιας περιοχής σε μικρά feature maps με σταθερές το ύψος(H) και το πλάτος(W). Το Rol χρησιμοποιήθηκε για να δημιουργεί ένα τετράγωνο πλαίσιο και έπειτα σε ένα convolutional feature map. Κάθε Rol έχει τέσσερα tuples (x,y,w,h) , όπου το (x,y) δηλώνουν την πάνω αριστερή γωνία και το (H,W) το ύψος και το πλάτος.([Fast R-CNN](#)).



Εικόνα 2.7: Fast R-CNN αρχιτεκτονική. μία εικόνα εισόδου και πολλαπλές περιοχές ενδιαφέροντος(Rol) εισάγονται σε ένα πλήρες CNN. Κάθε Rol συγκεντρώνεται σε έναν χάρτη χαρακτηριστικών σταθερού μεγέθους και στη συνέχεια αντιστοιχίζει σε ένα διάνυσμα χαρακτηριστικών με πλήρως συνδεδεμένα επίπεδα(FCs). Το δίκτυο έχει δύο διανύσματα εξόδου ανά Rol: πιθανότητες softmax και για κάθε κλάση του bounding-box . Η αρχιτεκτονική είναι εκπαιδευμένη end-to-end μ.

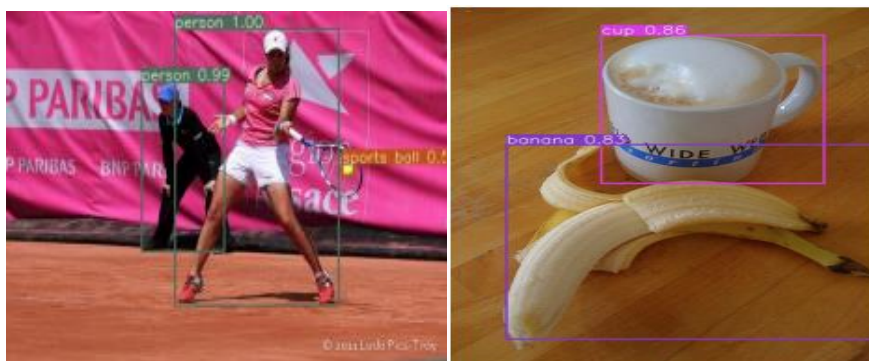
2.4.2 YOLOv3

Η μέθοδος αυτή είναι μία από τις πιο διαδεδομένες και ευρέως χρησιμοποιούμενες για την ανίχνευση αντικειμένων από φωτογραφία. Χρησιμοποιεί τη μέθοδο του συμπλέγματος k-means για να εκτιμήσει το αρχικό πλάτος και ύψος των προβλεπόμενων bounding-boxes. Με αυτή τη μέθοδο, το εκτιμώμενο πλάτος και ύψος είναι ευαίσθητα στα αρχικά κέντρα συμπλεγμάτων (clusters centers), καθώς και η επεξεργασία συνόλων δεδομένων μεγάλης κλίμακας είναι χρονοβόρα. Προκειμένου να αντιμετωπιστούν αυτά τα προβλήματα, μία νέα μέθοδος συμπλεγμάτων για την εκτίμηση του αρχικού ύψους και πλάτους των προβλεπόμενων bounding-boxes έχει αναπτυχθεί.

Πρώτα, επιλέγεται τυχαία ένα πλάτος και τιμές ύψους ως ένα αρχικό cluster center χωριστά από το πλάτος και το ύψος των πραγματικών bounding-boxes (σχεδιάζεται ένα τυχαίο κουτί μέσα στη φωτογραφία το οποίο μπορεί να μην περιέχει κάποιο αντικείμενό της).

Δεύτερον, κατασκευάζονται αλυσίδες Markov με βάση την επιλεγμένη αρχική συστάδα και χρησιμοποιούνται τα τελευταία σημεία κάθε αλυσίδας σαν τα άλλα αρχικά κέντρα συστάδων. Στην κατασκευή των αλυσίδων Markov, η μέθοδος intersection-over-union χρησιμοποιείται για τον υπολογισμό της απόστασης μεταξύ των επιλεγμένων αρχικών συμπλεγμάτων (initial clusters) και κάθε υποψήφιου σημείου αντί της μεθόδου τετραγωνικής ρίζας.

Τέλος, αυτή η μέθοδος μπορεί να χρησιμοποιηθεί για τη συνεχή ενημέρωση του cluster center με κάθε σύνολο τιμών πλάτους και ύψους που είναι μόνο ένα μέρος των δεδομένων που επιλέγονται από τα σύνολα δεδομένων ([Object Detection Algorithm Based on Improved YOLOv3, 2020](#)).

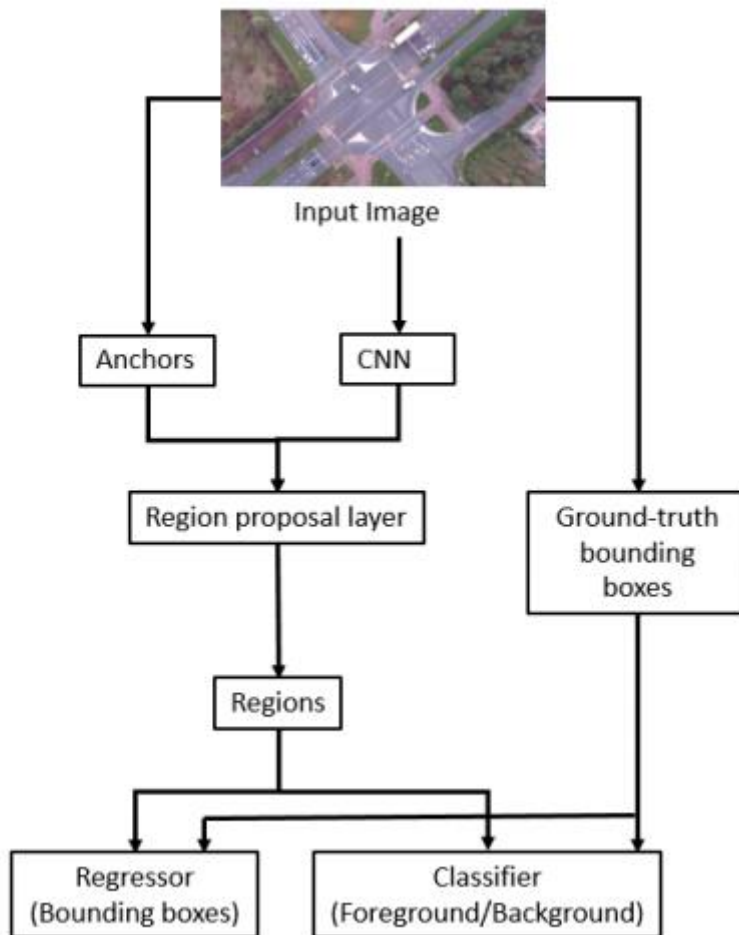


Εικόνα 2.8: Παραδείγματα του YOLOv3 από την αναγνώριση αντικειμένων από μια φωτογραφία

2.4.3 YOLOv3 Vs Fast R-CNN

Σε αντίθεση με τις παραλλαγές του R-CNN, το YOLO δεν εξάγει προτάσεις περιοχής (region proposals), αλλά επεξεργάζεται την πλήρη εικόνα εισόδου μόνο μια φορά χρησιμοποιώντας ένα ολοκληρωμένο CNN που προβλέπει τα bounding-boxes και τις αντίστοιχες πιθανές κλάσεις τους, με βάση το καθολικό πλαίσιο της εικόνας. Η πρώτη έκδοση του δημοσιεύτηκε το 2016, και έπειτα το 2017 προτάθηκε και μία δεύτερη έκδοση YOLOv2, η οποία

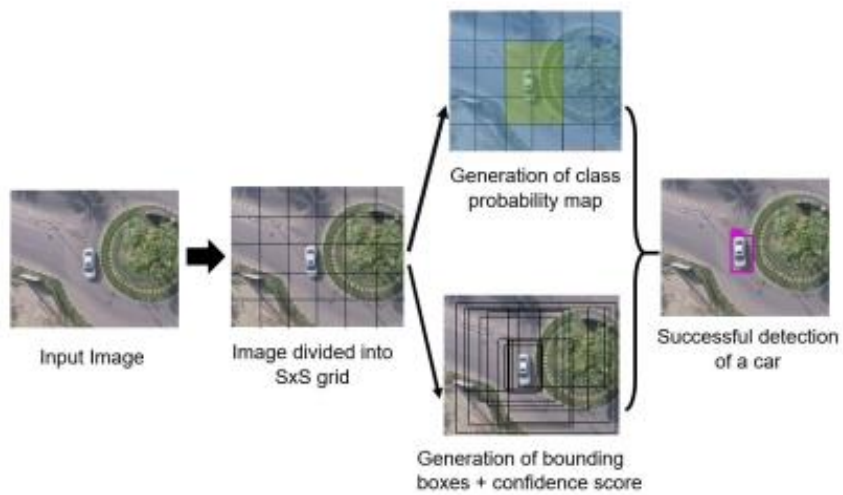
εισήγαγε την ομαλοποίηση παρτίδας, μια φάση επανασύνδεσης για το δίκτυο ταξινόμησης και τις συστάδες διαστάσεων ως anchor boxes για την πρόβλεψη των bounding boxes. Τέλος, το 2018, το YOLOv3 βελτίωσε περαιτέρω τον εντοπισμό, υιοθετώντας πολλές νέες δυνατότητες:



Εικόνα 2.9: Αρχιτεκτονική Region Proposal Network (RPN)

Πάλι, σε αντίθεση με το Fast R-CNN, σε κάθε αντικείμενο στο YOLOv3 αντιστοιχεί μόνο ένα όριο bounding box. Αυτές οι διαδοχικές παραλλαγές του YOLO αναπτύχθηκαν με στόχο την απόκτηση ενός μέγιστου mAP (mean average precision) διατηρώντας την ταχύτερη εκτέλεση που το καθιστά κατάλληλο για εφαρμογές σε πραγματικό χρόνο.

