Τμήμα Μηχανικών
Βιομηχανικής Σχεδίασης και Παραγωγής

ΕΦΑΡΜΟΓΗ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗΝ ΠΡΟΛΗΨΗ ΚΑΙ ΚΑΤΑΣΒΕΣΗ ΠΥΡΚΑΓΙΑΣ

ΧΟΝΔΡΟΓΙΑΝΝΗΣ ΜΙΧΑΗΛ

ΕΠΙΒΛΕΠΟΝΤΑΣ ΚΑΘΗΓΗΤΗΣ:ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΙΟΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ

ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΦΕΒΡΟΥΑΡΙΟΣ 2022

Μέλη Επιτροπής:

Η Διπλωματική/Πτυχιακή Εργασία εξετάστηκε από την κάτωθι Εξεταστική Επιτροπή:

ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
ΝΙΚΟΛΑΟΥ Γ.	ΛΕΚΤΟΡΑΣ	GRIGORIOS NIKOLAOU Digitally signed by GRIGORIOS NIKOLAOU Date: 2022.03.22 12:23:31 +02'00'
ΒΑΣΙΛΕΙΑΔΟΥ Σ.	ΕΠΙΚΟΥΡΗ ΚΑΘΗΓΗΤΡΙΑ	Soultana Vasileiadou Digitally signed by Soultana Vasileiadou Date: 2022.03.24 07:53:30 +02'00'
ΔΡΟΣΟΣ Χ.	ΕΔΙΠ	Christos Drosos Digitally signed by Christos Drosos Date: 2022.03.22 14:15:15 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Χονδρογιάννης Μιχαήλ του Χρήστου , με αριθμό μητρώου 71446449 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. »

Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Ευχαριστίες

Η παρούσα διπλωματική εργασίας εκπονήθηκε τον Φεβρουάριο του 2022 και δεν θα μπορούσε να είχε υλοποιηθεί χωρίς την πολύτιμη βοήθεια κάποιον ατόμων που θα ήθελα να ευχαριστήσω καθώς με στήριξαν ο καθένας με τον δικό του μοναδικό τρόπο.

Για αρχή θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Νικολάου Γρηγόριο που χάρη σε αυτόν και στο μαθήματα του «ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ - ΑΝΑΛΥΣΗ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ» γνώρισα για πρώτη φορά το πόσες δυνατότητες και εφαρμογές μπορεί να έχει η μηχανική μάθηση, καθώς και για τη στήριξη του κατά την διάρκεια της υλοποίησης της εργασίας αυτής.

Στην συνέχεια θα ήθελα να ευχαριστήσω πολύ την σύντροφο μου Αικατερίνη Μαργαρίτα Γκιώνη που ήταν διπλά μου από την αρχή μέχρι το τέλος την εργασίας αυτής και με στήριζε τόσο με τα λόγια της αλλά και με την βοήθεια της σε ότι χρειαζόμουν ώστε να μπορέσω να υλοποιήσω αυτή την εργασία.

Επίσης θα ήθελα να ευχαριστήσω τον Αθανάσιο Νικολέρη, χάρη στον οποίο μπόρεσα να έχω πρόσβαση σε ένα μεγάλο μέρος των υλικών της διπλωματικής μου, χορήγησε την κάμερα που χρησιμοποιήθηκε στην κατασκευή μου.

Καθώς και μέλη από την Ομάδα Εθελοντές Δασοπροστασίας & Πολιτικής Προστασίας Δήμου Ελληνικού Αργυρούπολη όπου με την μικρή συνδρομή τους μαζεύτηκε ένα ποσό για την αγορά υλικών για την κατασκευή του RFT. Επίσης θα ήθελα να ευχαριστώ ιδιαίτερα τον κ. Βασίλη Κρητικό Αντιδήμαρχο Περιβάλλοντος & Πολιτικής Προστασίας του Δήμου Ελληνικού Αργυρούπολης, τον κ. Νικόλαο Σταματόπουλο και την κ. Πασχαλιά Παλαιού που συμμετείχαν σε παρουσίαση της κατασκευής και έδωσαν τα σχόλια τους σχετικά με την χρήση της σε ρεαλιστικά σενάρια για να καλυφθούν οι ανάγκες του Υμηττού.

Τέλος θα ήθελα να ευχαριστώ την οικογένεια μου που ήταν εκεί να με στηρίζει και να με ενθαρρύνει να προσπαθώ να κάνω το καλύτερο που μπορώ ώστε να δημιουργήσω το παρακάτω σύστημα, που μπορεί μελλοντικά να βοηθήσει το περιβάλλον και την προστασία της ανθρώπινης ζωής.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την ανάπτυξη ενός σύνθετου συστήματος το οποίο θα μπορεί να παρακολουθεί μέσω κάμερας ένα συγκεκριμένο χώρο όπως πχ μια δασική έκταση και να ελέγχει σε ζωντανό χρόνο τον περιβάλλοντα χώρο για ένδειξη πυρκαγιάς.

Είτε σε μορφή καπνού είτε σε φλόγας, σε περίπτωση που γίνει ανίχνευση, το σύστημα θα ενημερώνει τον χρήστη σχετικά με την ύπαρξη κίνδυνου μέσω e-mail/sms/push- notification και θα μπορεί να επεμβαίνει απομακρυσμένα από οποιοδήποτε σημείο με πρόσβαση στο ιντερνέτ και να βλέπει σε ζωντανό χρόνο την εξέλιξη της πυρκαγιάς μέσω της κάμερας. [6]

Στην συνέχεια θα έχει πρόσβαση σε ένα ενσωματωμένο συστήματα πυρόσβεσης με 2 άξονες κίνησης, ώστε να μπορεί να γίνει άμεσος περιορισμός της πυρκαγιάς και να μην προλάβει να επεκταθεί, με αποτέλεσμα να κινδυνέψει μεγάλη δασική έκταση, η πανίδα και ενδεχομένως οι ανθρώπινες ζωές.

Η υλοποίηση του συστήματος αυτού στα πλαίσια της διπλωματικής θα γίνει σε περιορισμένη κλίμακα σε σύγκριση με την ιδανική μορφή που θα μπορούσε να κατασκευαστεί και να αναπτυχθεί σε κρίσιμα σημεία που υπάρχει μεγάλος κίνδυνος πυρκαγιάς.

Το σύστημα αποτελείται από 2 βασικά ανεξάρτητα μέρη, το μοντέλο μηχανικής μάθησης που σε συνεργασία με την κάμερα λαμβάνει εικόνες κάθε λεπτό και κάνει έλεγχο σχετικά με το αν υπάρχει πυρκαγιά ή όχι και ενημερώνει αναλόγως τον χρήστη με την χρήση ηλεκτρονικού υπολογιστή.

Αυτό το μοντέλο έχει αναπτυχθεί με την χρήση python και συγκεκριμένα με την χρήση TensorFlow[1] για να γίνει ανάπτυξη του μοντέλου αναγνώρισης εικόνας.

Επίσης, υπάρχει και το σύστημα κίνησης ενός ακροφυσίου συνδεδεμένο με ηλεκτροβάνα για παροχή νερού όπου επιτρέπει στον χρήστη να το ελέγχει μέσω ενός, συνδεδεμένου στο ιντερνέτ, μικροελεγκτή. Με την χρήση αυτού μπορεί να γίνει εύκολα και με ασφάλεια κατάσβεση στον περίγυρο χώρο όπου είναι τοποθετημένο το σύστημα αυτό, το οποίο θα αναφέρεται στην συνέχεια ως RFT (Remote Fire Tower).

Στην υλοποίηση χρησιμοποιήθηκε και επιπρόσθετη εμπορική πλατφόρμα στο νέφος για την εύκολη χρήση του μηχανισμού κίνησης και κατάσβεσης μέσω της πλατφόρμας Thinger.io[2] όπως και για την αυτόματη αποθήκευση εικόνων από την κάμερα έγινε χρήση του προγράμματος MacroRecorder[3].

Abstract

The main objective of this dissertation is the development of a complex system that will be able to monitor a specific area with a camera, such as a forest area and to monitor in real time the surrounding area for fire indication.

Either in the form of smoke or in a flame where in case of detection, the system will inform the user about the existence of danger via e-mail/ sms/push- notification and will be able to intervene remotely from anywhere with internet access and see in real time the evolution of fire through the camera. [6]

It will then have access to move an integrated fire extinguishing system with 2 drive axles, so that the fire can be reduced immediately and prevented from spreading, resulting in endangering a large area of forest, animals and potentially human lives.

The implementation of this system in the dissertation will be done on a limited scale compared to the ideal form that could be built and developed in critical areas where there is a high risk of fire.

The system consists of 2 basic independent parts, the machine learning model that in collaboration with the camera takes images every minute and checks whether there is a fire or not and informs the user accordingly using a computer.

The model has been developed using python and specifically the use of TensorFlow [1] to develop the image recognition model.

And the movement system of a nozzle connected to a solenoid valve and water supply where it allows the user to control through an Microcontroller connected to the internet. With this system you can easily and safely extinguish fires in the surrounding area where this system is located. The system will be referred to as RFT (Remote Fire Tower).

Additional commercial cloud based software from the was used in the implementation, both for the easy use of the movement and extinguishing mechanism through the Thinger.io platform [2], but also for the automatic storage of images from the camera, the MacroRecorder program was used [3].

Περιεχόμενα

Εισαγωγή

Σύμφωνα με μελέτες οι δασικές πυρκαγιές είναι το βασικότερο πρόβλημα το οποίο αντιμετωπίζουν τα δάση τόσο στην Ελλάδα αλλά και σε ολόκληρο τον κόσμο.

Στην Ελλάδα, συγκεκριμένα, πάνω από το 10% της επιφάνειάς της έχει μετατραπεί σε άγονη και βραχώδη γη με το πέρας των επαναλαμβανόμενων πυρκαγιών. Το 25,4% της συνολικής έκτασής της είναι δάση, φέρνοντας την έτσι τέταρτη στην Ευρώπη όσον αφορά τον δασικό πλούτο.

Οι δασικές πυρκαγιές είναι ένα φαινόμενο άλλοτε φυσικό (κεραυνοί, υψηλές θερμοκρασίες) άλλοτε όμως και λόγω ανθρώπινης παρέμβασης(εμπρησμός – αμέλεια). Εξαιτίας της έντονης αλλαγής του περιβάλλοντος από την κλιματική αλλαγή οι πυρκαγιές γίνονται ολοένα και πιο συχνές με αποτελέσματα την εμφάνιση επιπτώσεων για το περιβάλλον αλλά και τον άνθρωπο. [7]

Τα κυριότερα από αυτά είναι ότι η πανίδα και η χλωρίδα καταστρέφονται και κάποιες φορές η βλάβη αυτή είναι μη αναστρέψιμη, οδηγώντας έτσι στην εξαφάνιση πολλών ειδών, ζωικών και φυτικών, και στην πολυετή διάρκεια της προσπάθειας ανάκαμψης της φύσης για να επανέλθει στην αρχική της κατάσταση. Εν συνεχεία, λόγω της έλλειψης των δασικών περιοχών, υπάρχει μείωση στην παραγωγή οξυγόνου αλλά και αδυναμία συγκράτησης της βροχής, ειδικά όταν η φωτιά έχει κάψει δάση κοντά σε αστικό περιβάλλον όπως για παράδειγμα το 2017 στην Μάνδρα Αττικής, καθώς οι αιφνίδιες αστικές πλημμύρες έχουν άμεση σχέση με την απουσία δασικής έκτασης στην γύρω περιοχή ενώ συχνά στις αστικές περιοχές παρατηρούνται εμπρηστικές ενέργειες με σκοπό την εκμετάλλευση της γης αυτής.[8]

Η υπηρεσία της πυροσβεστικής, οι εθελοντικές ομάδες δασοπροστασίας και άλλοι οργανισμοί-φορείς ανά την Ελλάδα κάνουν το μέγιστο δυνατό για να αντιμετωπισθούν όλες οι δασικές και όχι μόνο πυρκαγιές. Όμως λόγω του περιορισμένου υλικοτεχνικού εξοπλισμού δεν είναι εφικτή η δυνατότητα άμεσης αναγνώρισης και έπειτα η κατάσβεση μιας μικρής πυρκαγιάς σε ένα ενδεχομένως δύσβατο σημείο ενός βουνού.

Μια δασική πυρκαγιά μπορεί να έχει τεραστία εξέλιξη μέσα σε δευτερόλεπτα λαμβάνοντας υπόψη τους παράγοντες που έχουν άμεση σχέση με τον ρυθμό εξάπλωσής της.

Αν θεωρήσουμε ότι η βλάστηση είναι ξηρή, όταν γίνει η έναρξη της πυρκαγιάς σε ένα απομακρυσμένο μέρος πχ 10 λεπτά μακριά από αστικό δρόμο μέσω δύσβατων δρόμων πάνω σε βουνό μπορεί από μια πυρκαγιά του 1τμ να εξαπλωθεί ανεξέλεγκτα ενώ αν υπάρχει και έντονος αέρας η πυρκαγιά θα μπορέσει να κάψει ολοσχερώς ολόκληρα εκτάρια.

Οπότε είναι πολύ σημαντικό να υπάρχει κάποιο επιπλέον μετρό ασφάλειας και πρόσληψης πυρκαγιών ώστε να μπορέσει να γίνει άμεση αναγνώριση τους, πριν λάβει μεγάλη έκταση η φωτιά.

Σκοπός της Διπλωματικής Εργασίας

Πάνω σε όσα προηγήθηκαν, ερχόμαστε και εστιάζουμε εμείς ώστε να μπορέσουμε να δημιουργήσουμε ένα σύστημα που θα μπορεί να είναι αυτόνομο, αξιόπιστο ώστε να προσφέρει άμεση ενημέρωση για την ύπαρξη πυρκαγιάς αλλά και άμεση κατάσβεση στην γύρω περιοχή.

Το σύστημά μας ονομάστηκε Remote Fire Tower(RFT), στα ελληνικά Ασύρματος Πύργος Πυρόσβεσης (ΑΠΠ), και βασίζεται σε μια ιδέα μετά από προσωπική επαφή με το αντικείμενο των δασικών πυρκαγιών.

Σαν μέλος της Εθελοντικής Ομάδας Δασοπροστασίας Πυρόσβεσης Δήμου Ελληνικού Αργυρούπολης ήρθα αντιμέτωπος με το πόσο εύκολο είναι να υπάρχει μια φωτιά σε ένα βουνό που έχουμε διπλά μας, είτε από αμέλεια των επισκεπτών είτε από φυσικά αίτια.

Λόγω αυτού σκέφτηκα ότι υπάρχει ένα κενό το οποίο θα μπορούσε έως ένα βαθμό να καλύψει η κατασκευή της Διπλωματικής εργασίας μου όταν αυτή αναπτυχθεί σε ρεαλιστική μορφή με την ανάλογη χρηματική χορηγία.

Σκοπός μου λοιπόν είναι η κατασκευή ενός ασύρματου συστήματος που με την χρήση απλής κάμερας και την χρήση Μηχανικής μάθησης να μπορέσει να εκπαιδευτεί και να αναγνωρίζει τα πρώτα σημάδια από μια πυρκαγιά ώστε να μπορεί να ειδοποιεί τόσο τις αρμόδιες αρχές για την άμεση αποστολή πυροσβεστικού προσωπικού αλλά και την άμεση αντιμετώπιση σε πρώτο χρόνο για να γίνει είτε ολική κατάσβεση είτε μερικώς περιορισμός της έως ότου έρθει το εξειδικευμένο προσωπικό με τον απαραίτητο εξοπλισμό να κάνει την τελική κατάσβεση.

Παρακάτω βλέπουμε το λογότυπο που δημιουργήθηκε δωρεάν με την χρήση του https://www.freelogodesign.org/ για να αντιπροσωπεύει την Διπλωματική μου.

Εικόνα 1 Υποθετικό Λογότυπο Εφαρμογής

Ιστορική Αναδρομή

Η πρώτη εμφάνιση των Νευρωνικών δικτύων έγινε το 1943, όταν ο νευροφυσιολόγος Warren McCulloch και ο μαθηματικός Walter Pitts έγραψαν μια εργασία για τους νευρώνες και τον τρόπο λειτουργίας τους. Αποφάσισαν να δημιουργήσουν ένα μοντέλο, χρησιμοποιώντας ένα ηλεκτρικό κύκλωμα και ως εκ τούτου γεννήθηκε το νευρικό δίκτυο.

Το 1950, ο Alan Turing δημιούργησε το παγκοσμίου φήμης Turing Test. Αυτή η δοκιμή είναι αρκετά απλή - για να περάσει το τεστ ένας υπολογιστής, πρέπει να είναι σε θέση να πείσει έναν άνθρωπο ότι είναι άνθρωπος και όχι υπολογιστής.

Το 1952 εμφανίστηκε το πρώτο πρόγραμμα υπολογιστή που μπορούσε να μάθει καθώς έτρεχε. Ήταν ένα παιχνίδι με πούλια, δημιουργήθηκε από τον Arthur Samuel.

Ο Frank Rosenblatt σχεδίασε το πρώτο τεχνητό νευρικό δίκτυο το 1958, που ονομάζεται Perceptron. Ο κύριος στόχος αυτού ήταν η αναγνώριση προτύπων και σχημάτων.

Ένα άλλο εξαιρετικά πρώιμο παράδειγμα ενός νευρικού δικτύου ήρθε το 1959, όταν ο Bernard Widrow και ο Marcian Hoff δημιούργησαν δύο μοντέλα στο Πανεπιστήμιο του Στάνφορντ. Το πρώτο ονομάστηκε ADELINE και μπορούσε να εντοπίσει δυαδικά μοτίβα. Για παράδειγμα, σε μια ροή δυαδικών ψηφίων, θα μπορούσε να προβλέψει τι θα είναι το επόμενο. Η επόμενη γενιά ονομάστηκε MADELINE και θα μπορούσε να εξαλείψει την ηχώ στις τηλεφωνικές γραμμές, έτσι είχε μια χρήσιμη εφαρμογή πέραν του πειράματος, στην καθημερινότητα. Χρησιμοποιείται ακόμα και σήμερα.

Παρά την επιτυχία του MADELINE, δεν σημειώθηκε μεγάλη πρόοδος μέχρι τα τέλη της δεκαετίας του 1970 για πολλούς λόγους, κυρίως λόγω της δημοτικότητας της αρχιτεκτονικής Von Neumann. Πρόκειται για μια αρχιτεκτονική όπου οι οδηγίες και τα δεδομένα αποθηκεύονται στην ίδια μνήμη, η οποία είναι αναμφισβήτητα πιο απλή στην κατανόηση από ένα νευρωνικό δίκτυο, με την πλειοψηφία των προγραμμάτων που δημιουργήθηκαν μετέπειτα να βασίζονται σε αυτό.

Το 1982 ήταν η χρονιά στην οποία το ενδιαφέρον για τα νευρικά δίκτυα άρχισε να αυξάνεται και πάλι, όταν ο Τζον Χόπφιλντ πρότεινε τη δημιουργία ενός δικτύου που είχε αμφίδρομες γραμμές, παρόμοιο με τον τρόπο που λειτουργούν πραγματικά οι νευρώνες. Επιπλέον, το 1982, η Ιαπωνία ανακοίνωσε ότι επικεντρώνεται σε πιο προηγμένα νευρωνικά δίκτυα, τα οποία κίνησαν την αμερικανική χρηματοδότηση και έρευνα στην περιοχή.

Τα νευρωνικά δίκτυα χρησιμοποιούν πολλαπλασιασμό στο παρασκήνιο και αυτό το σημαντικό βήμα ήρθε το 1986, όταν τρεις ερευνητές από το τμήμα ψυχολογίας του Στάνφορντ αποφάσισαν να επεκτείνουν έναν αλγόριθμο που δημιουργήθηκε από τους Widrow και Hoff το 1962. Επομένως, αυτό επέτρεψε την δημιουργία πολλαπλών στρωμάτων που θα χρησιμοποιηθούν σε ένα νευρικό δίκτυο, οδηγώντας σε αυτό που είναι γνωστό ως «αργοί μαθητές», το οποίο μαθαίνει για μεγάλο χρονικό διάστημα. [12]

Τα τέλη της δεκαετίας του 1980 και του 1990 δεν έφεραν πολλά στο πεδίο των νευρωνικών δικτύων. Ωστόσο, το 1997, ο υπολογιστής IBM Deep Blue, που ήταν υπολογιστής σκακιού, κέρδισε τον παγκόσμιο πρωταθλητή σκακιού.

Έκτοτε, σημειώθηκαν πολλές ακόμη πρόοδοι στον τομέα, όπως το 1998, όταν η έρευνα στα AT&T Bell Laboratories σχετικά με την αναγνώριση ψηφίων είχε ως αποτέλεσμα την καλή ακρίβεια στον εντοπισμό χειρόγραφων ταχυδρομικών κωδικών από την Ταχυδρομική Υπηρεσία των ΗΠΑ.

Από τις αρχές του 21ου αιώνα, πολλές επιχειρήσεις είχαν συνειδητοποιήσει ότι η μηχανική μάθηση θα αυξήσει το δυναμικό υπολογισμό. Αυτός είναι και ο λόγος για τον οποίο ερευνάται πιο έντονα, προκειμένου να μείνει μπροστά από τον ανταγωνισμό.

Μερικά μεγάλα έργα περιλαμβάνουν:

GoogleBrain (2012) - Αυτό ήταν ένα βαθύ νευρωνικό δίκτυο που δημιουργήθηκε από τον Jeff Dean της Google, το οποίο επικεντρώθηκε στην ανίχνευση μοτίβων σε εικόνες και βίντεο. Κατάφερε να χρησιμοποιήσει τους πόρους της Google, γεγονός που το έκανε απαράμιλλο με πολύ μικρότερα νευρωνικά δίκτυα. Χρησιμοποιήθηκε αργότερα για τον εντοπισμό αντικειμένων στα βίντεο του YouTube.

AlexNet (2012) - Η AlexNet κέρδισε τον διαγωνισμό ImageNet με μεγάλο περιθώριο το 2012, γεγονός που οδήγησε στη χρήση GPU και Convolutional Neural Networks στη μηχανική μάθηση. Δημιούργησαν επίσης το ReLU, το οποίο είναι μια λειτουργία ενεργοποίησης που βελτιώνει σημαντικά την αποδοτικότητα των CNN.

