ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αυτόματη επανεκπαίδευση και ενημέρωση μοντέλων μηχανικής μάθησης σε συσκευές IoT

Διπλωματική Εργασία

Δήμητρα Μαυροδή

Επιβλέπων: Λάλης Σπύρος

Σεπτέμβριος 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Automated ML model retraining and update in IoT devices

Diploma Thesis

Dimitra Mavrodi

Supervisor: Lalis Spyros

September 2025

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Λάλης Σπύρος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Αντωνόπουλος Χρήστος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Μπέλλας Νικόλαος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του Προπτυχιακού Προγράμματος Σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

Θα ήθελα να εκφράσω τις θερμές ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Σπύρο Λάλη για την πολύτιμη βοήθειά του, τις γνώσεις του, καθώς και τον χρόνο που αφιέρωσε στην καθοδήγησή μου.

Επίσης, δεν θα μπορούσα να παραλείψω την οικογένειά μου, για την αμέριστη αρωγή και τη βαθιά στήριξή τους όλα αυτά τα χρόνια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου για την υποστήριξή τους, αλλά και για όλες τις όμορφες αναμνήσεις που δημιουργήσαμε μαζί.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο/Η Δηλών/ούσα

Δήμητρα Μαυροδή

Διπλωματική Εργασία

Αυτόματη επανεκπαίδευση και ενημέρωση μοντέλων μηχανικής μάθησης σε συσκευές IoT

Δήμητρα Μαυροδή

Περίληψη

Στη σύγχρονη εποχή, η μηχανική μάθηση έχει κυριαρχήσει στον τεχνολογικό –και όχι μόνο– τομέα και αποτελεί κομμάτι της καθημερινότητας πολλών ανθρώπων, είτε το γνωρίζουν είτε όχι. Εκατοντάδες μοντέλα έχουν δημιουργηθεί και χρησιμοποιούνται σε αμέτρητες εφαρμογές, με στόχο την αυτοματοποίηση εργασιών, τη λήψη αποφάσεων, την πρόβλεψη γεγονότων και την ανάλυση μεγάλων όγκων δεδομένων. Ορισμένα από αυτά χρειάζονται μεγάλη υπολογιστική ισχύ, ώστε να μπορέσουν να εκπαιδευτούν και να παράγουν τα αποτελέσματά τους.

Ωστόσο, υπάρχουν και πιο απλοϊκά μοντέλα, τα οποία μπορούν να λειτουργούν και σε συσκευές με μικρότερες δυνατότητες απόδοσης, όπως είναι οι μικροελεγκτές (microcontrollers). Η χρήση τους προτείνεται σε περιπτώσεις όπου απαιτείται χαμηλό latency, καθώς δεν μεσολαβεί η επικοινωνία με το cloud, όπου δεν υπάρχει πρόσβαση στο διαδίκτυο και η συσκευή λειτουργεί offline, ή όπου είναι αναγκαία η εξοικονόμηση ενέργειας και κόστους. Παρ’ όλα αυτά, προκειμένου το μοντέλο που εκτελείται σε αυτές τις συσκευές να παραμένει διαρκώς ενημερωμένο (up to date), θα πρέπει να επανεκπαιδεύεται με νέα δεδομένα που θα προστίθενται στα υπάρχοντα. Ο ρυθμός αυτής της διαδικασίας εξαρτάται από το μέγεθος του μοντέλου και από την ανάγκη που υπάρχει κάθε φορά για βελτίωσή του.

Η επανεκπαίδευση μπορεί να πραγματοποιηθεί είτε μέσω επικοινωνίας με το cloud είτε πάνω στον ίδιο τον μικροελεγκτή (microcontroller). Στην πρώτη προσέγγιση, η διαδικασία που ακολουθείται, εν συντομία, είναι η εξής: οι συσκευές στέλνουν επιπλέον δεδομένα στο cloud, το οποίο εκπαιδεύει ένα καινούριο μοντέλο και το επιστρέφει πίσω στις συσκευές. Στη δεύτερη προσέγγιση, ολόκληρη η διαδικασία πραγματοποιείται πάνω στη συσκευή και δεν απαιτείται ανταλλαγή δεδομένων. Ωστόσο, λόγω των περιορισμένων πόρων, απαιτείται το μοντέλο να είναι αρκετά απλοϊκό.

Μια επιπλέον μέθοδος που χρησιμοποιείται ευρέως είναι το Federated Learning (FL). Με αυτήν, όλα τα μοντέλα μηχανικής μάθησης που εκτελούνται πάνω στους μικροελεγκτές,

στην περίπτωση της επανεκπαίδευσης, αποστέλλονται στο cloud. Το cloud, με τη χρήση αυτών των μοντέλων, δημιουργεί ένα ενιαίο (συγκεντρωτικό) μοντέλο, το οποίο επιστρέφει στη συνέχεια σε κάθε συσκευή. Αξίζει να σημειωθεί ότι οι συσκευές πρέπει να στείλουν στο cloud ένα ήδη εκπαιδευμένο μοντέλο, ώστε να αποφευχθεί η μεταφορά των ίδιων των δεδομένων. Το βασικό πλεονέκτημα αυτής της μεθόδου είναι ότι τελικά δημιουργείται και αποστέλλεται ένα μοντέλο το οποίο έχει εκπαιδευτεί με δεδομένα που προέρχονται από διαφορετικά περιβάλλοντα, και έτσι έχουν καλυφθεί περισσότερες περιπτώσεις.

Στα πλαίσια αυτής της εργασίας αναλύθηκαν, υλοποιήθηκαν και αξιολογήθηκαν οι τρεις προσεγγίσεις που αναφέρθηκαν παραπάνω, καθώς και το περιβάλλον που δημιουργήθηκε για τον σκοπό αυτό. Η μελέτη αυτή ανοίγει τον δρόμο για μελλοντικές επεκτάσεις σε πιο σύνθετα μοντέλα και εφαρμογές σε πραγματικές έξυπνες συσκευές.

Λέξεις-κλειδιά:

Machine Learning, Retraining, Microcontrollers/Embedded Systems, On-Device Training, Federated Learning

Diploma Thesis

Automated ML model retraining and update in IoT devices

Dimitra Mavrodi

Abstract

In the modern era, machine learning has dominated the technology sector—and beyond—and has become a part of many people's daily lives, whether they realize it or not. Hundreds of models have been created and are used in countless applications, aiming to automate tasks, make decisions, predict events, and analyze large volumes of data. Some of these models require significant computational power to be trained and produce results.

However, there are simpler models that can operate on devices with lower performance capabilities, such as microcontrollers. Their use is recommended in cases where low latency is needed, because there is no communication with the cloud, where there is no internet access and the device operates offline, or where there is a need to save energy and costs. Nevertheless, in order for the model running on these devices to stay constantly updated, it must be retrained with new data added to the existing ones. The frequency of this process depends on the size of the model and the need for improvement each time.

Retraining can be done either via communication with the cloud or directly on the microcontroller itself. In the first approach, the process is briefly as follows: devices send additional data to the cloud, which trains a new model and sends it back to the devices. In the second approach, the entire process takes place on the device without data exchange. However, due to limited resources, the model must be quite simple.

Another widely used method is Federated Learning (FL). In this method, all machine learning models running on microcontrollers are sent to the cloud for retraining. The cloud uses these models to create a single (aggregated) model, which is then sent back to each device. It is worth noting that devices must send an already trained model to the cloud to avoid transferring the actual data. The main advantage of this method is that, in the end, a model is created and sent which has been trained with data coming from different environments, thereby covering a wider range of cases.

In the context of this work, the three approaches mentioned above were analyzed, implemented, and evaluated, along with the environment that was developed for this purpose.

This study paves the way for future extensions to more complex models and applications in real-world smart devices

Keywords:

Machine Learning, Retraining, Microcontrollers/Embedded Systems, On-Device Training, Federated Learning

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κατάλογος πινάκων

Συντομογραφίες

ML	Machine Learning
FL	Federated Learning
IoT	Internet of Things
AI	Artificial intelligence
TFLM	TensorFlow Lite Micro
JSON	JavaScript Object Notation
ECDWS	Enhanced Class Distribution Weighted Sum
GPIO	General Purpose Input/Output
ADC	Analog to Digital Converter
DAC	Digital to Analog Converters
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver/Transmitter
I²C	Inter-Integrated Circuit
PWM	Pulse Width Modulation
RTOS	Real Time Operating System
CPU	Central Processing Unit
RAM	Random Access Memory
SRAM	Static Random Access Memory
KB	Kilobyte
IP	Internet Protocol
HTTP	HyperText Transfer Protocol
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
OTA	Over The Air

ID	Identification
IL-SBR	Independent learning with server-based retraining
IL-ODR	Independent learning with on-device retraining
FL-SS	Federated learning with shared server
ViFLa	Virtual Federated Learning Approach
SISA	Sharded, Isolated, Sliced, and Aggregated Training

Κεφάλαιο 1

Εισαγωγή

Είναι γεγονός ότι τα τελευταία χρόνια, στον κόσμο των υπολογιστών, παρατηρείται μια έντονη στροφή προς το λεγόμενο Διαδίκτυο των Πραγμάτων (Internet of Things – IoT), με όλο και περισσότερες συσκευές να κατασκευάζονται και να χρησιμοποιούνται για την κάλυψη αναγκών σε διάφορους τομείς της καθημερινότητας, όπως στην ιατρική, στη βιομηχανία, στο περιβάλλον και αλλού. Στις ενσωματωμένες αυτές συσκευές μπορούν να συνδεθούν διάφοροι τύποι αισθητήρων, από τους οποίους συλλέγονται δεδομένα και υφίστανται κατάλληλη επεξεργασία. Ο ρυθμός ανάγνωσης τιμών μπορεί να είναι ιδιαίτερα υψηλός, κάτι που καθίσταται χρήσιμο στα συστήματα που απαιτούν χαμηλή απόκριση (latency).

Ένα ακόμη κίνητρο που ελκύει τη χρήση αυτών των συσκευών είναι το γεγονός ότι προσφέρουν ιδιωτικότητα. Συλλέγουν δεδομένα από τους αισθητήρες και, σε έναν βαθμό, μπορούν να τα επεξεργαστούν τοπικά και να λάβουν αποφάσεις. Έτσι, δεν είναι απαραίτητη η αποστολή τους σε πιο ισχυρά συστήματα, όπως στο νέφος (cloud), για να γίνει η κατάλληλη επεξεργασία τους. Η συγκεκριμένη τάση ονομάζεται edge computing, καθώς οι υπολογισμοί εκτελούνται στα άκρα του δικτύου, δηλαδή στις ενσωματωμένες συσκευές. Ωστόσο, αυτό εξαρτάται σε μεγάλο βαθμό από την πολυπλοκότητα της απόφασης, καθώς ορισμένα συστήματα δεν διαθέτουν υψηλή υπολογιστική ισχύ και, σε κάποιες περιπτώσεις, ίσως χρειάζεται να καταφεύγουν σε πιο ισχυρές λύσεις, όπως στο νέφος.

Μια ακόμη τάση της εποχής, είναι η επεξεργασία των δεδομένων σε τέτοιες συσκευές με τη χρήση μοντέλων μηχανικής μάθησης (machine learning models), ώστε να πραγματοποιούνται προβλέψεις αλλά και να ανιχνεύονται καταστάσεις. Όμως, πέρα από την εκπαίδευση τους, για να μπορέσουν να εκτελεστούν, τα μοντέλα αυτά, θα πρέπει πρώτα να μετατραπούν σε κατάλληλη, πιο απλοποιημένη μορφή, ώστε να είναι συμβατά με τη χαμηλή

υπολογιστική ισχύ της συσκευής.

Παράλληλα, έχει διαπιστωθεί ότι, όταν ένα μοντέλο μηχανικής μάθησης εκτελείται για μεγάλο χρονικό διάστημα, είναι πιθανό με τον καιρό να αρχίσει να αποκλίνει από τον στόχο του και να παράγει όλο και περισσότερες λανθασμένες προβλέψεις. Αυτό έχει επιβεβαιωθεί και από τους συγγραφείς του [1], οι οποίοι αναφέρουν την ύπαρξη φαινομένου της εννοιολογικής απόκλισης (concept drift) κατά τη μακράς διάρκειας λειτουργία μοντέλων σε ενσωματωμένα συστήματα και προτείνουν έναν αλγόριθμο για τον περιορισμό του. Ένας ακόμη τρόπος για να αντιμετωπιστεί αυτό το πρόβλημα είναι η επανεκπαίδευση του μοντέλου με νέα δεδομένα, ώστε να προσαρμοστεί στις καινούριες συνθήκες. Ωστόσο, η διαδικασία αυτή καθίσταται ιδιαίτερα δύσκολη όταν το μοντέλο εκτελείται σε ενσωματωμένα συστήματα, στα οποία οι υπολογιστικοί πόροι είναι εξαιρετικά περιορισμένοι. Αν και η εκτέλεση του μοντέλου μπορεί να είναι εφικτή, η επανεκπαίδευσή του παραμένει μια σημαντική πρόκληση. Αυτή την πρόκληση επιχειρεί να αντιμετωπίσει η παρούσα εργασία, προτείνοντας και υλοποιώντας διάφορες μεθόδους για την αποτελεσματική επανεκπαίδευση μοντέλων σε περιβάλλοντα με περιορισμένους υπολογιστικούς πόρους.

1.1 Αντικείμενο της διπλωματικής

Όπως αναφέρθηκε και στην προηγούμενη ενότητα, η παρούσα εργασία επικεντρώνεται στη διαδικασία της επανεκπαίδευσης, με σκοπό τη βελτίωση της απόδοσης ενός μοντέλου μηχανικής μάθησης όταν αυτή αρχίζει να μειώνεται. Στο πλαίσιο αυτό, προτείνονται, υλοποιούνται και αξιολογούνται τρεις διαφορετικές μέθοδοι που στοχεύουν στην αντιμετώπιση αυτής της πρόκλησης.

Η πρώτη μέθοδος βασίζεται στην ιδέα ότι η επανεκπαίδευση δεν πραγματοποιείται απευθείας πάνω στο ενσωματωμένο σύστημα, αλλά σε έναν εξωτερικό εξυπηρετητή (server) με τον οποίο αυτό επικοινωνεί. Έτσι, η πολυπλοκότητα του μοντέλου παύει να αποτελεί περιοριστικό παράγοντα, αφού η εκπαιδευτική διαδικασία εκτελείται από έναν υπολογιστικά ισχυρό εξυπηρετητή. Κάθε φορά που τα σφάλματα που έχουν καταγραφεί από τη συσκευή υπερβαίνουν ένα προκαθορισμένο όριο, αποστέλλονται τα συλλεχθέντα δεδομένα στον εξυπηρετητή. Εκείνος με τη σειρά του επανεκπαιδεύει το μοντέλο, το μετατρέπει σε μορφή κατάλληλη για ενσωματωμένα συστήματα και το επιστρέφει στη συσκευή. Μια επιπλέον βελτιστοποίηση που εφαρμόζεται είναι η αποστολή μόνο των διαφορών (μεταβολών) μεταξύ του νέου και

του παλαιού μοντέλου, ώστε να μειωθεί το μέγεθος των δεδομένων που διακινούνται στο δίκτυο.

Η δεύτερη μέθοδος υλοποιείται σε περιπτώσεις όπου το μοντέλο είναι αρκετά απλό, επιτρέποντας την επανεκπαίδευσή του απευθείας πάνω στη συσκευή, χωρίς την ανάγκη χρήσης εξυπηρετητή. Τα βάρη του μοντέλου αποθηκεύονται τοπικά σε μορφή πινάκων, και η επανεκπαίδευση πραγματοποιείται μέσω της υλοποίησης των συναρτήσεων προώθησης (forward) και οπισθοδιάδοσης (backward), υπολογίζοντας τις νέες τιμές των βαρών βάσει των νεότερων δεδομένων.

Η τρίτη μέθοδος συνδυάζει τις υλοποιήσεις των δύο προηγούμενων και βασίζεται στη φιλοσοφία της ομοσπονδιακής μάθησης (federated learning). Σε αυτό το σενάριο, πολλές συσκευές λειτουργούν παράλληλα και συνεργάζονται για να εκπαιδεύσουν συλλογικά ένα κοινό μοντέλο. Αρχικά, κάθε συσκευή ”τρέχει” το δικό της τοπικό, απλοποιημένο μοντέλο. Όταν απαιτείται επανεκπαίδευση, η συσκευή δεν στέλνει τα δεδομένα της, αλλά τα επικαιροποιημένα βάρη του μοντέλου της σε έναν προσυμφωνημένο εξυπηρετητή. Ο εξυπηρετητής συλλέγει τα βάρη από όλες τις συσκευές που συμμετέχουν στον εκάστοτε γύρο εκπαίδευσης και υπολογίζει τον μέσο όρο αυτών. Στη συνέχεια, αποστέλλει τα ενοποιημένα βάρη πίσω στις συσκευές, οι οποίες αντικαθιστούν τα παλαιότερα με τα νέα. Με τον τρόπο αυτό, εξασφαλίζεται καλύτερη ιδιωτικότητα, καθώς δεν διακινούνται προσωπικά δεδομένα, αλλά μόνο παράγωγα αυτών (τα βάρη του μοντέλου).

Τέλος, πραγματοποιείται μια συνολική σύγκριση και αξιολόγηση των τριών μεθόδων, με σκοπό την εξαγωγή συμπερασμάτων σχετικά με το ποια προσφέρει την καλύτερη απόδοση υπό διαφορετικές συνθήκες.

1.2 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν οι ανάγκες και οι προκλήσεις που σχετίζονται με την επανεκπαίδευση μοντέλων μηχανικής μάθησης, ειδικά σε περιβάλλοντα με περιορισμένους υπολογιστικούς πόρους, όπως τα ενσωματωμένα συστήματα.

2. Σχεδιάστηκε και υλοποιήθηκε μία απλή και κατανοητή συσκευή, κατάλληλη για τις ανάγκες της εργασίας, ώστε να είναι εύκολος ο εντοπισμός λανθασμένων προβλέψεων και η αξιολόγηση της αποτελεσματικότητας της επανεκπαίδευσης.

3. Υλοποιήθηκαν τρεις μέθοδοι επανεκπαίδευσης μοντέλων: (i) *Independent learning with server-based retraining (IL-SBR)*, όπου η επανεκπαίδευση πραγματοποιείται σε απομακρυσμένο εξυπηρετητή, (ii) *Independent learning with On-Device retraining (IL-ODR)*, όπου η επανεκπαίδευση γίνεται τοπικά στη συσκευή, και (iii) *Federated learning with shared server (FL-SS)*, όπου πολλές συσκευές εκπαιδεύουν το μοντέλο συνεργατικά, ανταλλάσσοντας μόνο τα βάρη των τοπικών μοντέλων τους.

4. Σχεδιάστηκε η αρχιτεκτονική του εξυπηρετητή, ώστε να διαχειρίζεται τα αιτήματα των συσκευών, να επανεκπαιδεύει μοντέλα και να επιστρέφει τις ενημερώσεις με αποδοτικό και ασφαλή τρόπο.

5. Μελετήθηκαν εναλλακτικές προσεγγίσεις ομοσπονδιακής μάθησης, όπως τα ViFLa και SISA, με στόχο τη βελτίωση της αποδοτικότητας και της ασφάλειας.

6. Πραγματοποιήθηκε συγκριτική αξιολόγηση των μεθόδων επανεκπαίδευσης ως προς την ακρίβεια, τον απαιτούμενο χρόνο και την κατανάλωση πόρων. Από τα πειραματικά αποτελέσματα προέκυψε ότι η πιο αποδοτική μέθοδος είναι η IL-ODR, η οποία όμως είναι αρκετά περιοριστική. Για τον λόγο αυτό, η αμέσως επόμενη και πιο γενικής χρήσης λύση είναι η IL-SBR με τη βελτιωμένη εκδοχή της, η οποία θα αναλυθεί παρακάτω.

1.3 Δομή της εργασίας

Το υπόλοιπο κείμενο της διπλωματικής εργασίας είναι δομημένο ως εξής:

Το Κεφάλαιο 2 παρουσιάζει μια θεωρητική θεμελίωση, η οποία περιλαμβάνει τις βασικές έννοιες που είναι απαραίτητες για την κατανόηση του έργου. Παρουσιάζονται, με συνοπτικό τρόπο, στοιχεία μηχανικής μάθησης (ML), η προσέγγιση του λεγόμενου TinyML, καθώς και οι αρχές λειτουργίας της ομοσπονδιακής μάθησης.

Το Κεφάλαιο 3 εστιάζει στην περιγραφή της συσκευής και του εξυπηρετητή που χρησιμοποιούνται στην υλοποίηση. Παρουσιάζεται ο μικροελεγκτής *ESP32* και αναλύεται η δομή του συστήματος, του μοντέλου και του νέφους, καθώς και ο τρόπος συλλογής ημερήσιων δεδομένων από τη συσκευή. Επιπλέον, γίνεται αναφορά στις πλατφόρμες που αξιοποιήθηκαν για την ανάπτυξη και υλοποίηση του παρόντος έργου.

Στα επόμενα τρία κεφάλαια παρουσιάζονται αναλυτικά οι μέθοδοι επανεκπαίδευσης που

υλοποιήθηκαν. Πιο συγκεκριμένα, το κεφάλαιο 4 περιγράφει την μέθοδο *IL-SBR*, όπου η επανεκπαίδευση γίνεται σε απομακρυσμένο εξυπηρετητή με ισχυρούς υπολογιστικούς πόρους, το κεφάλαιο 5 περιγράφει την προσέγγιση *IL-ODR*, όπου η επανεκπαίδευση πραγματοποιείται τοπικά στη συσκευή, χωρίς εξωτερική υποδομή, και το κεφάλαιο 6 περιγράφει την διαδικασία της *FL-SS*, στην οποία πολλές συσκευές συνεργάζονται για την εκπαίδευση ενός κοινού μοντέλου, χωρίς να ανταλλάσσουν άμεσα δεδομένα.

Στο Κεφάλαιο 7 δίνεται μια συγκριτική αξιολόγηση των παραπάνω τεχνικών ως προς την ακρίβεια, την απόδοση, τη χρήση πόρων και τη γενική αποδοτικότητα.

Η εργασία ολοκληρώνεται με το Κεφάλαιο 8, όπου παρουσιάζονται τα συμπεράσματα που προέκυψαν από την εργασία και προτείνονται πιθανές μελλοντικές κατευθύνσεις και βελτιώσεις.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Εισαγωγή

Στο κεφάλαιο που ακολουθεί παρουσιάζονται οι βασικές αρχές της Μηχανικής Μάθησης και των Νευρωνικών Δικτύων. Αναλύεται η έννοια του TinyML και οι δυνατότητες που προσφέρει το TensorFlow Lite. Επίσης, εξετάζονται οι προκλήσεις της διαδικασίας της επανεκπαίδευσης σε ενσωματωμένα περιβάλλοντα και, τέλος, παρουσιάζεται το μοντέλο της ομοσπονδιακής μάθησης ως μια πολλά υποσχόμενη λύση για τη μελλοντική ανάπτυξη έξυπνων και συνεργατικών συστημάτων.

2.2 Μηχανική μάθηση και νευρωνικά δίκτυα

Η Μηχανική Μάθηση (ML) είναι ένας κλάδος της Τεχνητής Νοημοσύνης (AI) που επιτρέπει στους υπολογιστές να μαθαίνουν χωρίς ρητό προγραμματισμό. Αντί να ακολουθούν ένα σύνολο οδηγιών, οι αλγόριθμοι ML χρησιμοποιούν δεδομένα για να αναγνωρίζουν μοτίβα, να κάνουν προβλέψεις και να βελτιώνονται με την πάροδο του χρόνου. Η ML μπορεί γενικά να κατηγοριοποιηθεί σε ορισμένους τύπους, με βάση την προσέγγιση δεδομένων και μάθησης:

- **Εποπτευόμενη μάθηση**: Σε αυτήν την περίπτωση, τα δεδομένα φέρουν ετικέτες με το επιθυμητό αποτέλεσμα. Το μοντέλο μαθαίνει να αντιστοιχίζει τις εισόδους με τις αντίστοιχες εξόδους.

- **Μη εποπτευόμενη μάθηση**: Αυτός ο τύπος ML χρησιμοποιεί δεδομένα χωρίς ετικέ-

τες. Το μοντέλο προσπαθεί να εντοπίσει μοτίβα και σχέσεις μέσα στα δεδομένα, χωρίς καθορισμένες εξόδους.

- **Ημι-εποπτευόμενη μάθηση**: Συνδυάζει δεδομένα με και χωρίς ετικέτες για την εκπαίδευση του μοντέλου. Είναι ιδιαίτερα χρήσιμη όταν τα δεδομένα με ετικέτα είναι περιορισμένα, αλλά υπάρχουν διαθέσιμα πολλά μη επισημασμένα δεδομένα.