Περισσότερη έμφαση δόθηκε στον χρόνο εκτέλεσης, έτσι ώστε το YOLOv3 να ισοδυναμεί με τον state-of-the-art αλγόριθμο ανίχνευσης όπως ο SSD (Single Shot Detector) όσον αφορά την ακρίβεια αλλά με το πλεονέκτημα ότι είναι τρεις φορές πιο γρήγορο.



Εικόνα 2.10: Αρχιτεκτονική YOLOV3



Εικόνα 2.11: Παράδειγμα εξόδου από τον αλγόριθμο του YOLOv3

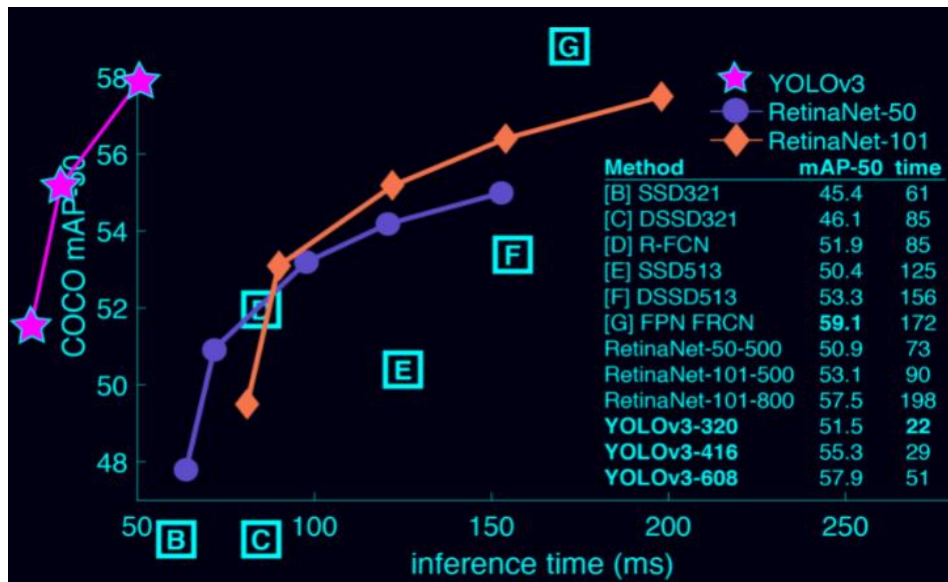
TABLE II
THEORETICAL COMPARISON OF FASTER R-CNN AND YOLOV3

	YOLOv3	Faster R-CNN
Phases	Concurrent bounding-box regression, and classification.	RPN + Fast R-CNN object detector.
Neural network type	Fully convolutional.	Fully convolutional (RPN and 4 detection network).
Backbone feature extractor	Darknet-53 (53 convolutional layers).	VGG-16 or Zeiler & Fergus (ZF). Other feature extractors can also be incorporated.
Location detection	Anchor-based (dimension clusters).	Anchor-based.
Number of anchors boxes	Only one bounding-box prior for each ground-truth object.	3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position.
Default Anchor sizes	(10,13),(16,30),(33,23), (30,61),(62,45), (59,119),(116,90), (156,198), (373,326).	Scales: (128,128),(256,256), (512,512). Aspect ratios: 1:1, 1:2, 2:1.
IoU thresholds	One (at 0.5).	Two (at 0.3 and 0.7).
Loss function	Binary cross-entropy loss.	Multi-task loss: - Log loss for classification. - Smooth L1 for regression.
Input size	Three possible input sizes (320x320, 416x416, and 608x608).	- Conserves the aspect ratio of the original image. - Either the smallest dimension is 600, or the largest dimension is 1024.
Momentum	Default value: 0.9.	Default value: 0.9.
Weight decay	Default value: 0.0005.	Default value: 0.0005.
Batch size	Default value: 64.	Default value: 1.

Για το συγκεκριμένο παράδειγμα χρησιμοποιήθηκε το PSU dataset το οποίο περιέχει λήψεις που έχουν τραβηχτεί από ψηλά καταγράφοντας πολλά αντικείμενα (αμάξια, φορητά, ποδήλατα, ανθρώπους κλπ.). Στην εικόνα [2.10] απεικονίζονται τα κύρια στάδια του αλγορίθμου όταν εφαρμόζει την ανίχνευση των αντικειμένων. Για την διεξαγωγή του πειράματος δοκιμάστηκαν και τα τρία μεγέθη εισόδου που χρησιμοποιούνται συνήθως: 320x320, 416x416, 608x608.

Στον πίνακα II είναι εμφανής η σύγκριση ανάμεσα στις παραμέτρους του Faster R-CNN και του YOLOv3. Ενώ διαδοχικές βελτιστοποιήσεις και

αμοιβαίες εμπνεύσεις έφεραν τη μεθοδολογία των δύο αλγορίθμων σχετικά κοντά, η κύρια διαφορά τους παραμένει ότι το Faster R-CNN έχει δύο ξεχωριστές φάσεις του region proposal και του classification, ενώ το YOLO συνδυάζει πάντα το classification και τη διαδικασία του bounding-box regression ([Aerial Images Processing for Car Detection using Convolutional Neural Networks: Comparison between Faster R-CNN and YoloV3, 2020](#)).



Εικόνα 2.12: Σύγκριση του Yolon3 με άλλου detectors.

3. Αρχιτεκτονική εφαρμογής

Στο κεφάλαιο αυτό θα δούμε εργαλεία για την κατασκευή της εφαρμογής και θα εξηγήσουμε τι κάνουν και πώς τα χρησιμοποιούμε καθώς και ποια δομή ακολουθήθηκε για την αρχιτεκτονική της εφαρμογής (front-end, back-end, database schema κτλ.)

3.1 Βιβλιοθήκες και εργαλεία

Σε αυτή την ενότητα θα εξετασθούν οι βιβλιοθήκες και τα frameworks που είναι κατάλληλα για το neural network development και το machine learning, αλλά και το web development, τα οποία χρησιμοποιήθηκαν για την ανάπτυξη της παρούσας εφαρμογής.

Η πιο κοινή γλώσσα για την δημιουργία ενός neural network προς το παρόν είναι η Python. Υπάρχουν διάφοροι λόγοι για τους οποίους αυτή η γλώσσα έχει κερδίσει αυτόν τον τομέα.

- Η Python είναι αρκετά εύκολη γλώσσα για να την μάθει κάποιος και για αυτόν τον λόγο χρησιμοποιείται σαν starter γλώσσα προγραμματισμού στον εκπαιδευτικό τομέα.
- Ο τυπικός διερμηνέας cpython διευκολύνει τη δημιουργία δεσμών για κλήσεις C συναρτήσεων, επιτρέποντας στην Python να χρησιμοποιηθεί ως βολική διεπαφή για βιβλιοθήκες χαμηλού επιπέδου.
- Το community της έχει δημιουργήσει ένα ευρύ φάσμα εργαλείων για εκτέλεση κώδικα της Python αλλά και για το data visualization. Ειδικότερα είναι πολύ χρήσιμη για επιστημονικές μελέτες, όπου σχεδόν πάντα δεν υπάρχει αλγόριθμος για τις λύσεις και είναι απαραίτητη η διεξαγωγή επιστημονικής αναζήτησης ([Computer Science and Computer Engineering, 2019](#)).

Οι βιβλιοθήκες που χρησιμοποιήθηκαν για την εκτέλεση της εφαρμογής είναι open source.

3.1.1 Machine Learning Libraries

Το NumPy είναι η βιβλιοθήκη που υλοποιεί πίνακες υψηλής απόδοσης και εργαλεία για αυτούς. Ο υπολογιστικός του πυρήνας είναι κατά 52% γραμμένος σε C με το κομμάτι της αλληλεπίδρασης με την Python να είναι στο 48%. Οι συναρτήσεις της γραμμικής άλγεβρας βασίζονται σε έναν μεγάλο βαθμό στη βιβλιοθήκη LAPACK. Το NumPy υλοποιεί μεγάλη ποικιλία από μεθόδους γραμμικής άλγεβρας για προβλήματα με διανύσματα, πίνακες και tensors (πολυδιάστατοι πίνακες). Επίσης, υποστηρίζει παράλληλο προγραμματισμό χρησιμοποιώντας διανυσματικές δυνατότητες των σύγχρονων CPU ([Computer Science and Computer Engineering, 2019](#)).

Το OpenCV είναι μία open source βιβλιοθήκη και περιέχει περισσότερους από 500 βελτιστοποιημένους αλγόριθμους: γενικές συναρτήσεις για την επεξεργασία εικόνων, image pyramids, γεωμετρικές περιγραφές, τμηματοποίηση, μετασχηματισμούς, εξαγωγή χαρακτηριστικών καθώς και πολλές ακόμα συναρτήσεις. Έχει βρει εφαρμογή σε διαφορετικά πεδία, από τη διαδραστική τέχνη μέχρι και την ρομποτική ([scikit-image: image processing in Python, 2014](#)).

Το Matplotlib, είναι και αυτό, μια βιβλιοθήκη της python πιο πολύ για την σχεδίαση 2D που παράγει γραφικές παραστάσεις. Υποστηρίζει διαδραστική και μη πλοκή, και μπορεί να αποθηκεύσει εικόνες σε διάφορες μορφές εξόδου. Επίσης μπορεί να χρησιμοποιήσει πολλαπλές εργαλειοθήκες παραθύρων και παρέχει μια μεγάλη ποικιλία τύπων σχεδίου (γραμμές, ράβδους, διαγράμματα πίτας, ιστογράμματα και πολλά ακόμα). Εκτός από αυτό, είναι ιδιαίτερα προσαρμόσιμη, ευανάγνωστη και εύκολη στη χρήση της. Η διπλή φύση της, της επιτρέπει να χρησιμοποιείται τόσο σε διαδραστικό όσο και σε μη διαδραστικό script. Έχει τη δυνατότητα να χρησιμοποιηθεί σε scripts χωρίς γραφική εμφάνιση, ενσωματωμένο σε γραφικές εφαρμογές ή και σε ιστοσελίδες ([Matplotlib for Python Developers, 2009](#)).