DeepFace (2014) - Αυτό είναι ένα Deep Neural Network που δημιουργήθηκε από το Facebook, το οποίο ισχυρίστηκε ότι μπορεί να αναγνωρίσει άτομα με την ίδια ακρίβεια ενός ανθρώπινου ματιού.

DeepMind (2014) - Αυτή η εταιρεία αγοράστηκε από την Google και μπορεί να παίξει βασικά βιντεοπαιχνίδια στα ίδια επίπεδα με τους ανθρώπους. Το 2016, κατάφερε να νικήσει έναν επαγγελματία στο παιχνίδι Go, το οποίο θεωρείται ένα από τα πιο δύσκολα επιτραπέζια παιχνίδια στον κόσμο.

OpenAI (2015) - Πρόκειται για έναν μη κερδοσκοπικό οργανισμό που δημιουργήθηκε από τον Elon Musk και άλλους, για τη δημιουργία ασφαλούς τεχνητής νοημοσύνης που μπορεί να ωφελήσει την ανθρωπότητα.

Amazon Machine Learning Platform (2015) - Αυτό είναι μέρος των Amazon Web Services και δείχνει πώς οι περισσότερες μεγάλες εταιρείες θέλουν να συμμετάσχουν στη μηχανική μάθηση. Προσδιορίζεται πως οδηγεί πολλά από τα εσωτερικά τους συστήματα, από υπηρεσίες που χρησιμοποιούνται τακτικά, όπως προτάσεις αναζήτησης και η Alexa, καθως και πιο πειραματικές όπως η Prime Air και η Amazon Go.

U-net (2015) - Πρόκειται για μια αρχιτεκτονική CNN που ειδικεύεται στην τμηματοποίηση της βιοϊατρικής εικόνας. Εισήγαγε ίση ποσότητα επιπέδων δειγματοληψίας καθώς και παράλειψη συνδέσεων. [12]

Η σημασία των GPU

Η Nvidia βρίσκεται πίσω από ένα από τα μεγαλύτερα συνέδρια για την τεχνητή νοημοσύνη και αυτό είναι για καλό λόγο - οι GPU είναι εξαιρετικά σημαντικές στον κόσμο της μηχανικής μάθησης. Οι GPU έχουν περίπου 200 φορές περισσότερους επεξεργαστές ανά τσιπ από τις CPU. Η άλλη πλευρά αυτού του γεγονότος, ωστόσο, είναι ότι ενώ οι CPU μπορούν να εκτελέσουν οποιοδήποτε είδος υπολογισμού, οι GPU είναι προσαρμοσμένες μόνο σε συγκεκριμένες περιπτώσεις χρήσης, όπου λειτουργίες (προσθήκη, πολλαπλασιασμός κ.λπ.) πρέπει να εκτελούνται σε διανύσματα, τα οποία είναι ουσιαστικά λίστες αριθμών.

Μια CPU θα εκτελούσε κάθε λειτουργία σε κάθε αριθμό στο διάνυσμα σειριακά, δηλαδή μία προς μία, δίδοντας επομένως πιο αργά τα αποτελέσματα των λειτουργιών. Μια GPU θα εκτελούσε λειτουργίες σε κάθε αριθμό στο διάνυσμα παράλληλα, δηλαδή ταυτόχρονα.

Τα διανύσματα και οι πίνακες, τα οποία είναι πλέγματα αριθμών (ή λίστες διανυσμάτων) είναι απαραίτητα για τις εφαρμογές μηχανικής εκμάθησης, και λόγω αυτού, είναι μικρότερα, επομένως μπορούν να χωρέσουν περισσότερα σε ένα τσιπ. Η Nvidia έκανε την πρώτη GPU στον κόσμο, το GeForce 256 το 1999. Εκείνη την εποχή, η κυκλοφορία του προϊόντος δημιουργούσε ενδοιασμούς καθώς ήταν ένα εντελώς νέο είδος προϊόντος.

Ωστόσο, λόγω της χρήσης υπολογιστικών διανυσμάτων σε βιντεοπαιχνίδια, οι GPU πολλαπλασιάστηκαν, καθότι τα βιντεοπαιχνίδια επωφελήθηκαν με ένα τεράστιο άλμα στην απόδοση. Χρόνια αργότερα, οι μαθηματικοί, οι επιστήμονες και οι μηχανικοί συνειδητοποίησαν ότι οι GPU θα μπορούσαν να χρησιμοποιηθούν για τη βελτίωση της ταχύτητας των υπολογισμών που χρησιμοποιούνται στους υπολογισμούς τους, λόγω της χρήσης διανυσμάτων.

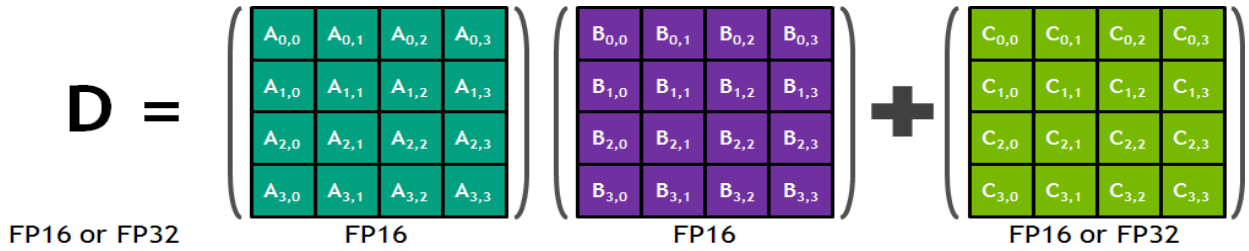
Αυτό οδήγησε τις εταιρείες GPU, ιδιαίτερα την Nvidia, να επωφεληθούν σε μεγάλο βαθμό από την «επανάσταση της μηχανικής μάθησης». Η τιμή της μετοχής της Nvidia αυξήθηκε περίπου 18 φορές από το 2012, έτος κατά την οποία η AlexNet έδειξε τη σημασία των GPU στη μηχανική μάθηση.

Nvidia Tensor Cores - 2017

Η Nvidia χρησιμοποιείται από την Amazon για να τροφοδοτήσει την πλατφόρμα Amazon Web Service Machine Learning. Αυτό οφείλεται στο γεγονός ότι δημιουργούν GPU ειδικά για μηχανική εκμάθηση, για παράδειγμα το Tesla V100, που ανακοινώθηκε τον Μάιο του 2017. Αυτό χρησιμοποίησε το Tensor Cores, το οποίο χρησιμοποιείται για την αριθμητική μήτρα στη μηχανική μάθηση. [12] [15]

Ένας πυρήνας τανυστή μπορεί να υπολογίσει 64 λειτουργίες σταθερού σημείου ανά κύκλο ρολογιού, καθώς παρέχει μια σειρά επεξεργασίας μήτρας 4x4x4 και εκτελεί τη λειτουργία που φαίνεται στην παρακάτω εικόνα. Τα A, B, C και D είναι πίνακες 4x4.

Αυτό σημαίνει ότι πολλές λειτουργίες μπορούν να υποβληθούν σε επεξεργασία σε έναν κύκλο ρολογιού, ο οποίος είναι πολύ πιο αποτελεσματικός από έναν επεξεργαστή και ακόμη περισσότερο από μια μη βελτιστοποιημένη GPU.



Εικόνα 2 Πώς πολλαπλασιάζονται οι πυρήνες του τανυστή

Google Tensor Processing Unit (TPU) - 2016

Τα TPU τροφοδοτούν πολλές από τις κύριες υπηρεσίες της Google, όπως Αναζήτηση, Street View, Φωτογραφίες και Μετάφραση. Επιτρέπουν στα νευρωνικά δίκτυα πίσω από αυτές τις υπηρεσίες να λειτουργούν γρηγορότερα και έτσι λειτουργούν με πιο προσιτό κόστος.

Παρόμοια με τους πυρήνες Tensor, βελτιστοποιεί τον τρόπο με τον οποίο οι λειτουργίες πολλαπλασιασμού, πρόσθεσης και ενεργοποίησης εφαρμόζονται σε δεδομένα στα CNN, καθιστώντας τη διαδικασία πολύ πιο γρήγορη. Σε αντίθεση με τους πυρήνες του τανυστή, οι οποίοι αποτελούν μέρος γενικής χρήσης GPU, τα TPU είναι τσιπ σχεδιασμένα αποκλειστικά για την επιτάχυνση υπολογισμών που απαιτούνται για νευρωνικά δίκτυα.

Αυτό τους καθιστά ακόμη πιο γρήγορους από τους GPU γενικού σκοπού κατά την εκτέλεση εργασιών μηχανικής εκμάθησης, καθώς οι GPU πρέπει να χειρίζονται άλλες περιπτώσεις χρήσης και έτσι είναι λιγότερο εξειδικευμένες. [12]

Intel - Nervana Neural Processor - 2017

Αφού οι GPU επέτρεψαν να εξελιχθεί η μηχανική μάθηση, η Intel, ο μεγαλύτερος κατασκευαστής CPU στον κόσμο, έμεινε στάσιμος σε αυτόν τομέα. Το πιο κοντινό πράγμα σε GPU που παρήγαγε η Intel ήταν ενσωματωμένες GPU, δηλαδή GPU που ενσωματώθηκαν στις CPU. Ωστόσο, αυτά δεν έχουν συγκρίσιμη απόδοση, καθώς έπρεπε να είναι πολύ μικρά για να χωρέσουν στην CPU.

Η τιμή της μετοχής της Intel δεν έχει αυξηθεί ιδιαίτερα, κοντά στο ρυθμό που έχει η Nvidia τα τελευταία χρόνια. Ωστόσο, η Intel εργάζεται πάνω σε μια απάντηση συνειδητοποιώντας πως η ιδέα μιας ενιαίας CPU που μπορεί να κάνει όλες τις εργασίες που απαιτούνται σε έναν υπολογιστή δεν είναι ρεαλιστική, ειδικά λόγω της επικείμενης κατάρρευσης του νόμου του Moore, που ανέφερε τον αριθμό των τρανζίστορ σε ένα μια CPU πως θα διπλασιαζόταν κάθε 18-24 μήνες, δηλαδή η CPU θα διπλασιαζόταν με ταχύτητα κάθε 2 χρόνια. Μέρος της απάντησής τους είναι ένα προσαρμοσμένο τσιπ αφιερωμένο στην επιτάχυνση των υπολογισμών που εκτελούνται σε νευρωνικά δίκτυα, που ονομάζεται Nervana Neural Processor. Αυτό είναι παρόμοιο με το TPU της Google. Οι πολλαπλασιασμοί και συνελεύσεις Matrix είναι οι δύο βασικές λειτουργίες που εκτελούνται από τον επεξεργαστή. [12]

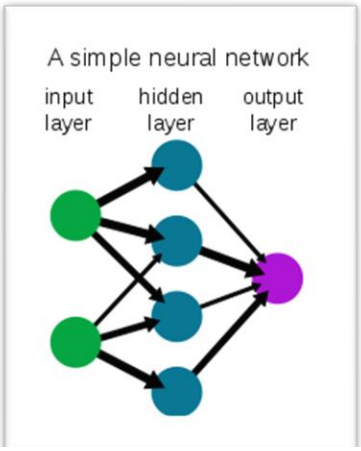
Τι είναι Νευρωνικό δίκτυο?

Παρακάτω θα αναφέρουμε σε απλοϊκή μορφή το τι είναι το Νευρωνικό δίκτυο και πως λειτουργεί ώστε να μπορέσουμε να καταλάβουμε χωρίς μαθηματικά και χωρίς ιδιαίτερες γνώσης την λογική που έγινε επιλογή των Νευρωνικών δικτύων για την υλοποίηση του μοντέλου αναγνώρισης εικόνας στο κομμάτι ανίχνευσης φωτιάς στα πλαίσια της διπλωματικής εργασίας.

Το Νευρωνικό δίκτυο είναι ένα σύνολο από Νευρώνες όπου ενώνονται μεταξύ τους και χρησιμοποιούνται ώστε να γίνει επίλυση κάποιου προβλήματος. Με τον όρο Νευρώνες εννοούμε μια οντότητα λογισμικού στην οποία υπάρχει ένας απλός κόμβος με υπολογιστική δυνατότητα.

Όλα τα στοιχεία που εμείς εισάγουμε και λαμβάνουμε στο τέλος σαν έξοδος από ένα Νευρωνικό δίκτυο είναι της μορφής αριθμών ώστε να μπορούν να γίνουν τα στοιχεία κατανοητά από τον ηλεκτρονικό υπολογιστή και να μπορούν να γίνουν συνθέτες μαθηματικές πράξεις για να λάβουμε το αποτέλεσμα που χρειαζόμαστε ώστε να κάνουμε την πρόβλεψη. Όλα αυτά βέβαια στην δική μας περίπτωση γίνονται στο παρασκήνιο χωρίς να χρειαστεί σε πρώτο χρόνο να γράψουμε και να λύσουμε εμείς μαθηματικές εξισώσεις για να δούμε αν το σήμα είναι φωτιά ή όχι, με τα μοντέλα που θα χρησιμοποιήσουμε να έχουν δημιουργηθεί μετά από χρονιά μελέτης και ερευνάς για να φτάσουν πλέον στα χεριά μας σαν απλά εργαλεία με την μορφή κώδικα. [9]

Οι νευρώνες χωρίζονται σε 3 τύπους: εισόδου , κρυμμένου επίπεδου και εξόδου.



Εικόνα 3 Παράδειγμα τεχνητού νευρωνικού δικτύου

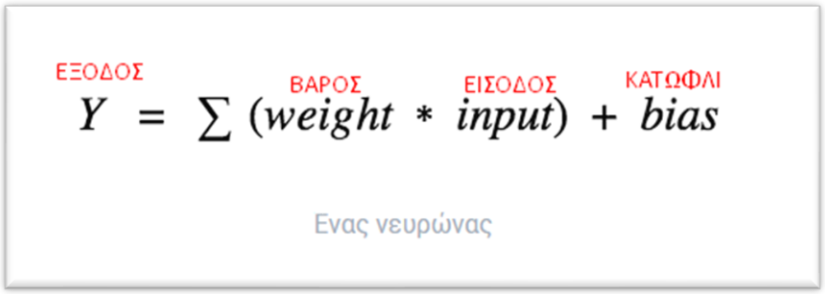
Οι νευρώνες εισόδου(input layer) δεν εκτελούν κανένα είδους υπολογισμού , είναι απλά το σημείο που εμείς δίνουμε στο Νευρωνικό δίκτυο τα δεδομένα του ώστε να μπορέσει να κάνει μια πρόβλεψη.

Οι νευρώνες του κρυμμένου επιπέδου παίρνουν τα στοιχεία της εισόδου και τα πολλαπλασιάζουν με το συναπτικό βάρος του κάθε νευρώνα και υπολογίζουν το άθροισμα των γινομένων.

Το άθροισμα αυτό μετά προωθείται προς την έξοδο αφού πρώτα περάσει από την συνάρτηση ενεργοποίησης, η οποία βρίσκεται μέσα σε κάθε ένα νευρώνα ξεχωριστά, κάθε ένα αποτέλεσμα είναι συγκεκριμένο ανάλογα τα βάρη και την είσοδο κάθε περίπτωσης. [9]

Τέλος οι νευρώνες εξόδου μαζεύουν το τελικό αποτέλεσμα μετά από όλους τους νευρώνες του κρυφού επιπέδου και το προωθούν στην έξοδο του συστήματος.

Τα βάρη(weight) και το κατώφλι(bias) συνήθως αναφέρονται ως *w και b* και είναι οι μαθησιακές παράμετροι ενός μοντέλου μηχανικής μάθησης.



Εικόνα 4 Υπολογισμός εξόδου ενός νευρώνα

Όταν οι είσοδοι μεταδίδονται μεταξύ νευρώνων, τα βάρη εφαρμόζονται στις εισόδους μαζί με το κατώφλι, τα βάρη ελέγχουν το σήμα (ή την ισχύ της σύνδεσης) μεταξύ δύο νευρώνων. Με άλλα λόγια, ένα βάρος αποφασίζει πόση επιρροή θα έχει η είσοδος στην έξοδο.

Το κατώφλι, που έχει σταθερές τιμές, είναι μια επιπλέον είσοδος στο επόμενο επίπεδο που θα έχει πάντα την τιμή 1. Το κατώφλι δεν επηρεάζεται από το προηγούμενο αλλά έχει τα δικά του βάρη και τα προσθέτει στο σύνολο της εξόδου από τον κάθε νευρώνα.

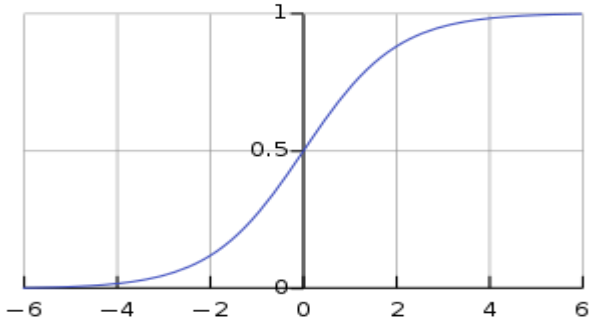
Εάν το τελικό άθροισμα από το σύνολο των εισόδων του νευρώνα είναι μεγαλύτερο από την τιμή του κατωφλιού, τότε ο νευρώνας ενεργοποιείται. Εάν είναι μικρότερο, τότε ο νευρώνας παραμένει ανενεργός.

Υπάρχουν διάφορες συναρτήσεις ενεργοποίησης που θα τις αναφέρουμε ονοματικά: βηματική συνάρτηση, γραμμική συνάρτηση και μη γραμμική συνάρτηση. Γενικά στα νευρωνικά δίκτυα επιλέγουμε την μη γραμμική συνάρτηση που καλείται επίσης και σιγμοειδής συνάρτηση η οποία συνάρτηση έχει μορφή τύπου «S» , με τον μαθηματικό της τύπο και την απεικόνιση της να εμφανίζεται παρακάτω.[9]

$$S(t) = \frac{1}{1 + e^{-t}}$$

Εικόνα 5

Μη γραμμική συνάρτηση ενεργοποίησης



Εικόνα 6 Γραφική Απεικόνιση Μη γραμμικής Συνάρτησης

Μια από τις βασικότερες χρήσεις των Νευρωνικών δικτύων είναι αυτή της μάθησης. Ως μάθηση αναφερόμαστε στην δυνατότητα να μπορεί σταδιακά να βελτιώνετε το δίκτυό μας ώστε να μπορέσει να προβλέψει ένα αποτέλεσμα ή να λύσει ένα πρόβλημα που του έχουμε παρουσιάσει.

Για να μπορέσει να μάθει να δίνει σωστές λύσεις στα προβλήματα που του παρουσιάζουμε πρέπει να του μάθουμε πρώτα το τι είναι σωστό και τι είναι λάθος, όπως για παράδειγμα στην δική μας προσέγγιση κάνουμε εκπαίδευση στο δίκτυο μας αναφέροντάς του τι είναι και τι δεν είναι φωτιά ή καπνός. Η εκπαίδευση αυτή γίνεται με το πέρασμα των δεδομένων εισόδου από το δίκτυό μας ώστε να μπορέσει να γίνει παραμετροποίηση των βαρών και του κατωφλίου ώστε να έρθει όσο πιο κοντά γίνεται στην σωστή πρόβλεψη. Αυτή η διαδικασία πραγματοποιείται πολλές φορές να γίνει βελτίωση των βαρών και των τιμών του κατωφλίου.

Αφού ένα Νευρωνικό δίκτυο έχει εκπαιδευτεί σε σημείο που μας λύνει το πρόβλημα που του παρουσιάζουμε μπορούμε να σταματήσουμε την εκπαίδευση αυτή ώστε να παραμείνουν σταθερά από εδώ και πέρα τα βάρη και οι τιμές από το κατώφλι ώστε στη συνέχεια να γίνεται χρήση του δικτύου μόνο και μόνο για προβλέψεις. Το σημαντικό και αυτό που πρέπει να προσέξουμε σε ένα τέτοιο δίκτυο είναι το να μην υπερπροσαρμοστεί στα συγκεκριμένα δεδομένα που του έχουμε δώσει κατά την εκπαίδευση και να προβεί σε λανθασμένες προβλέψεις σε αλλά δεδομένα της ιδίας κατηγορίας με αυτής που έχει εκπαιδευτεί, αποτέλεσμα που ονομάζεται γενίκευση. [9]

Η εκπαίδευση των Νευρωνικών δικτύων

Όταν θέλουμε να εκπαιδεύσουμε ένα Νευρωνικό δίκτυο έχουμε δυο βασικές επιλογές : τη μάθηση με επίβλεψη (supervised learning) και τη μάθηση χωρίς επίβλεψη (unsupervised learning).