Ορισμένες διαδικασίες στον κόσμο της Μηχανικής Μάθησης (ML) χαρακτηρίζονται ως παλινδρόμηση (*regression*), καθώς ο στόχος τους είναι, βάσει κάποιων τιμών και ακολουθιών, να προβλέψουν το αποτέλεσμα του μέλλοντος. Από την άλλη, υπάρχουν και οι διαδικασίες ομαδοποίησης (*classification*), στις οποίες το μοντέλο προσπαθεί να εντοπίσει σε ποια από τις δοθείσες κατηγορίες ανήκει ένα αντικείμενο, με βάση άλλα προηγούμενα παραδείγματα. Αυτές αποτελούν δύο από τις μεγαλύτερες κατηγορίες προβλημάτων που καλείται να λύσει σήμερα η Μηχανική Μάθηση. Επιπλέον περισσότερες πληροφορίες παρουσιάζονται στο [2].

Επιπλέον, ένα από τα συχνά προβλήματα που παρατηρούνται στα ML μοντέλα (όπως και στις μεθόδους που υλοποιήθηκαν στην παρούσα εργασία, και αξιολογούνται στο Κεφάλαιο 7) είναι το **concept drift**. Συγκεκριμένα, πρόκειται για την περίπτωση κατά την οποία αλλάζει η σχέση μεταξύ εισόδου και εξόδου ενός μοντέλου με την πάροδο του χρόνου. Δηλαδή, το μοντέλο «μαθαίνει» μια σχέση ανάμεσα στα χαρακτηριστικά X και την πρόβλεψη Y, η οποία όμως δεν παραμένει σταθερή και μεταβάλλεται σε μεταγενέστερο χρόνο. Αυτό μπορεί να οφείλεται σε περιπτώσεις όπου οι συνθήκες ή οι νόμοι που καθορίζουν το αποτέλεσμα αλλάζουν με τον χρόνο.

Επίσης, μια πολύ μεγάλη κατηγορία των μοντέλων μηχανικής μάθησης είναι τα νευρωνικά δίκτυα, ή αλλιώς neural networks. Ένα νευρωνικό δίκτυο βασίζεται στην δομή ενός βιολογικού εγκεφάλου. Οι τεχνητοί νευρώνες ονομάζονται κόμβοι και οργανώνονται σε πολλαπλά στρώματα, λειτουργώντας παράλληλα. Όταν ένας από αυτούς λαμβάνει ένα αριθμητικό σήμα, το επεξεργάζεται και σηματοδοτεί τους υπόλοιπους με τους οποίους είναι συνδεδεμένος. Όπως και στον ανθρώπινο εγκέφαλο, η «νευρική ενίσχυση» έχει ως αποτέλεσμα τη βελτίωση της αναγνώρισης προτύπων, της εξειδίκευσης και της συνολικής ικανότητας για μάθηση. Ένα βασικό νευρωνικό δίκτυο περιλαμβάνει διασυνδεδεμένους τεχνητούς νευρώνες οργανωμένους σε τρία επίπεδα:

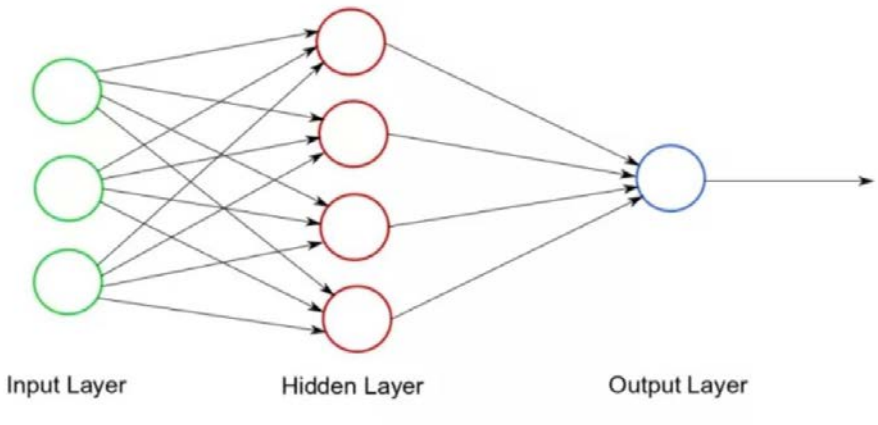
- **Επίπεδο εισόδου (input layer)**: Πρόκειται για το πρώτο επίπεδο σε ένα νευρωνικό

δίκτυο που λαμβάνει τα αρχικά δεδομένα εισόδου. Οι κόμβοι επεξεργάζονται τα δεδομένα αυτά, τα αναλύουν και τα μεταβιβάζουν στο επόμενο επίπεδο.

- **Κρυμμένο επίπεδο (hidden layer)**: Τα κρυμμένα στρώματα αποτελούν τα ενδιάμεσα επίπεδα ανάμεσα σε αυτό της εισόδου και της εξόδου και επιτελούν το μεγαλύτερο μέρος των υπολογισμών. Ενδέχεται να υπάρχουν πολλά κρυμμένα επίπεδα σε ένα νευρωνικό δίκτυο.

- **Επίπεδο εξόδου (output layer)**: Το επίπεδο εξόδου είναι το τελικό σε ένα νευρωνικό δίκτυο και οδηγεί στην έξοδο του δικτύου. Ο αριθμός των νευρώνων εξαρτάται από το εκάστοτε πρόβλημα που επιχειρείται να λυθεί.

Στην εικόνα 2.1 απεικονίζεται ένα απλό νευρωνικό δίκτυο, στο οποίο φαίνονται όλοι οι κόμβοι καθώς και οι συνδέσεις μεταξύ των επιπέδων του.



Σχήμα 2.1: Απεικόνιση ενός απλού Νευρωνικού Μοντέλου

2.3 TinyML – Ορισμός – TensorFlow Lite

Είναι γεγονός οτι τα σημερινά μοντέλα ML και AI έχουν έναν σημαντικό περιορισμό: απαιτούν τεράστια υπολογιστική και επεξεργαστική ισχύ για να επιτευχθούν τα επιθυμητά αποτελέσματα με ακρίβεια. Αυτό συχνά περιορίζει τη χρήση τους σε συσκευές υψηλής ικανότητας, με εκτεταμένους διαθέσιμους πόρους. Όμως, δεδομένων των προόδων που έγιναν στην τεχνολογία των ενσωματωμένων συστημάτων και της ουσιαστικής ανάπτυξης στη βιομηχανία του Διαδικτύου των Πραγμάτων (IoT), είναι επιθυμητό να ενσωματωθεί η χρήση

τεχνικών και εννοιών ML σε ένα ενσωματωμένο σύστημα με περιορισμένους πόρους. Η επιθυμία αυτή είναι ο κύριος κινητήριος παράγοντας πίσω από την ανάπτυξη του TinyML.

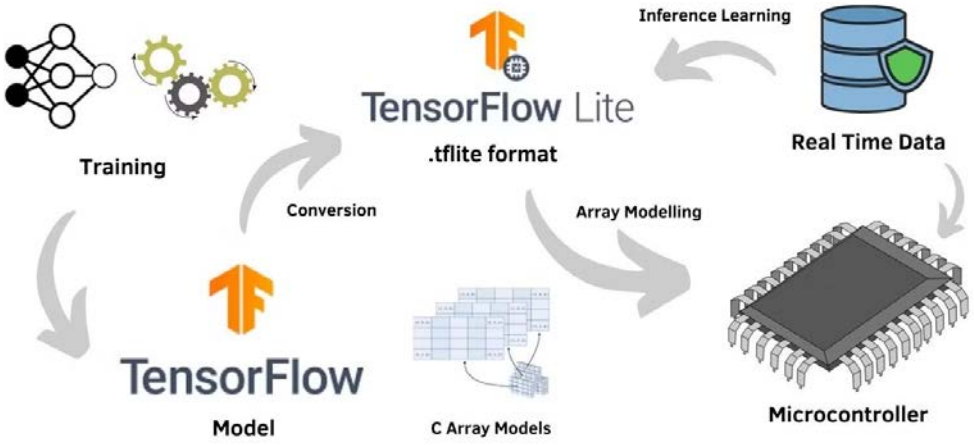
Με λίγα λόγια, το TinyML είναι ένας τύπος Μηχανικής Μάθησης που επιτρέπει στα μοντέλα να εκτελούνται σε μικρότερες, λιγότερο ισχυρές συσκευές και να τις προσφέρει νοημοσύνη. Περιλαμβάνει υλικό, αλγορίθμους και λογισμικό ικανό να αναλύει δεδομένα από αισθητήρες με ελάχιστη κατανάλωση ενέργειας, καθιστώντας το ιδανικό για συνεχή λειτουργία και για συσκευές που τροφοδοτούνται με μπαταρία. Κάποια από τα βασικά του χαρακτηριστικά είναι:

- **Ενεργειακή Απόδοση**: Οι συσκευές συχνά λειτουργούν με περιορισμένη ενέργεια, κάτι που δημιουργεί προβλήματα στους χρήστες. Τα μοντέλα TinyML είναι εξαιρετικά ελαφριά και αποδοτικά, καταναλώνοντας πολύ λιγότερη ενέργεια σε σύγκριση με τα παραδοσιακά μοντέλα ML που τρέχουν στο νέφος. Αυτό σημαίνει ότι μπορούν να εφαρμοστούν σε συσκευές χαμηλής ισχύος και να λειτουργούν για μεγάλα χρονικά διαστήματα χωρίς την ανάγκη συχνής φόρτισης.

- **Υπολογισμός στην αιχμή (Edge Computing) και Μειωμένη Καθυστέρηση**: Η τεχνολογία TinyML επιτρέπει την εκτέλεση μοντέλων μηχανικής μάθησης απευθείας στις συσκευές, χωρίς να απαιτείται συνεχής μετάδοση δεδομένων σε κεντρικούς διακομιστές. Αυτό δίνει τη δυνατότητα για ομαλή και αποδοτική λειτουργία των εφαρμογών ακόμη και χωρίς σύνδεση στο διαδίκτυο, μειώνοντας τον χρόνο απόκρισης.

- **Κλιμάκωση και Χαμηλό Κόστος**: Το TinyML επιτρέπει την ανάπτυξη εφαρμογών μηχανικής μάθησης χωρίς την ανάγκη ακριβού εξοπλισμού. Έτσι, οι οργανισμοί μπορούν να μειώσουν τα κόστη, καθώς εξαρτώνται λιγότερο από υποδομές νέφους.

Το TensorFlow Lite είναι μια ελαφριά έκδοση του TensorFlow (δωρεάν και ανοιχτού κώδικα βιβλιοθήκη για Machine Learning) και έχει σχεδιαστεί για να λειτουργεί σε συσκευές με χαμηλή υπολογιστική ισχύ, όπως κινητά ή συσκευές IoT. Είναι ιδανικό για την εκτέλεση μοντέλων Μηχανικής Μάθησης απευθείας πάνω σε αυτές. Η ροή που ακολουθείται για τη δημιουργία ενός τέτοιου μοντέλου φαίνεται στην εικόνα 2.2. Αρχικά, κατασκευάζεται ένα TensorFlow model, το οποίο στη συνέχεια μετατρέπεται σε TensorFlow Lite με τη δημιουργία ενός αρχείου .tflite, που περιέχει το μοντέλο σε μορφή C πινάκων. Στη συνέχεια, το αρχείο αυτό αποθηκεύεται στον μικροελεγκτή και ξεκινά η εκτέλεση του μοντέλου.



Σχήμα 2.2: Διαδικασία Δημιουργίας ενός TinyML Model

Στο [3] γίνεται μια πολύ καλή προσέγγιση γύρω από το TFLM (TensorFlow Lite Micro) καθώς περιγράφει τις σχεδιαστικές αποφάσεις πίσω από το αυτό και παρουσιάζει μια αξιολόγηση για να αποδείξει τις χαμηλές απαιτήσεις πόρων και την ελάχιστη καθυστέρηση εκτέλεσης.

2.4 Σημασία της επανεκπαίδευσης και προκλήσεις σε ενσωματωμένα συστήματα

Υπάρχουν πολλές περιπτώσεις όπου ένα μοντέλο μηχανικής μάθησης, το οποίο μπορεί να εκτελείται είτε σε κάποιο ισχυρό μηχάνημα είτε σε κάποιο ενσωματωμένο σύστημα, να έχει εμφανίσει μείωση της ικανότητάς του να προβλέπει σωστά αυτό για το οποίο είναι υπεύθυνο. Αυτή η έλλειψη της απόδοσής του μπορεί να προκύψει μετά από μικρό έως και μεγάλο χρονικό διάστημα από τη στιγμή που έχει αρχίσει να λειτουργεί. Ένας από τους συνηθέστερους λόγους για τους οποίους μπορεί να συμβαίνει αυτό είναι ότι τα δεδομένα πάνω στα οποία έχει εκπαιδευτεί το μοντέλο μπορεί να έχουν αρχίσει να αποκλίνουν από τα πραγματικά, καθώς το περιβάλλον στο οποίο λειτουργεί, ενδέχεται να αλλάζει. Η αλλαγή αυτή οδηγεί σε ορισμένες λανθασμένες προβλέψεις — προβλέψεις που είναι εκτός των ορίων που έχουν τεθεί — του μοντέλου.

Προκειμένου να αντιμετωπιστεί το παραπάνω φαινόμενο, η επανεκπαίδευση του μοντέλου αποτελεί μια καλή λύση. Έτσι, όταν οι λανθασμένες προβλέψεις ξεπεράσουν ένα ορι-

σμένο όριο (threshold), τότε ενεργοποιείται η διαδικασία της επανεκπαίδευσης (retraining).

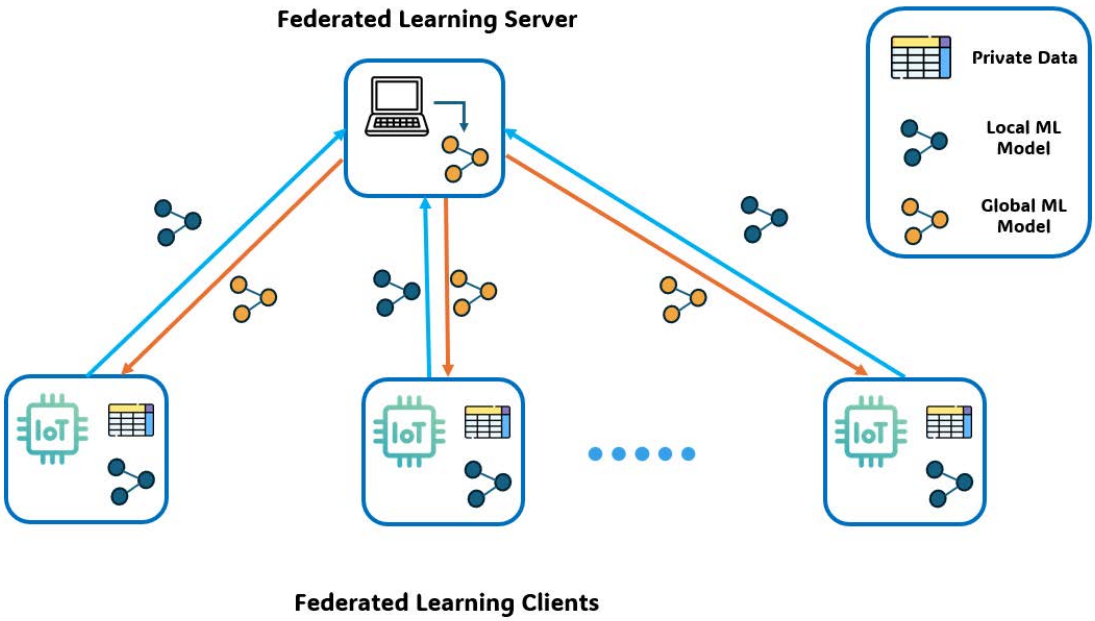
Ωστόσο, όταν πρόκειται για μοντέλο που εκτελείται σε ένα ισχυρό μηχάνημα, τότε αυτό μπορεί να συλλέξει νέα δεδομένα και να εκπαιδεύσει το νέο μοντέλο χωρίς να απαιτείται η παρέμβαση κάποιας άλλης συσκευής, καθώς διαθέτει τους απαραίτητους πόρους για να το πραγματοποιήσει. Αντίθετα, όταν πρόκειται για ενσωματωμένα συστήματα, όπου οι δυνατότητες είναι περιορισμένες, η εκπαίδευση του μοντέλου απευθείας στη συσκευή αποτελεί πρόκληση. Προκειμένου να είναι εφικτή μια εκπαίδευση επί της συσκευής (on-device training), είναι απαραίτητη προϋπόθεση το μοντέλο να είναι ιδιαίτερα απλό και μικρό, ώστε οι υπολογισμοί των βαρών (weights) του να μπορούν να εκτελεστούν με τους διαθέσιμους πόρους της συσκευής.

Στην παρούσα διπλωματική εργασία υλοποιούνται τρεις μέθοδοι επανεκπαίδευσης σε ενσωματωμένα συστήματα. Στην πρώτη μέθοδο, επιγραμματικά, η συσκευή συλλέγει δεδομένα και, όταν είναι απαραίτητη η επανεκπαίδευση, τα αποστέλλει (σε μορφή JSON) σε έναν εξυπηρετητή. Αυτός εκπαιδεύει ένα νέο μοντέλο χρησιμοποιώντας τόσο τα καινούρια όσο και τα παλαιότερα δεδομένα, το μετατρέπει σε μορφή κατάλληλη για TinyML και το επιστρέφει πίσω σε αυτήν. Στη δεύτερη προσέγγιση, η συσκευή εκπαιδεύει ένα πιο απλοϊκό μοντέλο, γεγονός που καθιστά εφικτή την εκπαίδευση απευθείας ”πάνω” της (on-device training). Ενώ στην τρίτη, συμμετέχει και πάλι ο εξυπηρετητής αλλά αυτή την φορά συλλέγει τα βάρη των μοντέλων από όλες τις συσκευές που συμμετέχουν και δημιουργεί το ενιαίο. Οι παραπάνω υλοποιήσεις περιγράφονται αναλυτικά στα επόμενα Κεφάλαια 4, 5 και 6 αντίστοιχα.

2.5 Ορισμός και πλεονεκτήματα της ομοσπονδιακής μάθησης

Ομοσπονδιακή μάθηση είναι μια μέθοδος μηχανικής μάθησης κατά την οποία οι συσκευές που συνυπάρχουν σε ένα κοινό οικοσύστημα συνεργάζονται για να εκπαιδεύσουν από κοινού ένα μοντέλο μηχανικής μάθησης. Η μέθοδος αυτή είναι ιδιαίτερα χρήσιμη για ενσωματωμένα συστήματα, καθώς αυτά, μεμονωμένα, δεν διαθέτουν επαρκείς υπολογιστικούς πόρους για την εκπαίδευση ενός ισχυρού και ικανού μοντέλου. Σε αυτό το πλαίσιο, κάθε συσκευή συλλέγει δεδομένα από το περιβάλλον στο οποίο βρίσκεται και εκπαιδεύει ένα απλοποιημένο μοντέλο. Αν αυτό πραγματοποιεί μεγάλο αριθμό λανθασμένων προβλέψεων, τότε η συσκευή αποστέλλει τα βάρη του, ώστε να συμπεριληφθούν στη διαδικασία

της ομοσπονδιακής μάθησης. Ένας κεντρικός εξυπηρετητής λαμβάνει τα βάρη από όλες τις συσκευές και, μέσω του συνδυασμού τους, δημιουργεί ένα κοινό, πιο ισχυρό μοντέλο. Στη συνέχεια, τα νέα συνδυασμένα βάρη αποστέλλονται πίσω σε όλες τις συσκευές που συμμετέχουν, ώστε να επικαιροποιήσουν το τοπικό τους μοντέλο. Περαιτέρω ανάλυση της μεθόδου, συμπεριλαμβανομένων περισσότερων περιπτώσεων και παραδειγμάτων, παρέχεται στο [4]. Μία ενδεικτική απεικόνιση της παραπάνω διαδικασίας παρουσιάζεται στο σχήμα 2.3.



Σχήμα 2.3: Διάγραμμα απεικόνισης της διαδικασίας της ομοσπονδιακής μαθησης

Ένα από τα βασικά πλεονεκτήματα της ομοσπονδιακής μάθησης είναι ότι οι συσκευές αποστέλλουν στον κεντρικό διαχειριστή (server) μόνο τα βάρη των μοντέλων τους και όχι τα δεδομένα που συλλέγουν. Με αυτόν τον τρόπο διασφαλίζεται η ιδιωτικότητα των δεδομένων, καθώς αυτά δεν διακινούνται στο δίκτυο και δεν υπάρχει κίνδυνος να υποκλαπούν. Ένα ακόμη σημαντικό πλεονέκτημα είναι ότι το τελικό μοντέλο που δημιουργείται σε κάθε γύρο εκπαίδευσης ενσωματώνει πληροφορίες από όλες τις συσκευές, και κατά συνέπεια από όλα τα περιβάλλοντα στα οποία αυτές βρίσκονται. Έτσι, το μοντέλο είναι συνολικά καλύτερα εκπαιδευμένο και καλύπτει περισσότερες περιπτώσεις χρήσης και παραλλαγές συνθηκών.

2.6 Εναλλακτικές μέθοδοι για ομοσπονδιακή μάθηση

Πέρα από τους παραπάνω τρόπους που έχουν αναφερθεί για την βελτίωση των μοντέλων μηχανικής μάθησης σε ενσωματωμένα συστήματα υπάρχουν άλλες μέθοδοι όπως είναι το ViFLa και το SISA, που στηρίζονται αρκετά στην έννοια της ομοσπονδιακής μάθησης. Περισσότερες πληροφορίες για αυτές τις τεχνικές αναφέρονται παρακάτω.

2.6.1 ViFLa: Virtual Federated Learning Approach

Μία από τις μεθόδους που μπορούν να χρησιμοποιηθούν για τη βελτίωση ενός μοντέλου μηχανικής μάθησης (ML) είναι το *machine unlearning*. Σε αντίθεση με την κλασική επανεκπαίδευση του μοντέλου με νέα δεδομένα, το machine unlearning στοχεύει στην αφαίρεση των δειγμάτων που αποκλίνουν σημαντικά από την πραγματικότητα, ώστε να μην επηρεάζουν αρνητικά την τελική πρόβλεψη του μοντέλου.

Μια σύγχρονη προσέγγιση πάνω σε αυτή τη διαδικασία, η οποία περιγράφεται και υλοποιείται στο [5], είναι η **ViFLa (Virtual Federated Learning Approach)**.

Στην προσέγγιση αυτή, όλα τα δεδομένα βρίσκονται συγκεντρωμένα σε έναν κεντρικό εξυπηρετητή (server) και ομαδοποιούνται με βάση την εκτιμώμενη πιθανότητα απεκμάθησης. Κάθε ομάδα δεδομένων θεωρείται ως ένας *εικονικός πελάτης* (virtual client) στο πλαίσιο της ομοσπονδιακής μάθησης (federated learning). Ωστόσο, σε αντίθεση με την παραδοσιακή ομοσπονδιακή μάθηση, όλοι οι εικονικοί πελάτες βρίσκονται στο ίδιο φυσικό μηχάνημα, επομένως δεν απαιτείται επικοινωνία μέσω δικτύου.

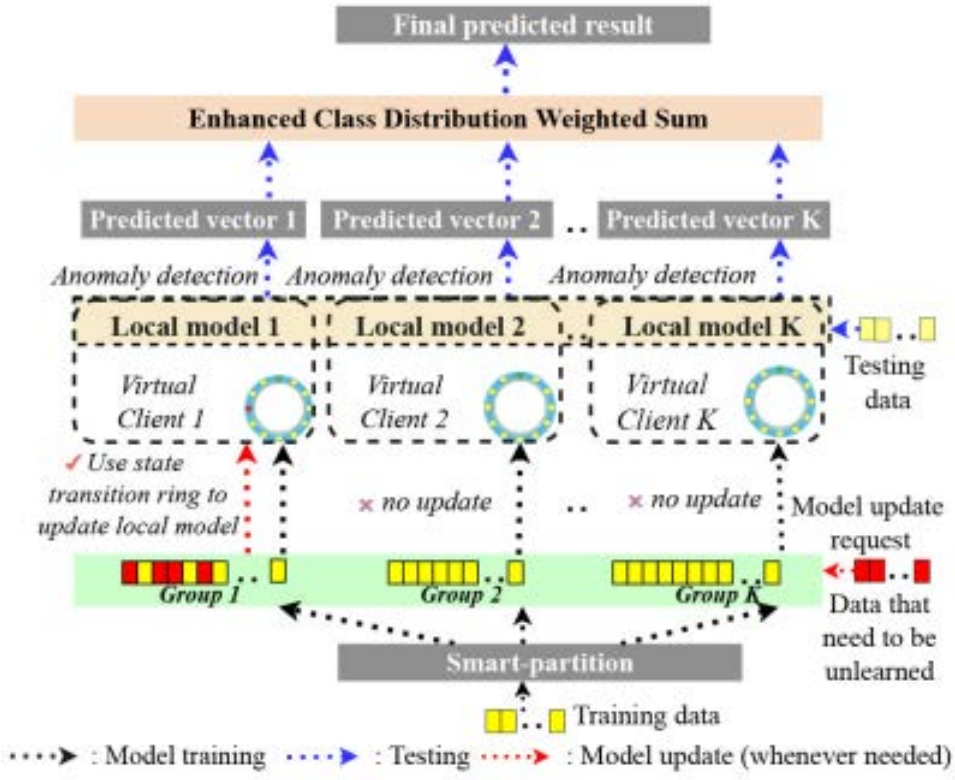
Η εκτιμώμενη πιθανότητα για απεκμάθηση υπολογίζεται μέσω της διαδικασίας έξυπνου κατακερματισμού (*smart partitioning*), η οποία χρησιμοποιεί τον ταξινομητή *Naive Bayes* για να υπολογίσει το **βαθμό πεποίθησης** (*belief*) κάθε δείγματος, δηλαδή την πιθανότητα ένα δείγμα να ανήκει σε μία συγκεκριμένη κλάση (αυτή με το μεγαλύτερο ποσοστό θεωρείται ότι είναι η σωστή). Βάσει αυτών των τιμών, τα δεδομένα κατανέμονται στους αντίστοιχους εικονικούς πελάτες.