3.1.2 Front-End

Τα δύο βασικά εργαλεία που χρησιμοποιήθηκαν για να στηθεί η εφαρμογή, είναι το HTML και το CSS. Για να είναι όμως η εφαρμογή πιο φιλική προς τον χρήστη και πιο σύγχρονη χρησιμοποιήθηκε ένα template της HTML, που είναι σχεδιασμένο πάνω στο template του Django, το *Jinja2*. Είναι γρήγορο καθώς και ευρέως χρησιμοποιούμενο και ασφαλές, με την εκτέλεση του προαιρετικού sandboxed template. Το Jinja2 είναι πιο ευανάγνωστο επειδή η σύνταξή του είναι ορατά ξεχωριστή από τον κώδικα του HTML ([Efficient Way Of Web Development Using Python And Flask, 2015](#)).

3.1.3 Back-End

Για την δημιουργία μιας εφαρμογής με τη χρήση της γλώσσας Python, υπάρχουν 2 βασικά, και πιο γνωστά frameworks. Το πρώτο είναι το Django και το δεύτερο το Flask. Η συγκεκριμένη εφαρμογή επιλέχθηκε να γίνει με το Flask.

Το Flask είναι ένα αρκετά ελαφρύ web application framework γραμμένο σε python και βασισμένο στην εργαλειοθήκη του WSGI και του Jinja2, για το front-end. Το Flask παίρνει τον ευέλικτο προγραμματισμό της python και παρέχει ένα απλό template για την δημιουργία μιας εφαρμογής. Μόλις γίνει import στην python, το Flask μπορεί να χρησιμοποιείται για εξοικονόμηση χρόνου κατά τη κατασκευή εφαρμογών. Προσπαθεί και διατηρεί τον βασικό του κορμό απλό και επεκτάσιμο. Υποστηρίζει διάφορα extensions όπως: για object-relation mappers, form validations, upload handling, καθώς και authentication. Δυστυχώς όμως, το Flask δεν υποστηρίζει database abstraction layer, form validation ή κάποιο άλλο component.

Μία ακόμα χρήσιμη βιβλιοθήκη που χρησιμοποιείται στο web development, και ειδικότερα στο κομμάτι του authentication είναι η *flask_login* καθώς παρέχει ένα μεγάλο πλήθος συναρτήσεων. Η συγκεκριμένη βιβλιοθήκη είναι πολύ χρήσιμη για τον λόγο ότι δίνει τη δυνατότητα διαχείρισης users. Είναι υπεύθυνη να εκτελεί το logging in, το

logging out καθώς και πότε τελειώνει το session ενός user. Κάποιες από τις συναρτήσεις που παρέχει είναι οι εξής:

- **login_user**
- **login_required**
- **logout_user**
- **current_user**
- **UserMixin**
- **LoginManager**

Και αντίστοιχα από το Flask χρησιμοποιούνται οι εξής:

- **Blueprint**
- **render_template**
- **request**
- **flash**
- **redirect**
- **url_for**

Οι παραπάνω συναρτήσεις θα εξηγηθούν σε παρακάτω κεφάλαιο ([Flask API](#)).

3.1.4 Database

Η SQLAlchemy είναι μία από τις βιβλιοθήκες της python για να προσφέρει υψηλή, σε περιβάλλον python, αλληλεπίδραση με τις σχεσιακές βάσεις δεδομένων όπως η Oracle, DB2, MySQL, PostgreSQL, SQLite. Η βιβλιοθήκη αυτή προσπαθεί να είναι απλή μέσα στον κώδικα, επιτρέποντας να αντιστοιχηθούν παλιά αντικείμενα της python και να μετατραπούν σε σχεσιακές βάσεις δεδομένων χωρίς όμως να αλλαχτεί κάτι ουσιαστικό στον κώδικα. Περιλαμβάνει επίσης, έναν ανεξάρτητο server με εκφράσεις της SQL αλλά και object-relational mapper (ORM) που της επιτρέπει να ενημερώνει αυτόματα τα αντικείμενα της εφαρμογής ([Essential SQLAlchemy, 2010](#)).

3.2 Αρχιτεκτονική προγράμματος

Με σκοπό την καλύτερη διαχείριση του κώδικα καθώς και την ευκολότερη προσθήκη νέων features, η οργάνωση του συνολικού κώδικα έχει διχοτομηθεί σε back-end και front-end. Ο διαχωρισμός αυτός κάνει τον κώδικα πιο ευανάγνωστο, πιο εύκολο στη συντήρηση και μειώνει συνολικά τη χρονική δαπάνη για debugging. Έχοντας το front-end και το back-end ξεχωριστά, ο προγραμματιστής δύναται να κάνει αλλαγές, διορθώσεις και επεκτάσεις στο εκάστοτε κομμάτι δίχως να επηρεάζει άλλα μέρη του κώδικα, άθελα ή και θελημένα, δημιουργώντας έτσι μια συνολική εφαρμογή που αποτελείται από ανεξάρτητα κομμάτια πλήρως εναρμονισμένα μεταξύ τους. Σημαντικότερο σημείο του διαχωρισμού αυτού είναι επίσης η δυνατότητα εξειδίκευσης που μπορεί να δημιουργηθεί, καθώς όσο η εφαρμογή φορτίζεται και επεκτείνεται με περισσότερα modules και features, η συντήρηση της μπορεί να διαχωριστεί σε εξειδικευμένους back-end και front-end developers, οι οποίοι θα μπορούν, όπως προαναφέρθηκε, να κάνουν αλλαγές και διορθώσεις δίχως η δουλειά του ενός να επηρεάζει τον άλλο.

3.2.1 Back-end Authentication

Οι περισσότερες εφαρμογές στο διαδίκτυο πλέον έχουν το χαρακτηριστικό πως ένας χρήστης μπορεί να έχει τον δικό του προσωπικό λογαριασμό, ο οποίος θα είναι αποθηκευμένος στο cloud ή σε κάποιο local database.

Με τη χρήση λοιπόν modules και βιβλιοθηκών που αναφέρθηκαν παραπάνω (βλ. 2.5.3) επιτυγχάνεται η διαδικασία του authentication.

Στη συνέχεια, θα δούμε τις συναρτήσεις που χρησιμοποιήθηκαν για την υλοποίηση του authentication στην εφαρμογή, από την βιβλιοθήκη του flask-login το οποίο προσφέρει, όπως προαναφέρθηκε, ένα user session management, δηλαδή το login, logout και sign up του χρήστη.

- **login_user:** Η συγκεκριμένη συνάρτηση ελέγχει αν ο χρήστης συνδέθηκε και επιστρέφει μία boolean μεταβλητή για κάθε περίπτωση, True αν κατάφερε να συνδεθεί και False αν όχι.

```
if user:
    if check_password_hash(user.password, password):
        flash("Logged in", category="success")
        login_user(user, remember=True)
        return redirect(url_for("views.home"))
    else:
        flash("Incorrect password", category="error")
else:
    flash("Incorrect email", category="error")
```

Όπως φαίνεται και στον κώδικα, αν η συνθήκη είναι αληθής τότε με την εντολή login_user, ο χρήστης μπορεί και κάνει login.

- **logout_user:** Σε αντιστοιχία με το login_user, αυτή η συνάρτηση αποσυνδέει τον χρήστη από την εφαρμογή.

```
def logout():
    logout_user()
    return redirect(url_for("auth.login"))
```

Μόλις καλείται αυτή η συνάρτηση, logout(), τότε εκτελείται και το logout_user() και οδηγεί τον user στο login page.

- **login_required:** Με αυτή τη συνάρτηση εξασφαλίζουμε ότι ο συγκεκριμένος χρήστης έχει συνδεθεί, αρχικά, στην εφαρμογή και έχει γίνει authenticated πριν κληθεί το συγκεκριμένο view. Όπως φαίνεται στην κάτωθι εικόνα όταν το route πηγαίνει τον χρήστη σε συγκεκριμένο endpoint αυτού του URL.

```
@auth.route("/logout")
@login_required
def logout():
    logout_user()
    return redirect(url_for("auth.login"))
```

Με το login_required στη συγκεκριμένη συνάρτηση το πρόγραμμα εξασφαλίζει ότι ο χρήστης πρέπει να είναι ήδη συνδεδεμένος για να μπορέσει να κάνει logout_out.

- **current_user:** Δείχνει ποιος user είναι συνδεδεμένος εκείνη την στιγμή.

- **UserMixin:** Παρέχει προεπιλεγμένες εφαρμογές και συναρτήσεις για τις μεθόδους που το Flask-login περιμένει από τα user objects να έχει.

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))
    first_name = db.Column(db.String(150))
    last_name = db.Column(db.String(150))
    images = db.relationship('Image', backref='author', lazy=True)
```

Στον προηγούμενο κώδικα φαίνονται τα αντικείμενα που περιέχονται στο μοντέλο του χρήστη.

- **LoginManager:** Το συγκεκριμένο είναι μία κλάση που περιέχει τα settings για το logging in.

```
login_manager = LoginManager()
login_manager.login_view = "auth.login"
login_manager.init_app(app)
```

Εξασφαλίζει ότι ο χρήστης δεν θα μπορέσει να βρεθεί μέσα στην εφαρμογή αν δεν έχει κάνει log in ή sign-up ([Flask API](#))

3.2.2 Front-end

Σε κάθε εφαρμογή αυτό που ενδιαφέρει τον χρήστη είναι η εφαρμογή να είναι εύκολη στη χρήση της αλλά και κατανοητή (π.χ μηνύματα error η success). Εκτός από το κομμάτι του HTML που είναι ο σκελετός της εφαρμογής, πρέπει να ληφθεί υπόψη και το κομμάτι της λειτουργικότητας, πώς δηλαδή θα περαστούν τα δεδομένα που έχουμε στη βάση δεδομένων και σε κάποιο local folder στην οθόνη του χρήστη.

Για αυτόν ακριβώς τον λόγο το Flask παρέχει κάποιες συναρτήσεις που είναι ειδικά για αυτόν το σκοπό και θα πάμε να εξηγήσουμε ποιες χρησιμοποιήθηκαν και πώς.

- **Blueprint:** Ο βασικός λόγος που χρησιμοποιούνται τα blueprints είναι επειδή καταγράφουν λειτουργίες για να εκτελεστούν όταν ο χρήστης συνδεθεί στην εφαρμογή. Το Flask συνδυάζει τις views functions με τα

blueprints όταν θέλει να στείλει κάποια request και να αναπαράγει κάποια URLs από κάποιο endpoint σε κάποιο άλλο.

- **render_template:** Φορτώνει ένα αρχείο HTML για να το εμφανίσει στον χρήστη.