- Η μάθηση με επίβλεψη είναι μια διαδικασία συνδυασμού εξωτερικών εκπαιδευτών με γενικές ή ευρείες πληροφορίες. Ορισμένες μέθοδοι που εμπίπτουν σε αυτήν την κατηγορία είναι η μάθηση διόρθωσης σφαλμάτων και η στοχαστική μάθηση. Παραδείγματα αντιπροσωπευτικής εποπτευόμενης μάθησης περιλαμβάνουν αποφάσεις σχετικά με το πότε πρέπει να σταματήσει η διαδικασία εκπαίδευσης, αποφάσεις σχετικά με τη συχνότητα της παρουσίασης προτύπων εκπαίδευσης στο δίκτυο και αποφάσεις για παρουσιάσεις προόδου του δικτύου. Η εποπτευόμενη μάθηση χωρίζεται σε δύο κατηγορίες: δομική μάθηση και προσωρινή μάθηση. Ο πρώτος τύπος αλγορίθμου χρησιμοποιείται για την εύρεση της καλύτερης σχέσης μεταξύ της εισόδου και της εξόδου κάθε ζεύγους προτύπων. Παραδείγματα δομικής μάθησης είναι η αναγνώριση προτύπων και η ταξινόμηση αυτών ενώ, παραδείγματα προσωρινής εκμάθησης είναι η πρόβλεψη και ο έλεγχος.
- Η μάθηση χωρίς επίβλεψη είναι ο μαθησιακός αλγόριθμος που ονομάζεται και αυτο-οργάνωτος και είναι ένα πρόγραμμα που δεν απαιτεί την παρουσία ενός «εξωτερικού» δασκάλου ή εκπαιδευτή. Στην πραγματικότητα, βασίζονται μόνο σε τοπικές πληροφορίες καθ’ 'όλη τη διάρκεια της εκπαιδευτικής διαδικασίας του τεχνητού νευρικού δικτύου. Αυτοί οι αλγόριθμοι οργανώνουν δεδομένα και ανακαλύπτουν σημαντικά συλλογικά χαρακτηριστικά. Για παράδειγμα, αλγόριθμοι εκπαίδευσης χωρίς επίβλεψη είναι αλγόριθμος Hebbian, διαφορικός αλγόριθμος Hebbian και αλγόριθμος Min-Max. [9]

Το μεγαλύτερο μέρος της εκπαιδευτικής διαδικασίας είναι εκτός σύνδεσης. Όταν χρησιμοποιείται ολόκληρο το πρότυπο δείγματος για να τροποποιηθούν οι τιμές βάρους, πριν τελικά χρησιμοποιηθεί το δίκτυο ως εφαρμογή, αυτό ονομάζεται εκπαίδευση εκτός σύνδεσης. Οι αλγόριθμοι εκπαίδευσης εκτός σύνδεσης απαιτούν να περιλαμβάνονται όλα τα πρότυπα στην εκπαίδευση δικτύου. Αυτό απέκλεισε τη δυνατότητα εισαγωγής νέων πληροφοριών μέσω νέων προτύπων. Φυσικά υπάρχουν και Τεχνητά Νευρωνικά Δίκτυα που δεν αποκλείουν τη δυνατότητα εισαγωγής νέων πληροφοριών μετά την τελική μοντελοποίηση. Εάν πρέπει να εισαχθεί ένα νέο πρότυπο στο δίκτυο, μπορεί να το γίνει απευθείας χωρίς να χαθούν οι αρχικές πληροφορίες. Τα πλεονεκτήματα των δικτύων που χρησιμοποιούν διαδικασίες κατάρτισης εκτός σύνδεσης επικεντρώνονται κυρίως στην ικανότητα παροχής καλύτερων λύσεων σε δύσκολα προβλήματα.[9]

Ένα ακόμα χαρακτηριστικό για να διαχωρίσουμε τους τύπους μοντέλων Μηχανικής μάθησης είναι βάση του αποτελέσματος που δίνεται σε κάθε μια περίπτωση. Τα μοντέλα που υπάρχουν είναι τα παρακάτω.

- Η ταξινόμηση (classification) είναι όταν τα δεδομένα εισόδου και εξόδου κατά συνεπεία είναι διακριτά στοιχεία, όπως στο δικό μας σενάριο διότι έχουμε 2 κατηγορίες (classes) το καπνός και όχι καπνός. Γενικά οποιοδήποτε μοντέλο προσπαθεί να προβλέψει αν κάτι ανήκει σε κάποια σταθερή κατηγορία και όχι μια τιμή είτε αυτό είναι μεταξύ δυο πραγμάτων ή ακόμα και χιλίων(1000) κατηγοριών. Η χρήση της μεθόδου ταξινόμησης είναι συχνά με επιβλεπομένη μάθηση. Άλλο ένα παράδειγμα εφαρμογής της ταξινόμησης είναι τα Spam Filters που έχουν πλέον όλες οι υπηρεσίες ηλεκτρονικού ταχυδρομείου και με βάση την εκπαίδευση που έχουν λάβει κάνουν μια πρόβλεψη αν κάποιο e-mail είναι Spam(ανεπιθύμητη αλληλογραφία) ή όχι .

- Η παλινδρόμηση είναι μια ευρέως χρησιμοποιούμενη τεχνική στατιστικής μοντελοποίησης για τη μελέτη της συσχέτισης μεταξύ μιας εξαρτημένης μεταβλητής και μιας ή περισσότερων ανεξάρτητων μεταβλητών. Για παράδειγμα το να μπορέσει να προβλέψει μια τιμή για τη θερμοκρασία της αυριανής μέρας με βάση δεδομένα από τις προηγούμενες ημέρες, είτε αυτό είναι μόνο η θερμοκρασία του προηγούμενου 20ημερου ή ακόμα και πιο πολλών μεταβλητών όπως για παράδειγμα υγρασία , ένταση ανέμου, εμφάνιση βροχής ή όχι και πολλά αλλά.

- Η Συσταδοποίηση είναι η διαδικασία διαίρεσης ενός συνόλου "αντικειμένων" σε ένα σύνολο λογικών ομάδων. Η εισαγωγή αντικειμένων στην ίδια ομάδα λέγετε ομοιότητα αντικειμένων και αντίστροφα (τα αντικείμενα που ανήκουν σε διαφορετικές ομάδες είναι διαφορετικά). Το αν τα αντικείμενα είναι παρόμοια ή όχι εξαρτάται βασικά από τη συγκεκριμένη ερώτηση και τη μορφή του "αντικειμένου". Σε αντίθεση με την ταξινόμηση, οι ομάδες δεν είναι γνωστές εκ των προτέρων, καθιστώντας αυτόν τον διαχωρισμό τυπική εργασία μη επιτηρούμενης μάθησης. [11]

Εφαρμογές Νευρωνικών δικτύων

Τα νευρωνικά δίκτυα είναι κατάλληλα για σχεδόν οποιαδήποτε κατάσταση όπου υπάρχει σχέση μεταξύ μεταβλητών πρόβλεψης (ανεξάρτητες μεταβλητές, μεταβλητές εισόδου, μεταβλητές εξόδου) ακόμη κι αν η σχέση είναι πολύ περίπλοκη για να εκφραστεί από τη συνήθη συσχέτιση.

Τυπικά αντιπροσωπευτικά παραδείγματα προβλημάτων που έχουν εφαρμοστεί με επιτυχία στην ανάλυση Νευρωνικών δικτύων είναι τα εξής: [9]

- Ιατρική διάγνωση: Μπορεί να παρακολουθεί μια ποικιλία ενδείξεων που σχετίζονται με την ιατρική, όπως ο καρδιακός ρυθμός, το περιεχόμενο διαφόρων ουσιών στο αίμα και ο συνδυασμός αναπνευστικών ρυθμών. Η απόδοση μιας συγκεκριμένης ιατρικής κατάστασης μπορεί να σχετίζεται με έναν πολύπλοκο συνδυασμό αλλαγών σε ένα υποσύνολο ελεγχόμενων μεταβλητών. Τα νευρωνικά δίκτυα έχουν χρησιμοποιηθεί για τον προσδιορισμό αυτού του προγνωστικού μοτίβου για την παροχή κατάλληλης θεραπείας.

- Πρόβλεψη χρηματιστηρίου: Η αστάθεια των τιμών των μετοχών και των χρηματιστηριακών δεικτών είναι ένα άλλο παράδειγμα ενός πολύπλοκου πολυδιάστατου φαινομένου, αλλά σε ορισμένες περιπτώσεις είναι ένα εν μέρει ντετερμινιστικό φαινόμενο. Πολλοί τεχνικοί αναλυτές χρησιμοποιούν νευρωνικά δίκτυα για να κάνουν προβλέψεις τιμών μετοχών με βάση πολλούς παράγοντες (για παράδειγμα, την προηγούμενη απόδοση άλλων αποθεμάτων και διάφορους οικονομικούς δείκτες) [9]

- Παρακολούθηση της κατάστασης των μηχανών: Τα νευρικά δίκτυα μπορούν να βοηθήσουν στη μείωση του κόστους παρέχοντας πρόσθετη τεχνογνωσία για την οργάνωση της προληπτικής συντήρησης των μηχανών. Το νευρικό δίκτυο μπορεί στη συνέχεια να εκπαιδευτεί για να το διακρίνει από τον ήχο που παράγεται από το μηχάνημα, ανεξάρτητα από το αν το μηχάνημα λειτουργεί κανονικά ή βρίσκεται στα πρόθυρα τυχόν βλάβης. Μετά από αυτήν την περίοδο εκπαίδευσης, η εμπειρία του ίδιου δικτύου μπορεί να χρησιμοποιηθεί για να προειδοποιήσει τους τεχνικούς για επικείμενες αποτυχίες και μπορεί να προκαλέσει δαπανηρές και απρόβλεπτες χρονικές καθυστερήσεις. [9]

Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks – CNN)

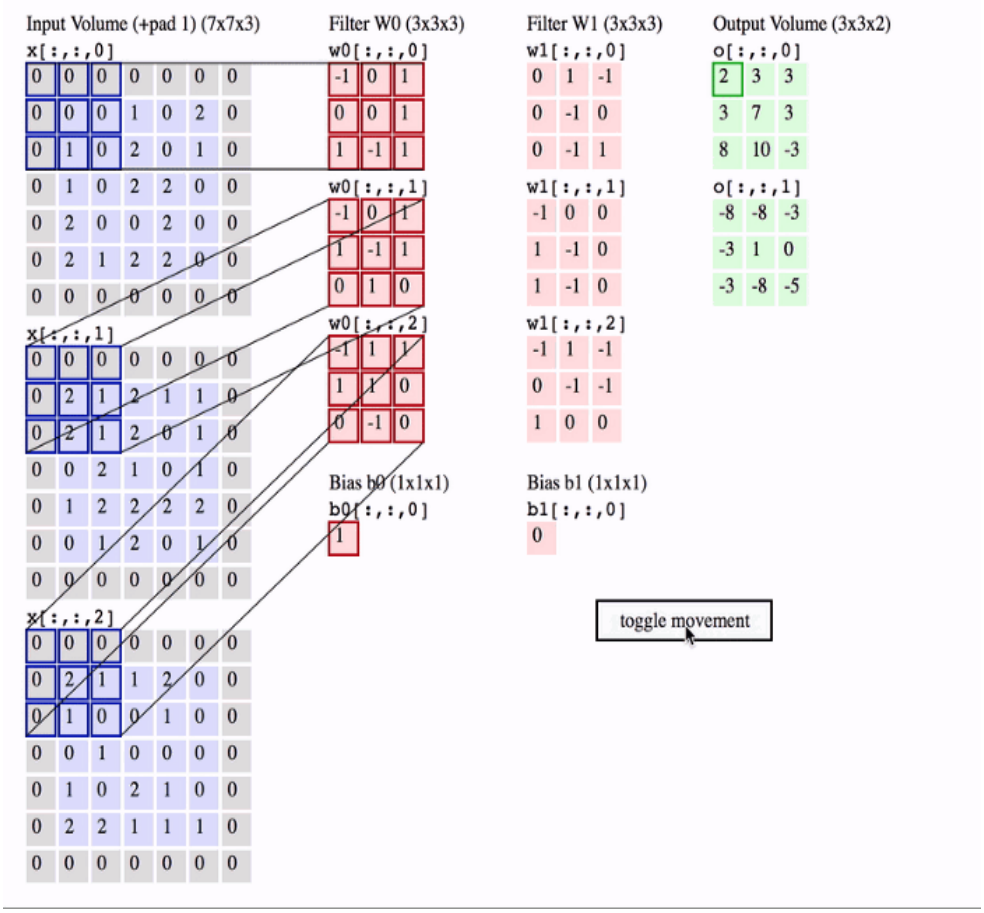
Ένα από τα κύρια μέρη των Νευρωνικών Δικτύων είναι τα Συνελικτικά νευρωνικά δίκτυα (CNN). Τα CNN χρησιμοποιούν αναγνώριση εικόνας και ταξινόμηση προκειμένου να ανιχνεύσουν αντικείμενα, να αναγνωρίσουν πρόσωπα κ.λπ. Αποτελούνται από νευρώνες με βάρη και κατώφλι όπως και τα απλά Νευρωνικά δίκτυα.

Τα CNN χρησιμοποιούνται κυρίως για την ταξινόμηση των εικόνων, την ομαδοποίησή τους με ομοιότητες και, στη συνέχεια, την αναγνώριση αντικειμένων. Πολλοί αλγόριθμοι που χρησιμοποιούν CNN μπορούν να αναγνωρίσουν πρόσωπα, πινακίδες, ζώα κ.λπ. [10]

Πώς λειτουργούν όμως τα CNN, σε αντίθεση με τις επίπεδες εικόνες που οι άνθρωποι μπορούν να δουν ότι έχουν μόνο πλάτος και ύψος, τα CNN δεν μπορούν να το αναγνωρίσουν. Λόγω ψηφιακών έγχρωμων εικόνων με κωδικοποίηση κόκκινου-μπλε-πράσινου (RGB), τα CNN

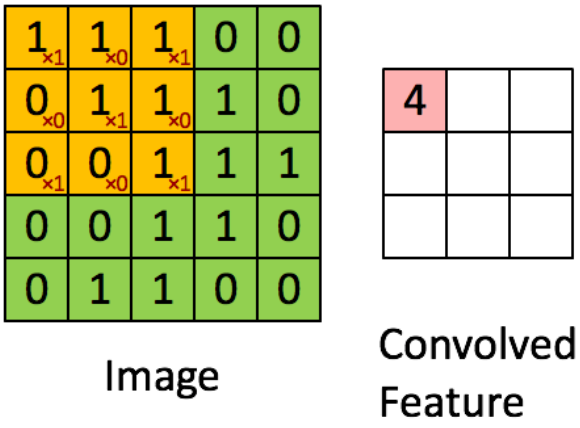
συνδυάζουν αυτά τα τρία χρώματα για να παράγουν το φάσμα χρωμάτων που αντιλαμβάνονται οι άνθρωποι.

Ένα συνελικτικό δίκτυο απορροφά εικόνες όπως τρία ξεχωριστά στρώματα χρώματος που στοιβάζονται το ένα πάνω από το άλλο. Μια κανονική έγχρωμη εικόνα θεωρείται ως ορθογώνιο κουτί του οποίου το πλάτος και το ύψος μετρούνται από τον αριθμό των pixel από αυτές τις διαστάσεις. Τα επίπεδα βάθους στα τρία στρώματα χρωμάτων (RGB) που ερμηνεύονται από τα CNN αναφέρονται ως κανάλια. [10]



Εικόνα 7 Συνελικτικό δίκτυο με 3 στρωματά (RGB)

Το πρώτο επίπεδο σε ένα δίκτυο CNN είναι το Συνελικτικό Επίπεδο (Convolutional Layer), το οποίο είναι το βασικό δομικό στοιχείο και κάνει το μεγαλύτερο μέρος της υπολογιστικής ανάγκης. Τα δεδομένα ή οι εικόνες συσπειρώνονται χρησιμοποιώντας φίλτρα ή πυρήνες. Τα φίλτρα είναι μικρές μονάδες που εφαρμόζουμε στα δεδομένα μέσω ενός συρόμενου παραθύρου. Το βάθος της εικόνας είναι το ίδιο με την είσοδο, για μια έγχρωμη εικόνα που η τιμή βάθους RGB είναι 4, θα εφαρμοζόταν επίσης ένα φίλτρο βάθους 4. Αυτή η διαδικασία περιλαμβάνει τη λήψη του στοιχείου των φίλτρων στην εικόνα και αθροίζοντας τις συγκεκριμένες τιμές για κάθε συρόμενη ενέργεια. Η έξοδος μιας συνέλιξης που έχει ένα 3d φίλτρο με χρώμα θα ήταν μια 2η μήτρα. [14]



Εικόνα 8 Συνελικτικό Επίπεδο

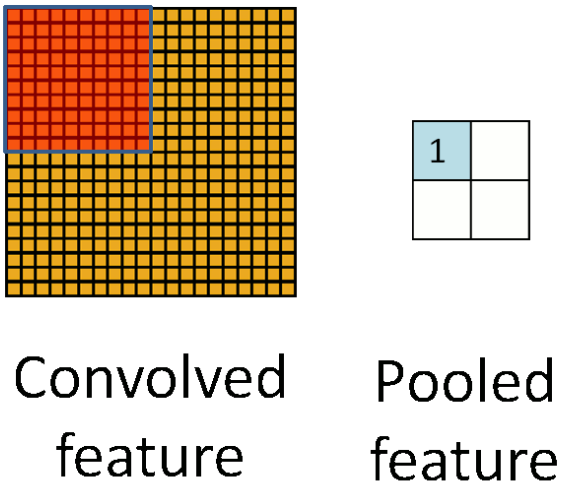
Τώρα, ο καλύτερος τρόπος για να κατανοηθεί ένα συνελικτικό επίπεδο είναι να φανταστείτε έναν φακό που λάμπει πάνω αριστερά από την εικόνα. Για να καταλάβετε πώς λειτουργεί αυτό, φανταστείτε πως ένας φακός λάμπει το φως του και καλύπτει μια περιοχή 5 x 5. Και τώρα, ας φανταστούμε ότι αυτός ο φακός ολισθαίνει σε όλες τις περιοχές της εικόνας εισαγωγής. Αυτός ο φακός ονομάζεται φίλτρο (ή μερικές φορές αναφέρεται ως νευρώνας ή πυρήνας) και η περιοχή που λάμπει ονομάζεται δεκτικό πεδίο (receptive field). Αυτό το φίλτρο είναι επίσης μια σειρά αριθμών (οι αριθμοί ονομάζονται βάρη ή παράμετροι). [14]

Δεύτερο είναι το Επίπεδο Ενεργοποίησης(Activation Layer) που εφαρμόζει το ReLu (Rectified Linear Unit), σε αυτό το βήμα εφαρμόζουμε τη λειτουργία ανορθωτή για να αυξήσουμε τη μη γραμμικότητα στο CNN. Οι εικόνες αποτελούνται από διαφορετικά αντικείμενα που δεν είναι γραμμικά μεταξύ τους.

Τρίτο, είναι το Επίπεδο Συγκέντρωσης (Pooling Layer) , το οποίο περιλαμβάνει υποδειγματοληψία χαρακτηριστικών. Εφαρμόζεται σε κάθε επίπεδο στον τρισδιάστατο(RGB). Συνήθως υπάρχουν υπερ– παράμετροι σε αυτό το επίπεδο: [14]

Η διάσταση της χωρικής έκτασης: είναι η τιμή του n που μπορούμε να πάρουμε τη διασταύρωση N και να παρουσιάσουμε χαρακτηριστικά και να χαρτογραφήσουμε σε μία μόνο τιμή.

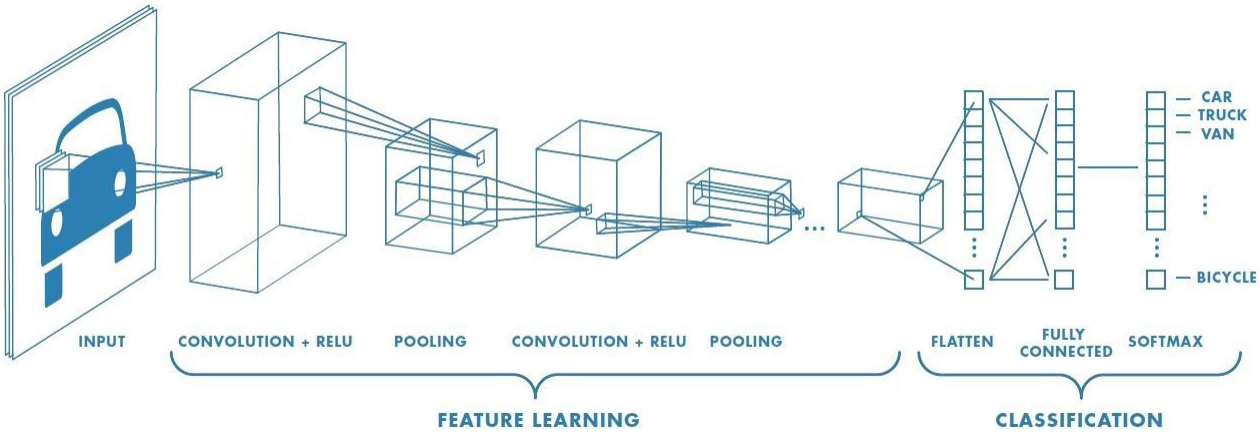
Stride: δηλαδή πόσες δυνατότητες παραλείπει το συρόμενο παράθυρο κατά μήκος και πλάτος



Εικόνα 9 Pooling Layer

Μια άλλη σημαντική έννοια των CNN είναι το Pooling layer, το οποίο είναι μια μορφή μη γραμμικής καθοδικής δειγματοληψίας. Υπάρχουν πολλές μη γραμμικές συναρτήσεις για την υλοποίηση του Pooling, όπου το Max Pooling είναι η πιο κοινή. Διαχωρίζει την εικόνα εισόδου σε ένα σύνολο ορθογωνίων και, για κάθε τέτοια υποπεριοχή, εξάγει το μέγιστο. Το Pooling Layer συνήθως λειτουργεί ανεξάρτητα σε κάθε βάθος ή κομμάτι της εισόδου και αλλάζει το μέγεθός του χωρικά[10]

Τέλος , είναι το Πλήρως Συνδεδεμένο Επίπεδο(Fully Connected Layer) το οποίο περιλαμβάνει την ισοπέδωση(Flattening). Αυτό συνεπάγεται με την μετατροπή ολόκληρης της μήτρας χαρτογράφησης χαρακτηριστικών σε μία μόνο στήλη η οποία στη συνέχεια τροφοδοτείται στο νευρωνικό δίκτυο για επεξεργασία. Με τα πλήρως συνδεδεμένα επίπεδα, συνδυάσαμε αυτές τις δυνατότητες μαζί για να δημιουργήσουμε ένα μοντέλο. Τέλος, έχουμε μια λειτουργία ενεργοποίησης όπως το softmax ή Sigmoid για να ταξινομήσουμε την έξοδο. [14] [10]



Εικόνα 10 Πλήρως Συνδεδεμένο Επίπεδο

Παρόμοιες εφαρμογές

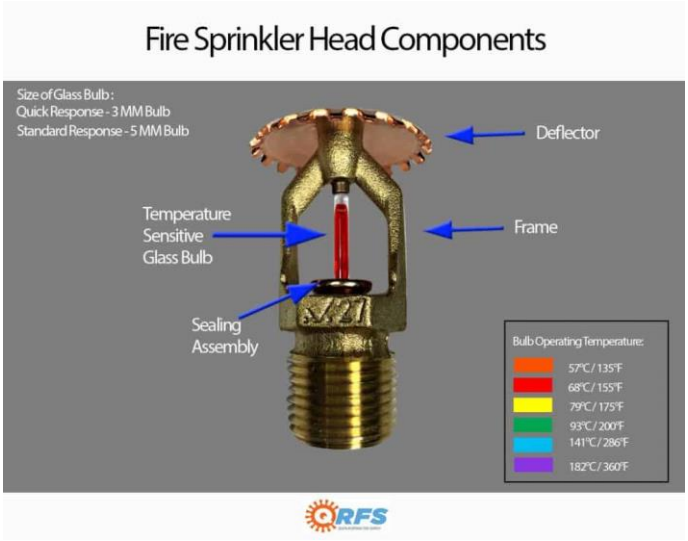
Στην αγορά αλλά και σε αναπτυξιακό επίπεδο μπορούμε να βρούμε αρκετές εφαρμογές της μηχανικής μάθησης και της αναγνώρισης εικόνας σε συνδυασμό με κάποιο σύστημα πυρόσβεσης.