Για την αντιμετώπιση του προβλήματος της μη ανεξάρτητης και ομοιόμορφης κατανομής των δεδομένων (*non-iid*), χρησιμοποιείται μια προσαρμοστική μέθοδος συγχώνευσης των τοπικών μοντέλων, η *Enhanced Class Distribution Weighted Sum* (ECDWS), η οποία λαμβάνει υπόψη την κατανομή των κλάσεων κατά την ομαδοποίηση.

Τέλος, σε κάθε γύρο βελτίωσης του μοντέλου, αφαιρούνται σταδιακά οι πληροφορίες που

κρίνονται λανθασμένες ή ανεπιθύμητες. Με αυτόν τον τρόπο επιτυγχάνεται η πλήρης απομάκρυνση της γνώσης που το μοντέλο πρέπει να «ξεχάσει», βελτιώνοντας έτσι την ακρίβεια και την προσαρμοστικότητά του.

Μια ενδεικτική εικόνα της διαδικασίας είναι η 2.4, η οποία προήλθε από το [5].



Σχήμα 2.4: Δομή του ViFLa

2.6.2 SISA: Sharded, Isolated, Sliced, and Aggregated Training

Το **SISA** είναι ακρωνύμιο του *Sharded, Isolated, Sliced, and Aggregated* training. Πρόκειται για μια τεχνική που επιτρέπει την αποτελεσματική διαγραφή συγκεκριμένων δειγμάτων από ένα εκπαιδευμένο μοντέλο, χωρίς να απαιτείται πλήρης επανεκπαίδευση από την αρχή. Για τον λόγο αυτό, εντάσσεται στη γενικότερη κατηγορία του *model unlearning*.

Κάθε λέξη στην ονομασία του SISA αντικατοπτρίζει και ένα βασικό χαρακτηριστικό της μεθόδου. Πιο αναλυτικά:

- **Sharded (Κατακερματισμένο)**: Το σύνολο των δεδομένων χωρίζεται σε ανεξάρτητα υποσύνολα (*shards*). Κάθε υποσύνολο εκπαιδεύει ένα ξεχωριστό μοντέλο ή ένα επιμέρους τμήμα του συνολικού μοντέλου.

- **Isolated (Απομονωμένο)**: Τα υποσύνολα είναι απολύτως απομονωμένα μεταξύ τους, γεγονός που επιτρέπει τον εντοπισμό του ποια δεδομένα επηρέασαν ποιο τμήμα του μοντέλου.

- **Sliced (Τμηματική εκπαίδευση)**: Κάθε υποσύνολο εκπαιδεύεται σε διαδοχικά στάδια εκπαίδευσης (*slices*), π.χ., ανά εποχές ή φάσεις. Αυτό επιτρέπει καλύτερο έλεγχο στην επίδραση κάθε δείγματος.

- **Aggregated (Συγχώνευση αποτελεσμάτων)**: Τα αποτελέσματα από όλα τα υποσύνολα και τα τμήματα εκπαίδευσης συγχωνεύονται για την παραγωγή του τελικού μοντέλου.

Αν κάποιο δείγμα πρέπει να «ξεχαστεί», εντοπίζεται πρώτα το υποσύνολο στον οποίο ανήκει. Στη συνέχεια, γίνεται επανεκπαίδευση μόνο αυτού, αποκλείοντας το δείγμα. Τέλος, όλα τα υπόλοιπα που έχουν μείνει συγχωνεύονται ξανά στο συνολικό μοντέλο. Με αυτόν τον τρόπο, αποφεύγεται η πλήρης επανεκπαίδευση, ενώ διασφαλίζεται η ακριβής αφαίρεση της επίδρασης του συγκεκριμένου δείγματος από το μοντέλο.

Αναλυτικότερα, η περιγραφή και ο τρόπος υλοποίησης αυτής της μεθόδου βρίσκεται στο [6]

Κεφάλαιο 3

Εφαρμογή, Ενσωματωμένη Συσκευή και Είδος Πρόβλεψης

3.1 Εισαγωγή

Για να μπορέσει να εξεταστεί η διαδικασία της επανεκπαίδευσης με τις τρεις τεχνικές (*IL-SBR*, *IL-ODR* και *FL-SS*) που αναφέρθηκαν στο εισαγωγικό Κεφάλαιο 1 και επεξηγούνται αναλυτικά στα επόμενα κεφάλαια, έχει σχεδιαστεί μία συσκευή πάνω στην οποία μπορούν να εφαρμοστούν. Η λειτουργία της είναι να διαβάζει τιμές, ανά λεπτό, από τον αισθητήρα ήχου που είναι συνδεδεμένος σε αυτήν και, χρησιμοποιώντας τις προηγούμενες πιο πρόσφατες τιμές, να επιχειρεί την πρόβλεψη της έντασης για το επόμενο λεπτό. Περισσότερες πληροφορίες σχετικά με τη συσκευή, τα επιμέρους στοιχεία της, καθώς και τη δομή του μοντέλου μηχανικής μάθησης παρουσιάζονται στη συνέχεια του κεφαλαίου.

3.2 Μικροελεγκτές και ESP32

Ο μικροελεγκτής (microcontroller) είναι ένας τύπος επεξεργαστή, ουσιαστικά μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα. Η ευελιξία ανάπτυξης διαφο-

ρετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα, η οποία δεν είναι εξειδικευμένη.

Στην παρούσα εργασία, ο μικροελεγκτής που θα χρησιμοποιηθεί είναι το ESP32. Ανήκει στην κατηγορία των συσκευών χαμηλού κόστους και χαμηλής κατανάλωσης, καθώς είναι αρκετά οικονομικός και με πολύ μικρές απαιτήσεις σε ενέργεια. Διαθέτει 48 κεφαλές, από τις οποίες μπορούν να χρησιμοποιηθούν για προγραμματισμό περίπου 25 έως 30. Αυτές χωρίζονται σε διάφορες κατηγορίες, ανάλογα με τη λειτουργικότητά τους. Συγκεκριμένα υπάρχουν: γενικού σκοπού Εισόδου/Εξόδου (GPIO), μετατροπείς Αναλογικού σε Ψηφιακό (ADC), μετατροπείς Ψηφιακού σε Αναλογικό (DAC), αισθητήρες αφής (Capacitive sensing GPIOs), 3 διεπαφές SPI, 3 διεπαφές UART, 2 διεπαφές I^2C και κανάλια εξόδου μορφής PWM. Ακόμη, το ESP32 ξεχωρίζει από πολλούς μικροελεγκτές, καθώς διαθέτει ενσωματωμένα WiFi και Bluetooth, τα οποία χρησιμοποιούνται σε πλήθος εφαρμογών. Στις κεφαλές του, μπορεί να συνδεθεί μεγάλη ποικιλία αισθητήρων, όπως θερμοκρασίας, υγρασίας ή επιταχυνσιόμετρα (accelerometers). Όλα αυτά, καθώς και επιπλέον πληροφορίες, αναλύονται στο documentation που παρατίθεται στο [7].

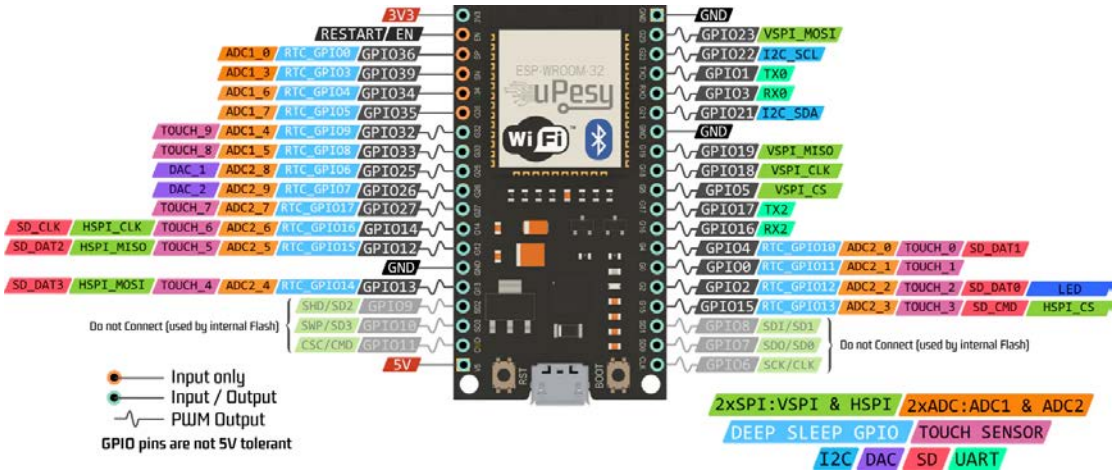
Το ESP32 δεν εκτελεί ένα παραδοσιακό λειτουργικό σύστημα όπως σε υπολογιστές ή κινητές συσκευές, αλλά υποστηρίζει αρκετά ελαφριά λειτουργικά συστήματα και πλαίσια που έχουν σχεδιαστεί ειδικά για ενσωματωμένα συστήματα. Ένα από αυτά είναι το FreeRTOS, που χαρακτηρίζεται ως ένα δημοφιλές, ανοιχτού κώδικα λειτουργικό σύστημα πραγματικού χρόνου (RTOS) για ενσωματωμένα συστήματα, σχεδιασμένο να προσφέρει ένα απλό και αποδοτικό περιβάλλον για την ανάπτυξη εφαρμογών που απαιτούν ακριβή χρονισμό και διαχείριση εργασιών.

Η εικόνα 3.1 αναπαριστά ένα ESP32 και περιέχει μια γενική περιγραφή κάθε κεφαλή του (Pinout image).

3.3 Εξαρτήματα και συνδεσμολογία

Για την υλοποίηση και την εφαρμογή των μεθόδων που αναφέρονται στην παρούσα διπλωματική, έχει κατασκευαστεί μία συσκευή, η οποία επιγραμματικά περιλαμβάνει ένα ESP32, έναν αισθητήρα ήχου και 2 διοδικά φώτα εκπομπής.

Συγκεκριμένα, το ESP32 αποτελεί τον πυρήνα της συσκευής, καθώς σε αυτό συνδέονται



Σχήμα 3.1: ESP32 PinOut

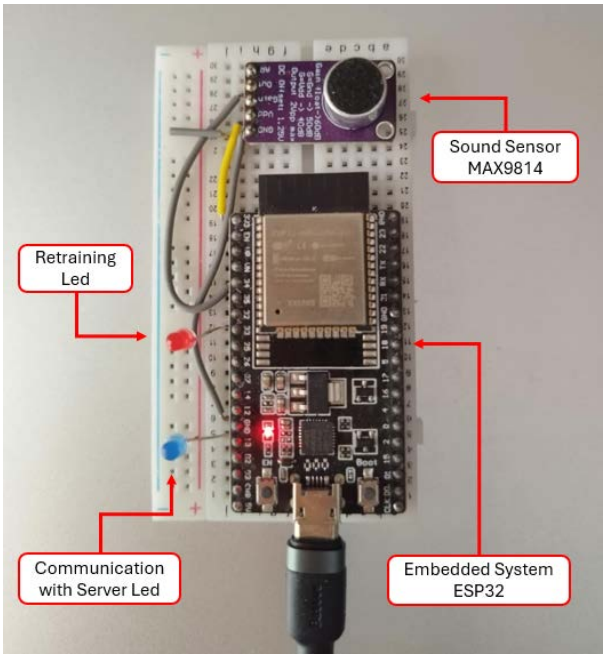
όλα τα υπόλοιπα εξαρτήματα. Επίσης όλα τα μοντέλα μηχανικής μάθησης που υλοποιούνται σε κάθε στάδιο, εκτελούνται πάνω σε αυτό. Ακόμη, η επικοινωνία με τον εξυπηρετητή γίνεται μέσω του μικροελεγκτή.

Τα δεδομένα που χρησιμοποιούνται από τα ML μοντέλα για την εκπαίδευση και για τις προβλέψεις τους, προέρχονται από έναν αισθητήρα ήχου. Συγκεκριμένα έχει επιλεγεί ο MAX9814, ο οποίος επιστρέφει τιμές απο -1500 DB εως και 1500DB (που παραμετροποιείται ανάλογα με την τιμή του Gain Pin). Συνδέεται στο esp32 μέσω μίας αναλογικής κεφαλής, για την μεταφορά των μετρήσεων και μέσω της 3.3 Volt ακίδας για την τροφοδοσία. Η επιλογή αυτου του τύπου αισθητήρα έγινε διότι οι τιμές του αλλάξουν συνεχώς καθόλη την διάρκεια της λειτουργείας του, σε σύγκριση με έναν αισθητήρα θερμοκρασίας (όπου κατά την διάρκεια της ημέρας οι μεταβολές είναι ελάχιστες), και έτσι θα είναι πιο εύκολο και γρήγορο να φανούν αλλαγές και η αποδοτικότητα του μοντέλου.

Ακόμα χρησιμοποιούνται και δύο διοδικά φώτα εκπομπής τα οποία συνδέονται σε μία αναλογική κεφαλή του ESP32 και στην γείωση. Αυτά ενημερώνουν τον χρήστη για το στάδιο στο οποίο βρίσκεται ο μικροελεγκτής. Συγκεκριμένα, το κόκκινο σηματοδοτεί την έναρξη της επανεκπαίδευσης ή της ομοσπονδιακής μάθησης, και η διάρκεια του είναι λιγότερο από ένα λεπτό. Η διαδικασία αυτή δεν συμβαίνει κάθε μέρα, αλλά όποτε είναι απαραίτητο. Στην συγκεκριμένη εφαρμογή έχει οριστεί να γίνεται στις 12 το βράδυ (στην λήξη της ημέρας). Το δεύτερο φως (που το χρώμα του για κάθε συσκευή είναι διαφορετικό) ανάβει κάθε ένα λεπτό, καθώς κάθε τότε αποστέλλεται ένας παλμός στον εξυπηρετητή, ώστε να ενημερώνεται για την κατάσταση της συσκευής. Με αυτόν τον τρόπο ο χρηστής μπορεί να καταλάβει ότι συνεχίζει να λειτουργεί, είτε παρατηρώντας το φως (αν βρίσκεται κοντά της), είτε μέσω της

παρακολούθησης των καταγραφών (logs) του εξυπηρετητή.

Στην Εικόνα 3.2 φαίνεται η πραγματική συσκευή που χρησιμοποιείται και επισημαίνονται τα σημαντικά στοιχεία της.



Σχήμα 3.2: Φωτογραφία του πραγματικού board

3.4 Είδος πρόβλεψης και μοντέλο μηχανικής μάθησης

Βασικό κριτήριο για την επιλογή της μορφής και του είδους πρόβλεψης του ML μοντέλου που χρησιμοποιήθηκε ήταν η απλοϊκότητα του, καθώς θα έπρεπε να μπορεί να εκπαιδεύεται και να εκτελείται πάνω στον μικροελεγκτή (τουλάχιστον για την υλοποίηση μίας από τις τρεις μεθόδους που έχουν αναλυθεί). Επίσης, θα έπρεπε να είναι δυνατή η άμεση αξιολόγηση της πρόβλεψης σε σωστή ή λάθος, έτσι ώστε στο τέλος κάθε κύκλου, όπου θα πρέπει να ελεγχτεί αν χρειάζεται εκκίνηση της επανεκπαίδευσης ή της ομοσπονδιακής μαθησης, να παίρνεται αυτή η απόφαση από το αν ο αριθμός των λαθών έχει ξεπεράσει κάποιο καθορισμένο όριο.

Το μοντέλο που εκτελείται στις συσκευές παίρνει τιμές από τον αισθητήρα ήχου που είναι τοποθετημένος. Η είσοδος του είναι οι τιμές του ήχου στα τελευταία πέντε λεπτά και δίνονται ως συνδυασμός *(χρόνος, τιμή ήχου)*. Η έξοδος του μοντέλου είναι η τιμή που θα έχει ο ήχος στο επόμενο λεπτό, και όταν αυτό το λεπτό περάσει θα είναι δυνατή η αξιολόγηση της πρόβλεψης ως σωστή ή λανθασμένη. Η επιλογή που έγινε για τη χρήση των τελευταίων

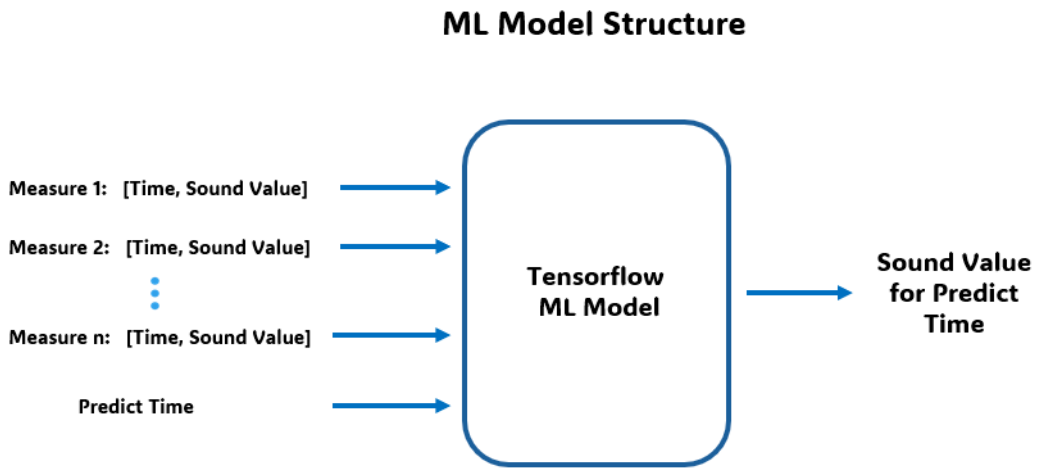
πέντε λεπτών ως είσοδο στο μοντέλο αποτελεί μια εσωτερική παράμετρο της εφαρμογής και μπορεί να αλλάξει ανάλογα με τον σκοπό. Γενικά, όσο περισσότερα είναι τα προηγούμενα λεπτά που χρησιμοποιούνται, τόσο πιο εύκολο είναι για το μοντέλο να κατανοήσει τη ροή των δεδομένων και να πραγματοποιήσει πιο ακριβή πρόβλεψη.

Ως προς τα δομικά του χαρακτηριστικά, αποτελείται από: 11 χαρακτηριστικά (*features*) ως είσοδο, 25 νευρώνες στο κρυφό επίπεδο (*hidden layer*) και 1 νευρώνα στην έξοδο. Ο τύπος δεδομένων που χρησιμοποιείται για την αναπαράσταση είναι `4B float` (δηλαδή η τυπική ακρίβεια που υποστηρίζει το `ESP32`). Το συνολικό μέγεθος του μοντέλου είναι 1,304 bytes, το οποίο προκύπτει ως εξής:

- **Input → Hidden**: $(11 + 1) \times 25 = 300$ floats
- **Hidden → Output**: $(25 + 1) \times 1 = 26$ floats
- **Σύνολο βαρών**: $300 + 26 = 326$ floats
- **Μέγεθος**: $326 \times 4 = 1{,}304$ bytes (≈ 1.3 KB)

Σημείωση: Το $+1$ αντιστοιχεί στη *σταθερά μετατόπισης (bias)* που διαθέτει κάθε νευρώνας του επόμενου *επιπέδου (layer)*.

Στην εικόνα 3.3 υπάρχει μία εικονική απεικόνιση της δομής του μοντέλου με τις εισόδους και την έξοδο που διαθέτει.



Σχήμα 3.3: Δομή του μοντέλου μηχανικής μάθησης που εκτελείται στην συσκευή

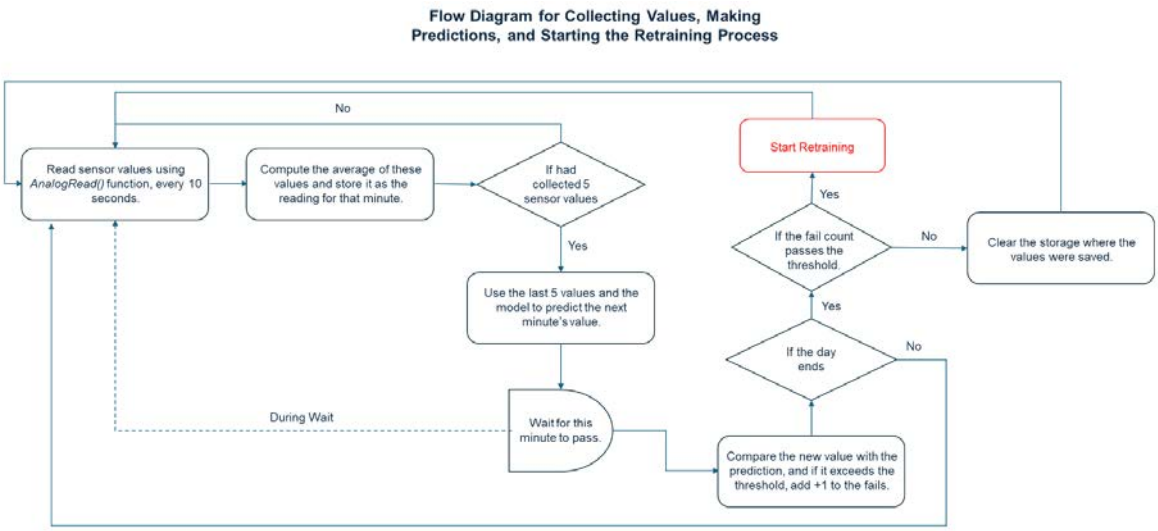
3.5 Συλλογή τιμών, προβλέψεις και ενεργοποίηση επανεκπαίδευσης

Ο τρόπος με τον οποίο χρησιμοποιείται το μοντέλο στη συσκευή, καθώς και η δομή του, είναι κοινοί για καθεμία από τις τρεις τεχνικές που θα αναλυθούν. Ωστόσο, η αρχικοποίησή του και η αποθήκευση των δομικών του στοιχείων από την πλευρά της συσκευής, αλλά και η δημιουργία του από τον εξυπηρετητή, διαφέρουν. Για τον λόγο αυτό, στα επόμενα κεφάλαια (4, 5 και 6) υπάρχει ξεχωριστό υποκεφάλαιο (για την συσκευή και τον εξυπηρετητή) στο οποίο αυτές οι διαφορές αναλύονται.

Αρχικά, κατά τη διάρκεια της ημέρας καταγράφονται όλες οι τιμές που συλλέγει ο αισθητήρας σε έναν πίνακα με τη μορφή *(λεπτό μέσα στην ημέρα, τιμή αισθητήρα)*. Για την ανάγνωση των τιμών χρησιμοποιείται η συνάρτηση `analogRead()` που παρέχεται στο ESP32, καθώς ο αισθητήρας είναι συνδεδεμένος στην αναλογική κεφαλή της συσκευής. Γενικά, αποθηκεύεται μία τιμή ήχου ανά λεπτό ωστόσο, προκειμένου η τιμή αυτή να προσεγγίζει καλύτερα ολόκληρο το λεπτό, πραγματοποιούνται πολλαπλές μετρήσεις καθ' όλη τη διάρκειά του. Η τελική τιμή που του αποδίδεται είναι ο μέσος όρος όλων των μετρήσεων που καταγράφηκαν μέσα σε αυτό.

Παράλληλα, για κάθε πέντε διαδοχικές τιμές ήχου, το μοντέλο προβλέπει την ένταση του ήχου στο επόμενο λεπτό. Όταν αυτό το λεπτό περάσει και είναι διαθέσιμη η πραγματική τιμή, συγκρίνεται με την προβλεπόμενη ώστε να διαπιστωθεί αν η πρόβλεψη ήταν επιτυχής. Για να θεωρείται σωστή, η απόκλιση μεταξύ προβλεπόμενης και πραγματικής τιμής δεν πρέπει να υπερβαίνει ένα προκαθορισμένο όριο. Σε περίπτωση λανθασμένης πρόβλεψης, αυτή καταγράφεται. Αν ο συνολικός αριθμός των λανθασμένων προβλέψεων υπερβεί ένα προκαθορισμένο κατώφλι (*threshold*), τότε ορίζεται ότι πρέπει να πραγματοποιηθεί επανεκπαίδευση του μοντέλου, σε προκαθορισμένη χρονική στιγμή (εχει οριστεί ότι αυτή είναι το τέλος της ημέρας). Αν η επανεκπαίδευση δεν είναι αναγκαία, τότε στο τέλος της ημέρας ο πίνακας καθαρίζεται και αρχίζει εκ νέου να γεμίζει με νέες τιμές για την επόμενη ημέρα.

Στο διάγραμμα ροής (Flow Diagram) 3.4 περιγράφεται η παραπάνω διαδικασία.