```

19 @auth.route("/login", methods=['GET', 'POST'])
20 def login():
21     if request.method == "POST":
22         email = request.form.get("email")
23         password = request.form.get("password")
24
25         user = User.query.filter_by(email=email).first()
26
27         if user:
28             if check_password_hash(user.password, password):
29                 flash("Logged in", category="success")
30                 login_user(user, remember=True)
31                 return redirect(url_for("views.home"))
32             else:
33                 flash("Incorrect password", category="error")
34         else:
35             flash("Incorrect email", category="error")
36     return render_template("login.html", user=current user)

```

Στο παραπάνω απόσπασμα κώδικα βλέπουμε ότι έχει επιλεγθεί ένα συγκεκριμένο route (το /login) και μόλις γίνεται αίτηση σε αυτό το URL εκτελείται η συνάρτηση login() μαζί με τις παραμέτρους της αίτησης (requests). Στο τέλος αυτής της συνάρτησης φαίνεται να επιστρέφεται το αρχείο (template) του login.html.

- **requests**

Τα HTTP requests χρησιμοποιούνται για πληροφορίες που στέλνονται, δημιουργούνται, επεξεργάζονται συνήθως προς/από μία βάση δεδομένων. Υπάρχουν πολλά HTTP requests, όπως POST, GET, DELETE, PUT, OPTIONS, HEADER, όμως αυτά που χρησιμοποιήθηκαν στην παρούσα εφαρμογή είναι: το GET το οποίο παίρνει κάποια δεδομένα είτε από την βάση είτε από κάποια φόρμα, και το POST, το οποίο καταχωρεί καινούρια δεδομένα στην βάση από μία φόρμα.

.

- **flash**

Μηνύματα ενημέρωσης προς τον χρήστη.

```
if user:
    flash("This email already exists!", category="error")
elif len(firstName) < 2:
    flash("First Name must be at least 2 characters", category="error")
elif len(lastName) < 2:
    flash("Last Name must be at least 2 characters", category="error")
elif len(email) < 4:
    flash("Email must be at least 4 characters", category="error")
elif len(password) < 8:
    flash("Password must be at least 8 characters", category="error")
elif password != passwordConf:
    flash("Password is not the same", category="error")
```

Ενημερώνουν τον χρήστη για κάθε περίπτωση που στην φόρμα sign up έχει κάνει κάποιο λάθος, ή δεν έχει συμπληρώσει κάποιο πεδίο.

- **redirect:** Οδηγεί τον χρήστη σε κάποιο άλλο route.
- **url_for:** Δημιουργεί ένα καινούριο URL στο συγκεκριμένο endpoint που έχει περαστεί σαν παράμετρος.

```

```

Στην συγκεκριμένη φωτογραφία δημιουργείται URL στον φάκελο static, για το αρχείο image.jpeg που βρίσκεται μέσα στο φάκελο pre-uploaded_images ([Flask API](#)).

3.3.3 Database Schema

Κάθε βάση δεδομένων έχει κάποια συγκεκριμένη δομή και ένα συγκεκριμένο περιεχόμενο, πάντα με βάση την εφαρμογή. Στην παρακάτω ενότητα θα εξηγήσουμε την δομή, τους πίνακες, και τις εγγραφές που δημιουργήθηκαν και αξιοποιήθηκαν για αυτό το επιθυμητό αποτέλεσμα.

User model

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))
    first_name = db.Column(db.String(150))
    last_name = db.Column(db.String(150))
    images = db.relationship('Image', backref='author', lazy=True)
```

Το μοντέλο του χρήστη περιέχει τα βασικά tuples που είναι: το όνομά του χρήστη, το επίθετο, το email του και ο κωδικός του καθώς επίσης και το id. Στη φωτογραφία όμως φαίνεται ότι υπάρχει και ακόμα ένα πεδίο, το images. Παρόλα αυτά δεν εμφανίζεται στη βάση για τον λόγο ότι είναι μία σχέση που συνδέει τον χρήστη με το μοντέλο του Image που θα εξηγήσουμε πιο κάτω.

	id	email	password	first_name	last_name
		Filter	Filter	Filter	Filter
1	1	jojosiliopoulos1998@gmail.com	sha256\$i2AszQJE\$490ed6f3aacdb74e286...	george	iliopoulos

Στη φωτογραφία φαίνεται πώς αποθηκεύονται τα δεδομένα από το front-end στη βάση δεδομένων. Στο πεδίο του password οι χαρακτήρες είναι hashed διότι χρησιμοποιήθηκε μια συγκεκριμένη βιβλιοθήκη, στο back-end, που ονομάζεται **werkzeug.security**. Από αυτήν τη βιβλιοθήκη χρησιμοποιήθηκε η συνάρτηση `generate_password_hash`.

Image model

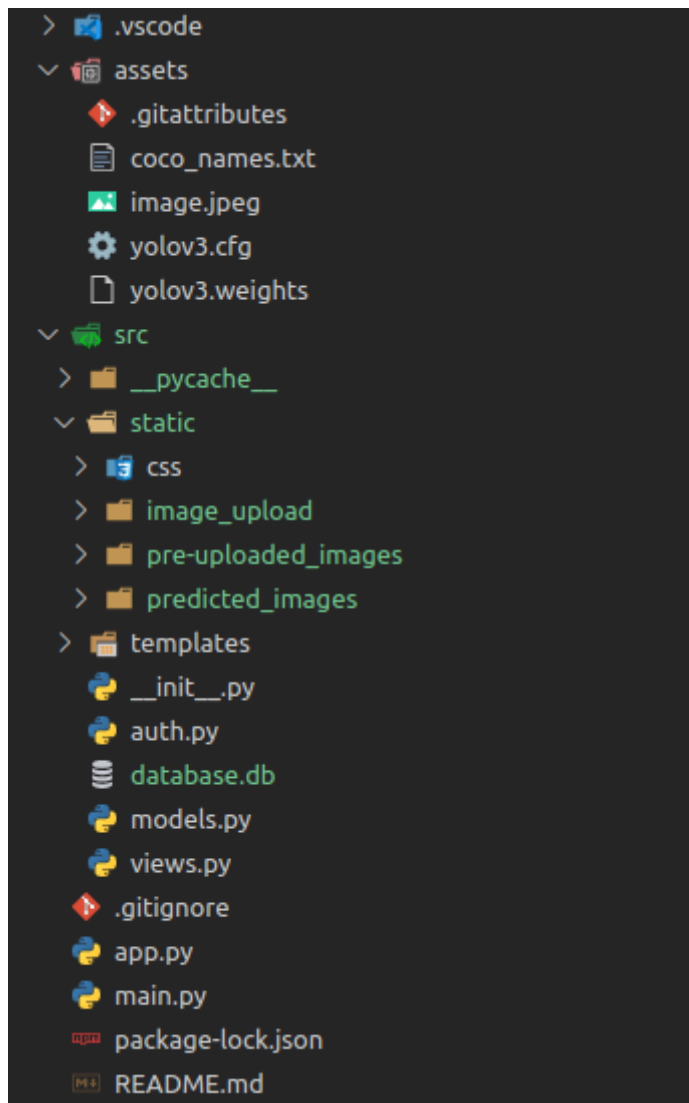
```
class Image(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    relational_id = db.Column(db.Integer)
    data = db.Column(db.String(100))
    date = db.Column(db.DateTime(timezone=True), default=datetime.now())
    title = db.Column(db.String(150))
    description = db.Column(db.String(500))
    labels = db.Column(db.String(200))
    quantity = db.Column(db.Integer)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
```

Όπως και το μοντέλο του χρήστη έτσι και αυτό περιέχει κάποιες εγγραφές οι οποίες είναι: το id της εικόνας, το relational_id που είναι για τα labels και quantities κάθε φωτογραφίας, το data που περιέχει το path για την φωτογραφία, το date που δείχνει πότε ανέβηκε στη βάση, τον τίτλο της φωτογραφίας, κάποια περιγραφή που επιθυμούσε να βάλει ο χρήστης καθώς και τα labels και quantity που αναγνώρισε το μοντέλο. Το user_id είναι το δευτερεύον κλειδί που συνδέει το user με τα images, διότι η σύνδεσή τους είναι one-to-many, πράγμα που σημαίνει ότι ο χρήστης έχει πολλές φωτογραφίες.

	id	relational_id	data	date	title	description	labels	quantity	user_id
		Filter	Filter	Filter	Filter	Filter	F...	Fil...	F...
1	1	1	src/static/image_upload/istockphoto-1060...	2021-09-08 22:56:17.022174	istockphoto-1060529042-612x612.jpg		NULL	NULL	1
2	2	1	NULL	2021-09-08 22:56:17.022174	NULL	NULL	person	1	1
3	3	2	src/static/image_upload/image.jpeg	2021-09-08 22:56:17.022174	image.jpeg		NULL	NULL	1
4	4	2	NULL	2021-09-08 22:56:17.022174	NULL	NULL	cup	1	1
5	5	2	NULL	2021-09-08 22:56:17.022174	NULL	NULL	person	5	1
6	6	2	NULL	2021-09-08 22:56:17.022174	NULL	NULL	laptop	1	1
7	7	3	src/static/image_upload/istockphoto-1146...	2021-09-08 23:20:33.793734	istockphoto-1146473249-612x612.jpg		NULL	NULL	1
8	8	3	NULL	2021-09-08 23:20:33.793734	NULL	NULL	person	8	1

Στην παραπάνω φωτογραφία όπου το user_id είναι ένα για τον πρώτο χρήστη που έκανε εγγραφή στην εφαρμογή, βλέπουμε πως ο χρήστης έχει 3 φωτογραφίες και κάθε φωτογραφία έχει το δικό της label και το δικό της quantity.