Στην συνέχεια θα δούμε κάποια από αυτά από τα οποία εμπνεύστηκα την ιδέα καθώς και κάποια άλλα που ανακάλυψα κάνοντας έρευνα μέσα στα πλαίσια της διπλωματικής μου εργασίας.

Οι εφαρμογές της μηχανικής μάθησης είναι πρακτικά πλέον παντού και ας μην το καταλαβαίνουμε τόσο στο κομμάτι της κατηγοριοποίησης που κάνουμε εμείς αλλά και σε μορφή παλινδρόμησης. Για παράδειγμα, υπάρχουν συστήματα που κάνουν την αναγνώριση με αναγνώριση εικόνας από κάμερα μέσα σε ένα χώρο πχ εργοστασίου και ενεργοποιεί το σταθερό σύστημα πυρόσβεσης. Αυτό είναι ένα παράδειγμα που μπορεί να γίνει ευκολά αξιοποίηση του ήδη υπάρχων συστήματος ασφάλειας για να μπορέσουμε να λαμβάνουμε από αυτό το σύνολο των εικόνων από τον περιβάλλοντα χώρο και να προβούμε σε ενεργοποίηση της κατάσβεσης πριν επεκταθεί η φωτιά.

Ένα τέτοιο σύστημα μπορεί και πρέπει πάντα να συνδυάζεται και με άλλους αισθητήρες από την στιγμή που είναι σε εσωτερικό χώρο μπορεί ευκολά να εγκατασταθεί και σύστημα ενεργοποίησης του συστήματος από ανιχνευτές καπνού ή από τα κλασικά συστήματα κατάσβεσης κατά τα οποία άμα ένα ειδικό υγρό ζεσταθεί πάνω από μια συγκεκριμένη θερμοκρασία, το δοχείο που βρίσκεται μέσα διογκώνεται, στην συνέχεια σπάει και αρχίζει η ροή του νερού .

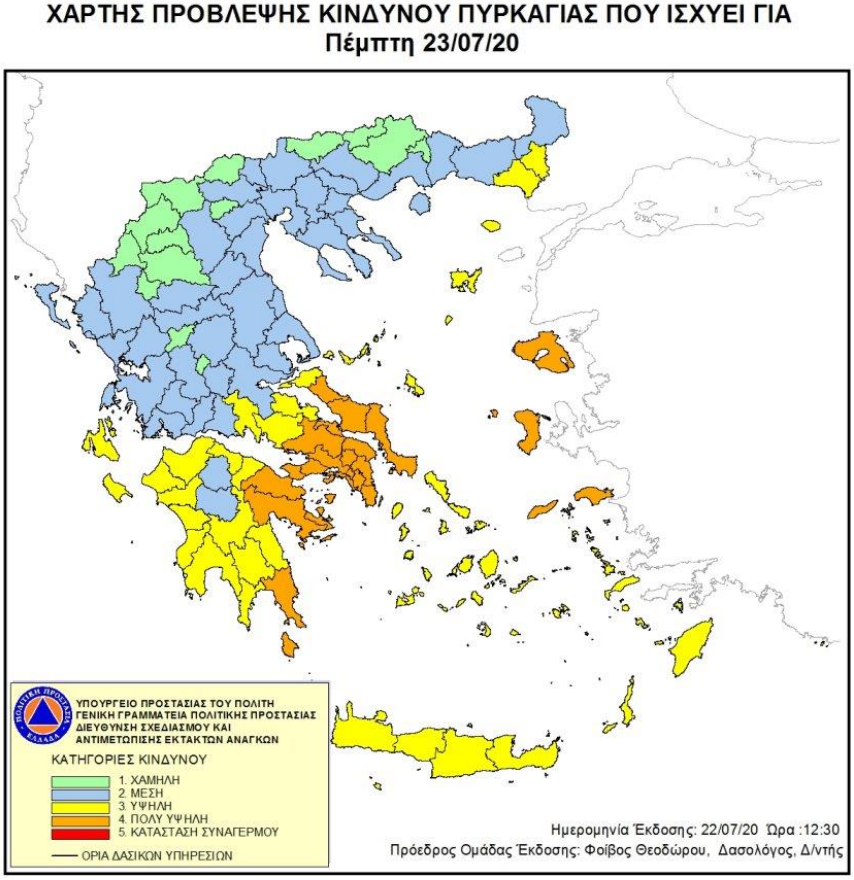
Παρακάτω μπορούμε να δούμε στην εικόνα τα μέρη ενός τέτοιου ακροφυσίου (Sprinkler Head) .



Εικόνα 11 Μέρη Ακροφυσίου

Άλλη μια εφαρμογή είναι ένα σύστημα μηχανικής μάθησης που μπορεί να λαμβάνει δεδομένα από διάφορους αισθητήρες στην περιοχή που μας ενδιαφέρει να παρακολουθήσουμε καθώς και το ιστορικό της περιοχής αυτής, όσον αφορά τις μετρήσεις από προηγούμενο διάστημα, και να κάνει μια πρόβλεψη ποσό μεγάλο είναι το ενδεχόμενο να πάρει εκ νέου φωτιά το σημείο αυτό, καθαρά και μόνο από δεδομένα χωρίς την παροχή κάποιας εικόνας της περιοχής. Κάτι τέτοιο εφαρμόζεται ήδη, ακόμα και στον ελληνικό χώρο από την Γενική Γραμματεία Προστασίας του Πολίτη με την Ημερήσιο Χάρτη Πρόβλεψης Κινδύνου Πυρκαγιάς.

Όπως μπορούμε να δούμε σαν αποτέλεσμα εμφανίζεται σε 5 κατηγορίες το πόσο πιθανό είναι να υπάρξει φωτιά στις περιοχές αυτές,. Με γνώμονα αυτό μπορούμε από την προηγουμένη μέρα ή και 2 μέρες πριν να έχουμε φροντίσει να εφαρμόσουμε επιπλέον προστατευτικά μέτρα στις περιοχές αυτές τόσο από θέμα προσωπικού, όπως για παράδειγμα επιπλέον πυροσβεστικά οχήματα και πλήρωμα, αλλά και να γίνει η πιθανή απαγόρευση πρόσβασης σε δασικές εκτάσεις όπως πχ ο Υμηττός στην Αττική για να μειώσουμε την πιθανότητα του κινδύνου εμφάνισης πυρκαγιάς από αμέλεια καθώς και για να είναι πιο εύκολη η πρόσβαση από την Πυροσβεστική Υπηρεσία λόγω του ότι δεν θα υπάρχουν οχήματα και άνθρωποι στους δασικούς δρόμους.



Εικόνα 12 Χάρτης Πρόβλεψης Κίνδυνου Πυρκαγιάς

Τέλος θα δούμε άλλο ένα σύστημα το οποίο ταυτίζεται αρκετά με το δικό μας με την μόνη διαφορά ότι αυτό κάνει χρήση θερμικής κάμερας για την αναγνώριση των πυρκαγιών αντί για απλή κάμερα ασφάλειας και μηχανική μάθηση , κάτι που στα πλαίσια της διπλωματικής εργασίας δεν ήταν ευκολά υλοποιήσιμο λόγω του πολύ αυξημένου κόστους μιας τέτοιας κάμερας.

Το FireRover είναι ένα σύνθετο σύστημα που παρέχει προστασία σαν την δικής μας εφαρμογής και με την χρήση κάμερας θερμικής κάνει ανάλυση εικόνας και αν εντοπίσει θερμοκρασία πάνω από τα επιτρεπτά όρια προχωράει σε ενημερώσει του κέντρου και ο χείριστης μπορεί απομακρυσμένα να κάνει κατάσβεση της φωτιάς με την χρήση ενός ενσωματωμένου συστήματος πυρόσβεσης . [13]

Η Υλοποίηση

Το Υλικό Κομμάτι

Το σύστημα μας αποτελείται από τα εξής υλικά:

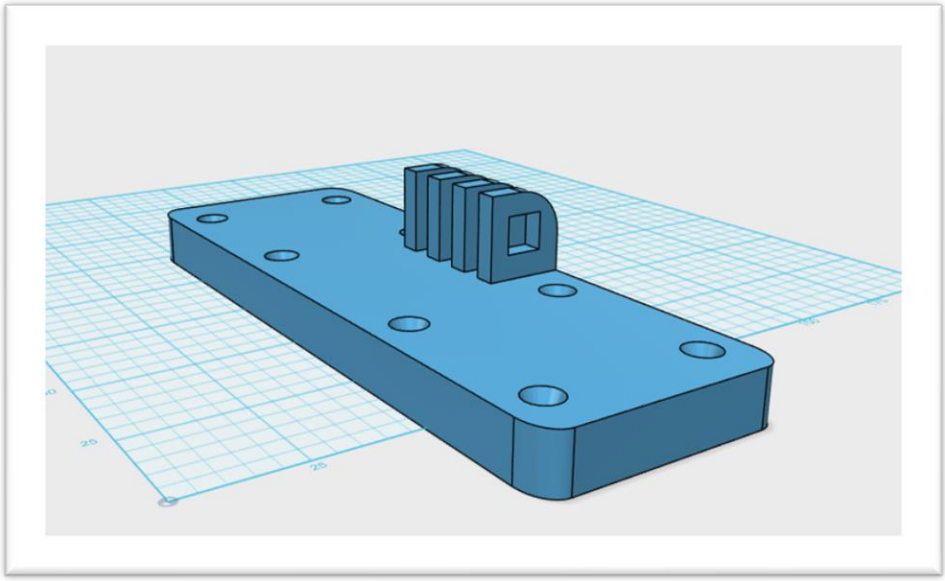
1. Κάμερα ασφάλειας μεγάλης ανάλυσης(DS-2CD2383G0-I(U) by Hikvision)
2. Ηλεκτροβάνα Μπαταρίας (Hunter PGV 100 GB DC 9v)
3. 2x Servo (DSservo 3230MG 270/360 Degree 30KG Torque)
4. Μεταλλική βάση για τα Servo
5. Wemos D1
6. 3x 3.3V 1-CHANNEL RELAY MODULE BOARD
7. 2x LM2596 3~40V DC-DC Buck Adjustable Converter / Step Down Module
8. 2xDC Adapter 12v 7A και 1.5A
9. 1 Εκτοξευτής Νερού
10. Λάστιχο νερού υψηλής πίεσης 16bar
11. Σύνδεσμοι Λάστιχου
12. 3D printed κουτί για τα ηλεκτρονικά
13. Τρίποδο
14. Κομμάτι ξύλο για στήριξη όλων των παραπάνω

Εικόνα 13 Πρόσοψη Συστήματος

Εικόνα 14 Μηχανισμός Κίνησης - Ακροφύσιο

Στα αριστερά βλέπουμε το σύστημα από την μπροστινή του όψη και στα δεξιά βλέπουμε από κοντά την βάση με τα Servo και το ακροφύσιο που είναι μέρος του συστήματος κατάσβεσης.

Για να μπορέσουμε να προσαρμόσουμε όλα τα υλικά μας πάνω στο κομμάτι ξύλου και στην συνέχεια να το ενώσουμε με το τρίποδο της κάμερας, χρειάστηκε να γίνει σχεδιασμός ενός επιπλέον κομματιού που θα αντικαταστήσει τον κλασικό σύνδεσμο που έχουν τα τρίποδα για να εφαρμόζονται επάνω τους κάμερες.



Εικόνα 15 Βάση προσαρμογής για Τρίποδο

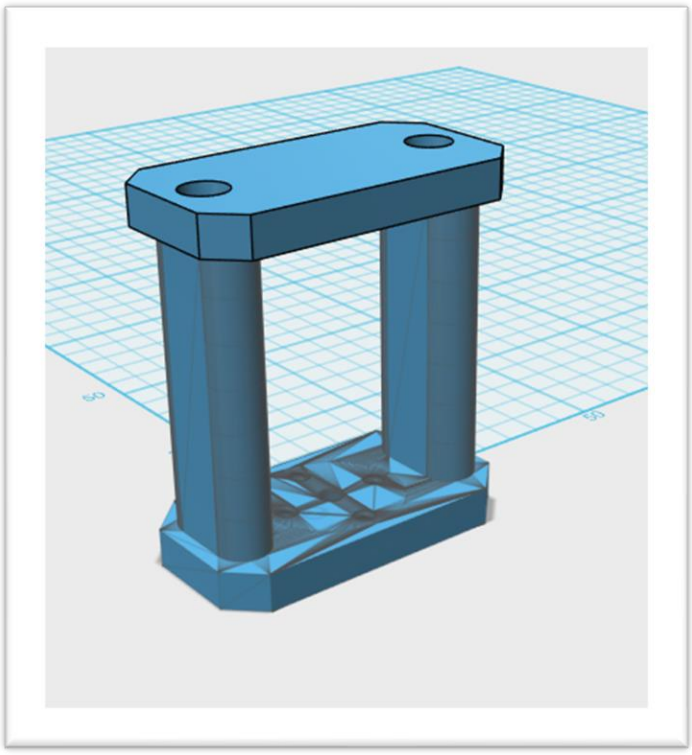
Ο σχεδιασμός του παραπάνω κομματιού έγινε με την χρήση ταχυμέτρου για να μπορέσουμε να πάρουμε τις μετρήσεις που χρειαζόμασταν από τον κανονικό σύνδεσμο και στην συνέχεια με την χρήση του 123D Design By Autodesk έγινε σχεδιασμός του κομματιού που βλέπουμε παραπάνω όπου αποτελείται από ένα μεγάλο ορθογώνιο με στρογγυλεμένες γωνίες, ώστε να μπορεί να πιάνει μια ικανοποιητική επιφάνεια πάνω στο ξύλο για να υπάρχει σταθερότητα, καθώς και 8 σημεία που μπορεί να περαστεί ξυλόβιδα ώστε να βιδωθεί με το ξύλο και να μπορεί να εφαρμόσει μετά πάνω σε οποιοδήποτε τρίποδο έχει το ίδιο μηχανισμό «κουμπώματος» . Το σχέδιο μετά εκτυπώθηκε σε 3d Εκτυπωτή με την χρήση υλικού PLA που το καθιστά φιλικό προς το περιβάλλον, δεδομένου της φύσης του υλικού που είναι το Γαλακτικό οξύ που έχουμε εξάγει από καλαμπόκι.

Το υλικό αυτό φυσικά χρησιμοποιήθηκε μόνο για το πειραματικό κομμάτι της διπλωματικής μας και δεν θα μπορούσε να γίνει χρήση του στη τελική έκδοση με κανονικό μέγεθος, λόγω του ότι το υλικό αυτό δεν μπορεί να μας παρέχει τόσο μεγάλη αξιοπιστία σε βάθος χρόνου με γνώμονα την θέση που θα βρίσκεται η κατασκευή αυτή.

Η μεγάλη έκθεση στον ήλιο, ειδικά κατά τους καλοκαιρινούς μήνες θα μπορούσε να είχε σαν κίνδυνο την θέρμανση του υλικού αυτού και να γίνει παραμόρφωση του σχήματός τους , καθώς το υλικό έχει glass transition temperature , περίπου 60-65 βαθμούς Κελσίους, με αποτέλεσμα την μη σωστή λειτουργία του.

Για αυτό το λόγο στην τελική έκδοση θα πρέπει όλα τα υλικά κατασκευής μας να είναι είτε μεταλλικά είτε εκεί που χρειάζεται πλαστικό να είναι πλαστικό με κατάλληλες ιδιότητες ώστε να μπορέσει να ανταπεξέλθει τόσο στις καιρικές αλλαγές όσο και σε χρονική διάρκεια , διότι δεν είναι επιθυμητό να χρειάζεται κάθε 5-6 μήνες αντικατάσταση κάποιο μέρος του RFT διότι αυτό θα το καθιστούσε αναξιόπιστο και με μεγάλη πιθανότητα να πάθει κάποια βλάβη κατά την διάρκεια ενός συμβάντος και να μην μπορέσει να ανταπεξέλθει και να αποβεί μοιραία η έλλειψη της άμεσης επέμβασής του.

Με ακριβώς την ίδια λογική κατασκευάσαμε και την βάση για το ακροφύσιο που μπορεί να προσαρμοστεί πάνω στην μεταλλική βάση για τους κινητήρες σέρβο και να έχουμε πλέον ένα γερό σημείο που ακουμπάει πάνω στην κατασκευή και μας δίνει την δυνατότητα να περιστρέφεται ανάλογα με την κίνηση των Servo.

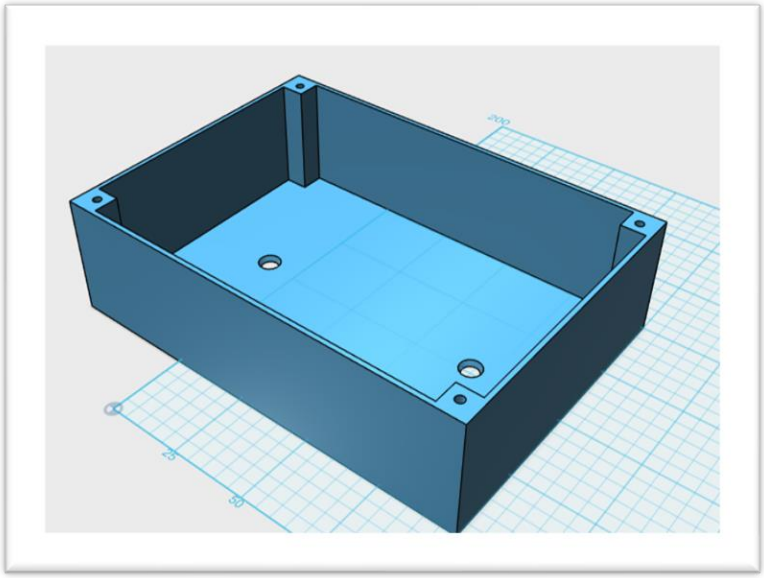


Εικόνα 16 Βάση στήριξης Ακροφυσίου

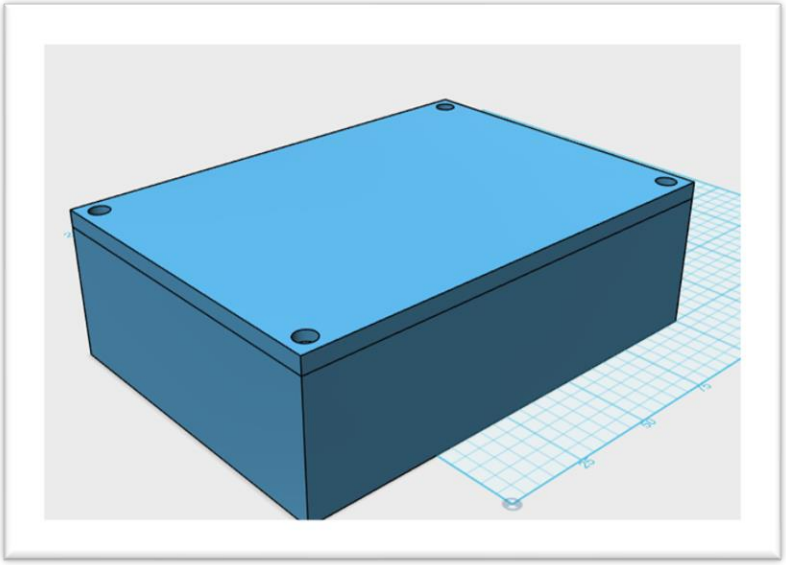
Δεδομένου του περιορισμένου χώρου και τις διατάξεις στην κάτω πλευρά της ξύλινης βάσης μας χρειαζόμασταν ένα κουτί ώστε να μπορέσουμε να βάλουμε μέσα όλα τα ηλεκτρονικά μέρη ώστε να είναι ασφαλή από ενδεχόμενες σταγόνες που θα μπορούσαν να τρέξουν από το ακροφύσιο και να βλάψουν τα ηλεκτρονικά μέρη.

Με εξαίρεση τα Servo, την κάμερα και την Ηλεκτροβάνα που είναι αδιάβροχα και δεν υπάρχει ανησυχία για το αν θα υπάρξει βραχυκύκλωμα, για όλα τα υπόλοιπα μετρήθηκαν οι διαστάσεις τους και βρήκαμε τις επιθυμητές διαστάσεις από ένα κουτί με καπάκι που εκτυπώθηκε και παρέχει μια αρκετά στοιχειώδη κάλυψη από το νερό. Σε καμία περίπτωση το συγκεκριμένο κουτί βέβαια δεν θεωρείται τελείως αδιάβροχο διότι υπάρχουν σημεία που δεν σφραγίζουν λόγω έλλειψης κάποιου ελαστικού όπως εφαρμόζει το καπάκι ώστε να δημιουργεί μια 100% αδιάβροχη προστασία.

Εξαιτίας του συγκεκριμένου μεγέθους δεν μπορούσε να γίνει αγορά κάποιου έτοιμου κουτιού από το εμπόριο με αδιαβροχοποίηση και για αυτό κατασκευάστηκε το παρακάτω κουτί.



