ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Τεχνητή Νοημοσύνη στο Άκρο: Ενδυναμώνοντας τα Ενσωματωμένα Συστήματα για Έξυπνα Περιβάλλοντα

Συγγραφέας: Κωνσταντίνος Μπουρλέσης

ΑΜ: 71343862

Επιβλέπων: Αν. Καθ. Παναγιώτης Καρκαζής

Αθήνα, Σεπτέμβριος 2024

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF INFORMATICS & COMPUTER ENGINEERING

Diploma Thesis

Artificial Intelligence at the Edge: Empowering Embedded Systems for Smart Environments

Student name and surname: Konstantinos Bourlesis

Registration Number: 71343862

Supervisor name and surname: Associate Professor Panagiotis Karkazis

Athens, September 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνητή Νοημοσύνη στο Άκρο: Ενδυναμώνοντας τα Ενσωματωμένα Συστήματα για Έξυπνα Περιβάλλοντα

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Καρκαζής Παναγιώτης	Αναπληρωτής Καθηγητής	PANAGIOTIS KARKAZIS Digitally signed by PANAGIOTIS KARKAZIS Date: 2024.10.15 11:35:41 +03'00'
2	Νικόλαος Μυριδάκης	Αναπληρωτής Καθηγητής	Nikolaos Myridakis Digitally signed by Nikolaos Myridakis Date: 2024.10.15 11:44:51 +03'00'
3	Κωνσταντίνος Μαυρομμάτης	Λέκτορας	KONSTANTINOS MAVROMMATIS Digitally signed by KONSTANTINOS MAVROMMATIS Date: 2024.10.16 19:52:07 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Μπουρλέσης Κωνσταντίνος του Παναγιώτη, με αριθμό μητρώου 71343862 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/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».

**Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή*

Ο Δηλών

*** Ονοματεπώνυμο /Ιδιότητα** (Υπογραφή)

Ψηφιακή Υπογραφή Επιβλέποντα

**** Σε εξαιρετικές περιπτώσεις και μετά από αιτιολόγηση και έγκριση του επιβλέποντα, προβλέπεται χρονικός περιορισμός πρόσβασης (embargo) 6-12 μήνες. Στην περίπτωση αυτή θα πρέπει να υπογράψει ψηφιακά ο/η επιβλέπων/ουσα καθηγητής/τρια, για να γνωστοποιεί ότι είναι ενημερωμένος/η και συναινεί. Οι λόγοι χρονικού αποκλεισμού πρόσβασης περιγράφονται αναλυτικά στις πολιτικές του Ι.Α. (σελ. 6):***

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό του υπολογισμού στο άκρο χρησιμοποιώντας ενσωματωμένα συστήματα και τεχνητή νοημοσύνη. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω. Ευχαριστώ πολύ την μητέρα μου, γιατί χωρίς εκείνη δεν θα είχα την ευκαιρία να αποκτήσω αυτό το πτυχίο, καθώς και τους φίλους-συμφοιτητές μου, που ήταν δίπλα μου κατά την διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Αυτή η διπλωματική διατριβή διερευνά την ενοποίηση διαφόρων τεχνολογιών, συμπεριλαμβανομένης της εικονικοποίησης, του υπολογιστικού νέφους και άκρου, των ενσωματωμένων συστημάτων και της τεχνητής νοημοσύνης με έμφαση στη μηχανική μάθηση. Ξεκινά με μια εισαγωγή στην εικονικοποίηση, περιγράφοντας λεπτομερώς την ιστορική εξέλιξη, τους τύπους και τις τεχνικές της. Στη συνέχεια, εμβαθύνει στο νέφος και στην υπολογιστική στο άκρο, συζητώντας τα χαρακτηριστικά, τα μοντέλα ανάπτυξης και τα οφέλη τους. Η ενότητα για τα ενσωματωμένα συστήματα υπογραμμίζει την εξέλιξη, τις περιπτώσεις χρήσης και τις αρχιτεκτονικές τους. Ενώ η ενότητα της τεχνητής νοημοσύνης και μηχανικής μάθησης περιγράφει βασικές εξελίξεις, δημοφιλείς αλγόριθμους και την ενσωμάτωση της μηχανικής μάθησης στα ενσωματωμένα συστήματα. Στο τελευταίο κεφάλαιο παρουσιάζεται η ανάπτυξη ενός αυτόνομου οχήματος που μέσω ενός συνελικτικού νευρωνικού δικτύου αναγνωρίζει και κινείται ανάλογα με τις πινακίδες κυκλοφορίας. Παρέχει επίσης, πρακτικές γνώσεις για τη δημιουργία και την εκπαίδευση μοντέλων μηχανικής μάθησης, χρησιμοποιώντας πλατφόρμες όπως το Arduino και το Raspberry Pi. Τέλος, η διατριβή ολοκληρώνεται με ευρήματα και βελτιστοποιήσεις, προσφέροντας συστάσεις για μελλοντικές βελτιώσεις στην ανάπτυξη και την αποτελεσματικότητα του συστήματος.

Επιστημονική Περιοχή: Υπολογιστική στο άκρο

Λέξεις Κλειδιά: εικονικοποίηση, ενσωματωμένα συστήματα, νεφοϋπολογιστική, εικονικοί πόροι, μηχανική μάθηση, τεχνητή νοημοσύνη, διαδίκτυο των αντικειμένων

ABSTRACT

This thesis explores the integration of various technologies, including virtualization, cloud and edge computing, embedded systems, and artificial intelligence with an emphasis on machine learning. It begins with an introduction to virtualization, detailing its historical development, types, and techniques. It then delves into cloud and edge computing, discussing their features, deployment models, and benefits. The embedded systems section highlights their evolution, use cases, and architectures. While the artificial intelligence and machine learning section describes key developments, popular algorithms and the integration of machine learning into embedded systems. In the last chapter, the development of a self-driving vehicle that, through a convolutional neural network, recognizes and moves according to traffic signs. It also provides practical insights into building and training machine learning models using platforms such as Arduino and Raspberry Pi. Finally, the thesis concludes with findings and optimizations, offering recommendations for future improvements in system development and efficiency.

Scientific Area: Edge Computing

Keywords: virtualization, embedded systems, cloud computing, edge computing, virtual resources, machine learning, artificial intelligence, internet of things

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ABI Application Binary Interface

AC_to_DC Analog to Digital Converter

AI Artificial Intelligence

AIaaS Artificial Intelligence as a Service

AMD Advanced Micro Devices

ANN Artificial Neural Network

API Application Programming Interface

AR Augmented Reality

ARM Advanced RISC Machine

BIOS Basic Input Output System

CDN Content Delivery Network

CGROUPS Control Groups

CMS Cambridge Monitor System

CNN Convolutional Neural Network

CPU Central Process Unit

CP-40 Control Program 40

LAN Local Area Networks

MIT Massachusetts Institute of Technology

ML Machine Learning

NASA National Aeronautics and Space Administration

NEMA National Electrical Manufacturers Association

NIST National Institute of Standards and Technology

NVRAM Non-Volatile Random Access Memory

OCI Open Container Initiative

OS Operating System

PaaS Platform as a Service

PCA Principal Component Analysis

PLC Programmable Logic Controllers

CSP Cloud Service Provider

DSP Digital Signal Processor

GB Gigabyte(s)

GPIO General-Purpose Input/Output

GPU Graphics Processing Unit

GPS Global Positioning System

GUI Graphical User Interface

HDD Hard Disk Drive

HDMI High Definition Multimedia Interface

IaaS Infrastructure as a Service

ICSP In Circuit Serial Programming

IDE Integrated Development Environment

IoT Internet of Things

IIoT Industrial Internet of Things

ISA Instruction Set Architecture

ISS International Space Station

JVM Java Virtual Machine

LXC Linux Containers

PoE Power over Ethernet

PWM Pulse Width Modulation

RAM Random Access Memory

RGB Red Green Blue

RPA Robotic Process Automation

RTOS Real Time Operating System

SaaS System as a Service

SoC System on a Chip

SSD Solid State Drive

SVD Singular Value Decomposition

USB Universal Serial Bus

VBIOS Virtual Basic Input Output System

VM Virtual Machine

VMM Virtual Machine Monitor

VR Virtual Reality

VS Versus

ΕΕΜ Ελεγκτής Εικονικών Μηχανών

ΕΜ Εικονική Μηχανή

ΚΜΕ Κεντρική Μονάδα Επεξεργασίας

Η ΔΟΜΗ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ

ΕΣΑΓΩΓΗ:

Στην εισαγωγή αναλύεται το αντικείμενο της διπλωματικής εργασίας και γίνεται μια ιστορική αναδρομή γύρω από τις μεθόδους που έχουν παρουσιαστεί σε αυτήν την περιοχή.

ΚΕΦΑΛΑΙΟ 1: ΕΙΚΟΝΙΚΟΠΟΙΗΣΗ

Συζητά την έννοια της εικονικοποίησης, την ιστορία, τους τύπους, τις τεχνικές και τα πλεονεκτήματα που προσφέρει. Αυτό το κεφάλαιο θέτει τα θεμέλια για την κατανόηση του τρόπου με τον οποίο η εικονικοποίηση επηρεάζει τα σύγχρονα υπολογιστικά περιβάλλοντα.

ΚΕΦΑΛΑΙΟ 2: ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΕΦΟΥΣ & ΑΚΡΟΥ

Εξερευνά τις αρχές του υπολογιστικού νέφους και του υπολογιστικού άκρου, συμπεριλαμβανομένης της ιστορικής εξέλιξης, των χαρακτηριστικών, των μοντέλων ανάπτυξης και υπηρεσιών και των πλεονεκτημάτων τους. Επίσης, συγκρίνει τα δύο παραδείγματα και τις εφαρμογές τους.

ΚΕΦΑΛΑΙΟ 3: ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Αναφέρει λεπτομερώς το ιστορικό, τα χαρακτηριστικά, τους τύπους, τις αρχιτεκτονικές και τις περιπτώσεις χρήσης των ενσωματωμένων συστημάτων. Αυτό το κεφάλαιο επεξηγεί την ενσωμάτωση αυτών των συστημάτων σε διάφορες εφαρμογές.

ΚΕΦΑΛΑΙΟ 4: ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ & ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Καλύπτει την εξέλιξη της μηχανικής μάθησης, βασικές έννοιες όπως η βαθιά μάθηση και τα νευρωνικά δίκτυα, οι δημοφιλείς αλγόριθμοι και η διασταύρωση της τεχνητής νοημοσύνης με τα ενσωματωμένα συστήματα. Εισάγει επίσης εργαλεία όπως το TensorFlow.

ΚΕΦΑΛΑΙΟ 5: ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ

Περιγράφει την πρακτική ανάπτυξη ενός συστήματος, περιγράφοντας λεπτομερώς στοιχεία όπως το Arduino, το Raspberry Pi, και διαδικασίες δημιουργίας, εκπαίδευσης και αξιολόγησης ενός συνελικτικού νευρωνικού δικτύου.

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ & ΕΠΟΜΕΝΑ ΒΗΜΑΤΑ

Συνοψίζει τα ευρήματα, συζητά τις επιπτώσεις της έρευνας και προτείνει πιθανούς τομείς για μελλοντική εργασία.

ΕΙΣΑΓΩΓΗ

Με τα χρόνια, τα δεδομένα που αναλύουμε, αυξάνονται ολοένα και περισσότερο, συνεπώς η ανάγκη για περισσότερους υπολογιστικούς πόρους μεγαλώνει. Μια λύση ήταν, και είναι, τα μεγάλα κέντρα δεδομένων τα οποία λαμβάνουν, επεξεργάζονται και αποθηκεύουν κάθε είδους πληροφορία. Αυτό δεν αποτελούσε πρόβλημα στο παρελθόν, καθώς οι τότε διαθέσιμες υποδομές μπορούσαν να υποστηρίξουν την αρχιτεκτονική αυτή. Με την αύξηση όμως των δεδομένων που χρήζουν επεξεργασίας έχουν αρχίσει να αποδοκιμάζονται, καθώς πολλές εφαρμογές απαιτούν ελάχιστο latency το οποίο δεν είναι πάντα εφικτό ακολουθώντας την προαναφερθείσα αρχιτεκτονική, κυρίως επειδή πρέπει να μεταφερθούν μέσω του δικτύου τεράστιοι όγκοι δεδομένων σε κάποιο κεντρικό σημείο προκειμένου να επεξεργαστούν. Παράλληλα η τεχνολογία των ενσωματωμένων συστημάτων γίνεται ολοένα και πιο ισχυρή, με αποτέλεσμα τη δημιουργία μικρών συσκευών, ικανών υποστήριξης σημαντικών εφαρμογών. Βάσει του προβλήματος αυτού φτάσαμε σε ένα σημείο όπου δεν αρκεί μόνο η νεφοϋπολογιστική ή μόνο η τοπική επεξεργασία δεδομένων, αλλά ένας συνδυασμός αυτών. Ο συνδυασμός αυτός βασίζεται στην ανάλυση των δεδομένων, από διάφορα ενσωματωμένα συστήματα π.χ raspberry pi, την στιγμή που παράγονται και σε επόμενο χρόνο την αποθήκευση των επεξεργασμένων, πλέον, δεδομένων σε μεγάλα κέντρα δεδομένων. Η παρούσα διπλωματική εργασία αναλύει, συγκρίνει και επεξηγεί τις παραπάνω αρχιτεκτονικές, καθώς και τεχνολογίες μηχανικής μάθησης που εφαρμόζονται σε τέτοιες συσκευές.

Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Αντικείμενο της διπλωματικής αυτής εργασίας είναι η μελέτη τεχνολογιών ανάλυσης δεδομένων στα άκρα του δικτύου (extreme edge computing). Δεδομένου ότι τα σύγχρονα ενσωματωμένα συστήματα γίνονται ολοένα και πιο ισχυρά, μπορούν να εκτελέσουν πιο περίπλοκες διεργασίες ανάλυσης δεδομένων. Στο πλαίσιο της διπλωματικής εργασίας θα μελετηθούν οι έννοιες εικονικοποίηση (virtualization), νεφοϋπολογιστική (cloud computing), υπολογισμός στο άκρο (edge computing), μηχανική μάθηση (machine learning) και τεχνολογίες όπως η πλατφόρμα Tensorflow που υλοποιούν αλγορίθμους μηχανικής μάθησης. Τέλος θα υλοποιηθεί ένα σύστημα, το οποίο θα συλλέγει δεδομένα και θα τα αναλύει τοπικά με χρήση αλγορίθμων μηχανικής μάθησης.

Σκοπός

Σκοπός της παρούσας διπλωματικής εργασίας, είναι να αναδείξει την χρησιμότητα της υπολογιστικής στο άκρο στις μέρες μας και να αναλύσει τεχνολογίες μηχανικής μάθησης που χρησιμοποιούνται στην επεξεργασία πληροφοριών. Πιο συγκεκριμένα, να εξηγήσει τι είναι το cloud computing, τι είναι και γιατί εμφανίστηκε η ανάγκη για την υλοποίηση του edge computing, σε ποιες περιπτώσεις πλεονεκτεί το edge έναντι του cloud, ποιες οι τεχνολογίες, πως βοηθάει η χρήση της μηχανικής μάθησης στην επίτευξη της αυτοματοποιημένης ανάλυσης δεδομένων και με ποιους τρόπους αναλύονται τα δεδομένα στα άκρα του δικτύου. Τέλος, να δημιουργηθεί εφαρμογή η οποία επιδεικνύει τη λειτουργία και την αποτελεσματικότητα του edge computing.

ΕΙΚΟΝΙΚΟΠΟΙΗΣΗ
(Virtualization)

Η εικονικοποίηση αναφέρεται στη διαδικασία διαχωρισμού του υλικού από το λειτουργικό σύστημα σε ένα φυσικό μηχάνημα, μετατρέποντας τις λειτουργίες του υλικού σε λογισμικό. Ουσιαστικά, η εικονικοποίηση δημιουργεί έναν υπολογιστή μέσα σε έναν υπολογιστή, μιμούμενη συσκευές όπως κάρτα ήχου, κεντρική μονάδα επεξεργασίας, μνήμη και μέσα αποθήκευσης. Ένα λειτουργικό σύστημα που εκτελείται σε εικονικό περιβάλλον ονομάζεται εικονική μηχανή (VM). Οι τεχνολογίες εικονικοποίησης επιτρέπουν σε πολλαπλές εικονικές μηχανές με διαφορετικά λειτουργικά συστήματα να λειτουργούν παράλληλα και μεμονωμένα στο ίδιο φυσικό μηχάνημα. Κάθε εικονική μηχανή μιμείται ένα πλήρες σύστημα υλικού, επιτρέποντάς της να μοιράζεται τους φυσικούς πόρους του συστήματος με τις υπόλοιπες, ενώ κάθε μια από αυτές αντιλαμβάνεται ένα σύνολο εικονικών στοιχείων υλικού, ανεξάρτητα από το πραγματικό φυσικό υλικό [1]. Ένα εικονικοποιημένο σύστημα περιλαμβάνει ένα νέο επίπεδο λογισμικού, που ονομάζεται ελεγκτής εικονικών μηχανών (VMMs), όπου πρωταρχικός του ρόλος είναι να διαχειρίζεται την πρόσβαση στους πόρους του φυσικού συστήματος. Αντιθέτως, σε ένα μη εικονικοποιημένο σύστημα, ένα μόνο λειτουργικό σύστημα ελέγχει όλους τους πόρους της πλατφόρμας υλικού [2].

Η εικονικοποίηση καθίσταται δυνατή βάσει τεχνικών όπως, η πλήρης εικονικοποίηση (full virtualization), η παραεικονικοποίηση (paravirtualization) και η εικονικοποίηση υποβοηθούμενη από το υλικό (hardware assisted virtualization). Στις περισσότερες περιπτώσεις, χρησιμοποιούνται νέες ακολουθίες εντολών (instructions) που αφορούν τους εικονικούς πόρους ή υπερκλήσεις (hypercalls) στον hypervisor για την αντικατάσταση συγκεκριμένων μη εικονικοποιήσιμων εντολών του επεξεργαστή σχετικά με τους φυσικούς πόρους [3].

1.1 Ιστορική Αναδρομή

Η εικονικοποίηση χρησιμοποιείται ευρέως για την κοινή χρήση των φυσικών πόρων του υπολογιστή μεταξύ πολλαπλών λειτουργικών συστημάτων. Η ιδέα ξεκίνησε το 1964 όταν η IBM ανέπτυξε το σύστημα CP/CMS, το οποίο εξελίχθηκε στο Virtual Machine Facility/370.

Το Πρόγραμμα Ελέγχου (CP) ήταν ένα λειτουργικό σύστημα ικανό να χρησιμοποιεί μια υπολογιστική μηχανή, έτσι ώστε να λειτουργεί σαν πολλαπλά αντίγραφα του εαυτού της, το καθένα ελεγχόμενο από το δικό του λειτουργικό σύστημα (CMS) [4]. Αυτό επέτρεψε την κοινή χρήση χρόνου (time sharing) και πόρων, βελτιώνοντας την παραγωγικότητα και μειώνοντας το κόστος του υλικού [5]. Το έργο της IBM και οι προσπάθειες της ερευνητικής κοινότητας, όρισαν τη βάση της εικονικοποίησης χαμηλού επιπέδου και στα μέσα της δεκαετίας του 1970, οι Goldberg και Popek δημοσίευσαν έργα που έδωσαν μια σαφή κατανόηση των εννοιών και των πλεονεκτημάτων που σχετίζονται με τις τεχνολογίες εικονικοποίησης [4]. Ωστόσο, καθώς το υλικό έγινε φθηνότερο και εμφανίστηκαν λειτουργικά συστήματα πολλαπλής επεξεργασίας, η χρήση των εικονικών μηχανών μειώθηκε κατά τις δεκαετίες του 1970 και του 1980 [5] μέχρι το 1999, που η εταιρεία VMware παρουσίασε την εικονική πλατφόρμα VMware για την αρχιτεκτονική x86. Η αρχιτεκτονική δεν σχεδιάστηκε για να υποστηρίζει πλήρη εικονικοποίηση με την έννοια που περιέγραψαν οι Popek και Goldberg, ωστόσο, οι τότε σύγχρονες αρχιτεκτονικές x86 και Itanium της Intel υποστήριζαν την εικονικοποίηση μέσω επεκτάσεων που ονομάζονταν VT-x για την x86 και VT-i για την Itanium. Επίσης, οι επεξεργαστές της AMD υποστήριζαν την εικονικοποίηση μέσω της τεχνολογίας Pacifica [4]. Κατά την δεκαετία του 90’ οι περισσότερες εταιρείες χρησιμοποιούσαν φυσικούς εξυπηρετητές και τεχνολογίες λογισμικού ενός παρόχου, τα οποία εμπόδιζαν τις, παλαιότερης τεχνολογίας, εφαρμογές να εκτελεστούν σε υλικό διαφορετικού παρόχου. Ως επακόλουθο, σε κάποιες περιπτώσεις ένας ολόκληρος φυσικός εξυπηρετητής χρησιμοποιούνταν μόνο από μια εφαρμογή η οποία, προφανώς, δεν μπορούσε να τον αξιοποιήσει πλήρως. Η τεχνολογία της εικονικοποίησης έδωσε λύση σε δύο προβλήματα:το ένα είναι η αποδοτική αξιοποίηση των πόρων των συστημάτων και το άλλο είναι, πως οι εξυπηρετητές μπορούσαν πλέον να χωριστούν σε τμήματα όπου το καθένα ήταν τελείως ανεξάρτητο από τα υπόλοιπα και μπορούσε να υποστηρίξει διαφορετικές εφαρμογές ανεξαρτήτως έκδοσης και λειτουργικού συστήματος. Αυτό βοήθησε στην μείωση της κατανάλωσης ενέργειας, του κόστους αγοράς, της τροφοδοσίας ρεύματος, της ψύξης και της συντήρησης των μηχανημάτων, που καλούνταν να πληρώσουν οι εκάστοτε εταιρείες [6]. Σήμερα, οι περισσότερες από τις λύσεις εικονικοποίησης, όπως η πλατφόρμα VMware, κάνουν χρήση των τεχνολογιών Intel VT-x ή AMD-V (τελευταία τεχνολογία εικονικοποίησης της AMD [7]). Η εικονικοποίηση χρησιμοποιείται κατά κόρον από απλούς χρήστες έως πολυεθνικές εταιρείες, έχει καθοριστικό

ρόλο όσον αφορά την οικονομική πλευρά της υπολογιστικής νέφους και αποτελεί κύριο θεμέλιο της αρχιτεκτονικής της.

1.2 Εικονικοί Πόροι (Virtual Resources)

Φυσικός πόρος ενός υπολογιστικού συστήματος χαρακτηρίζεται το υλικό (hardware) το οποίο πραγματοποιεί την ανάγνωση, επεξεργασία ή αποθήκευση των δεδομένων. Πιο συγκεκριμένα είναι, η Κεντρική Μονάδα Επεξεργασίας (CPU), η Μονάδα Επεξεργασίας Γραφικών (GPU), η -προσωρινής αποθήκευσης- Μνήμη Τυχαίας Προσπέλασης (RAM) και ο Σκληρός Δίσκος (HDD) ή ο Δίσκος Στερεάς Κατάστασης (SSD) ως μνήμη μόνιμης αποθήκευσης. Κάθε σύστημα έχει συγκεκριμένο σκοπό για τον οποίο δημιουργήθηκε και επομένως απαιτεί διαφορετικούς πόρους όσον αφορά την τεχνολογία, την απόδοση, την ενεργειακή κατανάλωση κ.α. Αυτοί είναι οι πόροι που διαχειρίζεται ένας hypervisor για την δημιουργία εικονικών πόρων οι οποίοι είναι προσομοιώσεις υλικού και αποτελούν τμήματα του συνόλου των φυσικών πόρων του συστήματος, συμπεριφέρονται όμως σαν φυσικοί πόροι και παρέχονται κατόπιν ζήτησης. Μερικά παραδείγματα τέτοιων πόρων είναι μια εικονική συσκευή αποθήκευσης, μια εικονική ΚΜΕ ή μια εικονική κάρτα δικτύου [1].

1.3 Ελεγκτής Εικονικών Μηχανών (Hypervisor)

Ο hypervisor ή αλλιώς ελεγκτής εικονικών μηχανών (VMMs) είναι ένα επίπεδο λογισμικού που εικονικοποιεί τους πόρους ενός υπολογιστικού συστήματος για να υποστηρίξει πολλαπλές εικονικές μηχανές (VM) [4]. Εικονικοποιεί πόρους υλικού όπως η κεντρική μονάδα επεξεργασίας (ΚΜΕ), η μνήμη και ο φυσικός δίσκος, δημιουργώντας εξομοιούμενες συσκευές και διαχειρίζεται την κατανομή και παρουσίαση αυτών στις εικονικές μηχανές [1]. Πάνω από το επίπεδο αυτό εκτελούνται και παρακολουθούνται πολλαπλά λειτουργικά συστήματα. Με αυτόν τον τρόπο, λειτουργικά συστήματα που είναι ίδια ή διαφορετικά από εκείνο του host μπορούν να μοιραστούν τους φυσικούς του πόρους [8]. Αυτό επιτυγχάνεται δημιουργώντας ξεχωριστές εικονικές μηχανές απομονωμένες από τους πόρους του συστήματος και από το λειτουργικό σύστημα του hypervisor. Το υλικό όταν χρησιμοποιείται ως ελεγκτής εικονικών μηχανών ονομάζεται host, ενώ οι εικονικές μηχανές που χρησιμοποιούν τους πόρους του ονομάζονται guests. Ο hypervisor αντιμετωπίζει τους πόρους του συστήματος ως ένα σύνολο

που μπορεί εύκολα να ανακατανεμηθεί μεταξύ υπαρχόντων guests ή σε νέες εικονικές μηχανές. Όλοι οι ελεγκτές εικονικών μηχανών χρειάζονται μερικές λειτουργίες επιπέδου λειτουργικού συστήματος, όπως διαχειριστή μνήμης (memory manager), προγραμματιστή διεργασιών (process scheduler), στοίβα εισόδου-εξόδου (I/O stack), προγράμματα οδήγησης συσκευών (device drivers), διαχειριστή ασφαλείας (security manager), στοίβα δικτύου (network stack) κ.α, έτσι ώστε να εκτελούν εικονικές μηχανές [9]. Αξίζει να σημειωθεί ότι η μνήμη που έχει διαθέσιμη η εικονική μηχανή είναι διαφορετική από την μνήμη που καταλαμβάνει στο φυσικό μέσο. Για παράδειγμα, εκτός μερικών περιπτώσεων, αν έχουμε δημιουργήσει μια εικονική μηχανή με 1GB διαθέσιμης μνήμης δεν σημαίνει ότι τόσα GB έχουν δεσμευτεί για την εν λόγω μηχανή, αλλά μέχρι τόσα μπορεί να χρησιμοποιήσει. Το ποσοστό της συνολικής μνήμης που θα δεσμευτεί εξαρτάται από ρυθμιζόμενους παράγοντες και τον συνολικό φόρτο της μνήμης του μηχανήματος [10].

Το 1974 στο άρθρο των Gerald J. Popek και Robert P. Goldberg «Formal Requirements for Virtualizable Third Generation Architectures» εισήχθησαν οι απαιτήσεις εικονικοποίησης Popek και Goldberg, είναι ένα σύνολο συνθηκών επαρκών ώστε μια αρχιτεκτονική υπολογιστή να υποστηρίζει αποτελεσματικά την εικονικοποίηση συστήματος. Ως λογισμικό, ένας ελεγκτής εικονικών μηχανών (EEM) έχει τρία βασικά χαρακτηριστικά. Πρώτον, ο ελεγκτής παρέχει ένα περιβάλλον για τα προγράμματα που είναι πανομοιότυπο με το πρωτότυπο μηχάνημα. Δεύτερον, τα προγράμματα που εκτελούνται σε αυτό το περιβάλλον εμφανίζουν, στη χειρότερη, μόνο μικρές μειώσεις στην ταχύτητα και τελευταίο, ο ελεγκτής έχει τον πλήρη έλεγχο των πόρων του συστήματος [11].

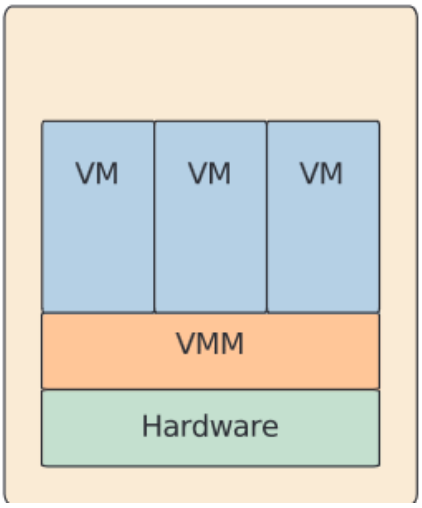
Για να κατανοήσουμε με λίγο περισσότερη λεπτομέρεια την λειτουργία των hypervisors πρέπει να εξετάσουμε πώς τα προγράμματα εκτελούνται στον υπολογιστή, καθώς υπάρχουν ομοιότητες στην διαδικασία. Υπάρχουν κάποια σήματα στους υπολογιστές τα οποία δέχεται η ΚΜΕ από το λογισμικό ή το υλικό σε περίπτωση που κάποια διεργασία ή κάποιο γεγονός (event) χρήζει άμεσης προσοχής. Αυτά τα σήματα συνήθως διακόπτουν την λειτουργία του τρέχοντος προγράμματος και θέτουν σε προτεραιότητα άλλες διεργασίες και για τον λόγο αυτό ονομάζονται διακοπές (interrupts). Η κεντρική μονάδα επεξεργασίας εκτελεί μία προς μία τις εντολές του προγράμματος, ακολουθώντας την ροή εκτέλεσης που υποδεικνύει εκείνο. Η ροή αυτή συχνά ονομάζεται νήμα (thread). Μπορούν να υπάρχουν

πολλά ταυτόχρονα threads σε ένα σύστημα, αλλά στην περίπτωση που είναι μονοπύρηνο δηλαδή με μόνο μια ΚΜΕ, αναφερόμαστε σε ψευδο-παράλληλη επεξεργασία. Αυτό σημαίνει ότι σε κάθε συγκεκριμένη χρονική στιγμή, μπορεί να εκτελείται μόνο ένα thread και όπως είναι αναμενόμενο, δεν μπορεί να το κάνει συνεχόμενα μέχρι να τερματίσει, αφού υπάρχουν περιπτώσεις στις οποίες η εκτέλεση του προγράμματος σταματάει προσωρινά, περιμένοντας είτε είσοδο από τον χρήστη, είτε κάποιο γεγονός να συμβεί. Προχωράμε ένα επίπεδο πιο κάτω, στο λειτουργικό σύστημα όπου είναι υπεύθυνο για την κατανομή του χρόνου εκτέλεσης σε κάθε thread, την διαχείριση σφαλμάτων (π.χ. πρόσβαση σε περιοχή μνήμης στην οποία δεν προβλέπεται), των γεγονότων υλικού (π.χ. -timer interrupt- τέλος του προβλεπόμενου χρόνου εκτέλεσης του thread) και άλλων, τα οποία δεν είναι απαραίτητο να αναφερθούν και να αναλυθούν στην παρούσα φάση. Παρόμοια με το λειτουργικό σύστημα λειτουργούν και οι hypervisors, αλλά αντί να έχουμε ένα λειτουργικό σύστημα και πολλά threads, έχουμε έναν ελεγκτή εικονικών μηχανών και πολλά λειτουργικά συστήματα. Ο ελεγκτής προγραμματίζει ένα λειτουργικό σύστημα για να εκτελεστεί και στην συνέχεια αφήνει την εκτέλεση στην κεντρική μονάδα επεξεργασίας. Αντίστοιχα με το παράδειγμα των threads, ένα λειτουργικό δεν μπορεί να εκτελείται για πάντα και όπως αυτό διαθέτει συγκεκριμένο χώρο μνήμης για κάθε thread, έτσι και ο ελεγκτής εικονικών μηχανών διαθέτει συγκεκριμένους πόρους του υλικού για κάθε ένα λειτουργικό, δίνοντας του έτσι την ψευδαίσθηση ότι εκτελείται σε ξεχωριστό υπολογιστικό σύστημα. Όταν κάποιο λειτουργικό προσπαθήσει να προσπελάσει πόρους που δεν του ανήκουν, τότε ο ελεγκτής καλείται να διαχειριστεί το πρόβλημα. Ένας άλλος λόγος για τον οποίο μπορεί να παρέμβει στην σειρά εκτέλεσης ο ελεγκτής, είναι τα γεγονότα υλικού, όπως αυτό του timer, στο οποίο βασίζεται ο προγραμματισμός του χρόνου των προς εκτέλεση λειτουργικών συστημάτων. Κάθε φορά που τελειώνει ο χρόνος εκτέλεσης ενός προγράμματος δημιουργείται ένα interrupt το οποίο αναγκάζει την ΚΜΕ να αδρανοποιήσει, προσωρινά, το τρέχον thread και να τρέξει το επόμενο, αντίστοιχα ο ελεγκτής εικονικών μηχανών παγιδεύει αυτά τα interrupts και διαχειρίζεται τους χρόνους εκτέλεσης των λειτουργικών [12].

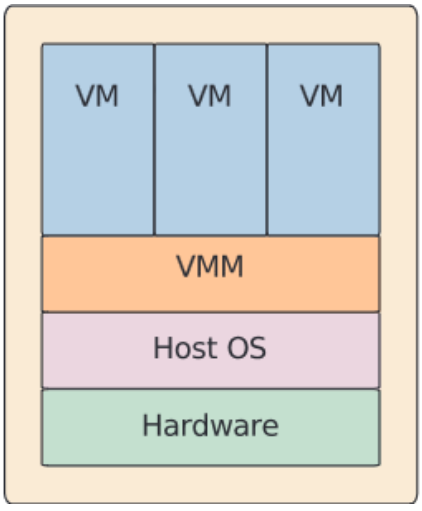
Ο ελεγκτής εικονικών μηχανών περιλαμβάνει τρεις βασικές μονάδες για την εξομοίωση του υποκείμενου υλικού: τον διεκπεραιωτή, τον κατανεμητή και τον διερμηνέα. Ο διεκπεραιωτής είναι αυτός που αλληλεπιδρά με την εικονική μηχανή και είναι υπεύθυνος για

την διανομή των οδηγιών της σε οποιαδήποτε από τις άλλες δύο μονάδες. Ο κατανεμητής καθορίζει την κατανομή των πόρων του συστήματος στην εικονική μηχανή, την οποία επικαλείται ο διεκπεραιωτής, όταν απαιτείται αλλαγή πόρων του μηχανήματος. Τέλος, η μονάδα διερμηνέα, περιέχει ρουτίνες που ενεργοποιούνται κάθε φορά που η εικονική μηχανή εκτελεί μια προνομιακή εντολή, διασφαλίζοντας τη σωστή εξομοίωση της λειτουργικότητας του υλικού [13].

Σύμφωνα με τον Robert P. Golberg [14], γνωστό αμερικανό ερευνητή στο πεδίο των λειτουργικών συστημάτων και της εικονικοποίησης, οι ελεγκτές εικονικών μηχανών κατατάσσονται σε δύο κατηγορίες, τους τύπου I και τους τύπου II [15]. Οι hypervisors τύπου I, γνωστοί και ως bare-metal, εκτελούνται ακριβώς πάνω από το υλικό στο Δαχτυλίδι με τα υψηλότερα δικαιώματα και ελέγχουν όλες τις εικονικές μηχανές. Διαθέτουν στοιχεία διαχείρισης πόρων όπως μνήμη, ΚΜΕ και συσκευές εισόδου/εξόδου. Για παράδειγμα, υπάρχει ένας προγραμματιστής εικονικών μηχανών που χρησιμοποιείται για τη διαχείριση πόρων της ΚΜΕ [4]. Αυτού του τύπου οι ελεγκτές εικονικών μηχανών είναι πιο αποδοτικοί, καθώς επικοινωνούν άμεσα με τους φυσικούς πόρους. Επίσης, είναι πιο ασφαλείς διότι δεν παρεμβάλλεται τίποτα μεταξύ του ελεγκτή και της ΚΜΕ το οποίο θα μπορούσε να παραβιαστεί με σκοπό την απόκτηση ελέγχου του συστήματος [16]. Τέτοιου τύπου hypervisors αποτελούν οι VMware ESX, Hyper-V και Xen. Οι hypervisors τύπου II εκτελούνται ως εφαρμογές σε ένα υπάρχον λειτουργικό σύστημα και διαχειρίζονται τις εικονικές μηχανές, ως πρόσθετες διεργασίες λειτουργικού συστήματος. Λόγω αυτού, για τη διαχείριση των πόρων, βασίζονται στον προγραμματιστή διεργασιών του λειτουργικού συστήματος πάνω από το οποίο τρέχουν [4]. Αυτού του τύπου οι ελεγκτές σπάνια χρησιμοποιούνται σε περιβάλλοντα διακομιστών



Εικόνα 2 - Hypervisor Type I [4]



Εικόνα 1 - Hypervisor Type II [4]

(servers), είναι όμως κατάλληλοι για χρήστες υπολογιστών που χρειάζεται να έχουν πολλαπλά λειτουργικά συστήματα (π.χ επαγγελματίες ασφαλείας συστημάτων). Στην παραπάνω κατηγορία η πρόσβαση στην ΚΜΕ, την μνήμη και τους δικτυακούς πόρους γίνεται μέσω του λειτουργικού συστήματος το οποίο έχει άμεση πρόσβαση στο υλικό. Λόγω της διαμεσολάβησης του λογισμικού παρουσιάζονται προβλήματα καθυστέρησης (latency) και επηρεάζεται η απόδοση. Επίσης, παρουσιάζονται πιθανοί κίνδυνοι ασφαλείας καθώς, αν κάποιος παραβιάσει και πάρει τον έλεγχο του host λειτουργικού συστήματος, μπορεί να διαχειριστεί και τα υπόλοιπα λειτουργικά που εκτελούνται υπό τον έλεγχο του τύπου ΙΙ hypervisor [16]. Παραδείγματα τύπου ΙΙ αποτελούν ο VMware server και ο VirtualBox [4].

1.4 Είδη Εικονικοποίησης

Η εικονικοποίηση στη νεφοϋπολογιστική αναφέρεται ως η διαδικασία δημιουργίας μιας εικονικής, αντί πραγματικής, εκδοχής από κάτι που περιλαμβάνει λειτουργικό σύστημα, συσκευές υλικού, συσκευές αποθήκευσης, δίκτυο και πολλούς άλλους πόρους. Παρακάτω παρουσιάζονται παραδείγματα εικονικοποιήσιμων πόρων [17].