3.3 Επεξήγηση κώδικα



Για την επεξήγηση του κώδικα θα χρειαστεί η αρχιτεκτονική των φακέλων που χρησιμοποιήθηκαν, αλλά κυρίως θα έχουμε να κάνουμε με ένα συγκεκριμένο αρχείο.

Αυτό που προσπάθησα να κάνω ήταν να χωρίσω κάθε αρχείο με βάσει αυτό που πρέπει να κάνει (π.χ `auth.py` έχει να κάνει με το authentication της εφαρμογής, `templates` περιέχει όλα τα αρχεία `html` κτλ) για να μπορεί ο κώδικας να είναι ευανάγνωστος σε όλους και όχι χαοτικός.

Τα τρία βασικά αρχεία που χρειάστηκαν για την κατασκευή και δομή της εφαρμογής, και όχι αυτό που θα βλέπει ο χρήστης είναι: το `auth.py` το οποίο εξηγήθηκε παραπάνω, το `views.py` και το `__init__.py`.

Το αρχείο `__init__.py` είναι ένα αρχείο για το initialize της εφαρμογής καθώς και το `views.py` είναι οι συναρτήσεις που είναι συνδεδεμένες με κάποια URLs τα οποία επιστρέφουν κάποια HTTP Requests.

3.3.1 Feature extraction

Το κομμάτι του feature extraction, δηλαδή αυτό που αφορά σε όλη την εφαρμογή, βρίσκεται μέσα στο αρχείο `views.py` και εκτελείται όταν ο χρήστης ανεβάζει μία φωτογραφία. Π.Χ.:

Ξεκινώντας από τα προαναφερθέντα, το μοντέλο που χρησιμοποιήθηκε είναι το YoloV3. Είναι ένα pre-trained μοντέλο το οποίο διαθέτει ένα δικό του dataset που περιέχει 80+ ονόματα αντικειμένων.

```

39 def image_specification():
40     net = cv2.dnn.readNet('assets/yolov3.weights', 'assets/yolov3.cfg')
41     classes = []
42     # Passing the names of the txt file in a array
43     with open('assets/coco_names.txt', 'r') as f:
44         classes = f.read().splitlines()
45
46     user = User.query.get(current_user.id)
47     images = Image.query.filter_by(user_id=User.id).all()
48     for im in user.images:
49         if im.data:
50             ext = im.data.split(".")[-1]
51             if im.title:
52                 filename = im.title
53                 fullname = filename
54             else:
55                 filename = im.data.split("/")
56     img = plt.imread(user.images[-1].data)
57     height, width, deapth = img.shape
58
59     # Make sure all the imported image will have the same size and also scaling down the image
60     blob = cv2.dnn.blobFromImage(
61         img, 1/255, (416, 416), (0, 0, 0), swapRB=True, crop=False)
62     net.setInput(blob)
63     output_layers = net.getUnconnectedOutLayersNames()
64     layerOutputs = net.forward(output_layers)
65
66     pouding_boxes = []
67     predictions = []
68     class_ids = []
69

```

Όπως φαίνεται στις σειρές από 40-44, αυτό που κάνει ο κώδικας είναι να διαβάζει, αρχικά, τα weights και το configuration του μοντέλου και να τα αρχικοποιεί σε μια μεταβλητή. Αφού ολοκληρωθεί αυτή η εναρκτήρια διαδικασία, ακολουθεί η αρχικοποίηση των ονομάτων σε νέα μεταβλητή.

Αυτό που γίνεται στις σειρές από 46-56 είναι το extraction του ονόματος της φωτογραφίας σε περίπτωση που ο χρήστης δεν επιθυμήσει να βάλει όνομα.

Σε κάθε περίπτωση που γίνεται import μία φωτογραφία στο μοντέλο, πρέπει να προσαρμοστεί με βάση το configuration του μοντέλου. Στις σειρές 50-54 αυτό που γίνεται είναι η μετατροπή των pixels από τα χρώματα της φωτογραφίας από 0-255 σε 0-1, μια διαδικασία γνωστή και ως scale down. Μόλις ολοκληρωθεί, η φωτογραφία εισάγεται μέσα στο μοντέλο.

```

70     for output in layerOutputs:
71         for detection in output:
72
73             # Stores all the classes predictions of the image, finding the location of the highest score and then store it
74             scores = detection[5:]
75             class_id = np.argmax(scores)
76             prediction = scores[class_id]
77
78             if prediction > 0.5:
79
80                 center_x = int(detection[0] * width)
81                 center_y = int(detection[1] * height)
82                 w = int(detection[2] * width)
83                 h = int(detection[3] * height)
84                 x = int(center_x - w/2)
85                 y = int(center_y - h/2)
86
87                 # Finding out how many objects image contains
88                 pouding_boxes.append([x, y, w, h])
89                 predictions.append((float(prediction)))
90                 class_ids.append(class_id)
91
92             # Finding out how many objects found
93             boxes = cv2.dnn.NMSBoxes(pouding_boxes, predictions, 0.5, 0.4)
94             font = cv2.FONT_HERSHEY_PLAIN
95             colors = np.random.uniform(0, 255, size=(len(pouding_boxes), 3))
96
97             # Showing in the new image the pouding boxes with label name and a prediction
98
99             distinct_labels = []
100             quantity_labels = []
101             all_labels = []
102
103             fig, axes = plt.subplots()

```

Στις σειρές 70-90, αυτό που συμβαίνει είναι το prediction της φωτογραφίας, δηλαδή ο υπολογισμός των αντικειμένων που κατάφερε να εντοπίσει το μοντέλο έχοντας βρει το υψηλότερο score, max prediction, με το αντίστοιχο όνομα του label.

Στις σειρές 93-95, δημιουργούνται τα bounding boxes, τα οποία δείχνουν στον χρήστη τα αντικείμενα που εντοπίστηκαν πάνω στη φωτογραφία.

```

104
105     for i in boxes.flatten():
106         x, y, w, h = bounding_boxes[i]
107         label = str(classes[class_ids[i]])
108         prediction = str(round(predictions[i], 2) * 100)
109         color = colors[i]
110
111         rect = plt.Rectangle((x, y), w, h, linewidth=1,
112                             edgecolor='b', facecolor='none')
113         label_text = plt.text((x + 2), (y + 20), label)
114         prediction_text = plt.text((x + 2), (y + 37), prediction + "%")
115         plt.xticks([])
116         plt.yticks([])
117         axes.add_patch(rect)
118
119         distinct_labels.append(label)
120         all_labels.append(label)
121         distinct_labels = list(set(distinct_labels))
122
123         quantity_labels = []
124         for value in distinct_labels:
125             quantity_labels.append(
126                 len([temp_label for temp_label in all_labels if value == temp_label]))
127
128         rel_id = []
129         distinct_labels = np.array(distinct_labels)
130         user = User.query.get(current_user.id)
131         images = Image.query.filter_by(user_id=current_user.id).all()
132         for im in images:
133             if im.data:
134                 rel_id.append(im.data)
135
136         for upd in range(len(distinct_labels)):
137             db.session.add(Image(relational_id=len(
138                 rel_id), labels=distinct_labels[upd], quantity=quantity_labels[upd], user_id=current_user.id))
139
140         db.session.commit()
141         axes.imshow(img)
142         if user.images[-1].title:
143             prediction_image = f"src/static/predicted_images/{fullname}"
144             plt.savefig(prediction_image, bbox_inches="tight")
145         else:
146             prediction_image = f"src/static/predicted_images/{filename}"
147             plt.savefig(prediction_image, bbox_inches="tight")

```

Στη συνέχεια, στις σειρές 105-128 το μοντέλο τοποθετεί τα ήδη κατασκευασμένα bounding boxes πάνω στη φωτογραφία για την επίτευξη του καλύτερου δυνατού visualization του αποτελέσματος.

Τέλος, στις σειρές 142-147 αποθηκεύει την φωτογραφία σε έναν local folder, με βάση τις μεταβλητές fullname και filename, που είχαν δημιουργηθεί προηγουμένως στον κώδικα, και στις οποίες είναι κατοχυρωμένο το όνομα της φωτογραφίας.

3.3.2 Initialize

Για να μπορέσει μια εφαρμογή να εκτελέσει όλες αυτές τις συναρτήσεις που υπάρχουν μέσα στον κώδικα, πρέπει πρώτα να επιλυθούν κάποιες θεμελιώδεις συναρτήσεις. Αυτές οι συναρτήσεις είναι υπεύθυνες για την εκκίνηση του προγράμματος καθώς θα τρέξουν πρώτες. Πρόκειται για μια

μέθοδο γνωστή και ως initialization της εφαρμογής.

```

1  from flask import Flask
2  from flask_sqlalchemy import SQLAlchemy
3  from os import path
4  from flask_login import LoginManager
5
6  db = SQLAlchemy()
7  DB_NAME = "database.db"
8
9  | You, 5 months ago • Initialize the app
10 def create_app():
11     app = Flask(__name__)
12     app.config['SECRET_KEY'] = "thisisasecretkey"
13     app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:/// {DB_NAME}'
14     app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
15     db.init_app(app)
16
17     from .views import views
18     from .auth import auth
19
20     app.register_blueprint(views, url_prefix="/")
21     app.register_blueprint(auth, url_prefix="/")
22
23     from .models import User, Image
24
25     login_manager = LoginManager()
26     login_manager.login_view = "auth.login"
27     login_manager.init_app(app)
28
29     @app.before_first_request
30     def create_tables():
31         from .models import Image, User
32         db.create_all(app=app)
33         print("Created database")
34
35
36     @login_manager.user_loader
37     def load_user(id):
38         return User.query.get(int(id))
39
40     return app

```

Όπως φαίνεται και στη φωτογραφία, υπάρχουν κάποιες συναρτήσεις και κλάσεις που αρχικοποιούνται σε κάποιες μεταβλητές, με σκοπό αυτές οι μεταβλητές να χρησιμοποιηθούν αργότερα.