Σχήμα 3.4: Διάγραμμα ροής για τη συλλογή δεδομένων, τις προβλέψεις και την εκκίνηση της διαδικασίας retraining

3.6 Επικοινωνία με το νέφος

Για να είναι δυνατή η διαδικασία της επανεκπαίδευσης εκτός της συσκευής, καθώς και της ομοσπονδιακής μάθησης, θα πρέπει το ESP32 να επικοινωνεί με κάποιον εξυπηρετητή στο νέφος. Συγκεκριμένα μέσα από την διαδικτυακή πλατφόρμα ViMa GRNET έχει παραχωρηθεί μία εικονική μηχανή όπου θα αποτελέσει τον εξυπηρετητή που χρειάζεται στα πλαίσια αυτής της εργασίας. Συγκεκριμένα, διαθέτει δημόσια διεύθυνση IP έτσι ώστε να είναι προσπελάσιμος από τις συσκευές που θέλουν να επικοινωνήσουν *(με μόνη προϋπόθεση να είναι συνδεδεμένες στο διαδίκτυο)*. Από τεχνικά χαρακτηριστικά διαθέτει 2GB μνήμη, 20GB δίσκο και 2 CPUs.

Στον εξυπηρετητή αυτό εκτελούνται στο παρασκήνιο 2 python scripts, ένα για την διαδικασία της επανεκπαίδευσης και ένα για το FL. Οι συσκευές επικοινωνούν αποστέλλοντας αιτήματα (request) και, ανάλογα με το περιεχόμενό τους, ο εξυπηρετητής εκτελεί τις αντίστοιχες ενέργειες. Στη συνέχεια, αποστέλλει μηνύματα στις συσκευές ως απάντηση στα μηνύματα τους, και αυτό γιατί δεν γνωρίζει την δημόσια διεύθυνση IP τους για να στείλει ο ίδιος μήνυμα εξαρχής.

3.7 Τοποθέτηση συσκευής και εκκίνηση λειτουργίας

Η συσκευή είναι ιδιαίτερα εύκολη στην τοποθέτηση της καθώς το μόνο που απαιτείτε είναι να συνδεθεί σε παροχή ρεύματος μέσο ενός USB καλωδίου. Για να μπορέσει να στέλνει μηνύματα με τον εξυπηρετητή θα πρέπει να συνδεθεί στο δίκτυο. Έτσι χρειάζεται να διαθέτει τα διαπιστευτήρια του Wi-Fi, τα οποία θα βρίσκονται κάθε φόρα στον κώδικα του ESP32. Κατά την εκκίνηση το πρώτο πράγμα που θα κάνει θα είναι να συνδεθεί στο δίκτυο και στην συνεχεία θα αρχίσει κανονικά την λειτουργεία του.

Κατά την λειτουργία της, ο χρήστης θα βλέπει ανά ένα λεπτό να ανάβει το δεύτερο φως, το οποίο σημαίνει ότι η συσκευή στέλνει μήνυμα (παλμούς) στον εξυπηρετητή. Σε περίπτωση που δεν συμβαίνει αυτό τότε θα πρέπει να επανακεινηθεί πατώντας το reset κουμπί που βρίσκεται πάνω στο ESP32 κα θα ξεκινήσει να λειτουργεί από την αρχή.

3.8 Αποστολή παλμού από το ESP32

Για να μπορεί ο διαχειριστής του εξυπηρετητή να γνωρίζει την κατάσταση των συσκευών ανά πάσα στιγμή, έχει υλοποιηθεί η αποστολή ενός παλμού από την κάθε μία. Συγκεκριμένα, κάθε *ESP32* αποστέλλει ανά λεπτό ένα μήνυμα, το οποίο περιλαμβάνει πληροφορίες όπως:

- Τον συνολικό αριθμό ημερών λειτουργίας της συσκευής,
- Πόσες από αυτές τις ημέρες έχει πραγματοποιηθεί επανεκπαίδευση,
- Τον αριθμό των λανθασμένων προβλέψεων που έχουν καταγραφεί έως εκείνη τη στιγμή, για την τρέχουσα ημέρα,
- Ένα flag που δηλώνει αν θέλουν να συμμετέχουν στην διαδικασία του FL-SS (ισχύει μόνο για αυτήν την τεχνική)

Με την παραλαβή κάθε τέτοιου μηνύματος, ο εξυπηρετητής επιβεβαιώνει ότι η συσκευή βρίσκεται σε λειτουργία και παρακολουθεί την ομαλή λειτουργία της. Παράλληλα, συλλέγει γενικές πληροφορίες για την κατάστασή της.

3.9 Over-The-Air Update του κώδικα του ESP32 μέσω του εξυπηρετητή

Οι συσκευές που συμμετέχουν στην παρούσα εργασία ενδέχεται να βρίσκονται σε διαφορετικά γεωγραφικά σημεία και να εκτελούν το μοντέλο τους βάσει δεδομένων από ξεχωριστά περιβάλλοντα. Ωστόσο, είναι πιθανό να απαιτείται η τροποποίηση του λογισμικού τους χωρίς να υπάρχει φυσική ή άμεση πρόσβαση σε αυτές. Για τον λόγο αυτό, έχει υλοποιηθεί η δυνατότητα απομακρυσμένης ενημέρωσης του κώδικα (*remote code update*) μέσω του εξυπηρετητή.

Συγκεκριμένα, αυτός διατηρεί για κάθε συσκευή έναν ξεχωριστό φάκελο στον οποίο μπορεί να αποθηκευτεί η νέα έκδοση του κώδικα. Αν ο χρήστης επιθυμεί να ενημερώσει το λογισμικό μιας συσκευής, δημιουργεί και μεταγλωττίζει τον νέο κώδικα, παράγοντας το αντίστοιχο αρχείο `.bin`. Στη συνέχεια, το αρχείο αυτό τοποθετείται στον κατάλληλο φάκελο.

Η συσκευή, σε τακτά χρονικά διαστήματα (ανά ένα λεπτό), αποστέλλει μήνυμα στον εξυπηρετητή ελέγχοντας αν υπάρχει διαθέσιμη αναβάθμιση λογισμικού. Αν εντοπιστεί νέο αρχείο, ο εξυπηρετητής αποστέλλει το `.bin` αρχείο στη συσκευή και το διαγράφει αμέσως μετά. Από την πλευρά της, η συσκευή αποθηκεύει τον νέο κώδικα στην εσωτερική της μνήμη και προχωρά σε αυτόματη επανεκκίνηση (*restart*). Μετά την επανεκκίνηση, ξεκινά τη λειτουργία της με τη νέα έκδοση του λογισμικού.

3.10 Πλατφόρμες ανάπτυξης

Ακολουθεί η λίστα με όλα τα λογισμικά και τις πλατφόρμες που χρησιμοποιήθηκαν κατά τη διάρκεια της ανάπτυξης της παρούσας διπλωματικής εργασίας:

1. **Visual Studio Code**: Χρησιμοποιήθηκε για την ανάπτυξη του κώδικα τόσο για το ESP32 (σε γλώσσα C++) όσο και για τον διακομιστή (σε Python).

2. **PlatformIO**: Αποτελεί επέκταση του Visual Studio Code και χρησιμοποιήθηκε για τη μεταγλώττιση (compile) και τη μεταφορά (upload) του κώδικα στη συσκευή ESP32.

3. **ViMa (Virtual Machines)**: Πρόκειται για υπηρεσία του ΕΔΥΤΕ (GRNET), η οποία παρέχει εικονικές μηχανές με δημόσιες IP και τεχνικά χαρακτηριστικά που μπορούν

να επιλεγούν ανάλογα με τις ανάγκες. Στην παρούσα εργασία έγινε σχετικό αίτημα και παραχωρήθηκε μια εικονική μηχανή κατάλληλη για τις απαιτήσεις του έργου.

4. **WinSCP**: Χρησιμοποιήθηκε για την απομακρυσμένη σύνδεση με την εικονική μηχανή, διευκολύνοντας τη διαχείρισή της. Σε αντίθεση με ένα απλό τερματικό, παρέχει γραφικό περιβάλλον για πλοήγηση, δημιουργία και διαχείριση αρχείων. Επιπλέον, υποστηρίζει τη μεταφορά αρχείων μεταξύ του τοπικού συστήματος και της εικονικής μηχανής με απλό *σύρε και άφησε (drag-and-drop)*.

5. **Microsoft PowerPoint**: Χρησιμοποιήθηκε για τη δημιουργία διαγραμμάτων και σχημάτων που παρουσιάζονται στο παρόν έργο.

6. **Microsoft Excel**: Χρησιμοποιήθηκε κατά τη διάρκεια των πειραμάτων για την αποθήκευση των τιμών των μετρικών σε πίνακες και στη συνέχεια για τη δημιουργία των αντίστοιχων διαγραμμάτων σύγκρισης.

7. **LaTeX**: Χρησιμοποιήθηκε για τη σύνταξη του παρόντος τεχνικού κειμένου, προσφέροντας επαγγελματική μορφοποίηση και τυπογραφική ακρίβεια.

Κεφάλαιο 4

Ανεξάρτητη μάθηση με επανεκπαίδευση σε εξυπηρετητή (IL-SBR)

4.1 Εισαγωγή

Μία από τις πιο απλές υλοποιήσεις επανεκπαίδευσης μοντέλου είναι η μεταφορά της διαδικασίας εκτός του ενσωματωμένου συστήματος και η εκτέλεσή της σε έναν απομακρυσμένο εξυπηρετητή με κατάλληλη υπολογιστική ισχύ. Με αυτόν τον τρόπο, ο μικροελεγκτής ESP32 που χρησιμοποιείται στην εργασία μπορεί να εκτελεί οποιοδήποτε μοντέλο μηχανικής μάθησης, ανεξαρτήτως της πολυπλοκότητάς του, αρκεί το μοντέλο να έχει προηγουμένως μετατραπεί (από τον εξυπηρετητή) σε μορφή κατάλληλη για ενσωματωμένα συστήματα.

Στο πλαίσιο της παρούσας εργασίας, η μέθοδος που υλοποιήθηκε ονομάστηκε: **Ανεξάρτητη μάθηση με επανεκπαίδευση σε εξυπηρετητή** (*independent learning with server-based retraining*) ή αλλιώς εν συντομία **IL-SBR**. Ο όρος προκύπτει επειδή το *ESP32* πραγματοποιεί επανεκπαίδευση του μοντέλου του ανεξάρτητα, χωρίς τη χρήση δεδομένων που προέρχονται από άλλες συσκευές. Επίσης, η συνεισφορά του εξυπηρετητή είναι καθοριστική, καθώς συγκεντρώνει όλα τα δεδομένα, δημιουργεί το μοντέλο και το αποστέλλει στη συσκευή.

4.2 Γενική περιγραφή

Η διαδικασία που ακολουθείται επιγραμματικά έχει ως εξής: το *ESP32* ξεκινά με ένα αρχικό μοντέλο, το οποίο χρησιμοποιείται για να πραγματοποιεί προβλέψεις. Στην παρούσα

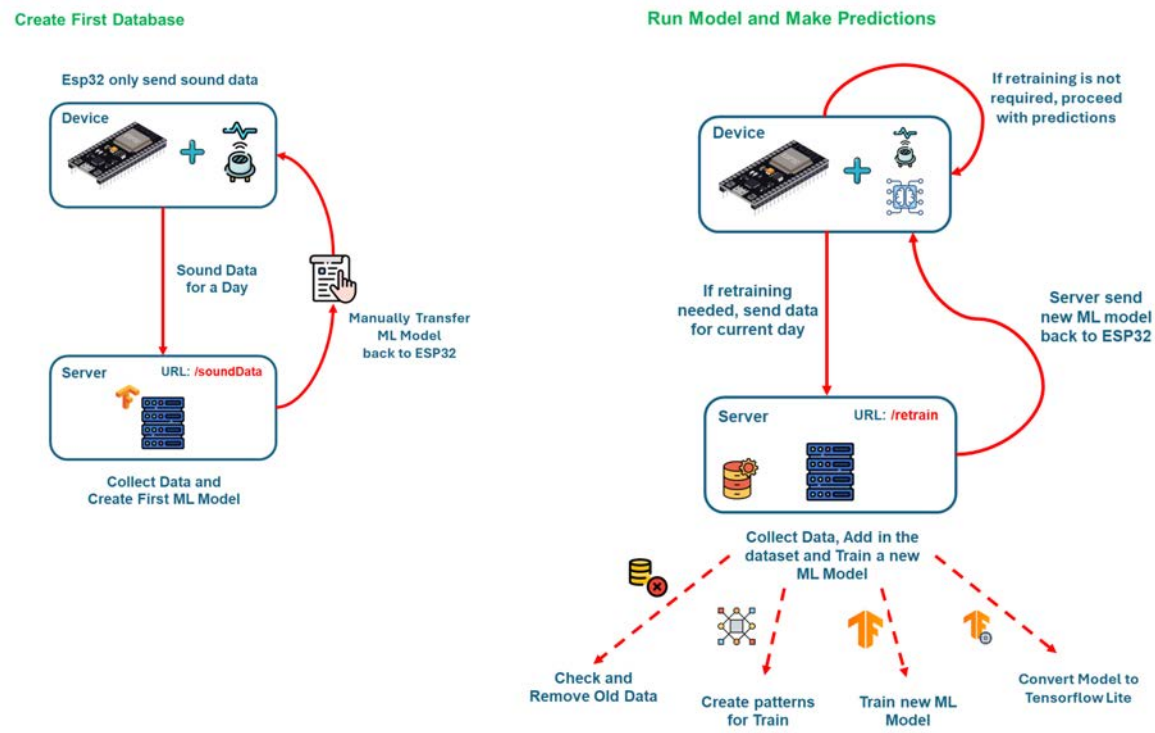
εργασία, το μοντέλο προβλέπει τη τιμή του ήχου για το επόμενο λεπτό, βασιζόμενο στις μετρήσεις των προηγούμενων πέντε λεπτών. Μια πρόβλεψη θεωρείται λανθασμένη όταν η προβλεπόμενη τιμή αποκλίνει από την πραγματική περισσότερο από ένα προκαθορισμένο όριο. Καθ' όλη τη διάρκεια της ημέρας, η συσκευή συλλέγει όλες τις τιμές της έντασης ήχου, ώστε — σε περίπτωση που απαιτηθεί επανεκπαίδευση — τα δεδομένα να είναι διαθέσιμα για την κατασκευή ενός νέου μοντέλου. Εάν, στο τέλος της ημέρας, ο αριθμός των λανθασμένων προβλέψεων ξεπερνά κάποιο όριο, τότε η συσκευή αποστέλλει όλα τα δεδομένα της ημέρας στον εξυπηρετητή. Αυτός συλλέγει, αποθηκεύει και ενσωματώνει αυτά τα δεδομένα (μαζί με ενδεχομένως προηγούμενα δεδομένα), ώστε να εκπαιδεύσει εκ νέου το μοντέλο. Αν κάποια από τα υπάρχοντα δεδομένα είναι αρκετά παλιά τότε ο εξυπηρετητής δεν τα συμπεριλαμβάνει στην εκπαίδευση. Παρόλα αυτά συνεχίζουν να βρίσκονται αποθηκευμένα για ιστορικούς λόγους. Μόλις ολοκληρωθεί η διαδικασία επανεκπαίδευσης, το νέο μοντέλο μετατρέπεται σε μορφή κατάλληλη για ενσωματωμένα συστήματα (*TinyML*) και αποστέλλεται πίσω στο *ESP32*, το οποίο προχωρά στην αντικατάσταση του παλαιού και συνεχίζει κανονικά τη λειτουργία του. Η ενσωμάτωση των παλαιότερων δεδομένων από τον εξυπηρετητή αποτελεί τον βασικό παράγοντα βελτίωσης της απόδοσης του μοντέλου.

Επίσης, η διαδικασία επανεκπαίδευσης μπορεί να χαρακτηριστεί ως ασύγχρονη, καθώς, όσο η συσκευή περιμένει από τον εξυπηρετητή την απάντηση που περιέχει το νέο μοντέλο, λειτουργεί κανονικά, συλλέγει δεδομένα και εκτελεί προβλέψεις με το προηγούμενο μοντέλο. Αυτό επιτυγχάνεται με τη χρήση ξεχωριστού νήματος (*thread*) που εκτελεί την διαδικασία της επανεκπαίδευσης.

Στην εικόνα 4.1 φαίνεται το συνολικό workflow που εφαρμόζεται κατά τη διάρκεια αυτής της διαδικασίας.

4.3 Από την πλευρά της συσκευής

Εν συντομία, η συσκευή συλλέγει δεδομένα από τον αισθητήρα, πραγματοποιεί προβλέψεις και ζητά επανεκπαίδευση όταν το κρίνει αναγκαίο. Σε αυτό το υποκεφάλαιο παρουσιάζονται αναλυτικά όλες οι επιμέρους λειτουργίες της. Ο συνδυασμός τους συνθέτει τη συνολική λειτουργικότητα της συσκευής, η οποία απεικονίζεται στο διάγραμμα ροής (*Flow Diagram*) 4.2 και αναλύεται περαιτέρω στη συνέχεια.



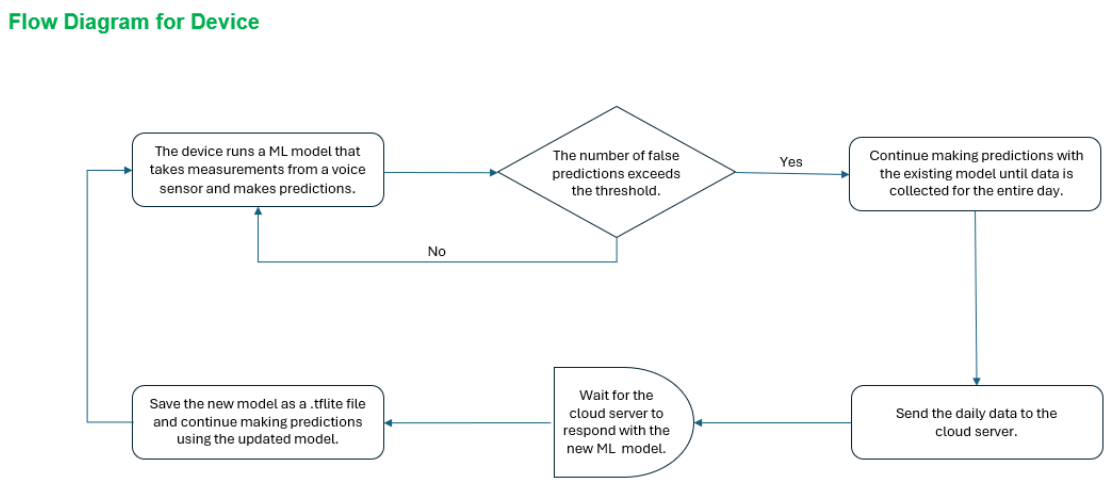
Σχήμα 4.1: Αναπαράσταση της λειτουργίας του IL-SBR με τη μορφή διαγράμματος

4.3.1 Αρχικοποίηση και διαχείριση του μοντέλου στη συσκευή

Για να μπορέσει το *ESP32* να εκτελέσει ένα μοντέλο μηχανικής μάθησης, θα πρέπει το μοντέλο να έχει μετατραπεί σε μορφή *TinyML*. Η μετατροπή αυτή πραγματοποιείται με τη χρήση των κατάλληλων συναρτήσεων που παρέχει το *TensorFlow*, μέσω του *TensorFlow Lite Converter*. Την διαδικασία αυτή, εκτελεί ο εξυπηρετητής κάθε φορά που επανεκπαιδεύει το μοντέλο.

Αποτέλεσμα αυτής της μετατροπής είναι η παραγωγή ενός αρχείου με κατάληξη `.tflite`, το οποίο αποτελεί την κωδικοποιημένη εκδοχή του μοντέλου και περιέχει πληροφορίες για τη δομή και τις παραμέτρους του. Επιπλέον, μπορεί να παραχθεί και ένα αρχείο `.cc`, το οποίο περιλαμβάνει το μοντέλο σε δυαδική μορφή (*binary representation*), έτοιμο προς ενσωμάτωση στον πηγαίο κώδικα του *ESP32*.

Ωστόσο, έχει επιλεγεί να αποστέλλεται το αρχείο *.tflite* αντί για το *.cc*, καθώς μπορεί να ενσωματωθεί πιο εύκολα στο εσωτερικό της συσκευής και να αντικαταστήσει το παλαιότερο αρχείο, ώστε να γίνεται αρχικοποίηση από αυτό. Συγκεκριμένα η χρήση του *.tflite* είναι τεχνικά και πρακτικά πιο κατάλληλη, καθώς πρόκειται για δυαδικό αρχείο που μπορεί να φορτωθεί δυναμικά στη μνήμη (RAM ή flash) της συσκευής χωρίς να απαιτείται εκ νέου μεταγλώττιση του λογισμικού συστήματος, σε αντίθεση με το *.cc*, το οποίο περιέχει το μοντέλο



Σχήμα 4.2: Διάγραμμα ροής της συσκευής κατά τη διαδικασία του IL-SBR

ως πίνακα από bytes και απαιτεί μεταγλώττιση και επανεγγραφή του λογισμικού κάθε φορά που αλλάζει.

Το *ESP32* χρησιμοποιεί αυτό το αρχείο για να δημιουργήσει το μοντέλο και να το εκτελέσει. Με τη βοήθεια της βιβλιοθήκης [8], παρέχονται όλες οι απαραίτητες συναρτήσεις για την αρχικοποίηση του, την εισαγωγή δεδομένων στην είσοδο και την εξαγωγή των προβλέψεων από την έξοδο.

Κατά την εκκίνηση της συσκευής υπάρχει ένα αρχικά εκπαιδευμένο μοντέλο, το οποίο χρησιμοποιείται για να πραγματοποιηθούν οι πρώτες προβλέψεις. Αυτό αποθηκεύεται κατευθείαν στο σύστημα αρχείων, ώστε να μπορεί από εκεί να το διαβάζει και να το επεξεργάζεται. Από εκεί και έπειτα, κάθε φορά που πραγματοποιείται επανεκπαίδευση, το νέο μοντέλο αντικαθιστά το προηγούμενο και η διαδικασία συνεχίζεται κανονικά με βάση το επικαιροποιημένο μοντέλο.

4.3.2 Εκκίνηση διαδικασίας εκπαίδευσης και αποστολή δεδομένων

Η διαδικασία της επανεκπαίδευσης ενεργοποιείται όταν ο αριθμός των λανθασμένων προβλέψεων υπερβεί ένα προκαθορισμένο κατώφλι (threshold). Όταν συμβεί αυτό, στο τέλος της ημέρας η συσκευή αποστέλλει όλα τα δεδομένα που έχει συλλέξει στον εξυπηρετητή, έτσι ώστε να περιλαμβάνονται στο νέο σύνολο εκπαίδευσης όλα τα ημερήσια δεδομένα. Η αποστολή των δεδομένων πραγματοποιείται σε μορφή JSON, μέσω του πρωτοκόλλου HTTP, και στη συνέχεια η συσκευή αναμένει την απάντηση από τον εξυπηρετητή. Κατά τη διάρκεια

της αναμονής, συνεχίζει κανονικά τη λειτουργία της και εκτελεί προβλέψεις χρησιμοποιώντας το προηγούμενο μοντέλο. Αυτό είναι εφικτό, καθώς η διαδικασία της επανεκπαίδευσης εκτελείται από διαφορετικό νήμα (*thread*). Το χρονικό διάστημα αναμονής επηρεάζεται από τον χρόνο που απαιτείται για την εκ νέου εκπαίδευση του μοντέλου από τον εξυπηρετητή, ο οποίος με τη σειρά του επηρεάζεται από την πολυπλοκότητα του μοντέλου. Γι' αυτό, αναμένεται κάθε νέος κύκλος επανεκπαίδευσης να γίνεται ολοένα και πιο χρονοβόρος, καθώς τα δεδομένα που θα χρησιμοποιούνται θα αυξάνονται διαρκώς.

4.3.3 Ανανέωση του μοντέλου

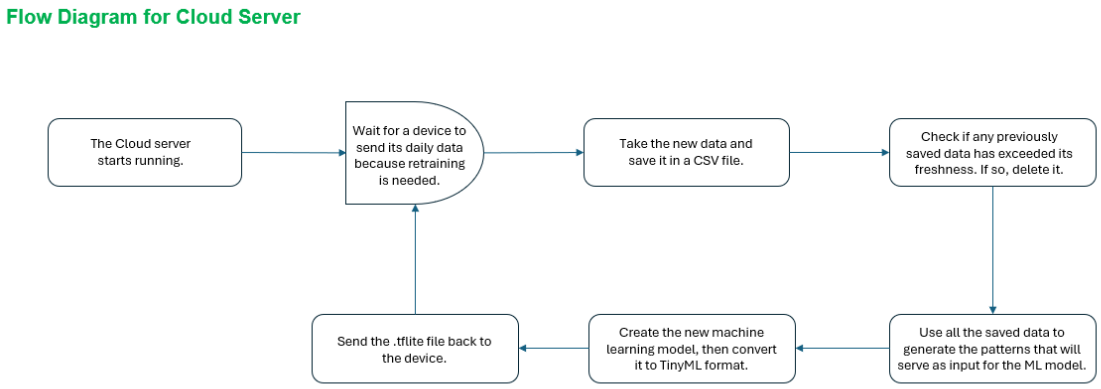
Όταν τελικά φτάσει η απάντηση από τον εξυπηρετητή, αυτή περιλαμβάνει όλα τα δεδομένα του αρχείου .tflite που έχει παραχθεί κατά την εκπαίδευση και τη μετατροπή του μοντέλου σε μορφή συμβατή με το TensorFlow Lite. Το ESP32, στη συνέχεια, αντικαθιστά το αντίστοιχο αρχείο του μοντέλου που είναι αποθηκευμένο στην εσωτερική του μνήμη και επαναορίζει τη δομή του μοντέλου. Από εκείνο το σημείο και έπειτα, οι προβλέψεις πραγματοποιούνται χρησιμοποιώντας το νέο μοντέλο.