Εικονικοποίηση Εφαρμογών: Η εικονικοποίηση εφαρμογών επιτρέπει στις εφαρμογές να εκτελούνται σε runtime περιβάλλον, χωρίς εγγενή υποστήριξη ή εγκατάσταση. Αυτό επιτυγχάνεται μέσω ενός επιπέδου εικονικοποίησης, που ανακατευθύνει τις λειτουργίες του δίσκου σε μια εικονική τοποθεσία. Έτσι επιτρέπει στις εφαρμογές να έχουν πρόσβαση σε εικονικούς πόρους σαν να ήταν πραγματικοί, διευκολύνοντας τη συμβατότητα μεταξύ διαφορετικών συστημάτων [17]. Ένα παράδειγμα εικονικοποίησης εφαρμογών είναι όταν κάποιος χρησιμοποιεί το Microsoft Office στον υπολογιστή του, αλλά η κατάσταση λειτουργίας και τα προσωπικά δεδομένα της εφαρμογής διαχειρίζονται και αποθηκεύονται σε εικονικό λογισμικό. Ενώ ο υπολογιστής παρέχει υποστήριξη υλικού όπως επεξεργασία RAM και CPU, η εφαρμογή και τα αρχεία δεν αποθηκεύονται στο σκληρό δίσκο του υπολογιστή. Αυτή η ρύθμιση επιτρέπει στην εφαρμογή να εκτελείται τοπικά στον υπολογιστή, ενώ η λογική και η διαχείριση των δεδομένων της, πραγματοποιούνται εξ' αποστάσεως σε μια εικονική μηχανή [18]. Τεχνολογίες όπως το streaming εφαρμογών, οι υπηρεσίες απομακρυσμένου ελέγχου και η εικονικοποίηση επιφάνειας εργασίας, εμπίπτουν σε αυτήν την κατηγορία. Τα οφέλη περιλαμβάνουν μειωμένη χρήση πόρων, σε σύγκριση με τις

εικονικές μηχανές, χαμηλότερο κόστος διαχείρισης και δυνατότητα εκτέλεσης εφαρμογών στους guests χωρίς εγκατάσταση [17].

Εικονικοποίηση Γλώσσας Προγραμματισμού (Programming Language Virtualization): Η εικονικοποίηση γλώσσας προγραμματισμού, περιλαμβάνει τη δημιουργία πλατφορμών για την ανάπτυξη διαδικτυακών εταιρικών εφαρμογών σε Java. Η εικονική μηχανή της Java (JVM) μπορεί να εκτελέσει προγράμματα γραμμένα σε Java και άλλες γλώσσες όπως Python, Ruby, Pascal και Groovy, εκτελώντας τους μεταγλωττισμένους δυαδικούς κώδικες (bytecodes) τους σε οποιοδήποτε λειτουργικό σύστημα ή πλατφόρμα. Αυτή η προσέγγιση απλοποιεί τη διαδικασία ανάπτυξης, αποφεύγοντας την ανάγκη για πολλαπλές εκδόσεις κώδικα, βελτιώνοντας τη φορητότητα και διασφαλίζοντας τη διαχειριζόμενη εκτέλεση. Ο δυαδικός κώδικας ερμηνεύεται με βάση το υποκείμενο σύνολο εντολών υλικού κατά το χρόνο εκτέλεσης, παρέχοντας ασφάλεια μέσω φιλτραρισμένων λειτουργιών εισόδου/εξόδου. Ωστόσο, αυτή η μέθοδος μπορεί να έχει ως αποτέλεσμα χαμηλότερη απόδοση και μικρότερη σημασία για τέτοιες εφαρμογές λόγω της έλλειψης άμεσης πρόσβασης στη μνήμη [17].

Εικονικοποίηση Λειτουργικού Συστήματος (OS Virtualization): Η εικονικοποίηση λειτουργικού συστήματος επιτρέπει στον πυρήνα να υποστηρίζει πολλαπλούς απομονωμένους χώρους χρήστη, δημιουργώντας ξεχωριστά περιβάλλοντα για ταυτόχρονα διαχειρίσιμες εφαρμογές. Κάθε χώρος χρήστη περιλαμβάνει λεπτομέρειες όπως, προβολές συστήματος αρχείων, διευθύνσεις IP, διαμορφώσεις λογισμικού και πρόσβαση σε συσκευές. Επίσης, προσφέρει ισχυρές δυνατότητες χώρου ονομάτων (namespaces) και κοινής χρήσης χρόνου (time-sharing). Αυτή η εικονικοποίηση δεν φέρει επιπλέον επιβάρυνση, καθώς οι εφαρμογές πραγματοποιούν άμεσες κλήσεις συστήματος χωρίς εξομοίωση και δεν απαιτεί καμία τροποποίηση στο υλικό ή τις εφαρμογές [17].

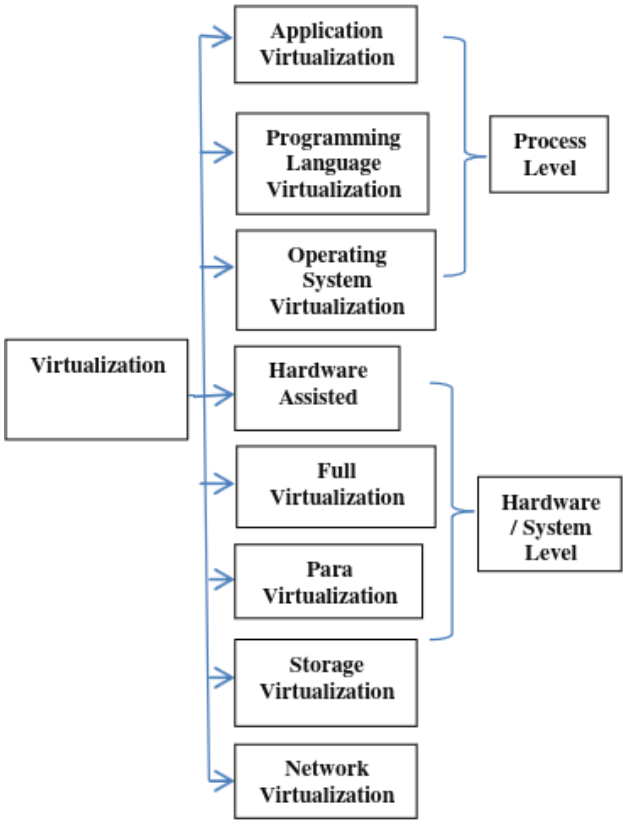
Εικονικοποίηση Υλικού (Hardware Virtualization): Η εικονικοποίηση σε επίπεδο υλικού παρέχει ένα περιβάλλον εκτέλεσης, όπου φιλοξενούμενα λειτουργικά συστήματα μπορούν να εκτελεστούν. Σε αυτήν την περίπτωση, ο guest είναι το λειτουργικό σύστημα, ο host είναι το φυσικό σύστημα, η εικονική μηχανή είναι η εξομοίωση του host και ο διαχειριστής της εικονικής μηχανής είναι ο hypervisor. Αυτός ο τύπος εικονικοποίησης κάνει

μια εικονική μηχανή να λειτουργεί σαν πραγματικός υπολογιστής με λειτουργικό σύστημα, επιτρέποντάς της να εκτελεί διάφορα ανεξάρτητα λογισμικά. Η εικονικοποίηση με τη βοήθεια υλικού, η παρα-εικονικοποίηση και η πλήρης εικονικοποίηση είναι οι διαφορετικοί τύποι τεχνικών εικονικοποίησης υλικού [17].

Εικονικοποίηση Αποθήκευσης (Storage Virtualization): Η εικονικοποίηση αποθήκευσης ενισχύει τα συστήματα αποθήκευσης παρέχοντας αξιόπιστη προστασία δεδομένων και γρήγορη πρόσβαση για υπολογιστές και επεξεργασία δεδομένων, μέσω εξειδικευμένου λογισμικού, υλικού και μονάδων σκληρού δίσκου. Οι χρήστες μπορούν να έχουν πρόσβαση στα δεδομένα μέσω λογικών μονοπατιών, χωρίς να ανησυχούν για τη φυσική τους θέση. Υπάρχουν δύο κύριοι τύποι εικονικοποίησης αποθήκευσης: Εικονικοποίηση μπλοκ και εικονικοποίηση αρχείων. Η εικονικοποίηση μπλοκ διαχωρίζει τη λογική αποθήκευση από τη φυσική αποθήκευση, επιτρέποντας την πρόσβαση στα δεδομένα χωρίς να λαμβάνεται υπόψη η πραγματική τους θέση. Η εικονικοποίηση αρχείων αφαιρεί τις εξαρτήσεις μεταξύ της θέσης δεδομένων και της πρόσβασης. Τα κύρια πλεονεκτήματα περιλαμβάνουν την απλότητα στο σχεδιασμό και την κωδικοποίηση, την υποστήριξη για διάφορους τύπους αποθήκευσης και τη βελτιωμένη χρήση αποθήκευσης χωρίς περιορισμούς [17].

Εικονικοποίηση Δικτύου (Network Virtualization): Η εικονικοποίηση δικτύου δημιουργεί ένα εικονικό δίκτυο που ενσωματώνει πόρους και λειτουργίες δικτύου, τόσο λογισμικού, όσο και υλικού. Διακρίνεται σε δύο τύπους: εξωτερικό και εσωτερικό. Η εικονικοποίηση εξωτερικού δικτύου, γνωστή και ως εικονικό LAN, συνδυάζει μέρη ή ολόκληρα δίκτυα σε μια εικονική λογική μονάδα. Η εικονικοποίηση εσωτερικού δικτύου παρέχει λειτουργίες δικτύου εντός λογισμικού, επιτρέποντας στους guests να επικοινωνούν μέσω διεπαφών εικονικού δικτύου που διαμοιράζονται από τον κεντρικό υπολογιστή. Αυτή η τεχνολογία είναι επωφελής σε εφαρμογές δοκιμών λογισμικού (testing software), επιτρέποντας

την προσομοίωση ενός περιβάλλοντος δικτύου με την εξομοίωση συνδέσεων μεταξύ υπηρεσιών, εφαρμογών, εξαρτήσεων και χρηστών χωρίς φυσική δοκιμή [17].

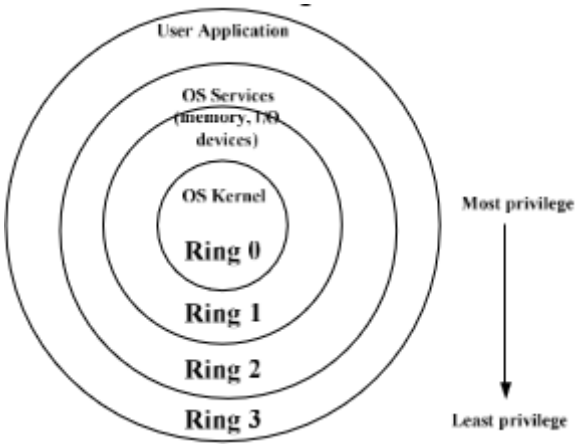


Εικόνα 3 - Είδη Εικονικοποίησης [17]

1.5 Τεχνικές Εικονικοποίησης

Τα λειτουργικά συστήματα που είναι βασισμένα στην αρχιτεκτονική x86 (32 bit αρχιτεκτονική επεξεργαστών Intel και AMD), είναι σχεδιασμένα να εκτελούνται κατευθείαν πάνω στο υλικό του συστήματος, γι' αυτό θεωρούν πως τους ανήκει ολοκληρωτικά. Η αρχιτεκτονική αυτή, προσφέρει στο λειτουργικό σύστημα και στις εφαρμογές, τέσσερα επίπεδα προνομιακής διαχείρισης του υλικού, γνωστά και ως Δαχτυλίδι (Ring) 0,1,2 και 3. Οι εφαρμογές χρήστη τυπικά εκτελούνται στο Δαχτυλίδι 3 (user space), ενώ το λειτουργικό σύστημα πρέπει να έχει απευθείας πρόσβαση στο υλικό και να εκτελεί τις προνομιούχες εντολές στο Δαχτυλίδι 0 (kernel space). Η εικονικοποίηση της x86 αρχιτεκτονικής, απαιτεί την τοποθέτηση ενός επιπέδου εικονικοποίησης κάτω από το λειτουργικό σύστημα για την δημιουργία και διαχείριση των εικονικών μηχανών που παρέχουν κοινόχρηστους πόρους. Αυτό το επίπεδο υλοποιούν οι EEM, εκμεταλλευόμενοι αυτή την θέση, χρησιμοποιούν τις

παρακάτω τεχνικές για την εικονικοποίηση του υλικού. Είναι σημαντικό να σημειωθεί, πως κάποιες εντολές δεν μπορούν να εικονικοποιηθούν αποτελεσματικά, καθώς έχουν διαφορετική σημασιολογία όταν δεν εκτελεστούν στο Δαχτυλίδι 0 [3].

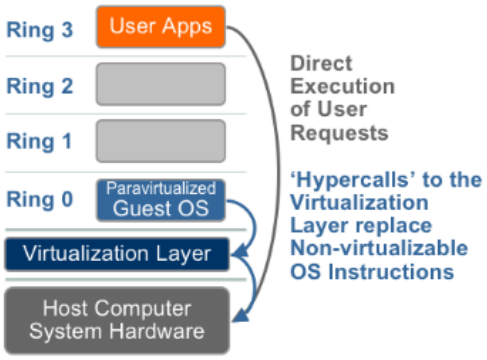


Εικόνα 4 - Δαχτυλίδια Προνομίων [20]

Πλήρης Εικονικοποίηση (Full Virtualization): Η δυαδική μετάφραση (BT) χρησιμοποιείται για την εξομοίωση μιας αρχιτεκτονικής επεξεργαστή σε μια άλλη, επιτρέποντας σε μη τροποποιημένα guest λειτουργικά συστήματα να εκτελούνται, μεταφράζοντας κώδικα από το ένα σύνολο εντολών (instruction set) στο άλλο. Αυτή η προσέγγιση παρέχει φορητότητα σε πολλαπλές πλατφόρμες για τον κώδικα της εφαρμογής και τα λειτουργικά συστήματα, αλλά έχει επιπλέον επεξεργαστική επιβάρυνση, λόγω της εξομοίωσης όλου του συνόλου εντολών της αρχιτεκτονικής [4]. Η πλήρης εικονικοποίηση με χρήση δυαδικής μετάφρασης, περιλαμβάνει τη μετάφραση του κώδικα του πυρήνα μέσω της δυαδικής μετάφρασης και τεχνικών άμεσης εκτέλεσης, αντικαθιστώντας τις μη εικονικές εντολές με εντολές συμβατές με το υλικό [19]. Σε αντίθεση με την πλήρη εξομοίωση, το BT μεταφράζει μόνο ένα μικρό σύνολο προνομιακών εντολών, όπως εκείνες που χρειάζονται εκτέλεση σε λειτουργία πυρήνα (Δαχτυλίδι 0), ενώ οι υπόλοιπες εντολές, εκτελούνται απευθείας από τον επεξεργαστή του host, επιτυγχάνοντας υψηλής απόδοσης εικονικοποίηση [4]. Κάθε εικονική μηχανή λαμβάνει υπηρεσίες από το φυσικό σύστημα, όπως εικονικό BIOS, εικονικές συσκευές και εικονική διαχείριση μνήμης, που παρέχονται από τον ελεγκτή εικονικών μηχανών. Το επίπεδο εικονικοποίησης αποσυνδέει πλήρως το guest λειτουργικό σύστημα, το οποίο δεν γνωρίζει ότι εκτελείται σε εικονικό περιβάλλον και δεν απαιτεί καμία

τροποποίηση. Αυτή η μέθοδος δεν χρειάζεται βοήθεια υλικού ή αλλαγές στο λειτουργικό σύστημα για την εικονικοποίηση ευαίσθητων εντολών [19], ξεπερνά την πλήρη εξομοίωση και παρουσιάζει χαμηλό κόστος επεξεργασίας. Από την άλλη πλευρά, σε αντίθεση με την εξομοίωση, η υποστήριξη των μη τροποποιημένων guest λειτουργικών συστημάτων, περιορίζεται σε εκείνα που είναι συμβατά με την ΚΜΕ του host [4]. Ο hypervisor μεταφράζει και αποθηκεύει προσωρινά όλες τις εντολές του λειτουργικού συστήματος, διασφαλίζοντας υψηλή ασφάλεια, απομόνωση και εύκολη μετεγκατάσταση των εικονικών μηχανών [19]. Βάσει των λεγόμενων της VMware, η πλήρης εικονικοποίηση με δυαδική μετάφραση είναι η πιο καθιερωμένη και αξιόπιστη διαθέσιμη τεχνολογία εικονικοποίησης [20].

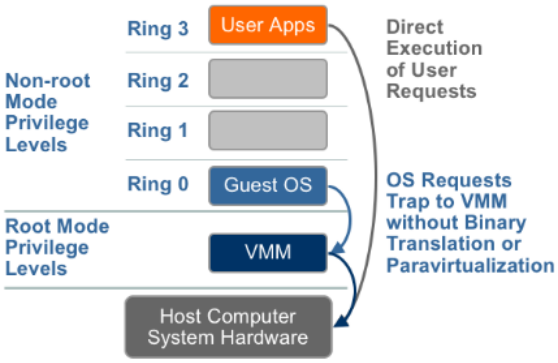
Παραεικονικοποίηση (Paravirtualization): Σε αντίθεση με την τεχνική της Πλήρους Εικονικοποίησης όπου το λειτουργικό σύστημα δεν τροποποιείται για να εκτελεστεί σε εικονικό περιβάλλον, υπάρχει η τεχνική της παραεικονικοποίησης όπου ο πυρήνας του λειτουργικού συστήματος τροποποιείται με σκοπό να αντικατασταθούν όλες οι μη εικονικοποιήσιμες εντολές με υπερκλήσεις που επικοινωνούν άμεσα με τον ελεγκτή εικονικών μηχανών του επιπέδου εικονικοποίησης. Ο ελεγκτής παρέχει, επίσης, διεπαφές και για άλλες κρίσιμες λειτουργίες του πυρήνα όπως, η διαχείριση μνήμης, η διαχείριση διακοπών και η διατήρηση χρόνου. Αυτό που προσφέρει η παραεικονικοποίηση είναι λιγότερος υπολογιστικός φόρτος για την ανάγκη της εικονικοποίησης, αλλά το πλεονέκτημα της επίδοσης της έναντι της Πλήρους Εικονικοποίησης ποικίλλει σημαντικά ανάλογα με τον φόρτο εργασίας. Αφού η παραεικονικοποίηση δεν μπορεί να υποστηρίξει μη τροποποιήσιμα λειτουργικά συστήματα (π.χ Windows), η συμβατότητα και η φορητότητα της είναι ελάχιστη [3].



Εικόνα 5 – Paravirtualization [3]

Εικονικοποίηση Υποβοηθούμενη από Υλικό (Hardware Assisted Virtualization): Η εικονικοποίηση υποβοηθούμενη από υλικό, που εφαρμόστηκε για πρώτη φορά στο IBM System/370 το 1972, έχει εξελιχθεί από εταιρείες κατασκευής επεξεργαστών όπως η AMD και η Intel με την εισαγωγή επεκτάσεων εικονικοποίησης, ξεκινώντας το 2006. Αυτή η τεχνική χρησιμοποιεί μια υψηλότερη προνομιακή λειτουργία στην αρχιτεκτονική του επεξεργαστή,

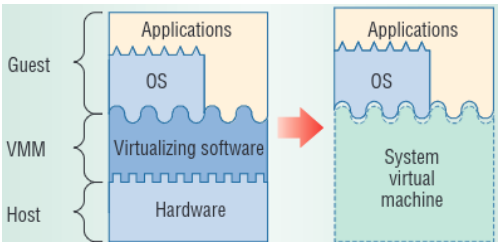
επιτρέποντας σε μη τροποποιημένα λειτουργικά συστήματα να εκτελούνται στο Δαχτυλίδι 0 (non-root mode) και ο Hypervisor στο Δαχτυλίδι -1 (root mode). Αυτές οι επεκτάσεις της κεντρικής μονάδας επεξεργασίας εξαλείφουν την ανάγκη για δυαδική μετάφραση ή παρα-εικονικοποίηση εκτελώντας λειτουργίες παγίδευσης και εξομοίωσης (trap-and-emulate) μέσω υλικού αντί λογισμικού, μειώνοντας έτσι τον επεξεργαστικό φόρτο [4]. Αυτή η διαμόρφωση διασφαλίζει ότι οι προνομιακές λειτουργίες που επιχειρούνται από το guest λειτουργικό σύστημα, παγιδεύονται αυτόματα από τον ελεγκτή εικονικών μηχανών. Το νέο επίπεδο προνομίων επιτρέπει στον EEM να χρησιμοποιεί με ασφάλεια και διαφάνεια την άμεση εκτέλεση για τις εικονικές μηχανές, απλοποιώντας έτσι τη σχεδίασή του. Παρά την αντιμετώπιση των προκλήσεων εικονικοποίησης της ΚΜΕ, οι υλοποιήσεις πρώτης γενιάς των Intel-VT και AMD-V συχνά δεν βελτίωναν σημαντικά την απόδοση, λόγω της άκαμπτης και υψηλού κόστους μέθοδο χειρισμού των μεταβάσεων μεταξύ EEM (Δαχτυλίδι -1) και guest λειτουργικού συστήματος (Δαχτυλίδι 0). Η πλήρης εικονικοποίηση με δυαδική μετάφραση παραμένει εξαιρετικά συμβατή και αξιόπιστη, αν και πάσχει από εγγενείς περιορισμούς λογισμικού, όπως επιβάρυνση του συστήματος για τις ανάγκες πρόσβασης στη μνήμη και εκτέλεσης στην ΚΜΕ. Κατά συνέπεια, η εικονικοποίηση με τη βοήθεια του υλικού θεωρείται το μέλλον της εικονικοποίησης, με μεθόδους λογισμικού που θα χρησιμοποιούνται ως ενδιάμεσα μέσα για την ενίσχυση της απόδοσης [20]. Ωστόσο, η εικονικοποίηση μέσω υλικού δεν είναι διαθέσιμη σε όλους τους επεξεργαστές x86/x86_64. Παρόλο που επιτρέπει περισσότερα εικονικά μηχανήματα με μη τροποποιημένα guest λειτουργικά συστήματα, μπορεί να οδηγήσει σε ζητήματα όπως υψηλή επιβάρυνση της κεντρικής μονάδας επεξεργασίας και μειωμένη επεκτασιμότητα. Το πρωταρχικό πλεονέκτημα της είναι η μείωση των γενικών εξόδων συντήρησης που σχετίζονται με την παραεικονικοποίηση εξαλείφοντας την ανάγκη για τροποποιήσεις στο guest λειτουργικό σύστημα [17].



Εικόνα 6 - Hardware Assisted Virtualization [3]

1.6 Εικονικές Μηχανές (Virtual Machines)

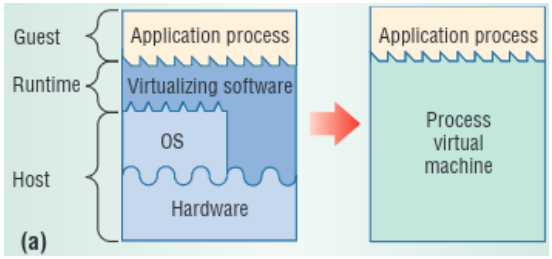
Καθώς τα μέρη που αποτελούν έναν φυσικό υπολογιστή είναι υλικά και απτά, οι εικονικές μηχανές θεωρούνται εικονικοί υπολογιστές ή υπολογιστές καθορισμένοι από λογισμικό, που υπάρχουν μόνο ως κώδικας εντός φυσικών συστημάτων [21]. Μια εικονική μηχανή είναι η εικονική αναπαράσταση ή προσομοίωση ενός φυσικού υπολογιστικού συστήματος [22], προσφέρει ένα πλήρες και μόνιμο περιβάλλον που υποστηρίζει ένα λειτουργικό σύστημα με τις διεργασίες του χρήστη. Παρέχει το guest λειτουργικό με πρόσβαση στους εικονικούς πόρους του υλικού και συχνά με γραφικό περιβάλλον (GUI). Όπως αναφέρθηκε στο προηγούμενο υποκεφάλαιο το λογισμικό εικονικοποίησης μιας εικονικής μηχανής συστήματος είναι ο ελεγκτής εικονικών μηχανών, το οποίο προσομοιώνει το σετ εντολών της αρχιτεκτονικής του υλικού (ISA), επιτρέποντας στο λογισμικό να εκτελέσει μιας διαφορετικής αρχιτεκτονικής σετ εντολών από αυτό του υλικού του host [23]. Εκτελείται σε ένα απομονωμένο τμήμα του φυσικού υπολογιστή με την «δική της» ΚΜΕ, μνήμη, λειτουργικό σύστημα (π.χ GNU Linux, Windows, MacOS) και άλλους πόρους. Οι χρήστες μπορούν να εκτελέσουν εφαρμογές μέσω των εικονικών μηχανών και να τις χρησιμοποιήσουν όπως θα έκαναν με ένα μη εικονικό υπολογιστικό σύστημα [24]. Πρακτικά οι εικονικές μηχανές αποτελούνται από αρχεία, βασικά από τα οποία είναι το αρχείο καταγραφής (log file), το αρχείο ρυθμίσεων της NVRAM (NVRAM setting file), το αρχείο εικονικού δίσκου (virtual disk file) και το αρχείο διαμόρφωσης (configuration file) [25].



Εικόνα 7 - Εικονικές Μηχανές Συστήματος [23]

Μέχρι αυτό το σημείο της εργασίας έχουμε μάθει ότι οι εικονικές μηχανές είναι τα εικονικά περιβάλλοντα που προσομοιώνουν ένα ολόκληρο υπολογιστικό σύστημα εικονικοποιώντας τους πόρους του υλικού (ΚΜΕ, μνήμη κ.α) που παρέχει αυτό, χρησιμοποιώντας ένα ενδιάμεσο λογισμικό διαχείρισης που ονομάζεται ελεγκτής εικονικών μηχανών. Αυτές οι εικονικές μηχανές ονομάζονται εικονικές μηχανές συστήματος (System Virtual Machines). Υπάρχει όμως άλλη μια κατηγορία εικονικών μηχανών, οι εικονικές μηχανές διεργασίας (Process Virtual Machines). Μια διεργασία μπορεί να υποστηρίξει μόνο δυαδικά αρχεία του προγράμματος (program binaries) μεταγλωττισμένα (compiled) για την αρχιτεκτονική του επεξεργαστή που χρησιμοποιείται από το μηχάνημα που την εκτελεί.

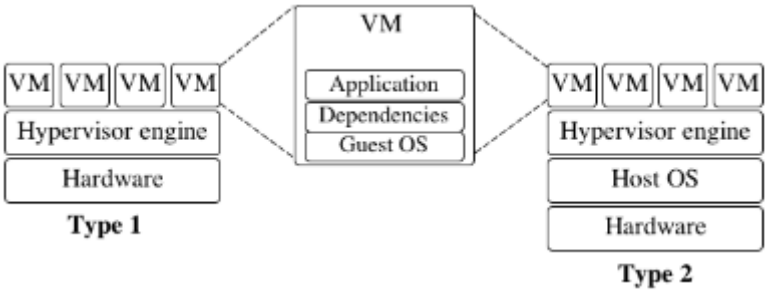
Επομένως, αν προσπαθήσουμε να εκτελέσουμε ένα πρόγραμμα μεταγλωττισμένο για ARM (Advanced RISC Machine) αρχιτεκτονική σε υπολογιστή που ο επεξεργαστής του είναι x86 αρχιτεκτονικής (π.χ Intel), δεν θα λειτουργήσει. Αυτό ακριβώς το πρόβλημα λύνουν οι εικονικές μηχανές διεργασίας μέσω της προσομοίωσης, η οποία επιτρέπει στις διεπαφές (interfaces) και στις λειτουργίες ενός συστήματος, να υλοποιηθούν σε ένα άλλο σύστημα με διαφορετικές διεπαφές και λειτουργίες [25]. Μια εικονική μηχανή διεργασιών είναι μια εικονική πλατφόρμα που έχει σχεδιαστεί για να εκτελεί μια ενιαία διαδικασία, η οποία υπάρχει μόνο για τη διάρκεια αυτής της διαδικασίας. Όπως στις εικονικές μηχανές συστήματος έχουμε τον EEM για τον διαχωρισμό του υλικού και των εικονικών μηχανών, έτσι και στις εικονικές μηχανές διεργασίας έχουμε ένα λογισμικό εικονικοποίησης που καθιστά ανεξάρτητες τις διεργασίες από το λειτουργικό σύστημα. Το λογισμικό αυτό ονομάζεται runtime, λειτουργεί σε επίπεδο δυαδικής διεπαφής εφαρμογής (ABI) ή διεπαφής προγραμματισμού εφαρμογής (API) πάνω από το λειτουργικό σύστημα και το υλικό. Το runtime προσομοιώνει τις οδηγίες επιπέδου χρήστη και τις κλήσεις συστήματος ή βιβλιοθήκης, υποστηρίζοντας τη διεργασία καθ' όλη την εκτέλεσή της [23]. Ένα παράδειγμα εικονικής μηχανής διεργασίας είναι η εικονική μηχανή της Java (JVM), όπου με την βοήθεια της τα προγράμματα σε Java εκτελούνται αυτούσια σε οποιοδήποτε λειτουργικό σύστημα. Οι εικονικές μηχανές συστήματος αποδίδουν καλύτερα σε σχέση με τις εικονικές μηχανές διεργασίας και αυτό οφείλεται στο ότι οι EEM εκμεταλλεύονται την υποστήριξη της εικονικοποίησης που υπάρχει μέσα στις KME (Hardware Assisted Virtualization) [26].



Εικόνα 8 - Εικονικές Μηχανές Διεργασίας [23]

Όπως προαναφέρθηκε οι εικονικές μηχανές αποτελούν το κυριότερο μέρος για την λειτουργίας της νεφοϋπολογιστικής. Η χρησιμότητα τους όμως δεν σταματάει εκεί. Χρησιμοποιούνται και σε διάφορες άλλες εμπορικές ή μη εμπορικές περιπτώσεις όπως, στην διερεύνηση κακόβουλου λογισμικού, στην δοκιμή λειτουργικών συστημάτων, στην εκτέλεση μη συμβατού λογισμικού με το υποκείμενο λειτουργικό σύστημα και την ασφαλή περιήγηση σε διάφορους ιστότοπους. Σε κάθε μια από αυτές τις περιπτώσεις οι εικονικές μηχανές παρέχουν, ταχύτατα, ένα περιβάλλον το οποίο μπορεί να κλωνοποιηθεί και σε περίπτωση

καταστροφής του, για τον οποιονδήποτε λόγο, να αντικατασταθεί άμεσα, χωρίς να προκληθεί ζημιά σε αρχεία αξίας, είτε προσωπικής, είτε εταιρικής [22].



Εικόνα 9 - Περιεχόμενα Εικονικών Μηχανών [30]

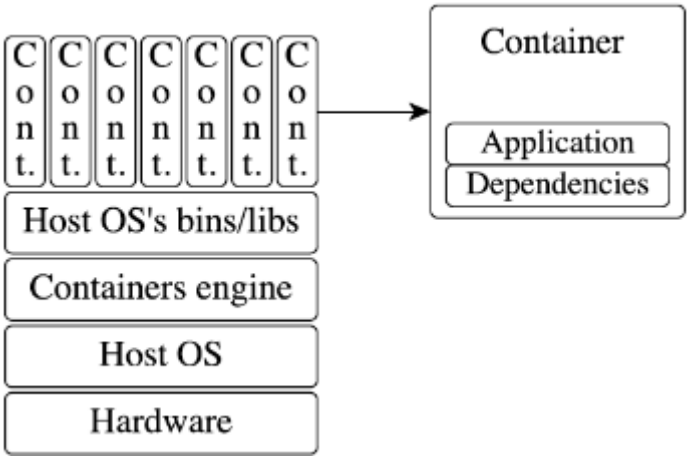
Αφού αναλύσαμε τα δύο είδη των εικονικών μηχανών, από αυτό το σημείο και έπειτα όταν αναφερόμαστε σε εικονικές μηχανές θα εννοούμε τις εικονικές μηχανές συστήματος, εκτός αν αναφερθούμε συγκεκριμένα σε κάποια άλλη τεχνολογία.

1.7 Περιέκτες (Containers)

Οι περιέκτες, αρχικά, δημιουργήθηκαν για το λειτουργικό σύστημα GNU Linux εκμεταλλευόμενοι μηχανισμούς του πυρήνα όπως οι ομάδες ελέγχου (cgroups) και οι χώροι ονομάτων (namespaces). Στην συνέχεια όμως, επεκτάθηκαν και στο λειτουργικό σύστημα των Windows [27]. Η τεχνολογία αυτή βελτιώνει την κοινή χρήση πόρων υλικού καταργώντας ένα επίπεδο διαμόρφωσης και ρύθμισης της υποδομής. Σε αντίθεση με τις εικονικές μηχανές, οι περιέκτες δεν χρειάζονται ξεχωριστό λειτουργικό σύστημα και μπορούν επίσης να λειτουργούν μέσα σε εικονικές μηχανές [28]. Οι τεχνολογίες εικονικοποίησης των hypervisors και των περιεκτών είναι κατάλληλες για διαφορετικές περιπτώσεις χρήσης. Και οι δύο τύποι αφαιρούν την απευθείας αλληλεπίδραση με το υλικό, διευκολύνοντας την ευκολότερη μεταφορά των εικονικών μηχανών ή των εφαρμογών, την καλύτερη χρήση των πόρων, τη μείωση του κόστους και την εξοικονόμηση ενέργειας [29]. Η εικονικοποίηση που βασίζεται στους περιέκτες προσφέρει μια «ελαφριά» εναλλακτική αυτής των hypervisors, παρέχοντας απομόνωση των διεργασιών και θέτοντας όρια χρήσης των πόρων υλικού (ΚΜΕ, μνήμη, δίκτυο) σε επίπεδο λειτουργικού συστήματος (εικονικοποίηση λειτουργικού συστήματος) αντί εικονικοποίησης υλικού. Αυτό επιτυγχάνεται με τη χρήση των cgroups και namespaces [30]. Η προσέγγιση αυτή αποφεύγει την επιβάρυνση που σχετίζεται με τους ελεγκτές εικονικών μηχανών, οι οποίοι εκτελούν πλήρη λειτουργικά συστήματα σε κάθε εικονική μηχανή, με

αποτέλεσμα ταχύτερη εκκίνηση και πιο αποτελεσματική λειτουργία [29]. Κάθε περιέκτης περιλαμβάνει μία ή περισσότερες μικρού μεγέθους εικόνες περιεκτών, οι οποίες περιλαμβάνουν τα απαραίτητα δυαδικά αρχεία, βιβλιοθήκες ή ενδιάμεσο λογισμικό για την εκτέλεση μιας εφαρμογής. Αυτές οι εικόνες μπορούν να στοιβάζονται, δημιουργώντας επίπεδα με συστήματα αρχείων που υποστηρίζουν μόνο ανάγνωση (read-only), κάτω από το ανώτατο επίπεδο που παρέχει σύστημα αρχείων με υποστήριξη εγγραφής, επιτρέποντας την κοινή χρήση του λειτουργικού συστήματος, των δυαδικών αρχείων και των βιβλιοθηκών [31]. Πολλοί περιέκτες μπορούν να εκτελούνται σε ένα μόνο μηχάνημα, όπου μοιράζονται τον πυρήνα του λειτουργικού συστήματος αλλά λειτουργούν ως μεμονωμένες διεργασίες στο χώρο του χρήστη [28]. Αυτό επιτυγχάνεται με την βοήθεια μιας μηχανής εκτέλεσης (runtime engine) ανοιχτού κώδικα (π.χ docker runtime engine), παρόμοια αλλά διαφορετική με τον ελεγκτή εικονικών μηχανών, που εγκαθίσταται στο λειτουργικό σύστημα και αποτελεί το μέσο για τον διαμοιρασμό του λειτουργικού συστήματος, των βιβλιοθηκών και των διαφόρων αρχείων μεταξύ των περιεκτών [32]. Ωστόσο, η πολυεπίπεδη αρχιτεκτονική των εικόνων περιεκτών εισάγει προβλήματα πολυπλοκότητας και πιθανών προβλήματα επιδόσεων [31]. Επιπλέον, η κοινή χρήση του πυρήνα σημαίνει ότι, για παράδειγμα, οι περιέκτες των Windows δεν μπορούν να εκτελεστούν σε ένα υπολογιστικό σύστημα Linux, επίσης οι περιέκτες δεν απομονώνουν τους πόρους τόσο αποτελεσματικά όσο οι hypervisors, θέτοντας δυνητικά κινδύνους ασφάλειας λόγω της έκθεσης του πυρήνα του host [30]. Για να συνοψίσουμε, οι περιέκτες απαιτούν λιγότερο χώρο στο δίσκο και στη μνήμη από τις εικονικές μηχανές, είναι μικρότεροι σε μέγεθος αποθήκευσης, καταναλώνουν λιγότερη ενέργεια και είναι έτοιμοι για χρήση σε λίγα δευτερόλεπτα από την εκκίνηση τους. Περιέχουν όλες τις απαραίτητες εξαρτήσεις, όπως κώδικα, runtime, εργαλεία συστήματος, βιβλιοθήκες συστήματος και ρυθμίσεις για την εκτέλεση μιας εφαρμογής [28]. Αυτή η αποτελεσματικότητα και η μικρότερη ανάγκη για πόρους έχουν κάνει την εικονικοποίηση περιεκτών όλο και πιο δημοφιλή, προσφέροντας ισχυρά οφέλη απόδοσης και ασφάλειας [29]. Σημαντικό είναι να τονίσουμε πως υπήρχαν διάφορες μορφές εικόνων ανάλογα τη μηχανή εκτέλεσης που χρησιμοποιούνταν. Το 2015

όμως, δημιουργήθηκε η Open Container Initiative (OCI) όπου καθόριζε συγκεκριμένα πρότυπα και από τότε πολλές μηχανές εκτέλεσης τα έχουν υιοθετήσει [33].