Έχοντας κάνει πρώτα import βιβλιοθήκες και modules, η διαδικασία που έπεται είναι η δημιουργία μιας βάσης.

Αφού ολοκληρωθεί και αυτό, τότε περνάμε στον άμεσο σχεδιασμό της εφαρμογής, εξετάζοντας με τι τρόπο θα τρέξει. Το πρώτο πράγμα που πρέπει να γίνει είναι να δομηθεί η εφαρμογή, και αυτό γίνεται στη σειρά 11, όπου αρχικοποιείται στη μεταβλητή `app` η κλάση `Flask`. Στις σειρές 12-15, εξασφαλίζεται η ασφάλεια της βάσης με ένα `SECRET_KEY`, έπειτα το endpoint του URI που θα έχει η βάση και τέλος μία συνάρτηση, η οποία κάνει `initialize` την εφαρμογή με το `setup` της συγκεκριμένης βάσης.

Στη συνέχεια στις σειρές 17-27 το πρώτο πράγμα που παρατηρούμε είναι ένα διαφορετικού είδους `import`. Οι σειρές 17 και 18 αντικατοπτρίζουν την επιλογή μου ως δημιουργού να ερμηνεύονται τα αρχεία `auth` και `views` ως βιβλιοθήκες. Τέλος, στις σειρές 25-27 αρχικοποιώ την κλάση `LoginManager` σε μια μεταβλητή και την κάνει και αυτή `initialize` στην εφαρμογή.

3.3.3 HTTP Requests

Το διαδίκτυο διαθέτει μια τεράστια ποικιλία δεδομένων τα οποία υπάρχουν σε διάφορους servers. Για να μπορέσει κάποιος να έχει πρόσβαση σε αυτά τα δεδομένα θα πρέπει πρώτα ο browser να μπορέσει να στείλει ένα αίτημα στον server για να εμφανίσει τα κατάλληλα δεδομένα που αιτήθηκε ο χρήστης. Το πρωτόκολλο HTTP, είναι η βασική μορφή που χρησιμοποιείται για τη δομή αιτήσεων και απαντήσεων για την αποτελεσματικότερη επικοινωνία μεταξύ ενός client και ενός server. Το μήνυμα που θα σταλθεί από τον client στον server ονομάζεται **HTTP Request**. Ένα συνηθισμένο παράδειγμα που χρησιμοποιείται για να κατανοηθούν τα HTTP Request είναι αυτό του σερβιτόρου-μάγειρα. Όταν ένας πελάτης (client) επιθυμεί να δειπνίσει σε ένα εστιατόριο, το πιάτο που επιθυμεί να πάρει δεν θα το ζητήσει από τον ίδιο τον μάγειρα (server) αλλά από τον σερβιτόρο. Οπότε αυτό που γίνεται είναι ότι ο σερβιτόρος πηγαίνει στον πελάτη και παίρνει το αίτημά του. Στη συνέχεια, αφού ο σερβιτόρος καταθέτει το αίτημα του πελάτη στον μάγειρα και μόλις η παραγγελία (response) είναι έτοιμη, τότε ο σερβιτόρος την παραδίδει στον πελάτη. Στο συγκεκριμένο όμως παράδειγμα αξίζει να αναφερθεί και ο ρόλος του σερβιτόρου ο οποίος μπορεί να παρομοιαστεί ως το API, το οποίο θα αναλυθεί παρακάτω.

Υπάρχουν πολλοί τύποι HTTP Requests, αλλά θα αναλυθούν παρακάτω οι πιο συνηθισμένοι:

- **GET:** Το GET χρησιμοποιείται όταν θέλουμε να τραβήξουμε δεδομένα από κάπου αλλά και να αιτηθούμε από τον server. Η μέθοδος αυτή είναι η πιο γνωστή από της μεθόδους HTTP.
- **POST:** Μία ακόμα γνωστή μέθοδος HTTP είναι η POST. Το POST είναι αυτό που στέλνει δεδομένα στον server, για να δημιουργήσει ή να επεξεργαστεί μια πληροφορία.
- **PUT:** Η μέθοδος PUT είναι παρόμοια με την μέθοδο POST, επειδή και αυτή στέλνει δεδομένα στον server για να δημιουργήσει ή να επεξεργαστεί μία πληροφορία. Η διαφορά τους είναι πως η PUT, αν κληθεί πολλές φορές, το αποτέλεσμα θα είναι πάντα ίδιο.
- **DELETE:** Είναι η πιο απλή μέθοδος, κάνει αυτό που λέει, χρησιμοποιείται για τη διαγραφή πληροφοριών από συγκεκριμένο URL
- **HEAD:** Η μέθοδος HEAD είναι παρόμοια με την μέθοδο GET με την διαφορά ότι η HEAD δεν έχει κάποιο message-body στο response, δηλαδή δεν μπορεί να δει κάτι ο χρήστης. Είναι χρήσιμη για την ανάκτηση metadata που έχουν γραφτεί ανάλογα με τα headers, αλλά χωρίς να μεταφέρει ολόκληρο το περιεχόμενο. Αυτή η μέθοδος χρησιμοποιείται κυρίως για το testing κάποιων hyperlink, από που θα είναι προσβάσιμη η εφαρμογή, πόσο έγκυρη είναι ([HTTP Request Methods – What are HTTP Requests?](#)).

Application Programming Interface (API)

Από την στιγμή που καλύφθηκε ένα μεγάλο μέρος ως προς την επεξήγηση των HTTP Requests και το πώς δουλεύουν, πρέπει να εξηγηθεί πώς ακριβώς επικοινωνούν ο client με τον server, διότι είπαμε πως η κάθε HTTP μέθοδος είναι ο τρόπος με τον οποίο ένα χρήστης ζητάει ή στέλνει κάποια δεδομένα στον server. Στο σημείο αυτό θα απαντηθεί πώς στέλνονται αυτά τα δεδομένα με την μέθοδο του API.

Το API ή αλλιώς *Application Programming Interface* είναι αυτό που συνδέει εφαρμογές μεταξύ τους. Μπορεί να χαρακτηριστεί σαν ένας τύπος

διεπαφής λογισμικού που προσφέρει υπηρεσίες σε άλλα κομμάτια του λογισμικού.

Τα Web APIs είναι οι καθορισμένες διεπαφές μέσω των οποίων συμβαίνουν κάποιες αλληλεπιδράσεις μεταξύ της επιχείρησης και των εφαρμογών που χρησιμοποιούν τα assets της επιχείρησης. Ας πάρουμε δηλαδή για παράδειγμα το API του Google Search.

The left screenshot shows the Google Search API documentation page. It includes a sidebar with navigation links like 'Introduction', 'Docs', 'Example Code', 'Authentication', 'Endpoints', 'Status', 'Search', 'News', 'Images', 'Crawl', and 'SERP'. The main content area is titled 'Perform Google Search' and shows 'Parameters' (query: the string query to perform search) and 'Returns' (json: a list of results with the link, description, and title for each result). It also lists 'AUTHORIZATIONS: apiKey' and 'PATH PARAMETERS: query' (required, string (Query)).

The right screenshot shows a 'Request samples' interface. It has tabs for 'cURL' and 'Node.js'. The 'Node.js' tab is selected, showing a code snippet for a search request. Below it, the 'Response samples' section shows a JSON response for a search query, including fields like 'title', 'link', and 'description' for various results.

Όπως φαίνεται στην φωτογραφία αυτό που έκανα ήταν να ζητήσω από το API του Google Search να μου εμφανίσει αποτελέσματα με τον τίτλο `searchForInstagramWith100Results` και στα δεξιά της οθόνης εμφάνισε, σε μορφή JSON, ένα array από objects που περιέχει μέσα τον τίτλο του αποτελέσματος από την αναζήτηση, το link, καθώς και ένα μικρό description (API).

3.4 Δημοσίευση Εφαρμογής (Deployment)

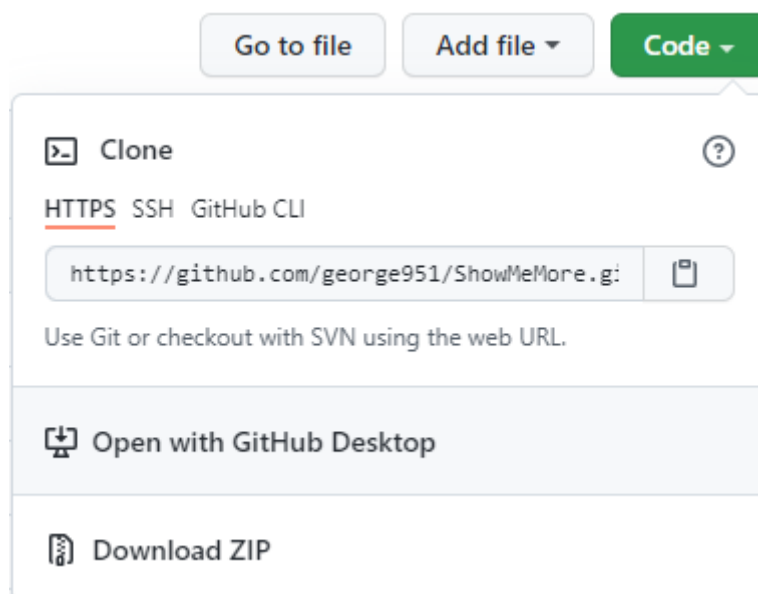
Το release μιας εφαρμογής είναι ένα πολύ σημαντικό βήμα για την αναγνώρισή της μέσα από το web. Όταν δημοσιεύεται κάποια εφαρμογή, αυτό που πρέπει να ληφθεί υπόψιν είναι να μπορεί να υποστηριχθεί από διαφορετικά λογισμικά (π.χ Windows, MacOS, Android, IOS, Linux). Οι clients έχουν διαφορετικά είδη υλικού και λογισμικού που απαιτούνται κατά την πρόσβασή τους στις υπηρεσίες της εφαρμογής. Οι σταθεροί και φορητοί υπολογιστές κατασκευάζονται χρησιμοποιώντας διαφορετικές αρχιτεκτονικές επεξεργαστών και λειτουργικά συστήματα από αυτά που

παρέχονται στα κινητά τηλέφωνα. Όμως πρέπει όλοι οι clients να έχουν πρόσβαση και να χρησιμοποιούν την ίδια υπηρεσία, ανεξάρτητα από τις δυνατότητες και τις προϋποθέσεις τους.