4.4 Από την πλευρά του εξυπηρετητή

Ο εξυπηρετητή νέφους που χρησιμοποιείται βασίζεται σε μία *εικονική μηχανή (virtual machine)* που έχει παραχωρηθεί από το *VIMA του GRNet*. Έχει ως κύρια αρμοδιότητα τη συλλογή των *trace δεδομένων* κάθε συσκευής, τη δημιουργία ενός μοντέλου μηχανικής μάθησης, χρησιμοποιώντας τα, και την αποστολή του πίσω στη συσκευή. Στο διάγραμμα 4.3 φαίνεται η ροή λειτουργίας του.

4.4.1 Συλλογή δεδομένων από τα ESP32

Ο εξυπηρετητής βρίσκεται μόνιμα σε λειτουργία σε κατάσταση αναμονής (standby mode), καθώς περιμένει μηνύματα επανεκπαίδευσης από τις συσκευές. Τα μηνύματα αυτά αποστέλλονται μία φορά στο τέλος κάθε ημέρας από κάθε συσκευή, εφόσον πληρούνται οι προϋποθέσεις για την εκκίνηση της διαδικασίας. Για κάθε συσκευή που επικοινωνεί μαζί του, ο εξυπηρετητής δημιουργεί έναν ξεχωριστό φάκελο, στον οποίο αποθηκεύει τις πληροφορίες που του αποστέλλονται. Όταν λαμβάνει ένα μήνυμα επανεκπαίδευσης, εξάγει τα δεδομένα



Σχήμα 4.3: Διάγραμμα ροής του εξυπηρετητή κατά τη διαδικασία του IL-SBR

που περιλαμβάνει και τα καταχωρεί σε ένα αρχείο .csv μέσα στον αντίστοιχο φάκελο. Με αυτόν τον τρόπο δημιουργείται σταδιακά ένα ιστορικό δεδομένων για κάθε συσκευή, το οποίο αξιοποιείται κατά την εκπαίδευση ώστε να βελτιώνεται η απόδοση του μοντέλου σταδιακά καθώς αυξάνονται όλο και περισσότερο τα δεδομένα. Ωστόσο, εάν κάποια από τα δεδομένα είναι αρκετά παλιά, παραλείπονται από τη διαδικασία της εκπαίδευσης, αλλά διατηρούνται ως αρχείο για ιστορικούς λόγους.

4.4.2 Δημιουργία της βάσης και των συνδυασμών εισόδου/εξόδου

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, τα δεδομένα που αποστέλλει κάθε συσκευή αποθηκεύονται σε αρχεία .csv και χρησιμοποιούνται για τη δημιουργία του νέου μοντέλου. Η αξιοποίησή τους γίνεται μέσω της παραγωγής όλων των δυνατών συνδυασμών εισόδων και εξόδων (κάθε πέντε συνεχόμενες τιμές μπορούν να αποτελούν την είσοδο, ενώ η έκτη θα αντιστοιχεί στην έξοδο αυτής), οι οποίοι στη συνέχεια χρησιμοποιούνται για τη δημιουργία των συνόλων εκπαίδευσης (training dataset) και αξιολόγησης (test dataset), στο πλαίσιο της διαδικασίας επανεκπαίδευσης του μοντέλου.

4.4.3 Εκίνηση διαδικασίας και εκπαίδευση μοντέλου

Μόλις ο εξυπηρετητής λάβει το αίτημα `/retraining` από μία συσκευή και αποθηκεύσει τα δεδομένα που περιέχονται σε αυτό, ξεκινά η διαδικασία εκπαίδευσης ενός νέου μοντέλου μηχανικής μάθησης.

Αρχικά, συγκεντρώνονται τόσο τα νέα όσο και τα παλαιότερα δεδομένα που έχουν κατα-

γραφεί στον φάκελο της συγκεκριμένης συσκευής. Από αυτά τα δεδομένα δημιουργούνται όλοι οι δυνατοί συνδυασμοί εισόδων-εξόδων (*patterns*). Τα δεδομένα στη συνέχεια χωρίζονται σε δύο σύνολα: σύνολο εκπαίδευσης και σύνολο ελέγχου, με αναλογία 80% και 20% αντίστοιχα.

Η εκπαίδευση γίνεται χρησιμοποιώντας τη βιβλιοθήκη *Keras* του *TensorFlow*, με στόχο την κατασκευή ενός νέου νευρωνικού μοντέλου. Το μοντέλο είναι τύπου παλινδρόμησης (*regression*), καθώς προβλέπει μια συνεχή αριθμητική τιμή βάσει των ιστορικών δεδομένων. Η απόδοση του μοντέλου αξιολογείται μέσω της μετρικής της *Ρίζας του Μέσου Τετραγωνικού Σφάλματος (RMSE)*. Στόχος είναι κάθε φορά που πραγματοποιείται επανεκπαίδευση, η τιμή του RMSE να μειώνεται, γεγονός που υποδηλώνει βελτίωση της ακρίβειας και της γενικευσιμότητας του μοντέλου.

4.4.4 Προετοιμασία του μοντέλου και αποστολή

Μετά την εκπαίδευση του μοντέλου, είναι απαραίτητο να μετατραπεί σε μορφή κατάλληλη για εκτέλεση σε ενσωματωμένα συστήματα. Ο εξυπηρετητής εκτελείται σε Python, γι' αυτό και είναι εύκολη η χρήση των συναρτήσεων που παρέχονται από το TensorFlow. Συγκεκριμένα, χρησιμοποιείται ο μετατροπέας `TFLiteConverter`, ο οποίος μετατρέπει το εκπαιδευμένο μοντέλο του *TensorFlow* σε ένα αρχείο `.tflite`. Το αρχείο αυτό περιέχει ένα συμπιεσμένο, ελαφρύ και βελτιστοποιημένο μοντέλο, κατάλληλο για υλοποιήσεις σε πλατφόρμες όπως το ESP32.

Επιπλέον, με τη χρήση της συνάρτησης `port(model, pretty_print = True)` από τη βιβλιοθήκη *tinymlgen*, δημιουργείται το μοντέλο σε μορφή πηγαίου κώδικα C και αποθηκεύεται σε αρχείο με το όνομα `model_quantized.cc`. Το αρχείο αυτό περιλαμβάνει μία δυαδική (binary) αναπαράσταση του μοντέλου. Αν και η δομή του είναι πιο κατανοητή, ωστόσο, όπως έχει αναφερθεί και παραπάνω, δεν μπορεί να χρησιμοποιηθεί.

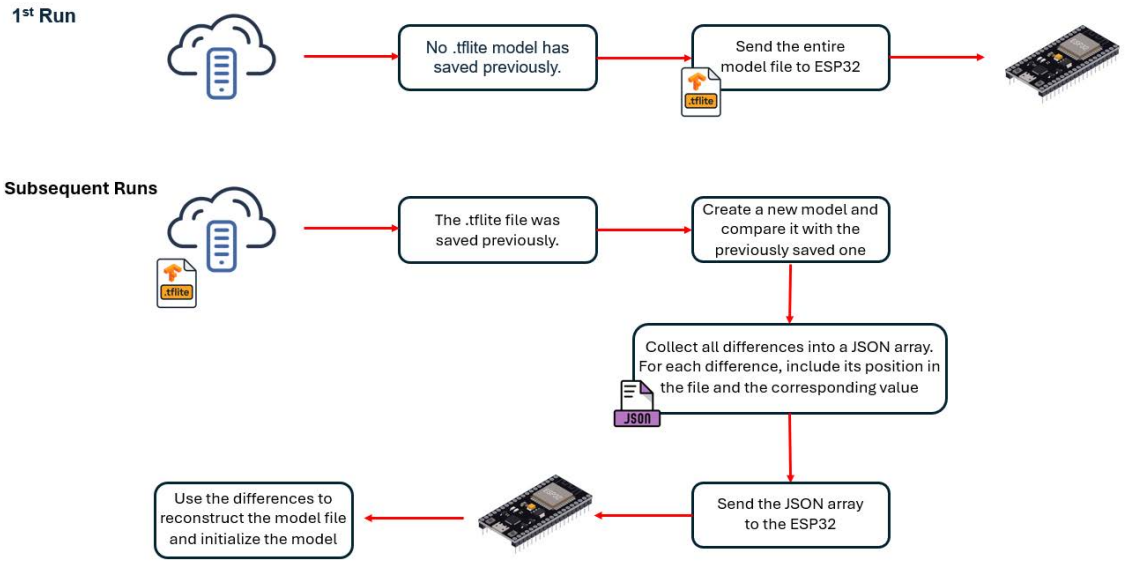
Τελικά, αυτό που αποστέλλεται στη συσκευή είναι το αρχείο `.tflite` (για τους λόγους που έχουν ήδη αναφερθεί), και η αποστολή πραγματοποιείται ως απάντηση στο αίτημα επανεκπαίδευσης της συσκευής.

4.5 Μείωση των δεδομένων που αποστέλλονται στο ESP32

Κάθε φορά που η συσκευή φτάνει το ανώτατο όριο λανθασμένων προβλέψεων που έχει οριστεί για επανεκπαίδευση, αποστέλλει τα δεδομένα που έχει συλλέξει στον εξυπηρετητή. Εκεί πραγματοποιείται η εκ νέου εκπαίδευση του μοντέλου, το οποίο στη συνέχεια επιστρέφεται πίσω στη συσκευή σε μορφή `TensorFlow Lite` (`.tflite` αρχείο). Το μοντέλο αυτό αποθηκεύεται στην εσωτερική μνήμη της συσκευής και χρησιμοποιείται για την αρχικοποίηση του συστήματος.

Ωστόσο, έχει παρατηρηθεί ότι το νέο μοντέλο που προκύπτει μετά από αυτήν την διαδικασία συχνά δεν διαφέρει σημαντικά από το προηγούμενο. Η παρατήρηση αυτή μπορεί να αξιοποιηθεί για τη μείωση του όγκου των δεδομένων που αποστέλλει ο εξυπηρετητής. Συγκεκριμένα, αντί να αποστέλλεται ολόκληρο το `.tflite` αρχείο, μπορούν να αποστέλλονται μόνο οι διαφορές (*byte-level differences*) ανάμεσα στο νέο και στο προηγούμενο μοντέλο. Στην πρώτη επανεκπαίδευση, επειδή ο εξυπηρετητής δεν διαθέτει ήδη αποθηκευμένο μοντέλο για τη συγκεκριμένη συσκευή, αποστέλλεται ολόκληρο το `.tflite` αρχείο. Στη συνέχεια, για κάθε επόμενο γύρο επανεκπαίδευσης, διατηρεί το τελευταίο μοντέλο που έχει δημιουργηθεί για τη συσκευή και, όταν λαμβάνει νέο αίτημα, συγκρίνει το καινούριο με το προηγούμενο *byte προς byte*.

Η εικόνα 4.4 παρουσιάζει την διαδικασία του Diff που εκτελείται, αποτυπωμένη σε μορφή διαγράμματος ροής.



Σχήμα 4.4: Διαδικασία επανεκπαίδευσης αποστέλοντας μόνο την διαφορά των μοντέλων

Το αρχείο `TensorFlow Lite Model (.tflite)` είναι ένα *FlatBuffer* αρχείο. Τα *FlatBuffers* προσφέρουν έναν εξαιρετικά συμπαγή τρόπο αποθήκευσης δεδομένων και μπορούν να διαβαστούν απευθείας από τη μνήμη ή τον δίσκο, χωρίς να απαιτείται αποσυμπίεση. Στο εσωτερικό του αρχείου περιέχονται όλες οι απαραίτητες πληροφορίες για την εκτέλεση του μοντέλου, όπως τα ονόματα των *tensors*, οι διαστάσεις τους (*shapes*), οι τύποι δεδομένων (*data types*), καθώς και οι *buffers* που περιέχουν τα βάρη και τις μετατοπίσεις αυτών των *buffers* μέσα στο αρχείο.

Η πρώτη προσπάθεια υλοποίησης βασίστηκε στην καταγραφή όλων των διαφορών κατά μήκος του αρχείου και την αποστολή τους στη συσκευή. Ωστόσο, όταν το ESP32 εφάρμοζε αυτές τις αλλαγές, επηρέαζε και τα μεταδεδομένα του μοντέλου, με αποτέλεσμα να αλλοιώνεται η εσωτερική δομή του `.tflite` και το αρχείο να καθίσταται μη χρησιμοποιήσιμο.

Για αυτόν τον λόγο, στη δεύτερη προσπάθεια, η οποία και επικράτησε, ο εξυπηρετητής αποστέλλει μόνο τις αλλαγές που εντοπίζονται στα *buffers* που περιέχουν τα βάρη του μοντέλου και όχι στα μεταδεδομένα. Συγκεκριμένα, για κάθε διαφοροποιημένο *buffer*, εντοπίζονται (αν υπάρχουν) οι αλλαγές και για κάθε μία αποστέλλεται η μετατόπιση (*offset*), το μήκος (*length*) και η νέα τιμή. Στη συνέχεια, το ESP32 λαμβάνει αυτές τις πληροφορίες και τις εφαρμόζει στο αποθηκευμένο `.tflite` αρχείο, μετακινώντας τον περιγραφέα αρχείου στη σωστή θέση (*offset*) και γράφοντας τα νέα bytes, διατηρώντας έτσι άθικτη τη δομή του μοντέλου.

Κεφάλαιο 5

Ανεξάρτητη μάθηση με επανεκπαίδευση πάνω στην συσκευή (IL-ODR)

5.1 Εισαγωγή

Ένα σημαντικό στοίχημα αποτελεί η δυνατότητα δημιουργίας και επανεκπαίδευσης ενός μοντέλου μηχανικής μάθησης απευθείας πάνω στο ESP32, χωρίς τη διαμεσολάβηση διακομιστή. Η πρόκληση έγκειται στο ότι οι δυνατότητες της συσκευής, τόσο σε μνήμη όσο και σε υπολογιστική ισχύ, είναι ιδιαίτερα περιορισμένες. Η επανεκπαίδευση ενός μοντέλου απαιτεί πολύπλοκες μαθηματικές διεργασίες, τις οποίες το ESP32 αδυνατεί να εκτελέσει μόνο του.

Στο παρόν έργο έχει γίνει προσπάθεια για την υλοποίηση μιας τέτοιας τεχνικής, με ιδιαίτερο ενδιαφέρον να παρουσιάζει η μέτρηση της απόδοσής της και η σύγκρισή της με τις υπόλοιπες. Το όνομα που έχει επιλεγεί για αυτήν είναι: **Ανεξάρτητη μάθηση με επανεκπαίδευση πάνω στη συσκευή** (*independent learning with on-device retraining*), ή αλλιώς **IL-ODR**, καθώς η συσκευή λειτουργεί ανεξάρτητα από άλλες, δεν απαιτείται η ύπαρξη εξυπηρετητή (*serverless*), και η επανεκπαίδευση πραγματοποιείται απευθείας πάνω στη συσκευή.

5.2 Αναλύσεις από άλλα έργα

Το ζήτημα που αναφέρθηκε παραπάνω, επιχειρούν να αντιμετωπίσουν οι συγγραφείς των άρθρων [9] και [10]. Όπως αναφέρουν, τα μεγάλα μοντέλα μηχανικής μάθησης διαθέτουν τεράστιο αριθμό παραμέτρων και απαιτούν σημαντική υπολογιστική ισχύ (CPU) για να παραγάγουν αποτελέσματα. Στην περίπτωση μοντέλων που εκτελούνται σε μικροσυσκευές, η

προσέγγιση πρέπει να είναι διαφορετική, καθώς οι πόροι σε μνήμη και ενέργεια είναι περιορισμένοι. Αντί, λοιπόν, να επιχειρείται η εκτέλεση πολύπλοκων μοντέλων απευθείας στη συσκευή, η ευκαιρία φαίνεται να βρίσκεται στη χρήση απλούστερων μοντέλων, σχεδιασμένων με συγκεκριμένες απαιτήσεις.

Στα πλεονεκτήματα αυτής της διαδικασίας αναφέρονται οι συγγραφείς του άρθρου [11]. Συγκεκριμένα, η εκπαίδευση απευθείας στον μικροελεγκτή έχει περιορισμούς και επί του παρόντος αποτελεί μια εξειδικευμένη προσέγγιση, αλλά μπορεί να ανταποκρίνεται καλύτερα σε συγκεκριμένες ανάγκες και να ανοίγει νέες δυνατότητες εφαρμογών. Επιτρέπει τη μετάβαση από στατικά μοντέλα (τα οποία εκπαιδεύονται μία φορά και δεν αλλάζουν) σε δυναμικά, τα οποία μπορούν να προσαρμόζονται σε νέα δεδομένα. Επιπλέον, υπάρχει δυνατότητα εξοικονόμησης ενέργειας, λόγω της χαμηλής κατανάλωσης των μικροελεγκτών. Τέλος, για την ανάπτυξη εφαρμογών τεχνητής νοημοσύνης, η εκπαίδευση στη συσκευή μειώνει την ανάγκη πρόσβασης σε υπολογιστικά συστήματα υψηλών επιδόσεων. Τέλος, βελτιώνει την ασφάλεια και την ιδιωτικότητα των δεδομένων, καθώς δεν απαιτείται η αποστολή τους σε κεντρικούς εξυπηρετητές για εκπαίδευση, μειώνοντας έτσι τον κίνδυνο διαρροής ευαίσθητων πληροφοριών.

5.3 Βιβλιοθήκη για αρχικοποίηση μοντέλου, εκπαίδευση και πρόβλεψη

Χρησιμοποιώντας τον κώδικα [12] που έχει συνταχθεί από τους συγγραφείς του παραπάνω άρθρου, γίνεται δυνατή η αρχικοποίηση αλλά και η εκπαίδευση ενός απλού μοντέλου απευθείας πάνω στον μικροελεγκτή. Συγκεκριμένα, όλα τα βάρη του μοντέλου για κάθε επίπεδο (input, hidden και output) βρίσκονται αποθηκευμένα σε float πίνακες με μέγεθος ανάλογο με τις παραμέτρους του μοντέλου κάθε φορά.

Υπάρχει συνάρτηση που αρχικοποιεί αυτές τις δομές, έτσι ώστε κάθε βάρος να ξεκινά με μια μικρή τυχαία τιμή κοντά στο μηδέν. Επιπλέον, παρέχεται η συνάρτηση για το Forward Propagation (υπολογισμός εξόδου), η οποία παίρνει τις εισόδους και τις περνάει μέσα από το δίκτυο ώστε να προκύψει η έξοδος. Αρχικά παράγει τις τιμές για το κρυφό επίπεδο (hidden layer) και ο υπολογισμός που ακολουθεί είναι:

$$h_i = \sigma\left(\sum_{j=1}^{N_{\text{in}}} x_j \cdot w_{ji} + b_i\right)$$

- x_j: είσοδος j
- w_{ji}: βάρος από input $j \rightarrow$ hidden i
- b_i: bias του hidden νευρώνα
- $\sigma(\cdot)$: συνάρτηση ενεργοποίησης sigmoid

Στη συνέχεια, οι τιμές του κρυφού επιπέδου (hidden layer) χρησιμοποιούνται για τον υπολογισμό της εξόδου (output layer):

$$o_k = \sigma\left(\sum_{i=1}^{N_h} h_i \cdot v_{ik} + c_k\right)$$

- h_i: έξοδος από το hidden νευρώνα i
- v_{ik}: βάρος από hidden $i \rightarrow$ output k
- c_k: bias του output νευρώνα
- $\sigma(\cdot)$: συνάρτηση ενεργοποίησης sigmoid

Να σημειωθεί ότι η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η sigmoid, η οποία μετατρέπει οποιονδήποτε αριθμό σε τιμή μεταξύ 0 και 1, π.χ. αν x είναι πολύ θετικό, η έξοδος της συνάρτησης sigmoid είναι περίπου 1. Αν αντίθετα x είναι πολύ αρνητικό, η έξοδος προσεγγίζει το 0. Τέλος, αν $x = 0$, η έξοδος γίνεται 0.5. Ο τύπος της είναι ο εξής:

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

Ακόμα υπάρχει και η συνάρτηση που υποστηρίζει το Backward Propagation (υπολογισμός λαθών και ενημέρωση βαρών), η οποία χρησιμοποιείται για να υπολογίζει πόσο ευθύνεται κάθε βάρος για το τελικό σφάλμα και να το αλλάζει ανάλογα. Η διαδικασία της είναι η εξής:

1. **Υπολογισμός Output Delta - Σφάλμα του Output νευρώνα**

$$\delta_k^o = (t_k - o_k) \cdot o_k \cdot (1 - o_k)$$

- t_k: η επιθυμητή τιμή για τον output νευρώνα k
- o_k: η πραγματική τιμή που έδωσε το δίκτυο μετά τη sigmoid

- $o_k \cdot (1 - o_k)$: παράγωγος της sigmoid, δείχνει πόσο αλλάζει η έξοδος όταν αλλάξει η είσοδος
- δ_k^o: "σφάλμα" του output νευρώνα k, σταθμισμένο με την ευαισθησία του

2. **Υπολογισμός Hidden Delta - Σφάλμα των Hidden neurons**

$$\delta_i^h = \left(\sum_{k=1}^{N_{\text{out}}} v_{ik} \cdot \delta_k^o \right) \cdot h_i \cdot (1 - h_i)$$

- v_{ik}: βάρος από το hidden i προς το output νευρώνα k
- δ_k^o: σφάλμα του output νευρώνα k
- Accum: άθροισμα του σφάλματος των outputs, σταθμισμένο με τα βάρη προς το hidden νευρώνα
- h_i: τιμή ενεργοποίησης του hidden νευρώνα i
- $h_i \cdot (1 - h_i)$: παράγωγος της sigmoid στο hidden επίπεδο
- δ_i^h: βαθμός συμμετοχής του hidden νευρώνα i στο συνολικό σφάλμα

3. **Ενημέρωση βαρών με Momentum**

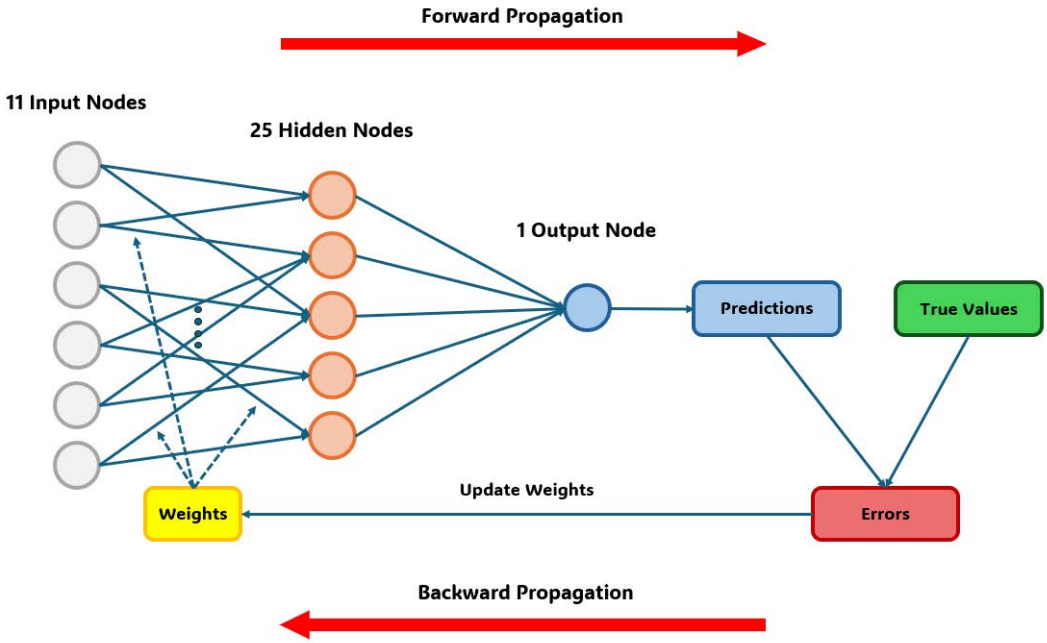
$$\Delta w = \eta \cdot (\text{input}) \cdot \delta + \alpha \cdot \Delta w_{\text{previous}}$$

$$w_{\text{new}} = w_{\text{old}} + \Delta w$$

- η (LearningRate): πόσο μεγάλο βήμα γίνεται στην ενημέρωση
- α (Momentum): ποσοστό της προηγούμενης αλλαγής βάρους που διατηρείται

Κατά την πρόβλεψη καλείται μόνο η *forward propagation*, καθώς απλά χρησιμοποιούνται τα βάρη για να παραχθεί το αποτέλεσμα. Αντίθετα, κατά την εκπαίδευση εκτελούνται τόσο η *forward* όσο και η *backward propagation*, διότι για κάθε συνδυασμό εισόδων του συνόλου εκπαίδευσης παράγεται το αποτέλεσμα και υπολογίζεται το σφάλμα, ώστε να πραγματοποιηθεί η κατάλληλη διόρθωση των λαθών.

