ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Ανάλυση και σύγκριση αλγορίθμων συσσωμάτωσης για βελτιστοποίηση τεχνικών Ομοσπονδιακής Μάθησης σε cross-silo σενάρια

Φοιτητής: Λεωνίδας Μαζαράκης
ΑΜ: 50346515

Επιβλέπων Καθηγητής

Γρηγόριος Κουλούρας
Αναπληρωτής Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2023

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Analysis and comparison of aggregation algorithms for optimizing Federated Learning techniques in cross-silo scenarios

Student: Leonidas Mazarakis
Registration Number: 50346515

Supervisor

Grigorios Koulouras
Associate Professor

ATHENS-EGALEO, OCTOBER 2023

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Γρηγόριος Κουλούρας, Αναπληρωτής Καθηγητής	**Σωτήριος Καραμπέτσος,** Αναπληρωτής Καθηγητής	**Ηλίας Ζώης,** Αναπληρωτής Καθηγητής
 Digitally signed by Grigorios Koulouras DN: c=GR, l=Athens, o=University of West Attica, ou=Class A - Private Key created and stored in hardware CSP, sn=Koulouras, givenName=Grigorios, serialNumber=6090016933, cn=Grigorios Koulouras, email=gregkoul@uniwa.gr Date: 2023.10.24 16:31:23 +03'00'	Sotirios Karampetsos Digitally signed by Sotirios Karampetsos Date: 2023.10.24 12:40:49 +03'00'	Ilias Zois Digitally signed by Ilias Zois Date: 2023.10.20 15:05:14 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Λεωνίδας Μαζαράκης του Παναγιώτη, με αριθμό μητρώου 50346515 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.»

Ο Δηλών
Λεωνίδας Μαζαράκης

LEONIDAS MAZARAKIS
Digitally signed by LEONIDAS MAZARAKIS
Date: 2023.10.20 10:23:48 +03'00'

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω εγκάρδια τους γονείς μου, Παναγιώτη και Ευανθία και τον αδερφό μου, Κωνσταντίνο, για την υποστήριξη τους και τις θυσίες τους. Χάρη σε εκείνους διατηρώ το κίνητρο να κυνηγάω τους στόχους και τα όνειρα μου.
Θα ήθελα να ευχαριστήσω πολύ τον καθηγητή μου, κ. Γρηγόριο Κουλούρα, για τη καθοδήγηση του, την υποστήριξη και τον χρόνο που αφιέρωσε σε συναντήσεις και προτάσεις. Ακόμα και για την εμπιστοσύνη του, με την ανάθεση ενός θέματος που φάνηκε ιδιαίτερα ενδιαφέρον.
Τέλος, τους φίλους μου για την δική τους υποστήριξη, τις εμπειρίες και τους προβληματισμούς που μοιραστήκαμε κατά την διάρκεια των σπουδών μου.

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την ανάλυση και την σύγκριση αλγορίθμων συσσωμάτωσης σε εφαρμογές Ομοσπονδιακής Μάθησης. Μέσω της εργασίας αυτής, επιδιώκεται η παρουσίαση ορισμένων από τους πλέον δημοφιλείς αλγόριθμους συσσωμάτωσης που χρησιμοποιούνται από τα κεντρικοποιημένα συστήματα, προκειμένου να γίνει συλλογή των δεδομένων των μοντέλων υπολογιστικής μάθησης που χρησιμοποιούνται από τα τοπικά συστήματα. Συγκεκριμένα εξετάζεται η εφαρμογή των αλγορίθμων σε cross-silo σενάρια Ομοσπονδιακής Μάθησης, όπου τα τοπικά συστήματα αποτελούνται από συστήματα οργανισμών με σχετικά μικρό αριθμό συμμετεχόντων. Επιμέρους στόχοι είναι η εισαγωγή στο συνολικό περιβάλλον της Ομοσπονδιακής Μάθησης, η σύγκριση και η αξιολόγηση των αλγορίθμων μέσω αποτελεσμάτων που προκύπτουν με διαφορετικές παραμέτρους εκπαίδευσης. Στο πλαίσιο της διπλωματικής εργασίας, η πρακτική υλοποίηση που αναπτύχθηκε προκειμένου να γίνουν οι συγκρίσεις, γίνεται μέσω προσομοίωσης πραγματικής εφαρμογής Ομοσπονδιακής Μάθησης η οποία αφορά ένα πρόβλημα ταξινόμησης εικόνων που λαμβάνονται από γνωστά σύνολα δεδομένων (Datasets). Η πρόβλεψη λύσεων του προβλήματος γίνεται με την χρήση δύο τεχνητών νευρωνικών δικτύων διαφορετικής αρχιτεκτονικής και πολυπλοκότητας, με σκοπό την παραγωγή περισσότερων αποτελεσμάτων που θα βοηθήσουν στην καλύτερη σύγκριση των αλγορίθμων. Η σύγκριση πραγματοποιείται μέσω εξέτασης των συντελεστών απόδοσης του συνολικού μοντέλου που παράγεται στο κεντρικό σύστημα.

Λέξεις – κλειδιά

Τεχνητή Νοημοσύνη, Ομοσπονδιακή Μάθηση, Αλγόριθμοι Συσσωμάτωσης, Υπολογιστική Μάθηση, Μηχανική Μάθηση, Τεχνητά Νευρωνικά Δίκτυα, Σύγκριση Αλγορίθμων, Cross-Silo, Cross-Device

Abstract

This thesis deals with the analysis and comparison of aggregation algorithms in Federated Learning applications. The aim of the present work is to present some of the most popular aggregation algorithms used by centralized systems in order to collect the data of computational learning models used by local systems. Specifically, the application of the algorithms to cross-silo Federated Learning scenarios is examined, where the local systems consist of organization systems with a relatively small number of participants. Sub-goals are to introduce the overall environment of Federated Learning, to compare and evaluate the algorithms through results obtained with different training parameters. In the context of the thesis, the practical implementation developed in order to make the comparisons, is done by simulating a real application of Federated Learning which concerns a problem of classification of images obtained from known datasets. The prediction of solutions to the problem is done by using two artificial neural networks of different architecture and complexity, in order to produce more results that will help to better compare the algorithms. The comparison is carried out by examining the coefficients of performance of the overall model produced in the central system.

Keywords

Artificial Intelligence, Federated Learning, Aggregation Algorithms, Machine Learning, Artificial Neural Networks, Comparison of Algorithms, Cross-Silo Federated Learning, Cross-Device Federated Learning

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Κατάλογος Διαγραμμάτων

Αλφαβητικό Ευρετήριο

Ακρωνύμιο	**Περιγραφή**
CNN	Convolutional Neural Network
CV	Computer Vision
FedAvg	Federated Average Algorithm
FedNova	Federated Normalized Averaging Algorithm
FedNova	Federated Normalized Averaging Algorithm
FedProx	Federated Algorithm (with Proximal Term)
FL	Federated Learning
FTL	Federated Transfer Learning
GPU	Graphics Processing Unit
HFL	Horizontal Federated Learning
IID Data	Independent and Identically Distributed Data
ResNet	Residual Neural Network
SGD	Stochastic Gradient Descent Algorithm
VFL	Vertical Federated Learning
ΚΟΜ	Κάθετη Ομοσπονδιακή Μάθηση
ΟΜ	Ομοσπονδιακή Μάθηση
ΟΜΜ	Ομοσπονδιακή Μάθηση Μεταφοράς
ΟΟΜ	Οριζόντια Ομοσπονδιακή Μάθηση
ΥΟ	Υπολογιστική Όραση

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια η ανάπτυξη της τεχνητής νοημοσύνης έχει δημιουργήσει μια αμεσότερη σχέση του κόσμου με την τεχνολογία, επιλύοντας απλά καθημερινά ή ακόμα και πιο σύνθετα προβλήματα. Πλέον με την ανάπτυξη τόσο των υπολογιστικών πόρων που είναι διαθέσιμοι, όσο και των θεωρητικών μεθόδων παρατηρείται αυξημένη χρήση της τεχνητής νοημοσύνης είτε άμεσα, είτε έμμεσα μέσω των συσκευών που χρησιμοποιούμε σε διαδικασίες που καν δεν αντιλαμβανόμαστε. Ωστόσο, για να επιλυθούν τα ζητήματα αυτά, όπως οποιοδήποτε πρόβλημα, απαιτούνται δεδομένα προκειμένου να δοθεί κάποια απάντηση.

Οι κλασικές μέθοδοι υπολογιστικής μάθησης, χρησιμοποιούν ενιαία μοντέλα υπολογιστικής μάθησης που υλοποιούνται σε κεντρικοποιημένα υπολογιστικά συστήματα πολύ υψηλών δυνατοτήτων. Προκειμένου να γίνει η εκπαίδευση των μοντέλων αυτών απαιτείται η μεταφορά μεγάλου όγκου δεδομένων, η οποία όμως δημιουργεί προβλήματα. Τα προβλήματα αυτά υπόκεινται τόσο στο διαμοιρασμό πολύτιμων προσωπικών δεδομένων όσο και στα τεχνικά θέματα που απαιτούνται γι’ αυτό. Αρχικά, τα αυστηρά νομοθετικά πλαίσια τα οποία υπάρχουν και οριοθετούν την αποστολή προσωπικών δεδομένων (GDPR, CCPA, PIPEDA, κ.ά.), αλλά και η ίδια η επιφυλακτικότητα των ατόμων πολλές φορές να μοιραστούν τα δεδομένα τους δυσχεραίνουν εκ των προτέρων την διαδικασία. Επιπλέον είναι απαραίτητοι μεγάλοι διαδικτυακοί πόροι για την μεταφορά των δεδομένων αλλά και μεγάλη αποθηκευτική και υπολογιστική ικανότητα στα κεντρικοποιημένα συστήματα, αφού θα πρέπει να γίνει τόσο αποθήκευση των δεδομένων όσο και απαιτητική εκπαίδευση μεγάλων μοντέλων καθώς θα χρησιμοποιούν όλα αυτά τα δεδομένα.