Εικόνα 10 - Περιεχόμενα Περιεκτών [30]

Οι περιέκτες χωρίζονται σε δύο κατηγορίες, τους περιέκτες εφαρμογής ή διεργασίας (application/process containers) και τους περιέκτες συστήματος (system containers). Η πρώτη κατηγορία είναι εκείνη που μας έρχεται στο μυαλό όταν ακούμε την λέξη περιέκτες, καθώς αφορά όλα αυτά που έχουμε αναλύσει μέχρι στιγμής στο συγκεκριμένο υποκεφάλαιο. Σε αυτή την κατηγορία κατατάσσονται οι μηχανές εκτέλεσης Docker, containerd, Hyper-V & Windows Containers, LXC, runC κ.α. Η δεύτερη κατηγορία των περιεκτών συστήματος είναι παρόμοια σε λειτουργία με αυτή των εικονικών μηχανών, αφού περιλαμβάνουν ένα σχεδόν ολόκληρο λειτουργικό σύστημα εκτός του πυρήνα καθώς εκμεταλλεύονται και αυτοί τον ήδη υπάρχον, χωρίς όμως την περαιτέρω επιβάρυνση που υπάρχει σε αυτές λόγω της εικονικοποίησης. Η διαχείριση τους είναι επίσης ίδια, με εκείνη των εικονικών μηχανών πράγμα που σημαίνει ότι μπορείς να εγκαταστήσεις ενημερώσεις και πακέτα λογισμικού, να διαχειριστείς υπηρεσίες, να ορίσεις πολιτικές αντιγράφων ασφαλείας (backup policies) κ.α. Παραδείγματα περιεκτών συστήματος είναι LXC, BSD jails και Solaris Zones [34]. Με κάθε εκκίνηση ενός περιέκτη συστήματος έχουμε ένα νέο λειτουργικό, μικρό σε μέγεθος, καλύτερο σε απόδοση, που μπορεί να εκτελέσει πολλές διεργασίες ταυτόχρονα, χωρίς την χρήση εικονικοποίησης και με μοναδικό περιορισμό τη συμβατότητα των εφαρμογών με τον πυρήνα του host λειτουργικού συστήματος. Επιπρόσθετα, χρησιμοποιώντας διαφορετικούς χώρους χρήστη μέσα στον ίδιο περιέκτη δημιουργείται ένα επιπλέον επίπεδο απομόνωσης μεταξύ των διεργασιών κάθε χώρου.

Επομένως, κρίνεται βέλτιστο να χρησιμοποιούνται περιέκτες συστήματος, αντί εικονικών μηχανών, πλην των περιπτώσεων που θέλουμε να εκκινήσουμε διαφορετικό λειτουργικό σύστημα ή η λειτουργικότητα που χρειαζόμαστε δεν υποστηρίζεται από τον πυρήνα του συστήματος μας [35]. Επίσης, οι περιέκτες διεργασίας είναι ιδανικοί για την υποστήριξη της αρχιτεκτονικής μικροϋπηρεσιών (microservices architecture), κατά την οποία οι εφαρμογές αποτελούνται από πολλές συνδεδεμένες, μικρότερες και ανεξάρτητα αναπτυσσόμενες υπηρεσίες. Λόγω της εύκολης μεταφοράς και εκτέλεσης τους σε διάφορα περιβάλλοντα, αποτελούν ιδανική αρχιτεκτονική για εφαρμογές σε υβριδικά υπολογιστικά νέφη (hybrid clouds) και πολυ-υπολογιστικά νέφη (multi-clouds) [36].

1.8 Πλεονεκτήματα

Η εικονικοποίηση προσφέρει πολλά πλεονεκτήματα, συμπεριλαμβανομένης της διαχειριζόμενης εκτέλεσης και απομόνωσης λειτουργικών συστημάτων και διεργασιών, φορητότητας των εικονικών μηχανών, αποτελεσματικής χρήσης των πόρων, εύκολης ανάκτησης δεδομένων, ασφαλούς πλατφόρμας για τον έλεγχο των αλλαγών του λογισμικού και βελτιωμένη ασφάλεια και αξιοπιστία του συστήματος [17]. Μειώνει επίσης την ανάγκη για φυσικές μηχανές, όπως υπολογιστές και servers, τους οποίους χρησιμοποιούν τα κέντρα δεδομένων για την αποθήκευση δεδομένων και λογισμικού. Αυτό οδηγεί σε αυξημένα κέρδη για τις εταιρείες και ωφελεί τους πελάτες μειώνοντας το κόστος που σχετίζεται με τους servers, την ηλεκτρική ενέργεια και την ψύξη των συστημάτων, με αποτέλεσμα σημαντικές μειώσεις στο κόστος για τη χρήση της νεφοϋπολογιστικής [19]. Επίσης, γίνεται ευκολότερη η επεκτασιμότητα και η διαχείριση των servers, καθώς δεν χρειάζεται κάποιο φυσικό πρόσωπο να εγκαταστήσει ή να αφαιρέσει υλικό από τον εκάστοτε server ανάλογα με τις απαιτήσεις της εφαρμογής για την οποία θα χρησιμοποιηθεί. Στην πραγματικότητα είναι ελάχιστες οι φορές που γίνονται φυσικές αλλαγές στο υλικό του μηχανήματος, αφού η δέσμευση και απελευθέρωση των πόρων του γίνεται μέσω των ρυθμίσεων της εικονικής μηχανής [37]. Άλλη μια χρησιμότητα των εικονικών μηχανών είναι πως επειδή ο ελεγκτής τις διαχωρίζει από το υλικό, το λογισμικό δεν βασίζεται πλέον σε συγκεκριμένες συσκευές υλικού ή προγράμματα οδήγησης (drivers) για να λειτουργήσει [38].

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΕΦΟΥΣ & ΑΚΡΟΥ
(Cloud & Edge Computing)

2.1 Νεφοϋπολογιστική (Cloud Computing)

Διάφορες αναπτυσσόμενες επιχειρήσεις αναγκάζονται να αγοράζουν τεχνολογικό εξοπλισμό (π.χ επεξεργαστική ισχύ, αποθηκευτικό χώρο, δικτυακές υπηρεσίες), καθώς οι ανάγκες εξυπηρέτησης των πελατών τους αυξάνονται. Νέα μηχανήματα και τεχνολογίες σημαίνουν κόστος αγοράς, ανάγκη για περισσότερο χώρο στέγασης των συστημάτων, μεγαλύτερα ή περισσότερα συστήματα ψύξης, μεγαλύτερο κόστος συντήρησης, περισσότερη κατανάλωση ενέργειας και πρόσληψη εξειδικευμένου ή εκ νέου εκπαίδευση προσωπικού. Επίσης ο χρόνος που χρειάζεται για την εύρεση του κατάλληλου εξοπλισμού και της εγκατάστασής του είναι πολύ σημαντικός. Το κόστος με το οποίο επιβαρύνονται αυτές οι επιχειρήσεις, μπορεί να είναι πολύ υψηλό και παρόλα αυτά να μην αξιοποιούν τον εξοπλισμό αυτό στο εκατό τοις εκατό. Προκειμένου να αποφευχθεί αυτή η τεράστια δαπάνη, οι εν λόγω εταιρείες, τείνουν να νοικιάζουν υπηρεσίες και συστήματα από τους Παρόχους Νεφοϋπολογιστικών Υπηρεσιών (CSPs), πληρώνοντας συγκεκριμένα τις υπηρεσίες και τους πόρους που χρειάζονται. Ως επακόλουθο, τα δεδομένα μαζεύονται και επεξεργάζονται σε μεγάλα κέντρα δεδομένων, δημιουργώντας ένα κεντρικοποιημένο σύστημα διαχείρισης δεδομένων.

Οι νεφοϋπολογιστικές υπηρεσίες είναι βασισμένες στην έννοια της κοινής χρήσης υπολογιστικών πόρων, λογισμικού και πληροφοριών, κατ’ απαίτηση μέσω του διαδικτύου. Οι πάροχοι δίνουν πρόσβαση σε εικονικές ομάδες κοινόχρηστων πόρων, συμπεριλαμβανομένων των υπηρεσιών υπολογισμού, αποθήκευσης και δικτύωσης, οι οποίες βρίσκονται σε απομακρυσμένους servers που ανήκουν και διαχειρίζονται εκείνοι, έναντι μηνιαίας συνδρομής ή χρέωσης ανάλογα με τη χρήση. Ένα από τα πολλά πλεονεκτήματα της νεφοϋπολογιστικής είναι πως πληρώνεις μόνο ό,τι χρησιμοποιείς. Αυτό έχει ως αποτέλεσμα, την ταχύτερη και πιο αποτελεσματική επέκταση των οργανισμών, χωρίς να χρειάζεται να αγοράζουν και να συντηρούν τα δικά τους φυσικά κέντρα δεδομένων και servers [39].

Σύμφωνα με το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST), ως υπολογιστικό νέφος, ορίζεται το μοντέλο που επιτρέπει την εύκολη, βολική κατ' απαίτηση πρόσβαση δικτύου σε μια κοινόχρηστη ομάδα διαμορφώσιμων υπολογιστικών πόρων (π.χ δίκτυα, servers, χώροι αποθήκευσης, εφαρμογές και υπηρεσίες), που μπορεί να παρασχεθεί και να απελευθερωθεί γρήγορα με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδραση παρόχου υπηρεσιών. Το μοντέλο αυτό αποτελείται από πέντε βασικά χαρακτηριστικά (on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service), τρία μοντέλα υπηρεσιών (IaaS, PaaS, SaaS) και τέσσερα μοντέλα ανάπτυξης (private, community, public and hybrid cloud) [40].

2.1.1 Ιστορική Αναδρομή

Το κόστος αγοράς υλικού γύρω στις αρχές του 90' ήταν εξωπραγματικό. Καθώς το διαδίκτυο γινόταν όλο και πιο προσβάσιμο, περισσότεροι άνθρωποι ήθελαν να το χρησιμοποιούν. Επομένως έπρεπε να βρεθεί μια λύση, ώστε οι τιμές να ελαχιστοποιηθούν και να συμβαδίσουν με την οικονομία της εποχής. Ένας τρόπος με τον οποίο επιτεύχθηκε αυτό, ήταν η εικονικοποίηση εξυπηρετητών χρησιμοποιώντας την ίδια λειτουργικότητα με εκείνη της κοινόχρηστης πρόσβασης στα υπολογιστικά συστήματα (mainframes) του 60'. Για παράδειγμα, ας υποθέσουμε πως μια εταιρεία χρειάζεται δεκατρείς φυσικούς servers για τις ανάγκες ιστοσελίδων και εφαρμογών. Εικονικοποιώντας αυτά τα δεκατρία διαφορετικά συστήματα, μπορούν να χωριστούν σε δύο φυσικούς servers. Με αυτόν τον τρόπο εξοικονομούνται έξοδα υποδομών και μειώνεται η ποσότητα του υλικού που θα χρειαζόταν να αγοράσει η εταιρεία, αν δεν υπήρχε η εικονικοποίηση. Καθώς το κόστος υλικού των εξυπηρετητών μειώθηκε, περισσότεροι χρήστες μπορούσαν να αγοράσουν τους δικούς τους αποκλειστικούς servers. Δημιουργήθηκε όμως ένα άλλο πρόβλημα, ένας server δεν ήταν αρκετός για να προσφέρει όλους τους απαραίτητους πόρους. Αυτό είχε ως αποτέλεσμα η αγορά, αντί να προσπαθεί να διαχωρίσει τους πόρους ενός συστήματος για την καλύτερη αξιοποίηση του, να ψάχνει πλέον τρόπους να συνδυάσει τους servers ώστε να φαίνονται σαν ένα σύνολο διαθέσιμων πόρων που θα μπορούσε να κατατμιστεί ανάλογα με τις εκάστοτε ανάγκες. Η λύση βρέθηκε στην εγκατάσταση ενός λογισμικού, γνωστού από προηγούμενο κεφάλαιο, που ονομάζεται ελεγκτής εικονικών μηχανών σε πολλαπλούς κόμβους (servers). Ο

ελεγκτής μπορεί να διαμορφωθεί με τέτοιο τρόπο ώστε, το σύστημα να παρουσιάζει όλους τους πόρους του περιβάλλοντος σαν να βρίσκονται σε έναν μόνο φυσικό κόμβο. Για την οπτικοποίηση αυτού του περιβάλλοντος χρησιμοποιήθηκε ο όρος «υπολογιστικό νέφος», αφού αποτελούνταν από μια νεφελώδη μάζα υπολογιστικών πόρων. Σε αυτά τα περιβάλλοντα ήταν εύκολο να προσθέσεις πόρους, καθώς το μόνο που χρειαζόταν ήταν να προσθέσεις άλλον ένα εξυπηρετητή και να τον διαμορφώσεις έτσι, ώστε να γίνει μέρος του ευρύτερου συστήματος. Καθώς οι τεχνολογίες και οι ελεγκτές εικονικών μηχανών βελτιώθηκαν, όσον αφορά την κοινή χρήση και την παράδοση πόρων, πολλές επιχειρηματικές εταιρείες αποφάσισαν να ακολουθήσουν την τακτική της νεφοϋπολογιστικής. Προκειμένου όλοι να έχουν πρόσβαση στα πλεονεκτήματα της, οι εταιρείες αυτές, νοίκιαζαν στους χρήστες τους, πόρους που εκείνοι χρειάζονταν αλλά δεν μπορούσαν να αγοράσουν, από τους διαθέσιμους της ευρύτερης ομάδας πόρων, που υπήρχαν στο νέφος (cloud). Έτσι δημιουργήθηκε το μοντέλο της νεφοϋπολογιστικής και των κατ' απαίτηση υπηρεσιών [41].

2.1.2 Χαρακτηριστικά

Για να θεωρήσουμε ένα μοντέλο ως μοντέλο νέφους, θα πρέπει ο χρήστης να μπορεί να αιτηθεί υπολογιστικές δυνατότητες, όπως χρόνο χρήσης του server και δικτυακό χώρο αποθήκευσης, αυτόματα χωρίς να απαιτείται ανθρώπινη αλληλεπίδραση με τους παρόχους υπηρεσιών (On-demand self-service). Οι δυνατότητες αυτές, να είναι διαθέσιμες μέσω του δικτύου και προσβάσιμες μέσω συγκεκριμένων μηχανισμών, που προωθούν τη χρήση διαφόρων συσκευών όπως κινητά, φορητούς υπολογιστές κ.α. (broad network access). Οι υπολογιστικοί πόροι του παρόχου να διατίθενται, χρησιμοποιώντας ένα μοντέλο όπου πολλαπλοί καταναλωτές μπορούν να πληρώνουν γι’ αυτούς, ενώ διαφορετικοί φυσικοί και εικονικοί πόροι ανατίθενται και ξανά ανατίθενται, ανάλογα με τις απαιτήσεις των καταναλωτών. Σε γενικές γραμμές, οι καταναλωτές δεν γνωρίζουν ή έχουν έλεγχο της ακριβής τοποθεσίας των πόρων που τους ανατίθενται, αλλά μπορεί να είναι σε θέση να την προσδιορίσουν σε επίπεδο χώρας, πολιτείας ή κέντρου δεδομένων (resource pooling). Οι διαθέσιμες υπολογιστικές δυνατότητες, θα πρέπει να φαίνονται απεριόριστες και να μπορούν να αποδοθούν σε οποιαδήποτε ποσότητα ανά πάσα στιγμή στον καταναλωτή, ευέλικτα και σε κάποιες περιπτώσεις αυτόματα για να κλιμακώνονται γρήγορα ανάλογα με την ζήτηση (rapid

elasticity). Τέλος θα πρέπει τα συστήματα του νέφους να παρακολουθούν, να ελέγχουν, να βελτιστοποιούν και να αναφέρουν αυτόματα την χρήση των πόρων, παρέχοντας διαφάνεια τόσο για τον πάροχο όσο και για τον καταναλωτή της χρησιμοποιούμενης υπηρεσίας (measured service) [40].

2.1.3 Μοντέλα Ανάπτυξης (Deployment Models)

Τα μοντέλα ανάπτυξης στην νεφοϋπολογιστική, αναφέρονται στους τρόπους με τους οποίους παρέχονται και διατίθενται οι πόροι του υπολογιστικού νέφους στους χρήστες. Υπάρχουν τέσσερα κύρια μοντέλα ανάπτυξης στο cloud computing:

Δημόσια Δίκτυα Νέφους (Public Clouds): Το δημόσιο υπολογιστικό νέφος παρέχεται για χρήση από το ευρύ κοινό. Μπορεί να ανήκει, να διαχειρίζεται και να λειτουργεί από επιχείρηση, ακαδημαϊκό ή κυβερνητικό οργανισμό ή από κάποιον συνδυασμό αυτών. Η υποδομή του βρίσκεται στις εγκαταστάσεις του παρόχου [40]. Γνωστές δημόσια διαθέσιμες πλατφόρμες είναι, η Amazon Web Services, η Microsoft Azure και η Google Cloud Platform. Μέσω του υπολογιστικού νέφους, οι χρήστες μπορούν να αποκτήσουν πρόσβαση σε υπολογιστικούς πόρους (όπως επεξεργαστική ισχύ και αποθηκευτικό χώρο) και να τους χρησιμοποιήσουν για να αναπτύξουν, να διαχειριστούν και να εκτελέσουν εφαρμογές και υπηρεσίες έναντι συνδρομής ή μοντέλων πληρωμής ανά χρήση. Το δημόσιο υπολογιστικό νέφος προσφέρει ευελιξία, κλιμακωσιμότητα και εξοικονόμηση κόστους στους χρήστες, επιτρέποντας τους να αναπτύσσουν και να εκτελούν εφαρμογές με μεγαλύτερη αποδοτικότητα και αποτελεσματικότητα [42].

Ιδιωτικά Δίκτυα Νέφους (Private Clouds): Το ιδιωτικό υπολογιστικό νέφος, μπορεί να παρέχεται αποκλειστικά σε έναν οργανισμό με πολλούς καταναλωτές, όπως επιχειρηματικές μονάδες. Μπορεί να ανήκει, να διαχειρίζεται και να λειτουργεί από τον ίδιο τον οργανισμό, από κάποιον άλλο ή συνδυασμό και των δύο, και η υποδομή του υπάρχει εντός ή εκτός των εγκαταστάσεων [40]**.** Σε αντίθεση με το δημόσιο νέφος, το ιδιωτικό νέφος εξυπηρετεί μόνο τον συγκεκριμένο οργανισμό και δεν είναι προσβάσιμο από το κοινό. Παρέχει πλήρη έλεγχο και ασφάλεια των δεδομένων, καθώς και τη δυνατότητα προσαρμογής των υπηρεσιών σύμφωνα με τις ανάγκες της εταιρείας. Επίσης, το ιδιωτικό νέφος προσφέρει

καλύτερη απόκριση και απόδοση σε εφαρμογές που απαιτούν υψηλό επίπεδο ασφάλειας ή προσαρμογής [42].

Κοινοτικά Δίκτυα Νέφους (Community Clouds): Το κοινοτικό υπολογιστικό νέφος, διατίθεται για αποκλειστική χρήση από μια συγκεκριμένη κοινότητα πελατών από οργανισμούς με κοινά ενδιαφέροντα, όπως απαιτήσεις ασφαλείας ή πολιτικής και συμμόρφωσης. Το κοινοτικό νέφος μπορεί να ανήκει, να διαχειρίζεται και να το λειτουργεί η ίδια η ομάδα των οργανισμών, κάποιος άλλος οργανισμός ή κάποιος συνδυασμός αυτών και η υποδομή του μπορεί να υπάρχει είτε εντός, είτε εκτός των εγκαταστάσεων τους [40]. Οι χρήστες μοιράζονται τους πόρους και τις δυνατότητες του νέφους, με αποτέλεσμα να προσφέρεται μεγαλύτερη αποδοτικότητα και οικονομία, όσον αφορά το κόστος της επέκτασης των υποδομών του κάθε οργανισμού. Επίσης, το κοινοτικό νέφος παρέχει επιπλέον ασφάλεια και απόρρητο, καθώς και προσαρμογή των υπηρεσιών στις ανάγκες των μελών της ομάδας [42].

Υβριδικά Δίκτυα Νέφους (Hybrid Clouds): Αυτή η κατηγορία είναι μια σύνθεση δύο ή περισσότερων διακριτών υποδομών νέφους (ιδιωτική, κοινοτική ή δημόσια), που παραμένουν μοναδικές οντότητες, αλλά συνδέονται μεταξύ τους με τυποποιημένη (standardized) ή αποκλειστική (proprietary) τεχνολογία, που επιτρέπει τη φορητότητα δεδομένων και εφαρμογών [40]. Ένα παράδειγμα χρήσης της, είναι η διαχείριση αιχμών φόρτου εργασίας για την εξισορρόπηση του φορτίου μεταξύ νεφών (cloud bursting for load balancing between cloud), όπου ο φόρτος εργασίας ενός οργανισμού διαχειρίζεται από την υποδομή του ιδιωτικού νέφους κατά τη διάρκεια των κανονικών περιόδων. Ωστόσο, σε περίπτωση απότομης αύξησης του φόρτου εργασίας, μπορεί να μεταφορτωθεί στο δημόσιο νέφος, για να αντιμετωπιστεί η αυξημένη ζήτηση υπηρεσιών. Αυτή η τεχνική επιτρέπει στους οργανισμούς να διατηρούν το βασικό επίπεδο των πόρων τους και να επεκτείνουν την υποδομή τους όταν η ζήτηση είναι υψηλή, παρέχοντας έτσι μια δυνατότητα κλιμάκωσης κατόπιν απαίτησης. Τα υβριδικά δίκτυα επιτρέπουν την αξιοποίηση της ασφάλειας και του ελέγχου που παρέχουν τα ιδιωτικά νέφη, καθώς και την ευελιξία και την διαθεσιμότητα που προσφέρουν τα δημόσια. Με αυτό τον τρόπο, οι χρήστες μπορούν να δημιουργήσουν μια προσαρμοσμένη λύση, που ανταποκρίνεται στις ανάγκες τους, ωφελούμενοι από τα θετικά των δύο προσεγγίσεων [42].

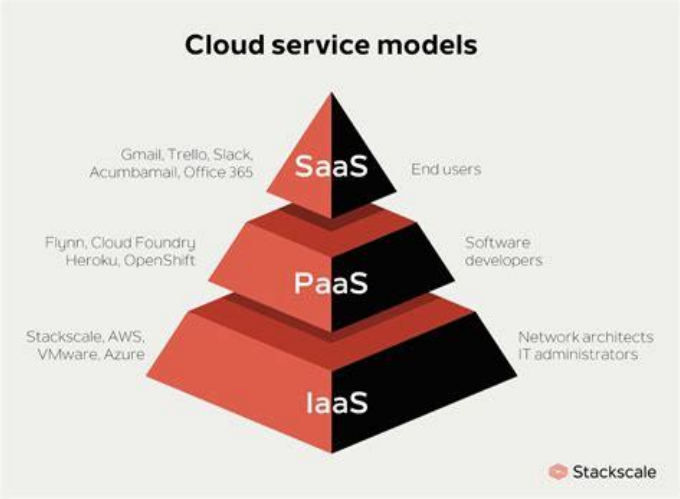
Πέρα από τις βασικές κατηγορίες νεφών, μπορούμε να δημιουργήσουμε άλλες δύο κατηγορίες, οι οποίες αποτελούνται από συνδυασμούς των κύριων κατηγοριών. Η πρώτη είναι τα πολυ-νέφη (multi clouds) και αναφέρεται στη χρήση πολλαπλών υπολογιστικών νεφών ίδιας κατηγορίας (π.χ δημόσια) από διαφορετικούς παρόχους. Αυτό μπορεί να κυμαίνεται από τη χρήση διαφορετικών εφαρμογών λογισμικού, ως υπηρεσία (SaaS) διαφορετικών παρόχων, έως τη χρήση πολλαπλών υπηρεσιών νέφους, όπως λογισμικού ως υπηρεσία, πλατφόρμα ως υπηρεσία (PaaS) και υποδομή ως υπηρεσία (IaaS), από πολλαπλούς παρόχους δημόσιων νεφών. Η δεύτερη κατηγορία είναι τα υβριδικά πολυ-νέφη (hybrid multi clouds), τα οποία είναι ένας συνδυασμός πολλαπλών δημόσιων και ιδιωτικών νεφών διαφορετικών παρόχων. Οι οργανισμοί υιοθετούν την προσέγγιση του πολυ-νέφους, για να αποφύγουν το «vendor lock-in» (βλ. παρακάτω), να αυξήσουν τις επιλογές υπηρεσιών και να αποκτήσουν πρόσβαση σε περισσότερη καινοτομία. Ωστόσο, η χρήση πολλαπλών νεφών, μπορεί να περιπλέξει τη διαχείριση του περιβάλλοντος, ιδίως όταν καθένα από αυτά χρησιμοποιεί διαφορετικά εργαλεία διαχείρισης, ρυθμούς μετάδοσης δεδομένων και πρωτόκολλα ασφαλείας. Οι πλατφόρμες διαχείρισης πολυ-νέφους προσφέρουν ένα περιβάλλον που παρέχει ορατότητα στα πολλαπλά νέφη, επιτρέποντας στις ομάδες να διαχειρίζονται τα έργα, τις αναπτύξεις (deployments), τις συστάδες υπολογιστών (clusters), τους κόμβους και τις απειλές κυβερνοασφάλειας [42].

Vendor lock-in: Ο όρος αναφέρεται στην κατάσταση όπου μια εταιρεία εξαρτάται από έναν συγκεκριμένο προμηθευτή, για προϊόντα ή υπηρεσίες και δεν μπορεί εύκολα να μεταβεί σε άλλο, χωρίς σημαντικό κόστος [43]. Αυτό μπορεί να συμβεί, όταν ένας προμηθευτής παρέχει ιδιόκτητες (proprietary) τεχνολογίες, που καθιστούν δύσκολη ή δαπανηρή τη μετάβαση του πελάτη σε προϊόν ή υπηρεσία κάποιου ανταγωνιστή. Ως αποτέλεσμα, περιορίζεται η ευελιξία και η διαπραγματευτική δύναμη του πελάτη και μπορεί επίσης να αποθαρρύνει τον ανταγωνισμό και την καινοτομία στην αγορά.

2.1.4 Μοντέλα Υπηρεσιών (Service Models)

Όπως αναφέρθηκε παραπάνω, σύμφωνα με το NIST, υπάρχουν τρεις κύριοι τύποι μοντέλων υπηρεσιών υπολογιστικού νέφους, που μπορούν να επιλεχθούν ανάλογα με το επίπεδο ελέγχου, ευελιξίας και διαχείρισης που χρειάζεται η εκάστοτε επιχείρηση. Τα μοντέλα

αυτά, διατίθενται από τους παρόχους υπηρεσιών νέφους, μέσω των μοντέλων ανάπτυξης που εξηγήσαμε προηγουμένως και αφορούν στο επίπεδο ελέγχου και διαχείρισης της υποδομής που έχει επιλέξει ο καταναλωτής.



Εικόνα 11 - Cloud Service Models
https://medium.com/@danialzafar/demystifying-cloud-computing-saas-vs-paas-vs-iaas-80b05729b93

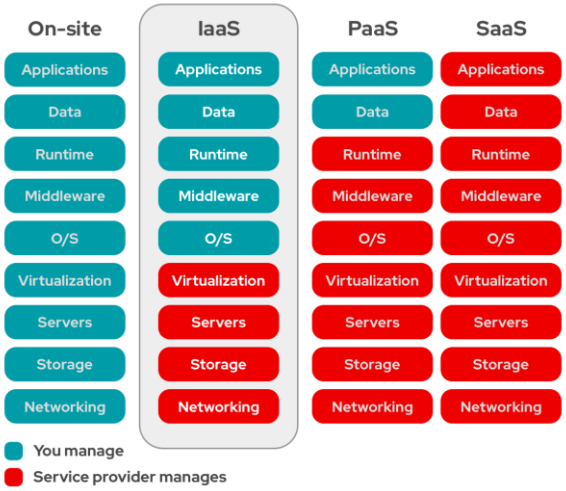
Λογισμικό ως Υπηρεσία (Software as a Service): Το Λογισμικό ως Υπηρεσία, είναι ένα μοντέλο υπηρεσιών υπολογιστικού νέφους, όπου το λογισμικό υπάρχει στο νέφος και οι χρήστες έχουν πρόσβαση σε αυτό, μέσω ενός προγράμματος περιήγησης στο διαδίκτυο ή μιας εφαρμογής. Ο καταναλωτής δεν διαχειρίζεται ή ελέγχει την υποκείμενη υποδομή, συμπεριλαμβανομένου του δικτύου, των διακομιστών, των λειτουργικών συστημάτων και του αποθηκευτικού χώρου, με πιθανή εξαίρεση αυτή των περιορισμένων ρυθμίσεων χρήστη για τη διαμόρφωση της εφαρμογής [40]. Τόσο η υπηρεσία, όσο και η υποδομή, διαχειρίζονται και συντηρούνται από τον πάροχο υπηρεσιών. Οι χρήστες συνήθως, καταβάλλουν μηνιαίο ή ετήσιο ποσό συνδρομής, ενώ ορισμένοι μπορεί να τιμολογούνται με βάση τη χρήση. Το Λογισμικό ως Υπηρεσία, προσφέρει πλεονεκτήματα όπως, αυτόματες αναβαθμίσεις και προστασία από την απώλεια δεδομένων. Αποτελεί το κύριο μοντέλο παροχής υπηρεσιών, για το μεγαλύτερο μέρος του εμπορικού λογισμικού σήμερα.

Πλατφόρμα ως Υπηρεσία (Platform as a Service): Η Πλατφόρμα ως Υπηρεσία, παρέχει στον καταναλωτή την δυνατότητα να αναπτύσσει στην υποδομή νέφους εφαρμογές, οι οποίες έχουν δημιουργηθεί με τη χρήση γλωσσών προγραμματισμού, βιβλιοθηκών, υπηρεσιών και εργαλείων, που υποστηρίζονται από τον πάροχο. Ο καταναλωτής δεν

διαχειρίζεται ούτε ελέγχει την υποκείμενη υποδομή νέφους, συμπεριλαμβανομένου του δικτύου, των servers, των λειτουργικών συστημάτων ή του αποθηκευτικού χώρου, αλλά έχει τον έλεγχο των εφαρμογών που αναπτύσσονται και ενδεχομένως των ρυθμίσεων διαμόρφωσης του περιβάλλοντος φιλοξενίας των εφαρμογών [40]. Η Πλατφόρμα, ως Υπηρεσία, δίνει στους προγραμματιστές την ελευθερία να επικεντρωθούν στον κώδικα των εφαρμογών, παρέχοντας τους όλα όσα χρειάζονται για την ανάπτυξη τους, συμπεριλαμβανομένης της υποδομής, των εργαλείων ανάπτυξης λογισμικού, των λειτουργικών συστημάτων και άλλων, χωρίς να χρειάζεται να ασχολούνται με την ενημέρωση και την συντήρηση τους. Αυτό έχει ως αποτέλεσμα, την απλούστερη, ταχύτερη και ασφαλέστερη ανάπτυξη εφαρμογών.

Υποδομή ως Υπηρεσία (Infrastructure as a Service): Η Υποδομή ως Υπηρεσία, είναι ένα μοντέλο υπολογιστικού νέφους που επιτρέπει σε επιχειρήσεις και ιδιώτες να έχουν πρόσβαση σε πόρους υποδομής κατα παραγγελία, όπως υπολογιστική ισχύ, αποθηκευτικό χώρο, δικτύωση και εικονικοποίηση, μέσω του νέφους για την ανάπτυξη και εκτέλεση λογισμικού, συμπεριλαμβανομένων λειτουργικών συστημάτων και εφαρμογών. Ωστόσο, ο καταναλωτής δεν διαχειρίζεται ούτε ελέγχει την υποκείμενη υποδομή, αλλά έχει μόνο τον έλεγχο των λειτουργικών συστημάτων, του αποθηκευτικού χώρου και των εφαρμογών που αναπτύσσονται, με περιορισμένο έλεγχο επιλεγμένων στοιχείων δικτύωσης (π.χ τείχος προστασίας) [40]. Το μοντέλο αυτό είναι ελκυστικό, επειδή εξαλείφει την ανάγκη των επιχειρήσεων να επενδύσουν σε φυσικούς χώρους, προμήθεια εξοπλισμού και εξειδικευμένο προσωπικό. Αντίθετα, οι οργανισμοί μπορούν απλώς να αγοράζουν και να χρησιμοποιούν πόρους, βασιζόμενοι σε ένα μοντέλο όπου χρεώνονται μόνο για τις υπηρεσίες και τους πόρους που χρησιμοποιούν, αυξάνοντας ή μειώνοντας την κλίμακα της υποδομής τους ανάλογα με τις

ανάγκες τους, χωρίς τον κίνδυνο δημιουργίας μιας υποδομής, που δεν εκμεταλλεύονται πλήρως ή που δεν τους αρκεί.



Εικόνα 12 - Πρόσβαση Σε Πόρους Ανα Μοντέλο
https://www.redhat.com/en/topics/cloud-computing/what-is-saas

2.1.5 Πλεονεκτήματα

Σε σύγκριση με το παραδοσιακό μοντέλο, όπου οι εταιρείες κατέχουν και διατηρούν φυσικά κέντρα δεδομένων και servers, η νεφοϋπολογιστική, προσφέρει πολυάριθμα οφέλη τόσο στις επιχειρήσεις, όσο και στους ιδιώτες. Ακολουθούν ορισμένα από τα βασικά πλεονεκτήματα της:

Εξοικονόμηση κόστους: Οι χρήστες πληρώνουν μόνο για τους υπολογιστικούς πόρους που χρησιμοποιούν, επομένως μειώνει την ανάγκη για φυσικό υλικό, κέντρα δεδομένων και εξειδικευμένο προσωπικό. Αυτό σημαίνει ότι, οι επιχειρήσεις μπορούν να εξοικονομήσουν κόστος που σχετίζεται με τη συντήρηση, τις αναβαθμίσεις, τις αγορές εξοπλισμού και την εκπαίδευση ή πρόσληψη προσωπικού [42].

Επεκτασιμότητα: Προσφέρει στις επιχειρήσεις τη δυνατότητα να αυξάνουν ή να μειώνουν την κλίμακα των υποδομών τους, ανάλογα με τις υπολογιστικές τους ανάγκες. Έτσι εξαλείφεται η ανάγκη αγοράς πλεονάζουσας χωρητικότητας για τις περιόδους αιχμής, βελτιστοποιώντας τη χρήση των πόρων. Αυτό σημαίνει, ότι μπορούν εύκολα να προσαρμόσουν τους πόρους που χρειάζονται, καθώς η επιχείρησή τους μεγαλώνει ή αλλάζει [42].

Ευελιξία: Επιτρέπει την πρόσβαση σε δεδομένα και εφαρμογές, από οπουδήποτε με σύνδεση στο διαδίκτυο. Επομένως, οι χρήστες μπορούν να εργάζονται απομακρυσμένα, να συνεργάζονται πιο εύκολα και να έχουν πρόσβαση σε πληροφορίες κάθε στιγμή [39].

Ασφάλεια: Οι πάροχοι υπηρεσιών νέφους, απασχολούν ειδικές ομάδες που προσφέρουν ισχυρά μέτρα και αντίγραφα ασφαλείας δεδομένων, προστατεύοντας τα δεδομένα από απώλεια, κλοπή ή καταστροφή. Επιπλέον, τα δεδομένα μπορούν να κρυπτογραφηθούν κατά τη μεταφορά και την αποθήκευση, για να διασφαλιστεί το απόρρητο και η εμπιστευτικότητα, καθιστώντας έτσι τους κινδύνους ασφαλείας σχετικά χαμηλούς [39].

Ενισχυμένη στρατηγική αξία: Η υπολογιστική νέφους, ενισχύει τη στρατηγική αξία, παρέχοντας πρόσβαση σε τεχνολογίες και καινοτομίες αιχμής, δίνοντας τη δυνατότητα στους οργανισμούς, να αποκτήσουν ανταγωνιστικό πλεονέκτημα. Για παράδειγμα, σε κλάδους που αντιμετωπίζουν πελάτες, όπως το λιανικό εμπόριο και οι τραπεζικές συναλλαγές, οι εικονικοί βοηθοί με τεχνητή νοημοσύνη που αναπτύσσονται στο νέφος, βελτιώνουν τους χρόνους απόκρισης και επιτρέπουν στις ομάδες των οργανισμών να επικεντρώνονται σε άλλες εργασίες [42].

Συνολικά, η νεφοϋπολογιστική, παρέχει μια πιο οικονομική, επεκτάσιμη, προσιτή και ευέλικτη λύση πληροφορικής, για επιχειρήσεις και ιδιώτες.

2.2 Υπολογιστική στο Άκρο (Edge Computing)

Η υπολογιστική στο άκρο, είναι ένα κατανεμημένο μοντέλο που φέρνει την διαχείριση, ανάλυση, επεξεργασία και αποθήκευση των δεδομένων, όσο πιο κοντά γίνεται στις συσκευές που τα παράγουν (π.χ κάμερες, αισθητήρες), κατ' επέκταση και στις φυσικές τοποθεσίες, αντί να σταλούν κάπου σε ένα κέντρο δεδομένων μέσω του διαδικτύου. Η αύξηση των συσκευών του Διαδικτύου των Αντικειμένων (IoT), οδηγεί σε όλο και μεγαλύτερους όγκους δεδομένων, που πλέον έχουν αρχίσει να υπερβαίνουν τις δυνατότητες του δικτύου και των υποδομών των κέντρων δεδομένων. Το νέφος με την κεντρικοποιημένη λογική του προκαλεί αυξημένη κινητικότητα στο δίκτυο και συνεπώς προβλήματα μη επαρκούς εύρους ζώνης (bandwidth), αύξησης του χρόνου καθυστέρησης (latency) και απορρήτου (privacy). Ως επακόλουθο, εφαρμογές που βασίζονται σε αυτά τα χαρακτηριστικά υπολειτουργούν και διάφορες

υποδομές, ακόμα και ζωές, μπορεί να τίθενται σε κίνδυνο. Η υπολογιστική στο άκρο, παρέχει ένα αποκεντρωμένο μοντέλο που βελτιώνει σημαντικά τα παραπάνω προβλήματα, καθώς περιλαμβάνει την κανονικοποίηση και την ανάλυση των δεδομένων, τοπικά, αποστέλλοντας μόνο τα απαραίτητα από αυτά μετά την ανάλυση στο κύριο κέντρο δεδομένων. Επομένως, δεν χρειάζεται όλος ο αρχικός όγκος δεδομένων να διασχίσει το δίκτυο για να φτάσει σε κάποιο κέντρο δεδομένων, ώστε ένα κομμάτι του να υποβληθεί σε επεξεργασία [44].