Μια ενδεικτική απεικόνιση της εσωτερικής δομής του μοντέλου, καθώς και της διαδικασίας εκπαίδευσής του με αυτή την μέθοδο, παρουσιάζεται στην εικόνα 5.1



Σχήμα 5.1: Εσωτερική δομή του μοντέλου

5.4 Αρχικοποίηση και διαχείριση του μοντέλου στη συσκευή

Έτσι, στην παρούσα εργασία, ακολουθώντας τις παραπάνω μεθόδους, έγινε η προσπάθεια για δημιουργία και εκπαίδευση ενός μοντέλου απευθείας επάνω στο ESP32. Ο τρόπος χρήσης του μοντέλου είναι αυτός που έχει περιγραφεί στην υποκεφάλαιο 3.4.

Στην υλοποίηση έχει ενσωματωθεί η παραπάνω βιβλιοθήκη και χρησιμοποιούνται οι μέθοδοι της. Αρχικά, ορίζονται οι σταθερές που καθορίζουν τη δομή και τον αριθμό των νευρώνων του μοντέλου, και στη συνέχεια αρχικοποιούνται όλοι οι πίνακες (weights και biases) με τυχαίες μικρές τιμές. Η αρχικοποίηση αυτή γίνεται μόνο μία φορά κατά την εκκίνηση και δεν χρειάζεται να επαναλαμβάνεται κατά τη διάρκεια λειτουργίας της συσκευής. Μετά την αρχικοποίηση, το μοντέλο πραγματοποιεί προβλέψεις μέσω της συνάρτησης `prediction`, οι οποίες αξιολογούνται σύμφωνα με τον γενικό τρόπο που έχει ήδη περιγραφεί.

Στην αρχική υλοποίηση, η επανεκπαίδευση γινόταν κάθε φορά στο τέλος της ημέρας, όταν ο αριθμός των λανθασμένων προβλέψεων είχε ξεπεράσει το ορισμένο όριο (threshold). Ωστόσο, με αυτή τη λογική, το μοντέλο παρουσίαζε αρκετά χαμηλή απόδοση και έφτανε σε σημείο να αποτυγχάνουν σχεδόν όλες οι ημερήσιες προβλέψεις του. Για να βελτιωθεί αυτή η κατάσταση, δόθηκε η δυνατότητα στη συσκευή να επανεκπαιδεύει το μοντέλο και κατά τη διάρκεια της ημέρας (κάθε 3 ώρες). Ωστόσο, αυτή η ενδιάμεση επανεκπαίδευση γινόταν με

μία μόνο επανάληψη (epoch), σε σχέση με την κανονική επανεκπαίδευση που χρησιμοποιεί 50. Με αυτόν τον τρόπο, το μοντέλο αποκτά μια μικρή βελτίωση μέσα στην ημέρα, έτσι ώστε να παραμένει πιο κοντά στη ροή των δεδομένων και να μην ξεφεύγει από αυτά. Επιπλέον, η επανεκπαίδευση πραγματοποιείται κάθε φορά χρησιμοποιώντας τα δεδομένα που έχουν συλλεχθεί κατά τη διάρκεια της τρέχουσας ημέρας, ώστε να προστεθεί επιπλέον γνώση στο υπάρχον μοντέλο και να βελτιωθεί η απόδοσή του.

Σε σύγκριση με τις άλλες τεχνικές, αυτή η προσέγγιση είναι πολύ πιο οικονομική, τόσο σε υπολογιστικούς πόρους όσο και σε χρόνο, καθώς όλη η διαδικασία πραγματοποιείται εσωτερικά στη συσκευή, χωρίς να απαιτείται σπατάλη χρόνου για την αποστολή και την αναμονή για παραλαβή δεδομένων.

Κεφάλαιο 6

Συνεργατική / Ομόσπονδη μάθηση με κοινό εξυπηρετητή (FL-SS)

6.1 Εισαγωγή

Κατά τη διαδικασία του IL-ODR που αναλύθηκε στο κεφάλαιο 4, κάθε συσκευή λειτουργεί αυτόνομα, ανεξάρτητα από τις υπόλοιπες. Επικοινωνεί με τον διακομιστή αποστέλλοντας τα νέα ημερήσια δεδομένα, τα οποία αποθηκεύονται χωριστά για καθεμία. Στη συνέχεια, ο διακομιστής εκπαιδεύει μοντέλο αποκλειστικά με βάση τα δεδομένα της συγκεκριμένης συσκευής (τόσο τα πρόσφατα όσο και τα παλαιότερα που έχει αποστείλει) και το επιστρέφει πίσω. Σε όλη αυτή τη διαδικασία, το μοντέλο παραμένει απομονωμένο από τα υπόλοιπα, χωρίς να αξιοποιεί πληροφορίες που προέρχονται από άλλες συσκευές σε διαφορετικά περιβάλλοντα.

Το παραπάνω ζήτημα έρχεται να αντιμετωπίσει μια νέα τεχνική, η ομοσπονδιακή μαθηση. Συνοπτικά, το πλεονέκτημα αυτής της μεθόδου είναι ότι όλες οι συσκευές που ανήκουν στο ίδιο οικοσύστημα συνεργάζονται για να εκπαιδεύσουν από κοινού ένα μοντέλο μηχανικής μάθησης, αξιοποιώντας τα δεδομένα που παράγονται από καθεμία. Οι συσκευές δεν επικοινωνούν άμεσα μεταξύ τους, αλλά έμμεσα, μέσω ενός κεντρικού διακομιστή. Ο τελευταίος αναλαμβάνει τη συλλογή και την επεξεργασία των δεδομένων, συντονίζοντας τη συνολική διαδικασία εκπαίδευσης.

Έτσι, στα πλαίσια της παρούσας εργασίας, έχει γίνει προσπάθεια υλοποίησης μίας αντίστοιχης τεχνικής, η οποία έχει ονομασθεί **Συνεργατική / Ομόσπονδη μάθηση με κοινό εξυπηρετητή** (*federated learning with shared server*), ή αλλιώς **FL-SS**. Το όνομα αυτό προ-

έκυψε από το γεγονός ότι στηρίζεται στη λογική του FL, καθώς όλες οι συσκευές συνεργάζονται για κοινό στόχο την βελτίωση του μοντέλου. Επιπλέον, όλες επικοινωνούν με τον ίδιο εξυπηρετητή για τη συλλογή των δεδομένων και τη δημιουργία του ενιαίου μοντέλου.

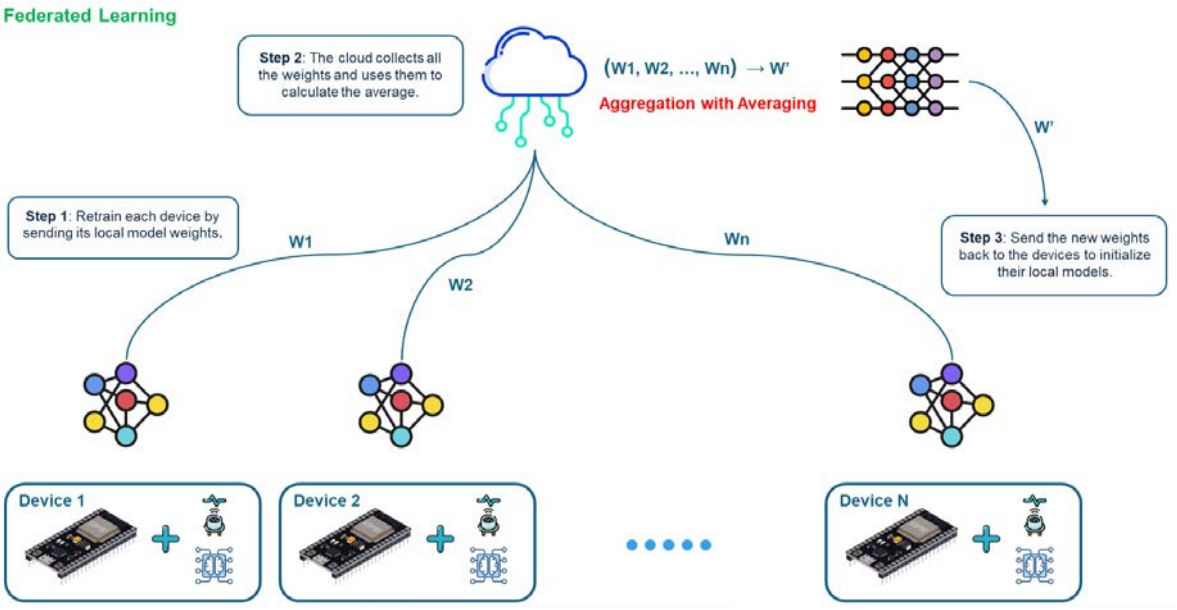
6.2 Γενική περιγραφή

Αρχικά, κάθε συσκευή εκπαιδεύει το δικό της μοντέλο, εφαρμόζοντας την τεχνική που αναλύθηκε στο Κεφάλαιο 5. Το στάδιο αυτό είναι ιδιαίτερα σημαντικό, καθώς, όταν απαιτείται η έναρξη της διαδικασίας επανεκπαίδευσης, η συσκευή δεν χρειάζεται να αποστείλει όλα τα ημερήσια δεδομένα, αλλά μόνο τα βάρη του μοντέλου της. Με αυτόν τον τρόπο, δεν διακινούνται προσωπικές πληροφορίες μέσω του δικτύου, γεγονός που ενισχύει την ασφάλεια έναντι πιθανών υποκλοπών. Όπως αναφέρθηκε, κατά την επανεκπαίδευση κάθε συσκευή που συμμετέχει στον κύκλο αποστέλλει τα βάρη του μοντέλου της στον διακομιστή. Ο διακομιστής, αφού τα συγκεντρώσει, εφαρμόζει μια τεχνική συνένωσής τους ώστε να παραχθεί ένα νέο, ενιαίο μοντέλο. Στη συνέχεια, τα βάρη του ανανεωμένου μοντέλου αποστέλλεται πίσω στις συσκευές, οι οποίες τα ενσωματώνουν. Μετέπειτα, εκπαιδεύουν τοπικά το μοντέλο με τα δεδομένα που έχουν συλλέξει από τη ημερίσια χρήση τους και συνεχίζουν τις προβλέψεις τους βασιζόμενες σε αυτό.

Η διαδικασία της επανεκπαίδευσης μπορεί να χαρακτηριστεί ως **ασύγχρονη**, καθώς οι συσκευές αποστέλλουν τα δεδομένα τους στον εξυπηρετητή και, όσο αναμένουν την απάντηση, συνεχίζουν κανονικά τη λειτουργία τους χρησιμοποιώντας το προηγούμενο μοντέλο για την παραγωγή προβλέψεων. Αυτό καθίσταται εφικτό μέσω της χρήσης ξεχωριστού νήματος, το οποίο επιτρέπει την παράλληλη εκτέλεση της διαδικασίας επανεκπαίδευσης χωρίς να διακόπτεται η κανονική λειτουργία της συσκευής.

Μια παρατήρηση είναι ότι, αν μια συσκευή έχει ικανοποιητική απόδοση, τότε δεν χρειάζεται να επανεκπαιδεύσει το μοντέλο της. Συνεπώς, δεν θα έχει ενημερώσει τον εξυπηρετητή για τη συμμετοχή της και έτσι αυτός δεν θα την περιμένει. Με αυτόν τον τρόπο, κατά τη διάρκεια της εκτέλεσής τους, υπάρχει πιθανότητα κάποιες συσκευές να μην έχουν εκπαιδευτεί τόσες φορές όσες κάποιες άλλες. Ωστόσο, αυτό δεν τις καθιστά πιο αδύναμες, καθώς το γεγονός ότι δεν συμμετέχουν δείχνει ότι το μοντέλο τους είναι ήδη ικανό και δεν χρειάζεται τόσο βελτίωση σε σχέση με άλλες συσκευές.

Μια ενδεικτική εικόνα της παραπάνω διαδικασίας είναι η 6.1



Σχήμα 6.1: Διάγραμμα λειτουργείας της τεχνικής του FL-SS

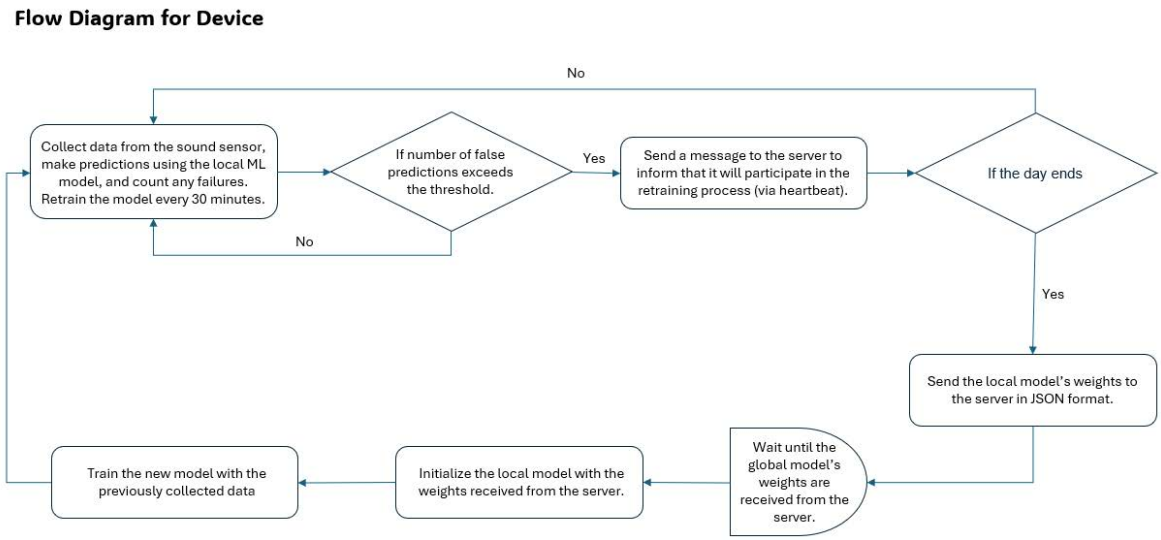
6.3 Από την πλευρά της συσκευής

Στο υποκεφάλαιο αυτό περιγράφεται η λειτουργία της συσκευής κατά τη διάρκεια της διαδικασίας του FL-SS. Εν συντομία, η διαδικασία περιλαμβάνει τη συλλογή δεδομένων από τον αισθητήρα, την εκτέλεση προβλέψεων από το μοντέλο που λειτουργεί στη συσκευή, την υποβολή αιτήματος για επανεκπαίδευση όταν αυτό κρίνεται απαραίτητο, την αναμονή για τη λήψη νέων βαρών και την εκπαίδευση του ανανεωμένου μοντέλου με τα δεδομένα που έχει ήδη συλλέξει. Επιπλέον, η λειτουργικότητά της παρουσιάζεται συνοπτικά στο διάγραμμα ροής (*Flow Diagram*) 6.2.

6.3.1 Αρχικοποίηση και διαχείριση του μοντέλου στη συσκευή

Η συσκευή δημιουργεί και χρησιμοποιεί ένα μοντέλο μηχανικής μάθησης απευθείας στη μνήμη της, χωρίς να απαιτείται η διαμεσολάβηση του εξυπηρετητή για την κατασκευή ή την εκπαίδευσή του. Αυτό είναι εφικτό χρησιμοποιώντας την υλοποίηση που έχει αναλυθεί στο Κεφάλαιο 5. Η εσωτερική δομή του μοντέλου είναι ίδια με αυτή που έχει περιγραφεί στην παραπάνω μέθοδο, και τα βάρη του είναι αποθηκευμένα σε αντίστοιχους πίνακες στο εσωτερικό της συσκευής. Κάθε φορά που χρειάζεται να γίνει πρόβλεψη της επόμενης τιμής του ήχου, καλείται η συνάρτηση `prediction()` η οποία εκτελεί *forward propagation*.

Επιπλέον, κάθε 3 ώρες μέσα στην ημέρα, πραγματοποιείται εκπαίδευση του μοντέλου,



Σχήμα 6.2: Διάγραμμα ροής της συσκευής κατά τη διάρκεια του FL-SS

ώστε αυτό να προσαρμόζεται στα νέα δεδομένα, συνδυάζοντας τόσο τα προηγούμενα όσο και τα πρόσφατα που συλλέχθηκαν στο διάστημα μεταξύ των δύο εκπαιδεύσεων. Κατά τη διάρκεια αυτής της διαδικασίας, τα βάρη (weights) και οι σταθερές μετατόπισης (biases) ενημερώνονται μέσω του αλγορίθμου εκμάθησης με στόχο τη μείωση του σφάλματος πρόβλεψης, επιτρέποντας στο μοντέλο να βελτιώνεται συνεχώς. Η διαδικασία αυτή εκτελείται σε όλες τις συσκευές ανεξαιρέτως, εξασφαλίζοντας ότι καθεμία εξελίσσεται σταδιακά και αυτόνομα, χωρίς καμία να μένει πίσω ως προς την προσαρμογή της σε αυτό το επίπεδο. Επίσης, όπως έχει ήδη αναφερθεί, η παραπάνω μέθοδος (IL-ODR) την έχει εξίσου ενσωματώσει, καθώς χωρίς αυτή το μοντέλο παρουσίαζε σημαντική αδυναμία στις προβλέψεις.

Οι τιμές από τον αισθητήρα συνεχίζουν να αποθηκεύονται έτσι ώστε να χρησιμοποιηθούν αποκλειστικά σε αυτό το σημείο της εκπαίδευσης, καθώς, όπως έχει ήδη αναφερθεί, σε αυτήν τη μέθοδο δεν χρειάζεται να αποσταλούν στον εξυπηρετητή.

6.3.2 Εκκίνιση διαδικασίας ομοσπονδιακής μάθησης και αποστολή βαρών

Αν, κατά τη διάρκεια της ημέρας, ο αριθμός των λανθασμένων προβλέψεων ξεπεράσει ένα καθορισμένο όριο, τότε, μέσω του παλμού (που στέλνει στον εξυπηρετητή κάθε λεπτό για να τον ενημερώνει για την κατάστασή της), τον ειδοποιεί ότι στο τέλος της ημέρας θα χρειαστεί επανεκπαίδευση. Έτσι, όταν τελειώσει η ημέρα, ενεργοποιείται η διαδικασία της ομοσπονδιακής μάθησης. Η διαδικασία αυτή εκτελείται σε διαφορετικό νήμα, ώστε να μην

διακοπεί η κανονική λειτουργία της συσκευής και να συνεχίσει να συλλέγει δεδομένα και να πραγματοποιεί προβλέψεις μέχρι να αρχικοποιηθεί το νέο μοντέλο. Η συσκευή συλλέγει όλα τα βάρη του μοντέλου και τα μετατρέπει σε μορφή *JSON* για να τα στείλει στον εξυπηρετητή. Στο μήνυμα περιλαμβάνονται τόσο τα *hidden* όσο και τα *output* βάρη. Η αποστολή πραγματοποιείται μέσω του πρωτοκόλλου *HTTP* και της μεθόδου *POST*.

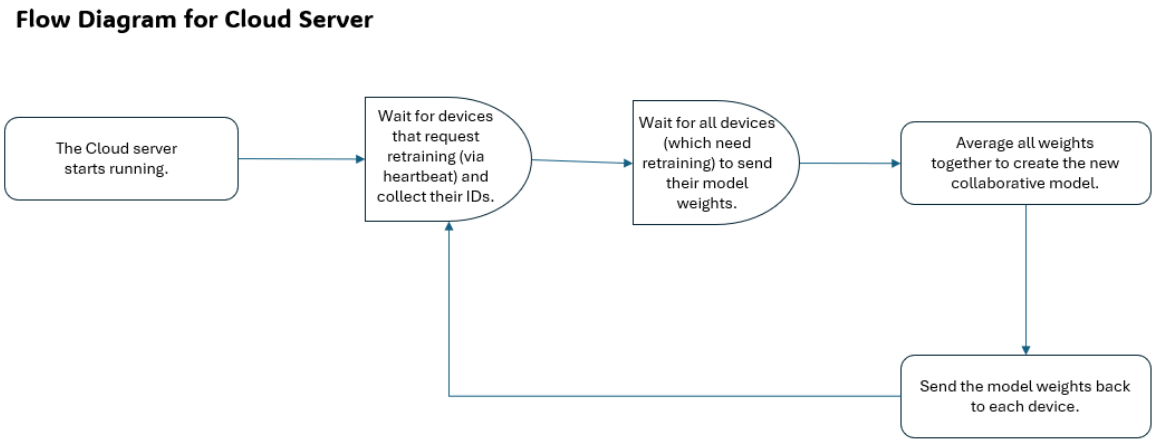
6.3.3 Αναμονή για τα νέα βάρη και ανανέωση μοντέλου

Στη συνέχεια, η συσκευή θα πρέπει να περιμένει τα νέα βάρη από τον εξυπηρετητή. Ωστόσο, αυτός δεν μπορεί από μόνος του να στείλει μήνυμα σε αυτήν, καθώς δεν γνωρίζει την δημόσια διεύθυνση IP της. Αντ' αυτού, κάθε φορά στέλνει μηνύματα ως απαντήση (*responses*) στα αιτήματα της. Για τον λόγο αυτό, στη συγκεκριμένη διαδικασία, όπου η συσκευή αναμένει νέα δεδομένα από τον εξυπηρετητή, θα πρέπει επαναληπτικά να αποστέλλει ένα συγκεκριμένο αίτημα έως ότου λάβει την επιθυμητή απάντηση από αυτόν. Μόλις λάβει την απάντηση, σταματά την συνεχή αποστολή μηνυμάτων και προχωρά στη διαδικασία αρχικοποίησης του νέου μοντέλου.

Το μήνυμα του εξυπηρετητή περιέχει, σε μορφή *JSON*, τα βάρη για το *hidden* και το *output* επίπεδο. Η συσκευή αντικαθιστά τα παλιά βάρη με τα νέα στους αντίστοιχους πίνακες και επαν-αρχικοποιεί το μοντέλο, καλώντας τις κατάλληλες συναρτήσεις. Στη συνέχεια, προχωρά στην εκπαίδευσή του με τα δεδομένα που έχει ήδη συλλέξει κατά τη διάρκεια της ημέρας, ώστε το μοντέλο να προσαρμοστεί στις συνθήκες του περιβάλλοντος της συγκεκριμένης συσκευής.

6.4 Από την πλευρά του εξυπηρετητή

Ο εξυπηρετητής νέφους, σε αυτή τη μέθοδο, είναι υπεύθυνος για τη συλλογή όλων των τιμών από τις συσκευές που επιθυμούν να επανεκπαιδεύσουν το μοντέλο τους. Στη συνέχεια, αξιοποιεί τα δεδομένα αυτά για να δημιουργήσει το βέλτιστο ενιαίο μοντέλο, το οποίο αποστέλλει ξανά πίσω σε αυτές. Στο παρόν υποκεφάλαιο περιγράφεται αναλυτικά η διαδικασία αυτή σε διακριτά στάδια, ενώ ακολουθεί και το αντίστοιχο διάγραμμα ροής 6.3



Σχήμα 6.3: Διάγραμμα ροής του εξυπηρετητή κατά τη διάρκεια του FL-SS

6.4.1 Προσθήκη συσκευής στη διαδικασία της επανεκπαίδευσης

Όσο ο εξυπηρετητής βρίσκεται σε κατάσταση αναμονής, δέχεται τα μηνύματα παλμών από τις συσκευές. Τα μηνύματα αυτά, πέρα από διάφορες λειτουργικές πληροφορίες, περιέχουν και ένα flag, με το οποίο δηλώνεται η πρόθεση της συσκευής να συμμετάσχει στη διαδικασία της επανεκπαίδευσης. Αν το flag είναι θετικό (*true*), τότε ο εξυπηρετητής καταμετρά τη συσκευή στη διαδικασία, αποθηκεύοντας το αναγνωριστικό (ID) της σε έναν αντίστοιχο πίνακα.

Έτσι, με αυτή τη λειτουργικότητα, δίνεται η δυνατότητα σε μια συσκευή να μην χρειάζεται επανεκπαίδευση (εφόσον το μοντέλο της δεν έχει ξεπεράσει τον αριθμό λαθών που απαιτείται για να προχωρήσει στη διαδικασία αυτή). Συνεπώς, διατηρεί συνεχώς την τιμή ψευδές (*false*) στο αντίστοιχο flag του παλμού και ο εξυπηρετητής την παραλείπει τελικά από τη διαδικασία της επανεκπαίδευσης. Λόγω αυτού, ενδέχεται κάποιες συσκευές να μην έχουν περάσει από τον ίδιο αριθμό κύκλων εκπαίδευσης σε σχέση με κάποιες άλλες.