Τα παραπάνω προβλήματα μπορούν να αντιμετωπιστούν χρησιμοποιώντας την Ομοσπονδιακή Μάθηση (ΟΜ) όπου στη διεθνή βιβλιογραφία χρησιμοποιείται ο όρος Federated Learning (FL). Η Ομοσπονδιακή Μάθηση αποτελεί ένα είδος υπολογιστικής μάθησης κατά την οποία δεν γίνεται μεταφορά των δεδομένων που χρησιμοποιούνται για την εκπαίδευση. Ο όρος Ομοσπονδιακή Μάθηση αναφέρθηκε για πρώτη φορά το 2016 από ερευνητές της Google [1]. Σύμφωνα με την αρχιτεκτονική αυτής της προσέγγισης, η κάθε συσκευή εκπαιδεύει ένα τοπικό μοντέλο υπολογιστικής μάθησης με δεδομένα που είτε έχουν συλλεχθεί από την ίδια, είτε παρέχονται σε αυτή και μόνο χωρίς την μεταφορά τους σε οποιαδήποτε άλλη συσκευή ή σύστημα. Έτσι τα δεδομένα με τα οποία γίνεται η εκπαίδευση διατηρούνται απόρρητα. Η επικοινωνία των επιμέρους αυτών συσκευών που πραγματοποιούν τοπική εκπαίδευση με κεντρικοποιημένο σύστημα αφορά μόνο την κοινοποίηση των παραμέτρων των εκπαιδευμένων μοντέλων. Αυτό γίνεται με σκοπό την συλλογή αυτών των παραμέτρων και την δημιουργία από αυτά, ενός νέου μοντέλου, για ανατροφοδότηση στα τοπικά συστήματα για βελτίωση της συνολικής διαδικασίας της εκπαίδευσης.

Αντικείμενο της διπλωματικής εργασίας

Στη παρούσα διπλωματική εργασία αναλύεται το περιβάλλον της Ομοσπονδιακής Μάθησης με έμφαση στην συγκέντρωση των παραμέτρων των μοντέλων μέσω των αλγορίθμων συσσωμάτωσης. Συγκεκριμένα γίνεται ερμηνεία και σύγκριση των αλγορίθμων αυτών για εφαρμογή σε σενάρια cross-silo. Σε τέτοιες περιπτώσεις, η Ομοσπονδιακή Μάθηση πραγματοποιείται με την συμμετοχή συστημάτων "οργανισμών". Αυτό σημαίνει ότι συμμετέχει μικρός σχετικά αριθμός συμμετεχόντων στην διαδικασία της εκπαίδευσης με μικρή πιθανότητα προβλήματος συνδεσιμότητας και επικοινωνίας με το κεντρικό σύστημα. Η σύγκριση και η αξιολόγηση γίνεται με τα τελικά δεδομένα και αποτελέσματα απόδοσης που προέρχονται από το κεντρικό μοντέλο, που δημιουργείται από τις παραμέτρους των τοπικών συστημάτων, για την λήψη και την συγκέντρωση των οποίων είναι υπεύθυνοι οι αλγόριθμοι συσσωμάτωσης που εξετάζονται.

Σκοπός και στόχοι

Ο σκοπός της εργασίας είναι η ανάλυση και η σύγκριση των αλγορίθμων συσσωμάτωσης μέσω διαφορετικών διατάξεων, έτσι ώστε να βγουν συμπεράσματα από τις συγκρίσεις σχετικά με την επιλογή του κατάλληλου αλγόριθμου συσσωμάτωσης ανά περίπτωση και των παραμέτρων της εκπαίδευσης που εφαρμόζονται με αυτόν. Επιμέρους στόχοι είναι η εξοικείωση με την υλοποίηση εφαρμογών Ομοσπονδιακής Μάθησης και η μελέτη των διαφόρων επιμέρους κομματιών της, αλλά και των προβλημάτων που δημιουργούνται.

Καινοτομία

Η καινοτομία της παρούσας εργασίας έγκειται στην σύγκριση των αλγορίθμων για σενάρια cross-silo με συγκεκριμένες παραμέτρους και στην αξιολόγηση της εφαρμογής τους. Η εργασία συνεισφέρει στη μελέτη των cross-silo σεναρίων. Παρουσιάζει τα αποτελέσματα για τρεις αλγόριθμους σε συγκεκριμένα σενάρια, προκειμένου να βγει ένα συμπέρασμα σχετικά με την καταλληλόλητα του κάθε αλγορίθμου ανάλογα με την εφαρμογή που χρησιμοποιείται.

Μεθοδολογία

Η μεθοδολογία που ακολουθήθηκε σε αυτή τη μελέτη, σχετίζεται με την προσομοίωση πειραματικής εφαρμογής Ομοσπονδιακής Μάθησης σε ένα πρόβλημα ταξινόμησης εικόνων, έτσι ώστε να μπορεί να γίνει ερμηνεία των διαφορετικών αποτελεσμάτων που παράγονται σε κάθε διαφορετική παραμετροποίηση.

Δομή

Η εργασία αποτελείται από 2 ενότητες. Στη πρώτη που αποτελείται από 2 κεφάλαια αναλύεται το θεωρητικό υπόβαθρο που απαιτείται και γίνεται εισαγωγή με τους γενικούς όρους και τις λειτουργίες που πρέπει να είναι γνωστές για να πραγματοποιηθεί το πρακτικό μέρος και να εξηγηθεί εκεί εκτενέστερα τι, γιατί και πώς χρησιμοποιήθηκε. Η δεύτερη ενότητα με 5 κεφάλαια έχει να κάνει με την ίδια την υλοποίηση της προσομοίωσης υλοποίησης Ομοσπονδιακής Μάθησης. Ειδικότερα, στην πρώτη ενότητα γίνεται αναφορά και επεξήγηση στις έννοιες και τα μέρη της τεχνητής νοημοσύνης, και της υπολογιστικής μάθησης. Ιδιαίτερη έμφαση δίνεται στην εισαγωγή στο περιβάλλον της Ομοσπονδιακής Μάθησης και την ανάλυση των λειτουργιών που πραγματοποιούνται προκειμένου να επιτευχθεί η εφαρμογή της. Αναλύεται η συνολική δομή της και ειδικότερα οι λειτουργίες που πραγματοποιούνται ανά γύρο επικοινωνίας με το κεντρικό σύστημα. Στην συνέχεια στην δεύτερη ενότητα ακολουθούν τα πιο πρακτικά κεφάλαια που έχουν να κάνουν με την ίδια την υλοποίηση που πραγματοποιήθηκε προκειμένου να γίνουν οι συγκρίσεις των αλγορίθμων και ερμηνεύονται εκ νέου τα δομικά στοιχεία της πειραματικής εφαρμογής. Γίνεται επεξήγηση της συνολικής διαδικασίας που ακολουθήθηκε προκειμένου να γίνει η προσομοίωση της εφαρμογής Ομοσπονδιακής Μάθησης. Ακόμα παρουσιάζονται και αναλύονται οι αλγόριθμοι που συγκρίθηκαν και ακολουθούν τα πειραματικά αποτελέσματα για κάθε διαφορετική υλοποίηση που πραγματοποιήθηκε. Τέλος, μέσω των αποτελεσμάτων που λαμβάνονται, εξάγονται συμπεράσματα και προτείνεται ποιος αλγόριθμος και ποιοι παράμετροι θα πρέπει να επιλέγονται για την βέλτιστη υλοποίηση έτσι ώστε να παραχθούν τα καλύτερα αποτελέσματα από αυτά που συγκρίθηκαν.

ΚΕΦΑΛΑΙΟ 1 ° : Τεχνητή Νοημοσύνη – Υπολογιστική Μάθηση

1.1 Τεχνητή Νοημοσύνη

Σήμερα ολοένα και περισσότερο η τεχνολογία στρέφεται γύρω από την τεχνητή νοημοσύνη. Τόσο σε επιστημονικό αλλά και σε επίπεδο εφαρμογής, η τεχνητή νοημοσύνη και η μηχανική μάθηση κατακτούν σημαντικό μερίδιο. Η τεχνητή νοημοσύνη αποτελεί έναν από τους νεότερους, πιο διαδεδομένους αλλά και πιο προσελκυστικούς ερευνητικούς τομείς στον χώρο αφού αυξάνεται συνεχώς το ενδιαφέρον για ενασχόληση από ερευνητές και επιστήμονες [2]. Η ακριβής περιγραφή του όρου της τεχνητής νοημοσύνης είναι αρκετά δύσκολη αφού αρκετά δύσκολη είναι και η περιγραφή του όρου της νοημοσύνης γενικότερα.

Ξεκινώντας από το 1950 και το παιχνίδι της μίμησης που είχε προταθεί από τον Alan Turing, γίνεται για πρώτη φορά αναφορά σε παραγωγή σκέψης από μηχανή. Εκεί προτείνεται η διεξαγωγή μιας δοκιμασίας (Turing Test) μέσω της οποίας αν η μηχανή κάνει έναν άνθρωπο που διεξάγει ερωτήσεις, να μην μπορεί να αντιληφθεί αν πρόκειται για μηχανή ή κάποιον άλλο άνθρωπο που απαντάει τις ερωτήσεις, τότε η μηχανή αυτή θα πρέπει να θεωρείται έξυπνη [3], [2]. Το 1956 θα δοθεί ο πρώτος ορισμός της τεχνητής νοημοσύνης από τον John McCarthy, διατυπώνοντας ότι η τεχνητή νοημοσύνη είναι η παροχή άδειας σε μία μηχανή να προσομοιώσει την ανθρώπινη ευφυΐα όσο το δυνατό καλύτερα.

Η τεχνητή νοημοσύνη αποτελεί ένα μεγάλο φάσμα υποκατηγοριών που την απαρτίζουν. Συστατικά της είναι οι τομείς της λογικής, της γνώσης, των μαθηματικών, της γλωσσικής κατανόησης και επεξεργασίας, της κατανόησης του περιβάλλοντος με εφαρμογές στην ρομποτική, στην ανάπτυξη αυτόνομων συστημάτων και πολλών άλλων. Τις βάσεις για την δημιουργία της έθεσαν αρχικά ερωτήματα που είχαν να κάνουν με τα παραπάνω και στις περισσότερες περιπτώσεις απαντήθηκαν μέσω των μαθηματικών αλλά και μέσω πιο εφαρμοσμένων τομών όπως η βιολογία, τα οικονομικά και η επιστήμη των υπολογιστών. Ξεκινώντας από φιλοσοφικά ερωτήματα και την θεωρία της λογικής που έχει την αφετηρία της στα αρχαία χρόνια και τον Αριστοτέλη, συνεχίστηκε με λοιπούς φιλοσοφικούς συλλογισμούς και αργότερα αναπτύχθηκε από μαθηματικούς με σημαντικά ορόσημα, όπως η συνεισφορά του George Boole κ.ά.. Επιπλέον σημαντικές προσφορές ήταν η επιστήμη των πιθανοτήτων, με ειδική μνεία στην δουλειά του Thomas Bayes και την συνεισφορά που είχε στην ανάπτυξη του κλάδου, η βιολογία και η νευροεπιστήμη που “δανείζει” πρακτικές του ανθρώπινου εγκεφάλου στις μηχανές, η ψυχολογία και ειδικότερα η γνωστική ψυχολογία χαρίζοντας τεχνικές για την δημιουργία της λειτουργίας της αντίληψης και την προσομοίωση της ουσιαστικά από τεχνητά κατασκευάσματα. Πολύ σημαντική είναι η ανάπτυξη των υπολογιστών, και η συμβολή τους στο πεδίο της τεχνητής νοημοσύνης, αφού εκ των πραγμάτων αποτελούν το μέσο από το οποίο μπορεί να εφαρμοστεί η τεχνητή νοημοσύνη και τέλος η θεωρία ελέγχου και αυτομάτου ελέγχου, παρέχοντας σημαντικά βοηθήματα επεξήγησης και επίλυσης προβλημάτων [2].