2.2.1 Ιστορική Αναδρομή της Υπολογιστικής στο Άκρο

Μέσα σε λίγες μόνο δεκαετίες, ο κόσμος της πληροφορικής έχει εξελιχθεί από τους κεντρικούς υπολογιστές (mainframes) στους υπολογιστές πελάτες/εξυπηρετητές (client/servers), στο υπολογιστικό νέφος (cloud computing) και τώρα στην υπολογιστική στο άκρο (edge computing). Παρατηρούμε ένα μοτίβο, όσον αφορά την εξέλιξη των υπολογιστικών μοντέλων, που ξεκίνησε από κεντρικοποιημένο μοντέλο, προχώρησε σε αποκεντρωμένο, επέστρεψε σε κεντρικοποιημένο και σήμερα πάλι, υοθετείται όλο και περισσότερο το αποκεντρωμένο. Η υπολογιστική στο άκρο, έχει τις ρίζες της στα δίκτυα παράδοσης περιεχομένου (CDN), που εισήγαγε η Akamai στα τέλη της δεκαετίας του 1990. Τα δίκτυα αυτά, χρησιμοποιούν κόμβους στο άκρο για την προφόρτωση (prefetch) και την προσωρινή αποθήκευση (caching) περιεχομένων του ιστού, βελτιώνοντας την απόδοση του. Η υπολογιστική στο άκρο, αποτελεί επέκταση αυτής της έννοιας και αξιοποιεί την υποδομή υπολογιστικού νέφους, για την εκτέλεση κώδικα πιο κοντά στον τελικό χρήστη, ώστε να πετύχει ταχύτερους χρόνους απόκρισης. Το 1997, ο Brian Noble και οι συνεργάτες του, απέδειξαν για πρώτη φορά τη δυνητική αξία της υπολογιστικής στο άκρο για κινητούς υπολογιστές, μεταφέροντας φόρτο του υπολογισμού σε έναν κοντινό server. Η εμφάνιση του υπολογιστικού νέφους στα μέσα της δεκαετίας του 2000, οδήγησε στην υιοθέτησή του για κινητές συσκευές, αλλά η εξάρτηση από ένα κέντρο δεδομένων στο νέφος, δεν ενδείκνυται για εφαρμογές που απαιτούν καθυστερήσεις από άκρο σε άκρο μικρότερες από μερικές δεκάδες χιλιοστά του δευτερολέπτου. Το 2009, μια ομάδα ερευνητών, έθεσε τα εννοιολογικά θεμέλια για την υπολογιστική στο άκρο, υποστηρίζοντας μια αρχιτεκτονική δύο επιπέδων. Το πρώτο επίπεδο είναι, η μη τροποποιημένη υποδομή νέφους, ενώ το δεύτερο επίπεδο αποτελείται από διάσπαρτα νέφη κατώτερου επιπέδου (cloudlets) [45]. Παρόλο που η υπολογιστική στο άκρο

παρέχει μια άνευ προηγουμένου ευκαιρία στους οργανισμούς να εκμεταλλευτούν ακόμα καλύτερα τα δεδομένα, το νέφος παραμένει απαραίτητο ως κεντρικό αποθετήριο δεδομένων και κέντρο επεξεργασίας.

2.2.2 Αρχιτεκτονική Άκρου

Ο όρος «άκρο» συχνά χρησιμοποιείται διφορούμενα, επομένως είναι χρήσιμο να γίνει διάκριση μεταξύ του «άκρου του Διαδικτύου» και του «άκρου IoT». Το άκρο του Διαδικτύου, αναφέρεται σε δικτυακή υποδομή, όπως δρομολογητές, μεταγωγείς και τείχη προστασίας, που παρέχει συνδεσιμότητα και λειτουργεί ως πύλη για το Διαδίκτυο, υποστηρίζοντας υπηρεσίες και δραστηριότητες επιχειρήσεων. Το άκρο του Διαδικτύου των Αντικειμένων, αποτελείται από τελικά σημεία του συστήματος, όπως αισθητήρες και έξυπνες συσκευές, που συλλέγουν και επικοινωνούν δεδομένα σε πραγματικό χρόνο από έξυπνα προϊόντα και υπηρεσίες. Για παράδειγμα, σε ένα IoT σύστημα παρακολούθησης της υγείας, το άκρο του IoT περιλαμβάνει αισθητήρες που συλλέγουν δεδομένα ασθενών, ενώ το άκρο του Διαδικτύου είναι η πύλη που στέλνει αυτά τα δεδομένα στο νέφος [46].

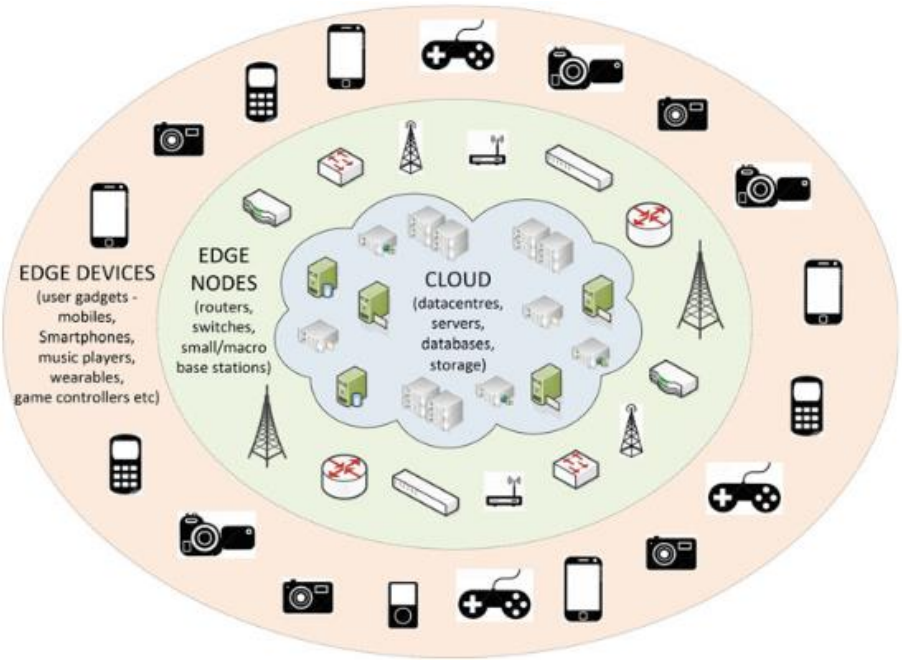
Η αρχιτεκτονική της υπολογιστικής άκρου, επεκτείνει τις υπηρεσίες νέφους στο άκρο του δικτύου, ενσωματώνοντας συσκευές άκρου μεταξύ τερματικών συσκευών και νεφοϋπολογιστικής. Αυτή η ενοποιημένη δομή δικτύου χωρίζεται σε τρία επίπεδα:

Τερματικό Επίπεδο (Terminal Layer): Το τερματικό επίπεδο περιλαμβάνει διάφορες συνδεδεμένες συσκευές, όπως αισθητήρες, έξυπνα κινητά τηλέφωνα, έξυπνα αυτοκίνητα και κάμερες. Αυτές οι συσκευές, χρησιμεύουν τόσο ως καταναλωτές, όσο και ως πάροχοι δεδομένων. Για να ελαχιστοποιηθούν οι καθυστερήσεις των παρεχόμενων υπηρεσιών, λαμβάνονται υπόψη μόνο οι δυνατότητες αντίληψης δεδομένων και όχι η υπολογιστική τους ισχύς. Κατά συνέπεια, εκατομμύρια τερματικές συσκευές, συλλέγουν ακατέργαστα δεδομένα και τα ανεβάζουν στα ανώτερα στρώματα για αποθήκευση και επεξεργασία [47].

Επίπεδο Άκρου (Edge Layer): Το επίπεδο άκρου είναι το κεντρικό στοιχείο της αρχιτεκτονικής τριών επιπέδων, τοποθετημένο μεταξύ των τερματικών συσκευών και του νέφους. Περιλαμβάνει κόμβους άκρου, όπως σταθμούς βάσης, σημεία πρόσβασης, δρομολογητές, μεταγωγείς και πύλες. Αυτό το επίπεδο διευκολύνει τη σύνδεση των

τερματικών συσκευών και χειρίζεται την αποθήκευση και τον υπολογισμό των δεδομένων από αυτές τις συσκευές. Συνδέεται επίσης στο νέφος, ανεβάζοντας επεξεργασμένα δεδομένα. Η εγγύτητα στους χρήστες επιτρέπει στο επίπεδο άκρου, να εκτελεί ανάλυση δεδομένων σε πραγματικό χρόνο και έξυπνη επεξεργασία πιο αποτελεσματικά, με περισσότερη ασφάλεια από τη νεφοϋπολογιστική [47].

Επίπεδο Νέφους (Cloud Layer): Στο υπολογιστικό πλαίσιο νέφους-άκρου, το επίπεδο υπολογιστικού νέφους παραμένει το κύριο κέντρο επεξεργασίας δεδομένων. Περιλαμβάνει servers και συσκευές αποθήκευσης υψηλής απόδοσης, με σημαντικές υπολογιστικές και αποθηκευτικές ικανότητες, οι οποίες υπερέχουν σε εργασίες που απαιτούν εκτεταμένη ανάλυση δεδομένων, όπως συντήρηση και υποστήριξη επιχειρηματικών αποφάσεων. Το κέντρο υπολογιστών νέφους, αποθηκεύει μόνιμα δεδομένα από το επίπεδο άκρου και χειρίζεται σύνθετες εργασίες ανάλυσης και επεξεργασίας, πέρα από τις δυνατότητες του επιπέδου αυτού. Επιπλέον, το νέφος μπορεί να προσαρμόσει δυναμικά τις στρατηγικές και τους αλγόριθμους ανάπτυξης του επιπέδου υπολογισμού άκρου, με βάση τις πολιτικές ελέγχου [47].



Εικόνα 13 - Απλοποιημένη Προβολή Του Παραδείγματος Υπολογιστικής Στο Άκρο [46]

2.2.2 Μοντέλα Ανάπτυξης: Κοντινό & Μακρινό Άκρο (Near & Far Edge)

Η υπολογιστική στο άκρο μπορεί να χωριστεί σε δύο τομείς, ανάλογα με την απόσταση που απέχουν οι κόμβοι υπολογισμού του άκρου από τα κέντρα δεδομένων. Η υπολογιστική στο μακρινό άκρο (far edge), αναφέρεται στην χρήση υπολογιστικών πόρων, όσο το δυνατόν πιο κοντά στο σημείο παραγωγής των δεδομένων (end point). Στις εφαρμογές αυτές, η υπολογιστική υποδομή βρίσκεται στο άκρο του δικτύου, συχνά εντός των εγκαταστάσεων του οργανισμού. Για παράδειγμα, στον τομέα της βιομηχανίας, οι επιχειρήσεις χρησιμοποιούν έναν υπολογιστικό κόμβο για να συλλέγουν, να συγκεντρώνουν και να μεταφράζουν δεδομένα από τα μηχανήματα του εργοστασίου. Κάποια από τα δεδομένα που συλλέγονται, αποστέλλονται στη συνέχεια για περαιτέρω επεξεργασία και ανάλυση, είτε σε κάποιον κόμβο κοντινού άκρου, είτε σε κέντρο δεδομένων. Οι εφαρμογές του μακρινού άκρου, συνήθως, παρέχουν στις επιχειρήσεις χρόνο απόκρισης, μικρότερο από είκοσι χιλιοστά του δευτερολέπτου [48]. Με την χρήση υπολογιστικών πόρων στο μακρινό άκρο, οι οργανισμοί μπορούν να μειώσουν την καθυστέρηση του δικτύου, να βελτιώσουν την ταχύτητα επεξεργασίας των δεδομένων και να καταστήσουν ικανή τόσο τη λήψη αποφάσεων τοπικά, όσο και την αυτοματοποίηση. Αυτή η προσέγγιση είναι ιδιαίτερα επωφελής, όταν απαιτούνται αλληλεπιδράσεις σε πραγματικό χρόνο (real time interactions) ή χαμηλής καθυστέρησης (low latency), επιτρέποντας τη γρήγορη ανταπόκριση σε γεγονότα ή συνθήκες. Εφαρμογές όπως, επαυξημένη ή εικονική πραγματικότητα (AR/VR), βιντεοπαιχνίδια, ζωντανή μετάδοση βίντεο (video live streaming) και εφαρμογές βιομηχανικού διαδικτύου των πραγμάτων (IIoT), είναι παραδείγματα εφαρμογών για τα οποία μπορεί να χρησιμοποιηθεί το μακρινό άκρο. Η υπολογιστική του υποδομή, αναφέρεται με διαφορετικά ονόματα, ανάλογα με τις εφαρμογές που εκτελούνται σε αυτό. Για παράδειγμα, μπορεί να ονομάζεται Εταιρικό Άκρο (Enterprise Edge), όταν εκτελούνται επιχειρηματικές εφαρμογές ή Άκρο Διαδικτύου των Αντικειμένων (IoT Edge), όταν εκτελούνται εφαρμογές Διαδικτύου των Αντικειμένων [49].

Η δεύτερη κατηγορία είναι η υπολογιστική στο κοντινό άκρο (near edge) και αναφέρεται στην υποδομή υπολογιστικών πόρων μεταξύ του μακρινού άκρου και των κέντρων δεδομένων, παρέχοντας ισορροπία μεταξύ της τοπικής επεξεργασίας και των υπηρεσιών νέφους. Σε εγκαταστάσεις κοντινού άκρου, οι υποδομές υπολογισμού και αποθήκευσης, βρίσκονται πιο κοντά στο σημείο παραγωγής δεδομένων, από ό,τι το κέντρο δεδομένων, αλλά

εξακολουθούν να βρίσκονται πιο μακριά από εκείνες του μακρινού άκρου. Όπως το μακρινό άκρο, έτσι και το κοντινό, έχει ως σκοπό την ελαχιστοποίηση της καθυστέρησης, παρέχει ταχύτερη ανάλυση δεδομένων, λήψη αποφάσεων και ανταπόκριση σε γεγονότα που απαιτούν ελάχιστο χρόνο απόκρισης, σε πραγματικό ή σχεδόν πραγματικό χρόνο, αξιοποιώντας ταυτόχρονα τα οφέλη τόσο των τοπικών, όσο και των πόρων του νέφους. Ενώ η υπολογιστική υποδομή του μακρινού άκρου χρησιμοποιείται για συγκεκριμένες εφαρμογές που αφορούν την τοποθεσία στην οποία βρίσκεται, το κοντινό άκρο χρησιμοποιείται για γενικές υπηρεσίες. Για παράδειγμα, ως μνήμη δικτύων παράδοσης περιεχομένου (CDN cache) και υποδομή υπολογιστικής ομίχλης (fog computing infrastructure) [49]. Ένα ακόμα παράδειγμα αποτελεί η επιτήρηση βίντεο, όπου τα δεδομένα αναλύονται σε πραγματικό χρόνο στο άκρο του δικτύου αντί να αποστέλλονται σε μια κεντρική τοποθεσία για επεξεργασία, χαραμίζοντας πολύτιμο χρόνο. Οι εφαρμογές του κοντινού άκρου, παρέχουν συνήθως χρόνο απόκρισης κάτω από εβδομήντα-πέντε χιλιοστά του δευτερολέπτου. Πλεονέκτημα του κοντινού άκρου είναι το γεγονός πως οι εταιρείες έχουν περισσότερο χώρο και ελευθερία να επεκτείνουν την επεξεργαστική ισχύ τους, σε σχέση με το μακρινό άκρο [48]. Ο υπολογισμός στο κοντινό άκρο γεφυρώνει το χάσμα μεταξύ του μακρινού άκρου και των κέντρων δεδομένων, παρέχοντας μια τοπική υπολογιστική υποδομή που επιτρέπει την αποτελεσματική και έγκαιρη επεξεργασία των δεδομένων.

Συνοπτικά, το μακρινό άκρο σχετίζεται κατά κύριο λόγο με το Διαδίκτυο των Αντικειμένων και συσκευές όπως έξυπνους αισθητήρες, ρομπότ, αυτόνομα συστήματα, ενσωματωμένα συστήματα και γενικότερα έξυπνες συσκευές, ενώ το κοντινό άκρο, παρέχει περισσότερη υπολογιστική ισχύ, σχετίζεται με μικρά κέντρα δεδομένων (cloudlets) και απομακρυσμένες εφαρμογές.

2.2.3 Μοντέλα Υπηρεσιών στο Άκρο

Όπως προαναφέραμε, ανάλογα με τις εφαρμογές που υλοποιούνται, το άκρο χαρακτηρίζεται διαφορετικά. Με βάση αυτό, μπορούμε να θεωρήσουμε ότι υπάρχουν τρια βασικά είδη υπολογιστικής/μοντέλα υπηρεσιών στο άκρο, που χρησιμοποιούνται κατά κόρον από τους διάφορους οργανισμούς.

Επιχειρηματικό Άκρο (Enterprise Edge): Το επιχειρηματικό άκρο είναι μια επέκταση του κέντρου δεδομένων της επιχείρησης, αποτελούμενη από κέντρα δεδομένων σε απομακρυσμένες τοποθεσίες, μικρά κέντρα δεδομένων ή διακομιστές. Διαχειρίζεται από το τμήμα πληροφορικής του οργανισμού, παρόμοια με ένα κέντρο δεδομένων, αλλά με πιθανούς περιορισμούς χώρου ή ισχύος που επηρεάζουν το σχεδιασμό του. Παραδείγματα φόρτων εργασίας στο επιχειρηματικό άκρο, περιλαμβάνουν έξυπνες αποθήκες και κέντρα διανομής, όπου η αποδοτικότητα και η αυτοματοποίηση βασίζονται σε ισχυρές τεχνολογίες πληροφοριών, δεδομένων και λειτουργίας. Η εφαρμογή λύσεων τεχνητής νοημοσύνης (AI) για αυτά τα περιβάλλοντα, επιτρέπει ταχύτερους και πιο ευέλικτους ελέγχους προϊόντων και ταχύτερη εκπλήρωση παραγγελιών [50].

Βιομηχανικό Άκρο (Industrial Edge): Το βιομηχανικό άκρο περιλαμβάνει την ανάπτυξη μικρότερων υπολογιστικών μονάδων, όπως διακομιστές ή ενσωματωμένα συστήματα, εκτός των παραδοσιακών περιβαλλόντων κέντρων δεδομένων. Αυτή η υποδομή υπολογιστών άκρου, εξυπηρετεί διάφορες περιπτώσεις βιομηχανικής χρήσης, όπως η ρομποτική, η αυτόνομη ταμειακή μηχανή, οι εφαρμογές έξυπνης πόλης, ο έλεγχος της κυκλοφορίας, και οι έξυπνες συσκευές. Το βιομηχανικό άκρο παρουσιάζει μοναδικές προκλήσεις που σχετίζονται με το χώρο, την ψύξη, την ασφάλεια και τη διαχείριση, καθώς λειτουργεί ανεξάρτητα από τις τυπικές δομές των κέντρων δεδομένων [50]. Το μοντέλο αυτό αφορά τη συλλογή, την επεξεργασία και τη χρήση δεδομένων επί τόπου. Στο πλαίσιο της βιομηχανίας, αυτή η προσέγγιση επιτρέπει την εφαρμογή της τεχνητής νοημοσύνης και της μηχανικής μάθησης (ML), για τη βελτίωση της επιχειρησιακής και επιχειρηματικής αποδοτικότητας. Αυτό επιτυγχάνεται, μέσω της ανάλυσης σε πραγματικό χρόνο των δεδομένων που συλλέγονται από τους αισθητήρες του Βιομηχανικού Διαδικτύου των Πραγμάτων (IIoT) που υπάρχουν στο χώρο του εργοστασίου. Με την αξιοποίηση αυτών των δεδομένων, οι κατασκευαστές μπορούν να εντοπίζουν και να αντιμετωπίζουν τις λειτουργικές προκλήσεις, οδηγώντας σε βελτιωμένη παραγωγικότητα και βελτιστοποίηση των διαδικασιών [51].

Άκρο Παρόχου (Provider Edge): Αυτή η προσέγγιση αναφέρεται σε ένα δίκτυο υπολογιστικών πόρων, προσβάσιμο μέσω του διαδικτύου, που χρησιμοποιείται κυρίως για την παροχή υπηρεσιών από εταιρείες τηλεπικοινωνιών, παρόχους υπηρεσιών, εταιρείες μέσων

ενημέρωσης ή φορείς εκμετάλλευσης δικτύων διανομής περιεχομένου (CDN). Δίνει την δυνατότητα για περιπτώσεις χρήσης, όπως η παράδοση περιεχομένου, τα διαδικτυακά παιχνίδια και η τεχνητή νοημοσύνη, ως υπηρεσία (AIaaS) [50]. Η άκρη του παρόχου υπηρεσιών έχει ως στόχο να προσφέρει αξιόπιστα, χαμηλής καθυστέρησης και υψηλής απόδοσης υπολογιστικά περιβάλλοντα, σε κοντινή απόσταση από τους πελάτες και τις συσκευές. Οι εταιρείες αναβαθμίζουν τα δίκτυά τους για να βελτιώσουν την αποδοτικότητα και να μειώσουν την καθυστέρηση, ιδίως με την παγκόσμια εξάπλωση των δικτύων πέμπτης γενιάς (5G). Αν και οι βελτιώσεις αυτές μπορεί να μην είναι παρατηρήσιμες από τους χρήστες κινητής τηλεφωνίας, επιτρέπουν στους παρόχους υπηρεσιών να αυξήσουν γρήγορα τη χωρητικότητα, ελαχιστοποιώντας παράλληλα το κόστος [51].

2.2.4 Πλεονεκτήματα & Περιπτώσεις χρήσης

Οι συσκευές του Διαδικτύου των Αντικειμένων και η υπολογιστική στο άκρο, μεταμορφώνουν ραγδαία τον τρόπο με τον οποίο οι βιομηχανίες σε όλο τον κόσμο αλληλεπιδρούν με τα δεδομένα. Η τοποθέτηση συσκευών με δυνατότητα επεξεργασίας των δεδομένων, εκεί που δημιουργούνται ή κοντά σε αυτά, η χαμηλή καθυστέρηση, καθώς και η περιορισμένη χρήση του εύρους ζώνης των δικτύων ευρείας περιοχής, είναι από τα βασικότερα πλεονεκτήματα και χαρακτηριστικά της υπολογιστικής στο άκρο. Βάσει αυτών, καθίσταται ιδανική για οργανισμούς, όπως η NASA που την εφαρμόζει για να επεξεργάζεται δεδομένα στο διάστημα, κοντά στην πηγή, αντί να τα μεταδίδει πίσω στη Γη, κάτι που μπορεί να πάρει σημαντικό χρόνο. Για παράδειγμα, οι ειδικοί στον Διεθνή Διαστημικό Σταθμό (ISS), διεξάγουν διάφορες μελέτες και αντί να στέλνουν τα δεδομένα στη Γη για ανάλυση, η οποία θα διαρκούσε εβδομάδες, πραγματοποιούν την εκτέλεση των αναλύσεων στον σταθμό. Η προσέγγιση αυτή, μειώνει σημαντικά τον «χρόνο μέχρι τη γνώση» από μήνες σε λεπτά, επιτρέποντας ταχύτερη επεξεργασία και ανάλυση των δεδομένων στο διάστημα [51]. Άλλη περίπτωση χρήσης, είναι στον τομέα των αυτόνομων συστημάτων. Η ενσωματωμένη αυτονομία για τα αυτόνομα συστήματα, όπως τα ρομπότ εξερεύνησης του διαστήματος στον Άρη και τα αυτόνομα οχήματα στη Γη, περιορίζεται από την επεξεργαστική ισχύ, λόγω της ανάγκης αξιολόγησης πολλαπλών δεδομένων από τους αισθητήρες σε πραγματικό χρόνο. Για παράδειγμα, τα αυτόνομα οχήματα που ταξιδεύουν με εκατό (100) χιλιόμετρα την ώρα, πρέπει να προβλέπουν

πιθανούς κινδύνους αρκετά δευτερόλεπτα νωρίτερα για να διασφαλίσουν την ασφάλεια. Αυτό απαιτεί γρήγορους και πολύπλοκους υπολογισμούς. Η κύρια πρόκληση στο σχεδιασμό υπολογιστικών συστημάτων άκρου αυτόνομων οχημάτων, είναι η αποτελεσματική επεξεργασία μεγάλων ποσοτήτων δεδομένων σε πραγματικό χρόνο, εντός περιορισμένου ενεργειακού προϋπολογισμού, διατηρώντας παράλληλα την ασφάλεια των επιβατών [52]. Είναι σημαντικό να σημειωθεί, ότι στέλνοντας μόνο τα απαραίτητα δεδομένα μέσω του διαδικτύου στα κέντρα δεδομένων παρέχεται και επιπλέον ασφάλεια σε θέματα υποκλοπής της πληροφορίας.

2.3 Ομοιότητες και Διαφορές Μεταξύ Υπολογιστικής Νέφους και Άκρου

Η υπολογιστική στο άκρο δεν αντικαθιστά, ούτε θα αντικαταστήσει τη νεφοϋπολογιστική. Αντίθετα, συνυπάρχουν και αλληλοσυμπληρώνονται. Αυτή η συνεργασία βοηθά στον ψηφιακό μετασχηματισμό σε διάφορους τομείς, όπως η έξυπνη βιομηχανία, η ενέργεια, η ασφάλεια, η προστασία προσωπικών δεδομένων και τα έξυπνα σπίτια. Τα δεδομένα που υποβάλλονται σε επεξεργασία στο άκρο, εξακολουθούν να συγκεντρώνονται στο νέφος για εις βάθος ανάλυση. Η νεφοϋπολογιστική χειρίζεται ανάλυση δεδομένων μεγάλης κλίμακας και επεξεργασία σε μη πραγματικό χρόνο, ενώ ο υπολογισμός στο άκρο υπερέχει στην τοπική, έξυπνη ανάλυση σε πραγματικό χρόνο. Μαζί, καλύπτουν τις διαφορετικές ανάγκες των συσκευών του Διαδικτύου των Αντικειμένων, με τη νεφοϋπολογιστική να παρέχει κεντρική επεξεργασία και το άκρο να προσφέρει τοπικές υπηρεσίες [47].

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ
(Embedded Systems)

Τα ενσωματωμένα συστήματα είναι ολοκληρωμένα συστήματα που περιλαμβάνουν υλικό και λογισμικό σχεδιασμένο για συγκεκριμένες λειτουργίες. Σε αντίθεση με τους γενικής χρήσης υπολογιστές, τα ενσωματωμένα συστήματα είναι αποκλειστικά και μπορούν να λειτουργούν ανεξάρτητα ή ως μέρος ενός μεγαλύτερου συστήματος, εστιάζοντας σε συγκεκριμένες εργασίες. Μπορούν να λειτουργήσουν αυτόνομα ή με ελάχιστη ανθρώπινη παρέμβαση [53]. Ένα τέτοιο σύστημα, περιέχει λογισμικό που έχει σχεδιαστεί για να εκτελεί μια συγκεκριμένη λειτουργία, έναν περιορισμένο αριθμό εργασιών ή μια ομάδα συγκεκριμένων εργασιών, όπως η δειγματοληψία τιμών αισθητήρα ή η επικοινωνία με κάποιον υπολογιστή [54]. Επειδή ένα ενσωματωμένο σύστημα ελέγχει συνήθως τις φυσικές λειτουργίες του μηχανήματος εντός του οποίου είναι ενσωματωμένο, συχνά έχει υπολογιστικούς περιορισμούς στα πλαίσια πραγματικού χρόνου [55].

3.1 Ιστορική Αναδρομή

Η ανάπτυξη των ενσωματωμένων συστημάτων ξεκίνησε στις αρχές της δεκαετίας του 1960 με το πρώτο αναγνωρισμένο ενσωματωμένο σύστημα να είναι ο υπολογιστής καθοδήγησης του Apollo, ο οποίος αναπτύχθηκε από τον Charles Stark Draper στο πανεπιστήμιο MIT. Ο υπολογιστής αυτός, χρησιμοποιούσε μονολιθικά ολοκληρωμένα κυκλώματα και αποτελούσε μια εξέλιξη του πρώτου προγραμματιζόμενου υπολογιστή (Colossus), ο οποίος δημιουργήθηκε το 1944. Η Autonetics (τώρα Boeing), εισήγαγε το ευρέως αναγνωρισμένο ενσωματωμένο σύστημα μαζικής παραγωγής D-17B, για το σύστημα καθοδήγησης πυραύλων Minuteman 1 στα τέλη της δεκαετίας του 1960. Στη δεκαετία του 1970, σημειώθηκε αύξηση στη χρήση των ενσωματωμένων συστημάτων λόγω της πτώσης των τιμών των ολοκληρωμένων κυκλωμάτων, με τις εκδόσεις της Intel, για τον 4-bit επεξεργαστή 4004 το 1971 και τα επόμενα μοντέλα, όπως το 8008 και το 8080. Το πρώτο ενσωματωμένο λειτουργικό σύστημα, VXWorks, κυκλοφόρησε από τη Wind River το 1987, ακολουθούμενο από το Windows Embedded CE της Microsoft το 1996. Τα ενσωματωμένα συστήματα με βάση

το λειτουργικό Linux, εμφανίστηκαν στα τέλη της δεκαετίας του 1990 και παραμένουν διαδεδομένα στις σύγχρονες ενσωματωμένες συσκευές [54].

3.2 Περιπτώσεις Χρήσης

Το φάσμα των ενσωματωμένων συστημάτων ποικίλει από φορητές συσκευές, όπως τα ψηφιακά ρολόγια, έως μεγαλύτερες μηχανές όπως είναι οι οικιακές συσκευές, τα βιομηχανικά ρομπότ και τα οχήματα μεταφοράς. Εξυπηρετούν διάφορους σκοπούς, συμπεριλαμβανομένου του ελέγχου φωτεινών σηματοδοτών, συστημάτων ιατρικής απεικόνισης και βιομηχανικών γραμμών συναρμολόγησης. Τα ενσωματωμένα συστήματα, αποτελούν αναπόσπαστο μέρος των υποσυστημάτων σε αεροσκάφη και διαστημόπλοια και είναι απαραίτητα σε εγκαταστάσεις μεγάλης κλίμακας, όπως εργοστάσια και ηλεκτρικά δίκτυα, συχνά δικτυωμένα μεταξύ τους. Προσαρμόσιμα μέσω λογισμικού, ορισμένα ενσωματωμένα συστήματα, όπως προγραμματιζόμενοι λογικοί ελεγκτές (PLC), λειτουργούν ως ανεξάρτητες μονάδες σε μεγαλύτερα συστήματα. Τα σύγχρονα ενσωματωμένα συστήματα χρησιμοποιούν συνήθως μικροελεγκτές, οι οποίοι είναι μικροεπεξεργαστές με ενσωματωμένη μνήμη και περιφερειακές διεπαφές. Ωστόσο, οι μικροεπεξεργαστές σε συνδυασμό με εξωτερικά τσιπ για τη μνήμη και τις περιφερειακές διεπαφές είναι επίσης διαδεδομένοι, ιδιαίτερα σε πολύπλοκα συστήματα. Οι επεξεργαστές που χρησιμοποιούνται σε ενσωματωμένα συστήματα, μπορεί να διαφέρουν από γενικής χρήσης έως εξειδικευμένους τύπους προσαρμοσμένους για συγκεκριμένους υπολογισμούς ή σχεδιασμένους ειδικά για συγκεκριμένη εφαρμογή. Μια τυπική κατηγορία αποκλειστικών επεξεργαστών που χρησιμοποιούνται συνήθως σε ενσωματωμένα συστήματα, είναι οι επεξεργαστές ψηφιακού σήματος (DSP) [55].

3.3 Χαρακτηριστικά

Τα ενσωματωμένα συστήματα αποτελούνται συνήθως από τρία κύρια στοιχεία: υλικό, λογισμικό και λειτουργικό σύστημα. Το υλικό, το οποίο περιλαμβάνει μικροελεγκτές ή μικροεπεξεργαστές, είναι κρίσιμο και προσαρμοσμένο στις απαιτήσεις της συγκεκριμένης εφαρμογής. Οι μικροελεγκτές χρησιμοποιούνται συνήθως για απλούστερες εργασίες, ενώ οι μικροεπεξεργαστές χρησιμοποιούνται για πιο σύνθετες λειτουργίες. Το λογισμικό, προγραμματισμένο ειδικά για την εφαρμογή, κατευθύνει τις λειτουργίες του μικροελεγκτή ή

του μικροεπεξεργαστή. Επιπλέον, ένα λειτουργικό σύστημα, μπορεί να χρησιμοποιηθεί σε πιο δύσκολα ενσωματωμένα συστήματα, για να διασφαλιστεί ότι όλα τα στοιχεία συνεργάζονται απρόσκοπτα. Τα περισσότερα ενσωματωμένα συστήματα λειτουργούν σε λειτουργικό σύστημα πραγματικού χρόνου (RTOS), για να τηρούν λογικές και χρονικές προθεσμίες, με παραλλαγές, όπως Soft Real-Time OS και Hard Real-Time OS, ανάλογα με την κρισιμότητα της εφαρμογής. Τα ενσωματωμένα συστήματα, προσφέρουν πολλά πλεονεκτήματα, συμπεριλαμβανομένου του χαμηλού κόστους λόγω της προσαρμοσμένης λειτουργικότητάς τους, των συμπαγών μεγεθών που τα καθιστούν κατάλληλα για συσκευές περιορισμένου χώρου, της υψηλής τους αξιοπιστίας για κρίσιμα συστήματα, της ενεργειακής απόδοσης, της δυνατότητας προσαρμογής για την κάλυψη συγκεκριμένων απαιτήσεων εφαρμογής, της δυνατότητας λειτουργίας σε πραγματικό χρόνο και της εύκολης ενσωμάτωσης με άλλα συστήματα. Ωστόσο, παρουσιάζουν επίσης μειονεκτήματα ή προκλήσεις, όπως περιορισμένους πόρους, ευελιξία, δυσκολία στον προγραμματισμό και τον εντοπισμό σφαλμάτων, εξάρτηση από το υλικό και περιορισμένη υποστήριξη και τεκμηρίωση σε σύγκριση με τους υπολογιστές γενικής χρήσης [56].

3.4 Τύποι Ενσωματωμένων Συστημάτων

Τα ενσωματωμένα συστήματα κατηγοριοποιούνται σε τέσσερις τύπους με βάση τις λειτουργικές απαιτήσεις και τις απαιτήσεις απόδοσης:

Αυτόνομα ενσωματωμένα συστήματα: Αυτές οι συσκευές λειτουργούν ανεξάρτητα, χωρίς να χρειάζονται κεντρικά συστήματα, όπως υπολογιστές. Παραδείγματα περιλαμβάνουν συσκευές αναπαραγωγής MP3, ψηφιακές φωτογραφικές μηχανές και φούρνους μικροκυμάτων [56].

Ενσωματωμένα συστήματα πραγματικού χρόνου: Τηρούν αυστηρές προθεσμίες και χωρίζονται σε ευέλικτα και μη ευέλικτα συστήματα πραγματικού χρόνου, όσον αφορά τους χρονικούς περιορισμούς για την ολοκλήρωση των εργασιών. Τα μη ευέλικτα συστήματα πραγματικού χρόνου, χρησιμοποιούνται σε βιομηχανικά μηχανήματα όπου πρέπει να τηρούνται αυστηρές προθεσμίες [56].

Ενσωματωμένα συστήματα δικτύου: Αυτές οι συσκευές είναι συνδεδεμένες σε ένα δίκτυο, όπως τοπικό δίκτυο, δίκτυο ευρείας περιοχής ή διαδίκτυο, επιτρέποντας την πρόσβαση σε διάφορους πόρους. Οι ενσωματωμένοι διακομιστές ιστού, επιτρέπουν τον έλεγχο και την παρακολούθηση μέσω προγραμμάτων περιήγησης [56].

Ενσωματωμένα συστήματα φορητών συσκευών: Οι φορητές συσκευές, όπως τα κινητά τηλέφωνα και οι ψηφιακές φωτογραφικές μηχανές, χρησιμοποιούν ενσωματωμένα συστήματα συγκεκριμένα για τέτοιου είδους συσκευές, αλλά αντιμετωπίζουν περιορισμούς στη μνήμη και στους πόρους [56].

3.5 Αρχιτεκτονικές

Το 1978, η Εθνική Ένωση Κατασκευαστών Ηλεκτρικής Ενέργειας (NEMA), κυκλοφόρησε το ICS 3-1978, ένα πρότυπο για προγραμματιζόμενους μικροελεγκτές που περιλαμβάνει σχεδόν όλους τους ελεγκτές που βασίζονται σε υπολογιστή, όπως υπολογιστές μονής πλακέτας (single-board computers) και ελεγκτές που βασίζονται σε συμβάντα (events). Λόγω αυτού του προτύπου, υπάρχουν διάφοροι κοινόχρηστοι τύποι αρχιτεκτονικής λογισμικού.

- **Απλός βρόχος ελέγχου (Simple control loop):** Το λογισμικό λειτουργεί σε συνεχή βρόχο ελέγχου, όπου το λογισμικό παρακολουθεί συνεχώς τις συσκευές εισόδου και εκτελεί εργασίες διαδοχικά. Ο βρόχος καλεί υπορουτίνες, για τη διαχείριση διαφόρων στοιχείων υλικού ή λογισμικού [55].
- **Σύστημα ελεγχόμενο από διακοπές (Interrupt-controlled system):** Ορισμένα ενσωματωμένα συστήματα ελέγχονται κυρίως από διακοπές, όπου οι εργασίες ενεργοποιούνται από συμβάντα, όπως ένα χρονόμετρο ή δεδομένα από έναν ελεγκτή σειριακής θύρας. Αυτή η αρχιτεκτονική, ταιριάζει σε συστήματα που χρειάζονται απλούς χειριστές συμβάντων χαμηλής καθυστέρησης. Μια απλή εργασία, εκτελείται σε έναν κύριο βρόγχο, χωρίς ευαισθησία στις καθυστερήσεις. Οι χειριστές διακοπής, μπορούν να βάλουν στην ουρά μεγαλύτερες εργασίες για μεταγενέστερη εκτέλεση από τον κύριο βρόγχο, καθιστώντας το σύστημα παρόμοιο με έναν πυρήνα πολλαπλών εργασιών με διακριτές διεργασίες [55].