6.4.2 Αναμονή συλλογής όλων των δεδομένων από τις συσκευές

Όταν σταλεί το πρώτο μήνυμα μιας συσκευής που περιέχει τα βάρη του μοντέλου της, θεωρείται ότι όλες οι συσκευές που επιθυμούν να συμμετάσχουν στη διαδικασία έχουν ήδη δηλώσει την πρόθεσή τους μέσω του παλμού. Έτσι, ο εξυπηρετητής περιμένει να λάβει τόσα μηνύματα όσα και ο αριθμός των συσκευών που συμμετέχουν στην επανεκπαίδευση. Κάθε σύνολο βαρών που λαμβάνεται από κάθε συσκευή αποθηκεύεται σε ξεχωριστή θέση ενός πίνακα, ώστε να συγκεντρωθούν όλα και να προχωρήσει στο επόμενο βήμα, που είναι ο

συνδυασμός τους. Γενικά αξίζει να σημειωθεί ότι, η διαδικασία που εκτελεί είναι **σύγχρονη**, καθώς μέχρι να συλλεχθούν όλα τα δεδομένα από όλες τις συσκευές δεν εκτελεί καμία άλλη διαδικασία.

6.4.3 Προετοιμασία του μοντέλου και αποστολή

Μόλις ο εξυπηρετητής έχει συλλέξει όλα τα δεδομένα που περιμένει από τις συσκευές, μπορεί να προχωρήσει στο επόμενο στάδιο, που είναι η δημιουργία του νέου μοντέλου. Κάθε συσκευή έχει αποστείλει ολόκληρο το σύνολο των βαρών της από τα hidden και output επίπεδα. Έτσι αυτός μπορεί να τα χρησιμοποιήσει για τη δημιουργία του ενιαίου μοντέλου. Για να το επιτύχει, εφαρμόζει τη μέθοδο του Μέσου Όρου (Averaging), υπολογίζοντας τον για κάθε σύνολο βαρών (δηλαδή για κάθε νευρώνα σε κάθε επίπεδο), ώστε να προκύψει το τελικό μοντέλο. Έτσι, αυτό περιέχει έμμεσα χαρακτηριστικά από όλες τις συσκευές και γι' αυτό μπορεί να θεωρηθεί ως κοινό τους μοντέλο. Στο τέλος, πακετάρει όλα τα νέα βάρη σε μήνυμα μορφής JSON και τα αποστέλλει σε όλες τις συσκευές. Η αποστολή γίνεται απαντώντας σε ένα από τα συνεχόμενα μηνύματα που δέχεται από κάθε συσκευή.

Κεφάλαιο 7

Αξιολόγηση και Σύγκριση των Μεθόδων Εκπαίδευσης

7.1 Εισαγωγή

Μετά την υλοποίηση των τριών μεθόδων που έχουν αναλυθεί στις παραπάνω ενότητες (4, 5 και 6), είναι απαραίτητο να δοκιμαστούν και να αξιολογηθούν υπό κανονικές συνθήκες λειτουργίας. Για να γίνει η αξιολόγηση ταχύτερα και με μεγαλύτερη αξιοπιστία, έχει προστεθεί μια αντίστοιχη λειτουργικότητα στις υλοποιήσεις. Επιπλέον, τα αποτελέσματα των δοκιμών παρουσιάζονται και σε γραφικές απεικονίσεις, ώστε να είναι δυνατή η εξαγωγή μιας αντικειμενικής γνώμης για την απόδοση κάθε τεχνικής.

7.2 Προσθήκη πειραματικής λειτουργίας στην υλοποίηση

Ένας δίκαιος τρόπος αξιολόγησης των μεθόδων που έχουν αναπτυχθεί είναι να δοκιμαστούν υπό τις ίδιες ακριβώς συνθήκες λειτουργίας. Αυτό σημαίνει ότι, για κάθε χρονική στιγμή μέσα στην ημέρα, θα πρέπει να δέχονται την ίδια είσοδο από τον αισθητήρα ο οποίος είναι συνδεδεμένος σε αυτές. Επιπλέον, θα ήταν ιδιαίτερα χρήσιμο να επιταχυνθεί η διαδικασία λειτουργίας της συσκευής, καθώς στη βασική υλοποίηση οι τιμές ανακτώνται σταδιακά με την πάροδο του χρόνου, και για την ενεργοποίηση της διαδικασίας της επανεκπαίδευσης απαιτείται τουλάχιστον μία ημέρα αναμονής, γεγονός που καθιστά την αξιολόγηση αρκετά χρονοβόρα.

Για να επιτευχθούν τα παραπάνω, ο κώδικας σε όλες τις μεθόδους τροποποιήθηκε ώστε

να προστεθεί πειραματική λειτουργία. Με την ενεργοποίησή της, η συσκευή δεν περιμένει την πάροδο του χρόνου για να συλλέξει δεδομένα και να εκτελέσει προβλέψεις, αλλά ζητά από τον εξυπηρετητή το σύνολο τιμών μιας ημέρας (χρόνου και ήχου), το οποίο αποθηκεύει στη μνήμη της. Στη συνέχεια, αντί να διαβάζει τιμές από τον αισθητήρα, χρησιμοποιεί τα αποθηκευμένα δεδομένα και ακολουθεί την ίδια διαδικασία με τη βασική λειτουργία. Ο εξυπηρετητής διαθέτει αποθηκευμένα αρχεία με ημερήσια δεδομένα και, κάθε φορά που λαμβάνει το αντίστοιχο αίτημα, αποστέλλει το αρχείο που ζητείται. Σε περίπτωση που ζητηθεί αρχείο που δεν υπάρχει, απαντά αρνητικά και η διαδικασία ολοκληρώνεται. Με αυτή την υλοποίηση, εξασφαλίζεται ότι όλες οι συσκευές χρησιμοποιούν τις ίδιες εισόδους και αξιολογούνται υπό κοινές συνθήκες, ενώ η εκτέλεση επιταχύνεται σημαντικά, καθώς τα απαραίτητα δεδομένα είναι διαθέσιμα από την αρχή και μπορούν να χρησιμοποιηθούν άμεσα.

7.3 Μετρικές αξιολόγησης

Για να μπορέσουν να αξιολογηθούν όλες οι υλοποιήσεις, έχουν επιλεγεί ορισμένες μετρικές. Ανάλογα με τις τιμές που θα προκύψουν για κάθε μέθοδο, θα είναι δυνατός ο εντοπισμός τόσο των σημείων στα οποία υπερέχει η κάθε μία, όσο και αυτών στα οποία υστερεί. Συγκεκριμένα, οι μετρικές είναι οι εξής:

1. **Ποσοστό λανθασμένων προβλέψεων**:Αναπαριστά το ποσοστό των λανθασμένων προβλέψεων για κάθε ημέρα εκτέλεσης των συσκευών. Όσο μεγαλύτερο είναι, τόσο πιο αδύναμο θεωρείται το μοντέλο και τόσο μεγαλύτερη είναι η ανάγκη για επανεκπαίδευσή του.

2. **Μέγεθος μηνυμάτων κατά την επανεκπαίδευση**: Ορίζει το μέγεθος των δεδομένων που διακινούνται για την επικοινωνία, τόσο από τη συσκευή προς τον εξυπηρετητή όσο και αντίστροφα. Όσο μικρότερο είναι το μέγεθος, τόσο ταχύτερη είναι η μετάδοση και τόσο χαμηλότερη η κατανάλωση πόρων.

3. **Απόδοση μοντέλου**: Αξιολογείται μετά από κάθε κύκλο επανεκπαίδευσης. Επειδή το πρόβλημα ανήκει στην κατηγορία της παλινδρόμησης, χρησιμοποιείται η **Ρίζα Μέσου Τετραγωνικού Σφάλματος (RMSE)**. Όσο μικρότερη είναι η τιμή της, τόσο ακριβέστερες είναι οι προβλέψεις του μοντέλου και τόσο καλύτερη η συνολική του απόδοση.

4. **Χρόνος ολοκλήρωσης επανεκπαίδευσης και αρχικοποίησης του μοντέλου**: Ο χρόνος που απαιτείται ώστε να αποσταλούν τα δεδομένα στον εξυπηρετητή, να ληφθεί η απάντηση (εκτός από την IL-ODR), να αρχικοποιηθεί το μοντέλο κατάλληλα, και να είναι ξανά διαθέσιμο για προβλέψεις. Μικρότερη διάρκεια συνεπάγεται ταχύτερη λειτουργία, αν και η πολυπλοκότητα του μοντέλου μπορεί να τον αυξήσει.

5. **Κατανάλωση πόρων**: Η χρήση **CPU** και **μνήμης**, τόσο στη συσκευή όσο και στον εξυπηρετητή, κατά τη διάρκεια της επανεκπαίδευσης. Όσο πιο πολύπλοκη είναι η διαδικασία, τόσο μεγαλύτερη είναι και η κατανάλωση.

7.4 Περιβάλλον και δεδομένα δοκιμών

Η συσκευή που χρησιμοποιήθηκε βασίζεται σε διπύρηνο επεξεργαστή, διαθέτει 520 KB ενσωματωμένης SRAM και 4 MB εξωτερικής μνήμης Flash. Αντίστοιχα, ο εξυπηρετητής διαθέτει δύο πυρήνες επεξεργαστή, 2 GB μνήμη RAM και 20 GB αποθηκευτικό χώρο.

Η μέθοδος IL-SBR και η αντίστοιχη με χρήση Diff εκτελέστηκαν με τη χρήση μιας συσκευής, και έτσι ο εξυπηρετητής επιβαρύνεται μόνο από αυτή. Αντίθετα, στην περίπτωση του FL-SS, χρησιμοποιούνται δύο συσκευές κατά τη διάρκεια του πειράματος. Στην IL-ODR η εκτέλεση γίνεται πάντα ατομικά.

Η αξιολόγηση των μεθόδων πραγματοποιείται σε δεδομένα 18 συνεχόμενων ημερών, τα οποία προέρχονται από το ίδιο περιβάλλον, με παρόμοια μέση ένταση ήχου, και έχουν συλλεχθεί από μία μόνο συσκευή. Κάθε ημερήσιο αρχείο περιέχει **1440 εγγραφές**, που αντιστοιχούν στη μέτρηση της έντασης του ήχου ανά λεπτό μέσα στη μέρα.

Οι τρεις πρώτες μέθοδοι χρησιμοποιούν σειριακά όλα τα δεδομένα αυτών των ημερών. Αντίθετα, στην περίπτωση της μεθόδου **FL-SS**, τα δεδομένα κατανέμονται μεταξύ δύο συσκευών με τον εξής τρόπο:

- Από κάθε ημερήσιο αρχείο δημιουργούνται δύο νέα αρχεία (ένα για κάθε συσκευή).
- Οι εγγραφές κατανέμονται εναλλάξ: η πρώτη τιμή (πρώτο λεπτό) αποθηκεύεται στη συσκευή 1, η δεύτερη στη συσκευή 2, η τρίτη ξανά στη συσκευή 1 κ.ο.κ.
- Έτσι, στο τέλος, η συσκευή 1 περιέχει όλα τα δεδομένα των *μονών* λεπτών, ενώ η συσκευή 2 όλα των *ζυγών*.

Με αυτόν τον τρόπο, και οι δύο συσκευές διαθέτουν δεδομένα από όλες τις ημέρες, αλλά με **μισή συχνότητα δειγματοληψίας (sample rate)**. Κατά την εκπαίδευση, στο τέλος κάθε

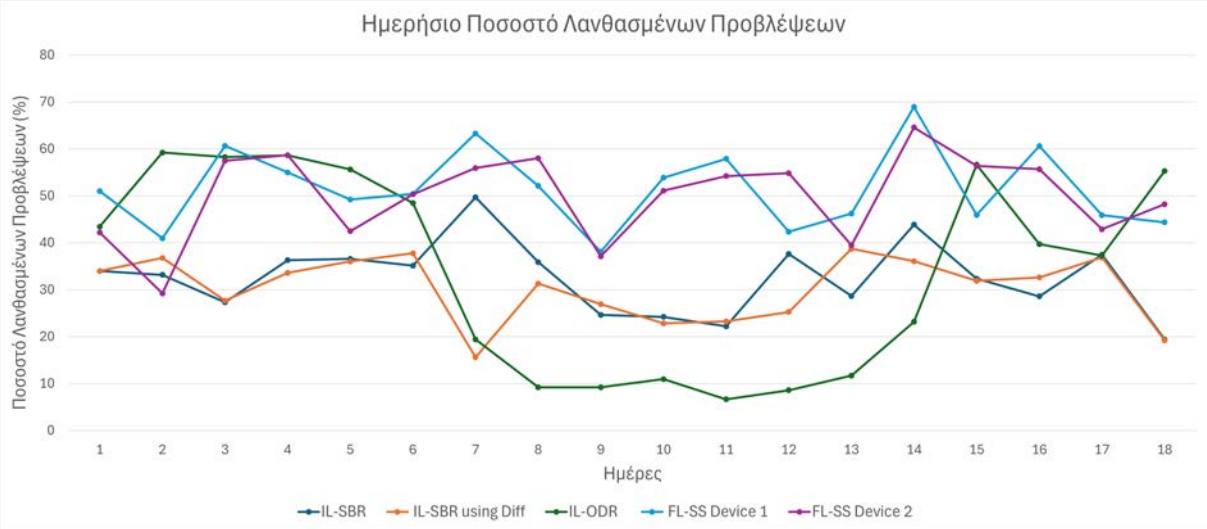
κύκλου, τα μοντέλα των δύο συσκευών συγχωνεύονται, ενσωματώνοντας πληροφορίες και από τα δύο υποσύνολα. Συνεπώς, στο τέλος της διαδικασίας, το προκύπτον μοντέλο έχει εκπαιδευτεί ουσιαστικά πάνω στο **ίδιο σύνολο δεδομένων** όπως και οι υπόλοιπες μέθοδοι, κάτι που επιτρέπει την **άμεση και δίκαιη σύγκρισή τους**.

Η διαδικασία επανεκπαίδευσης ενεργοποιείται όταν οι λανθασμένες προβλέψεις υπερβούν τις 300. Μια πρόβλεψη χαρακτηρίζεται ως λανθασμένη όταν η τιμή της αποκλίνει από την πραγματική κατά τουλάχιστον ± 100 μονάδες. Οι παράμετροι αυτοί είναι προκαθορισμένοι σε κάθε πρόγραμμα, αλλά μπορούν να προσαρμοστούν ανάλογα με το περιβάλλον χρήσης.

7.5 Απόδοση υλοποιήσεων

Μετά την εκτέλεση όλων των μεθόδων, πραγματοποιήθηκε η συλλογή των παραπάνω μετρικών, ώστε να καταστεί δυνατή η αξιολόγηση και η ερμηνεία της απόδοσής τους.

Παρακάτω παρουσιάζονται γραφήματα ή πίνακες για κάθε μετρική, τα οποία απεικονίζουν τις τιμές που προέκυψαν από όλες τις μεθόδους. Με τον τρόπο αυτό, η αξιολόγηση και η σύγκριση τους καθίσταται πιο άμεση και αποτελεσματική.



Σχήμα 7.1: Διάγραμμα αναπαράστασης του ποσοστού των λανθασμένων προβλέψεων ανά ημέρα για κάθε μέθοδο

Για το **ποσοστό των λανθασμένων προβλέψεων** ανά ημέρα, προέκυψε το γράφημα 7.1. Αναλύοντάς το, φαίνεται ότι οι μέθοδοι **IL-SBR** και η αντίστοιχη με χρήση **Diff** παρουσιάζουν παρόμοια συμπεριφορά. Σταδιακά έχουν πτωτική πορεία στον ποσοστό λαθών, με μι-

κρές διακυμάνσεις. Αυτό δείχνει ότι η χρήση του εξυπηρετητή (με ή χωρίς επιστροφή μόνο της διαφοράς του μοντέλου) είναι σχετικά σταθερή και βελτιώνεται με τον χρόνο. Η μέθοδος **IL-ODR** στην αρχή έχει υψηλά σφάλματα, αλλά στη συνέχεια μειώνεται πολύ έντονα, φτάνοντας σε χαμηλές τιμές. Αυτό σημαίνει ότι το μοντέλο που εκπαιδεύεται πάνω στη συσκευή χρειάζεται λιγότερη επανεκπαίδευση μετά από κάποιο σημείο. Η απότομη αύξηση που φαίνεται μετά την ημέρα 14 είναι χαρακτηριστική και πιθανόν να οφείλεται σε αλλαγή της ροής και της φύσης των δεδομένων (concept drift). Τέλος, οι συσκευές που εκτελούν **FL-SS** έχουν σχετικά σταθερή συμπεριφορά. Ωστόσο, το επίπεδο των σφαλμάτων τους είναι πιο υψηλό σε σχέση με τις άλλες μεθόδους. Από αυτό φαίνεται ότι, αν και η εκπαίδευση είναι πιο ισορροπημένη μεταξύ των συσκευών, η απόδοση τους είναι χαμηλότερη.

Το **μέγεθος των μηνυμάτων κατά την επανεκπαίδευση** μετρήθηκε τόσο για αυτό που αποστέλλει η συσκευή προς τον εξυπηρετητή, όσο και για εκείνο που αποστέλλει ο εξυπηρετητής προς τη συσκευή. Να σημειωθεί ότι κατά τη διάρκεια της ημέρας ενδέχεται να υπάρξει το πολύ μία επικοινωνία για επανεκπαίδευση μεταξύ τους (η συσκευή θα στείλει δεδομένα και ο εξυπηρετητής θα απαντήσει με πληροφορίες του νέου μοντέλου). Παρακάτω παρουσιάζονται οι αντίστοιχοι πίνακες για τις δύο περιπτώσεις. Στον πίνακα 7.1 απεικονίζεται η πρώτη περίπτωση. Οι τιμές που εμφανίζονται για τις δύο πρώτες μεθόδους (IL-SBR και η αντίστοιχη με χρήση Diff) παρέμειναν σταθερές για κάθε μέρα του πειράματος. Αντίθετα, στην περίπτωση της μεθόδου FL-SS, οι τιμές παρουσίασαν μικρές διακυμάνσεις ($\pm$20 bytes) μεταξύ των ημερών, και για τον λόγο αυτό καταγράφεται η μέση τιμή τους. Αντίστοιχα, στον πίνακα 7.2 απεικονίζεται η δεύτερη περίπτωση, όπου οι τιμές παρέμειναν σταθερές καθ' όλη τη διάρκεια των ημερών. Αξίζει επίσης να σημειωθεί ότι στην περίπτωση της μεθόδου IL-ODR η συγκεκριμένη μετρική δεν υφίσταται, καθώς ο εξυπηρετητής δεν συμμετέχει στη διαδικασία.

Οι μέθοδοι **IL-SBR** και η αντίστοιχη **με χρήση Diff** παρουσιάζουν ίδιο μέγεθος μηνύματος, καθώς σε κάθε κύκλο αποστέλλουν όλα τα ημερήσια δεδομένα. Αντίθετα, στη μέθοδο **FL-SS** το μέγεθος είναι σημαντικά μικρότερο, καθώς μεταδίδονται μόνο τα βάρη του μοντέλου (hidden και output).

Σχετικά με τα μηνύματα που αποστέλλει ο εξυπηρετητής προς τη συσκευή κατά την επανεκπαίδευση, στην περίπτωση του **IL-SBR**, το μέγεθος ισούται με εκείνο του αρχείου `.tflite` που αποστέλλεται. Στον πρώτο κύκλο επανεκπαίδευσης του **IL-SBR με χρήση Diff**, το μήνυμα έχει επίσης το ίδιο μέγεθος, καθώς ο εξυπηρετητής δεν διαθέτει ακόμη προ-

Μέθοδος	Μέγεθος (bytes)
IL-SBR	5760
IL-SBR using Diff	5760
FL-SS Device 1	$\sim$ 3120
FL-SS Device 2	$\sim$ 3123

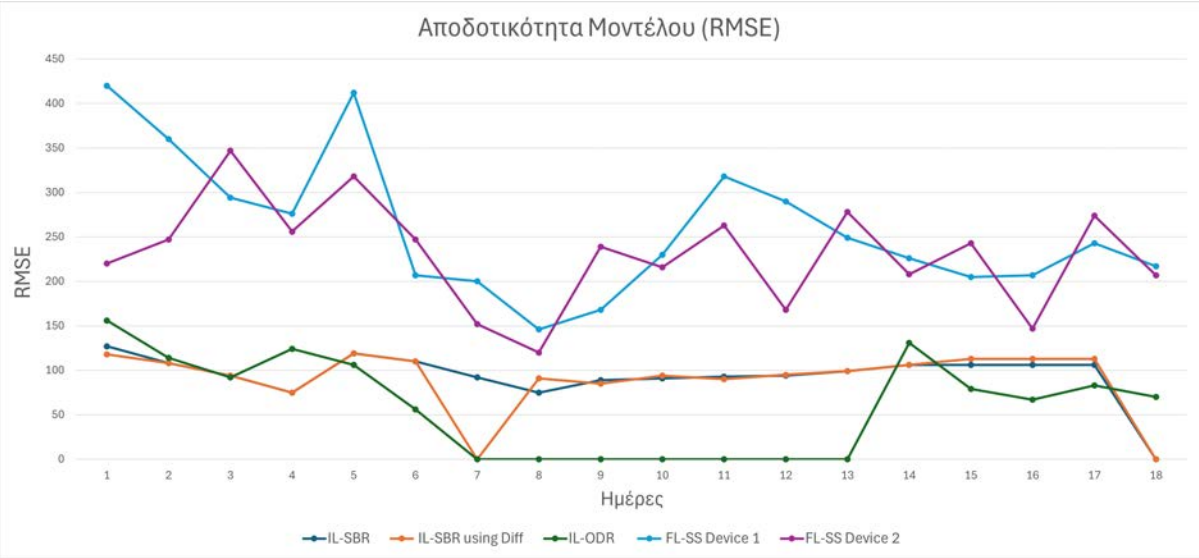
Πίνακας 7.1: Μέγεθος μηνυμάτων από τη συσκευή προς τον εξυπηρετητή κατά την επανεκπαίδευση για μία ημέρα.

Μέθοδος	Μέγεθος (bytes)
IL-SBR	13420
IL-SBR using Diff	13420 (first) 11572 (next)
FL-SS	3253

Πίνακας 7.2: Μέγεθος μηνυμάτων από τον εξυπηρετητή προς τη συσκευή κατά την επανεκπαίδευση για μία ημέρα.

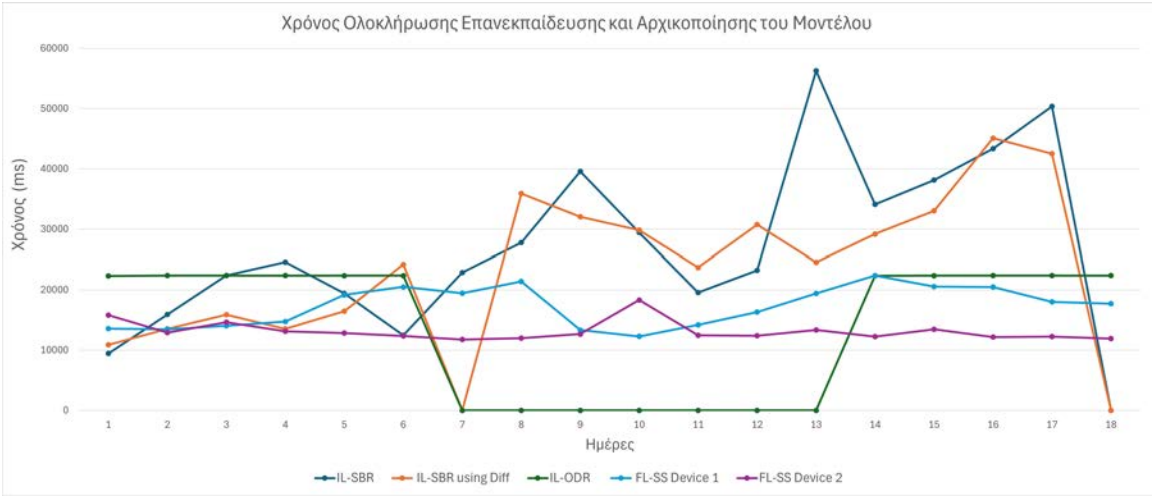
ηγούμενο μοντέλο για να πραγματοποιήσει σύγκριση και έτσι αποστέλλει ολόκληρο το αρχείο. Στις επόμενες επαναλήψεις, παραμένει ελαφρώς μικρότερο αλλά σταθερό. Αναμενόμενο θα ήταν αυτό να μειώνεται σταδιακά, καθώς θα υπήρχαν λιγότερες αλλαγές στο μοντέλο. Ωστόσο, αυτό δεν παρατηρείται, γεγονός που υποδηλώνει ότι κάθε φορά εμφανίζεται μεγάλος αριθμός αλλαγών, οι οποίες καλύπτουν όλους τους buffers του αρχείου. Έτσι, το μέγεθος του μηνύματος παραμένει περίπου σταθερό. Για τη μέθοδο **FL-SS**, παρατηρείται ότι τα μηνύματα είναι ελαφρώς μεγαλύτερα από αυτά που αποστέλλει η συσκευή στην αντίστοιχη περίπτωση. Αυτό είναι αναμενόμενο, καθώς ο εξυπηρετητής αποστέλλει τα βάρη του μοντέλου, όπως ακριβώς και οι συσκευές. Η μικρή αύξηση πιθανόν να οφείλεται στην `JSON` κωδικοποίηση του μηνύματος.