1.2 Υπολογιστική Μάθηση

Ίσως το σημαντικότερο κομμάτι της τεχνητής νοημοσύνης είναι η υπολογιστική μάθηση. Αποτελεί φυσική εξέλιξη αφού η νοημοσύνη γενικότερα εμπεριέχει σε πολύ σημαντικό βαθμό την μάθηση. Ο όρος διατυπώθηκε για πρώτη φορά το 1959 από τον Arthur Samuel ο οποίος αναφέρθηκε σε «ένα υπολογιστικό πρόγραμμα το οποίο μπορεί να κάνει παραγωγή μίας συμπεριφοράς χωρίς να έχει προγραμματιστεί αναλυτικά για αυτό» [4].

Η εξέλιξη της επιστήμης και των τεχνικών που απαιτούνται αλλά και η ανάπτυξη των υπολογιστικών δυνατοτήτων έχει φέρει πλέον την υπολογιστική μάθηση στο προσκήνιο, αφού μπορούν να επιλυθούν προβλήματα που με την κλασσική χρήση υπολογιστικών συστημάτων ήταν αδύνατη. Ακόμα όμως και σε ζητήματα που θα μπορούσε να δοθεί απάντηση, με χρήση της υπολογιστική μάθησης, αν προηγηθεί η διαδικασία της εκπαίδευσης μπορούν να παραχθούν αποτελέσματα εξαιρετικά γρήγορα και πολύ πιο αποτελεσματικά, σε αντίθεση με την κλασσική χρήση υπολογιστικών συστημάτων σε πραγματικό χρόνο. Επιπλέον η υπολογιστική μάθηση χωρίζεται σε κατηγορίες ανάλογα με τον τρόπο που πραγματοποιείται η διαδικασία της μάθησης. Οι κατηγορίες αυτές είναι:

- Μάθηση με Επίβλεψη (Supervised Learning - SL)
  Στην Μάθηση με Επίβλεψη που ανήκουν οι κλασσικές μέθοδοι εκπαίδευσης κυρίως νευρωνικών δικτύων όπου το μοντέλο θεωρείται ως ένα μαύρο κουτί ή μια μαθηματική συνάρτηση, όπου παρέχονται η είσοδος με τα δεδομένα και η έξοδος με το αποτέλεσμα που απαιτείται. Έτσι το μοντέλο θα πρέπει να εκπαιδευτεί κατάλληλα έτσι ώστε να καταλάβει την σχέση μεταξύ της εισόδου και της εξόδου, προκειμένου να είναι σε θέση τελικά να εξάγει αποτέλεσμα μόνο όταν του παρέχεται είσοδος.
- Μάθηση χωρίς Επίβλεψη (Unsupervised Learning – UL)
  Στην Μάθηση χωρίς Επίβλεψη, παρέχεται στο μοντέλο μόνο η είσοδος και απαιτείται από αυτό να δημιουργήσει αποτέλεσμα. Κυρίως βρίσκει εφαρμογή σε προβλήματα ομαδοποίησης όπου παρέχονται δεδομένα με το μοντέλο να πρέπει να αντιληφθεί τα κοινά στοιχεία και τις διαφορές τους και να τα ομαδοποιήσει.
- Μάθηση με Ενθάρρυνση (Reinforcement Learning - RL)
  Στην Μάθηση με Ενθάρρυνση, το μοντέλο δέχεται ερεθίσματα από το περιβάλλον στο οποίο εφαρμόζεται, είτε θετικά είτε αρνητικά ανάλογα με το αποτέλεσμα που παράγει και διαμορφώνει την συμπεριφορά του. Διαρκώς ψάχνει το αποτέλεσμα που απαιτείται με συνεχή ανάδραση να εισέρχεται σε αυτό [4].

Η υπολογιστική μάθηση όμως, ανεξάρτητα από την κατηγορία στην οποία ανήκει, πέρα από τις τεχνικές και τους υπολογιστικούς πόρους απαιτεί και άλλο ένα κυρίαρχο συστατικό. Όπως είναι φυσικό για να πραγματοποιηθεί η διαδικασία της μάθησης θα πρέπει να υπάρχουν στοιχεία ή δεδομένα έτσι ώστε να παραχθεί ένα συμπέρασμα από αυτά. Η άνθιση που υπάρχει σήμερα στην υπολογιστική μάθηση οφείλεται σε σημαντικό βαθμό σε αυτό. Πλέον υπάρχουν τα διαθέσιμα δεδομένα προκειμένου να αξιοποιηθούν για την εκπαίδευση μοντέλων υπολογιστικής μάθησης, και πολλές φορές αποτελούν μήλον της έριδος για εταιρίες και συστήματα. Τα δεδομένα αυτά, που είτε υπάρχουν είτε δημιουργούνται δυναμικά είναι τόσο

σημαντικά, ώστε η απόκτηση τους να αποτελεί σύνθετη και ακριβή συχνά διαδικασία αφού η ζήτηση τους ανεβαίνει συνεχώς.

ΚΕΦΑΛΑΙΟ 2 ° Ομοσπονδιακή Μάθηση

Η Ομοσπονδιακή Μάθηση (Federated Learning) αποτελεί ένα αποκεντρωμένο είδος υπολογιστικής μάθησης. Με κύριο μέλημα την ιδιωτικότητα και την προστασία των δεδομένων του κάθε χρήστη που συμμετέχει στην διαδικασία, η Ομοσπονδιακή Μάθηση, κερδίζει ολοένα και περισσότερη συμμετοχή στην έρευνα και την εφαρμογή της. Η πρώτη αναφορά της έγινε το 2016 από ερευνητές της Google, στην δημοσίευση με τίτλο «Communication-Efficient Learning of Deep Networks from Decentralized Data» [1]. Έκτοτε, πολλές επιστημονικές δημοσιεύσεις, έρευνες και μελέτες ασχολούνται με αυτό το αντικείμενο προσπαθώντας να βελτιώσουν τους διάφορους τομείς του, να βελτιώσουν υπάρχουσες εφαρμογές υπολογιστικής μάθησης ενσωματώνοντας την και αντικαθιστώντας ουσιαστικά τους κλασικούς τρόπους με αυτή, ή ακόμα και να δημιουργήσουν νέες εφαρμογές όπου η ευαισθησία των δεδομένων που έπρεπε να χρησιμοποιηθούν για την εκπαίδευση των μοντέλων δεν επέτρεπε την υλοποίηση τους, με κεντρικοποιημένη μάθηση.

2.1 Ιδιότητες Ομοσπονδιακής Μάθησης

Όπως ήδη αναφέρθηκε πρωταρχικός στόχος της Ομοσπονδιακής Μάθησης είναι η ιδιωτικότητα. Η ιδιωτικότητα αυτή έχει να κάνει με τα δεδομένα που χρησιμοποιούνται ως είσοδοι σε ένα πρόβλημα. Τα δεδομένα αυτά είτε συλλέγονται μέσω αισθητήρων κ.λπ., είτε δημιουργούνται σε ένα σύστημα μέσω διαφόρων διαδικασιών που μεσολαβούν, αν πρόκειται για υπολογιστική συσκευή, είτε παρέχονται αυτούσια από τους χρήστες. Αν το πρόβλημα αυτό επιδέχεται λύση μέσω υπολογιστικής μάθησης, η πιο συνηθισμένη διαδικασία είναι η αποστολή των δεδομένων σε ένα κεντρικοποιημένο σύστημα. Αυτό γίνεται αφού το κεντρικό σύστημα έχει υψηλούς υπολογιστικούς πόρους, που θα επιτρέψουν την εκπαίδευση μοντέλων από τα δεδομένα αυτά, με σκοπό την άμεση πρόταση και παροχή λύσης σε νέες δυναμικές συνθήκες του προβλήματος. Επομένως, προκειμένου να διασφαλιστεί η ιδιωτικότητα των δεδομένων που επιθυμείται, θα πρέπει να σταματήσει η αποστολή των δεδομένων, η μεταφοράς τους δηλαδή από την παραγωγή τους στο κεντρικό σύστημα.

Λύση σε αυτό, επιδιώκει να δώσει αυτή η σχετικά νέα τεχνική υπολογιστικής μάθησης. Μέσω αυτής αναδιαμορφώνεται το συνολικό περιβάλλον της υπολογιστικής μάθησης. Σε αυτή, η λειτουργία της εκπαίδευσης δεν γίνεται σε κάποιο κεντρικό σύστημα αλλά τοπικά εκεί που δημιουργήθηκαν τα δεδομένα και υπάρχουν ήδη. Έτσι αποτρέπεται η μεταφορά τους, και συνεπώς εμποδίζεται όχι μόνο η κοινοποίηση των πολύτιμων αυτών δεδομένων στο κεντρικό σύστημα το οποίο έχει πρόσβαση σε αυτά και μπορεί να τα διαχειριστεί και να τα χρησιμοποιήσει, αλλά και ο κίνδυνος υποκλοπής των δεδομένων τόσο από το κεντρικό σύστημα, όσο και κατά την μεταφορά τους [1].

Βασικό χαρακτηριστικό της εκπαίδευσης μοντέλων υπολογιστικής μάθησης είναι οι αυξημένες απαιτήσεις σε υπολογιστική ισχύ. Όπως ειπώθηκε, σκοπός της αποστολής των δεδομένων σε κεντρικό σύστημα, με μεθόδους κλασσικής υπολογιστικής μάθησης, είναι η εκπαίδευση των μοντέλων με τη χρήση αυτών εξαιτίας των δυνατοτήτων του συστήματος αυτού. Ωστόσο, πλέον οι συσκευές και τα τοπικά συστήματα που έχουν στην διάθεση τους οι χρήστες, έχουν και αυτά αυξημένη υπολογιστική ισχύ σε σχέση με παλαιότερα. Επιπλέον η ανάπτυξη ειδικών επεξεργαστών για χρήση κυρίως σε υπολογιστική μάθηση και κυρίως των

GPU κάνει την τοπική εκπαίδευση σε επιμέρους συστήματα ευκολότερη. Βέβαια, οι δυνατότητες αυτών δεν μπορούν να φτάσουν, σε ίδια επίπεδα με αυτές ενός κεντρικού συστήματος κατασκευασμένο με ρόλο τέτοιο ώστε να εξυπηρετεί την εκπαίδευση μοντέλων και νευρωνικών δικτύων.