Εικόνα 17 Εσωτερικό Κουτιού



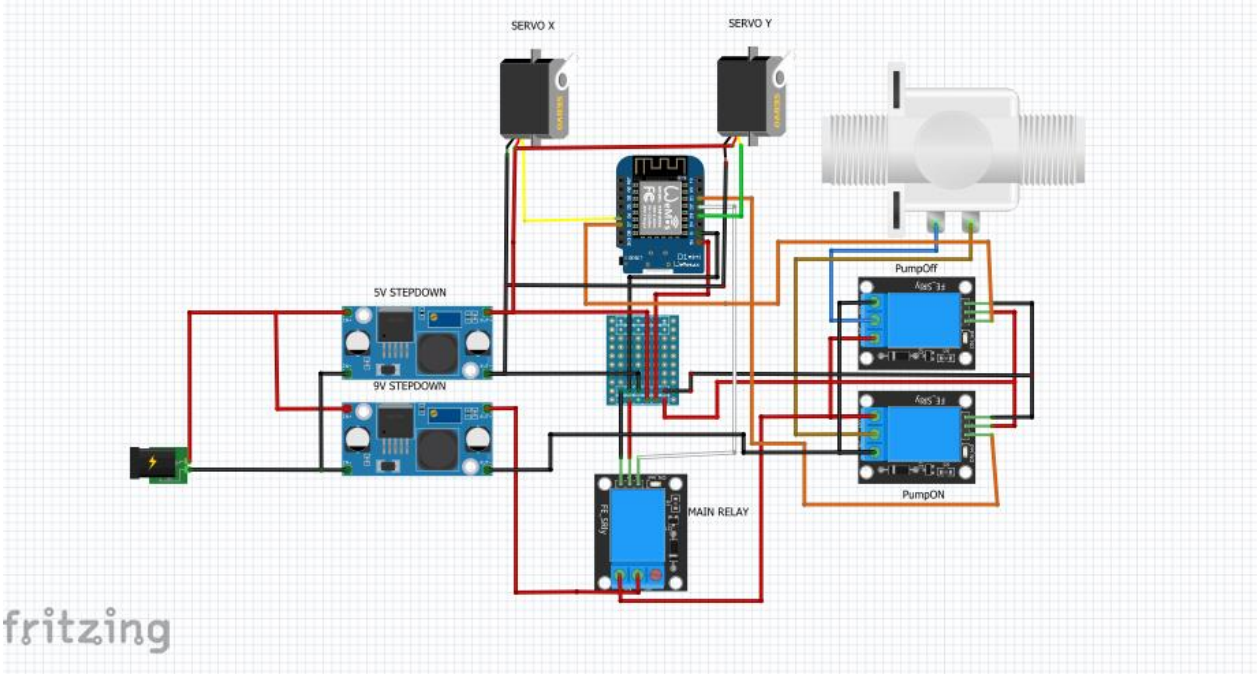
Εικόνα 18 Κουτί με καπάκι

Συνδεσμολογία του συστήματος

Το Σύστημα τροφοδοτείται από 2x12V (7A ,1.5A), το κομμάτι της κάμερας χρειάζεται περίπου 0,7A για να δουλέψει βέλτιστα οπότε του δίνουμε από το τροφοδοτικό του 1.5A, τα Servo θέλουν μέγιστα το κάθε ένα από περίπου 3,5 σε κατάσταση Stall, το solenoid της βάνας το θεωρούμε αμελητέο διότι η βάνα τραβάει ρεύμα στιγμιαία διότι το solenoid της είναι latching ενώ είναι φτιαγμένη για να δουλεύει με 9V μπαταρία οπότε, δεν υπάρχει μεγάλη απαίτηση σε Αμπέρ. Τέλος, το Wemos και τα Relay καταναλώνουν αμελητέο ρεύμα σε σχέση με τα υπόλοιπα στοιχεία οπότε τα τροφοδοτούμε με το 7A.

Μετά τα 12V τα μετατρέπουμε σε 5V ώστε να τροφοδοτήσουμε τα 2 servo για την κίνηση του συστήματος Κατάσβεσης και το Wemos, το οποίο με την σειρά του δίνει στα Main Relay, PumpON και PumpOff. Τέλος, από ένα DC-DC Stepdown κατεβάζουμε και δίνουμε μέσω ενός H-Bridge με την χρήση των PumpOn και PumpOff τάση 9V στο solenoid όπου αλλάζει την κατάσταση της ηλεκτροβάνα από ανοικτή σε κλειστή.

Χάρη στο Main Relay, αφού δώσουμε τάση στην βάνα για περίπου 200ms, κλείνει την παροχή τάσης προς τα PumpOn και PumpOff ώστε να μην υπάρχει συνεχομένη τάση στο solenoid και καταπονεί το σύστημα.



Εικόνα 19 Γραφική Απεικόνιση συνδεσμολογίας Συστήματος Arduino

Επεξήγηση κώδικα Arduino Wemos

Ο κώδικας που ελέγχει το σύστημα Κίνηση και Κατάσβεσης αποτελείται από έναν απλό κώδικα που χάρη στην συνεργασία του με το Thinger.io μπορεί να αποκτήσει ευκολά γραφικό περιβάλλον και εύκολη χρήση από τον τελικό χρήστη.

Ο κώδικας μας έχει τις βασικές εντολές για την σύνδεση με το Thinger.io και την βιβλιοθήκη που χρειάζεται για να κινηθούν τα Servo.

```
#include <ThingerESP8266.h>
#include <Servo.h>
#define USERNAME "calimancil"
#define DEVICE_ID "RFT1"
#define DEVICE_CREDENTIAL "QnN&xdQihAZt"

#define SSID "Hogwarts2.4"
#define SSID_PASSWORD "alohomora"

ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
```

Εικόνα 20 Ορισμός Στοιχείων Σύνδεσης στο Δίκτυο

Στην συνέχεια ορίζουμε τα pins και βοηθητικές μεταβλητές και οριστικοποιούμε τα Servo και την σύνδεση με το Thinger.io.

```
// defines pins numbers
#define SERVOX_PIN D6
#define SERVOY_PIN D3
#define pumpON D1
#define MAIN D2
#define pumpOFF D7
int servoposX;
Servo servoX;
int servoposY;
Servo servoY;
bool currentState;
```

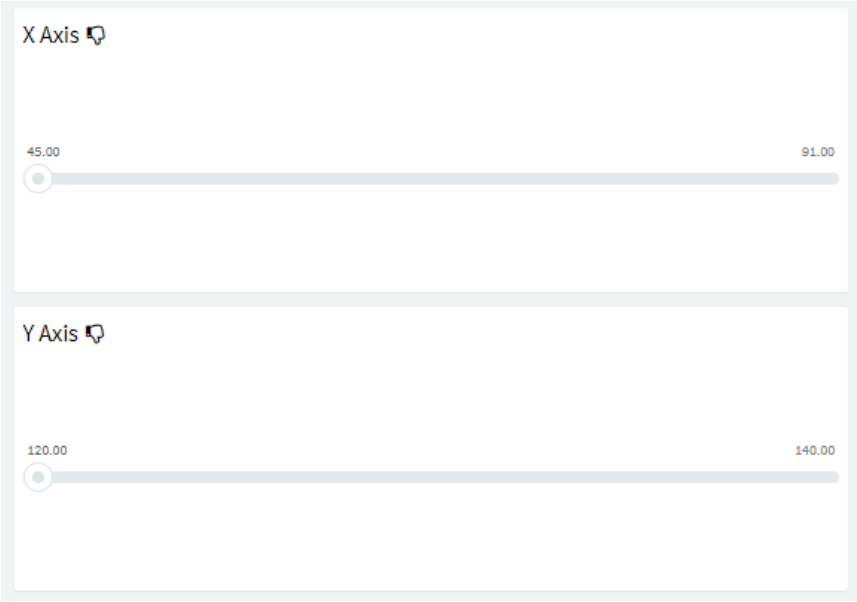
Εικόνα 21 Ορισμός των pins

```
void setup() {
  Serial.begin(9600);
  pinMode(pumpON, OUTPUT);
   pinMode(MAIN, OUTPUT);
  pinMode(pumpOFF, OUTPUT);
  servoX.attach(SERVOX_PIN);
  servoY.attach(SERVOY_PIN);
  thing.add_wifi(SSID, SSID_PASSWORD);
  digitalWrite(MAIN,HIGH);
```

Εικόνα 22 Αρχικοποίηση Servo

Μετά ορίζουμε τις εισόδους για την κίνηση τον Servo από τα Sliders που έχουμε βάλει στο thinger.io

```
 // controlling servo position
thing["servoposX"] << [](pson& in){
    if(in.is_empty())
      in = (int)servoposX;
    else
    {
      servoposX = (int)in;
      servoX.write(servoposX);
      Serial.println(servoposX);
    }
    servoposX = in;
};
thing["servoposY"] << [](pson& in){
    if(in.is_empty())
      in = (int)servoposY;
    else
    {
      servoposY = (int)in;
      servoY.write(servoposY);
      Serial.println(servoposY);
    }
    servoposY = in;
};
```



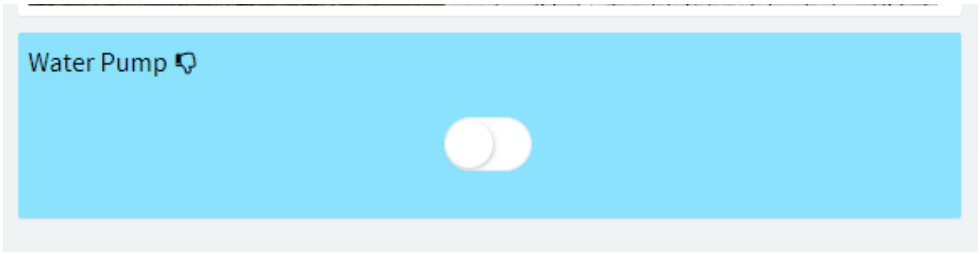
Εικόνα 23 Κώδικας Κίνησης Servo

Εικόνα 24 Γραφικό Περιβάλλον κίνησης στους δυο άξονες στο Thinger IO

Στην συνεχεία ορίζουμε και μια μεταβλητή που θα ελέγχει το άνοιγμα και κλείσιμο της ηλεκτροβάνας και θα έχει την μορφή διακόπτη στην πλατφόρμα μας.

```
//Waterpumb control
thing["WATER"] << [](pson& in){
  if(in.is_empty()){
      in = (bool) digitalRead(pumpON);
      }
    else{
      digitalWrite(MAIN,LOW);
      digitalWrite(pumpON, in ? HIGH : LOW);
      digitalWrite(pumpOFF, in ? LOW : HIGH);
      delay(150);
      digitalWrite(MAIN,HIGH);
      }
    };
```

Εικόνα 25 Ορισμός μεταβλητής ενεργοποίησης Ηλεκτροβάνας



Εικόνα 26 Γραφική Απεικόνιση διακόπτη ενεργοποίησης Ηλεκτροβάνας στο Thinger.io

Τέλος έχουμε το κομμάτι του Loop του προγράμματος μας, μια εντολή που ουσιαστικά λαμβάνει τα σήματα που μόλις ορίσαμε παραπάνω σχετικά με την κίνηση των Servo αλλά και με την κατάσταση της Ηλεκτροβανας, ανάλογα με το τι θα έχει πατήσει ο χρήστης από το Περιβάλλον του Thinger.io .

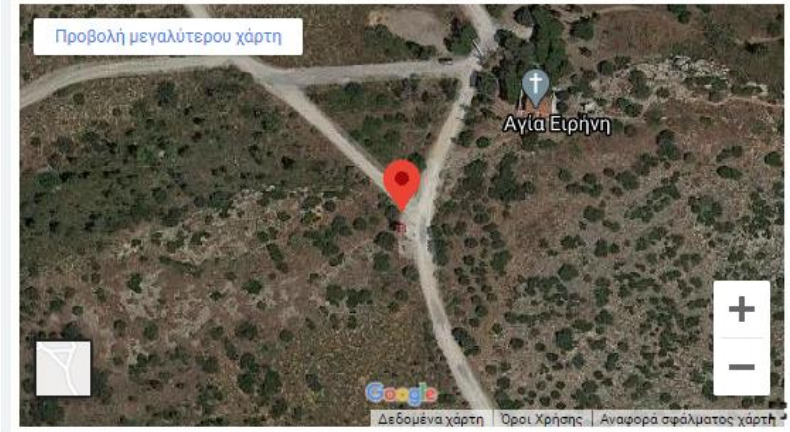
```
void loop() {
  thing.handle();
}
```

Εικόνα 27 Εντολή ενεργοποίησης της επικοινωνίας με το Thinger.io

Επιπρόσθετα στοιχεία που φαίνεται και στο τέλος της εργασίας που είναι το σύνολο της πλατφόρμας είναι ένα μεγάλο πεδίο με την Ονομασία του RFT σε περίπτωση που υπάρχει πάνω από ένα ώστε να γνωρίζουμε σε ποιο έχουμε συνδεθεί και ποιο ελέγχουμε καθώς και ένας χάρτης με το ακριβές σημείο που έχουμε τοποθέτηση το RFT για να γνωρίζουμε και την θέση του.

RFT Agia Eirini

Εικόνα 28 Γραφική απεικόνιση Ονομασίας Συσκευής στο Thinger.io



Εικόνα 29 Ενσωματωμένος Χάρτης για την Θέση του Συστήματος στο Thinger.io

Μοντέλο Μηχανικής Μάθησης

Το Μοντέλο μας δημιουργήθηκε μετά από μελέτη και παραμετροποίηση των μοντέλων που μπορούν να βρεθούν στην σειρά μαθήματων από το Udacity [5] [11].

Το μοντέλο έχει βασιστεί στο Transfer Learning κι έχουμε φορτώσει «Mobile Net» που έχει εκπαιδευτεί από την Google πάνω σε μια βάση με πάνω από 1,4 εκατομμύρια εικόνες με 1000 διαφορετικές κατηγορίες εικόνων.

Με την χρήση ενός ήδη εκπαιδευμένου μοντέλου το τελικό μας αποτέλεσμα θα έχει ακόμα μεγαλύτερη ακρίβεια στην πρόβλεψη του καθώς και θα έχει λιγότερες πιθανότητες να κάνει overfitting, ορίζοντας το Mobile Net σαν ένα επιπρόσθετο επίπεδο Keras και ορίζοντας τα βάρη του να μην εκπαιδεύονται εκ νέου από εδώ και πέρα ώστε να μπορέσουμε να προσθέσουμε εμείς άλλο ένα Layer που θα έχει δυο κατηγορίες .

Οι κατηγορίες που μας ενδιαφέρουν είναι το “Smoke” και το “No Smoke” κι έτσι έχουμε ορίσει τις εικόνες μας είτε είναι με μορφή καπνού είτε με μορφή φωτιάς.

Λάβαμε από το ίντερνετ ένα σύνολο εικόνων περίπου 2000 που χωριστήκαν τόσο σε εικόνες για εκμάθηση του μοντέλου αλλά και για δοκιμή αυτού με αναλογία 75% εκπαίδευση και 25% δοκιμή.[4]

Στην αρχή του κώδικα ορίζουμε όλες τις βιβλιοθήκες τις οποίες θα χρειαστούμε για να μπορέσουμε να φτιάξουμε το μοντέλο αλλά και να κάνουμε διαχείριση της αποστολής του e-mail στον χρήστη ώστε να ενημερωθεί σε περίπτωση φωτιάς. [6]

Αφού φορτώσουμε τις βιβλιοθήκες καλούμε το έτοιμο μοντέλο Mobile Net και του ορίζουμε τις μεταβλητές «non-Trainable» ώστε να μην ασχοληθούμε με την εκπαίδευση του πάνω σε αυτά που ήδη γνωρίζει, ορίζουμε τις διαστάσεις των εικόνων μας στα 224 pixel που είναι η τιμή που λαμβάνει το Mobile Net και ορίζουμε ότι θα χωρίσουμε τις εικόνες που θα του βάλουμε για εκμάθηση σε ομάδες των 32.

Στην συνέχεια ορίζουμε το path από οπού έχουμε βάλει τις εικόνες που θέλουμε να εκπαιδεύσουμε το μοντέλο μας, ορίζουμε στο μοντέλο μας τα paths από τους διαφορετικούς φακέλους σχετικά με τις δυο κατηγορίες εικόνων που έχουμε καθώς και 2 φακέλους για κάθε κατηγορία, μια που είναι για εκπαίδευση και μια που είναι για δοκιμή.

Για να αποφύγουμε όσο μπορούμε το overfitting και να δημιουργήσουμε παραπάνω εκδοχές από της εικόνες μας της τροποποιούμε κάνοντας τα εξής περιστροφή, μεγέθυνση ,μετακίνηση και αλλά.

Στην συνέχεια βλέπουμε τα αποτελέσματα από αυτή την επεξεργασία των αρχικών εικόνων, βάση της διαμόρφωσης των φακέλων μας και το μοντέλο αναγνωρίζει ότι έχει 2 κατηγορίες.

Ορίζουμε ότι πλέον θα εκπαιδευτεί σε αυτές τις δυο κατηγορίες και θα εκπαιδεύσει τις παραμέτρους μόνο αυτού του συστήματος και όχι ολοκλήρου του Mobile Net.

Εκπαιδεύουμε πλέον το μοντέλο μας για έναν Χ αριθμό εποχών, για να μπορέσουμε να δούμε πόσο καλά έχει εκπαιδευτεί το μοντέλο μας όταν ζητείται να εξάγει σε 2 γραφήματα το ποσοστό

επιτυχίας καθώς και το Loss από την εκπαίδευση. Στην πρώτη περίπτωση βλέπουμε ότι το μοντέλο μας έχει καταφέρει 100% επιτυχία, αυτό βέβαια είναι λανθασμένο.

Το μοντέλο μας δυστυχώς έχει μικρό πλήθος εικόνων και είναι αρκετά όμοιες μεταξύ τους και δυστυχώς τις έχει μάθει παραπάνω από όσο θέλαμε, έχει δηλαδή πλέον το λεγόμενο overfitting.

Συνεχίζοντας, του αναφέρουμε να μας ενημερώσει τι αριθμό έχει δώσει σε κάθε μια από τις κατηγορίες μας ώστε να ξέρουμε τι αναγνωρίζει ως καπνός και τι όχι.

Φορτώνοντας μια εικόνα με καπνό αρκετά όμοια με αυτές που έχει εκπαιδευτεί το μοντέλο μας κάνουμε την δοκιμή να δούμε αν θα το αναγνωρίσει σαν καπνό. Στην περίπτωση μας, αν εντοπίσει καπνό γίνεται επισύναψη του αρχείου και αποστέλλεται με e-mail στον χρήστη για να κάνει επιβεβαίωση και κατάσβεση άμα χρειαστεί. [6]

Τέλος, καλό είναι να σώσουμε το μοντέλο μας, αποθηκεύοντας το αρχικά σαν ένα απλό Keras Layer με μορφή .h5. Αυτό όμως μας δίνει περιορισμούς για την επανεκπαίδευση του, παρόλο που μπορούμε να το χρησιμοποιήσουμε για να κάνουμε προβλέψεις και να δούμε αν είναι όντως ίδιο με το προηγούμενο που μόλις αποθηκεύσαμε.

Για αυτό τον λόγο θα πρέπει να σώσουμε «ολόκληρο» το μοντέλο με μορφή Saved Model. Αυτό θα μας δώσει την δυνατότητα να το επεξεργαστούμε ξανά και να μπορέσουμε να το βελτιώσουμε από εκεί που το αφήσαμε χωρίς να πρέπει να το εκπαιδεύουμε ξανά για τις εικόνες που έχει ήδη δει. Τέλος, φορτώνουμε ξανά ολόκληρο το μοντέλο και ελέγχουμε ότι είναι ακόμα ίδιες οι προβλέψεις με αυτές που είχε και το αρχικό μας μοντέλο.

Τώρα πράττουμε με τρόπο ώστε να ξανά εκπαιδεύσουμε το μοντέλο μας με παραπάνω εικόνες καθώς και να του δώσουμε επιπλέον τεχνικές για να μπορέσει να μειωθεί το overfitting,(Κώδικας saved model script and retrain), φορτώνουμε το μοντέλο μας και προσθέτουμε αρκετές επιπλέον τεχνικές τύπου Dropout, Dense, MaxPooling2D, Conv2D.

Βλέπουμε πλέον ότι οι παράμετροι έχουν γίνει από 2.5 εκατομμύρια κοντά στα 9,6 εκατομμύρια, πράγμα που θα μας βοηθήσει να βελτιώσουμε το μοντέλο καθώς και με την προσθήκη νέων εικόνων διαφορετικών από πριν. Κάνοντας αυτά βλέπουμε ότι πλέον έχουμε πέσει στο 90% επιτυχία, το οποίο είναι πιο ρεαλιστικό, ελέγχουμε εκ νέου με μια εικόνα και βλέπουμε ότι δεν έχει καπνό και όντως το μοντέλο μας αναγνωρίζει ότι δεν έχει καπνό.

Τέλος ξανά αποθηκεύουμε το νέο εκπαιδευμένο μοντέλο ώστε στο μέλλον να εκπαιδευτεί εκ νέου όταν έχουμε νέες εικόνες.

Το τελικό αρχείο που δημιουργήθηκε (Final Python Script For Fire Check) έχει σαν σκοπό μόνο τον έλεγχο μιας εικόνας σχετικά με το αν έχει καπνό ή όχι καθώς και να κάνει ενημερώσεις μέσω e-mail στον χρήστη σε περίπτωση καπνού. Επιπρόσθετα έχει ενσωματωθεί κώδικας που κάθε μια πρόβλεψη που κάνει είτε αυτή είναι καπνός είτε όχι την αποθηκεύει τοπικά σε φάκελο ανάλογα του αποτελέσματος που έχει βγάλει το μοντέλο μας με μοναδικό όνομα την ημερομηνία και ώρα που έγινε η λήψη ώστε να μπορεί μετά ο χείριστης να πάρει αυτά τα δεδομένα και να κάνει έλεγχο αν όντως έχει κάνει σωστές ή όχι προβλέψεις και να εκπαιδευτεί ξανά το μοντέλο βάση αυτών με ρεαλιστικές εικόνες από τον τόπο που θα έχει εγκατασταθεί το RFT. [6]

Στην συνέχεια μπορούμε να δούμε το κομμάτι από το MacroRecorder, όπου χρησιμοποιήθηκε για να αντιγράφει τις κινήσεις από το ποντίκι και να κάνει αυτόματη αποθήκευση νέας εικόνας κάθε 45 δευτερόλεπτα περίπου από την κάμερα, να μπορεί κάθε 1 λεπτό περίπου το μοντέλο να έχει νέα δεδομένα για να κάνει εκ νέου έλεγχο , καθώς και τους διάφορους κώδικες σε μορφή αρχείων .bat που δημιουργήθηκαν ώστε με το πάτημα ενός αρχείου και μόνο να γίνεται η διαδικασία ανοίγματος τόσο του συστήματος της κάμερας αλλά και του μοντέλου αναγνώρισης για να είναι εύκολη και άμεση η χρήση του συστήματος από οποιοδήποτε άτομο, χωρίς να χρειάζεται να κάνει πολύπλοκες και χρονοβόρες διαδικασίες .