Ενώ πολλές ομάδες του development εξακολουθούν να επιλέγουν την μέθοδο του host που χρησιμοποιούν εσωτερική υποδομή πληροφορικής, οι πάροχοι υπηρεσιών cloud όπως Amazon Web Services, Google Cloud Platform, και Microsoft Azure προσφέρουν τα *IaaS(Infrastructure-as-a-Service)* και *PaaS(Platform-as-a-Service)* τα οποία βοηθούν τους προγραμματιστές να αναπτύξουν εφαρμογές σε ζωντανά περιβάλλοντα χωρίς το πρόσθετο οικονομικό, αλλά και διοικητικό βάρος της διαχείρισης των δικών τους storage και virtualization servers ([Cloud computing](#)).

Στην προκειμένη όμως περίπτωση της συγκεκριμένης εφαρμογής, ο τρόπος που χρησιμοποιήθηκε για να γίνει εγκατασταθεί αυτή η εφαρμογή είναι σε localhost server. Και αυτό, επειδή ο όγκος των χρηστών που χρησιμοποιούν την εφαρμογή αυτή δεν είναι πολύ μεγάλος. Συνεπώς, στην περίπτωση που ένας χρήστης επιθυμεί να κατεβάσει την εφαρμογή, τα μόνα πράγματα που πρέπει να έχει είναι ένας λογαριασμός στο GitHub και τον terminal του Git για να μπορέσει να κατεβάσει την εφαρμογή καθώς και τη βιβλιοθήκη της Python για να μπορέσει να την τρέξει.

Αυτό που καλείται να κάνει ο χρήστης για να μπορέσει να κατεβάσει την εφαρμογή είναι να μεταβεί στη σελίδα του [GitHub](#) και να πατήσει το κουμπί που αναγράφει Code.

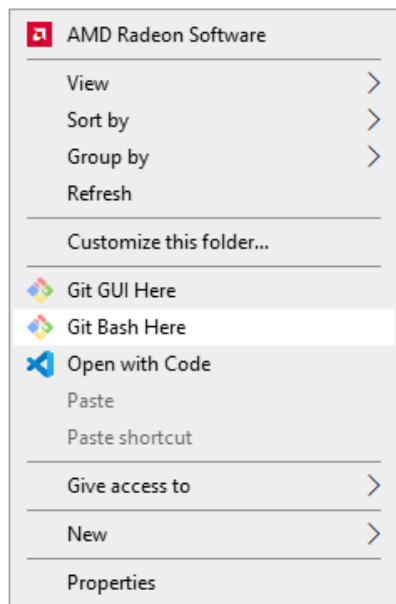


Στην περίπτωση που βρίσκεται σε περιβάλλον Windows, πρέπει να ακολουθήσει τα παρακάτω βήματα:

Να ανοίξει τον φάκελο με όνομα **ShowMeMore-master->Right-Click-Git**

Bash Here.

Name	Date modified	Type	Size
.vscode	9/9/2021 7:01 PM	File folder	
assets	9/9/2021 7:01 PM	File folder	
src	9/9/2021 7:01 PM	File folder	
.gitignore	9/9/2021 7:01 PM	Text Document	1 KB
app.py	9/9/2021 7:01 PM	PY File	1 KB
main.py	9/9/2021 7:01 PM	PY File	3 KB
package-lock.json	9/9/2021 7:01 PM	JSON File	1 KB
README.md	9/9/2021 7:01 PM	MD File	2 KB
requirements.txt	7/7/2021 1:01 PM	Text Document	1 KB



Μόλις το κάνει αυτό θα του ανοίξει έναν terminal στο οποίο και θα τρέξει πρώτα την εντολή **py -m pip install -r requirements.txt** για να εγκαταστήσει τις βιβλιοθήκες που χρειάζονται για να τρέξει η εφαρμογή και μετά την εντολή **py app.py**, και πατώντας **Ctrl + Left Click** θα τον κατευθύνει στον αντίστοιχο browser.

```
* Serving Flask app "src" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 118-758-228
```

Στην περίπτωση που βρίσκεται σε περιβάλλον Linux, πρέπει να ακολουθήσει τα παρακάτω βήματα:

Να πάει πάλι στο συγκεκριμένο project του GitHub, να αντιγράψει το link HTTPS, να ανοίξει ένα terminal και να γράψει την εντολή **git clone https://github.com/george951/ShowMeMore.git**. Στην περίπτωση που ο χρήστης δεν επιλέξει πού θέλει να αποθηκευτεί ο φάκελος το λογισμικό θα τον αποθηκεύσει στο home.

Τέλος, αυτό που πρέπει να κάνει ο χρήστης είναι να πάει στο φάκελο της εφαρμογής και να τρέξει **python3 -m pip install -r requirements.txt** για να εγκαταστήσει τις βιβλιοθήκες που χρειάζονται για να τρέξει η εφαρμογή και μετά την εντολή **python3 app.py** αν έχει την έκδοση 3.8 της Python.

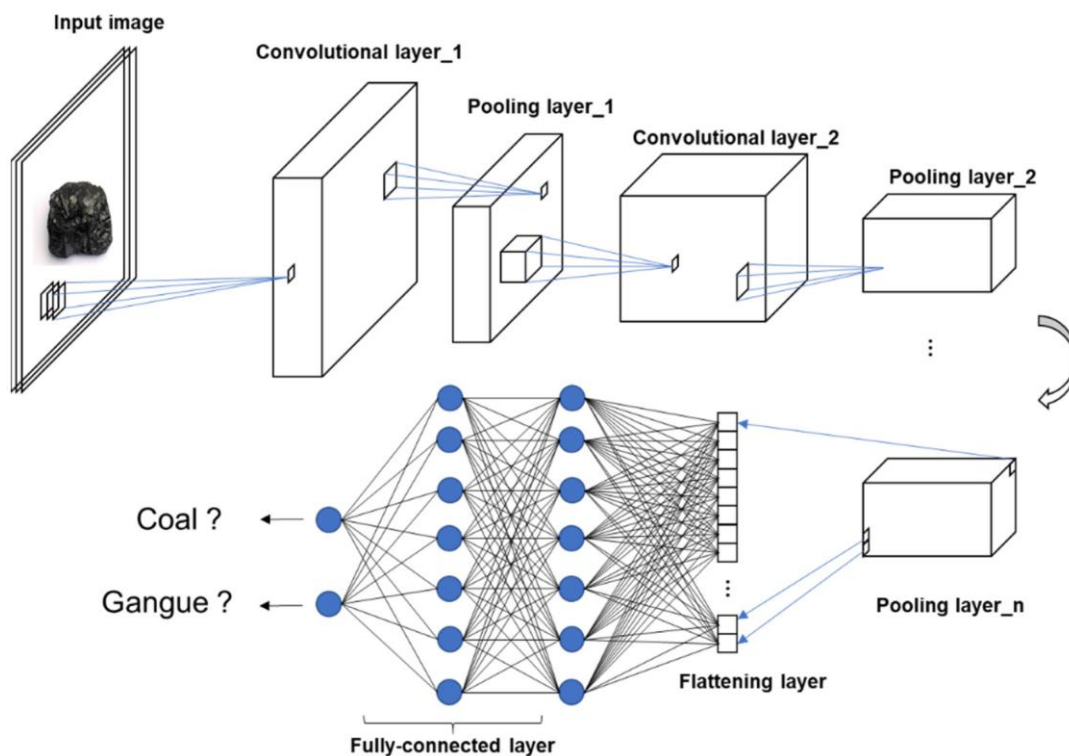
Σε κάθε ένα από τα δύο λειτουργικά συστήματα, όταν ο χρήστης πατήσει το link που τον κατευθύνει στον αντίστοιχο browser, υπάρχει περίπτωση να μην εμφανίζονται κάποια assets της εφαρμογής. Σε αυτή την περίπτωση ο χρήστης πατάει το F12 που του ανοίγει τα devtools του browser, ψάχνει την καρτέλα Network, κάνει check σε αυτό που λέει Disable Cache και τέλος κάνει ένα refresh.

4. Συμπεράσματα

Η εφαρμογή που αναπτύχθηκε χρησιμοποιεί τη βιβλιοθήκη επεξεργασίας εικόνας openCV. Το OpenCV παρέχει εύκολη πρόσβαση σε μια σειρά ισχυρών λειτουργιών επεξεργασίας εικόνας. Εκτός από αυτή του την ικανότητα, περιέχει και μια διόλου ευκαταφρόνητη σειρά λειτουργιών αναφορικά σε μοντέλα CNN. Η επιλογή αυτής της βιβλιοθήκης έγινε διότι διαθέτει τις κατάλληλες συναρτήσεις, κλάσεις και βιβλιοθήκες που χρειάζεται ένας προγραμματιστής για να δημιουργήσει μία τέτοια εφαρμογή χειρισμού φωτογραφιών αλλά και γιατί είναι η πιο γνωστή open-source βιβλιοθήκη για object detection, και γενικότερα για το machine learning. Στη συνέχεια, με την χρήση του YoloV3 αλγορίθμου αναγνώρισης αντικειμένων, ο οποίος μου παρείχε το μοντέλο καθώς και το dataset, ονόματι COCO, κατάφερα να δημιουργήσω ένα πρόγραμμα αναγνώρισης αντικειμένων, με τη δυνατότητα να αναγνωρίζει μέχρι και 80 αντικείμενα μέσα σε μια φωτογραφία.

Παρόλο που η εφαρμογή είναι γενικά λειτουργική, έχει κάποια όρια αναγνώρισης. Αρχικά, όταν μέσα σε μια φωτογραφία υπάρχουν πολλά αντικείμενα, το ένα πάνω στο άλλο, τότε ο αλγόριθμος δεν μπορεί να τα αναγνωρίσει όλα. Αυτό συμβαίνει διότι κατά τη διαδικασία του feature extraction από την φωτογραφία, τα pooling layers πρέπει να είναι περισσότερα για να μπορέσει να καλυφθεί το μεγαλύτερο μέρος της

φωτογραφίας.