Το πρόβλημα αυτό εξυπηρετείται μερικώς, από την βασική αρχή που ακολουθεί η Ομοσπονδιακή Μάθηση. Σε πρακτικές κεντρικοποιημένης μάθησης θα συσσωρεύονταν όλα τα δεδομένα, σε όσο το δυνατό μεγαλύτερο βαθμό προκειμένου να γίνει καλή και σωστή εκπαίδευση του μοντέλου στο κεντρικό σύστημα. Αυτός ο μεγάλος όγκος δεδομένων, μπορεί να είναι επιθυμητός αφού λειτουργεί θετικά στην γενίκευση του μοντέλου με την εκπαίδευση του σε αυτά και επομένως, παροχή σωστής πρόβλεψης σε μεγαλύτερο εύρος ωστόσο κάνει δυσκολότερη την διαδικασία απαιτώντας μεγάλη υπολογιστική ισχύ ώστε να γίνουν πολλές επαναλήψεις με την κάθε μία να πρέπει να προσπελάσει όλα αυτά τα δεδομένα. Αντίθετα, μέσω της λογικής που προτείνεται από την Ομοσπονδιακή Μάθηση, η εκπαίδευση γίνεται τοπικά με τα δεδομένα που υπάρχουν στο τοπικό αυτό σύστημα και μόνο. Αυτός ο μικρότερος αριθμός στοιχείων κάνει ευκολότερη την εκπαίδευση των τοπικών μοντέλων με αποτέλεσμα να απαιτείται η ύπαρξη μίας δεδομένης υπολογιστικής ισχύος που δεν χρειάζεται όμως να είναι συγκρίσιμη με την αντίστοιχη ενός κεντρικού συστήματος.

Όπως προδίδεται όμως και από το ίδιο το όνομα της, η Ομοσπονδιακή Μάθηση απαιτεί την συμμετοχή όχι ενός αλλά πολλών τοπικών συστημάτων. Μέσω της συμμετοχής πολλών συστημάτων εξυπηρετείται η ανάγκη για πολλά δεδομένα στην διαδικασία της εκπαίδευσης που προσφέρουν μεγαλύτερη γενίκευση. Συμμετέχουν δηλαδή πολλά συστήματα με σχετικά μικρό αριθμό δεδομένων το καθένα, τα οποία αθροιστικά ίσως να έχουν αντίστοιχο αριθμό με αυτόν που θα μεταφέρονταν κεντρικά σε αντίστοιχη περίπτωση κλασσικής μάθησης. Επιπλέον, μέσω της εφαρμογής της, η Ομοσπονδιακή Μάθηση μπορεί να μας δώσει πρόσβαση σε δεδομένα, που ενδεχομένως δεν θα είχαμε αν ακολουθούσαμε τεχνικές κεντρικοποιημένης υπολογιστικής μάθησης. Αυτό μπορεί να συμβεί επειδή τα δεδομένα αυτά ενδεχομένως να προστατεύονται από αυστηρά νομικά πλαίσια τα οποία να απαγορεύουν την αποστολή τους σε κάποιο, απομακρυσμένο από το αρχικό, σύστημα [5]. Ιδιαίτερα τώρα, που η ζήτηση για δεδομένα είναι μεγαλύτερη από ποτέ, έχουν δημιουργηθεί αναβαθμισμένες διατάξεις και δίνεται πρόσθετη προσοχή στο θέμα αυτό. Ακόμα, σημαντικό ρόλο παίζει και η ίδια η διστακτικότητα των χρηστών να κοινοποιήσουν τα δεδομένα τους, αφού δεν είναι σε θέση να γνωρίζουν πώς θα χρησιμοποιηθούν και ποιος ή ποιοι θα μπορούν να αποκτήσουν πρόσβαση σε αυτά. Τα παραπάνω δυσχεραίνουν αρκετά την ομαλή και σωστή λειτουργία μεθόδων που απαιτούν την μεταφορά πληροφοριών που χρησιμοποιούνται για την εκπαίδευση.

Πέρα της ιδιωτικότητας και της προστασίας που προσφέρει η έλλειψη της μεταφοράς δεδομένων σημαντική είναι και η τεχνική πλευρά που παρέχει. Όταν δεν μεταφέρονται αυτά τα στοιχεία εξοικονομούνται οι υψηλοί διαδικτυακοί πόροι που είναι απαραίτητοι για να γίνει η μεταφορά αυτή. Επίσης, οι ανάγκες για αποθηκευτικό χώρο που απαιτείται για να γίνει η αποθήκευση τους στα κεντρικά συστήματα μειώνονται σημαντικά. Σημαντική είναι και η περιβαλλοντική πλευρά αφού η μείωση των τεχνικών πόρων που χρειάζονται οι δύο αυτές σκοπιές συμβάλουν στην μείωση ενέργειας που απαιτείται για την τροφοδοσία τους. Παρ' όλα αυτά, όπως γίνεται λόγος και παρακάτω, στην Ομοσπονδιακή Μάθηση υπάρχει μεταφορά δεδομένων από τα τοπικά συστήματα προς κάποιο κεντρικό, με τα δεδομένα αυτά όμως να

αποτελούνται από τις παραμέτρους των μοντέλων που ανταλλάσσονται μεταξύ τους. Όπως, όμως εύκολα γίνεται αντιληπτό το μέγεθος αυτών των παραμέτρων με το αντίστοιχο μέγεθος των δεδομένων για εκπαίδευση είναι σημαντικά μικρότερο ανεξαρτήτως του είδους τους, (πόσο μάλλον όταν αυτά αποτελούνται από εικόνες και βίντεο, κάτι αρκετά σύνηθες). Έτσι έρχεται στο προσκήνιο η Ομοσπονδιακή Μάθηση και οι λύσεις που δίνει στα θέματα αυτά, διασφαλίζοντας την μη μεταφορά των ευαίσθητων αυτών στοιχείων, μειώνοντας δραματικά τον κίνδυνο διαρροής τους και τους πόρους μεταφοράς και αποθήκευσης που χρειάζονται.

2.2 Δομή της Ομοσπονδιακής Μάθησης

Η Ομοσπονδιακή Μάθηση αποτελείται από τα τοπικά συστήματα (clients), και το κεντρικό σύστημα (server) το οποίο ελέγχει και οργανώνει την συνολική διαδικασία. Όπως έχει ήδη ειπωθεί αυτό που κάνει την ΟΜ να διαφέρει είναι η τοπική εκπαίδευση. Οι clients δηλαδή που συμμετέχουν και έχουν τα δεδομένα τους, εκπαιδεύουν ένα τοπικό μοντέλο. Η δομή και η γενικότερη αρχιτεκτονική του «οικοσυστήματος» της ΟΜ, μπορεί να διαφοροποιηθεί, να εισαχθούν και να αλλάξουν παράμετροι, με πολλές έρευνες να αναζητούν αυτές τις αλλαγές για να κάνουν την μάθηση πιο αποτελεσματική σε συγκεκριμένες κάθε φορά συνθήκες. Εδώ περιγράφεται η γενική και πιο διαδεδομένη δομή της ΟΜ Ένα παράδειγμα αυτών των διαφοροποιήσεων είναι η προέλευση των αρχικών τοπικών μοντέλων. Η πιο κοινή λογική που εφαρμόζεται είναι το μοντέλο αυτό, να μοιράζεται στους clients από τον server, στον οποίο ορίζεται η οργάνωση του και αρχικοποιείται, τυχαία. Το τοπικό μοντέλο, ακολουθεί την διαδικασία της εκπαίδευσης, όπως αυτή πραγματοποιείται και στις κλασσικές μεθόδους υπολογιστικής μάθησης. Ορίζεται δηλαδή σε κάθε client το μοντέλο, και μετά για ορισμένες επαναλήψεις ανανεώνονται τα βάρη και γενικότερα οι παράμετροι του με σκοπό την ελαχιστοποίηση του σφάλματος με βάση τον αλγόριθμο βελτιστοποίησης που χρησιμοποιείται για δεδομένα που έχει ήδη, χωρίς κανένα άλλο σύστημα να έχει πρόσβαση σε αυτά. Οι τελικές παράμετροι του μοντέλου που θα παραχθεί από το τοπικό σύστημα με την ολοκλήρωση της εκπαίδευσης θα σταλούν πίσω στο server. Εκεί θα συλλεχθούν όλες οι παράμετροι όλων των επιμέρους μοντέλων των clients, θα γίνει συμψήφιση τους και με βάση το αποτέλεσμα αυτό θα δημιουργηθεί το κεντρικό μοντέλο [1].

Αυτό που μόλις περιγράφηκε, αποτελεί τον πρώτο γύρο επικοινωνίας (communication round) της Ομοσπονδιακής Μάθησης. Αναλυτικά σε ένα τυπικό γύρο επικοινωνίας πραγματοποιούνται:

Βήμα 1: Αρχικοποιείται/ Ενημερώνεται το γενικό μοντέλο από το server.

Βήμα 2: Ο server στέλνει το γενικό μοντέλο που δημιουργήθηκε από το αποτέλεσμα του προηγούμενου γύρου και οι clients το κατεβάζουν.

Βήμα 3: Γίνεται εκπαίδευση του μοντέλου τοπικά σε κάθε client, με χρήση των ιδιωτικών τους δεδομένων και παράγεται το ανανεωμένο τοπικό μοντέλο.

Βήμα 4: Οι clients στέλνουν το τοπικό μοντέλο που παράχθηκε στον server.

Οι γύροι επικοινωνίας αποτελούν πολύ σημαντικό κομμάτι στην συνολική λειτουργία της Ομοσπονδιακής Μάθησης. Ουσιαστικά αποτελούν τις επαναλήψεις εκπαίδευσης του γενικού μοντέλου. Ο αριθμός τους είτε ορίζεται εξαρχής, ή έχουμε περιπτώσεις όπου ορίζεται ως συνθήκη τερματισμού αυτής της επαναλαμβανόμενης διαδικασίας η επίτευξη συγκεκριμένου αριθμού επίδοσης του γενικού μοντέλου [6].