- **Συνεργατική πολλαπλή εργασία (Cooperative multitasking):** Η συνεργασία πολλαπλών εργασιών, μοιάζει με την λειτουργία του απλού βρόχου ελέγχου, αλλά με τον βρόχο κρυμμένο σε ένα API. Οι προγραμματιστές ορίζουν εργασίες, καθεμία με το δικό της περιβάλλον. Όταν μια εργασία είναι αδρανής, καλεί μια ρουτίνα αδράνειας, για να περάσει τον έλεγχο σε άλλη εργασία. Αυτή η μέθοδος, μοιράζεται τα πλεονεκτήματα και τα μειονεκτήματα του βρόχου ελέγχου, με το πρόσθετο πλεονέκτημα της ευκολότερης προσθήκης λογισμικού με τη σύνταξη νέων εργασιών ή την προσθήκη στην ουρά [55].
- **Πολλαπλών Νημάτων (Multi-threading):** Σε αυτό το σύστημα, ένας κώδικας χαμηλού επιπέδου, εναλλάσσεται μεταξύ εργασιών ή νημάτων με βάση μια διακοπή του χρονοδιακόπτη, υποδηλώνοντας έτσι έναν πυρήνα λειτουργικού συστήματος. Αυτό εισάγει πολυπλοκότητα στη διαχείριση πολλαπλών εργασιών που εκτελούνται παράλληλα. Χωρίς μονάδα διαχείρισης μνήμης, τα προγράμματα πρέπει να σχεδιάζονται προσεκτικά, για να αποτρέπεται η καταστροφή των δεδομένων, χρησιμοποιώντας στρατηγικές συγχρονισμού, όπως ουρές μηνυμάτων ή σημαφόρους. Οι οργανισμοί χρησιμοποιούν συχνά λειτουργικά συστήματα πραγματικού χρόνου, για να απλοποιήσουν την ανάπτυξη εφαρμογών, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στη λειτουργικότητα της συσκευής. Ωστόσο, η επιλογή ενός τέτοιου συστήματος νωρίς στην φάση της ανάπτυξης μπορεί να περιορίσει τη μελλοντική ευελιξία [55].
- **Μικροπυρήνες (Microkernels):** Ένας μικροπυρήνας χειρίζεται την εκχώρηση μνήμης και την εναλλαγή νημάτων της ΚΜΕ, ενώ οι διεργασίες επιπέδου χρήστη διαχειρίζονται σημαντικές λειτουργίες, όπως συστήματα αρχείων και διεπαφές δικτύου [55].
- **Μονολιθικοί πυρήνες (Monolithic kernels):** Ένας μονολιθικός πυρήνας είναι ένας μεγάλος, πολύπλοκος πυρήνας, που προσφέρει δυνατότητες παρόμοιες με τα λειτουργικά συστήματα επιτραπέζιου υπολογιστή, όπως το Linux ή τα Windows, καθιστώντας τον παραγωγικό για ανάπτυξη εφαρμογών. Ωστόσο, απαιτεί περισσότερους πόρους υλικού, είναι συχνά πιο ακριβός και μπορεί να είναι λιγότερο αξιόπιστος, λόγω της πολυπλοκότητάς του. Παραδείγματα ενσωματωμένων μονολιθικών πυρήνων περιλαμβάνουν embedded Linux, VXWorks και Windows CE.

Παρά το υψηλότερο κόστος του υλικού, αυτός ο τύπος ενσωματωμένου συστήματος αυξάνει σε δημοτικότητα, ιδιαίτερα σε ισχυρές συσκευές όπως ασύρματοι δρομολογητές και συστήματα πλοήγησης GPS [55].

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ & ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ
(AI & Machine Learning)

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης που χρησιμοποιεί δεδομένα, μαθηματικά μοντέλα και αλγόριθμους για να μιμηθεί τις ανθρώπινες διαδικασίες μάθησης, βελτιώνοντας σταδιακά την ακρίβειά της. Οι πρόσφατες εξελίξεις στον τομέα της αποθήκευσης και της επεξεργαστικής ισχύος, έχουν διευκολύνει την ανάπτυξη καινοτόμων εφαρμογών μηχανικής μάθησης, όπως η μηχανή συστάσεων (recommendation engine) της πλατφόρμας Netflix και τα αυτόνομα αυτοκίνητα. Η μηχανική μάθηση έχει κρίσιμο ρόλο στην επιστήμη των δεδομένων, χρησιμοποιώντας στατιστικές μεθόδους για την εκπαίδευση αλγορίθμων ταξινόμησης, πρόβλεψης και εύρεσης πληροφοριών σε έργα εξόρυξης δεδομένων [57]. Οι αλγόριθμοι, μαθαίνουν πληροφορίες απευθείας από δεδομένα, χωρίς να βασίζονται σε μια προκαθορισμένη εξίσωση ως μοντέλο και βελτιώνουν προσαρμοστικά την απόδοσή τους, καθώς αυξάνεται ο αριθμός των δειγμάτων που είναι διαθέσιμα για μάθηση [58]. Η μηχανική μάθηση είναι ιδιαίτερα χρήσιμη σε καταστάσεις όπου τα δεδομένα είναι δυναμικά, τα αιτήματα ή οι εργασίες αλλάζουν συνεχώς ή η ανάπτυξη κώδικα για την επίλυση ενός προβλήματος είναι ουσιαστικά αδύνατη. Η προσαρμοστικότητά του και η ικανότητά του να χειρίζεται εξελισσόμενα δεδομένα, το καθιστούν πολύτιμο εργαλείο σε διάφορους τομείς [59]. Καθημερινά παραδείγματα εφαρμογών της μηχανικής μάθησης αποτελούν τα εξής:

- **Αναγνώριση ομιλίας:** Μετατρέπει την ανθρώπινη ομιλία σε γραπτό κείμενο, διευκολύνει τη φωνητική αναζήτηση και τις δυνατότητες προσβασιμότητας σε κινητές συσκευές όπως η Siri [57].
- **Εξυπηρέτηση πελατών:** Τα διαδικτυακά chatbot αντικαθιστούν ανθρώπινους πράκτορες, παρέχοντας βοήθεια, απαντώντας σε συχνές ερωτήσεις και προσφέρουν εξατομικευμένες συμβουλές σε ιστότοπους ηλεκτρονικού εμπορίου και πλατφόρμες ανταλλαγής μηνυμάτων [57].
- **Όραση υπολογιστών:** Η τεχνητή νοημοσύνη επιτρέπει στους υπολογιστές να αντλούν σημαντικές πληροφορίες από ψηφιακές εικόνες, βίντεο και άλλες οπτικές εισόδους και στη συνέχεια να προβαίνουν στις κατάλληλες ενέργειες. Υποστηριζόμενη από συνελικτικά νευρωνικά δίκτυα, η όραση υπολογιστών έχει εφαρμογές που κυμαίνονται

από την προσθήκη ετικετών σε φωτογραφίες μέχρι την ακτινολογική απεικόνιση και τα αυτόνομα οχήματα [57].

- **Μηχανές συστάσεων:** Οι αλγόριθμοι τεχνητής νοημοσύνης αναλύουν τις προτιμήσεις των πελατών σε προϊόντα, βάσει των προηγούμενων επιλογών τους, έτσι ώστε να δημιουργήσουν εξατομικευμένες προτάσεις προϊόντων [57].
- **Ρομποτική αυτοματοποίηση διεργασιών (RPA):** Οι ευφυείς τεχνολογίες αυτοματισμού, αυτοματοποιούν επαναλαμβανόμενες χειροκίνητες εργασίες, βελτιώνοντας την αποδοτικότητα και την παραγωγικότητα [57].
- **Ανίχνευση απάτης:** Οι τράπεζες και άλλα χρηματοπιστωτικά ιδρύματα, μπορούν να χρησιμοποιήσουν τη μηχανική μάθηση για να εντοπίσουν ύποπτες συναλλαγές. Η εποπτευόμενη μάθηση, μπορεί να εκπαιδεύσει ένα μοντέλο χρησιμοποιώντας πληροφορίες γνωστών μοτίβων απάτης σε συναλλαγές. Ο εντοπισμός ανωμαλιών μπορεί να εντοπίσει συναλλαγές που φαίνονται άτυπες και αξίζουν περαιτέρω διερεύνηση [57].

Εφαρμογές όπως το TensorFlow και το PyTorch χρησιμοποιούνται συνήθως, για την επιτάχυνση της ανάπτυξης αλγορίθμων μηχανικής μάθησης.

4.1 Η Εξέλιξη

Η μηχανική μάθηση, αν και κερδίζει δημοτικότητα με τις προόδους στους υπολογιστές, έχει βαθιές ρίζες που χρονολογούνται από το 1950. Το τεστ Turing του Alan Turing πρότεινε μια μέθοδο, για να προσδιοριστεί εάν η τεχνητή νοημοσύνη κατανοεί τη γλώσσα. Η πρόοδος συνεχίστηκε με το πρόγραμμα πούλι (checkers program) του Arthur Samuel το 1952, το νευρωνικό δίκτυο του Frank Rosenblatt το 1957 και τη μάθηση βασισμένη στην επεξήγηση του Gerald DeJong το 1981. Η δεκαετία του 1990 σηματοδότησε μια στροφή προς προσεγγίσεις που βασίζονται σε δεδομένα (εποπτευόμενη μάθηση), οδηγώντας σε προγράμματα που αναλύουν μεγάλα σύνολα δεδομένων. Η μη εποπτευόμενη μάθηση, εμφανίστηκε τη δεκαετία του 2000, ανοίγοντας το δρόμο για τη βαθιά μάθηση. Στα βασικά ορόσημα περιλαμβάνονται η Deep Blue, που νίκησε τον γκραντ μάστερ του σκακιού Garry Kasparov το 1997 και η νίκη της AlphaGo επί του Lee Sedol το 2016. Σήμερα, οι εφαρμογές μηχανικής μάθησης και τεχνητής νοημοσύνης είναι πιο προσιτές, μεταβαίνοντας σε

συστήματα που βασίζονται στη νεφοϋπολογιστική. Μεγάλες εταιρείες τεχνολογίας, όπως η Google, η Amazon, η Microsoft, η Baidu και η IBM, προσφέρουν πλατφόρμες μηχανικής μάθησης. Οι αλγόριθμοι μηχανικής μάθησης, συνεχίζουν να διαμορφώνουν την έρευνα και τη βιομηχανία, πιέζοντας τα όρια του εφικτού [60].

4.2 Μηχανική Μάθηση & Βαθιά Μάθηση & Νευρωνικά Δίκτυα

Η βαθιά και η μηχανική μάθηση, είναι επιμέρους πεδία της τεχνητής νοημοσύνης με ορισμένες διαφοροποιήσεις. Ενώ τα νευρωνικά δίκτυα αποτελούν υποσύνολο της μηχανικής μάθησης, η βαθιά μάθηση είναι υποσύνολο των νευρωνικών δικτύων. Ο τρόπος με τον οποίο η βαθιά μάθηση και η μηχανική μάθηση διαφέρουν, είναι στο πώς μαθαίνει κάθε αλγόριθμος. Οι αλγόριθμοι βαθιάς μάθησης, μπορούν να μαθαίνουν από επισημασμένα σύνολα δεδομένων (labeled datasets), αλλά μπορούν επίσης να επεξεργάζονται μη δομημένα δεδομένα στην ακατέργαστη μορφή τους (π.χ κείμενο ή εικόνες), εντοπίζοντας αυτόματα διακριτικά χαρακτηριστικά διαφόρων κατηγοριών δεδομένων, χωρίς ιδιαίτερη ανθρώπινη παρέμβαση. Αντίθετα, η κλασική μηχανική μάθηση, βασίζεται περισσότερο στον ανθρώπινο παράγοντα για τον καθορισμό των χαρακτηριστικών και απαιτεί πιο δομημένα δεδομένα. Τα νευρωνικά δίκτυα, ή τεχνητά νευρωνικά δίκτυα (ANNs), αποτελούνται από στρώματα (layers) κόμβων οι οποίοι συνδέονται μεταξύ τους με βάρη και κατώτατα όρια. Τα στρώματα αυτά περιέχουν ένα επίπεδο εισόδου (input layer), ένα ή περισσότερα κρυφά επίπεδα (hidden layers) και ένα επίπεδο εξόδου (output layer). Η βαθιά μάθηση αναφέρεται σε νευρωνικά δίκτυα με περισσότερα από τρία στρώματα. Αυτή, όπως και τα νευρωνικά δίκτυα, έχουν συμβάλλει σημαντικά στην πρόοδο της όρασης υπολογιστών, της επεξεργασίας φυσικής γλώσσας και της αναγνώρισης ομιλίας [57].

Κατά την επιλογή μεταξύ της μηχανικής μάθησης και της βαθιάς μάθησης, για την υλοποίηση μιας εφαρμογής, πρέπει να λαμβάνονται υπόψη διάφοροι παράγοντες. Εάν δεν υπάρχει διαθέσιμη μονάδα επεξεργασίας γραφικών (GPU) υψηλής απόδοσης και επαρκή επισημασμένα δεδομένα, η μηχανική μάθηση πιθανότατα είναι η πιο κατάλληλη επιλογή. Η βαθιά μάθηση τείνει να είναι πιο πολύπλοκη και απαιτεί μεγαλύτερο σύνολο δεδομένων, συνήθως αρκετές χιλιάδες εικόνες, για να επιτύχει αξιόπιστα αποτελέσματα. Η επιλογή της μηχανικής μάθησης, παρέχει την ευελιξία εκπαίδευσης μοντέλων χρησιμοποιώντας διάφορους

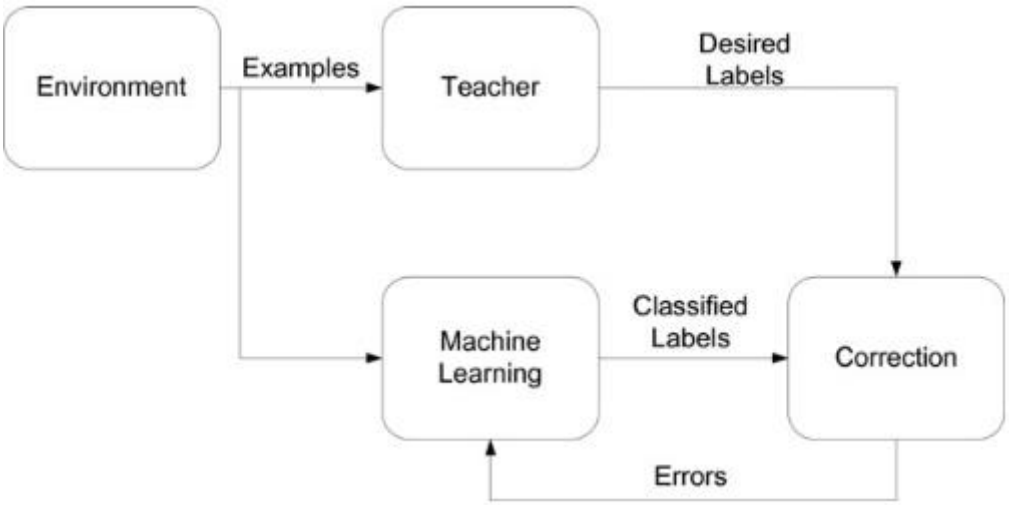
ταξινομητές (classifiers) και την εξαγωγή χαρακτηριστικών που αποδίδουν βέλτιστα αποτελέσματα. Επιπλέον, δίνει την δυνατότητα για πειραματισμούς με διαφορετικούς συνδυασμούς προσεγγίσεων, ταξινομητών και χαρακτηριστικών προκειμένου να καθοριστεί η πιο αποτελεσματική διάταξη για συγκεκριμένα δεδομένα [58]. Ωστόσο η βαθιά μάθηση έχει επιδείξει ανώτερη απόδοση σε σχέση με τις παραδοσιακές τεχνικές μηχανικής μάθησης στον τομέα της αναγνώρισης εικόνων. Τα μοντέλα βαθιάς μάθησης, ιδιαίτερα τα Συνελικτικά Νευρωνικά Δίκτυα, μπορούν να μάθουν αυτόματα ιεραρχικά χαρακτηριστικά απευθείας από ακατέργαστα δεδομένα εικόνας. Αυτή η ικανότητα επιτρέπει στα μοντέλα να καταγράφουν πιο πολύπλοκα μοτίβα και αποχρώσεις, με αποτέλεσμα σημαντικά υψηλότερη ακρίβεια στις εργασίες αναγνώρισης εικόνας (LeCun, Bengio, & Hinton, 2015) [61].

4.3 Μοντέλα Μηχανικής Μάθησης

Η μηχανική μάθηση μπορεί να κατηγοριοποιηθεί σε τρια κύρια μοντέλα αλγορίθμων, με βάση τη φύση της μάθησης και τον τύπο των δεδομένων που χρησιμοποιούνται.

Εποπτευόμενη Μάθηση (Supervised Learning): Η εποπτευόμενη μάθηση ορίζεται από τη χρήση επισημασμένων συνόλων δεδομένων για την εκπαίδευση αλγορίθμων για την ταξινόμηση δεδομένων ή την ακριβή πρόβλεψη των αποτελεσμάτων. Η εκμάθηση στο εποπτευόμενο μοντέλο, συνεπάγεται τη δημιουργία μιας συνάρτησης που μπορεί να εκπαιδευτεί, χρησιμοποιώντας ένα σύνολο επισημασμένων δεδομένων ως δεδομένα εκπαίδευσης και στη συνέχεια να εφαρμοστεί σε άγνωστα δεδομένα για την επίτευξη κάποιας προγνωστικής απόδοσης. Ο στόχος είναι να δημιουργηθεί η συνάρτηση έτσι, ώστε να γενικεύεται καλά σε δεδομένα που δεν έχει συναντήσει ποτέ [62]. Επισημασμένα δεδομένα σημαίνει ότι κάθε παράδειγμα στο σύνολο δεδομένων εκπαίδευσης, επισημαίνεται με την απάντηση που θα πρέπει να βρει μόνος του ο αλγόριθμος. Έτσι, ένα επισημασμένο σύνολο δεδομένων με εικόνες λουλουδιών, θα έλεγε στο μοντέλο ποιες φωτογραφίες είναι από τριαντάφυλλα, μαργαρίτες και νάρκισσους. Όταν εμφανίζεται μια νέα εικόνα, το μοντέλο τη συγκρίνει με τα παραδείγματα εκπαίδευσης, για να προβλέψει τη σωστή ετικέτα [63]. Καθώς τα δεδομένα εισόδου τροφοδοτούνται στο μοντέλο, αυτό προσαρμόζει τα βάρη του κατάλληλα, έως ότου η προγνωστική του απόδοση να είναι ικανοποιητική. Ορισμένες μέθοδοι που χρησιμοποιούνται στην εποπτευόμενη μάθηση, περιλαμβάνουν νευρωνικά δίκτυα, naïve

bayes, γραμμική παλινδρόμηση, λογιστική παλινδρόμηση, τυχαίο δάσος και μηχανή διανυσμάτων υποστήριξης (SVM) [62].

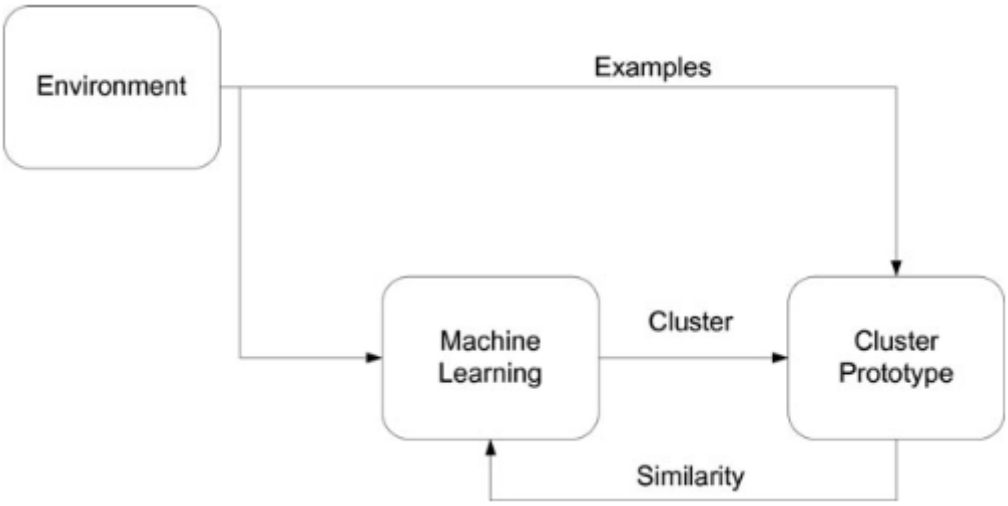


Εικόνα 14 - Εποπτευόμενη Μάθηση
https://doi. org/10.1007/978-3-030-65900-4

Υπάρχουν δύο βασικοί τομείς όπου η εποπτευόμενη μάθηση είναι χρήσιμη, τα προβλήματα ταξινόμησης (classification) και τα προβλήματα παλινδρόμησης (regression). Οι εργασίες ταξινόμησης, περιλαμβάνουν την πρόβλεψη διακριτών τιμών και την εκχώρηση δεδομένων εισόδου σε συγκεκριμένες κλάσεις ή ομάδες. Για παράδειγμα, σε ένα σύνολο δεδομένων από εικόνες ζώων, ο αλγόριθμος στοχεύει να προσδιορίσει σωστά, εάν κάθε φωτογραφία απεικονίζει μια γάτα, ένα κοάλα ή μια χελώνα. Η αξιολόγηση βασίζεται στην ακρίβεια του αλγόριθμου για την ταξινόμηση νέων εικόνων. Από την άλλη πλευρά, η παλινδρόμηση ασχολείται με συνεχή δεδομένα, όπως η πρόβλεψη της τιμής ενός διαμερίσματος με βάση διάφορους παράγοντες, όπως τα τετραγωνικά μέτρα και η τοποθεσία. Η εποπτευόμενη μάθηση, επομένως, εφαρμόζεται σε προβλήματα όπου υπάρχει ένα σύνολο διαθέσιμων σημείων αναφοράς, με τα οποία μπορεί να εκπαιδευτεί ο αλγόριθμος. Αυτά όμως, δεν είναι πάντα διαθέσιμα, άρα μπορεί να μην είναι πάντα εφαρμόσιμη [63].

Μη Εποπτευόμενη Μάθηση (Unsupervised Learning): Η μη εποπτευόμενη μάθηση, χρησιμοποιεί αλγόριθμους μηχανικής μάθησης για την ανάλυση και τη ομαδοποίηση συνόλων δεδομένων χωρίς ετικέτες. Αυτοί οι αλγόριθμοι ανακαλύπτουν κρυφά μοτίβα ή ομαδοποιήσεις δεδομένων, χωρίς την ανάγκη ανθρώπινης παρέμβασης [57]. Για τον λόγο αυτό, δεν έχει τρόπο μέτρησης της απόδοσης της. Ο στόχος είναι να δημιουργηθεί μια συνάρτηση αντιστοίχισης

(mapping function) που κατηγοριοποιεί τα δεδομένα σε κλάσεις, με βάση τα χαρακτηριστικά που κρύβονται μέσα σε αυτά. Αρχικά η συνάρτηση τμηματοποιεί ένα σύνολο δεδομένων σε κλάσεις, χωρίς να εφαρμόζει σαφείς ετικέτες σε αυτές. Αυτή η τμηματοποίηση, μπορεί να χρησιμεύσει ως αποτέλεσμα και η εφαρμογή της ποικίλλει ανάλογα με το περιβάλλον. Για παράδειγμα, σε ένα σύστημα συστάσεων, τα χαρακτηριστικά των χρηστών ή οι αγορές τους, μπορεί να αντιπροσωπεύονται ως διανύσματα εισόδου, ενώ οι χρήστες που ανήκουν σε μια κατηγορία, μοιράζονται παρόμοια ενδιαφέροντα, διευκολύνοντας το στοχευμένο μάρκετινγκ



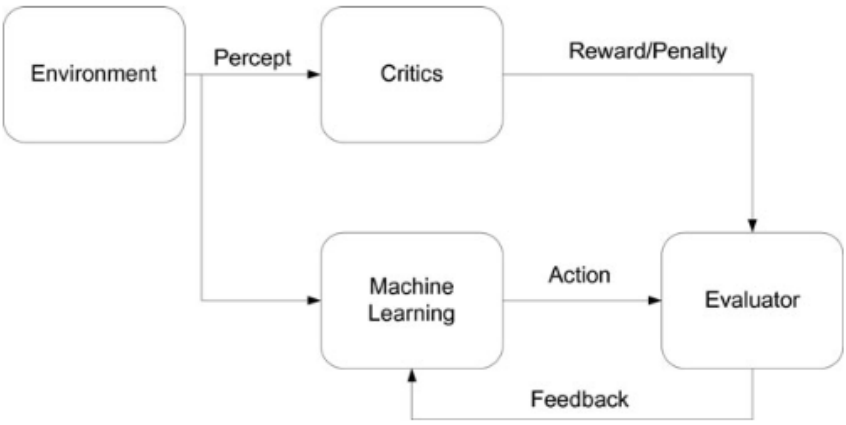
Εικόνα 15 - Μη Εποπτευόμενη Μάθηση
https://doi. org/10.1007/978-3-030-65900-4

ή τις προτάσεις προϊόντων [62].

Η μάθηση χωρίς επίβλεψη είναι ιδανική για διερευνητική ανάλυση δεδομένων, τμηματοποίηση πελατών και αναγνώριση εικόνας και μοτίβου. Αποδεικνύεται χρήσιμη για την ανακάλυψη ομοιοτήτων και διαφορών στις πληροφορίες. Επιπλέον, η μέθοδος αυτή χρησιμοποιείται για τη μείωση διαστάσεων σε μοντέλα με κοινές προσεγγίσεις, για αυτό περιλαμβάνουν την Ανάλυση Κύριων Συνιστωσών (PCA) και την Αποσύνθεση Ενιαίας Τιμής (SVD). Διάφοροι αλγόριθμοι χρησιμοποιούνται στα πλαίσια της μη εποπτευόμενης μάθησης, συμπεριλαμβανομένων των νευρωνικών δικτύων, της ομαδοποίησης k-means και των μεθόδων πιθανολογικής ομαδοποίησης [57].

Ενισχυτική Μάθηση (Reinforcement Learning): Η ενισχυτική μάθηση είναι ένα μοντέλο μηχανικής μάθησης παρόμοιο με την εποπτευόμενη μάθηση, με την διαφορά ότι ο

αλγόριθμος δεν εκπαιδεύεται με τη χρήση δειγμάτων δεδομένων. Αυτό το μοντέλο μαθαίνει με την πάροδο του χρόνου, χρησιμοποιώντας την λογική της δοκιμής και του σφάλματος (trial and error) [57]. Είναι ένα δυναμικό μοντέλο, ικανό να μαθαίνει ακολουθίες εισόδων προς εξόδους, βασιζόμενη σε εξαρτήσεις. Λειτουργεί μέσα σε ένα περιβάλλον που ορίζεται από καταστάσεις και πιθανές ενέργειες. Κατά τη διάρκεια της μαθησιακής διαδικασίας, ο αλγόριθμος εξερευνά ζεύγη κατάστασης-δράσεων για να δημιουργήσει έναν πίνακα και στη συνέχεια εκμεταλλευόμενος την πληροφορία που έχει αποκομίσει, επιλέγει την καλύτερη ενέργεια για μια δεδομένη κατάσταση [62]. Οι πράκτορες τεχνητής νοημοσύνης προσπαθούν να επιτύχουν συγκεκριμένους στόχους ή να βελτιώσουν την απόδοση των εργασιών, λαμβάνοντας ανταμοιβές για ενέργειες που οδηγούν στον στόχο. Ο στόχος είναι η πρόβλεψη των βέλτιστων ενεργειών που αποφέρουν την υψηλότερη ανταμοιβή, εξισορροπώντας την εκμετάλλευση των προηγούμενων μαθημάτων και την εξερεύνηση νέων στρατηγικών. Οι στρατηγικές που ακολουθούν οι πράκτορες, βελτιώνονται μέσω επαναληπτικών βρόχων ανατροφοδότησης. Αυτή η τεχνική, βρίσκει πρακτική εφαρμογή στην εκπαίδευση ρομπότ, για εργασίες που απαιτούν διαδοχική λήψη αποφάσεων, όπως η πλοήγηση σε αυτόνομα οχήματα



Εικόνα 16 - Ενισχυτική Μάθηση
https://doi. org/10.1007/978-3-030-65900-4

ή η διαχείριση του αποθέματος μιας αποθήκης. Κάθε αλγόριθμος μαθαίνει μοναδικά και η επιλογή της καταλληλότερης προσέγγισης είναι ζωτικής σημασίας, έτσι ώστε να υπάρχουν αποτελεσματικά μαθησιακά αποτελέσματα [63].

Ένα παράδειγμα σεναρίου, περιλαμβάνει την εκπαίδευση ενός πράκτορα για να παίξει blackjack, όπου οι καταστάσεις αντιπροσωπεύουν ποσά καρτών για τον παίκτη και οι ενέργειες

περιλαμβάνουν «το χτύπημα» ή «τη στάση». Οι ανταμοιβές απονέμονται με βάση τη νίκη ή την απώλεια χεριών, ενημερώνοντας τις αποφάσεις του πράκτορα. Σε αντίθεση με την εποπτευόμενη μάθηση, η ενισχυτική μάθηση βασίζεται σε διαλείπουσα ανατροφοδότηση, η οποία συνήθως λαμβάνεται μόνο όταν επιτυγχάνεται η επιθυμητή κατάσταση, όπως η απόκτηση ενός χεριού με ένα σύνολο εικοσιένα (21). Μεταξύ άλλων αλγόριθμων με διαφορετικά χαρακτηριστικά, η ενισχυτική μάθηση περιλαμβάνει τους Q-learning και State-action-reward-state-action. Η ενισχυτική μάθηση είναι ένα ιδανικό μοντέλο που μπορεί να μάθει πώς να λαμβάνει αποφάσεις σε ένα αβέβαιο περιβάλλον [62].

4.4 Είδη Νευρωνικών Δικτύων

Τα νευρωνικά δίκτυα διατίθενται σε διάφορους τύπους, καθένας κατάλληλος για διαφορετικές εργασίες. Οι πιο συνηθισμένοι τύποι περιλαμβάνουν:

- **Νευρωνικά Δίκτυα τροφοδοσίας (FNN):** Ο απλούστερος τύπος, όπου τα δεδομένα ρέουν προς μία κατεύθυνση, από την είσοδο στην έξοδο.
- **Επαναλαμβανόμενα νευρωνικά δίκτυα (RNN):** Αυτά τα δίκτυα είναι ιδανικά για δεδομένα ακολουθίας, έχουν συνδέσεις που σχηματίζουν κύκλους, επιτρέποντας στις πληροφορίες να παραμένουν.
- **Συνελικτικά νευρωνικά δίκτυα (CNN):** Είναι ειδικά σχεδιασμένα για δεδομένα εικόνας, εφαρμόζουν λειτουργίες συνέλιξης για την καταγραφή χωρικών ιεραρχιών.
- **Δίκτυα Generative Adversarial Networks (GAN):** Χρησιμοποιούνται για τη δημιουργία δεδομένων, αποτελούνται από δύο δίκτυα, ένα δημιουργό και ένα διαχωριστικό, που ανταγωνίζονται το ένα το άλλο.
- **Αυτοκωδικοποιητές:** Χρησιμοποιούνται κυρίως για μη εποπτευόμενη μάθηση και συμπίεση δεδομένων, στοχεύουν στην ανακατασκευή των δεδομένων εισόδου με αποτελεσματικό τρόπο.

Κάθε τύπος έχει τα δυνατά του σημεία, αλλά τα συνελικτικά νευρωνικά δίκτυα έχουν γίνει το πρότυπο για την αναγνώριση εικόνας λόγω της αρχιτεκτονικής τους, η οποία είναι ειδικά προσαρμοσμένη για το χειρισμό των χωρικών σχέσεων στις εικόνες [64].

4.5 Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα προτιμώνται για την αναγνώριση εικόνων λόγω της ικανότητάς τους να επεξεργάζονται δεδομένα με τοπολογία τύπου πλέγματος, όπως εικόνες. Χρησιμοποιούν συνελικτικά επίπεδα με φίλτρα που είναι ικανά να ανιχνεύουν μοτίβα, όπως ακμές, υφές και σχήματα, απευθείας από τα ακατέργαστα δεδομένα. Αυτά τα μοτίβα μαθαίνονται αυτόματα μέσω της διαδικασίας εκπαίδευσης, γεγονός που καθιστά αυτόν τον τύπο νευρωνικών δικτύων εξαιρετικά αποτελεσματικό για πολύπλοκες εργασίες αναγνώρισης εικόνων. Επιπλέον, τα τοπικά δεκτικά πεδία και τα κοινά βάρη στα συνελικτικά δίκτυα μειώνουν τον αριθμό των παραμέτρων, καθιστώντας τα υπολογιστικά αποδοτικά και λιγότερο επιρρεπή στην υπερπροσαρμογή σε σύγκριση με τα πλήρως συνδεδεμένα δίκτυα [65].

Τέτοιου είδους δίκτυα αποτελούνται από πολλά επίπεδα, συμπεριλαμβανομένων των συνελικτικών επιπέδων, των επιπέδων συγκέντρωσης (pooling layers) και των πλήρως συνδεδεμένων επιπέδων. Το κλειδί για την εκπαίδευσή τους είναι ο αλγόριθμος backpropagation, ο οποίος ενημερώνει τα βάρη του δικτύου για να ελαχιστοποιήσει τη διαφορά μεταξύ της προβλεπόμενης εξόδου και της πραγματικής εξόδου. Κατά την εκτέλεση του αλγορίθμου, η κλίση της συνάρτησης απώλειας σε σχέση με κάθε βάρος υπολογίζεται χρησιμοποιώντας τον κανόνα της αλυσίδας. Αυτές οι διαβαθμίσεις χρησιμοποιούνται στη συνέχεια για την προσαρμογή των βαρών, μειώνοντας το σφάλμα σε μελλοντικές προβλέψεις. Αυτή η διαδικασία συνεχίζεται επαναληπτικά, βελτιώνοντας σταδιακά την ακρίβεια του δικτύου [66].

4.6 Βελτιστοποιητής Adam

Ο gradient descent είναι ένας αλγόριθμος βελτιστοποίησης που χρησιμοποιείται για την ελαχιστοποίηση της συνάρτησης απώλειας (loss function) στα νευρωνικά δίκτυα. Στο πλαίσιο των συνελικτικών νευρωνικών δικτύων, λειτουργεί με επαναληπτική μετακίνηση των βαρών προς την αντίθετη κατεύθυνση από την κλίση της συνάρτησης απώλειας σε σχέση με τα βάρη. Ο ρυθμός εκμάθησης ελέγχει το μέγεθος αυτών των βημάτων. Κάνοντας μικρά βήματα προς τη σωστή κατεύθυνση, ο gradient descent βοηθά το δίκτυο να συγκλίνει στην

ελάχιστη απώλεια, με αποτέλεσμα καλύτερη απόδοση στις εργασίες αναγνώρισης εικόνας [67].

Ο βελτιστοποιητής Adam (Adaptive Moment Estimation) είναι μια επέκταση του gradient descent που υπολογίζει προσαρμοστικούς ρυθμούς εκμάθησης για κάθε παράμετρο. Συνδυάζει τα οφέλη δύο άλλων δημοφιλών μεθόδων: AdaGrad και RMSProp. Ο Adam παρακολουθεί έναν εκθετικά φθίνοντα μέσο όρο περασμένων κλίσεων (πρώτο βήμα) και περασμένων τετραγωνικών κλίσεων (δεύτερο βήμα), προσαρμόζοντας ανάλογα τον ρυθμό εκμάθησης. Αυτό καθιστά τον Adam ιδιαίτερα κατάλληλο για προβλήματα με μεγάλα σύνολα δεδομένων και χώρους παραμέτρων πολλών διαστάσεων, όπως αυτά που συναντώνται στη βαθιά μάθηση, παρέχοντας ταχύτερη σύγκλιση και καλύτερη γενίκευση [68].

4.7 Υποπροσαρμογή & Υπερπροσαρμογή (Under & Over fitting)

Η υποπροσαρμογή (underfitting) συμβαίνει όταν ένα μοντέλο είναι πολύ απλό για να καταγράψει τα υποκείμενα μοτίβα στα δεδομένα, με αποτέλεσμα κακή απόδοση τόσο στα σύνολα εκπαίδευσης όσο και στα σύνολα επικύρωσης. Η υπερπροσαρμογή (overfitting), από την άλλη πλευρά, συμβαίνει όταν ένα μοντέλο γίνεται πολύ περίπλοκο, καταγράφοντας θόρυβο και άσχετες λεπτομέρειες στα δεδομένα εκπαίδευσης, γεγονός που οδηγεί σε κακή γενίκευση σε νέα, αόρατα δεδομένα. Στα συνελικτικά νευρωνικά δίκτυα, η υπερπροσαρμογή μπορεί να μετριαστεί χρησιμοποιώντας τεχνικές όπως η απόρριψη (dropout), η επαύξηση δεδομένων (data augmentation) και η τακτοποίηση (regularization). Ο σωστός συντονισμός των υπερπαραμέτρων, μαζί με επαρκή δεδομένα εκπαίδευσης, είναι απαραίτητος για την επίτευξη μιας ισορροπίας όπου το μοντέλο δεν υποπροσαρμόζεται ούτε υπερπροσαρμόζεται [69].

4.8 Ενσωματωμένα Συστήματα & Μηχανική Μάθηση

Η ενσωμάτωση των δυνατοτήτων της μηχανικής μάθησης σε ενσωματωμένα συστήματα προσφέρει πολλά πλεονεκτήματα, συμπεριλαμβανομένων των βελτιωμένων αναλύσεων, των δυνατοτήτων πρόβλεψης και λήψης αποφάσεων απευθείας στο άκρο, εξαλείφοντας την ανάγκη για μετάδοση δεδομένων σε απομακρυσμένους διακομιστές για

επεξεργασία. Αυτό, όχι μόνο μειώνει την καθυστέρηση και επιτρέπει την ανάλυση σε πραγματικό χρόνο, αλλά ενισχύει επίσης το απόρρητο και την ασφάλεια των ευαίσθητων δεδομένων. Αξιοποιώντας αλγόριθμους τεχνητής νοημοσύνης και βαθιάς μάθησης, τα ενσωματωμένα συστήματα μπορούν να μαθαίνουν και να εξελίσσουν τη συμπεριφορά τους με την πάροδο του χρόνου, με βάση τις μεταβαλλόμενες περιβαλλοντικές συνθήκες και τις αλληλεπιδράσεις των χρηστών. Επίσης, μπορούν να ταυτοποιήσουν και να αναγνωρίσουν ανθρώπινα πρόσωπα, αντικείμενα και άλλα στοιχεία, οδηγώντας σε βελτιωμένη ανάλυση και μέτρηση δεδομένων. Αυτή η τεχνολογία, επιτρέπει επίσης την αυτοματοποίηση διαφόρων διαδικασιών, βελτιστοποιώντας τις επιχειρηματικές λειτουργίες και βελτιώνοντας τη συνολική απόδοση. Επιπλέον, η χρήση της μηχανικής μάθησης σε ενσωματωμένα συστήματα, διευκολύνει την ανάπτυξη προηγμένων εφαρμογών για κοινωνικούς και επιχειρηματικούς σκοπούς, όπως η προγνωστική ανάλυση, η επιλογή υποψηφίων ανθρωπίνων πόρων και η διαχείριση μιας πόλης. Συνολικά, η ενσωμάτωση της μηχανικής μάθησης σε ενσωματωμένα συστήματα, συμβάλλει στη δημιουργία πολύτιμων γνώσεων και στην πρόοδο της τεχνολογίας σε διάφορους τομείς [70].