Με 2 κλικ το σύστημα είναι ενεργό και κάνει έλεγχο συνέχεια μέχρι να διακοπεί ανά πάσα στιγμή από τον χειριστή του Η/Υ που είναι συνδεδεμένη πάνω το RFT.

Πειράματα – Στάδια Υλοποίηση

Η κατασκευή του RFT τόσο στο κομμάτι των υλικών αλλά και του κώδικα χρειάστηκε να περάσει από πολλά στάδια και πολλές δοκιμές ώστε να φτάσει στο κομμάτι που είναι στην παρούσα κατάσταση. Παρακάτω θα δούμε μερικά από αυτά για να μπορέσουμε να εστιάσουμε στα προβλήματα που αντιμετωπίσαμε με τις δοκιμές μας και στην συνέχεια διορθώσαμε και μάθαμε ποιες παραμέτρους πρέπει να προσέξουμε και να έχουμε υπόψιν όταν κάνουμε την τελική κατασκευή σε πραγματικό μέγεθος.

Ξεκινώντας από το κομμάτι της κατασκευής, στην αρχή έγινε δοκιμή με λάστιχο πολύ κακής ποιότητας με αποτέλεσμα το λάστιχο νερού να τσακίζει και να μην μπορεί να περνάει το νερό από την βρύση και να φτάνει μέχρι και την έξοδο του ακροφυσίου και να μας δίνει μια εμβέλεια πάνω από 1 μετρό. Μετά από αυτή την δοκιμή έγινε σχετική ερευνά και αγοράστηκε καλής ποιότητας λάστιχο ώστε να μην τσακίζει εύκολα και να μπορεί να αντέξει πολύ μεγαλύτερη μηχανική καταπόνηση καθώς και σε βάθος χρόνου να μην επηρεαστεί από τις καιρικές συνθήκες.

Ένα ακόμα σημαντικό κομμάτι είναι η πίεση του νερού. Οι πρώτες δοκιμές έγιναν σε λάστιχο ταράτσας στον 3ο όροφο με αποτέλεσμα λόγω της υψομετρικής διαφοράς από την επιφάνεια των αγωγών του νερού η πίεση που έφτανε στο RFT να είχε δυο αρνητικά αποτελέσματα.

1. Δεν υπήρχε μεγάλη εμβέλεια μετά την έξοδο του ακροφυσίου
2. Δεν υπήρχε αρκετή πίεση για να κλείσει η Ηλεκτροβάνα

Λόγω της φύσεως της ηλεκτροβάνας, για να μπορέσει να κλείσει δεν χρειάζεται μόνο η εφαρμογή τάσης στο solenoid της. Για να μπορέσει να λειτουργήσει σωστά χρειάζεται από 1,5bar έως 10bar ώστε κάθε φορά που γίνεται αλλαγή στην πολικότητα του solenoid να ανοίγει και να κλείνει αντιστοίχως.

Η δεύτερη δοκιμή έγινε στο ίδιο επίπεδο με τους αγωγούς οπού εκεί μπορούσαμε και είχαμε αρκετά bar ώστε να ανταποκρίνεται η ηλεκτροβάνα και να κλείνει σωστά, καθώς επίσης και να έχουμε μια εμβέλεια περίπου 6 μετρά με μόνη υψομετρική διαφορά το ύψος από το έδαφος μέχρι και το ακροφύσιο, περίπου 65 εκατοστά.

Παρόλα αυτά, τα Bar στην συγκεκριμένη έξοδο ήταν ακόμα σχετικά χαμηλά οπότε αν γίνει δοκιμή σε παροχή με μεγαλύτερη πίεση θα μπορέσουμε να έχουμε εμβέλεια περίπου 10 μετρά με το ίδιο ύψος από το έδαφος.

Λόγω της πίεσης που εφαρμόζεται κατά το άνοιγμα της Ηλεκτροβάνας στις πρώτες δοκιμές που κάναμε είχαμε στην διάθεση μας ένα Servo το οποίο είχε 20kg Torque με αποτέλεσμα σε κάποια σημεία της κίνησης του κάθετου άξονα το Servo έπαιρνε παραπάνω ρεύμα διότι η πίεση που του ασκούταν από την έξοδο του ήταν μεγάλη και υπήρχε έναν ηλεκτρονικός ήχος από την μεγάλη ένταση του ρεύματος. Το πρόβλημα αυτό επιλύθηκε με την χρήση ενός servo 30kg ώστε να μπορεί πιο ευκολά να ανταπεξέλθει στις πιέσεις χωρίς να υπάρχει κίνδυνος αστοχίας.

Ας περάσουμε τώρα στο κομμάτι της κάμερας όπου σαν συσκευή είχαμε να αντιμετωπίσουμε 2 βασικά προβλήματα. Το πρώτο είχε να κάνει με τις δυνατότητες της κάμερας καθώς η κάμερα έχει την δυνατότητα αν το φως στο περιβάλλον δεν είναι επαρκές να γυρίσει σε μια κατάσταση που η εικόνα που λαμβάνει πλέον δεν είναι έγχρωμη (RGB) αλλά γίνετε ασπρόμαυρη(Greyscale) φέροντας ως αποτέλεσμα οι εικόνες που θα αποθηκεύουμε και θα περνάμε στο μοντέλο μας να είναι λανθασμένες.

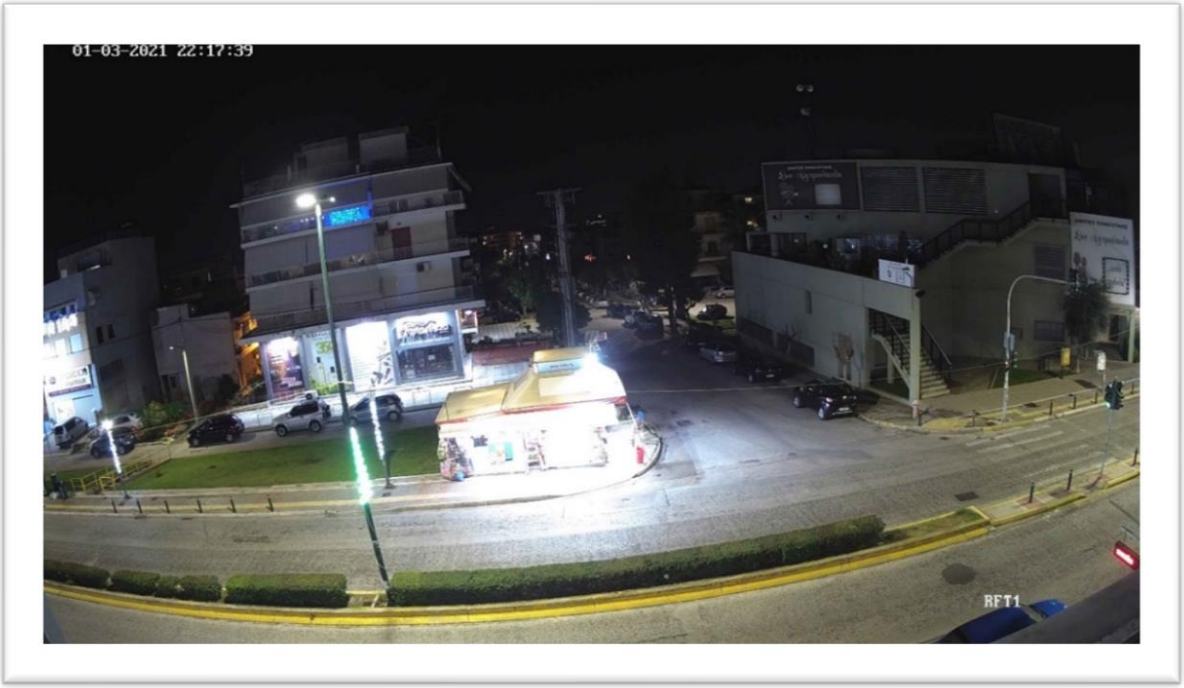
Από αυτή την οπτική, το μοντέλο μας έχει εκπαιδευτεί να δουλεύει και έχει μάθει να αναγνωρίζει φωτιά από εικόνες που έχουν κανονικά χρώματα(RGB) που σημαίνει ότι δεν θα μπορεί να κάνει σωστή πρόβλεψη και είτε θα μας λέει ότι υπάρχει φωτιά ενώ δεν υπάρχει διότι έχει μάθει ότι τα χρώματα που είναι ο καπνός είναι λευκό, γκρι και μαύρο ή δεν θα μπορεί να καταλάβει καν αν υπάρχει φωτιά ή όχι και θα μας ενημερώνει ότι δεν χρειάζεται παρέμβαση ακόμη και σε ύπαρξη φωτιάς.

Και στις δυο περιπτώσεις το αποτέλεσμα είναι λανθασμένο και μπορεί να έχει σαν αποτέλεσμα είτε μια μικρή ενόχληση ή ακόμα και μεγάλο κίνδυνο σε περίπτωση που υπάρχει φωτιά και δεν το ανιχνεύσει. Για αυτό τον λόγο απενεργοποιήσαμε την αυτόματη αυτή αλλαγή από την κάμερα.

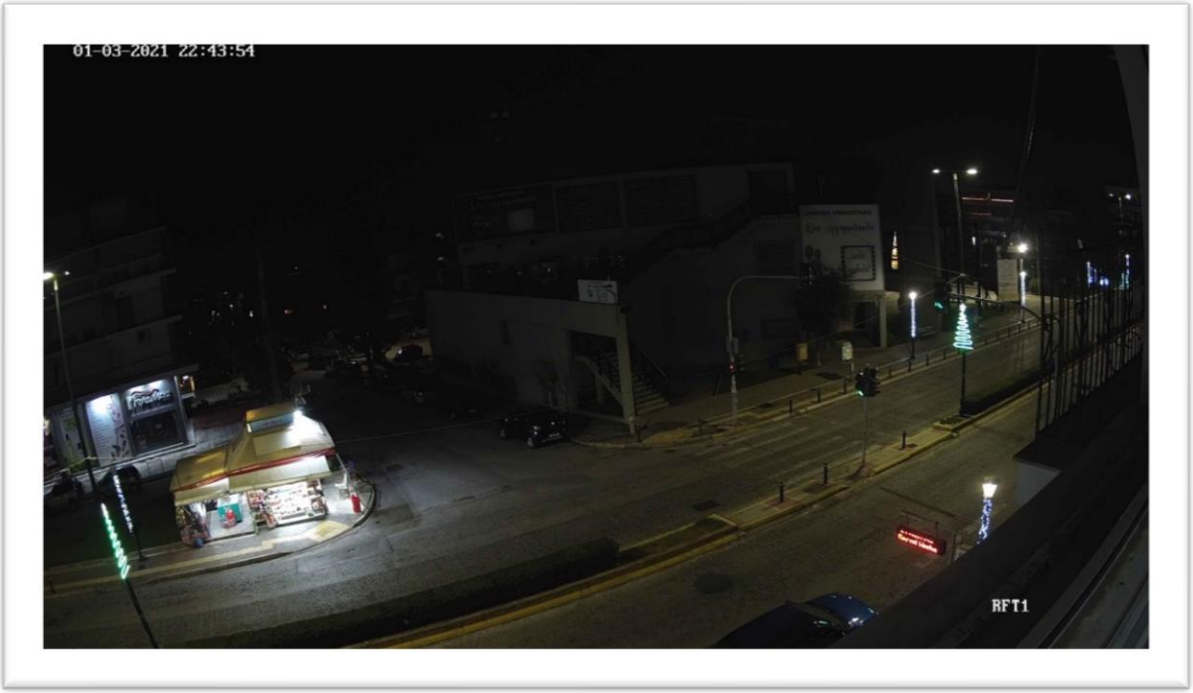
Ένα άλλο κομμάτι που χρειάζεται ρύθμιση ανάλογα με το περιβάλλον που θα μπει το RFT είναι η ευαισθησία της κάμερας στα φώτα. Σε παραδείγματα που έγιναν σε αστικό περιβάλλον, η κάμερα έβλεπε ένα περίπτερο που την νύχτα είχε ανάμενα πάρα πολλά φώτα για να είναι ευκολά ορατό από τους οδηγούς εξαιτίας του σημείου στο οποίο βρίσκεται ώστε να μην υπάρχει κίνδυνο ατυχήματος.

Ως αποτέλεσμα αυτών, όταν η ευαισθησία της κάμερας ήταν αρκετά υψηλή για να βλέπει όλη την περιοχή που μπορεί καθαρά, το περίπτερο που ήταν πολύ έντονα λευκά φωτισμένο έφερε ως συνέπεια την μη σωστή αναγνώρισή του φωτός, αναγνωρίζοντάς το ως φωτιά μέσω του συνδυασμού των χρωμάτων που υπήρχαν, άρα ταυτιζόταν αρκετά με αυτό μιας πυρκαγιάς. Το δεδομένο πρόβλημα διορθώθηκε μέχρι ένα σημείο με δυο κινήσεις, πρώτα μειώνοντας την ευαισθησία της κάμερας και δεύτερον κάνοντας εκ νέου εκμάθηση του μοντέλου μας με την χρήση εικόνων από το φωτισμένο περίπτερο ώστε να μάθει ότι αυτό δεν είναι φωτιά ή καπνός.

Παρακάτω βλέπουμε τα δυο παραδείγματα από τις εικόνες μας όπου στην πρώτη (Εικόνα 30) το μοντέλο λόγω της έντονης φωτεινότητας αναγνωρίζει το περίπτερο ως φωτιά και μετά η δεύτερη(Εικόνα 31) όταν έχει διορθωθεί το πρόβλημα αυτό.



Εικόνα 30Μεγάλη Ευαισθησία – Λανθασμένη αναγνώριση



Εικόνα 31 Διορθωμένη Ευαισθησία – Σωστή Αναγνώριση

Παρουσίαση Κατασκευής στην ΕΟΔΠΕΑ

Στα πλαίσια της διπλωματικής εργασίας κατά την ολοκλήρωση της κατασκευής οργανώθηκε μια μικρή παρουσίαση σε κύρια μέλη της ΕΟΔΠΕΑ ώστε να μπορέσω να λάβω την άποψη και τους σχολιασμούς από άτομα που έχουνε εμπειρία και ασχολούνται στο κομμάτι της Πυρόσβεσης και προστασίας του πράσινου.

Η Εθελοντική Ομάδα Δασοπροστασίας και Πυρόσβεσης Ελληνικού Αργυρούπολης γεννήθηκε μετά την καταστροφική πυρκαγιά στον Υμηττό το καλοκαίρι του 2015. Με διάθεση ανιδιοτελούς προσφοράς, δεκάδες εθελοντές κάθε ηλικίας ανταποκρίθηκαν στο κάλεσμα για την προστασία του βουνού μας. Έκτοτε έχουν γίνει σημαντικά έργα για τη θωράκιση του τομέα μας ενώ δίνεται βαρύτητα πρωτίστως στην πρόληψη και δευτερευόντως στην καταστολή.

Στην συνέχεια ακολουθούν τα σχόλια τον μελών της ομάδας όπως ακριβώς μας τα αναφέρανε.

Ο κ Βασίλης Κρητικός Επικεφαλής της ομάδας δηλώνει:

«Η παρουσίαση της διπλωματικής με θέμα «αυτόματος πύργος πυρόσβεσης» που παρακολούθησα στο Πυροφυλάκιο Αργυρούπολης μου προξένησε ιδιαίτερο ενδιαφέρον για την ευρηματικότητα του σχεδιασμού της και την αξιοποίηση της τεχνικής νοημοσύνης που χρησιμοποιεί.

Η γνώση που απορρέει από την θεσμική μου αρμοδιότητα ως Αντιδημάρχου Περιβάλλοντος και Πολιτικής Προστασίας και η εμπλοκή μου με τα μέσα πυρόσβεσης που είναι ανεπτυγμένα στα διοικητικά όρια του Δήμου Ελληνικού Αργυρούπολης στον Υμηττό, μου δίνει την βεβαιότητα ότι η εφαρμογή της παρουσίασης θα μπορούσε κάλλιστα να εξελίξει τα τοποθετημένα σταθερά πυροβόλα που χρησιμοποιούμαι τώρα , να αυτοματοποιήσει τον χειρισμό τους σε συνάρτηση με την οπτική η θερμική δυνατότητα έγκαιρης ανίχνευσης πυρκαγιάς.

Η απομνημόνευση του πεδίου επιτήρησης, η υπολογιστική δύναμη του μέσου, η μεταβίβαση της πληροφορίας στο κέντρο επιχειρήσεων του Δήμου σε 24ωρη βάση, η απομακρυσμένη ενεργοποίηση καθώς και ο τηλεχειρισμός των υδροβόλων θα αποτελέσει προνομιακή δυνατότητα για την πρόληψη αλλά και προσβολή της πυρκαγιάς που μπορεί να υπάρξει στο βουνό μας.

Η καταγραφή των οπτικών δεδομένων συμβάντων αλλά και η απομακρυσμένη επόπτευση του πεδίου ενδιαφέροντος είναι άλλη μια συνδυαστική δυνατότητα της εν λόγω εφαρμογής.

Εύχομαι ο ενθουσιασμός και το μεράκι του δημιουργού να είναι το εφαλτήριο της πραγμάτωσης της τόσο εμπνευσμένης αλλά ρεαλιστικής εφαρμογής.

Ο Δήμος Ελληνικού Αργυρούπολης και η Εθελοντική Ομάδα Δασοπροστασίας Πυρόσβεσης αγκαλιάζουν τέτοιες προσπάθειες και ως παράδειγμα αναφέρω την πρόταση του ιδίου σπουδαστή σας κ. Μιχάλη Χονδρογιάννη για δημιουργία πλατφόρμας πληρότητας νερού των δεξαμενών υδροληψίας του Υμηττού με αισθητήρες στάθμης GSM που υλοποιήσαμε πρόσφατα και επεκτείνουμε συνεχώς.»

Η κ. Πασχαλιά Παλαιού γραμματέας της ΕΟΔΠΕΑ δηλώνει:

«Ο Μιχάλης Χονδρογιάννης, παρουσίασε σε συντονιστική ομάδα της ΕΟΔΠΕΑ το σύστημα RFT (remote fire tower) στα πλαίσια της διπλωματικής του εργασίας. Από την αρχή, ξεχωρίζει κανείς την απλή και κατανοητή σχεδίαση και συνδεσμολογία, τον μικρό αριθμό εξαρτημάτων αλλά και τη διαχείριση χωρίς πολύπλοκες ρυθμίσεις.

Όλη η ιδέα αποσκοπεί στην καταστολή-κατάσβεση αλλά και πρόληψη δασικών πυρκαγιών καθώς το σύστημα «είναι εκπαιδευμένο» να διαβάζει εικόνες από τέτοιου είδους συμβάντα και να τίθεται αυτόματα σε λειτουργία.

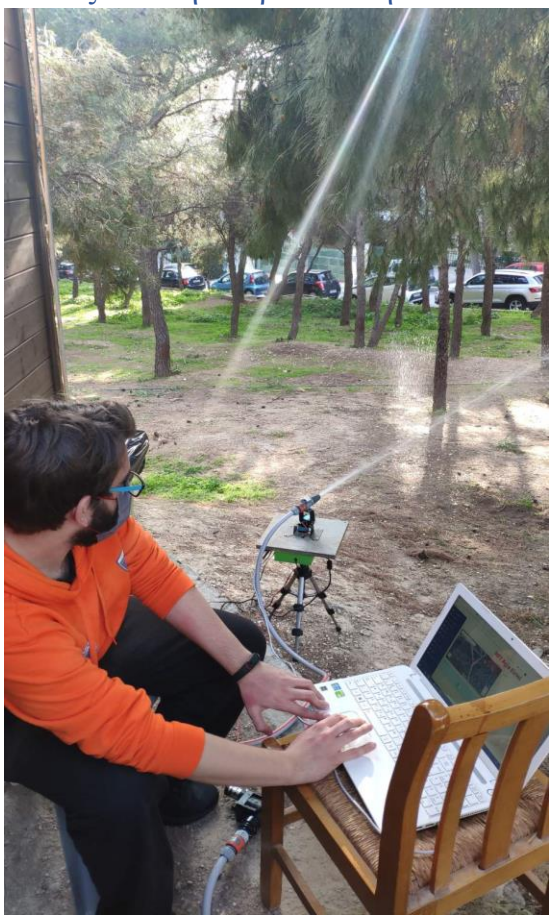
Εντυπωσιακή ήταν η απόσταση βολής νερού και η ταχύτητα αλλαγής πεδίου στόχευσης. Το RFT με τις ιδανικές διαστάσεις και με δεδομένα πραγματικών συνθηκών, συνδεδεμένο με ένα υδραυλικό σύστημα συνεχούς παροχής νερού από γεώτρηση, θα αποτελούσε το ιδανικό σύστημα πυρόσβεσης. Η αποστολή e-mail επίσης σε πραγματικό χρόνο συμβάντος αποτελεί την πιο έγκαιρη και έγκυρη ενημέρωση καθώς υπάρχει οπτικό υλικό για διασταύρωση πληροφορίας. Η προειδοποίηση αυτή ελαχιστοποιεί την πιθανότητα εξάπλωσης μιας δασικής πυρκαγιάς και συμβάλλει στην άμεση και αποτελεσματική αντιμετώπισή της. Επιπλέον η προληπτική διαβροχή ειδικά στις μέρες με υψηλό δείκτη επικινδυνότητας και σε σημεία με μεγάλη καύσιμη ύλη θα μείωνε σημαντικά τις πιθανότητες έναρξης συμβάντων. Θετικό στοιχείο ακόμα ότι το σύστημα αυτό επιδέχεται συνεχείς βελτιώσεις και προσαρμογές ανάλογα με τις ανάγκες και τις ιδιαιτερότητες του κάθε τομέα. Ένας μικρός αριθμός RFT θα μπορούσε να σαρώνει σε 24ωρη βάση έναν μεγάλο τομέα. Η τεχνητή νοημοσύνη δεν σταματά να μας εκπλήσσει!» [6]

Ο κ Νικόλαος Σταμούλους γραμματέας της ΕΟΔΠΕΑ δηλώνει:

«Πρόκειται για ένα αξιόλογο εργαλείο, το οποίο σε πειραματικό στάδιο θέτει τις βάσεις ενός αυτόνομου συστήματος πυρόσβεσης. Θα μπορούσε να αξιοποιηθεί επικουρικά στον δασικό τομέα ευθύνης μας, αν βεβαίως υπήρχε η δυνατότητα χρηματοδότησης για την ανάπτυξη ενός ολοκληρωμένου συστήματος το οποίο να μπορεί να ανταποκριθεί σε πραγματικές συνθήκες ενός συμβάντος.