2.3 Είδη και Κατηγορίες Ομοσπονδιακής Μάθησης

2.3.1 Είδη Ομοσπονδιακής Μάθησης

Η Ομοσπονδιακή Μάθηση μπορεί να χωριστεί σε δύο είδη και σε τρεις κατηγορίες. Αρχικά υπάρχουν δύο είδη ΟΜ ανάλογα με τα σενάρια εφαρμογής της. Τα σενάρια αυτά διαφέρουν από το είδος των clients που συμμετέχουν σε αυτή. Αυτά τα δύο διαφορετικά σενάρια που καθορίζουν την διαδικασία της ΟΜ είναι τα: cross-device (μεταξύ συσκευών) και cross-silo (μεταξύ οργανισμών - κέντρων δεδομένων).

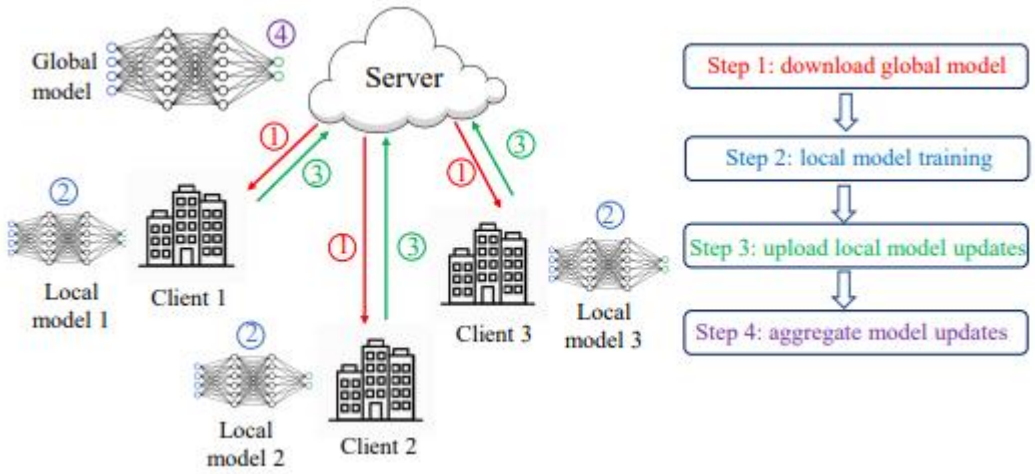
Τα cross-device σενάρια, είναι αυτά στα οποία εφαρμόστηκε πρώτα η ΟΜ. Ακόμα και στην πρώτη δημοσίευση [1], γινόταν λόγος για εφαρμογή και δοκιμή του αυτού του είδους υπολογιστικής μάθησης σε φορητές συσκευές που εκπαιδεύουν μοντέλα, οι οποίες οργανώνονταν από τον server χωρίς να μεταφέρουν τα δεδομένα τους. Σε αυτά τα σενάρια, τα τοπικά συστήματα αποτελούνται από μικρές υπολογιστικές συσκευές (π.χ. έξυπνα κινητά, IoT συσκευές κ.λπ.). Οι συσκευές αυτές έχουν συνήθως σχετικά μικρό αριθμό δεδομένων που χρησιμοποιούν για την εκπαίδευση του τοπικού τους μοντέλου. Αυτός ο τοπικά μικρός όγκος των δεδομένων κάνει επιτακτική την ανάγκη μεγάλης συμμετοχής τέτοιων συστημάτων για να υπάρχει ένα ικανοποιητικό αποτέλεσμα, με τον αριθμό αυτών που συμμετέχουν στην διαδικασία να είναι πολύ μεγάλος (εκατοντάδες ~ εκατομμύρια). Σε αυτού του είδους την εφαρμογή ΟΜ, αναμένονται και πολλές απώλειες από το σύνολο των συστημάτων [6]. Αυτό έχει να κάνει με την έλλειψη διαθεσιμότητας της εκάστοτε συσκευής, που μπορεί να είναι αποτέλεσμα χαμηλής συνδεσιμότητας στο δίκτυο ή και ακόμα και ολικής απώλειας ενέργειας, δηλαδή η συσκευή να είναι απενεργοποιημένη, όταν ο server προσπαθεί να επικοινωνήσει μαζί της. Κύριο χαρακτηριστικό των cross-device σεναρίων είναι η ανομοιογένεια των συσκευών [7]. Αυτό έχει ως αποτέλεσμα τόσο την παραγωγή ανομοιογενών δεδομένων (non-IID Data) όσο και διαφορετικές υπολογιστικές δυνατότητες σε κάθε client, δυσχεραίνοντας πολύ την διαδικασία τόσο της τοπικής εκπαίδευσης αλλά και της συνολικής διαδικασίας παραγωγής των γενικών μοντέλων.

Όπως αναφέρθηκε και παραπάνω η λογική της Ομοσπονδιακής Μάθησης ξεκίνησε από τα cross-device σενάρια. Ωστόσο, αυτή η δυναμική που προσέφερε αυτό το είδος υπολογιστικής μάθησης, δεν άργησε να επεκτείνει την εφαρμογή της και σε σενάρια μεταξύ κατανεμημένων συστημάτων οργανισμών, δηλαδή στα cross-silo σενάρια. Σε αντίθετη λογική

με τα cross-device, αυτά τα σενάρια έχουν να κάνουν με clients – silos[1] , οι οποίοι αποτελούνται κυρίως από οργανισμούς, εταιρίες ή ιδρύματα. Συνήθεις παραδείγματα τέτοιων σεναρίων αποτελούν νοσοκομεία, με σκοπό την πρόβλεψη ασθενειών σε πρώιμα στάδια και χρηματοπιστωτικά ιδρύματα με σκοπό την παραγωγή προβλέψεων των δεικτών της οικονομίας, με τα στοιχεία που κατέχουν να αποτελούν και στις δύο περιπτώσεις τον ορισμό των ευαίσθητων, προσωπικών δεδομένων και επομένως να επιβάλλουν την χρήση της Ομοσπονδιακής Μάθησης. Σε αυτά τα σενάρια η συμμετοχή των τοπικών συστημάτων είναι σχετικά μικρή (2~100) [6]. Όμως αυτά τα συστήματα συμβάλουν ιδιαίτερα αφού υπάρχει σημαντική ποσότητα δεδομένων στο κάθε ένα, κάτι που καθιστά την τοπική εκπαίδευση εδώ κρισιμότερη αφού μπορεί να επηρεάσει άμεσα το αποτέλεσμα. Αυτό κάνει σημαντική την συμμετοχή στην συνολική διαδικασία του κάθε client. Η ολική συμμετοχή αυτή ωστόσο, είναι εύκολο να επιτευχθεί σε αυτά τα σενάρια, αφού οι clients έχουν δεδομένη συνδεσιμότητα στο δίκτυο. Ακόμα, δεδομένη θα πρέπει να θεωρείται και η τροφοδοσία τους, και η σταθερή λειτουργία τους. Επομένως μπορούν να επικοινωνήσουν με τον server σε οποιαδήποτε στιγμή χρειαστεί χωρίς να δημιουργηθεί κανένα πρόβλημα, κάνοντας οποιαδήποτε πιθανότητα αποχώρησης από την διαδικασία εξαιρετικά μικρή. Επίσης, παρότι τα συστήματα είναι αρκετά ελεγχόμενα ως προς το υλικό τους και τα δεδομένα που έχουν, δεν σημαίνει ότι τα δεδομένα αυτά θα είναι ίδια. Επομένως, μπορεί σε κάποιες εφαρμογές τα δεδομένα να είναι όμοια (IID Data) αλλά είναι και αρκετά σύνηθες να παρουσιάζουν ανομοιογένειες (non-IID Data), σε σημαντικά μικρότερο βέβαια βαθμό από τα αντίστοιχα στα cross-device σενάρια.

Συνοψίζοντας, τα δύο αυτά είδη αποτελούν τις δύο κύριες διαφορές στην Ομοσπονδιακή Μάθηση με τα πλεονεκτήματα και τα μειονεκτήματα τους, και το καθένα χρίζει διαφορετικής αντιμετώπισης. Τα cross-device σενάρια αποτελούν το μεγαλύτερο κομμάτι εφαρμογής της ΟΜ με περισσότερες χρήσεις και επομένως και μεγαλύτερη ζήτηση για έρευνα και εφαρμογή, παρουσιάζοντας βέβαια δυσκολίες στο μέγεθος της υλοποίησης, την διαφορετικότητα των clients και των δεδομένων. Από την άλλη στα cross-silo σενάρια πραγματοποιούνται πιο στοχευμένες εφαρμογές με αναζήτηση συγκεκριμένων αποτελεσμάτων, που απαιτούν καλό στήσιμο της διαδικασίας και σωστή διαχείριση της εκπαίδευσης του κάθε client, αφού χρειάζεται να αξιοποιηθεί η δυνατότητα της μεγαλύτερης τοπικής εκπαίδευσης που προσφέρει, αλλά απαιτεί και προσοχή αφού λάθη σε αυτή μπορεί εύκολα να στρεβλώσουν τα αποτελέσματα.

[1] Στα cross-silo σενάρια τα τοπικά συστήματα μπορεί να αναφέρονται είτε clients είτε silos (δεδομένων).



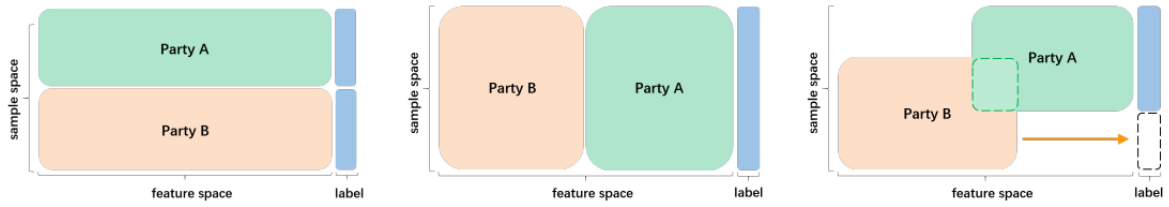
Εικόνα 2.3.1 Τυπικός Γύρος Επικοινωνίας σε Cross-Silo ΟΜ [6]

2.3.2 Κατηγορίες Ομοσπονδιακής Μάθησης

Πέρα από τα δύο είδη που αναλύθηκαν η Ομοσπονδιακή Μάθηση χωρίζεται και σε τρεις κατηγορίες ανάλογα με τα δεδομένα που υπάρχουν στους clients. Οι κατηγορίες αυτές είναι:

- Η Οριζόντια Ομοσπονδιακή Μάθηση (ΟΟΜ) ή Horizontal Federated Learning (HFL), υπάρχουν δεδομένα τα οποία ανήκουν σε διαφορετική τάξη ανά client που συμμετέχει σε αυτή, αλλά όλα έχουν παρόμοια χαρακτηριστικά. Για παράδειγμα αν επρόκειτο για το dataset MNIST, που αναφέρεται στο 3.2, οι clients όλοι θα έχουν ασπρόμαυρες φωτογραφίες χειρόγραφων αριθμών, με φωτογραφίες όμως που αναπαριστούν συγκεκριμένο ή συγκεκριμένους αριθμούς σε κάθε client, που οι άλλοι δεν θα έχουν.
- Η Κάθετη Ομοσπονδιακή Μάθηση (ΚΟΜ) ή Vertical Federated Learning (VFL), όπου υφίστανται δεδομένα με κοινές τάξεις σε κάθε client που συμμετέχει. Στην περίπτωση αυτή, ωστόσο τα χαρακτηριστικά των δεδομένων δεν θα είναι τα ίδια. Σε αντιστοιχία με το παράδειγμα του MNIST dataset, εδώ θα υπάρχουν φωτογραφίες από κάποιο αριθμό σε όλους του clients. Ο καθένας όμως θα έχει διαφορετικού είδους φωτογραφίες του κάθε αριθμού, είτε αυτή έχει να κάνει με το αν είναι έγχρωμη ή ασπρόμαυρη, με την ανάλυση της κ.τ.λ.
- Η Ομοσπονδιακή Μάθηση Μεταφοράς (ΟΜΜ) ή Federated Transfer Learning (FTL), που αποτελεί ουσιαστικά τον συνδυασμό των παραπάνω. Υπάρχουν δηλαδή δεδομένα που διαφέρουν ή όχι, τόσο στα χαρακτηριστικά τους όσο και στις τάξεις που ανήκουν. Επιπλέον σε αυτή την κατηγορία έχουμε και περιπτώσεις όπου μπορεί τα δεδομένα να έχουν κοινά χαρακτηριστικά και να ανήκουν σε κοινή τάξη, αλλά να είναι μοιρασμένα σε πολλούς clients. Συνεχίζοντας με αντίστοιχο παράδειγμα πλέον, μπορεί ένας client να έχει έγχρωμες φωτογραφίες ενός αριθμού, κάποιος άλλος ασπρόμαυρες διαφορετικού αριθμού ή και του ίδιου αλλά και συνδυασμό τους, δηλαδή να έχουν και οι δύο κάποιες με ίδια χρώματα που να απεικονίζουν τον ίδιο αριθμό [8].

Σε cross-silo σενάρια μπορεί να βρει εφαρμογή οποιαδήποτε από τις τρείς κατηγορίες αναφέρθηκαν, αφού είναι πιθανή η κάθε περίπτωση ανάλογα με την εφαρμογή που πραγματοποιείται. Αντιθέτως σε cross-device σενάρια είναι αρκετά σπάνιο έως αδύνατο να υπάρξει η δεύτερη περίπτωση αφού η συλλογή των δεδομένων σε αυτό το είδος, είναι τόσο αυξημένη που καθιστά ελάχιστη την πιθανότητα όλα αυτά τα δεδομένα να ανήκουν σε μία και μόνο τάξη.



(a) Horizontal Federated Learning (b) Vertical Federated Learning (c) Federated Transfer Learning

Εικόνα 2.3.2 Κατηγορίες Ομοσπονδιακής Μάθησης [8]

2.4 Ασφάλεια Δεδομένων

Το κύριο μέλημα της Ομοσπονδιακής Μάθησης είναι η ιδιωτικότητα και η ασφάλεια των δεδομένων. Όπως πλέον, έχει αναφερθεί αρκετά αυτό επιτυγχάνεται αφού δεν αποστέλλονται τα ίδια τα δεδομένα που χρησιμοποιούνται για την εκπαίδευση αλλά τα μοντέλα που εκπαιδεύονται τοπικά και οι παράμετροι τους. Ωστόσο, αυτό μπορεί να μην είναι αρκετό. Μέσω ανάλυσης των διαφορών των παραμέτρων του κάθε μοντέλου που αποστέλλεται στον server μπορεί να ανακτηθούν (ή τουλάχιστον μέρος τους) τα απόρρητα και ιδιωτικά δεδομένα που επιχειρείται να γίνει φύλαξη τους. Έτσι έχουν αναπτυχθεί τεχνικές ώστε ακόμα και τα δεδομένα που μεταφέρονται δηλαδή οι παράμετροι των μοντέλων να φιλτράρονται ώστε να αποφευχθεί η παραπάνω περίπτωση.

Η πιο συνηθισμένη τεχνική που εφαρμόζεται την αποτροπή διαρροής πολύτιμων δεδομένων, είναι η differential privacy. Αυτή είναι μία αρκετά διαδεδομένη τεχνική που βρίσκει εφαρμογή σε πολλές περιπτώσεις μεταφοράς δεδομένων που πρέπει να προστατευτούν. Αυτό κάνει την χρήση της στην περίπτωση της Ομοσπονδιακής Μάθησης ευκολότερη. Η βασική αρχή της τεχνικής αυτής είναι η εισαγωγή θορύβου που αλλοιώνει την πληροφορία που μεταφέρεται και έτσι προστατεύει τα πραγματικά χρήσιμα και ευαίσθητα δεδομένα. Στην Ομοσπονδιακή Μάθηση αποτελεί ακόμα ένα μέτρο ασφαλείας. Σε εφαρμογή ΟΜ αυτό που γίνεται είναι προτού γίνει η συνάθροιση των συντελεστών των τοπικών μοντέλων στο server, στους διάφορους clients που συμμετέχουν προστίθεται κάποιος θόρυβος στους συντελεστές του κάθε μοντέλου. Έτσι «θωρακίζοντας» ακόμα και αυτές τις πληροφορίες αυξάνεται αρκετά η δυσκολία να γίνει εξαγωγή των δεδομένων που χρησιμοποιούνται για τις τοπικές εκπαιδεύσεις και αποτελούν πολύτιμο στοιχείο στην συνολική διαδικασία.

Ωστόσο αυτή η διαδικασία αποτελεί ένα περαιτέρω μέτρο ασφαλείας πέρα από αυτή που προσφέρει η ίδια η φύση της ΟΜ. Αξίζει όμως να αναφερθεί αφού μέσω της εφαρμογής της και της λειτουργίας της εισαγωγής του θορύβου στις παραμέτρους που μεταφέρονται επηρεάζει σημαντικά την απόδοση και την επίδοση της συνολικής διαδικασίας. Η δυσκολία

της συσσωμάτωσης δεδομένων που έχουν «αλλοιωθεί» με την προσθήκη θορύβου προκαλεί δυσκολία και καθυστέρηση, σε σχέση με δεδομένα χωρίς θόρυβο, της συγκέντρωσης τους και της παραγωγής του γενικού μοντέλου. Επηρεάζεται δηλαδή η συνάθροιση των παραμέτρων και η λειτουργία των αλγορίθμων συσσωμάτωσης, με μερικούς να δίνουν έμφαση στην εισαγωγή τέτοιων διαδικασιών και αντιμετώπισης των τεχνικών μειονεκτημάτων που επιφέρει [9].

ΚΕΦΑΛΑΙΟ 3 ° : Πρόβλημα Εφαρμογής – Δεδομένα και Προσομοίωση Προβλήματος σε Εφαρμογή Ομοσπονδιακής Μάθησης

Προκειμένου να γίνει η σύγκριση των αλγορίθμων συσσωμάτωσης Ομοσπονδιακής Μάθησης, που είναι και ο σκοπός της συγκεκριμένης εργασίας, χρειάζεται να βρεθεί ένα πρόβλημα για να λυθεί μέσω εφαρμογής Ομοσπονδιακής Μάθησης η οποία και θα πραγματοποιηθεί μέσω των αλγορίθμων αυτών. Χρειάζεται δηλαδή να βρεθεί ένα πρόβλημα που επιδιώκεται να λυθεί μέσω υπολογιστικής μάθησης, ώστε να αναπτυχθεί σε εφαρμογή Ομοσπονδιακής Μάθησης και στην συνέχεια να συγκριθούν οι λύσεις που έχουν προκύψει εφαρμόζοντας κάθε φορά διαφορετικό αλγόριθμο για συγκεκριμένες παραμέτρους. Για αυτό το λόγο επιλέχθηκε ένα πολύ σύνηθες πρόβλημα υπολογιστικής μάθησης ένα πρόβλημα ταξινόμησης εικόνων σε κατηγορίες (image classification problem).

3.1 Πρόβλημα Εφαρμογής

Η επιλογή πραγματοποίησης του πειράματος με πρόβλημα ταξινόμησης εικόνων έγινε αφού αποτελεί πλέον βασική διαδικασία υπολογιστική μάθησης. Αυτό το είδος προβλήματος είναι βασικό πρόβλημα υπολογιστική μάθησης εδώ και χρόνια, αφού αποτελούν κυρίαρχο μέρος πολλών εφαρμογών σε πολλές περιπτώσεις. Είτε μιλάμε για την εύρεση προσώπων σε φωτογραφίες ή για την αναγνώριση χαρακτήρων που γράφονται με το δάχτυλο ή με ειδικό στυλό σε μία οθόνη αφής. Ειδικότερα με την ανάπτυξη συσκευών, αισθητήρων και λειτουργιών αποτύπωσης και ακόμα με την ανάπτυξη του διαδικτύου των πραγμάτων, αποτελεί εδώ και χρόνια καίρια κατηγορία προβλήματος του συγκεκριμένου τομέα. Μπορεί να χαρακτηριστεί τόσο σημαντικό που χρησιμοποιείται ακόμα και σε εκπαιδευτικά παραδείγματα για εισαγωγή στην υπολογιστική μάθηση και των μοντέλων της. Έτσι υπάρχουν πολλές υλοποιήσεις παρόμοιων προβλημάτων. Αυτό έχει σαν αποτέλεσμα την σχετικά εύκολη εύρεση τόσο δεδομένων όσο και λύσεων σε προβλήματα που προκύπτουν κατά την υλοποίηση. Η δημοφιλία του προβλήματος, το έχει κάνει άμεσα συμβατό με τα περισσότερα και πιο ευρέως χρησιμοποιούμενα προγραμματιστικά πακέτα και βιβλιοθήκες υπολογιστικής μάθησης, κάνοντας ευκολότερη την εισαγωγή των βασικών συνόλων δεδομένων, όπως αυτά που χρησιμοποιήθηκαν και αναφέρονται στο κεφάλαιο 3.2. Επίσης, εξαιτίας των πάρα πολλών υλοποιήσεων του προβλήματος είναι εύκολη και η ανάπτυξη των μοντέλων που απαιτείται για την λύση του αφού τα παραδείγματα και οι διάφορες δοκιμές που έχουν γίνει ήδη, απλοποιούν κατά πολύ την διαδικασία.