4.9 Tensorflow & Tensorflow Lite

Το **TensorFlow** είναι μια ανοιχτού κώδικα βιβλιοθήκη μηχανικής μάθησης, που αναπτύχθηκε από την ομάδα Google Brain. Έχει σχεδιαστεί για να διευκολύνει τους προγραμματιστές να δημιουργούν και να αναπτύσσουν μοντέλα μηχανικής μάθησης, προσφέροντας ένα ολοκληρωμένο οικοσύστημα, που υποστηρίζει μια ποικιλία εργασιών μηχανικής μάθησης, συμπεριλαμβανομένων των βαθιών νευρωνικών δικτύων. Το TensorFlow, παρέχει ένα εκτεταμένο σύνολο εργαλείων, βιβλιοθηκών και πόρων που επιτρέπουν στους προγραμματιστές να δημιουργούν εξελιγμένα μοντέλα μηχανικής μάθησης για διάφορες εφαρμογές, από αναγνώριση εικόνας και ομιλίας, μέχρι επεξεργασία φυσικής γλώσσας και αναλυτική πρόβλεψη. Χρησιμοποιούμε το TensorFlow γιατί προσφέρει ευελιξία, κλιμακωσιμότητα και υποστηρίζεται από μια ισχυρή κοινότητα. Η αρχιτεκτονική του TensorFlow, επιτρέπει την ανάπτυξη σε μια σειρά από πλατφόρμες, συμπεριλαμβανομένων επιτραπέζιων υπολογιστών, υπολογιστικών συστάδων (clusters), κινητών συσκευών και διαδικτύου (web). Υποστηρίζει APIs υψηλού επιπέδου, για ευκολότερη δημιουργία μοντέλων

και λειτουργίες χαμηλού επιπέδου, για λεπτομερή βελτιστοποίηση. Η εκτεταμένη τεκμηρίωση, τα μαθήματα και η υποστήριξη της κοινότητας, καθιστούν το TensorFlow προσβάσιμο, τόσο για αρχάριους, όσο και για προχωρημένους χρήστες στον τομέα της μηχανικής μάθησης [71][72].

Το **TensorFlow Lite** είναι μια ελαφριά έκδοση του TensorFlow, που σχεδιάστηκε ειδικά για κινητές και ενσωματωμένες συσκευές. Επιτρέπει την εκτέλεση μηχανικής μάθησης στη συσκευή με χαμηλή καθυστέρηση και μικρό μέγεθος εκτελέσιμου αρχείου, καθιστώντας το ιδανικό για περιβάλλοντα με περιορισμένους πόρους. Παρέχει επίσης, βελτιστοποιημένα προκαθορισμένα μοντέλα, ένα σύνολο εργαλείων για τη μετατροπή μοντέλων (από Tensorflow σε Tensorflow Lite) και έναν διερμηνέα για την εκτέλεση μοντέλων σε συσκευές όπως smartphones, μικροελεγκτές και Raspberry Pi [72][73].

Οι κύριες διαφορές μεταξύ του TensorFlow και του TensorFlow Lite, βρίσκονται στο σχεδιασμό και τη βελτιστοποίηση για συγκεκριμένες χρήσεις. Το TensorFlow είναι σχεδιασμένο για την ανάπτυξη και την εκπαίδευση μοντέλων που απευθύνεται σε ισχυρές πλατφόρμες με άφθονους υπολογιστικούς πόρους, όπως GPUs και TPUs. Είναι πλούσιο σε λειτουργίες και υποστηρίζει ένα ευρύ φάσμα δυνατοτήτων που απαιτούνται για εκτεταμένη έρευνα και ανάπτυξη στη μηχανική μάθηση. Αντίθετα, το TensorFlow Lite εστιάζει στην εκτέλεση (inference), πράγμα που σημαίνει ότι είναι βελτιστοποιημένο για να εκτελεί προκαθορισμένα μοντέλα σε συσκευές με περιορισμένη υπολογιστική ισχύ και μνήμη. Το TensorFlow Lite, μειώνει το μέγεθος του μοντέλου και βελτιστοποιεί την απόδοση, χρησιμοποιώντας τεχνικές, όπως η ποσοτικοποίηση (quantization) και η κλάδευση (pruning). Αυτές οι βελτιστοποιήσεις, έχουν ως αποτέλεσμα γρηγορότερους χρόνους εκτέλεσης και χαμηλότερη κατανάλωση πόρων, που είναι κρίσιμες για εφαρμογές σε πραγματικό χρόνο σε ενσωματωμένες συσκευές [71][72].

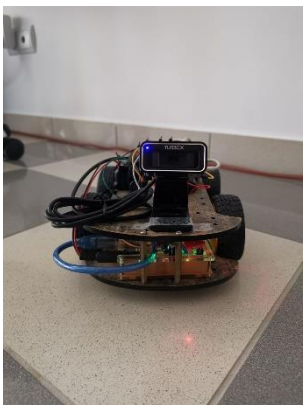
Συνοψίζοντας, το TensorFlow είναι ένα ολοκληρωμένο πλαίσιο για τη δημιουργία και την ανάπτυξη μοντέλων μηχανικής μάθησης, ενώ το TensorFlow Lite είναι μια βελτιστοποιημένη έκδοση που προορίζεται για εκτέλεση σε κινητές και ενσωματωμένες συσκευές. Οι βελτιστοποιήσεις απόδοσης και αποδοτικότητας του TensorFlow Lite, το

καθιστούν την προτιμώμενη επιλογή για την ανάπτυξη μοντέλων μηχανικής μάθησης σε συσκευές όπως το Raspberry Pi.

ΣΧΕΔΙΑΣΜΟΣ ΑΥΤΟΝΟΜΟΥ ΟΧΗΜΑΤΟΣ

Το σύστημα που αναλύεται και υλοποιείται είναι ένα αυτόνομο όχημα, το οποίο διαβάζει, αναγνωρίζει και κινείται αναλόγως με τις πινακίδες κυκλοφορίας. Ένα τέτοιο όχημα είναι σημαντικό να ανταποκρίνεται γρήγορα και κάποιες φορές ακαριαία, καθώς σε περίπτωση καθυστέρησης, αποτυχίας ή λάθους, μπορεί να αποφέρει καταστροφικές συνέπειες όπως μόνιμος τραυματισμός ή ακόμα και απώλεια ζωής. Γι' αυτό, το ακόλουθο παράδειγμα είναι ιδανικό για την ανάδειξη της χρησιμότητας της τοπικής επεξεργασίας, χρησιμοποιώντας ενσωματωμένα συστήματα με την βοήθεια της τεχνητής νοημοσύνης έναντι του νέφους. Λόγω του ότι το κύριο θέμα της διπλωματικής δεν είναι η πολλαπλή λειτουργικότητα του οχήματος, αλλά η δημιουργία του μοντέλου και η ανάδειξη της υπολογιστικής στο άκρο, έχουν υλοποιηθεί λειτουργίες για έξι (6) από τις σαραντατρείς (43) πινακίδες που αναγνωρίζει.

Εικόνα 17 – Πίσω Όψη Οχήματος



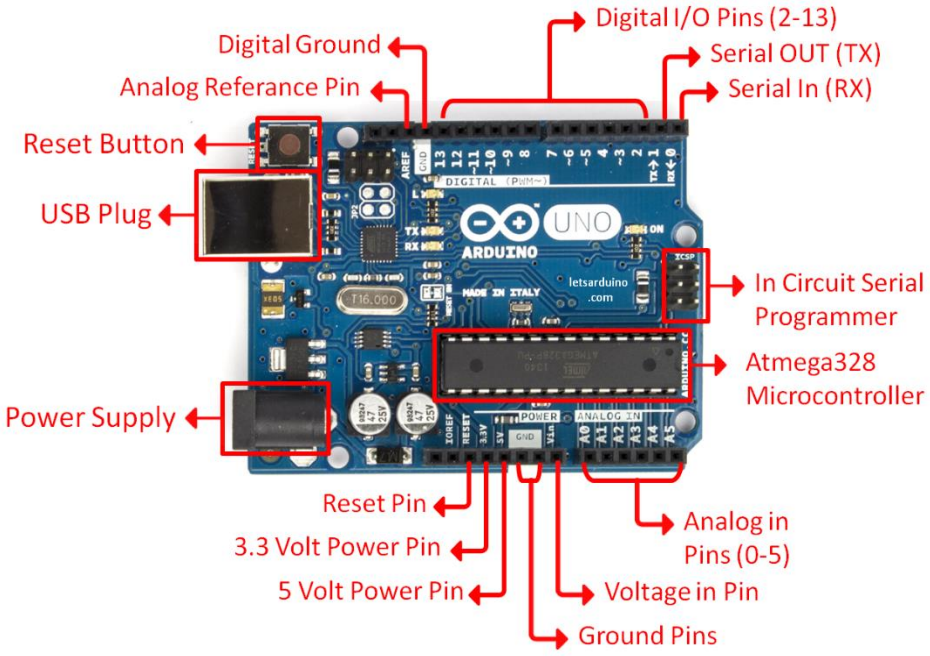
Εικόνα 19 – Μπροστά Όψη Οχήματος

Εικόνα 20 – Αριστερή Πλάγια Όψη Οχήματος

Εικόνα 18 – Δεξιά Πλάγια Όψη Οχήματος

5.1 Arduino Uno R3

Η Arduino UNO R3 είναι μια ανοιχτού κώδικα πλακέτα μικροελεγκτή, βασισμένη στον μικροελεγκτή ATmega328P της Atmel. Διαθέτει τον ATmega328P στα 16 MHz, μνήμες 2KB



Εικόνα 21 - Arduino Uno R3
https://ettron.com/wp-content/uploads/2018/12/labelled-arduino-parts.png

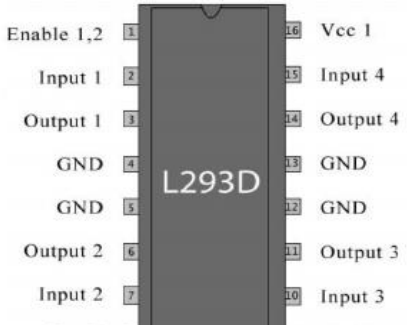
SRAM, 32KB flash, δεκατέσσερις (14) ψηφιακές ακίδες εισόδου/εξόδου (6 από τις οποίες μπορούν να χρησιμοποιηθούν ως έξοδοι PWM), έξι (6) αναλογικές εισόδους, κεραμικό συντονιστή (ceramic resonator) δεκαέξι (16) MHz, σύνδεση USB, υποδοχή τροφοδοσίας, κεφαλίδα ICSP και κουμπί επαναφοράς. Περιλαμβάνει όλα τα απαραίτητα εξαρτήματα για την υποστήριξη του μικροελεγκτή και μπορεί να τροφοδοτηθεί μέσω USB, προσαρμογέα AC-to-DC ή μπαταρία. Επιτρέπει στους χρήστες να διαβάζουν εισόδους (π.χ. δεδομένα αισθητήρα, πατήματα κουμπιών, μηνύματα Twitter) και να παράγουν εξόδους όπως ενεργοποίηση κινητήρων, ενεργοποίηση LED, δημοσίευση στο διαδίκτυο. Οι οδηγίες αποστέλλονται στον μικροελεγκτή, χρησιμοποιώντας το λογισμικό Arduino IDE. Η πλακέτα είναι φιλική προς το χρήστη και επιεικής, με το τσιπ του μικροελεγκτή να μπορεί να αντικατασταθεί, εάν καταστραφεί, καθώς δεν είναι κολλημένο πάνω της [74][75].

Για λόγους ευκολίας, λέγοντας «ο arduino» αναφερόμαστε στην πλακέτα και τον μικροελεγκτή μαζί. Ο Arduino χρησιμοποιήθηκε για τον έλεγχο των κινητήρων (βλ. Εικόνα 10) και ενδεχομένως των διάφορων αισθητηρίων (π.χ αισθητήρας εγγύτητας) που μπορούν να τοποθετηθούν πάνω σε ένα αυτόνομο όχημα. Οι παρακάτω κινητήρες έχουν τάση λειτουργίας 3-12V DC και ρεύμα φορτίου 70mA (3V έως 250mA Max) [76].

Εικόνα 22 - Ηλεκτροκινητήρες Συνεχούς Ρεύματος [76]

5.1.1 Ολοκληρωμένο L293D

Το ολοκληρωμένο αυτό, χρησιμοποιείται για την οδήγηση των κινητήρων. Μπορεί να οδηγήσει ταυτόχρονα δύο κινητήρες συνεχούς ρεύματος, καθώς έχει δύο κανάλια H-bridge, σε οποιαδήποτε κατεύθυνση. Το L293D έχει σχεδιαστεί για την παροχή δυνητικών ρευμάτων κίνησης μέχρι 600mA (ανά κανάλι), σε τάσεις από 4,5V έως 36V. Στη συγκεκριμένη εργασία, έχουμε ένα αυτοκινητάκι το οποίο, έχει τέσσερις ρόδες, άρα τέσσερις κινητήρες, επομένως χρησιμοποιήσαμε δύο ολοκληρωμένα τύπου L293D. Επίσης, επειδή οι ρόδες δεν στρίβουν, βραχυκυκλώσαμε τις δύο στην αριστερή μεριά μεταξύ τους και τις δύο στη δεξιά πάλι μεταξύ τους, έτσι ώστε δίνοντας τάση μόνο στους κινητήρες της μιας μεριάς, το όχημα να στρίβει στην αντίθετη [77].



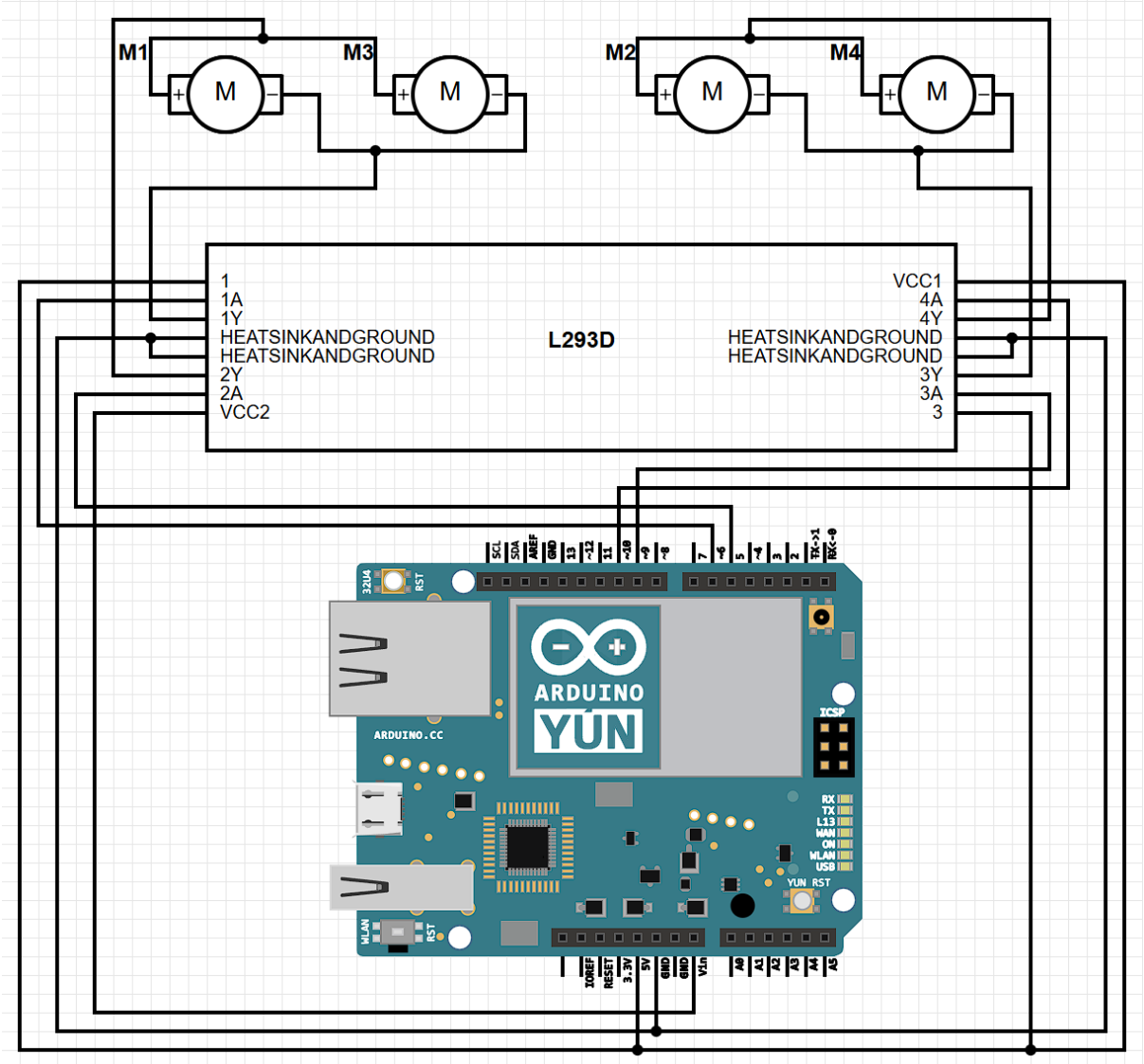
Εικόνα 23 - Ολοκληρωμένο L293D https://electrosome.com/wp-content/uploads/2012/06/L293D-pin-out.jpg

Έχοντας βραχυκυκλώσει τις ακίδες του δεύτερου L293D με του πρώτου, έτσι ώστε οι κινητήρες τρία (3) και τέσσερα (4) να παίρνουν τις ίδιες εντολές με τους ένα (1) και δύο (2) αντίστοιχα, έχουμε την παρακάτω συνδεσμολογία.

Ακίδες Arduino	Ακίδες L293D	DC Κινητήρες
+5V	1	-
6	2	-
-	3	1,3 (-)
GND	4	-
GND	5	-
-	6	1,3 (+)
5	7	-
Vin	8	-
+5V	9	-
9	10	-
-	11	2,4 (-)
GND	12	-
GND	13	-
-	14	2,4 (+)
10	15	-
+5V	16	-

Πίνακας 1 - Συνδεσμολογία μεταξύ arduino, L293D και κινητήρων

Οι ηλεκτροκινητήρες δεν έχουν θετικό και αρνητικό πόλο, τα πρόσημα έχουν μπει για διευκόλυνση περί των καλωδίων, έτσι ώστε στην συνδεσμολογία να μην γίνει κάποιο λάθος και πολωθούν «σωστά» οι δύο κινητήρες και ανάποδα οι άλλοι δύο, στην περίπτωση που θέλουμε το αμάξι να πάει μπροστά. Επίσης, υπάρχει μπαταρία 9V που συνδέεται στην εξωτεική τροφοδοσία του Arduino για την κίνηση των κινητήρων.

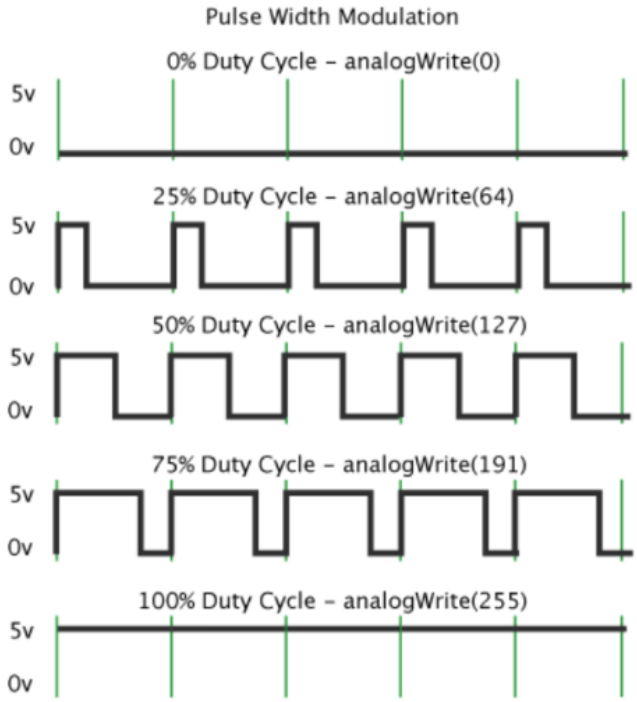


Εικόνα 24 – Συνδεσμολογία Arduino, L293D και Κινητήρων

5.1.2 Κώδικας Arduino

Η διαμόρφωση πλάτους παλμού (PWM), είναι μια τεχνική που χρησιμοποιεί ψηφιακά σήματα για την επίτευξη αναλογικών αποτελεσμάτων. Λειτουργεί δημιουργώντας ένα τετραγωνικό παλμό, εναλλάσσοντας τις καταστάσεις ενεργοποίησης και απενεργοποίησης. Προσαρμώζοντας την αναλογία του χρόνου που είναι ενεργοποιημένο το σήμα (πλάτος παλμού) σε σχέση με τον χρόνο που δεν είναι, η τεχνική της διαμόρφωση πλάτους παλμού, μπορεί να προσομοιώσει μεταβαλλόμενες τάσεις μεταξύ της πλήρους τάσης (Vcc) της πλακέτας (π.χ. 5V σε ένα Arduino UNO) και 0V. Αυτή η διαμόρφωση, όταν επαναλαμβάνεται

γρήγορα, κάνει συσκευές, όπως οι κινητήρες που χρησιμοποιούμε, να φαίνονται ότι λαμβάνουν σταθερή τάση, επιτρέποντας τον έλεγχο της ταχύτητας [78].



Εικόνα 25 - Pulse Width Modulation [78]

Οι κινητήρες, λοιπόν, συνδέονται στις ακίδες του ολοκληρωμένου L293D, οι οποίες με την σειρά τους ελέγχονται από τις PWM εξόδους του arduino. Σημαντικές μεταβλητές αποτελούν η «sign», η οποία είναι η ακέραια τιμή κάθε μιας από τις πινακίδες που αναγνωρίζει το μοντέλο, η «motorsStateChanged», όπου αν έχει δοθεί στους κινητήρες διαφορετικός παλμός από τον αρχικό, τους επαναφέρει στην αρχική τους κατάσταση και η «vSpeed», η οποία καθορίζει την ταχύτητα των κινητήρων.

```
//L293 Connection
const int leftMotorsForw  = 5; // Pin 2 of L293
const int leftMotorsBackw  = 6; // Pin 7 of L293
const int rightMotorsForw  = 9; // Pin 10 of L293
const int rightMotorsBackw  = 10; // Pin 15 of L293

//Useful Variables
int sign; // int value of traffic signs recognised by the model
int vSpeed = 200; // motors speed (0-255 PWM)
bool motorsStateChanged = true;
```

Εικόνα 29 - Μεταβλητές

Για την σειριακή επικοινωνία χρειάζεται να προσθέσουμε στο group «dialout» τον χρήστη μας, έτσι ώστε να έχει πρόσβαση στο αρχείο ttyACM0. Το αρχείο αυτό δημιουργείται στον κατάλογο /dev, όταν συνδέσουμε το arduino στον υπολογιστή με USB καλώδιο.

```
konstantinos@debian:~$ sudo usermod -aG dialout $USER
[sudo] password for konstantinos:
konstantinos@debian:~$ cat /etc/group | grep dialout
dialout:x:20:konstantinos
konstantinos@debian:~$ _
```

Εικόνα 30 - Προσθήκη Χρήστη Στην Ομάδα dialout

Η συνάρτηση setup() εκτελείται μια μόνο φορά. Χρησιμοποιείται για την αρχικοποίηση των ακίδων του arduino, οι οποίες σχετίζονται με τους κινητήρες, ως εξόδους και την έναρξη της σειριακής επικοινωνίας με ρυθμό μετάδοσης 9600 (~1.2 kilobytes) bit ανά δευτερόλεπτο.

```
void setup() {
// Set pins as outputs:
  pinMode(leftMotorsBackw,OUTPUT);
  pinMode(leftMotorsForw,OUTPUT);
  pinMode(rightMotorsBackw,OUTPUT);
  pinMode(rightMotorsForw,OUTPUT);
// Initialize serial communication at 9600 bits per second
  Serial.begin(9600);
}
```

Εικόνα 34 - Αρχικοποίηση Ακίδων Κινητήρων και Σειριακής Επικοινωνίας

Το ακόλουθο απόσπασμα κώδικα εκτελείται συνεχόμενα και αποτελεί τον κύριο κώδικα ελέγχου των κινητήρων. Παρακάτω εξηγούνται οι συναρτήσεις και η λογική του προγράμματος.

- **Serial.available():** Κάθε φορά που αποστέλλονται bytes μέσω της σειριακής πόρτας, αποθηκεύονται σε έναν buffer με μέγιστη χωρητικότητα 64 bytes. Όταν εκτελείται η συνάρτηση, ελέγχει αν υπάρχουν διαθέσιμα δεδομένα στον buffer και διαβάζει ένα-ένα τα bytes [79].

Στην δική μας περίπτωση, τα bytes αυτά αποτελούν τον αριθμό της εκάστοτε πινακίδας. Για παράδειγμα, η πινακίδα «STOP» είναι συσχετισμένη στο μοντέλο με τον αριθμό δεκατέσσερα (14). Ανάλογα με την πινακίδα τροφοδοτούνται οι κατάλληλοι κινητήρες και το αμάξι κινείται αντίστοιχα. Αυτό γίνεται με την συνάρτηση analogWrite().

- **analogWrite():** Η συνάρτηση αυτή, δημιουργεί ένα παλμό PWM σε μια καθορισμένη ακίδα, επιτρέποντας τον έλεγχο της φωτεινότητας των LED ή της ταχύτητας του κινητήρα. Ο παλμός αυτός, παραμένει, έως ότου πραγματοποιηθεί μια άλλη κλήση στην analogWrite(), digitalRead() ή digitalWrite(), που να αφορά την ίδια ακίδα [80].

Χρησιμοποιώντας την συνάρτηση delay(), για να σταματήσουμε προσωρινά την ροή του προγράμματος, μπορούμε να καθορίσουμε την διάρκεια του παλμού και επομένως την διάρκεια της κίνησης.

```
void loop() {
  if (motorsStateChanged){
    // Reset sign
    sign = -1;
    analogWrite(leftMotorsForw, vSpeed); analogWrite(leftMotorsBackw, 0);
    analogWrite(rightMotorsForw, vSpeed); analogWrite(rightMotorsBackw, 0);
    motorsStateChanged = false;
  }
  // If there are no data available there is no reason to check if you got a sign
  if(Serial.available() == 0){
    return;
  }
  sign = Serial.read();

  /************************Stop****************************/
  if (sign == 14) {
    Serial.println("Stop sign detected.");
    analogWrite(leftMotorsForw, 0); analogWrite(leftMotorsBackw, 0);
    analogWrite(rightMotorsForw, 0); analogWrite(rightMotorsBackw, 0);
    motorsStateChanged = true;
    delay(3000);
  }
  /*********************Forward Left***********************/
  else if (sign == 39) {
    Serial.println("Keep left sign detected.");
    analogWrite(leftMotorsForw, vSpeed-(0.5*vSpeed)); analogWrite(leftMotorsBackw, 0);
    analogWrite(rightMotorsForw, vSpeed); analogWrite(rightMotorsBackw, 0);
    motorsStateChanged = true;
    delay(3000);
  }
  /*********************Forward Right**********************/
  else if (sign == 38) {
    Serial.println("Keep right sign detected.");
    analogWrite(leftMotorsForw, vSpeed); analogWrite(leftMotorsBackw, 0);
    analogWrite(rightMotorsForw, vSpeed-(0.5*vSpeed)); analogWrite(rightMotorsBackw, 0);
    motorsStateChanged = true;
    delay(3000);
  }
  /*********************Speed Limit 50km/h*************************/
  else if (sign == 2 ) {
    Serial.println("Speed Limit 50km/h sign detected.");
    if (vSpeed != 50) {
      vSpeed = 50;
      motorsStateChanged = true;
    }
  }
  /************************Turn Left****************************/
  else if (sign == 34) {
    Serial.println("Turn left sign detected.");
    analogWrite(leftMotorsForw, 0); analogWrite(leftMotorsBackw, vSpeed);
    analogWrite(rightMotorsForw, vSpeed); analogWrite(rightMotorsBackw, 0);
    motorsStateChanged = true;
    delay(1500);
  }
  /************************Turn Right****************************/
  else if (sign == 33) {
    Serial.println("Turn right sign detected.");
    analogWrite(leftMotorsForw, vSpeed); analogWrite(leftMotorsBackw, 0);
    analogWrite(rightMotorsForw, 0); analogWrite(rightMotorsBackw, vSpeed);
    motorsStateChanged = true;
    delay(1500);
  }
}
```

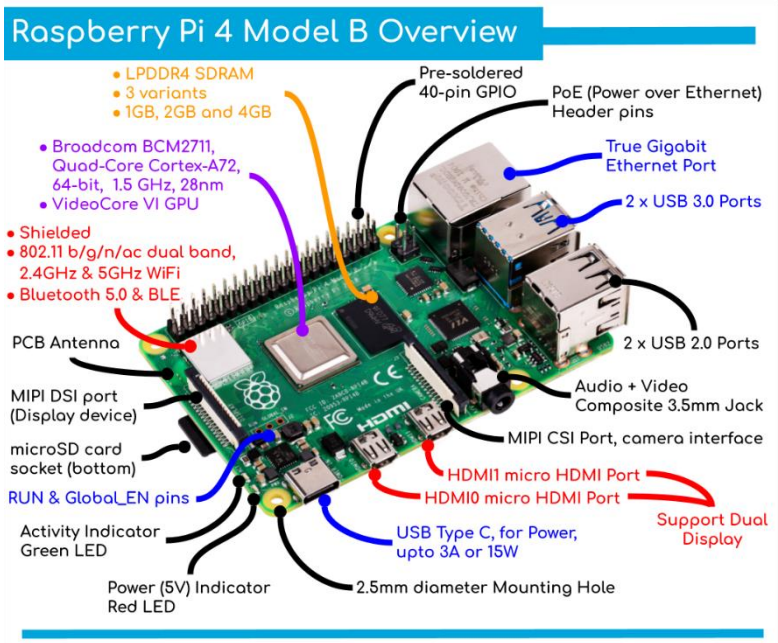
Εικόνα 38 - Βασικός Κώδικας Οχήματος

5.2 Raspberry Pi

Το Raspberry Pi, είναι μια σειρά μικρών υπολογιστών μιας πλακέτας (single-board computers) που αναπτύχθηκε από το Raspberry Pi Foundation στο Ηνωμένο Βασίλειο, σε συνεργασία με την Broadcom. Αρχικά είχε στόχο την προώθηση της βασικής εκπαίδευσης στην επιστήμη των υπολογιστών στα σχολεία. Το αρχικό μοντέλο κέρδισε απροσδόκητα δημοτικότητα πέρα από την αγορά για την οποία προοριζόταν. Χρησιμοποιείται ευρέως σε τομείς όπως η ρομποτική, ο οικιακός και βιομηχανικός αυτοματισμός και από χομπίστες υπολογιστών και ηλεκτρονικών, λόγω του χαμηλού κόστους, της αρθρωτής δομής, του ανοιχτού σχεδιασμού και της συμβατότητας με τα πρότυπα HDMI και USB [81]. Λόγω του μεγέθους, της διασυνδεσιμότητας, της χαμηλής κατανάλωσης ενέργειας και του αρκετά δυνατού υλικού του, είναι ιδανικό για εφαρμογές όπως ένα αυτόνομο όχημα. Συνεπώς, χρησιμοποιήθηκε στην εφαρμογή, ως το μέσο επεξεργασίας των δεδομένων για την αναγνώριση των πινακίδων. Το μοντέλο που χρησιμοποιήθηκε είναι το Raspberry Pi 4B με τα 4GB μνήμης, του οποίου τα χαρακτηριστικά αναφέρονται στην συνέχεια.

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
- 4GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (πλήρως συμβατό με προηγούμενες πλακέτες)
- 2 × micro-HDMI® ports (υποστηρίζει ανάλυση μέχρι 4k 60 frames per second)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.1, Vulkan 1.0
- 16GB Micro-SD για το λειτουργικό σύστημα και για αποθηκευτικό χώρο
- 5V DC via USB-C connector (ελάχιστο 3A*)
- 5V DC via GPIO header (ελάχιστο 3A*)

- Power over Ethernet (PoE) enabled (απαιτεί ξεχωριστό PoE HAT)



Εικόνα 42 - Raspberry Pi 4B
https://static.cytron.io/image/catalog/products/RASPBERRY-PI-4B-8G/RASPBERRY-PI-4B-8G-5.jpg

Παρόλο που έχει και αυτό γενικού σκοπού ακίδες εισόδου-εξόδου (GPIO) και θα μπορούσε να χρησιμοποιηθεί για τον έλεγχο των κινητήρων και των διάφορων αισθητήρων και πρακτικά θα μπορούσε να αντικαταστήσει το arduino, δεν προτιμήθηκε, καθώς η εκτέλεση του συνελικτικού νευρωνικού δικτύου για την αναγνώριση των πινακίδων, είναι αρκετά βαριά εργασία και απαιτεί αρκετούς πόρους συστήματος. Σε συνέχεια αυτού, οι δύο αυτές πλατφόρμες επικοινωνούν μέσω σειριακής επικοινωνίας, από την πλευρά του arduino μέσω USB-B και από τη μεριά του raspberry μέσω USB-A.

Η κάμερα που χρησιμοποιήθηκε είναι μια απλή USB webcam με ανάλυση 720p της TURBOX συνδεδεμένη και αυτή με το raspberry pi.

Η σειριακή επικοινωνία με το arduino, από την μεριά του raspberry pi και της python, πραγματοποιείται με την χρήση της βιβλιοθήκης pyserial και τον ακόλουθο κώδικα [82]:

```
arduino = serial.Serial(port='/dev/ttyACM0', baudrate=9600)
```

Εικόνα 46 - Αρχικοποίηση Σειριακής Επικοινωνίας Σε Python

5.3 Δημιουργία Μοντέλου

Η διαδικασία της επεξεργασίας του βίντεο και της αναγνώρισης των πινακίδων, που υπάρχουν μέσα σε αυτό, γίνεται μέσω ενός συνελικτικού νευρωνικού δικτύου (CNN), το οποίο, όπως προαναφέρθηκε, είναι ιδανικό για εργασίες ταξινόμησης εικόνων. Για την δημιουργία του, χρησιμοποιήθηκαν το σύνολο δεδομένων (dataset) German Traffic Sign Detection Benchmark (GTSDB) και οι βιβλιοθήκες, για την python, os, tensorflow, openCV, numpy, pandas, matplotlib, scikit-learn και keras.

```
import os
import cv2
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.optimizers import Adam
from keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

path = 'Dataset'
labelFile = 'labels.csv'
batch_size_val = 32
epochs_val = 20
imageDimesions = (32,32,3)
testRatio = 0.2
validationRatio = 0.2
```

Εικόνα 50 - Βιβλιοθήκες και Αρχικοποίηση Μεταβλητών

5.3.1 Προετοιμασία Δεδομένων

Το αρχικό βήμα περιλαμβάνει την προετοιμασία των δεδομένων για το μοντέλο. Το σύνολο των δεδομένων περιλαμβάνει εικόνες σημάτων κυκλοφορίας και τις αντίστοιχες ετικέτες για την κατηγορίας τους.

```
count = 0
images = []
classNo = []
myList = os.listdir(path)
print('Total Classes Detected:',len(myList))
noOfClasses=len(myList)
print('Importing Classes.....')
for x in range (0,len(myList)):
    myPicList = os.listdir(path+'/'+str(count))
    for y in myPicList:
        curImg = cv2.imread(path+'/'+str(count)+'/'+y)
        images.append(curImg)
        classNo.append(count)
    print(count, end =' ')
    count +=1
print(' ')
images = np.array(images)
classNo = np.array(classNo)
```

Εικόνα 54 - Προετοιμασία Δεδομένων

Ξεκινάμε φορτώνοντας τις εικόνες και τις ετικέτες των κλάσεων τους σε λίστες (images, classNo) στη μνήμη του υπολογιστή και στην συνέχεια μετατρέπουμε τις λίστες σε πίνακες NumPy καθώς οι πίνακες αυτοί παρέχουν καλύτερη και ευκολότερη διαχείριση των δεδομένων, συμβατότητα με την βιβλιοθήκη tensorflow και περεταίρω μαθηματικές λειτουργίες.

5.3.2 Διαχωρισμός Δεδομένων

Το σύνολο δεδομένων χωρίζεται σε σύνολα εκπαίδευσης, επικύρωσης και δοκιμών του μοντέλου για να διασφαλιστεί ότι η απόδοση του μπορεί να αξιολογηθεί επαρκώς.

```
images_train, images_test, labels_train, labels_test = train_test_split(images, classNo, test_size=testRatio)
images_train, images_validation, labels_train, labels_validation = train_test_split(images_train, labels_train, test_size=validationRatio)
```

Εικόνα 58 - Διαχωρισμός Δεδομένων

Η συνάρτηση train_test_split() χωρίζει πίνακες σε τυχαία υποσύνολα εκπαίδευσης και δοκιμής. Η πρώτη γραμμή χωρίζει τις εικόνες, όπου το υποσύνολο των δεδομένων εκπαίδευσης είναι μειωμένο κατά είκοσι τοις εκατό (20%) και το υποσύνολο των δεδομένων

δοκιμών αποτελείται από αυτό το είκοσι τοις εκατό (20%) των δεδομένων εκπαίδευσης. Το ίδιο κάνει και για τις ετικέτες. Η δεύτερη γραμμή χωρίζει τα υποσύνολα εκπαίδευσης, ετικετών και εικόνων, που δημιούργησε η πρώτη, σε νέα υποσύνολα εκπαίδευσης και επικύρωσης. Τα υποσύνολα εκπαίδευσης είναι μειωμένα κατά είκοσι τοις εκατό (20%) και τα υποσύνολα επικύρωσης αποτελούνται από το είκοσι τοις εκατό (20%) των δεδομένων εκπαίδευσης.

5.3.3 Τύπωση Δομής Εικόνων

Με τις παρακάτω εντολές τυπώνουμε την δομή των εικόνων και τον αριθμό των ετικετών που θα χρησιμοποιηθούν για την δημιουργία του μοντέλου, έτσι ώστε να επιβεβαιώσουμε ότι είναι σωστά.

```
print('Data Shapes')
print('Train',end = '');print(images_train.shape,labels_train.shape)
print('Validation',end = '');print(images_validation.shape,labels_validation.shape)
print('Test',end = '');print(images_test.shape,labels_test.shape)
```

Εικόνα 63 - Τύπωση Δομής Εικόνων

```
Data Shapes
Train(22271, 32, 32, 3) (22271,)
Validation(5568, 32, 32, 3) (5568,)
Test(6960, 32, 32, 3) (6960,)
```

Εικόνα 62 - Δομή Εικόνων

Παρατηρούμε πως οι εικόνες είναι μεγέθους 32x32 εικονοστοιχείων (pixel). Ο τελευταίος δείκτης (3) υποδεικνύει τα τρία διαφορετικά κανάλια χρωμάτων, Κόκκινο, Πράσινο και Μπλε (RGB).