Η ιδέα του Μιχάλη Χονδρογιάννη , με εντυπωσίασε πολύ. Γνωρίζουμε άλλωστε το δαιμόνιο πνεύμα του ως προς τις τεχνολογικές καινοτομίες και χαιρόμαστε ιδιαίτερα που τον έχουμε στην ομάδα μας. Τέτοια φωτεινά μυαλά έχει ανάγκη η κοινωνία μας.»

Εικόνες από την παρουσίαση του RFT στην ΕΟΔΠΕΑ



Μελλοντική Ερευνά – Βελτίωσης

Το Σύστημα όπως υλοποιήθηκε στην παρούσα διπλωματική δεν είναι τόσο αξιόπιστο και έχει ελάχιστες ρεαλιστικές δυνατότητες όσον αφορά την κατάσβεση.

Ωστόσο, όσον αφορά την βελτίωση του μηχανισμού αναγνώρισης πυρκαγιάς το μοντέλο μηχανικής μάθησης θα πρέπει να εκπαιδευτεί σε πολλές παραπάνω εικόνες από ότι έγινε στην δική μας περίπτωση (3000 εικόνες) για να μπορέσει να ανεβεί το ποσοστό σωστής πρόβλεψης πάνω από 80% το οποίο είναι τώρα.

Επιπρόσθετα για να μπορέσει να είναι ακόμα πιο ακριβής η πρόβλεψη, θα ήταν καλό να φτιαχτεί ένα σύνθετο Σύστημα σε συνεργασίας με την εικόνα από την κανονική κάμερα που θα περνάει από το μοντέλο αναγνώρισης εικόνας μας , να υπάρχει και μια θερμική κάμερα και να δίνουν και τα δυο τιμές και αναλόγως να μπορεί να δοθεί συνδυαστικά μια πιο ακριβής μέτρηση για να μπορέσουμε να αποφύγουμε όσο το δυνατόν γίνεται πιθανότητες να δώσει το Σύστημα « Λάθος Αρνητικό» (false negative), δεδομένου ότι είναι πιο σημαντικό να υπάρχει μια λανθασμένη εκτίμηση για την ύπαρξη φωτιάς και να υπάρξει δράση για την αντιμετώπισή της από το να υπάρχει λανθασμένη εκτίμηση για την μη ύπαρξη φωτιάς ενώ υπάρχει.

Το παραπάνω είναι ένας ακόμα λόγος που μέχρι να τελειοποιηθεί σαν Σύστημα και να μπορέσει να δώσει ένα ποσοστό επιτυχίας στην πρόβλεψη πάνω από 95% δεν θα το θεωρήσω κύριο Σύστημα για την αντιμετώπιση των πυρκαγιών έγκαιρα.

Αυτό βέβαια δεν το κάνει ασήμαντο γιατί ακόμα και να μην μπορέσει να αναγνωρίσει σωστά την πυρκαγιά σε πρώτο χρόνο, θα μπορούσε πολύ πιο ευκολά να γίνει η χρήση του συστήματος Κατάσβεσης ώστε να μπορέσει να ενισχύσει το έργο των πυροσβεστικών οχημάτων.

Σχετικά με το κομμάτι ενημέρωσης του χρήστη στην διπλωματική εργασία υλοποιήθηκε μόνο η ενημέρωση μέσω email, ένα μέσο όχι αρκετά άμεσο για όλους με αποτέλεσμα να χαθεί κρίσιμος χρόνος από την ανίχνευση της φωτιάς μέχρι την αναγνώριση από τον χρήστη αφού δει το email. Για αυτό στο τελικό μοντέλο θα πρέπει να εφαρμοστεί επιπρόσθετο σύστημα για την αποστολή SMS ή την δημιουργία αυτόνομης εφαρμογής για κινητά, tablet και υπολογιστή ώστε να μπορεί να πηγαίνει στον χρήστη μέσω push-notification και να γίνεται πιο άμεση διαχείριση. [6]

Σχετικά με το Σύστημα Κατάσβεσης , θα μπορούσε να υλοποιηθεί επιπρόσθετο Σύστημα που θα λαμβάνει την εικόνα από την κάμερα (είτε κανονική είτε θερμική) και θα ανιχνεύει ποια ακριβώς σημεία έχουν πάρει φωτιά ώστε να μπορεί μετά με την χρήση αλγορίθμου να κάνει αυτοματοποιημένη κατάσβεση είτε σειριακά στην αρχή ή ενδεχομένως και σε δεύτερο χρόνο με βελτιστοποίηση του συστήματος να κάνει «έξυπνη» κατάσβεση ανάλογα με την ένταση της φωτιάς σε κάθε σημείο.

Όσο αφορά την κατασκευή προφανώς θα πρέπει να υλοποιηθεί με διαφορετικά υλικά, τα υλικά θα πρέπει να είναι κατασκευασμένα ώστε να μπορούν να ενσωματωθούν με τα συστήματα της πυροσβεστικής υπηρεσίας. Να γίνει χρήση μεγάλων σωλήνων διότι υπάρχει η δυνατότητα εγκατάστασης του συστήματος αυτού σε σημεία του βουνού που έχει μεταφερθεί αγωγός νερού και συνδέεται είτε με το δίκτυο είτε με αντλιοστάσιο και γεώτρηση όπου υπάρχουν σημεία με εξόδους πυροσβεστικού τύπου (Storz). Με την σύνδεση του RFT σε τέτοιο Σύστημα ύδρευσης θα

μπορεί να έχει άμεση πρόσβαση σε νερό με αρκετή πίεση ώστε να μπορεί να λειτουργεί σε κάποιο επίπεδο χωρίς την χρήση επιπλέον αντλίας γιατί αυτά μπορούν να μεταφέρουν νερό με 8-10 bar πίεση.

Επιπρόσθετα μπορεί να τοποθετηθεί και ανεξάρτητα από το δίκτυο, να δημιουργηθεί δηλαδή ένα σύμπλεγμα από μια μεγάλη δεξαμενή (ιδανικά με δυνατότητα πλήρωσής της μέσω της βροχής) καθώς και ένας μικρός σταθμός που θα μπορεί να εγκατασταθεί αντλία ώστε να μπορεί να γίνεται κατάσβεση με τα βέλτιστα bar για μέγιστη απόσταση και σταθερές επιδόσεις.

Στον χώρο αυτό θα μπορούσε να εγκατασταθεί και ένα Σύστημα με μπαταρίες και φωτοβολταικα ώστε να μπορεί το RFT να είναι εντελώς αυτόνομο τόσο από θέμα ρεύματος όσο και νερού (ειδικά άμα είναι συνδεμένο με το δίκτυο ύδρευσης).

Στην περίπτωση της έκδοσης με την δεξαμενή νερού θα ήταν επίσης καλό να έχει ενσωματωμένο υποσύστημα που να παρακολουθεί και να ενημερώνει το κέντρο διαχειρίσεις των RFT για την ποσότητα νερού μέσα στην δεξαμενή ώστε να μπορεί να προγραμματιστεί ευκολά και έγκαιρα η αναπλήρωσή της .

Η πειραματική κατασκευή τέτοιου συστήματος έχει ήδη υλοποιηθεί από πλευράς μου και έχει περάσει σαν πρόταση για υλοποίηση στα πλαίσια ανάπλασης του Υμηττού μέσω της Ομάδας Εθελοντές Δασοπροστασίας & Πολιτικής Προστασίας Δήμου Ελληνικού Αργυρούπολη.

Σαφέστατα πρέπει να προστεθεί και δυνατότητα για πρόσβαση στο ιντερνέτ ώστε να υπάρξει σωστή λειτουργία και ενημέρωση του χρήστη σε περίπτωση πυρκαγιάς με κάτι τέτοιο να μπορεί να υλοποιηθεί με δυο τρόπους ανάλογα την περίπτωση.

Μια επιλογή είναι ασύρματες κεραίες Wi-Fi σε περίπτωση που υπάρχει η δυνατότητα και οπτική επαφή με κάποια βάση(πυροφυλάκιο) ώστε να έχει πρόσβαση σε σταθερή και αξιόπιστη σύνδεση ιντερνέτ. Σε περίπτωση που δεν είναι αυτό δυνατό μια δεύτερη επιλογή είναι να βασιστούμε στο δίκτυο κινητής τηλεφωνίας και να προσαρμόσουμε στο Σύστημα μας ένα GSM Router με δυνατότητες 4G και πάνω για να υπάρχει ικανοποιητική ταχύτητα ιντερνέτ και να γίνεται μεταφορά της εικόνας από την κάμερα σε πραγματικό χρόνο (ή πολύ μικρή καθυστέρηση <3 δευτερόλεπτα) ώστε να μπορεί ο χρήστης να κάνει αποτελεσματική κατάσβεση.

Τέλος θα ήταν ωφέλιμο να δημιουργηθεί όπως αναφέραμε και στην αρχή ένα σύνολο από τέτοια συστήματα ώστε να μπορεί να γίνει οχύρωση κρίσιμων σημείων, λόγω της περιορισμένης απόστασης που μπορεί ένα τέτοιο Σύστημα να καλύψει. Θα χρειαστεί επιπρόσθετη μελέτη για το που θα έχει βέλτιστη απόδοση. Θα μπορούσε να εγκατασταθεί είτε στις άκρες δασικής έκτασης με τον αστικό ιστό ώστε να είναι η τελευταία άμυνα σε περίπτωση που η πυρκαγιά έχει φτάσει τόσο κοντά στο να απειλούνται κτήρια. Σε αυτή την εκδοχή το Σύστημα μπορεί ευκολά να είναι συνδεδεμένο τόσο σε σταθερή παροχή νερού, ρεύματος αλλά και ιντερνέτ.

Και σε δεύτερη εκδοχή θα μπορούσε να προστατεύει κρίσιμες οδούς που μπορεί να χρειαστούν τα πυροσβεστικά οχήματα για να κινηθούν είτε προς την φωτιά είτε για να απομακρυνθούν από αυτή σε περίπτωση απότομης αλλαγής της έντασης και κατεύθυνσης της φωτιάς παίρνοντας μέσα από ένα ασφαλές δρόμο οπού τα RFT θα έχουν είτε ήδη καταβρέξει και διατηρήσει μακριά την φωτιά ή να έχουν κάνει κατάσβεση την περιοχή γύρω από τον δρόμο και να έχει πλέον κρατηθεί ασφαλής.

Βιβλιογραφία

[1] TensorFlow (2021, October 15). Retrieved October 15 ,2021 from https://www.tensorflow.org

[2] Thinger Io (2021, October 18) Retrieved October 18 ,2021 from https://thinger.io/

[3] Macro Recorder (2021, October 24) Retrieved October 24 ,2021 from https://www.jitbit.com/macro-recorder/

[4] Research Webpage about Smoke Detection for Fire Alarm: Datasets (2021, October 24) Retrieved October 24 ,2021 from http://smoke.ustc.edu.cn/datasets.htm

[5] Intro to TensorFlow for Deep Learning (2021, October 15). Retrieved October 15 ,2021 from https://classroom.udacity.com/courses/ud187

[6] Sending Emails With Python (2021, October 24) Retrieved October 24 ,2021 from https://realpython.com/python-send-email/

[7]Τα Δάση στην Ελλάδα (2021, May 15). Retrieved May 15 ,2021 from http://www.visitgreece.gr/el/nature/forests

[8] Το φαινόμενο των αιφνίδιων αστικών πλημμυρών (Urban Flash Floods) (202, October 7). Retrieved May 24 ,2021 from https://greenapple.gr/2020/10/07/to-φαινόμενο-των-αιφνίδιων-αστικών-πλημ/

[9] Νευρωνικό δίκτυο (2021, October 18) Retrieved October 18 ,2021 from https://el.wikipedia.org/wiki/Νευρωνικό_δίκτυο

[10] Convolutional neural network (2021, October 18) Retrieved October 18 ,2021 from https://en.wikipedia.org/wiki/Convolutional_neural_network

[11] Μηχανική μάθηση(2021, October 19) Retrieved October 19 ,2021 from https://el.wikipedia.org/wiki/Μηχανική_μάθηση

[12]History of Machine Learning(2021, September 17) Retrieved September 17 ,2021 from https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html

[13] Fire Rover (2021, October 18) Retrieved October 18 ,2021 from https://firerover.com/

[14] Convolutional Neural Networks (2017, October 9) Retrieved October 18 ,2021 from https://sites.cc.gatech.edu/~san37/post/dlhc-cnn/

[15] Programming Tensor Cores in CUDA 9 (2017, October 17) Retrieved October 18 ,2021 from

https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/

Βιβλιογραφία Εικόνων

1) FreeLogoDesign (2021, October 18) Retrieved October 18 ,2021 from https://www.freelogodesign.org/

2) Programming Tensor Cores in CUDA 9 (2017, October 17) Retrieved October 18 ,2021 from https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/

3) Νευρωνικό δίκτυο (2021, October 18) Retrieved October 18 ,2021 from https://el.wikipedia.org/wiki/Νευρωνικό_δίκτυο

4) Neural Networks Bias And Weights (2019, May 18) Retrieved October 18 ,2021 from https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da

5,6) Νευρωνικό δίκτυο (2021, October 18) Retrieved October 18 ,2021 from https://el.wikipedia.org/wiki/Νευρωνικό_δίκτυο

7) Convolutional Neural Networks (CNNs / ConvNets) (2021, October 18) Retrieved October 18 ,2021 from https://cs231n.github.io/convolutional-networks/

8-9) Convolutional Neural Networks (2017, October 9) Retrieved October 18 ,2021 from https://sites.cc.gatech.edu/~san37/post/dlhc-cnn/

10) A Comprehensive Guide to Convolution Neural Network (2020, November 20) Retrieved October 18 ,2021 from https://miro.medium.com/max/2510/1*vkQ0hXDaQv57sALXAJquxA.jpeg

11)Fire Systems (2021, October 18) Retrieved October 18 ,2021 from https://firesystems.net/

12) Γενική Γραμματεία Πολιτικής Προστασίας (2020, July 23) Retrieved October 18 ,2021 from https://www.civilprotection.gr/

19) Fritzing (2021, October 18) Retrieved October 18 ,2021 from https://fritzing.org/

32) Macro Recorder (2021, October 24) Retrieved October 24 ,2021 from https://www.macrorecorder.com/

Παραρτήματα

Model Creation and Save

Εισαγωγή πακέτων

In [2]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [51]:
```
import os
import time
import numpy as np
import matplotlib.pylab as plt

import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_datasets as tfds
tfds.disable_progress_bar()

from tensorflow.keras import layers
import PIL.Image as Image
```

In [ ]:
```
import email, smtplib, ssl

from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```

TRANSFER LEARNING

In [53]:
```
BATCH_SIZE = 32
IMG_SHAPE  = 224
```

In [54]:
```
URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
feature_extractor = hub.KerasLayer(URL,
                                   input_shape=(IMG_SHAPE, IMG_SHAPE,3))
```

In [55]:
```
feature_extractor.trainable = False
```

Φόρτωση δεδομένων

In [56]:
```
zip_dir = "C:/Users/Calimancil/Downloads/python things/dataset"
```

Ο διαχωρισμός τον φακέλων που έχουμε αποθήκευση της δυο κατηγορίες εικόνων (φωτιά και όχι φωτιά) έχει την παρακάτω μορφή σαν δέντρο.

|__ **train**

|__ **smoke**: [smoke.0.jpg, smoke.1.jpg, smoke.2.jpg ....]

|__ **nosmoke**: [nosmoke.0.jpg, nosmoke.1.jpg, nosmoke.2.jpg ...]

|__ **validation**

|___ **smoke**: [smoke.1000.jpg, smoke.1001.jpg, smoke.1002.jpg ....]

|___ **nosmoke**: [nosmoke.1000.jpg, nosmoke.1001.jpg, nosmoke.1002.jpg ...]

Στην συνείχα συνδικάζουμε το αρχικό path για το dataset μας και προσθέτουμε το κομμάτι για το train και το validation.

In [57]:
```
base_dir = os.path.join(os.path.dirname(zip_dir), 'dataset')
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
```

In [58]:
```
train_smoke_dir = os.path.join(train_dir, 'smoke')  # Path με της εικόνες εκπαίδευση πο
train_nosmoke_dir = os.path.join(train_dir, 'nosmoke')  # Path με της εικόνες εκπαίδευσ
validation_smoke_dir = os.path.join(validation_dir, 'smoke')  # Path με της εικόνες αξι
validation_nosmoke_dir = os.path.join(validation_dir, 'nosmoke')  #Path με της εικόνες
```

Έλεγχος και επισκόπηση τον δεδομένων.

In [59]:
```
num_smoke_tr = len(os.listdir(train_smoke_dir))
num_nosmoke_tr = len(os.listdir(train_nosmoke_dir))

num_smoke_val = len(os.listdir(validation_smoke_dir))
num_nosmoke_val = len(os.listdir(validation_nosmoke_dir))

total_train = num_smoke_tr + num_nosmoke_tr
total_val = num_smoke_val + num_nosmoke_val
```

In [60]:
```
print('total training smoke images:', num_smoke_tr)
print('total training nosmoke images:', num_nosmoke_tr)

print('total validation smoke images:', num_smoke_val)
print('total validation nosmoke images:', num_nosmoke_val)
print("--")
print("Total training images:", total_train)
print("Total validation images:", total_val)
```

```
total training smoke images: 750
total training nosmoke images: 750
total validation smoke images: 249
total validation nosmoke images: 249
--
Total training images: 1500
Total validation images: 498
```

Παραμετροποίηση δεδομένων

Η υπερφόρτωση(Overfitting) συμβαίνει συχνά όταν έχουμε έναν μικρό αριθμό παραδειγμάτων εκπαίδευσης. Ένας τρόπος για να διορθώσετε αυτό το πρόβλημα είναι να αυξήσετε το σύνολο δεδομένων μας έτσι ώστε να έχει επαρκή αριθμό και ποικιλία παραδειγμάτων εκπαίδευσης.

Η αύξηση δεδομένων ακολουθεί την προσέγγιση της δημιουργίας περισσότερων εκπαιδευτικών δεδομένων από υπάρχοντα εκπαιδευτικά δείγματα, αυξάνοντας τα δείγματα μέσω τυχαίων μετασχηματισμών που αποδίδουν πιστευτές εικόνες. Ο στόχος είναι ότι κατά την προπόνηση, το μοντέλο σας δεν θα δει ποτέ την ίδια εικόνα δύο φορές. Αυτό εκθέτει το μοντέλο σε περισσότερες πτυχές των δεδομένων, επιτρέποντάς του να γενικεύσει καλύτερα.

Στο Keras μπορούμε να το εφαρμόσουμε χρησιμοποιώντας συνάρτηση ImageDataGenerator.Με την χρήση αυτού μπορούμε να περάσουμε διαφορετικούς μετασχηματισμούς που θα θέλαμε στο σύνολο δεδομένων μας όπως την περιστροφή , καθρέφτισμα εστίαση(zoom) και αλλά .

In [61]:
```
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()
```

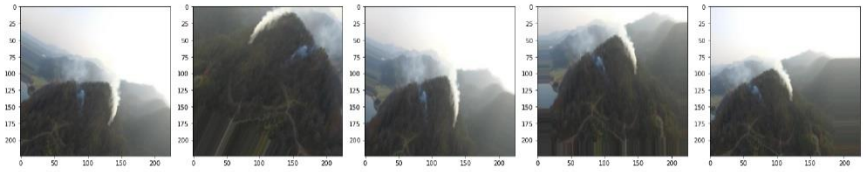
In [62]:
```
image_gen_train = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

train_data_gen = image_gen_train.flow_from_directory(batch_size=BATCH_SIZE,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_SHAPE,IMG_SHAPE),
                                                     class_mode='binary')
```

```
Found 1500 images belonging to 2 classes.
```

Στην συνέχεια θα εξάγουμε ένα δείγμα από μια εικόνα μας για να δούμε πως έχει αλλάξει μετά από όλες τις παραλλαγές που έχουμε εφαρμοστή στα δεδομένα μας.

In [63]:
```
augmented_images = [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmented_images)
```



Δημιουργία δεδομένων επικύρωσης σε παρτίδες.

Συνήθως, εφαρμόζουμε μόνο αύξηση δεδομένων σε παραδείγματα εκπαίδευσης, επειδή η αρχική εικόνα πρέπει να αντιπροσωπεύει αυτό που χρειάζεται το μοντέλο μας για τη διαχείριση.

Επομένως, σε αυτήν την περίπτωση, διαγράφουμε μόνο τις εικόνες επαλήθευσης και στη συνέχεια χρησιμοποιούμε το ImageDataGenerator για να τις μετατρέψουμε σε παρτίδες.

In [ ]:
```
image_gen_val = ImageDataGenerator(rescale=1./255)

val_data_gen = image_gen_val.flow_from_directory(batch_size=BATCH_SIZE,
                                                 directory=validation_dir,
                                                 target_size=(IMG_SHAPE, IMG_SHAPE),
                                                 class_mode='binary')
```

Δημιουργία μοντέλου αναγνώρισης

Το μοντέλο αποτελείται από τέσσερα μπλοκ Συνέλιξης με ένα μέγιστο επίπεδο συγκέντρωσης σε καθένα από αυτά.

Πριν από τα τελικά επίπεδα Dense, εφαρμόζουμε επίσης πιθανότητα πτώσης 0,5. Αυτό σημαίνει ότι το 50% των τιμών που εισέρχονται στο επίπεδο Dropout θα οριστεί στο μηδέν. Αυτό βοηθά στην αποφυγή του Overfitting. Στη συνέχεια έχουμε ένα πλήρως συνδεδεμένο στρώμα με 512 μονάδες, με λειτουργία ενεργοποίησης «relu». Το μοντέλο θα παράγει πιθανότητες κλάσης για δύο κατηγορίες - καπνός και όχι καπνός - χρησιμοποιώντας το "softmax".

In [65]:
```
model = tf.keras.Sequential([
  feature_extractor,
  layers.Dense(2)
])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 2562
=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

Σύνταξη του μοντέλου

Θα χρησιμοποιήσουμε το εργαλείο βελτιστοποίησης «adam». Λόγο του ότι έχουμε κατηγοριοποίηση softmax, θα χρησιμοποιήσουμε το «sparse_categorical_crossentropy» ως τη λειτουργία απώλειας. Θα θέλαμε επίσης να εξετάσουμε την ακρίβεια εκπαίδευσης και επικύρωσης σε κάθε εποχή καθώς εκπαιδεύουμε το δίκτυό μας.