Επιπλέον κύριο πλεονέκτημα τέτοιου είδους προβλημάτων είναι η επεκτασιμότητα τους. Τα προβλήματα αυτά μπορεί να είναι σχετικά απλά με μικρές ομάδες δεδομένων που αποτελούνται από απλές μικρές εικόνες, χαμηλής ανάλυσης, και απεικονίζουν μόνο ένα ή αρκετά μικρό αριθμό αντικειμένων. Από την άλλη, μπορεί όμως να αναπτυχθούν και να αποτελέσουν μια αρκετά δύσκολη δοκιμασία, με πάρα πολύ μεγάλο αριθμό εικόνων, πολύ υψηλής ανάλυσης, που περιέχουν πολλά αντικείμενα τα οποία απαιτείται να αναγνωριστούν. Τέτοιας υψηλής κλίμακας προβλήματα ταξινόμησης εικόνων αποτελούν σήμερα σημαντικό κομμάτι της υπολογιστικής μάθησης, καθώς βρίσκουν εφαρμογή σε περιπτώσεις υπολογιστικής όρασης (YO), στα αγγλικά computer vision, (CV). Σε αυτές τις περιπτώσεις ουσιαστικά μιλάμε για βίντεο, είτε αποθηκευμένα, είτε ζωντανής καταγραφής, στα οποία αποκόπτονται καρέ, τα οποία αποτελούν και τις εικόνες που αναλύονται, και βρίσκουν εφαρμογή σε συστήματα αυτόνομης οδήγησης, εύρεσης βάθους, αυτόματης εστίασης και σε πολλά άλλα.

Συνεπώς, τέτοια προβλήματα είναι εύκολα παραμετροποιήσιμα, αφού προσαρμόζεται το μέγεθος, τα δεδομένα και γενικότερα η πολυπλοκότητα τους. Ωστόσο, με την αύξηση της πολυπλοκότητας, αυξάνεται και η υπολογιστική ικανότητα που απαιτείται για την ανάλυση

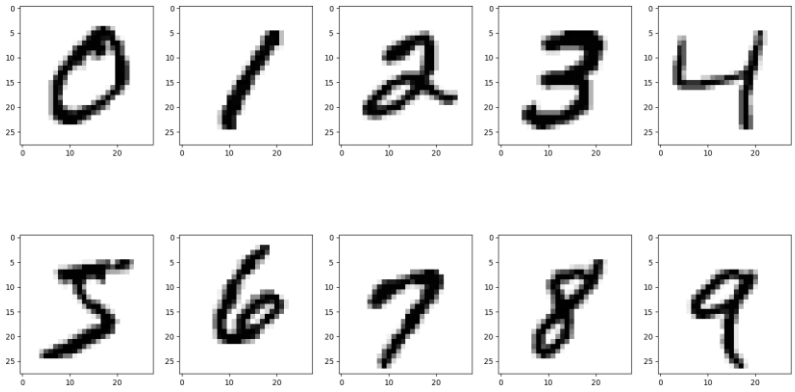
των δεδομένων και την εύρεση λύσεων των προβλημάτων. Αυτή η κλιμακούμενη φύση των προβλημάτων ταξινόμησης εικόνων και η αναλογία της με τους υπολογιστικούς πόρους τα καθιστά ιδανικά για μελέτη και έρευνα, τόσο σε χαμηλού όσο και σε υψηλού επιπέδου συστήματα.

Στην συγκεκριμένη εργασία, επιλέχθηκε ένας πολύ απλός βαθμός του προβλήματος αφού σκοπός δεν είναι η εύρεση λύσης για κάποιο συγκεκριμένο υψηλών προδιαγραφών και απαιτήσεων πρόβλημα, αλλά ούτε καν η εύρεση της βέλτιστης λύσης του προβλήματος καθαυτού. Στόχος είναι η καταγραφή της διαφοράς των επιλεγμένων αλγορίθμων συσσωμάτωσης στην επίδοση και εξαγωγή συμπερασμάτων για την λειτουργία και την χρήση τους για εφαρμογή Ομοσπονδιακής Μάθησης οποιοδήποτε και αν είναι το πρόβλημα. Επίσης, χαμηλού επιπέδου πρόβλημα απαιτεί και λιγότερο χρόνο για εκπαίδευση και επομένως λιγότερο χρόνο για την ολοκλήρωση της συνολικής διαδικασίας.

3.2 Σύνολα Δεδομένων

Για την υλοποίηση του προβλήματος επιλέχθηκαν σχετικά απλά σύνολα δεδομένων - Datasets τα οποία αποτελούν βασικό στοιχείο πλέον της μηχανικής μάθησης, και είναι γνωστά σε καθένα που ασχολείται με το αντικείμενο. Αυτά είναι τα datasets: MNIST, CIFAR10 και CIFAR100. Η επιλογή τους έγινε γιατί αποτελούνται από εύκολα διαχειρίσιμα δεδομένα, συγκεκριμένα χαμηλής ανάλυσης εικόνες και απαιτούν χαμηλής πολυπλοκότητας μοντέλα για να έχουν ένα καλό αποτέλεσμα. Τα δεδομένα αυτά και για τα τρία datasets αποτελούνται από τις ίδιες τις εικόνες και τις κατηγορίες όπου ανήκουν. Συγκεκριμένα κάθε δεδομένο αποτελείται από μία εικόνα και την κατηγορία (label) η οποία αντιστοιχεί στην συγκεκριμένη εικόνα.

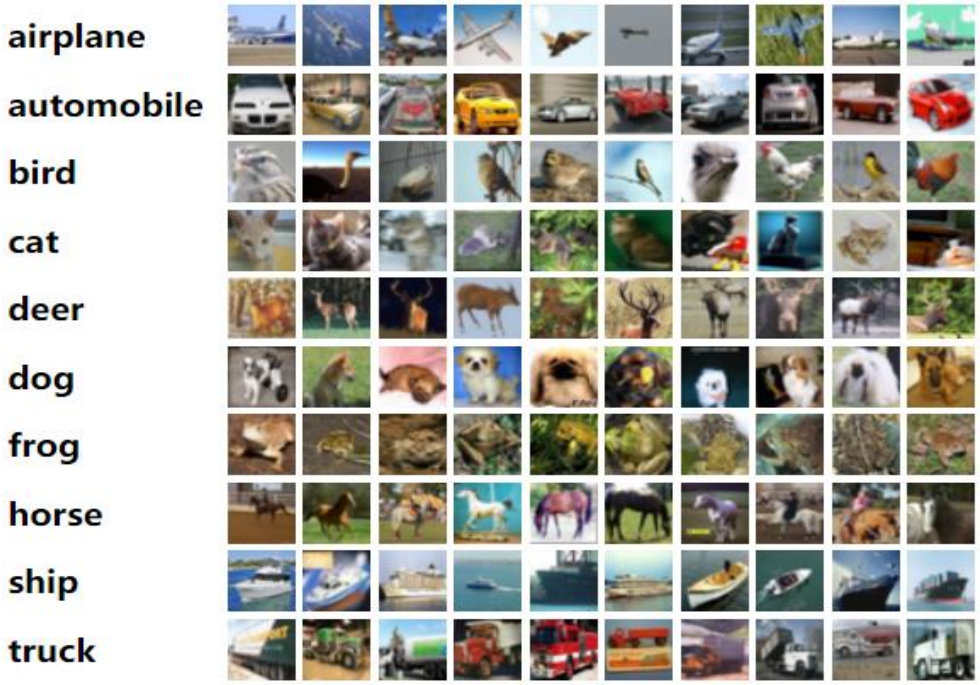
Το Dataset MNIST είναι μια βάση δεδομένων που αποτελείται από 70.000 εικόνες χειρόγραφων αριθμητικών χαρακτήρων (0 έως 9) (Εικόνα 4.2.1), από τις οποίες οι 60.000 χρησιμοποιούνται για την εκπαίδευση και οι 10.000 για την αξιολόγηση των μοντέλων. Οι εικόνες αποτελούνται από μία διάσταση (ένα κανάλι), αφού είναι σε κλίμακα του γκρι και αποτελούνται από 28x28 pixels. Λόγω του μικρού μεγέθους των εικόνων και της ευκολίας διαχείρισης τους αποτελεί μία από τις πιο διαδεδομένες, αν όχι την πιο διαδεδομένη, κοινές βάσεις δεδομένων με τεράστια εφαρμογή σε ερευνητικές εφαρμογές αλλά και σε εκπαιδευτικές εφαρμογές μηχανικής μάθησης και νευρωνικών δικτύων. [2]



Εικόνα 3.2.1 MNIST DATASET IMAGES

[2] http://yann.lecun.com/exdb/mnist

Το Dataset CIFAR10 είναι συλλογή που αποτελείται από 60.000 εικόνες αντικειμένων τα οποία κατηγοριοποιούνται σε 10 κλάσεις (Εικόνα 4.2.2), με την κάθε μία να αποτελείται από 6.000 εικόνες. Οι εικόνες αποτελούνται από τρεις διαστάσεις (τρία κανάλια) (RGB) και κάθε εικόνα έχει ανάλυση 32x32 pixels, κάνοντας το εξίσου εύκολα διαχειρίσιμο λόγω της μικρής ανάλυσης των εικόνων. [3]



Εικόνα 3.2.2 CIFAR10 DATASET

Το Dataset CIFAR100 είναι συλλογή ακριβώς ίδια με την CIFAR10, ωστόσο με την διαφορά ότι αποτελείται από 100 κλάσεις με τις 10 κλάσεις να σπάνε σε 10 υποκλάσεις η κάθε μία όπως φαίνονται στην παρακάτω εικόνα (Εικόνα 4.2.3). [3]

Classes
beaver, dolphin, otter, seal, whale
aquarium fish, flatfish, ray, shark, trout
orchids, poppies, roses, sunflowers, tulips
bottles, bowls, cans, cups, plates
apples, mushrooms, oranges, pears, sweet peppers
clock, computer keyboard, lamp, telephone, television
bed, chair, couch, table, wardrobe
bee, beetle, butterfly, caterpillar, cockroach
bear, leopard, lion, tiger, wolf
bridge, castle, house, road, skyscraper
cloud, forest, mountain, plain, sea
camel, cattle, chimpanzee, elephant, kangaroo
fox, porcupine, possum, raccoon, skunk
crab, lobster, snail, spider, worm
baby, boy, girl, man, woman
crocodile, dinosaur, lizard, snake, turtle
hamster, mouse, rabbit, shrew, squirrel
maple, oak, palm, pine, willow
bicycle, bus, motorcycle, pickup truck, train
lawn-mower, rocket, streetcar, tank, tractor

Εικόνα 3.2.3 CIFAR100 CLASSES

[3] https://www.cs.toronto.edu/~kriz/cifar.html

3.3 Προσομοίωση Εφαρμογής Ομοσπονδιακής Μάθησης

Το πρόβλημα ταξινόμησης εικόνων, που εφαρμόζεται, με χρήση των δεδομένων που αναφέρθηκαν παραπάνω όμως, θα πρέπει να προσαρμοστεί σε εφαρμογή Ομοσπονδιακής Μάθησης. Αυτό πραγματοποιείται μέσω προσομοίωσης πραγματικής εφαρμογής Ομοσπονδιακής Μάθησης. Σε μία πραγματική εφαρμογή θα είχαμε συλλογή δεδομένων και εκπαίδευση μοντέλου τοπικά είτε σε επίπεδο συσκευών, cross-device περιπτώσεις, είτε σε επίπεδο συστημάτων οργανισμών, cross-silo περιπτώσεις. Ωστόσο, τώρα που για τον σκοπό της εργασίας, γίνεται προσομοίωση μιας τέτοιας εφαρμογής, δεν υπάρχει παραγωγή των δεδομένων τοπικά αλλά θα πρέπει να διαμελιστούν τα συνολικά δεδομένα από τα παραπάνω datasets, σε κάθε τοπικό σύστημα, όπως περιγράφεται, στο κεφάλαιο διαμελισμού του δεδομένων 3.4. Η υλοποίηση μέσω του κώδικα που παρατίθεται στο Παράρτημα Α βασίζεται σε παρόμοια προσομοίωση [10].