5.3.4 Προεπεξεργασία Εικόνων

Παρακάτω ορίζουμε συναρτήσεις για την προεπεξεργασία των εικόνων, συμπεριλαμβανομένης της μετατροπής σε κλίμακα του γκρι (grayscale conversion), της εξισορρόπησης ιστογράμματος (histogram equalization) και της κανονικοποίησης (normalization).

- **grayscale()**: Μετατρέπει μια εικόνα σε κλίμακα του γκρι, τιμές φωτεινότητας εικονοστοιχείων (0-255). Αυτό γίνεται γιατί οι RGB εικόνες προσθέτουν πολυπλοκότητα λόγω περισσότερων διαστάσεων και μεγαλύτερη πολυπλοκότητα σημαίνει περισσότερη ανάγκη για υπολογιστική ισχύ και μνήμη.

- **equalize()**: Εφαρμόζει την εξίσωση ιστογράμματος στην εικόνα. Η εξισορρόπηση ιστογράμματος βελτιώνει την αντίθεση μιας εικόνας απλώνοντας αποτελεσματικά τις πιο συχνά εμφανιζόμενες τιμές έντασης φωτεινότητας των εικονοστοιχείων. Αυτό οδηγεί σε μια εικόνα όπου οι εντάσεις κατανέμονται πιο ομοιόμορφα σε όλο το ιστόγραμμα, βελτιώνοντας την ορατότητα των λεπτομερειών τόσο σε φωτεινές όσο και σε σκοτεινές περιοχές. Έτσι το μοντέλο μπορεί να διακρίνει καλύτερα μεταξύ των διαφορετικών χαρακτηριστικών της εικόνας, οδηγώντας ενδεχομένως σε καλύτερη απόδοση.
- **preprocessing()**: Αυτή η συνάρτηση πραγματοποιεί την επεξεργασία της εικόνας και προσθέτει ένα τελευταίο επίπεδο, αυτό της κανονικοποίησης. Η κανονικοποίηση διασφαλίζει ότι οι τιμές των εικονοστοιχείων των εικόνων έρχονται σε μια κοινή κλίμακα [0,1], η οποία είναι ζωτικής σημασίας για τη συνεπή και αποτελεσματική εκπαίδευση των μοντέλων μηχανικής εκμάθησης. Βοηθά στον μετριασμό των επιπτώσεων των διαφορετικών συνθηκών φωτισμού, των διαφορετικών συσκευών απεικόνισης και άλλων βημάτων προεπεξεργασίας, οδηγώντας σε καλύτερη γενίκευση και απόδοση του μοντέλου.

```
def grayscale(image):
    image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    return image
def equalize(image):
    image =cv2.equalizeHist(image)
    return image
def preprocessing(image):
    image = grayscale(image)
    image = equalize(image)
    image = image/255
    return image

images_train=np.array(list(map(preprocessing,images_train)))
images_validation=np.array(list(map(preprocessing,images_validation)))
images_test=np.array(list(map(preprocessing,images_test)))

images_train=images_train.reshape(images_train.shape[0],images_train.shape[1],images_train.shape[2],1)
images_validation=images_validation.reshape(images_validation.shape[0],images_validation.shape[1],images_validation.shape[2],1)
images_test=images_test.reshape(images_test.shape[0],images_test.shape[1],images_test.shape[2],1)
```

Εικόνα 67 - Προεπεξεργασία Εικόνων

Αυτή η επεξεργασία εφαρμόζεται και στα τρια σύνολα εικόνων (εκπαίδευσης, δοκιμής, επικύρωσης). Μετά τη μετατροπή των εικόνων σε κλίμακα του γκρι, τα δεδομένα τους είναι συνήθως ένας πίνακας 2D με διαστάσεις (ύψος, πλάτος). Ωστόσο, τα συνελικτικά νευρωνικά δίκτυα που δημιουργούνται από βιβλιοθήκες όπως η TensorFlow και Keras, συνήθως αναμένουν ότι η εικόνα εισόδου θα έχει σχήμα (ύψος, πλάτος, κανάλια) όπου τα κανάλια υποδεικνύουν τον αριθμό των έγχρωμων καναλιών. Για τις εικόνες σε κλίμακα του γκρι, η

διάσταση καναλιών θα πρέπει να είναι ένα (1). Επομένως, πρέπει να επαναδιαμορφωθεί ο δισδιάστατος πίνακας κάθε εικόνας σε τρισδιάστατο, προσθέτοντας ρητά μια διάσταση ενός καναλιού.

5.3.5 Επαύξηση Εικόνων

Σε αυτό το στάδιο δημιουργούμε και εφαρμόζουμε στο σύνολο εκπαίδευσης έναν «ImageDataGenerator» ο οποίος χρησιμοποιείται για την επαύξηση των εικόνων σε πραγματικό χρόνο κατά την εκπαίδευση μοντέλων μηχανικής μάθησης. Η επαύξηση δεδομένων είναι μια τεχνική για την τεχνητή αύξηση του μεγέθους και της ποικιλομορφίας του συνόλου των δεδομένων εκπαίδευσης με την εφαρμογή τυχαίων μετασχηματισμών (π.χ περιστροφή, μεγέθυνση, μετατόπιση πλάτους και ύψους) στις υπάρχουσες εικόνες. Οι τιμές 0.1 και 0.2 αφορούν την κλίμακα [0,1] και πρακτικά σημαίνουν 10% ή 20% αντίστοιχα (π.χ 20% μεγέθυνση εικόνας). Αυτό βοηθάει στη βελτίωση της γενίκευσης του μοντέλου εκθέτοντάς το σε μια ευρύτερη ποικιλία παραλλαγών των εικόνων κατά τη διάρκεια της εκπαίδευσης.

- **dataGen.fit():** Η μέθοδος αυτή υπολογίζει στατιστικά στοιχεία όπως ο μέσος όρος και η τυπική απόκλιση των τιμών των εικονοστοιχείων στις εικόνες εκπαίδευσης. Αυτά τα στατιστικά στοιχεία χρησιμοποιούνται στη συνέχεια για την τυποποίηση (standardization) των εικόνων, η οποία είναι κρίσιμη για την βέλτιστη απόδοση του μοντέλου. Η τυποποίηση εφαρμόζεται αφαιρώντας τον μέσο όρο και διαιρώντας με την τυπική απόκλιση σε κάθε εικονοστοιχείο.

```
dataGen= ImageDataGenerator(width_shift_range=0.1,
                            height_shift_range=0.1,
                            zoom_range=0.2,
                            shear_range=0.1,
                            rotation_range=10)
dataGen.fit(images_train)
```

Εικόνα 71 - Επαύξηση Εικόνων

Η χρήση παρτίδων στη βαθιά εκμάθηση είναι απαραίτητη για την αποτελεσματική χρήση της μνήμης, τον ταχύτερο υπολογισμό και την αξιοποίηση των δυνατοτήτων παράλληλης επεξεργασίας του σύγχρονου υλικού.

5.3.6 Κωδικοποίηση One-hot

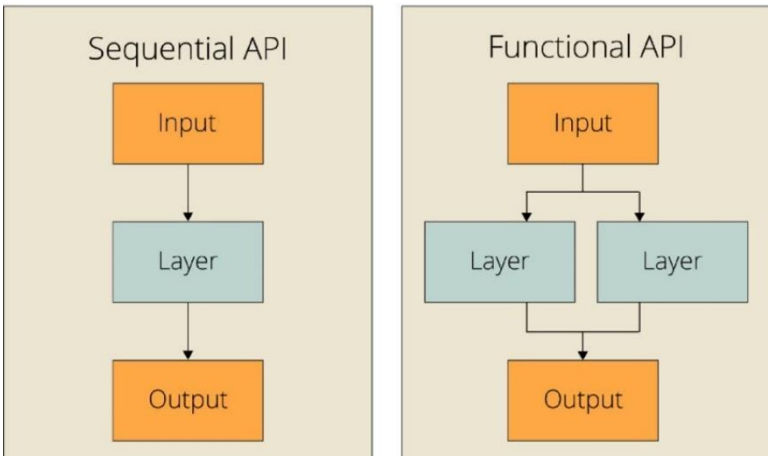
Τα νευρωνικά δίκτυα χρησιμοποιούν συνήθως συναρτήσεις απώλειας (loss functions) όπως η κατηγορική διασταυρούμενη εντροπία (categorical cross-entropy) για εργασίες ταξινόμησης. Αυτές οι συναρτήσεις απαιτούν οι ετικέτες να είναι κωδικοποιημένες στη μορφή one-hot. Η κωδικοποίηση αυτή μετατρέπει τις κατηγορικές ετικέτες σε μια αναπαράσταση δυαδικού πίνακα, όπου κάθε ετικέτα αναπαρίσταται ως διάνυσμα μηδενικών με έναν μόνο άσο, ο οποίος βρίσκεται στον δείκτη που αντιστοιχεί στην ετικέτα. Για παράδειγμα, εάν υπάρχουν 3 κλάσεις (0, 1, 2), η κωδικοποίηση one-hot μετατρέπει την ετικέτα 1 σε [0, 1, 0]. Αυτό παρακάτω εφαρμόζεται και στους τρεις πίνακες ετικετών.

```
labels_train = to_categorical(labels_train,noOfClasses)
labels_validation = to_categorical(labels_validation,noOfClasses)
labels_test = to_categorical(labels_test,noOfClasses)
```

Εικόνα 75 - Κωδικοποίηση One-hot

5.3.7 Συνελικτικό Νευρωνικό Δίκτυο

Ένα μοντέλο είναι ένα αντικείμενο που ομαδοποιεί επίπεδα και το οποίο μπορεί να εκπαιδευτεί σε δεδομένα. Ο απλούστερος τύπος μοντέλου είναι το διαδοχικό (Sequential), το οποίο είναι μια γραμμική στοίβα από διαδοχικά στρώματα (layers). Ένα τέτοιο μοντέλο είναι κατάλληλο για μια απλή στοίβα στρωμάτων όπου κάθε στρώμα έχει ακριβώς έναν τανυστή εισόδου και έναν τανυστή εξόδου. Οι τανυστές (tensors) είναι πολυδιάστατοι πίνακες τύπου dtype.



Εικόνα 79 - Sequential and Functional Models

Το μοντέλο αποτελείται από έντεκα (11), συνολικά, στρώματα εκ των οποίων ένα είναι της εισόδου, ένα της εξόδου και τα υπόλοιπα είναι κρυφά (hidden). Όλα τα στρώματα προστέθηκαν στο μοντέλο μέσω της μεθόδου add().

- **Στρώμα Εισόδου (Input Layer) Conv2D:** Το πρώτο συνελικτικό στρώμα εφαρμόζει εξήντα (60) φίλτρα μεγέθους 5x5 στις εικόνες εισόδου. Αυτό το επίπεδο καταγράφει βασικά χαρακτηριστικά, όπως άκρες (edges) και υφές (textures) από τις εικόνες εισόδου. Η συνάρτηση ενεργοποίησης διορθωμένης γραμμικής μονάδας (ReLU activation function) εισάγει τη μη γραμμικότητα, επιτρέποντας στο μοντέλο να μάθει πιο πολύπλοκα μοτίβα. Η συνάρτηση αυτή είναι μια συνάρτηση που υπολογίζει την έξοδο του νευρώνα με βάση τις επιμέρους εισόδους του και τα βάρη τους.
- **Δεύτερο Στρώμα (Layer 2) Conv2D:** Το δεύτερο συνελικτικό στρώμα επεξεργάζεται περαιτέρω τα χαρακτηριστικά που εξάγονται από το πρώτο στρώμα, επιτρέποντας την ανίχνευση πιο περίπλοκων μοτίβων. Η χρήση του ίδιου μεγέθους φίλτρου βοηθά στην ανάπτυξη των αρχικών χαρτών χαρακτηριστικών (features maps) που δημιουργήθηκαν από το πρώτο επίπεδο.
- **Τρίτο Στρώμα (Layer 3) MaxPooling2D:** Η μέγιστη συγκέντρωση (max pooling) μειώνει τις χωρικές διαστάσεις (spatial dimensions) των χαρακτηριστικών λαμβάνοντας τη μέγιστη τιμή σε κάθε μπλοκ 2x2. Αυτό μειώνει το υπολογιστικό φορτίο καθιστώντας το μοντέλο πιο ισχυρό σε μικρές μεταφράσεις της εισόδου.
- **Τέταρτο Στρώμα (Layer 4) Conv2D:** Το τρίτο συνελικτικό στρώμα, με μικρότερο μέγεθος φίλτρου (3x3), εντοπίζει περισσότερες λεπτομέρειες στους χάρτες χαρακτηριστικών. Η μείωση του αριθμού των φίλτρων (από 60 σε 30) εστιάζει τη μάθηση σε πιο συγκεκριμένα χαρακτηριστικά.
- **Πέμπτο Στρώμα (Layer 5) Conv2D:** Παρόμοια με το προηγούμενο επίπεδο, αυτό το επίπεδο συνεχίζει να βελτιώνει τους χάρτες χαρακτηριστικών, δίνοντας έμφαση στην εξαγωγή λεπτομερών και πολύπλοκων χαρακτηριστικών.
- **Έκτο Στρώμα (Layer 6) MaxPooling2D:** Αυτό το δεύτερο επίπεδο μέγιστης συγκέντρωσης μειώνει περαιτέρω τις χωρικές διαστάσεις (ύψος, πλάτος), συνεχίζοντας να παρέχει την ικανότητα στο μοντέλο να αναγνωρίζει μοτίβα ή χαρακτηριστικά σε μια

εικόνα ανεξάρτητα από τη χωρική τους θέση και μειώνοντας τον αριθμό των παραμέτρων.

- **Έβδομο Στρώμα (Layer 7) Dropout:** Το Dropout είναι μια τεχνική τακτοποίησης (regularization) που βοηθά στην αποφυγή της υπερβολικής προσαρμογής (overfitting) απορρίπτοντας τυχαία το 50% των ενεργοποιήσεων κατά τη διάρκεια της εκπαίδευσης. Αυτό ενθαρρύνει το μοντέλο να μάθει πιο ισχυρά χαρακτηριστικά.
- **Όγδοο Στρώμα (Layer 8) Flatten:** Το επίπεδο αυτό μετατρέπει τους χάρτες 2D χαρακτηριστικών σε ένα διάνυσμα (1D), προετοιμάζοντας τα δεδομένα για τα πλήρως συνδεδεμένα πυκνά στρώματα (dense layers).
- **Ένατο Στρώμα (Layer 9) Dense:** Το στρώμα αυτό έχει πεντακόσιους (500) νευρώνες και εκτελεί συλλογισμό υψηλού επιπέδου με βάση τα εξαγόμενα χαρακτηριστικά. Η συνάρτηση ενεργοποίησης ReLU προσθέτει και εδώ μη γραμμικότητα, επιτρέποντας την εκμάθηση πολύπλοκων σχέσεων.
- **Δέκατο Στρώμα (Layer 10) Dropout:** Άλλο ένα επίπεδο απόρριψης για την αποφυγή υπερ προσαρμογής στα δεδομένα εισόδου.
- **Ενδέκατο Στρώμα (Output Layer) Dense:** Στο τελικό στρώμα ο αριθμός των νευρώνων ισούται με τον αριθμό των κλάσεων που δώσαμε στο μοντέλο. Το στρώμα αυτό εξάγει μια κατανομή πιθανότητας στις κλάσεις των εικόνων, χρησιμοποιώντας τη συνάρτηση ενεργοποίησης softmax. Η συνάρτηση αυτή μετατρέπει ένα διάνυσμα K πραγματικών αριθμών σε κατανομή πιθανότητας K πιθανών αποτελεσμάτων. Αυτό επιτρέπει στο μοντέλο να κάνει προβλέψεις σχετικά με την κατηγορία στην οποία ανήκει η εικόνα εισόδου.

Αφού έχουμε εισάγει και ρυθμίσει τα στρώματα του μοντέλου, χρησιμοποιούμε την μέθοδο compile() για να ρυθμίσουμε τις παραμέτρους εκπαίδευσης του. Οι παράμετροι που χρησιμοποιήθηκαν είναι optimizer, loss function και metrics.

- **Optimizer (Adam):** Ο Adaptive Moment Estimation είναι ένας αλγόριθμος βελτιστοποίησης που μπορεί να χρησιμοποιηθεί αντί της κλασικής διαδικασίας στοχαστικής κλίσης κατάβασης (gradient descent) για την ενημέρωση των βαρών του δικτύου με βάση τα δεδομένα εκπαίδευσης. Έχει επιλεγεί για τις ιδιότητες προσαρμοστικού ρυθμού εκμάθησης και τον αποτελεσματικό υπολογισμό του. Ο

ρυθμός εκμάθησης 0,001 είναι ένα κοινό σημείο εκκίνησης που εξισορροπεί την ταχύτητα και τη σταθερότητα της σύγκλισης.

- **Loss Function (Categorical Crossentropy):** Αυτή η συνάρτηση είναι κατάλληλη για προβλήματα ταξινόμησης πολλών κατηγοριών. Μετρά την απόδοση του μοντέλου συγκρίνοντας την προβλεπόμενη κατανομή πιθανοτήτων με την αληθινή κατανομή (one-hot κωδικοποιημένες ετικέτες).
- **Metrics (Accuracy):** Η ακρίβεια είναι μια απλή και διαισθητική μέτρηση για την αξιολόγηση της απόδοσης των μοντέλων ταξινόμησης. Μετρά την αναλογία των σωστά ταξινομημένων εικόνων από το σύνολο των εικόνων.

```python
def myModel():
    model= Sequential()
    model.add((Conv2D(60, (5,5), input_shape=(imageDimesions[0],imageDimesions[1],1),activation='relu')))
    model.add((Conv2D(60, (5,5), activation='relu')))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add((Conv2D(30, (3,3), activation='relu')))
    model.add((Conv2D(30, (3,3), activation='relu')))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(500,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses,activation='softmax'))
    model.compile(Adam(learning_rate=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
    return model
```

Εικόνα 85 - Δημιουργία Μοντέλου Συνελικτικού Νευρωνικού Δικτύου

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 60)	1,560
conv2d_1 (Conv2D)	(None, 24, 24, 60)	90,060
max_pooling2d (MaxPooling2D)	(None, 12, 12, 60)	0
conv2d_2 (Conv2D)	(None, 10, 10, 30)	16,230
conv2d_3 (Conv2D)	(None, 8, 8, 30)	8,130
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 30)	0
dropout (Dropout)	(None, 4, 4, 30)	0
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 500)	240,500
dropout_1 (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 43)	21,543

Total params: 378,023 (1.44 MB)
Trainable params: 378,023 (1.44 MB)
Non-trainable params: 0 (0.00 B)

Εικόνα 92 - Στρώματα Μοντέλου

5.4 Εκπαίδευση Μοντέλου

Η εκπαίδευση του μοντέλου γίνεται μέσω της μεθόδου fit().

- **dataGen.flow():** Δέχεται τις εκπαιδευτικές εικόνες και τις ετικέτες ως είσοδο και δημιουργεί παρτίδες (batches) επαυξημένων ζευγών εικόνας-ετικέτας, σε πραγματικό χρόνο. Αυτό σημαίνει ότι μετασχηματισμοί, όπως περιστροφές, μετατοπίσεις, ανατροπές κ.λ.π εφαρμόζονται στις εικόνες κατά τη διάρκεια της εκπαίδευσης για να αυξηθεί τεχνητά η ποικιλομορφία του συνόλου δεδομένων εκπαίδευσης. Οι μετασχηματισμοί αυτοί έχουν οριστεί στον ImageDataGenerator, σε προηγούμενο βήμα.
    - **steps_per_epoch:** Είναι ο συνολικός αριθμός παρτίδων που υποβάλλονται σε επεξεργασία ανά epoch. Διαιρώντας τον συνολικό αριθμό των εικόνων εκπαίδευσης με τον αριθμό των εικόνων ανά παρτίδα, έχουμε τον συνολικό αριθμό των παρτίδων που χρειάζονται έτσι ώστε το μοντέλο να αναλύσει όλες τις εικόνες από μία φορά σε κάθε epoch.
    - **epochs:** Αυτή η παράμετρος καθορίζει τον αριθμό των πλήρων περασμάτων από το σύνολο δεδομένων εκπαίδευσης. Κάθε εποχή σημαίνει ότι το μοντέλο έχει δει ολόκληρο το σύνολο δεδομένων μία φορά.
    - **validation Data:** Αυτή η παράμετρος είναι ένα tuple που περιέχει τις εικόνες και τις ετικέτες επικύρωσης. Αυτά τα δεδομένα χρησιμοποιούνται για την αξιολόγηση της απόδοσης του μοντέλου μετά από κάθε εποχή, αλλά δεν χρησιμοποιούνται για εκπαίδευση.
    - **shuffle:** Αυτή η τιμή είναι δυαδική [0,1] και καθορίζει αν θα ανακατευτεί η σειρά των δειγμάτων στην αρχή κάθε εποχής.

Η διαδικασία της εκπαίδευσης με βάση τα προηγούμενα αποτελείται από τα παρακάτω βήματα:

- **Forward Pass**: Για κάθε παρτίδα, το μοντέλο κάνει προβλέψεις για τα δεδομένα εισόδου.
- **Loss Calculation**: Το μοντέλο υπολογίζει την απώλεια συγκρίνοντας τις προβλέψεις του με τις αληθινές ετικέτες.

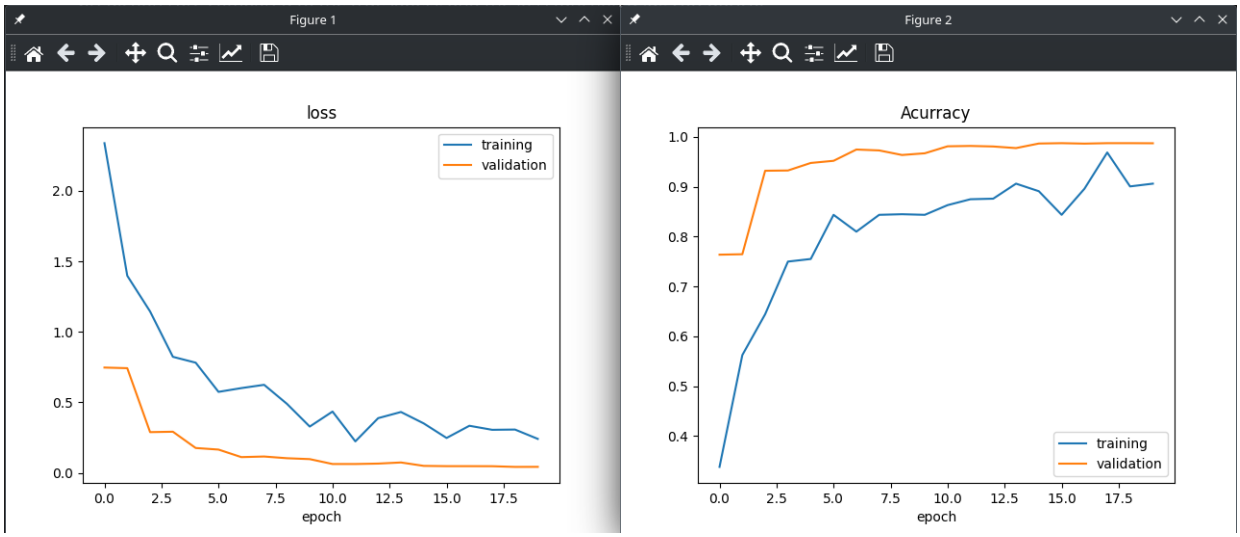
- **Backward Pass and Weight Update**: Το μοντέλο ενημερώνει τα βάρη του με βάση τις κλίσεις (gradients) της απώλειας, χρησιμοποιώντας backpropagation και τον καθορισμένο βελτιστοποιητή (π.χ., Adam).

```
model = myModel()
print(model.summary())
history=model.fit(dataGen.flow(images_train,labels_train,batch_size=32),
                  steps_per_epoch=len(images_train)//32,
                  epochs=epochs_val,
                  validation_data=(images_validation,labels_validation),
                  shuffle=1)
```

Εικόνα 96 - Εκπαίδευση Μοντέλου

5.5 Αξιολόγηση Μοντέλου

Μετά το τέλος της εκπαίδευσης, η μέθοδος fit() επιστρέφει στην μεταβλητή history ένα αντικείμενο, το οποίο περιέχει πληροφορίες σχετικά με τη διαδικασία εκπαίδευσης, συμπεριλαμβανομένων των τιμών της απώλειας και των μετρήσεων τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα επικύρωσης για κάθε εποχή. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν για τη σχεδίαση καμπυλών εκμάθησης και τη διάγνωση πιθανών ζητημάτων όπως η υπερπροσαρμογή.



Εικόνα 100 – Γραφήματα Εκπαίδευσης Μοντέλου

Ο παρακάτω κώδικας είναι υπεύθυνος για τον σχηματισμό των γραφημάτων και της αξιολόγησης του μοντέλου. Σημειώνεται πως για την δημιουργία των γραφημάτων χρειάζεται η βιβλιοθήκη PyQtX (όπου X=έκδοση), την περίοδο που γράφεται η πτυχιακή είναι η PyQt6.

```
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Acurracy')
plt.xlabel('epoch')
plt.show()

evaluation = model.evaluate(images_test,labels_test,verbose=0)
print("Evaluation:")
print('Loss:',evaluation[0])
print('Accuracy:',evaluation[1])
```

Εικόνα 104 - Αξιολόγηση Μοντέλου & Εκτύπωση Αποτελεσμάτων

Η αξιολόγηση πραγματοποιείται από την μέθοδο evaluate(), η οποία έχει ως ορίσματα το υποσύνολο των εικόνων και των ετικετών τους, που δημιουργήσαμε για την δοκιμή της ακρίβειας του μοντέλου σε εικόνες που δεν έχει «δει» ποτέ. Η τελευταία παράμετρος αφορά την τύπωση λεπτομερειών, όπως ακρίβεια και απώλεια, κατά την αξιολόγηση. Επειδή η αξιολόγηση δεν γίνεται σε παρτίδες εικόνων, έτσι ώστε να χρειάζεται να ξέρουμε την ακρίβεια για κάθε παρτίδα, αλλά κατευθείαν σε όλο το υποσύνολο, θέτουμε το verbose 0 και τυπώνουμε όπως εμείς θέλουμε το αποτέλεσμα.

```
Evaluation:
Loss: 0.18007513880729675
Accuracy: 0.9497126340866089
```

Εικόνα 108 - Αποτελέσματα Αξιολόγησης

5.6 Αποθήκευση Μοντέλου

Τέλος, για την αποθήκευση του μοντέλου χρησιμοποιείται η μέθοδος save(), η οποία έχει ως παράμετρο το όνομα και τη μορφή του αρχείου όπου θα αποθηκευτεί. Η μέθοδος αυτή δημιουργεί ένα αρχείο zip με το όνομα του αρχείου και την κατάληξη .keras. Τα μοντέλα

μπορούν να αποθηκευτούν σε τρεις μορφές: τη νέα μορφή .keras, SavedModel και HDF5 (.h5). Η μορφή .keras συνιστάται για αντικείμενα Keras λόγω της απλότητας, της αποτελεσματικότητας και της αποθήκευσης βάσει ονόματος, η οποία εξασφαλίζει ακριβή φόρτωση και ευκολότερο εντοπισμό σφαλμάτων [83]. Επειδή όμως το μοντέλο θα τρέχει πάνω στην πλατφόρμα Raspberry Pi, η οποία είναι περιορισμένη σε επεξεργαστικούς πόρους, αλλά και πόρους μνήμης, πρέπει να το μετατρέψουμε σε Tensorflow Lite. Αυτό γίνεται χρησιμοποιώντας τον tf.lite.TFLiteConverter δίνοντας του ως παράμετρο το μοντέλο που

```
model.save('road_sign_recognition-test.keras')

# Convert the model.
modelConverter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = modelConverter.convert()

# Save the TF Lite model.
with open("road_sign_recognition.tflite", 'wb') as file:
  file.write(tflite_model)
```

Εικόνα 112 - Αποθήκευση Μοντέλου

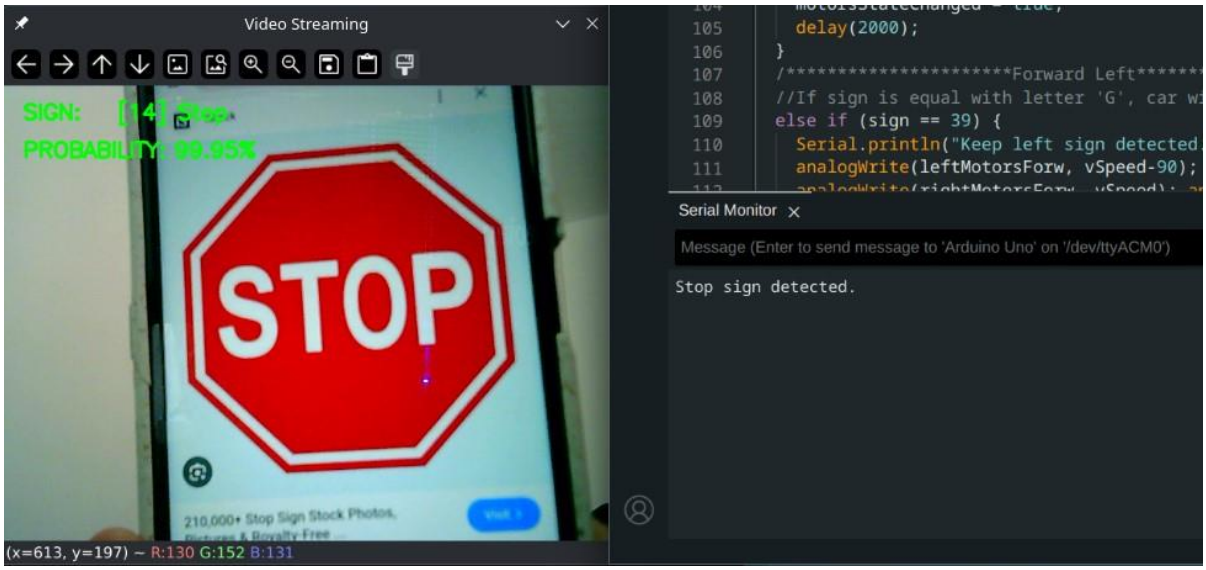
θέλουμε να μετατρέψουμε [84].

Η αρχιτεκτονική του μοντέλου έχει σχεδιαστεί για να επεξεργάζεται αποτελεσματικά εικόνες μεγέθους 32x32, σε κλίμακα του γκρι, των σημάτων κυκλοφορίας, εξάγοντας σχετικά χαρακτηριστικά μέσω μιας σειράς συνελικτικών και ομαδοποιημένων στρωμάτων. Η χρήση επιπέδων απόρριψης βοηθά στην αποφυγή της υπερ προσαρμογής, ενώ τα πλήρως συνδεδεμένα επίπεδα και το επίπεδο εξόδου softmax διασφαλίζουν την ακριβή ταξινόμηση των εικόνων εισόδου σε προκαθορισμένες κατηγορίες. Το μοντέλο αξιοποιεί τα δυνατά σημεία των συνελικτικών νευρωνικών δικτύων σε εργασίες αναγνώρισης εικόνων, παρέχοντας μια ισχυρή λύση για την αναγνώριση σημάτων κυκλοφορίας.

5.7 Εφαρμογή σε Γλώσσα Python

Αυτή η ενότητα εμβαθύνει στην υλοποίηση μιας εφαρμογής βασισμένης στη μηχανική μάθηση που έχει σχεδιαστεί για την αναγνώριση σημάτων κυκλοφορίας. Η εφαρμογή αξιοποιεί το μοντέλο που δημιουργήσαμε στο προηγούμενο κεφάλαιο, για τον ακριβή εντοπισμό διαφόρων σημάτων κυκλοφορίας από ροές βίντεο, σε πραγματικό χρόνο. Πέρα από τη λειτουργικότητα του μοντέλου, ενσωματώνει χαρακτηριστικά όπως προεπεξεργασία

εικόνας και ανάλυση βίντεο σε πραγματικό χρόνο. Το κείμενο που ακολουθεί παρέχει μια λεπτομερή εξέταση της δομής του κώδικα, τονίζοντας την ενσωμάτωση μοντέλων μηχανικής



Εικόνα 116 - Εφαρμογή

μάθησης και βοηθητικών στοιχείων που είναι κρίσιμα για τη λειτουργικότητα της εφαρμογής. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι η numpy, openCV, tflite-runtime και pyserial.

Στην εικόνα που ακολουθεί, φαίνεται ο κώδικας που αρχικοποιεί τις απαραίτητες μεταβλητές και παραμέτρους.

- **previouslySentSign:** Η μεταβλητή αυτή χρησιμοποιείται ως προσωρινή μνήμη, έτσι ώστε όταν το μοντέλο αναγνωρίσει μια πινακίδα να την στείλει μια φορά και όχι συνεχόμενα (μέχρι να φύγει από την κάμερα) στον arduino.
- **cv2.VideoCapture():** Δημιουργεί ένα αντικείμενο VideoCapture. Το όρισμά της μεθόδου μπορεί να είναι είτε ο αριθμός που αντιπροσωπεύει την συσκευή, είτε το όνομα ενός αρχείου βίντεο. Συνήθως ο αριθμός μηδέν (0) αντιπροσωπεύει την προκαθορισμένη συσκευή, αν υπάρχουν περισσότερες από μια κάμερες στο σύστημα, αυξάνοντας τον αριθμό, μπορείτε να επιλέξετε κάποια από τις υπόλοιπες. Στο τέλος, μην ξεχάσετε να απελευθερώσετε την κάμερα με την μέθοδο .release() [85].
- **tf.lite.Interpreter():** Αρχικοποιούμε τον interpreter με το μοντέλο.

- **.allocate_tensors():** Αυτή η μέθοδος δεσμεύει την απαραίτητη μνήμη για όλους τους τανυστές που θα χρειαστεί το μοντέλο κατά την εξαγωγή συμπερασμάτων, συμπεριλαμβανομένων των τανυστών εισόδου, εξόδου και ενδιάμεσων τανυστών [86].
- **.get_input_details():** Δημιουργεί ένα αντικείμενο τανυστή, με τις πληροφορίες του τανυστή εισόδου του μοντέλου [86].
- **.get_output_details():** Δημιουργεί ένα αντικείμενο τανυστή, με τις πληροφορίες του τανυστή εξόδου του μοντέλου [86].

```
import numpy as np
import cv2
import serial
import tensorflow.lite as tflite

previouslySentSign = -1
# INITIALIZE COMMUNICATION WITH ARDUINO
arduino = serial.Serial(port='/dev/ttyACM0', baudrate=9600)

# SETUP THE VIDEO CAMERA
camera = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX

# IMPORT THE TF-LITE TRAINNED MODEL
interpreter = tflite.Interpreter(model_path="road_sign_recognition.tflite")
interpreter.allocate_tensors()
num_tensors = len(interpreter.get_tensor_details())
print(f"Total number of tensors allocated: {num_tensors}")

#GET INPUT - OUTPUT DETAILS
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Εικόνα 117 - Αρχικοποίηση Παραμέτρων

5.7.1 Προεπεξεργασία Εικόνας

Σε αυτό το μπλοκ κώδικα διαβάζουμε το βίντεο από το αντικείμενο της κάμερας και επεξεργαζόμαστε το κάθε frame. Αρχικά αλλάζουμε το μέγεθος του σε 32x32, καθώς έτσι μειώνεται ο υπολογισμός της προεπεξεργασίας αφού εφαρμόζεται σε μικρότερα δεδομένα και το μοντέλο μας δέχεται ως είσοδο τέτοιου μεγέθους εικόνες. Στην συνέχεια εφαρμόζουμε ακριβώς την ίδια επεξεργασία που εφαρμόσαμε και στις εικόνες για την εκπαίδευση του μοντέλου. Το μετατρέπουμε δηλαδή σε κλίμακα του γκρι, εξισώνουμε το ιστόγραμμα και το κανονικοποιούμε έτσι ώστε η φωτεινότητα των εικονοστοιχείων να απεικονίζεται στην κλίμακα [0,1]. Τελευταίο βήμα, για να είναι έτοιμο το frame να δοθεί στο μοντέλο ως είσοδος,

το μετασχηματίζουμε σε μορφή του τανυστή εισόδου. Δηλαδή, σε τανυστή τεσσάρων διαστάσεων όπου αυτές είναι, μέγεθος παρτίδας, ύψος, πλάτος και κανάλια. Επίσης, με την μέθοδο .astype() σιγουρεύουμε πως τα δεδομένα του πίνακα είναι σε δεκαδική μορφή, κάτι που απαιτείται από τα νευρωνικά δίκτυα για τον ακριβή υπολογισμό.