Όσον αφορά την απόδοση του μοντέλου σε κάθε κύκλο επανεκπαίδευσης, αυτή μετράται με τη χρήση του **RMSE (Ρίζα Μέσου Τετραγωνικού Σφάλματος)**. Γενικά, όσο μικρότερη είναι η τιμή του RMSE, τόσο καλύτερα προσαρμόζεται το μοντέλο στα δεδομένα. Στο διάγραμμα 7.2 απεικονίζεται η τάση της απόδοσης κάθε μεθόδου κατά τη διάρκεια των ημερών. Συγκεκριμένα, οι μέθοδοι **IL-SBR** και **IL-SBR με χρήση Diff** εμφανίζουν παρόμοια συμπεριφορά, γεγονός που επιβεβαιώνει ότι και στη δεύτερη περίπτωση η διαδικασία μεταβολής του μοντέλου πραγματοποιείται σωστά. Επίσης, η απόδοσή τους μπορεί να θεωρηθεί ικανοποιητική, καθώς σε ορισμένα σημεία παρατηρείται σημαντική βελτίωση, ενώ στα σημεία με αυξητική πορεία η κλίση είναι μικρή. Η μέθοδος **IL-ODR** παρουσιάζει φθίνουσα πορεία, με εξαίρεση ένα σημείο όπου καταγράφεται αύξηση (ημέρα 14). Αυτό πιθανόν οφείλεται σε αλλαγή της τάσης των δεδομένων, με αποτέλεσμα το μοντέλο να χρειάζεται να προσαρμοστεί. Επιπλέον, υπάρχουν αρκετές ημέρες κατά τις οποίες δεν απαιτείται επανεκπαίδευση,



Σχήμα 7.2: Διάγραμμα αναπαράστασης του RMSE του μοντέλου που προκύπτει μετά από κάθε επανεκπαίδευση.

με συνέπεια να μην προκύπτει τιμή για το RMSE. Τέλος, οι συσκευές που εκτελούν τη μέθοδο **FL-SS** παρουσιάζουν υψηλότερες τιμές σε σύγκριση με τις μεθόδους **IL-SBR**. Ωστόσο, ακολουθούν μια μικρή καθοδική πορεία. Οι αντίστοιχες τιμές και στις δύο συσκευές εμφανίζονται σχεδόν παρόμοιες, γεγονός που είναι αναμενόμενο, καθώς και οι δύο λαμβάνουν το ίδιο μοντέλο (βάρη) από τον εξυπηρετητή. Οι μικρές αποκλίσεις προέρχονται από την εκπαίδευση που ακολουθεί, με τα δεδομένα που είχε συλλέξει η κάθε συσκευή κατά τη διάρκεια της ημέρας.



Σχήμα 7.3: Διάγραμμα αναπαράστασης του χρόνου που απαιτείται για την ολοκλήρωση της επανεκπαίδευσης και της αρχικοποίησης του μοντέλου για κάθε μέθοδο

Στο διάγραμμα 7.3 παρουσιάζει τον **χρόνο που απαιτείται για την ολοκλήρωση της επανεκπαίδευσης και την αρχικοποίηση του μοντέλου** για κάθε μέθοδο, κατά την πάροδο των ημερών. Οι μέθοδοι **IL-SBR** και **IL-SBR με χρήση Diff** εμφανίζουν παρόμοια αυξητική τάση, γεγονός που δείχνει ότι μετά από κάθε επανεκπαίδευση απαιτείται περισσότερος χρόνος μέχρι το μοντέλο να είναι εκ νέου έτοιμο προς χρήση. Αυτό πιθανόν οφείλεται στην αυξανόμενη πολυπλοκότητα του, καθώς κάθε φορά ο εξυπηρετητής χρησιμοποιεί ολοένα και περισσότερα δεδομένα για να το εκπαιδεύσει. Για τη μέθοδο **IL-ODR**, ο χρόνος αρχικοποίησης παραμένει σταθερός, εκτός από τις περιπτώσεις όπου δεν απαιτείται επανεκπαίδευση, οπότε είναι μηδενικός. Η σταθερότητα αυτή εξηγείται από το γεγονός ότι η επανεκπαίδευση πραγματοποιείται απευθείας στη συσκευή, χωρίς να μεσολαβεί αναμονή από τον εξυπηρετητή, και κάθε φορά διαρκεί 50 επαναλήψεις (*epochs*). Αντίθετα, η μέθοδος **FL-SS** καταγράφει μικρούς χρόνους, καθώς η αρχικοποίηση του μοντέλου είναι εξαιρετικά γρήγορη, αφού περιορίζεται στην τοποθέτηση των βαρών στις αντίστοιχες δομές του μοντέλου. Επιπλέον, η τοπική επανεκπαίδευσή του είναι πάντα σταθερή χρονικά, καθώς κάθε φορά χρησιμοποιείται ο ίδιος αριθμός δεδομένων (αυτά που συλλέχθηκαν κατά τη διάρκεια μιας ημέρας λειτουργίας) και ο ίδιος αριθμός επαναλήψεων (*epochs*). Ο μεγαλύτερος χρόνος αναμονής σε αυτή την περίπτωση μπορεί να προκύψει όταν οι συσκευές περιμένουν να ολοκληρωθεί η αποστολή των βαρών από όλες τις συμμετέχουσες συσκευές, ώστε να συνεχιστεί η διαδικασία.

Μια ακόμη μετρική που χρησιμοποιήθηκε αφορά την **κατανάλωση CPU και μνήμης**, τόσο στη συσκευή όσο και στον εξυπηρετητή. Αυτή φαίνεται στους πίνακες 7.3, 7.4, 7.5 και τα αποτελέσματα αναλύονται παρακάτω.

Όσον αφορά την υπολογιστική κατανάλωση, σε όλες τις μεθόδους πραγματοποιήθηκε μία αρχική μέτρηση, ώστε να προσδιοριστεί το ποσοστό χρήσης όταν η συσκευή βρίσκεται σε κατάσταση αδράνειας (idle) και όταν εκτελεί τις καθημερινές της ενέργειες (συλλογή δεδομένων και προβλέψεις). Στην πρώτη περίπτωση καταγράφηκε ποσοστό **26%**, γεγονός που δείχνει ότι ακόμη και χωρίς ενεργές λειτουργίες, οι εσωτερικές διεργασίες του συστήματος (π.χ. λειτουργικό) απορροφούν σημαντικό μέρος της επεξεργαστικής ισχύος. Κατά την κανονική λειτουργία μέσα στην ημέρα, η κατανάλωση φτάνει το **95,6%**, ποσοστό ιδιαίτερα υψηλό που υποδηλώνει πως τόσο η συλλογή δεδομένων από τον αισθητήρα όσο και η εκτέλεση του μοντέλου είναι απαιτητικές διαδικασίες για το σύστημα. Κατά την επανεκπαίδευση η κατανάλωση αυξάνεται κατά **4%** σε σχέση με τη ημερήσια λειτουργία, καθώς ταυτόχρονα με τις διαδικασίες επικοινωνίας με τον διακομιστή (όπου απαιτείται) και την αρχικοποίηση

Κατάσταση	**Συσκευή**
Idle	26.00%
Κανονική λειτουργία	95.60%

Πίνακας 7.3: Κατανάλωση CPU στην συσκευή σε βασικές καταστάσεις λειτουργίας για όλες τις μεθόδους.

Μέθοδος	**Συσκευή**	**Server**
IL-SBR	99.80%	0.00%
IL-SBR using Diff	99.90%	1.50%
IL-ODR	99.90%	-
FL-SS Dev. 1	99.80%	0.00%
FL-SS Dev. 2	99.80%	0.00%

Πίνακας 7.4: Κατανάλωση CPU κατά τη διαδικασία της επανεκπαίδευσης ανά ημέρα.

Μέθοδος	**Συσκευή**	**Server**
IL-SBR	$\sim$ 25.8%	45–51%
IL-SBR using Diff	$\sim$ 31%	53–58%
IL-ODR	$\sim$ 42.5%	-
FL-SS Dev. 1	$\sim$ 42.3%	63–64%
FL-SS Dev. 2	$\sim$ 40.9%	63–64%

Πίνακας 7.5: Κατανάλωση μνήμης κατά τη διαδικασία της επανεκπαίδευσης ανά ημέρα.

του μοντέλου, συνεχιζουν κανονικα να εκτελούνται και οι προβλέψεις, γεγονός που οδηγεί στη συνολική επιβάρυνση. Αντίθετα, στον εξυπηρετητή, ο οποίος λειτουργεί μόνο όταν χρειάζεται επανεκπαίδευση, η χρήση της CPU κυμαίνεται μεταξύ **0,0% και 1,5%**, κάτι που υποδηλώνει πολύ μικρότερο φόρτο εργασίας..

Σχετικά με τη μνήμη, η κατανάλωση στη συσκευή παραμένει σταθερή για όλες τις μεθόδους κατά τη διαδικασία της επανεκπαίδευσης ανά ημέρα. Ωστόσο, οι δύο πρώτες μέθοδοι παρουσιάζουν μικρότερη τιμή σε σχέση με τις υπόλοιπες. Αυτό μπορεί να αιτιολογηθεί από το γεγονός ότι στις πρώτες, η συσκευή διατηρεί αποθηκευμένο μόνο τον πίνακα με τις ημερήσιες τιμές, ενώ στις υπόλοιπες, πέρα από αυτόν, απαιτείται να υπάρχουν και όλα τα εσωτερικά

δεδομένα του μοντέλου, όπως οι πίνακες των βαρών. Επίσης, για τον εξυπηρετητή ισχύει ότι η κατανάλωση ξεκινά από μια αρχική τιμή και αυξάνεται ελαφρώς με την πάροδο του χρόνου. Εξαίρεση αποτελεί η μέθοδος **FL-SS**, όπου η χρήση μνήμης του παραμένει σταθερή, καθώς δεν αποθηκεύονται δεδομένα τοπικά αλλά γίνεται χρήση μόνο των πληροφοριών που λαμβάνονται από τις συσκευές σε πραγματικό χρόνο.

7.6 Γενική αξιολόγηση κάθε μεθόδου

Από τα παραπάνω διαγράμματα και τους πίνακες προκύπτει μία πιο ξεκάθαρη εικόνα για την απόδοση αλλά και τα μειονεκτήματα κάθε μεθόδου.

Η μέθοδος **IL-SBR** εμφανίζει σταθερή έως ελαφρώς καθοδική τάση στον αριθμό των ημερήσιων λανθασμένων προβλέψεων. Από περαιτέρω δοκιμές φαίνεται πως, μετά από κάποιο χρονικό διάστημα, το μοντέλο μπορεί να φτάσει σε σημείο όπου δεν θα απαιτείται πλέον επανεκπαίδευση. Η απόδοση του μοντέλου παρουσιάζει επίσης σταθερή έως καθοδική πορεία. Ωστόσο, το πλήθος δεδομένων που αποστέλλει στον εξυπηρετητή είναι αρκετά μεγαλύτερο σε σχέση με τις υπόλοιπες μεθόδους, αν και παραμένει σταθερό. Επιπλέον, με την πάροδο των ημερών το μοντέλο γίνεται πιο περίπλοκο, γεγονός που αυξάνει τον χρόνο που απαιτείται τόσο για την παραγωγή του στον εξυπηρετητή όσο και για την αρχικοποίησή του στη συσκευή.

Η μέθοδος **IL-SBR με χρήση Diff** παρουσιάζει βελτίωση σε σχέση με την προηγούμενη ως προς το μέγεθος του μηνύματος που αποστέλλεται από τον εξυπηρετητή στη συσκευή. Ωστόσο, η μείωση δεν είναι τόσο σημαντική όσο αναμενόταν. Αυτό οφείλεται στο γεγονός ότι τελικά, κάθε φορά προκύπτει μεγάλος αριθμός αλλαγών, οι οποίες καλύπτουν τα περισσότερα buffer. Για τον λόγο αυτό, το πλήθος των δεδομένων που αποστέλλονται παραμένει σημαντικό. Παράλληλα, βελτίωση εντοπίζεται και στον χρόνο αρχικοποίησης, καθώς η διαδικασία απαιτεί την αποστολή μόνο τμημάτων του αρχείου, γεγονός που την καθιστά ταχύτερη. Σε όλες τις υπόλοιπες μετρικές η μέθοδος εμφανίζει παρόμοια συμπεριφορά με την IL-SBR.

Η μέθοδος **IL-ODR** δεν διαθέτει μετρικές που σχετίζονται με τον εξυπηρετητή, καθώς λειτουργεί πλήρως τοπικά χωρίς επικοινωνία μαζί του. Όσον αφορά το ποσοστό των λανθασμένων προβλέψεων μέσα στην ημέρα αλλά και το RMSE, παρουσιάζει χαμηλότερες τιμές από όλες τις υπόλοιπες μεθόδους, γεγονός που της δίνει προβάδισμα. Ωστόσο, μειονεκτεί

στο ότι περιορίζεται από τους υπολογιστικούς πόρους της συσκευής, με αποτέλεσμα να υποστηρίζει μικρότερη πολυπλοκότητα μοντέλων. Ο χρόνος ολοκλήρωσης της επανεκπαίδευσης και της αρχικοποίησης του μοντέλου παραμένει σταθερός, αφού κάθε κύκλος απαιτεί μόνο εκπαίδευση του μοντελου πάνω στα νέα ημερίσια δεδομένα.

Τέλος, η μέθοδος **FL-SS** εμφανίζει μεγαλύτερο, αλλά φθίνων, ποσοστό λανθασμένων προβλέψεων, και παράλληλα υψιλή τιμή RMSE, σε σχέση με τις υπόλοιπες. Το γεγονός αυτό υποδηλώνει ότι η μέθοδος απαιτεί περισσότερο χρόνο ώστε να φτάσει σε ικανοποιητικό επίπεδο απόδοσης. Στο συγκεκριμένο πείραμα, τα δεδομένα κάθε ημέρας κατανέμονται εναλλάξ στις συσκευές, οπως εχει περιγραφει στο υποκεφαλαιο 7.4, με αποτέλεσμα σε κάθε κύκλο επανεκπαίδευσης το τελικό μοντέλο να ενσωματώνει τη συνολική γνώση της ημέρας. Ο χρόνος ολοκλήρωσης της διαδικασιας είναι σαφώς μικρότερος σε σχέση με τις υπόλοιπες μεθόδους, καθώς απαιτεί μόνο την αντικατάσταση των βαρών στους πίνακες και την εκπαιδευση του μοντελου με τα ημεριια συλεχθεντα δεδοεμενα.

7.7 Προτιμότερη υλοποίηση

Μετά τους πειραματισμούς που υλοποιήθηκαν, διαπιστώθηκε ότι η πιο αποδοτική (σε συνδυασμό κυρίως του αριθμού λανθασμένων προβλέψεων και του MSE) μέθοδος είναι η IL-ODR. Παρ' όλα αυτά, η λύση αυτή είναι αρκετά περιοριστική, καθώς το μοντέλο πρέπει να είναι ιδιαίτερα ελαφρύ τόσο σε απαιτήσεις μνήμης όσο και σε υπολογιστική ισχύ, προκειμένου να μπορεί να εκτελεστεί στο ESP32. Σε περίπτωση όμως που χρησιμοποιηθεί ένα πιο ισχυρό ενσωματωμένο σύστημα με περισσότερους πόρους, θα μπορούσε να υποστηριχθεί και ένα πιο απαιτητικό μοντέλο με μεγαλύτερη άνεση.

Αντίθετα, οι δύο πρώτες μέθοδοι (IL-SBR και IL-SBR με χρήση Diff) δεν αντιμετωπίζουν αυτόν τον περιορισμό και επομένως μπορούν να αποτελέσουν μια πιο γενική λύση. Η χρήση του Diff, αν και δεν μειώνει σημαντικά το μέγεθος των δεδομένων που διακινούνται στο δίκτυο, προσφέρει έστω και μια μετρήσιμη βελτίωση, η οποία σε πρακτικές συνθήκες μπορεί να είναι ιδιαίτερα χρήσιμη. Επιπλέον, συγκρίνοντάς τες με τη μέθοδο FL-SS, προκύπτει από τα πειράματα ότι οι πρώτες παρουσιάζουν σαφώς καλύτερη απόδοση και οδηγούν σε πιο γρήγορα αποτελέσματα.

Ωστόσο, ανάλογα με το εκάστοτε σενάριο χρήσης και τους περιορισμούς που το συνοδεύουν, μπορεί να γίνει πιο στοχευμένη επιλογή μεθόδου. Σε γενικές γραμμές, όλες καταφέρνουν

να προσφέρουν αξιόλογα αποτελέσματα και να συγκλίνουν προς το επιθυμητό αποτέλεσμα.

Κεφάλαιο 8

Συμπεράσματα

Στο τελευταίο αυτό κεφάλαιο παρουσιάζεται η συνολική ανασκόπηση της μελέτης που προηγήθηκε, των μεθόδων που υλοποιήθηκαν και αξιολογήθηκαν, καθώς και των αποτελεσμάτων που προέκυψαν.

8.1 Σύνοψη και συμπεράσματα

Σε αυτή την εργασία δόθηκε έμφαση στην ανάγκη της επανεκπαίδευσης ενός μοντέλου μηχανικής μάθησης που εκτελείται σε ένα ενσωματωμένο σύστημα. Έχουν παρουσιαστεί και αξιολογηθεί τρεις λύσεις πάνω σε αυτό το πρόβλημα: IL-SBR και η βελτίωση του με χρήση Diff, όπου και στις δύο χρησιμοποιείται ένας εξυπηρετητής που εκτελεί όλη τη διαδικασία για την επανεκπαίδευση του μοντέλου και την αποστολή του στη συσκευή. Η βελτίωση έγκειται στο ότι ο αυτός δεν απαιτείται να επιστρέψει ολόκληρο το νέο μοντέλο, αλλά μόνο τις διαφοροποιήσεις σε σχέση με το προηγούμενο. Επίσης, έχει δημιουργηθεί η μέθοδος IL-ODR, όπου η επανεκπαίδευση εκτελείται πάνω στην ενσωματωμένη συσκευή ωστόσο, για να γίνει αυτό θα πρέπει το μοντέλο να είναι περιορισμένης δυναμικής. Ακόμα, έγινε και η υλοποίηση του FL-SS, το οποίο ακολουθεί τη λογική της ομοσπονδιακής μαθησης, με τη χρήση του εξυπηρετητή για τον συνδυασμό των βαρών του μοντέλου κάθε συσκευής που συμμετέχει στη διαδικασία.

Πραγματοποιήθηκαν ορισμένα πειράματα έτσι ώστε να γίνει σύγκριση και αξιολόγηση των παραπάνω μεθόδων. Συνοπτικά, οι μετρικές που χρησιμοποιήθηκαν για κάθε ημέρα είναι: ο αριθμός των λανθασμένων προβλέψεων, το μέγεθος των μηνυμάτων που διακινούνται, η απόδοση του μοντέλου, ο χρόνος αρχικοποίησης κάθε φορά, καθώς και οι απαιτήσεις σε

CPU και μνήμη που χρειάζονται σε κάθε κύκλο επανεκπαίδευσης.

Τα αποτελέσματα που προέκυψαν είναι αξιόλογα για όλες τις μεθόδους και, ανάλογα με τις ανάγκες, μπορεί να γίνει η πιο κατάλληλη επιλογή. Όσον αφορά την απόδοση του μοντέλου, το καλύτερο αποτέλεσμα έχει επιτύχει η μέθοδος IL-ODR, ωστόσο η χρήση της περιορίζεται σε πιο απλά μοντέλα. Για τον λόγο αυτό, αν το μοντέλο είναι πιο περίπλοκο, τότε προτείνεται η χρήση του IL-SBR με Diff.

8.2 Μελλοντικές επεκτάσεις

Όλη η μελέτη που παρουσιάστηκε στην παρούσα διπλωματική, δίνει το έναυσμά για να εξελιχθεί η επανεκπαίδευση των μοντέλων σε ενσωματωμένα συστήματα, μέσο της διερεύνησης και της υλοποίησης πιο συνθέτων περιπτώσεων. Κάποια από τα κομμάτια που μπορούν να αναλυθούν περεταίρω είναι:

1. **Βελτίωση της εκπαίδευσης μοντέλων πάνω στην συσκευή**, η οποία θα ήταν ιδιαίτερα χρήσιμη αν επεκτεινόταν και σε πιο σύνθετα μοντέλα με ευρεία χρήση στην καθημερινή ζωή. Έτσι, όλη η επεξεργασία θα εκτελείται στο άκρο του δικτύου (end-device), χωρίς να απαιτείται η ύπαρξη συστήματος με μεγάλη υπολογιστική ισχύ, καθιστώντας τη συσκευή αυτόνομη και έτοιμη να διαχειριστεί πολύπλοκα αιτήματα.

2. **Ενσωμάτωση συνεργασίας συσκευής με νέφος (edge–cloud)** με διερεύνηση υβριδικών μοντέλων όπου ένα μέρος της εκπαίδευσης γίνεται στην συσκευή και ένα μέρος στο νέφος, με στόχο την εξισορρόπηση στην απόκριση, στην ενεργειακή κατανάλωση και την ακρίβεια.

3. **Χρήση μεθόδων AutoML ή meta-learning** για την αυτόματη επιλογή των υπερπαραμέτρων (όρια λανθασμένων προβλέψεων, όρια για το πότε μια πρόβλεψη θεωρείται λανθασμένη κ.λπ.), ώστε να μειωθεί η ανάγκη χειροκίνητης ρύθμισης.

4. Ενσωμάτωση τεχνικών, ώστε να **διασφαλιστεί η ασφάλεια** των δεδομένων που διακινούνται στο δίκτυο. Η προσθήκη αυτή θα ωφελήσει κυρίως τις δύο πρώτες τεχνικές (IL-SBR και IL-SBR με Diff), καθώς αποστέλλουν όλα τα ημερήσια δεδομένα τους στον εξυπηρετητή, τα οποία ενδέχεται να περιέχουν σημαντικές πληροφορίες.

5. **Ανάπτυξη μηχανισμού που να επιλέγει δυναμικά την καταλληλότερη μέθοδο** (IL-SBR, IL-ODR, FL-SS) με βάση τους πόρους της συσκευής, τις συνθήκες του δικτύου και τις απαιτήσεις του μοντέλου.

6. **Εφαρμογή των μεθόδων σε έξυπνες συσκευές καθημερινής χρήσης**, ώστε να γίνονται ολοένα και πιο «έξυπνες», ενημερώνοντας το μοντέλο τους όποτε χρειάζεται.

Βιβλιογραφία

[1] Simone Disabato and Manuel Roveri. Tiny machine learning for concept drift. 2021. Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano.

[2] Zainab Al-Qaysi and Jamal Al-Khateeb. An overview of machine learning classification techniques. 2023.

[3] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems. 2020. Google Research, TinyML Foundation.

[4] Mehmet Kemal Gullu Ferhat Ozgur Catak Maliha Tabassum Betul Yurdem, Murat Kuzlu. Federated learning: Overview, strategies, applications, tools and future directions. 15 October 2024. Department of Electrical and Electronics Engineering, Izmir Bakircay University, Izmir, Turkey,Batten College of Engineering and Technology, Old Dominion University, Norfolk, VA, USA, Department of Electrical Engineering and Computer Science, University of Stavanger, Rogaland, Norway.

[5] Jiamin Fan, Kui Wu, Yang Zhou, Zhengan Zhao, and Shengqiang Huang. Fast model update for iot traffic anomaly detection with machine unlearning. *IEEE Internet of Things Journal*, 10(10):8590–8602, 2023.

[6] Arijit Sehanobish Avinava Dubey Snigdha Chaturvedi Somnath Basu Roy Chowdhury, Krzysztof Choromanski. Towards scalble exact machine unlearning using parameter-efficient fine-tuning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2024.

[7] Espressif Systems. Esp-idf programming guide (v5.5.1). `https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html`.

[8] Atomic14. Tensorflow lite esp32 firmware source code. `https://github.com/atomic14/tensorflow-lite-esp32/tree/master/firmware/src`.

[9] Nil Llisterri Giménez, Marc Monfort Grau, Roger Pueyo Centelles, and Felix Freitag. On-device training of machine learning models on microcontrollers with federated learning. *Electronics*, 11(4):573, 2022.

[10] E. Milite F. Palmieri R. Pietrantuono S. Russo M. Ficco, A. Guerriero. Federated learning for iot devices: Enhancing tinyml with on-board training. *Engineering Applications of Artificial Intelligence*, 2024. ScienceDirect; combines federated learning and transfer learning to enable on-board ML training in IoT devices.

[11] Marc Monfort Grau, Roger Pueyo Centelles, and Felix Freitag. On-device training of machine learning models on microcontrollers with a look at federated learning. *Technical University of Catalonia, Barcelona, Spain*, 2025. marc.monfort@gmail.com, rpueyo@ac.upc.edu, felix@ac.upc.edu.

[12] Nil Llisterri and Marc Monfort. Federated learning with arduino nano 33 ble sense. `https://github.com/NilLlisterri/TinyML-FederatedLearning`, 2021.