Εικόνα 4.1: Παράδειγμα ενός μοντέλου αναγνώρισης που παίρνει σαν είσοδο ένα ορυκτό και προσπαθεί να δει αν είναι κάρβουνο ή κάποιο άλλο άχρηστο ορυκτό

Παράδειγμα του παραπάνω ισχυρισμού βλέπουμε στην εικόνα, όπου και το πρόγραμμα δύναται ουσιαστικά να αναγνωρίσει αν το προς απεικόνιση αντικείμενο είναι κάρβουνο ή ένα άχρηστο ορυκτό. Οι πιθανότητες να καταφέρει ο αλγόριθμος να αναγνωρίσει την φωτογραφία θα είναι πολύ μεγαλύτερες αν υπάρχουν περισσότερα conv και pooling layers με το pooling να μπορεί να συγκρίνει 1×1 . Ένας δεύτερος περιορισμός είναι ότι, το μοντέλο μπορεί να αναγνωρίσει ορισμένα μόνο από τα αντικείμενα της φωτογραφίας, δηλαδή μόνο τα αντικείμενα αυτά που υπάρχουν στο dataset. Αυτό μπορεί να τεθεί υπό έλεγχο με την καλύτερη μορφοποίηση του μοντέλου και την επέκταση του dataset, πράγμα το οποίο επιτυγχάνεται με νέα εκπαίδευση (training). Με αυτού του είδους τις αλλαγές το ποσοστό επιτυχίας αναγνώρισης αναμένεται να ανέλθει. ([New Object Detection Features in the OpenCV Library](#)).

Η χρήση του εν λόγω αλγορίθμου μπορεί και να συμπεριληφθεί στο back end τρίτων εφαρμογών, με παρόμοιο τρόπο που χρησιμοποιείται για εργαλεία τύπου Screen Readers και για στατιστικά στοιχεία από πληθώρα άλλων apps (π.χ Facebook, Instagram, Twitter).

4.1 Μελλοντικές Επεκτάσεις

Όσον αφορά στις μελλοντικές επεκτάσεις της εφαρμογής, αυτές είναι ομολογουμένως αρκετές. Αρχικά, θα μπορούσε να γίνει λόγος για εφαρμογή κινητού, έτσι ώστε ο χρήστης να μπορεί να αναγνωρίσει αντικείμενα μέσω live video και όχι ανεβάζοντας μια φωτογραφία, δηλαδή ζωντανά, εκείνη την στιγμή. Έπειτα, μια άλλη επέκταση/εν δυνάμει διαμόρφωση, με καλύτερη παρουσίαση της εφαρμογής, θα συνιστούσε μικρή αλλά σημαντική αναβάθμιση σε θέματα από 'στυλιστικής' αλλά και λειτουργικής άποψης. Τέλος, θα μπορούσε να δοθεί στον χρήστη η δυνατότητα να επιλέξει ο ίδιος, μεταξύ και άλλων αλγορίθμων, την αναγνώριση της φωτογραφίας, σε περίπτωση που επιθυμεί να βγάλει κάποια στατιστικά αποτελέσματα και συγκρίσεις.

5. Βιβλιογραφία - Δικτυογραφία

1. Stefan van der Walt, Johannes L. Schonberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors (2014), *scikit-image: image processing in Python*, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, USA
2. Ajith Abraham (2005), *Artificial neural Network*, Oklahoma State University, Stillwater, OK, USA
http://wsc10.softcomputing.net/ann_chapter.pdf
3. B. Yegnanarayana (2006), *Artificial Neural Networks*, Department of Computer Science and Engineering Indian Institute of Technology Madras Chennai
https://books.google.gr/books?hl=el&lr=&id=RTtvUVU_xL4C&oi=fnd&pg=PR9&dq=artificial+neural+networks&ots=Gd9ZxnCKQC&sig

[=7Lmr0PNbZdyiZJcp8C3LMpd343M&redir_esc=y#v=onepage&q&f=false](#)

4. Jason Brownlee (2011), *14 Different Types of Learning in Machine Learning*
<https://machinelearningmastery.com/types-of-learning-in-machine-learning/>
5. Wikipedia (2021), *Convolutional neural Network*,
https://en.wikipedia.org/wiki/Convolutional_neural_network
6. Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, Jurgen Schmidhuber, *Flexible, High Performance Convolutional Neural Networks for Image Classification*
<https://people.idsia.ch/~juergen/ijcai2011.pdf>
7. Fu Jie Huang, Yann LeCun, Leon Bottou (2003), *NORB: Generic Object Recognition in Images*, National Science Foundation
<https://cs.nyu.edu/~yann/research/norb/>
8. Ross Girshick, *Fast R-CNN*, Microsoft Research
https://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf
9. Liquan Zhao, Shuaiyang Li (2020), *Object Detection Algorithm Based on Improved YOLOv3*, Key Laboratory of Modern Power System Simulation and Control & Renewable Energy Technology
<https://www.mdpi.com/2079-9292/9/3/537>
10. Bilel Benjdira, Taha Khursheed, Anis Koubaa, Adel Ammar, Kais Ouni(2018), *Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3*
<https://arxiv.org/pdf/1812.10968.pdf>
11. Adel Ammar, Anis Koubaa1, Mohammed Ahmed1 , Abdulrahman Saad(2020), *Aerial Images Processing for Car Detection using Convolutional Neural Networks: Comparison between Faster R-CNN and YoloV3*
<https://arxiv.org/pdf/1910.07234.pdf>
12. Migran N. Gevorkyan, Anastasia V. Demidova, Tatiana S. Demidova, Anton A. Sobolev(2019), *Review and comparative analysis of machine learning libraries for machine learning*,

Department of Applied Probability and Informatics Peoples' Friendship University of Russia

13. P. N. Druzhkov, V. L. Erukhimov, N. Yu. Zolotikh, E. A. Kozinov, V. D. Kustikova, I. B. Meerov, and A. N. Polovinkin, *New Object Detection Features in the OpenCV Library*, Lobachevskii State University of Nizhni Novgorod
<http://www.uic.unn.ru/~zny/papers/pria2011.pdf>
14. Isromi Janwar (2009), *Matplotlib for Python Developers*, Academia and Packt Publishing Ltd
[https://d1wqtxts1xzle7.cloudfront.net/32168807/Matplotlib_for_Python_Developers_\(2009\)-with-cover-page-v2.pdf?Expires=1633196361&Signature=K7Fr5M5tpdFhxDP82dcwMO2-x9Gxd9VX-~~4Ui--j1~AKe9RyCIJQckybdjmTHRi2OOqmkps0YWZCWUYYZ2OnxY-XQNiNuoAUT829CD5jsQdM9FihWWJpa3R-8za0oQb2PTswJNgkfkZYT2DcRhtXMoUCsp9N2hjs6~1m1A4SQk1kkhE50jUJwxxS3LoDml-EDUFgcxNTENKNGNt~GA9L7Ik57LBEA9b8u3h0ykUdiKtGxrzMJNaus5DMQvHcZUaFCliG3~SahwxMXCeMWqziw6jUVQOYAFG5cx2ZIN~2TdDyD9rARw70AAo7GLqfdu4GlsaGIVN-pmvyLMowMw8Tw_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA](https://d1wqtxts1xzle7.cloudfront.net/32168807/Matplotlib_for_Python_Developers_(2009)-with-cover-page-v2.pdf?Expires=1633196361&Signature=K7Fr5M5tpdFhxDP82dcwMO2-x9Gxd9VX-~~4Ui--j1~AKe9RyCIJQckybdjmTHRi2OOqmkps0YWZCWUYYZ2OnxY-XQNiNuoAUT829CD5jsQdM9FihWWJpa3R-8za0oQb2PTswJNgkfkZYT2DcRhtXMoUCsp9N2hjs6~1m1A4SQk1kkhE50jUJwxxS3LoDml-EDUFgcxNTENKNGNt~GA9L7Ik57LBEA9b8u3h0ykUdiKtGxrzMJNaus5DMQvHcZUaFCliG3~SahwxMXCeMWqziw6jUVQOYAFG5cx2ZIN~2TdDyD9rARw70AAo7GLqfdu4GlsaGIVN-pmvyLMowMw8Tw_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA)
15. Fankar Armash Aslam, Hawa Nabeel Mohammed, Jummal Musab Mohd. Munir(2015), *Efficient Way Of Web Development Using Python And Flask*, Department Of Computer Engineering Anjuman-I-Islam's Kalsekar Technical Campus
<https://core.ac.uk/download/pdf/55305148.pdf>
16. Max Countryman(2021), *Flask-login*
<https://flask-login.readthedocs.io/en/latest/#flask-login>
17. Rick Copeland(2010), *Essential SQLAlchemy*
<https://kingauthor.net/books/Rick%20Copeland/Essential%20SQLAlchemy/Essential%20SQLAlchemy%20-%20Rick%20Copeland.pdf>
18. Flask API
<https://flask.palletsprojects.com/en/2.0.x/api/>

19. Last Call: The RapidAPI Blog, *HTTP Request Methods - What are HTTP Requests?*
<https://rapidapi.com/blog/api-glossary/http-request-methods/>
20. Wikipedia(2021), *API*
<https://en.wikipedia.org/wiki/API>
21. Ameet V Joshi (2020), *Machine Learning and Artificial Intelligence*, Microsoft (United States)
<http://203.162.88.123/bitstream/HVCNBCVT/2254/1/Ameet%20V%20Joshi%20-%20Machine%20Learning%20and%20Artificial%20Intelligence-Springer%20International%20Publishing%20%282020%29.pdf#page=22&zoom=100,0,0>
22. Deep AI, *Bias Vector*
<https://deepai.org/machine-learning-glossary-and-terms/bias-vector>
23. Towards Data Science, Anne Bonner (2019), *The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification*
<https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>
24. Wikipedia, *Cloud computing*
https://en.wikipedia.org/wiki/Cloud_computing#Service_models