```
while camera.isOpened():
    # READ IMAGE
    read, frame = camera.read()
    if not read:
        break
    # PROCESS IMAGE
    resizedInputFrame = cv2.resize(frame, (32, 32))
    preprocessedImage = preprocessing(resizedInputFrame)
    cv2.imshow('Preprocessed Image', preprocessedImage)
    image = preprocessedImage.reshape(1, 32, 32, 1).astype(np.float32)
    cv2.putText(frame, 'SIGN: ' , (20, 35), font, 0.75, (0, 255, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, 'PROBABILITY: ', (20, 75), font, 0.75, (0, 255, 0), 2, cv2.LINE_AA)
```

Εικόνα 121 - Προεπεξεργασία Εικόνας

Η μέθοδος .putText() χρησιμοποιείται για να σχεδιάσει μια συμβολοσειρά κειμένου σε οποιαδήποτε εικόνα [87]. Στην προκειμένη περίπτωση για να σχεδιάζουν συμβολοσειρές στο παράθυρο του βίντεο που δημιουργείται από την cv2.imshow().

5.7.2 Πρόβλεψη Πινακίδας

Αφού λοιπόν έχει γίνει η προεπεξεργασία των εικόνων και τα δεδομένα είναι έτοιμα να δοθούν στο δίκτυο, καλούμε την .set_tensor() με παραμέτρους τον τανυστή εισόδου και την εικόνα που θέλουμε να επεξεργαστεί το δίκτυο. Στην συνέχεια, με την .invoke() τρέχει το δίκτυο και πραγματοποιείται ο υπολογισμός και τέλος στην μεταβλητή predictions έχουμε ένα δισδιάστατο πίνακα, όπου η μια διάσταση του αφορά τις εικόνες (1) που εισήχθησαν στο δίκτυο για ανάλυση και η άλλη αφορά τις πιθανότητες για κάθε μια από τις κλάσεις στην

```
Predictions:
[[1.37149159e-09 7.78090383e-04 9.01911844e-05 5.30065387e-04
  4.25672943e-06 3.87500004e-05 2.34767385e-02 1.55279758e-06
  2.21797194e-08 1.28222382e-04 1.13956303e-05 5.53009340e-06
  3.10539734e-03 8.78565758e-03 1.58326458e-02 6.37654506e-04
  6.09678921e-07 1.58772615e-04 6.03715132e-04 2.41750453e-10
  1.95147051e-03 4.68900261e-08 7.57904218e-07 6.76571426e-06
  7.52372120e-09 4.82869582e-05 2.69116863e-04 1.06603068e-07
  4.16306102e-05 3.65840606e-07 7.03779961e-06 6.07429683e-06
  5.79376174e-05 1.28360167e-01 6.83728467e-06 3.84533661e-03
```

Εικόνα 122 - Πίνακας Πιθανοτήτων Κάθε Κλάσης

οποία μπορεί να ανήκει η εικόνα. Ουσιαστικά είναι ένας πίνακας που περιέχει πίνακες για κάθε εικόνα και αυτοί οι πίνακες περιέχουν τις πιθανότητες που δίνει το μοντέλο για κάθε κλάση.

Οι τιμές που φαίνονται στον πίνακα είναι πολύ μικρότερες από αυτό που φαίνονται για κάποιον που δεν γνωρίζει. Ας πάρουμε την πρώτη τιμή που είναι η 1.37149159e-09, το e-09 σημαίνει "δέκα φορές στη δύναμη του αρνητικού εννέα" ($\times 10^{-9}$), που δείχνει ότι η υποδιαστολή μετακινείται εννέα θέσεις προς τα αριστερά και ο αριθμός 1.37149159 είναι ο αριθμός βάσης. Αυτό σημαίνει ότι η τιμή αυτή 1.37149159e-09 στο δεκαδικό σύστημα είναι η $1.37149159 \times 10^{-9} = 0.00000000137149159$. Άρα, η μεγαλύτερη τιμή του πίνακα θα είναι εκείνη με τον μικρότερο (αρνητικό) εκθέτη και την μεγαλύτερη βάση, σε αυτή την περίπτωση είναι το στοιχείο τριανταέξι (36) 6.82787240e-01. Αυτό σημαίνει ότι η κλάση 36 έχει την μεγαλύτερη πιθανότητα να είναι η κλάση της πινακίδας που προσπαθεί να αναγνωρίσει το μοντέλο. Με βάση τα παραπάνω, η argmax() μας δίνει την θέση (36) της μεγαλύτερης τιμής στον πίνακα των πιθανοτήτων και η amax() την τιμή (0.68278724).

- **.set_tensor():** Ορίζει την τιμή του τανυστή εισόδου, αντιγράφοντας τα δεδομένα που υπάρχουν στην μεταβλητή (image) [86].
- **.invoke():** Εκτελεί το μοντέλο [86].
- **.get_tensor():** Αντιγράφει την τιμή του τανυστή εξόδου στην μεταβλητή predictions [86].
- **round():** Χρησιμοποιείται για την αναγωγή της τιμής της πιθανότητας σε ποσοστό.

```
# PREDICT IMAGE
interpreter.set_tensor(input_details[0]['index'], image)
interpreter.invoke()
predictions = interpreter.get_tensor(output_details[0]['index'])
signIndex = np.argmax(predictions,axis=1)
probabilityValue = np.amax(predictions)
probabilityPrecentage = round(probabilityValue*100,2)
if probabilityPrecentage >= 60:
    cv2.putText(frame,str(signIndex)+' '+str(getSignName(signIndex.item(),
                probabilityPrecentage)), (120, 35), font, 0.75,
                (0, 255, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, str(probabilityPrecentage)+'%', (180, 75), font,
                0.75, (0, 255, 0), 2, cv2.LINE_AA)
```

Εικόνα 126 - Πρόβλεψη Πινακίδας

Τελειώνοντας, καλείται η συνάρτηση getSignName() με παραμέτρους τον αριθμό που μας έδωσε argmax(), τον αριθμό της κλάσης δηλαδή και το ποσοστό της πιθανότητας να είναι αυτή η σωστή κλάση. Σε αυτό το σημείο να αναφέρουμε, ότι υπάρχει μια δομή map με όνομα «signs» όπου περιέχει αντιστοιχίες ακέραιων αριθμών (0-42) και ονομάτων των πινακίδων - κλάσεων. Οι ακέραιοι αυτοί αριθμοί, αντιστοιχούν επίσης και στους αριθμούς των κλάσεων που εκπαιδεύσαμε το μοντέλο.

```
signs = {
        0: 'No vehicles',
        1: 'Speed Limit 80 km/h',
        2: 'End of Speed Limit 80 km/h',
                    .
                    .
                    .
        40: 'Roundabout mandatory',
        41: 'Double curve',
        42: 'End of no passing by vehicles over 3.5 metric tons'
    }
```

Εικόνα 130 - Δομή signs

Η συνάρτηση αυτή, ψάχνει στην δομή τον αριθμό θέσης της πινακίδας και επιστρέφει το όνομα της, ώστε να εμφανιστεί στο παράθυρο του βίντεο. Αν το ποσοστό της πιθανότητας είναι μεγαλύτερο ή ίσο με 90% (θέλουμε να είμαστε πολύ σίγουροι) και η πινακίδα που αναγνώρισε δεν είναι ίδια με την πινακίδα που έστειλε τελευταία φορά στον arduino, τότε σημαίνει πως συνάντισε νέα πινακίδα, οπότε την στέλνει στον μικροελεγκτή και ανανεώνει την μεταβλητή «μνήμης» με τον ακέραιο αριθμό της νέας πινακίδας.

```
def getSignName(signNo,probabilityPrecentage):
    global previouslySentSign
    if probabilityPrecentage >= 90 and signNo != previouslySentSign:
        send_sign(signNo.to_bytes(1,'big'))
        previouslySentSign = signNo
    return signs[signNo]
```

Εικόνα 134 - Συνάρτηση getSignName

Η συνάρτηση send_sign περιλαμβάνει μόνο την γραμμή arduino.write(sign), όπου sign είναι ο αριθμός της πινακίδας σε μορφή byte.

- **.write(sign):** Γράφει τα bytes της μεταβλητής sign στη θύρα. Τα δεδομένα πρέπει να είναι τύπου byte ή bytearray ή memoryview. Οι συμβολοσειρές Unicode πρέπει να είναι κωδικοποιημένες (π.χ. 'hello'.encode('utf-8')) [82].

ΣΥΜΠΕΡΑΣΜΑΤΑ & ΕΠΟΜΕΝΑ ΒΗΜΑΤΑ

6.1 Συμπεράσματα

Η εικονικοποίηση βελτιώνει τη διαχείριση πόρων: Η εικονικοποίηση, συμπεριλαμβανομένης της χρήσης hypervisors και containers, επιτρέπει την πιο αποτελεσματική χρήση των υπολογιστικών πόρων δημιουργώντας εικονικές παρουσίες υλικού και λογισμικού. Αυτό οδηγεί σε βελτιωμένη επεκτασιμότητα, ευελιξία και εξοικονόμηση κόστους, καθώς οι πόροι μπορούν να κατανεμηθούν δυναμικά με βάση τη ζήτηση.

Το νέφος και η υπολογιστική στο άκρο αλληλοσυμπληρώνονται: Ενώ το νέφος προσφέρει κεντρική επεξεργασία και αποθήκευση δεδομένων, η υπολογιστική στο άκρο φέρνει αυτές τις δυνατότητες πιο κοντά στην πηγή δεδομένων, μειώνοντας την καθυστέρηση και βελτιώνοντας την επεξεργασία σε πραγματικό χρόνο. Η συνέργεια μεταξύ νέφους και άκρου μπορεί να οδηγήσει σε πιο αποτελεσματικό χειρισμό δεδομένων και καλύτερη απόδοση σε εφαρμογές που απαιτούν άμεση επεξεργασία δεδομένων.

Τα ενσωματωμένα συστήματα είναι ζωτικής σημασίας για τις εφαρμογές άκρου: Τα ενσωματωμένα συστήματα διαδραματίζουν βασικό ρόλο στην υπολογιστική άκρου παρέχοντας την απαραίτητη ενοποίηση υλικού και λογισμικού για την υποστήριξη της επεξεργασίας δεδομένων σε πραγματικό χρόνο. Αυτά τα συστήματα συχνά βελτιστοποιούνται για συγκεκριμένες εργασίες, οδηγώντας σε βελτιωμένη απόδοση και αξιοπιστία σε διάφορες εφαρμογές.

Η μηχανική μάθηση βελτιώνει τα ενσωματωμένα συστήματα: Η ενοποίηση της μηχανικής μάθησης με τα ενσωματωμένα συστήματα, επιτρέπει προηγμένες δυνατότητες ανάλυσης δεδομένων και λήψης αποφάσεων στο άκρο. Αυτό μπορεί να οδηγήσει σε πιο έξυπνα και αυτόνομα συστήματα ικανά να χειρίζονται πολύπλοκες εργασίες χωρίς να βασίζονται σε κεντρικούς πόρους του νέφους ή στην ανθρώπινη παρέμβαση.

Ανάπτυξη και εφαρμογή προηγμένων συστημάτων: Η διαδικασία ανάπτυξης προηγμένων συστημάτων περιλαμβάνει διάφορα στάδια, από την επιλογή υλικού (π.χ. Arduino, Raspberry Pi) έως τη δημιουργία, την εκπαίδευση και την αξιολόγηση μοντέλων

μηχανικής μάθησης. Αυτή η ολοκληρωμένη προσέγγιση διασφαλίζει ότι το τελικό σύστημα είναι ισχυρό, αποτελεσματικό και ικανό να καλύψει τις επιθυμητές απαιτήσεις εφαρμογής.

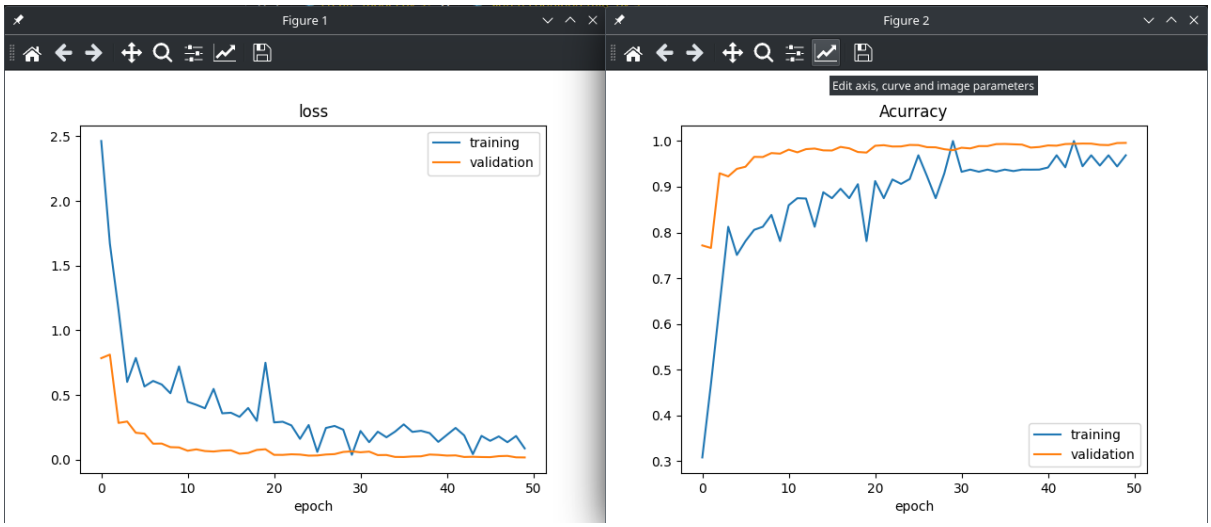
Ψευδώς Θετικά Σφάλματα: Παρατηρήθηκε πως αν στο ίδιο μοντέλο χρησιμοποιήσουμε καλύτερης ποιότητας κάμερα, αυτό συμπεριφέρεται καλύτερα όσον αφορά τα ψευδώς θετικά σφάλματα. Οι εικόνες χαμηλής ποιότητας μπορούν να επηρεάσουν σημαντικά την απόδοση ενός Συνελικτικού Νευρωνικού Δικτύου που έχει ως αποστολή την αναγνώριση εικόνων πινακίδων κυκλοφορίας, οδηγώντας συχνά σε αυξημένα ψευδώς θετικά σφάλματα. Όταν η ποιότητα της εικόνας είναι κακή, παράγοντες όπως η θαμπάδα, η χαμηλή ανάλυση ή ο θόρυβος μπορεί να αποκρύψουν κρίσιμα χαρακτηριστικά των πινακίδων κυκλοφορίας, καθιστώντας δύσκολο για το μοντέλο να τα αναγνωρίσει και να τα ταξινομήσει σωστά. Κατά συνέπεια, το δίκτυο μπορεί να παρερμηνεύσει ή να μην κάνει διάκριση μεταξύ διαφορετικών πινακίδων, ή άλλων αντικειμένων, με αποτέλεσμα λανθασμένες προβλέψεις όπου το μοντέλο αναγνωρίζει αντικείμενα ως σήματα κυκλοφορίας ενώ δεν είναι. Αυτή η υποβάθμιση στην ακρίβεια υπονομεύει την αξιοπιστία του μοντέλου, ιδιαίτερα σε εφαρμογές κρίσιμες για την ασφάλεια, όπως η αυτόνομη οδήγηση.

Tensorflow & Raspberry Pi: Το μοντέλο παρουσιάζει προβλήματα καθυστέρησης, στο παράθυρο του βίντεο, όταν χρησιμοποιείται το TensorFlow, λόγω των υψηλών υπολογιστικών απαιτήσεων και των περιορισμένων πόρων της συσκευής. Τα μοντέλα πλήρους κλίμακας του TensorFlow απαιτούν σημαντική επεξεργαστική ισχύ και μνήμη, η οποία μπορεί να υπερκαλύψει τις περιορισμένες δυνατότητες του Raspberry Pi, με αποτέλεσμα αργούς χρόνους εξαγωγής συμπερασμάτων και αναποτελεσματική απόδοση. Αντίθετα, χρησιμοποιώντας το TensorFlow Lite το μοντέλο λειτουργεί εξαιρετικά σε θέματα καθυστέρησης, καθώς η βιβλιοθήκη αυτή βελτιστοποιεί τα μοντέλα, ειδικά για συσκευές άκρου, εφαρμόζοντας τεχνικές όπως η κβαντοποίηση μοντέλων, η οποία μειώνει το μέγεθος και την υπολογιστική πολυπλοκότητα του δικτύου.

Κακή Φωτεινότητα Χώρου: Το μοντέλο παρουσιάζει μειωμένη απόδοση όταν απουσιάζει το φυσικό ηλιακό φως, αυτό λογικά συμβαίνει λόγω των συνθηκών φωτισμού των δεδομένων εκπαίδευσης. Ελλείψει φυσικού ηλιακού φωτός, το μοντέλο αντιμετωπίζει προκλήσεις όπως ανεπαρκής αντίθεση, αυξημένος θόρυβος, οδηγώντας σε δυσκολίες στον ακριβή εντοπισμό και ταξινόμηση των σημάτων κυκλοφορίας. Αυτό είναι ιδιαίτερα

προβληματικό σε πραγματικές εφαρμογές όπου οι συνθήκες φωτισμού είναι δυναμικές και απρόβλεπτες. Κατά συνέπεια, η ακρίβεια και η αξιοπιστία του μοντέλου στην αναγνώριση σημάτων κυκλοφορίας σε συνθήκες χαμηλού φωτισμού ή νύχτας μπορεί να διακυβευτεί σημαντικά, υπογραμμίζοντας την ανάγκη για ισχυρή αύξηση δεδομένων και προσαρμοστικές τεχνικές για τη βελτίωση της απόδοσης σε διάφορα σενάρια φωτισμού.

Υπερπροσαρμογή (Overfitting): Παρατηρήθηκε πως περίπου μετά τις είκοσι (20) εποχές το μοντέλο γίνεται πολύ συγκεκριμένο για το dataset που χρησιμοποιείται, καθώς ο δείκτης της εκπαίδευσης (training) συνεχίζει να αυξάνεται και φτάνει σε υψηλή τιμή, ενώ ο δείκτης της επικύρωσης (validation) αρχίζει να αυξάνεται αλλά στη συνέχεια μένει στάσιμος ή μειώνεται ενώ η ακρίβεια εκπαίδευσης βελτιώνεται συνεχώς.



Εικόνα 138 – Overfitting

6.2 Επόμενα Βήματα

Χρήση της Μεθόδου .tensor(): Η χρήση της interpreter.tensor() αντί της interpreter.set_tensor() μπορεί να βελτιώσει σημαντικά την αποτελεσματικότητα και την ευελιξία της ανάπτυξης μοντέλων μηχανικής μάθησης, ιδιαίτερα εκείνων που έχουν σχεδιαστεί για εφαρμογές σε πραγματικό χρόνο, όπως η αναγνώριση σημάτων κυκλοφορίας. Η μέθοδος .tensor() παρέχει άμεση πρόσβαση στους τανυστές εντός του διερμηνέα (interpreter) TensorFlow Lite, επιτρέποντας τον επιτόπιο χειρισμό των δεδομένων. Αυτή η προσέγγιση μειώνει την επιβάρυνση που σχετίζεται με την αντιγραφή δεδομένων, όπως απαιτείται από τη set_tensor, οδηγώντας έτσι σε ταχύτερους χρόνους επεξεργασίας δεδομένων και

συμπερασμάτων. Με τη βελτιστοποίηση της χρήσης πόρων, ειδικά σε συσκευές άκρου με περιορισμένη υπολογιστική ισχύ, η .tensor() επιτρέπει την πιο αποκριτική και αποτελεσματική εκτέλεση του μοντέλου. Αυτό είναι ιδιαίτερα ωφέλιμο σε σενάρια που απαιτούν άμεση λήψη αποφάσεων, όπως αυτόνομη οδήγηση, όπου η γρήγορη και ακριβής εξαγωγή συμπερασμάτων είναι κρίσιμη.

Χρήση της Τεχνικής Transfer Learning: Το μοντέλο μπορεί να επωφεληθεί από την τεχνική της μεταφοράς μάθησης (transfer learning), αξιοποιώντας τα προεκπαιδευμένα βάρη και τις μαθημένες λειτουργίες από ένα μεγαλύτερο, πιο ολοκληρωμένο μοντέλο που έχει εκπαιδευτεί σε εκτεταμένα και διαφορετικά σύνολα δεδομένων. Αυτή η προσέγγιση επιτρέπει στο μοντέλο να ξεκινήσει με μια ισχυρή βάση γνώσεων σχετικά με κοινά μοτίβα και χαρακτηριστικά σε εικόνες, μειώνοντας τον όγκο των δεδομένων και των υπολογιστικών πόρων που απαιτούνται για την εκπαίδευση του. Προσαρμόζοντας το προεκπαιδευμένο μοντέλο στο σύνολο δεδομένων από πινακίδες, θα βοηθήσει το νέο μοντέλο να προσαρμοστεί γρήγορα στην αναγνώριση πινακίδων κυκλοφορίας, οδηγώντας σε βελτιωμένη ακρίβεια και γενίκευση με λιγότερη εκπαίδευση και λιγότερα δεδομένα. Η τεχνική αυτή επιταχύνει έτσι τη διαδικασία εκπαίδευσης, ενισχύει την απόδοση και μετριάζει τους κινδύνους υπερβολικής προσαρμογής.

Επαύξηση Φωτεινότητας (Brightness Augmentation): Η επαύξηση φωτεινότητας στο μοντέλο μπορεί να βελτιώσει σημαντικά τις δυνατότητές γενίκευσης του. Η μέθοδος αυτή περιλαμβάνει τη μεταβολή της έντασης της φωτεινότητας της εικόνας κατά τη διάρκεια της εκπαίδευσης, η οποία βοηθά το μοντέλο να γίνει αμετάβλητο στις αλλαγές των συνθηκών φωτισμού. Αυτό είναι ιδιαίτερα ωφέλιμο σε σενάρια όπου μπορούν να ληφθούν εικόνες κάτω από διαφορετικές συνθήκες φωτισμού, όπως ποικίλο ηλιακό φως ή τεχνητός φωτισμός. Εκθέτοντας το μοντέλο σε ένα ευρύτερο φάσμα επιπέδων φωτεινότητας, μαθαίνει να αναγνωρίζει και να ταξινομεί αντικείμενα με μεγαλύτερη ακρίβεια, ανεξάρτητα από τις διακυμάνσεις του φωτισμού.

Χρήση Κάμερας με Καλύτερη Ανάλυση: Η αναβάθμιση σε κάμερα υψηλότερης ποιότητας για τη λήψη βίντεο μπορεί να βελτιώσει σημαντικά την απόδοση του μοντέλου, παρέχοντας πλουσιότερα και πιο λεπτομερή δεδομένα εισόδου. Οι κάμερες υψηλής ανάλυσης παρέχουν καθαρότερες εικόνες με περισσότερες λεπτομέρειες, γεγονός που επιτρέπει στο

μοντέλο να ανιχνεύει και να αναγνωρίζει αντικείμενα με μεγαλύτερη ακρίβεια και αξιοπιστία. Η βελτιωμένη ποιότητα της εικόνας μειώνει την ποσότητα του θορύβου και της παραμόρφωσης, βελτιώνοντας έτσι την εξαγωγή χαρακτηριστικών και ελαχιστοποιώντας τα σφάλματα στις προβλέψεις του μοντέλου. Αυτό είναι ιδιαίτερα σημαντικό για εργασίες όπως η αναγνώριση σημάτων οδικής κυκλοφορίας, όπου οι λεπτές οπτικές ενδείξεις μπορεί να είναι καθοριστικής σημασίας.

Χρήση αισθητήρων απόστασης: Οι αισθητήρες αυτοί είναι ζωτικής σημασίας για τα αυτόνομα οχήματα, καθώς επιτρέπουν την ανίχνευση κοντινών αντικειμένων σε πραγματικό χρόνο, αποτρέποντας τις συγκρούσεις και ενισχύοντας την επίγνωση της κατάστασης. Ο συνδυασμός της τεχνητής νοημοσύνης με τους αισθητήρες απόστασης επιτρέπει στο όχημα να ανταποκρίνεται με ακρίβεια τόσο σε άμεσους κινδύνους όσο και σε ρυθμιστικά σήματα, βελτιώνοντας σημαντικά τη συνολική οδηγική ασφάλεια και απόδοση.

Συμπερασματικά, η παρούσα διατριβή καταδεικνύει με επιτυχία την ενσωμάτωση της τεχνητής νοημοσύνης, ιδιαίτερα της μηχανικής μάθησης, με τον υπολογισμό στο άκρο για την επεξεργασία δεδομένων σε πραγματικό χρόνο στα αυτόνομα οχήματα. Κάθε κεφάλαιο υπογράμμισε τον σημαντικό ρόλο που διαδραματίζουν αυτές οι τεχνολογίες στη σύγχρονη επεξεργασία και ανάλυση δεδομένων. Μέσω της πρακτικής επίδειξης ενός αυτόνομου οχήματος που αναγνωρίζει σήματα κυκλοφορίας, χρησιμοποιώντας ένα συνελικτικό νευρωνικό δίκτυο, η έρευνα δείχνει πώς η υπολογιστική στο άκρο επιτρέπει την επεξεργασία δεδομένων σε πραγματικό χρόνο. Μελλοντικές βελτιώσεις, θα μπορούσαν να εξερευνήσουν πιο εξελιγμένα μοντέλα τεχνητής νοημοσύνης και ενσωματώσεις αισθητήρων για την περαιτέρω βελτιστοποίηση των αυτόνομων οχημάτων. Αυτές οι βελτιώσεις θα αυξήσουν την αποτελεσματικότητα, την ακρίβεια και την αξιοπιστία της λήψης αποφάσεων σε πραγματικό χρόνο στην αυτόνομη οδήγηση και όχι μόνο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Campbell S, Jeronimo M. An introduction to virtualization. Published in "Applied Virtualization", Intel. 2006:1-5.

[2] Uhlig R, Neiger G, Rodgers D, Santoni AL, Martins FC, Anderson AV, Bennett SM, Kagi A, Leung FH, Smith L. Intel virtualization technology. Computer. 2005 May 16;38(5):48-56.

[3] VMware. Understanding Full Virtualization, Paravirtualization and Hardware Assist. 2008.

[4] Rodríguez-Haro F, Freitag F, Navarro L, Hernánchez-sánchez E, Farías-Mendoza N, Guerrero-Ibáñez JA, González-Potes A. A summary of virtualization techniques. Procedia Technology. 2012 Jan 1;3:267-72.

[5] Chiueh SN, Brook S. A survey on virtualization technologies. Rpe Report. 2005 Jun;142.

[6] What is virtualization? [Internet]. Redhat. [cited 2024 May 15]. Available from: https://www.redhat.com/en/topics/virtualization/what-is-virtualization

[7] Lee I. AMD-V: Unleashing the power of virtualization - everything you need to know - vtechinsider [Internet]. vtechinsider. 2023 [cited 2024 May 15]. Available from: https://vtechinsider.com/what-is-amd-v/

[8] 1.1.2. Hypervisor [Internet]. Oracle. [cited 2024 May 15]. Available from: https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1011.html

[9] What is a hypervisor? [Internet]. Redhat. [cited 2024 May 16]. Available from: https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor

[10] Virtual machine memory [Internet]. VMware. [cited 2024 May 16]. Available from: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.resmgmt.doc/GUID-C25A8823-F595-4322-BD0D-4FD5B081F877.html

[11] Popek GJ, Goldberg RP. Formal requirements for virtualizable third generation architectures. Communications of the ACM. 1974 Jul 1;17(7):412-21.

[12] Martin R. Hypervisor Part 1- What is a Hypervisor and How Does it Work? [Internet]. Qnx. [cited 2024 May 18]. Available from: https://blackberry.qnx.com/content/dam/qnx/whitepapers/2017/what-is-a-hypervisor-and-how-does-it-work-pt1.pdf

[13] Hypervisor [Internet]. GeeksforGeeks. 2018 [cited 2024 May 19]. Available from: https://www.geeksforgeeks.org/hypervisor/

[14] Wikipedia contributors. Robert P. Goldberg [Internet]. Wikipedia, The Free Encyclopedia. 2024. Available from: https://en.wikipedia.org/w/index.php?title=Robert_P._Goldberg&oldid=1238682443

[15] Goldberg RP. ARCHITECTURAL SYSTEMS PRINCIPLES FOR VIRTUAL COMPUTERS [Internet]. Dtic.mil. [cited 2024 May 19]. Available from: https://apps.dtic.mil/sti/pdfs/AD0772809.pdf

[16] What are hypervisors? [Internet]. IBM. 2024 [cited 2024 May 19]. Available from: https://www.ibm.com/cloud/learn/hypervisors

[17] Tamane S. A review on virtualization: A cloud technology. International Journal on Recent and Innovation Trends in Computing and Communication. 2015;3(7):4582-5.

[18] Masood A, Sharif M, Yasmin M, Raza M. Virtualization tools and techniques: Survey. Nepal Journal of Science and Technology. 2014;15(2):141-50.

[19] Gouda K, Patro A, Dwivedi D, Bhat N. Virtualization approaches in cloud computing. International Journal of Computer Trends and Technology (IJCTT). 2014 Jun;12(4):161-6.

[20] Chen W, Lu H, Shen L, Wang Z, Xiao N, Chen D. A novel hardware assisted full virtualization technique. In2008 The 9th International Conference for Young Computer Scientists 2008 Nov 18 (pp. 1292-1297). IEEE.

[21] What is a virtual machine (VM)? [Internet]. Microsoft. [cited 2024 May 22]. Available from: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine/

[22] What is a virtual machine (VM)? [Internet]. IBM. 2024 [cited 2024 May 22]. Available from: https://www.ibm.com/topics/virtual-machines

[23] Smith JE, Nair R. The architecture of virtual machines. Computer. 2005 May 16;38(5):32-8.

[24] What is a Virtual Machine (VM) and How Do Virtual Machines Work? Citrix.com. Available at https://www.citrix.com/solutions/vdi-and-daas/what-is-a-virtual-machine.html

[25] What is a virtual machine? [Internet]. Vmware. [cited 2024 May 26]. Available from: https://www.vmware.com/topics/glossary/content/virtual-machine.html?resource=cat-786093672

[26] Virtualization for virtual machines [Internet]. Microsoft. [cited 2024 May 26]. Available from: https://learn.microsoft.com/en-us/training/modules/cmu-virtualization/1-virtual-machines

[27] What is a container? [Internet]. Docker. [cited 2024 May 30]. Available from: https://www.docker.com/resources/what-container/

[28] da Silva VG, Kirikova M, Alksnis G. Containers for virtualization: An overview. Applied Computer Systems. 2018 May 1;23(1):21-7.

[29] Eder M. Hypervisor-vs. container-based virtualization. Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM). 2016 Jul;1.

[30] Morabito R, Kjällman J, Komu M. Hypervisors vs. lightweight virtualization: a performance comparison. In2015 IEEE International Conference on cloud engineering 2015 Mar 9 (pp. 386-393). IEEE.

[31] Li Z, Kihl M, Lu Q, Andersson JA. Performance overhead comparison between hypervisor and container based virtualization. In2017 IEEE 31st International Conference on advanced information networking and applications (AINA) 2017 Mar 27 (pp. 955-962). IEEE.

[32] What is containerization? [Internet]. IBM. 2024 [cited 2024 Jun 5]. Available from: https://www.ibm.com/topics/containerization

[33] About the Open Container Initiative [Internet]. Opencontainers. [cited 2024 Jun 7]. Available from: https://opencontainers.org/about/overview/

[34] Aleksic M. What are Linux containers? [Internet]. Ubuntu. 2022 [cited 2024 Jun 7]. Available from: https://ubuntu.com/blog/what-are-linux-containers

[35] Containers and virtual machines [Internet]. Linuxcontainers. [cited 2024 Jun 10]. Available from: https://linuxcontainers.org/lxd/introduction/

[36] What are containers? [Internet]. IBM. 2024 [cited 2024 Jun 11]. Available from: https://www.ibm.com/topics/containers

[37] 5 Benefits of Virtualization. IBM. 2021. Available from: https://www.ibm.com/cloud/blog/5-benefits-of-virtualization

[38] What is a hypervisor? [Internet]. VMware. [cited 2024 Jun 14]. Available from: https://www.vmware.com/topics/glossary/content/hypervisor.html?resource=cat-1299087558

[39] Bisong E. What is cloud computing? Cloud Technologies [Internet]. 2019 [cited 2024 Jun 14]; Available from: https://cloud.google.com/learn/what-is-cloud-computing?cloudshell=false

[40] Cloud H. The nist definition of cloud computing. National institute of science and technology, special publication. 2011;800(2011):145.

[41] IBM Cloud Team. A Brief History of Cloud Computing [Internet]. IBM. 2017 [cited 2024 Jun 17]. Available from: https://www.ibm.com/cloud/blog/cloud-computing-history

[42] What is cloud computing? [Internet]. IBM. 2024 [cited 2024 Jun 18]. Available from: https://www.ibm.com/topics/cloud-computing

[43] What is vendor lock-in? | Vendor lock-in and cloud computing. Cloudflare. 2024 [cited 2024 Jun 18]. Available from: https://www.cloudflare.com/learning/cloud/what-is-vendor-lock-in/

[44] What is edge computing? [Internet]. Ibm.com. 2024 [cited 2024 Jun 20]. Available from: https://www.ibm.com/cloud/what-is-edge-computing

[45] Satyanarayanan M. The emergence of edge computing. Computer. 2017 Jan 5;50(1):30-9.

[46] Escamilla-Ambrosio PJ, Rodríguez-Mota A, Aguirre-Anaya E, Acosta-Bermejo R, Salinas-Rosales M. Distributing computing in the internet of things: cloud, fog and edge computing overview. InNEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico 2018 (pp. 87-115). Springer International Publishing.

[47] Cao K, Liu Y, Meng G, Sun Q. An overview on edge computing research. IEEE access. 2020 May 1;8:85714-28.

[48] Hamouda M. The edge - old, new, borrowed and blue [Internet]. Dell. 2021 [cited 2024 Jun 26]. Available from: https://www.dell.com/en-us/blog/the-edge-old-new-borrowed-and-blue/

[49] Velrajan S. Tech [Internet]. Tech | 5G, SDN/NFV & Edge Compute. [cited 2024 Jun 26]. Available from: https://tech.ginkos.in/2019/06/far-edge-vs-near-edge-in-edge-computing.html

[50] Saunders A. Different types of edge computing [Internet]. NVIDIA Technical Blog. 2022 [cited 2024 Jun 28]. Available from: https://developer.nvidia.com/blog/different-types-of-edge-computing/

[51] Richardson D. Edge computing benefits and use cases [Internet]. Redhat. 2021 [cited 2024 Jun 28]. Available from: https://www.redhat.com/en/blog/edge-computing-benefits-and-use-cases

[52] Liu S. Edge Computing for Autonomous Vehicles.

[53] Barkalov A, Titarenko L, Mazurkiewicz M. Introduction into embedded systems. Foundations of Embedded Systems [Internet]. 2019 [cited 2024 Jun 30]; Available from: https://www.geeksforgeeks.org/introduction-of-embedded-systems-set-1/

[54] Thomas M. What is an embedded system? [Internet]. Coderus. 2023 [cited 2024 Jun 30]. Available from: https://www.coderus.com/guide-to-embedded-systems/

[55] Wikipedia contributors. Embedded system [Internet]. Wikipedia, The Free Encyclopedia. 2024 [cited 2024 Jul 4]. Available from: https://en.wikipedia.org/w/index.php?title=Embedded_system&oldid=1245980598

[56] Advantages and Disadvantages of Embedded System [Internet]. Javatpoint. [cited 2024 Jul 4]. Available from: https://www.javatpoint.com/advantages-and-disadvantages-of-embedded-system

[57] What is machine learning (ML)? [Internet]. IBM. 2024 [cited 2024 Jul 7]. Available from: https://www.ibm.com/topics/machine-learning

[58] What is machine learning? [Internet]. Mathworks. [cited 2024 Jul 8]. Available from: https://www.mathworks.com/discovery/machine-learning.html

[59] What is machine learning? [Internet]. Microsoft. [cited 2024 Jul 10]. Available from: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-machine-learning-platform/

[60] What is machine learning (ML)? [Internet]. UCB-UMT. 2020 [cited 2024 Jul 10]. Available from: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

[61] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015 May 28;521(7553):436-44.

[62] M. Tim Jones. Models for machine learning [Internet]. IBM. 2017 [cited 2024 Jul 11]. Available from: https://developer.ibm.com/articles/cc-models-machine-learning/

[63] Salian I. Difference between supervised, unsupervised, & reinforcement learning [Internet]. NVIDIA Blog. 2018 [cited 2024 Jul 11]. Available from: https://blogs.nvidia.com/blog/supervised-unsupervised-learning/

[64] Goodfellow I, Bengio Y, Courville A. Deep Learning. London, England: MIT Press; 2016

[65] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM [Internet]. 2012 [cited 2024 Jul 13];60:84–90. Available from: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[66] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998 Nov;86(11):2278-324.

[67] Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016.

[68] Kingma DP. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.

[69] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014 Jan 1;15(1):1929-58.

[70] Nadeski M. Bringing machine learning to embedded systems. Texas Instruments. 2019 Mar:1-5.

[71] Guide [Internet]. TensorFlow. [cited 2024 Jul 25]. Available from: https://www.tensorflow.org/guide

[72] TensorFlow vs Tensorflow Lite [Internet]. StackShare. [cited 2024 Jul 25]. Available from: https://stackshare.io/stackups/tensorflow-vs-tensorflow-lite

[73] LiteRT overview [Internet]. Google AI for Developers. [cited 2024 Jul 27]. Available from: https://www.tensorflow.org/lite/guide

[74] UNO R3 [Internet]. Arduino Docs. [cited 2024 Jul 28]. Available from: https://docs.arduino.cc/hardware/uno-rev3/

[75] What is Arduino? [Internet]. Arduino Docs. 2022. [cited 2024 Jul 28]. Available from: https://docs.arduino.cc/learn/starting-guide/whats-arduino/

[76] Misppro [Internet]. Amazon. [cited 2024 Jul 28]. Available from: https://www.amazon.com/misppro-20Pcs-Geared-Bracket-Support/dp/B0C48ZMNNM

[77] Features 1, Description 3. L293x Quadruple Half-H Drivers [Internet]. Texas Instruments. [cited 2024 Jul 30]. Available from: https://www.ti.com/lit/ds/symlink/l293d.pdf?ts=1721717756833&ref_url=https%253A%252F%252Fwww.mouser.de%252F

[78] Basics of PWM (Pulse Width Modulation) [Internet]. Arduino Docs. 2022 [cited 2024 Jul 30]. Available from: https://docs.arduino.cc/learn/microcontrollers/analog-output/

[79] Serial.available() [Internet]. Arduino Docs. 2019 [cited 2024 Jul 30]. Available from: https://www.arduino.cc/reference/en/language/functions/communication/serial/available/

[80] analogWrite() [Internet]. Arduino Docs. 2024 [cited 2024 Aug 3]. Available at https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

[81] Wikipedia contributors. Raspberry Pi [Internet]. Wikipedia, The Free Encyclopedia. 2024 [cited 2024 Aug 3]. Available from: https://en.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=1247205246

[82] PySerial API — pySerial 3.4 documentation [Internet]. Readthedocs. [cited 2024 Aug 4]. Available from: https://pyserial.readthedocs.io/en/latest/pyserial_api.html

[83] Save and load models [Internet]. TensorFlow. [cited 2024 Aug 17]. Available from: https://www.tensorflow.org/tutorials/keras/save_and_load

[84] Convert TensorFlow models [Internet]. Google AI for Developers. [cited 2024 Aug 18]. Available from: https://www.tensorflow.org/lite/models/convert/convert_models

[85] OpenCV: Getting started with videos [Internet]. Opencv. 2024 [cited 2024 Aug 20]. Available from: https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html

[86] Tf.lite.interpreter [Internet]. Google AI for Developers. 2022 [cited 2024 Aug 30]. Available from: https://www.tensorflow.org/lite/api_docs/python/tf/lite/Interpreter

[87] Python OpenCV [Internet]. GeeksforGeeks. 2023 [cited 2024 Sep 6]. Available from: https://www.geeksforgeeks.org/python-opencv-cv2-puttext-method/