In [67]:
```
model.compile(
  optimizer='adam',
  loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])

EPOCHS = 3
```

Εκπαίδευση Μοντέλου

In [68]:
```
history = model.fit(train_data_gen,
                    epochs=EPOCHS,
                    validation_data=val_data_gen)
```

```
Epoch 1/3
47/47 [==============================] - 120s 3s/step - loss: 0.2336 - accuracy: 0.8907
- val_loss: 0.0742 - val_accuracy: 0.9739
Epoch 2/3
47/47 [==============================] - 116s 2s/step - loss: 0.0743 - accuracy: 0.9747
- val_loss: 0.0607 - val_accuracy: 0.9980
Epoch 3/3
47/47 [==============================] - 123s 3s/step - loss: 0.0598 - accuracy: 0.9880
- val_loss: 0.0430 - val_accuracy: 0.9980
```

Εμφάνιση αποτελεσμάτων Accuracy και Loss

In [70]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

In [20]:
```
train_data_gen.class_indices #Seeing what class is what"
```

Out[20]:
```
{'nosmoke': 0, 'smoke': 1}
```

Φόρτωση νέας εικόνας για δοκιμή

In [72]:
```
input_image = 'C:/Users/Calimancil/Downloads/fire3.jpg'
image_path=input_image
input_image = Image.open(input_image).resize((IMG_SHAPE, IMG_SHAPE))
input_image
```

Out[72]:

In [73]:
```
input_image = np.array(input_image)/255.0
input_image.shape
```

Out[73]:
```
(224, 224, 3)
```

In [105…
```
result = model.predict(input_image[np.newaxis, ...])
predicted_class = np.argmax(result[0], axis=-1)
```

Έλεγχος αποτελέσματος και ενημέρωση χρήστη

In [91]:
```
if predicted_class==0:
    print('No Smoke')
else:
    print('Smoke')
    subject = "Smoke Or Fire Detected!"
    body = "Smoke Or Fire Detected! at Agia Eirini"
    sender_email = "calimancil@gmail.com"
    receiver_email = "calimancil1995@hotmail.com"
    password = "*********"

# Ορισμός στοιχείων του email
    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = receiver_email
    message["Subject"] = subject

# Προσθηκη του σωματος στο mail
    message.attach(MIMEText(body, "plain"))

    filename = image_path # In same directory as script

# Άνοιγμα αρχείου σε διάδικη μορφή
    with open(filename, "rb") as attachment:
        part = MIMEBase("application", "octet-stream")
        part.set_payload(attachment.read())

# Κωδικοποιήσει του αρχείου με χαρακτήρες ASCII για αποστολή μέσω email
    encoders.encode_base64(part)

# Προσθήκη κεφαλίδας ως ζεύγος κλειδιού / τιμής στο συνημμένο
    part.add_header(
        "Content-Disposition",
        f"attachment; filename= {filename}",
    )

# Προσθήκη της εικόνας σαν συνημμένο και επεξεργασία κειμένου σαν string
    message.attach(part)
    text = message.as_string()

# Σύνδεση στο server Gmail και αποστολή του Email
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, text)
```

```
Smoke
```

Αποθήκευση μοντέλου ως Keras Layer .h5

In [84]:
```
t = time.time()
```

```
export_path_keras = "./{}.h5".format(int(t))
print(export_path_keras)

model.save(export_path_keras)
```

```
./1608376383.h5
```

Επαναφόρτωση αποθηκευμένου μοντέλου

In [85]:
```
reloaded = tf.keras.models.load_model(
  export_path_keras,
  custom_objects={'KerasLayer': hub.KerasLayer})

reloaded.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 2562
=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

Έλεγχος επαναφορτωμένου μοντέλου

In [94]:
```
result_batch = model.predict(input_image[np.newaxis, ...])
reloaded_result_batch = reloaded.predict(input_image[np.newaxis, ...])
```

In [95]:
```
(abs(result_batch - reloaded_result_batch)).max()
```

Out[95]:
```
0.0
```

In [114…
```
image_batch=input_image[np.newaxis, ...]
```

Αποθήκευση μοντέλου σαν Saved Model

In [116…
```
t = time.time()

export_path_sm = "./{}".format(int(t))
print(export_path_sm)
tf.saved_model.save(model, export_path_sm)
```

```
./1608378130
INFO:tensorflow:Assets written to: ./1608378130\assets
INFO:tensorflow:Assets written to: ./1608378130\assets
```

Φόρτωση του saved model

In [117…
```
reload_sm_keras = tf.keras.models.load_model(
  export_path_sm,
  custom_objects={'KerasLayer': hub.KerasLayer})
```

```
reload_sm_keras.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 2562
=================================================================
Total params: 2,260,546
Trainable params: 2,226,434
Non-trainable params: 34,112
_________________________________________________________________
```

Έλεγχος αν το επαναφορτισμένο μοντέλο είναι ίδιο

In [118…
```
result_batch = model.predict(image_batch)
reload_sm_keras_result_batch = reload_sm_keras.predict(image_batch)
```

In [119…
```
(abs(result_batch - reload_sm_keras_result_batch)).max()
```

Out[119… `0.0`

Load Saved Model and Retrain Code

Εισαγωγή πακέτων

In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import time
import numpy as np
import matplotlib.pylab as plt

import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_datasets as tfds
tfds.disable_progress_bar()

from tensorflow.keras import layers
import PIL.Image as Image
import email, smtplib, ssl

from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```

Φόρτωση αποθηκευμένου μοντέλου

In [2]:
```
model = tf.keras.models.load_model(
 "C:/Users/Calimancil/Downloads/python things/1608378130",
  custom_objects={'KerasLayer': hub.KerasLayer})

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 2562
=================================================================
Total params: 2,260,546
Trainable params: 2,226,434
Non-trainable params: 34,112
_________________________________________________________________
```

Προσθήκη επιπλέον επίπεδων στο Keras για αποφυγή του Overfitting

In [5]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
```

```
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2)
])
```

In [6]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               9437696
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 1026
=================================================================
Total params: 9,679,554
Trainable params: 9,679,554
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
IMG_SHAPE  = 224
BATCH_SIZE=32
```

Επανεκπαίδευση

In [44]:
```
zip_dir = "C:/Users/Calimancil/Downloads/python things/"
base_dir = os.path.join(os.path.dirname(zip_dir), 'firedataset')
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

train_smoke_dir = os.path.join(train_dir, 'smoke')  # Path με της εικόνες εκπαίδευση πο
train_nosmoke_dir = os.path.join(train_dir, 'nosmoke')  # Path με της εικόνες εκπαίδευσ
validation_smoke_dir = os.path.join(validation_dir, 'smoke')  # Path με της εικόνες αξι
validation_nosmoke_dir = os.path.join(validation_dir, 'nosmoke')  #Path με της εικόνες
```

In [45]:
```
num_smoke_tr = len(os.listdir(train_smoke_dir))
num_nosmoke_tr = len(os.listdir(train_nosmoke_dir))
```

```
num_smoke_val = len(os.listdir(validation_smoke_dir))
num_nosmoke_val = len(os.listdir(validation_nosmoke_dir))

total_train = num_smoke_tr + num_nosmoke_tr
total_val = num_smoke_val + num_nosmoke_val
```

In [46]:
```
print('total training smoke images:', num_smoke_tr)
print('total training nosmoke images:', num_nosmoke_tr)

print('total validation smoke images:', num_smoke_val)
print('total validation nosmoke images:', num_nosmoke_val)
print("--")
print("Total training images:", total_train)
print("Total validation images:", total_val)
```

```
total training smoke images: 94
total training nosmoke images: 445
total validation smoke images: 38
total validation nosmoke images: 96
--
Total training images: 539
Total validation images: 134
```

In [47]:
```
image_gen_train = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

train_data_gen = image_gen_train.flow_from_directory(batch_size=BATCH_SIZE,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_SHAPE,IMG_SHAPE),
                                                     class_mode='binary')
```

```
Found 539 images belonging to 2 classes.
```

In [48]:
```
image_gen_val = ImageDataGenerator(rescale=1./255)

val_data_gen = image_gen_val.flow_from_directory(batch_size=BATCH_SIZE,
                                                 directory=validation_dir,
                                                 target_size=(IMG_SHAPE, IMG_SHAPE),
                                                 class_mode='binary')
```

```
Found 134 images belonging to 2 classes.
```

In [50]:
```
model.compile(
  optimizer='adam',
  loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])

EPOCHS = 10
```

In [51]:
```
history = model.fit(train_data_gen,
                    epochs=EPOCHS,
                    validation_data=val_data_gen)
```

```
Epoch 1/10
17/17 [==============================] - 104s 6s/step - loss: 0.3839 - accuracy: 0.8404
- val_loss: 0.4703 - val_accuracy: 0.7090
Epoch 2/10
17/17 [==============================] - 109s 6s/step - loss: 0.3061 - accuracy: 0.8516
- val_loss: 0.4792 - val_accuracy: 0.7388
Epoch 3/10
17/17 [==============================] - 99s 6s/step - loss: 0.2722 - accuracy: 0.8794 -
val_loss: 0.4624 - val_accuracy: 0.7910
Epoch 4/10
17/17 [==============================] - 92s 5s/step - loss: 0.2708 - accuracy: 0.9054 -
val_loss: 0.4795 - val_accuracy: 0.7687
Epoch 5/10
17/17 [==============================] - 99s 6s/step - loss: 0.2514 - accuracy: 0.8887 -
val_loss: 0.5227 - val_accuracy: 0.8060
Epoch 6/10
17/17 [==============================] - 77s 5s/step - loss: 0.2653 - accuracy: 0.8850 -
val_loss: 0.3575 - val_accuracy: 0.8209
Epoch 7/10
17/17 [==============================] - 83s 5s/step - loss: 0.2691 - accuracy: 0.8738 -
val_loss: 0.4270 - val_accuracy: 0.7761
Epoch 8/10
17/17 [==============================] - 79s 5s/step - loss: 0.2478 - accuracy: 0.8980 -
val_loss: 0.3757 - val_accuracy: 0.8284
Epoch 9/10
17/17 [==============================] - 78s 5s/step - loss: 0.2652 - accuracy: 0.8868 -
val_loss: 0.6127 - val_accuracy: 0.7687
Epoch 10/10
17/17 [==============================] - 78s 5s/step - loss: 0.2805 - accuracy: 0.8961 -
val_loss: 0.3717 - val_accuracy: 0.8060
```

In [52]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Επανέλεγχος πρόβλεψης

In [96]:
```
input_image = 'C:/Users/Calimancil/Downloads/fire3.jpg'
image_path=input_image
input_image = Image.open(input_image).resize((IMG_SHAPE, IMG_SHAPE))
input_image
```

Out[96]:

In [97]:
```
input_image = np.array(input_image)/255.0
input_image.shape
```

Out[97]:
```
(224, 224, 3)
```

In [98]:
```
result = model.predict(input_image[np.newaxis, ...])
```

```
predicted_class = np.argmax(result[0], axis=-1)
```

In [99]:
```
if predicted_class==0:
    print('No Smoke')
else:
    print('Smoke')
    subject = "Smoke Or Fire Detected!"
    body = "Smoke Or Fire Detected! at Agia Eirini"
    sender_email = "calimancil@gmail.com"
    receiver_email = "calimancil1995@hotmail.com"
    password = "*********"

# Ορισμός στοιχείων του email
    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = receiver_email
    message["Subject"] = subject

# Προσθηκη του σωματος στο mail
    message.attach(MIMEText(body, "plain"))

    filename = image_path # In same directory as script

# Άνοιγμα αρχείου σε διάδικη μορφή
    with open(filename, "rb") as attachment:
        part = MIMEBase("application", "octet-stream")
        part.set_payload(attachment.read())

# Κωδικοποιήσει του αρχείου με χαρακτήρες ASCII για αποστολή μέσω email
    encoders.encode_base64(part)

# Προσθήκη κεφαλίδας ως ζεύγος κλειδιού / τιμής στο συνημμένο
    part.add_header(
        "Content-Disposition",
        f"attachment; filename= {filename}",
    )

# Προσθήκη της εικόνας σαν συνημμένο και επεξεργασία κειμένου σαν string
    message.attach(part)
    text = message.as_string()

# Σύνδεση στο server Gmail και αποστολή του Email
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, text)
```

```
No Smoke
```

Αποθήκευση του νέου μοντέλου

In [ ]:
```
t = time.time()

export_path_sm = "./{}".format(int(t))
print(export_path_sm)
tf.saved_model.save(model, export_path_sm)
```

Final Python Script For Fire Check

```
#!/usr/bin/env python
# coding: utf-8

# In[11]:


from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import time
import numpy as np
import matplotlib.pylab as plt
import sched
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_datasets as tfds
tfds.disable_progress_bar()

from tensorflow.keras import layers
import PIL.Image as Image
import email, smtplib, ssl

from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import datetime

from time import ctime
import shutil

# In[2]:


model = tf.keras.models.load_model(
 "C:/Users/Calimancil/Downloads/final_files_fire/1608667219",
  custom_objects={'KerasLayer': hub.KerasLayer})

#model.summary()


# In[3]:


IMG_SHAPE  = 224


# In[8]:


def checkfire():
    timenow= datetime.datetime.now().strftime("%Y%m%d%H%M%S")
    input_image = 'C:/Users/Calimancil/Downloads/picture.jpg'

    smoke_dest='C:/Users/Calimancil/Downloads/final_files_fire/smoke/picture{}.jpg'.forma
    t(int(timenow))

    nosmoke_dest='C:/Users/Calimancil/Downloads/final_files_fire/nosmoke/picture{}.jpg'.f
    ormat(int(timenow))
    image_path=input_image
    input_image = Image.open(input_image).resize((IMG_SHAPE, IMG_SHAPE))
    input_image
    input_image = np.array(input_image)/255.0
    input_image.shape
    result = model.predict(input_image[np.newaxis, ...])
    predicted_class = np.argmax(result[0], axis=-1)
    if predicted_class==0:
        print('No Smoke Deteced')
```

```
64          print("File saved in")
65          shutil.copy2(image_path,nosmoke_dest)
66          print("File saved in: "+nosmoke_dest)
67      else:
68          print('Smoke Deteced!! User Notified by Mail')
69          shutil.copy2(image_path,smoke_dest)
70          print("File saved in: "+smoke_dest)
71          subject = "Smoke Or Fire Detected!"
72          body = "Smoke Or Fire Detected! at Agia Eirini Login Here :
            https://console.thinger.io/#!/console/dashboards/RFTCONTROL Username:calimancil
            Pass:arc170!"
73          sender_email = "calimancil@gmail.com"
74          receiver_email = "calimancil1995@hotmail.com"
75          password = "the5elemenets!"
76
77  # Create a multipart message and set headers
78          message = MIMEMultipart()
79          message["From"] = sender_email
80          message["To"] = receiver_email
81          message["Subject"] = subject
82          message["Bcc"] = receiver_email  # Recommended for mass emails
83
84  # Add body to email
85          message.attach(MIMEText(body, "plain"))
86
87          filename = image_path # In same directory as script
88
89  # Open PDF file in binary mode
90          with open(filename, "rb") as attachment:
91      # Add file as application/octet-stream
92      # Email client can usually download this automatically as attachment
93              part = MIMEBase("application", "octet-stream")
94              part.set_payload(attachment.read())
95
96  # Encode file in ASCII characters to send by email
97          encoders.encode_base64(part)
98
99  # Add header as key/value pair to attachment part
100         part.add_header(
101             "Content-Disposition",
102             f"attachment; filename= {filename}",
103         )
104
105 # Add attachment to message and convert message to string
106         message.attach(part)
107         text = message.as_string()
108
109 # Log in to server using secure context and send email
110         context = ssl.create_default_context()
111         with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
112             server.login(sender_email, password)
113             server.sendmail(sender_email, receiver_email, text)
114
115
116 # In[10]:
117
118
119 starttime = time.time()
120 while True:
121     checkfire()
122     ctime()
123     time.sleep(60.0 - ((time.time() - starttime) % 60.0))
124
125
126
127
128
```

Macro Recorder Steps

Command			
DELAY	15000		
KEYBOARD	KeyPress	"Enter"	
LABEL	LOOP		
REPEAT	1		
DELAY	3000		
MOUSE	RightClick	X = 284	Y = 314
DELAY	2000		
MOUSE	Click	X = 310	Y = 350
DELAY	1500		
MOUSE	Click	X = 250	Y = 208
MOUSE	Click	X = 250	Y = 208
DELAY	2000		
MOUSE	Click	X = 736	Y = 378
DELAY	2000		
MOUSE	Click	X = 1345	Y = 697
DELAY	20000		
KEYBOARD	KeyPress	"F5"	
DELAY	2000		
GOTO	LOOP		

Εικόνα 32 Εντολές Macro Recorder για λήψη εικόνας από την Κάμερα

Chrome-MacroRecorder Bat File

start chrome http://192.168.1.64/ISAPI/Streaming/channels/101/picture --start-maximized

start "" "C:\Users\Calimancil\Downloads\python things\imagesave" /a

Activate Conda Bat

```
@REM Copyright (C) 2012 Anaconda, Inc
@REM SPDX-License-Identifier: BSD-3-Clause
@REM Test first character and last character of %1 to see if first character is a "
@REM   but the last character isn't.
@REM This was a bug as described in https://github.com/ContinuumIO/menuinst/issues/60
@REM When Anaconda Prompt has the form
@REM   %windir%\system32\cmd.exe "/K" "C:\Users\builder\Miniconda3\Scripts\activate.bat" "C:\Users\builder\Miniconda3"
@REM Rather than the correct
@REM    %windir%\system32\cmd.exe /K ""C:\Users\builder\Miniconda3\Scripts\activate.bat" "C:\Users\builder\Miniconda3""
@REM this solution taken from https://stackoverflow.com/a/31359867
@set "_argsl=%1"
@set _argsl_first=%_argsl:~0,1%
@set _argsl_last=%_argsl:~-1%
@set _argsl_first=%_argsl_first:"=+%
@set _argsl_last=%_argsl_last:"=+%
@set _argsl=

@if "%_argsl_first%"=="+" if NOT "%_argsl_last%"=="+" (
    @CALL "%~dp0..\condabin\conda.bat" activate
    @GOTO :End
)

@REM This may work if there are spaces in anything in %*
@CALL "%~dp0..\condabin\conda.bat" activate %*

:End
@set _argsl_first=
@set _argsl_last=
@CALL python C:\Users\Calimancil\Downloads\final_files_fire\firecheck.py
```

Εικόνα 33 Τροποποίηση ενεργοποίησης του Conda για εκκίνηση με εάν πάτημα

Runanacoda Bat File

call "C:\Users\Calimancil\Downloads\python things\chrome-macro.bat"

CD C:\Users\Calimancil\Downloads\

%windir%\System32\cmd.exe "/K" C:\Users\Calimancil\anaconda3\Scripts\activateConda.bat base

Κώδικας του Wemos (Σύστημα Κίνησης – Κατάσβεσης)

```
#include <ThingerESP8266.h>
#include <Servo.h>
#define USERNAME "calimancil"
#define DEVICE_ID "RFT1"
#define DEVICE_CREDENTIAL "QnN&xdQihAZt"

#define SSID "Hogwarts2.4"
#define SSID_PASSWORD "alohomora"

ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);


// defines pins numbers
#define SERVOX_PIN D6
#define SERVOY_PIN D3
#define pumpON D1
#define MAIN D2
#define pumpOFF D7
int servoposX;
Servo servoX;
int servoposY;
Servo servoY;
bool currentState;
void setup() {
  Serial.begin(9600);
  pinMode(pumpON, OUTPUT);
  pinMode(MAIN, OUTPUT);
  pinMode(pumpOFF, OUTPUT);
  servoX.attach(SERVOX_PIN);
  servoY.attach(SERVOY_PIN);
  thing.add_wifi(SSID, SSID_PASSWORD);
  digitalWrite(MAIN,HIGH);

 // controlling servo position
thing["servoposX"] << [](pson& in){
    if(in.is_empty())
      in = (int)servoposX;
    else
    {
      servoposX = (int)in;
      servoX.write(servoposX);
      Serial.println(servoposX);
    }
    servoposX = in;
};
thing["servoposY"] << [](pson& in){
    if(in.is_empty())
      in = (int)servoposY;
    else
    {
      servoposY = (int)in;
      servoY.write(servoposY);
      Serial.println(servoposY);
    }
```

```
    servoposY = in;
};
//Waterpumb control
thing["WATER"] << [](pson& in){
  if(in.is_empty()){
      in = (bool) digitalRead(pumpON);
      }
    else{
      digitalWrite(MAIN,LOW);
      digitalWrite(pumpON, in ? HIGH : LOW);
      digitalWrite(pumpOFF, in ? LOW : HIGH);
      delay(150);
      digitalWrite(MAIN,HIGH);
      }
    };
  thing["MANUAL"] << [](pson& in)
  // resource output example (i.e. reading a sensor value)
  thing["millis"] >> outputValue(millis());
}

void loop() {
  thing.handle();
}
```

User Interface (Thinger.io Dashboard)

Εικόνα 34 Σύνολο γραφιστικής απεικόνισης της διεπαφής του χρήστη μέσω Thinger.io