Η προσομοίωση που πραγματοποιείται, διαφέρει αρκετά από μία πραγματική υλοποίηση Ομοσπονδιακής Μάθησης. Αρχικά, η προσομοίωση γίνεται με γνωστούς υπολογιστικούς πόρους και η εκπαίδευση των τοπικών μοντέλων πραγματοποιείται από το ίδιο μηχάνημα κάθε φορά εν αντιθέσει με ένα πραγματικό περιβάλλον Ομοσπονδιακής Μάθησης, όπου η εκπαίδευση γίνεται στα τοπικά συστήματα, όπου είναι και ο κυρίαρχος σκοπός της. Σε αυτό το "εικονικό" σύστημα, πάντα θα υπάρχει, επιπλέον και δεδομένη επικοινωνία μεταξύ ενός τοπικού συστήματος με το κεντρικό, αφού όλα γίνονται σε ένα μηχάνημα και απλά οι παράμετροι των μοντέλων αποθηκεύονται και παραμετροποιούνται κατάλληλα κάθε φορά. Έτσι δεν υπάρχει αποστολή και λήψη τους σε διαφορετικά συστήματα και επομένως δεν υπάρχει και ο κίνδυνος μη πραγματοποίησης κάποιας εκ των διαδικασιών αυτών. Παραδόξως, αυτή η έλλειψη της αβέβαιης σύνδεσης που υπάρχει και είναι ένα ιδιαίτερα σημαντικό και κρίσιμο σημείο μιας πραγματικής υλοποίησης εφαρμογής Ομοσπονδιακής Μάθησης, δεν προκαλεί ιδιαίτερα προβλήματα στην συγκεκριμένη περίπτωση. Αυτό συμβαίνει γιατί η μελέτη αυτή ερευνά τις υλοποιήσεις και την εφαρμογή των αλγορίθμων σε cross-silo σενάρια, όπου, όπως αναφέρεται και στο 2.3.1, υπάρχει μεν πιθανότητα να μην επιτευχθεί σύνδεση με κάποιο τοπικό σύστημα, αλλά η πιθανότητα αυτή είναι πάρα πολύ μικρή σε σχέση με τα cross-device σενάρια. Επιπλέον αφού ο αριθμός των συμμετεχόντων συστημάτων είναι μικρός, μπορεί απλά αυτό το σύστημα που δεν έχει επικοινωνία ή ενδεχομένως να μην μπορεί να έρθει σε επαφή με το κεντρικό σύστημα να αφαιρεθεί από την συνολική διαδικασία της Ομοσπονδιακής Μάθησης.

Η προσομοίωση εφαρμογής Ομοσπονδιακής Μάθησης παρουσιάζει πολλά πλεονεκτήματα αλλά και μειονεκτήματα. Πρώτα και βασικότερα, ευνοεί την διενέργεια πειραμάτων αφού μπορεί να γίνει εύκολη και άμεση αλλαγή των παραμέτρων της εφαρμογής. Από τον αριθμό των συμμετεχόντων συστημάτων μέχρι και την καθολική αλλαγή του είδους του μοντέλου που χρησιμοποιείται μπορεί άνετα και άμεσα να τροποποιηθεί το σύστημα. Ωστόσο, προκειμένου να γίνει αυτό, και κυρίως να γίνει με εύκολο και γρήγορο τρόπο, είναι απαραίτητο να υπάρχει η πρόβλεψη της παραμετροποίησης αυτής. Αυτό απαιτεί την καλή, οργανωμένη και επομένως σωστή ανάπτυξη της προσομοίωσης. Από την άλλη, παρά τις όλες προσπάθειες ανάπτυξης που μπορεί να γίνουν παρουσιάζονται προβλήματα που σε μία πραγματική εφαρμογή δεν θα υπήρχαν. Αρχικά το ίδιο το στήσιμο της προσομοίωσης είναι απαιτητικό αφού όπως ειπώθηκε και πριν, θα πρέπει, έτσι ώστε να εξυπηρετεί και τον σκοπό της η προσομοίωση, να υπάρχει ευελιξία. Κάτι τέτοιο όμως απαιτεί μια εκ των προτέρων σωστή οικοδόμηση της όλης διαδικασίας με σωστή επικοινωνία μεταξύ των διαφόρων παραμέτρων και σωστή προετοιμασία σε περίπτωση αλλαγής ή και απουσίας κάποιας από αυτές. Ακόμα, αρκετά απαιτητικό είναι και ότι θα πρέπει να γίνει σωστή προσομοίωση των συνθηκών που είναι γνωστό πως θα δυσχέραιναν το εγχείρημα, αν επρόκειτο για πραγματική εφαρμογή όπως η ανομοιογένεια των συστημάτων, και των δεδομένων που αυτά παράγουν, (αν και ξανά αυτό το πρόβλημα θα ήταν αρκετά δυσκολότερο στην περίπτωση που δεν

εξετάζονταν τα cross-silo σενάρια). Επιπλέον, ένα σημαντικό μειονέκτημα της συγκεκριμένης τουλάχιστον προσομοίωσης είναι ότι η εκπαίδευση πραγματοποιείται ασύγχρονα. Σε αντίθεση με ένα πραγματικό σενάριο Ομοσπονδιακής Μάθησης όπου τα τοπικά συστήματα πραγματοποιούν την εκπαίδευση των μοντέλων τους παράλληλα και ταυτόχρονα μεταξύ τους, στο περιβάλλον της προσομοίωσης που υλοποιήθηκε, η εκπαίδευση του κάθε ενός τοπικού μοντέλου γίνεται σειριακά για κάθε client που συμμετέχει στον εκάστοτε γύρο επικοινωνίας. Τέλος, όπως και σε όλες τις προσομοιώσεις, ανεξάρτητα από το πρόβλημα που απαιτεί λύση, χάνεται η επαφή με το πραγματικό περιβάλλον διεξαγωγής του προβλήματος και είναι υπερβολικά δύσκολο να προβλεφθούν όλα τα πιθανά ζητήματα που μπορεί να προκύπταν στην περίπτωση της πραγματικής υλοποίησης και θα εμπόδιζαν την ομαλή διεξαγωγή της.

3.4 Διαμελισμός Δεδομένων

Ένα από τα κυρίαρχα προβλήματα της προσομοίωσης της εφαρμογής ΟΜ είναι η ίδια η προσομοίωση της συλλογής των δεδομένων. Όπως έχει ήδη αναφερθεί σε μία πραγματική εφαρμογή Ομοσπονδιακής Μάθησης αυτό το πρόβλημα δεν υπάρχει, αφού τα δεδομένα συλλέγονται από τα τοπικά συστήματα απευθείας, και χρησιμοποιούνται για να εκπαιδευτεί εκεί το μοντέλο, έτσι ώστε να μην γίνεται μεταφορά τους διατηρώντας την ιδιωτικότητα τους. Σε επίπεδο προσομοίωσης όμως, αυτό θα πρέπει να γίνει «εικονικά». Αυτό επιτυγχάνεται μέσω διαμελισμού των συνολικών datasets που αναφέρθηκαν στο κεφάλαιο 3.2. Δηλαδή τα δεδομένα που συλλέγονται από τα πακέτα αυτά θα πρέπει να μοιραστούν στους clients, που συμμετέχουν συνολικά στην διαδικασία, ώστε να έχει το κάθε σύστημα τα δεδομένα που απαιτούνται για την εκπαίδευση του μοντέλου του και κανονικά θα είχε εξαρχής αυτό και μόνο. Στην συγκεκριμένη εργασία οι πειραματικές προσομοιώσεις πραγματοποιούνται για cross-silo περιπτώσεις. Αυτό σημαίνει λιγότεροι χρήστες με κατά κύριο λόγο αυξημένο όγκο δεδομένων σε σχέση με τα cross-device σενάρια. Στις περισσότερες τέτοιες περιπτώσεις εφαρμογών το είδος των δεδομένων που συλλέγονται από κάθε ένα σύστημα είναι γνωστό. Αυτό έχει ως αποτέλεσμα ως επί το πλείστον, με αρκετές βέβαια εξαιρέσεις, να μπορούμε να μετατρέψουμε τα δεδομένα αυτά και να τα κάνουμε πιο εύκολα διαχειρίσιμα πριν τα τροφοδοτήσουμε σε ένα μοντέλο για εκπαίδευση.

Ο διαμελισμός αυτός των δεδομένων έγινε με τρεις μεθόδους για τις δοκιμαστικές εφαρμογές που έγιναν. Ο πρώτος τρόπος που εφαρμόστηκε είναι η ομοιογενής κατανομή δεδομένων. Στην ομοιογενή κατανομή τα δεδομένα από τα datasets διαμοιράζονται σε ίδια ποσότητα, με δεδομένα από ολόκληρο τον αριθμό των διαφορετικών κλάσεων σε ίδια ποσότητα για κάθε κλάση για κάθε σύστημα που συμμετέχει. Επομένως μιλάμε για IID δεδομένα ίσα μοιρασμένα στους clients που συμμετέχουν συνολικά στην διαδικασία της μάθησης. Ωστόσο, αυτό γίνεται όσο είναι δυνατό, αφού ανάλογα με τον αριθμό των δεδομένων και τον αριθμό των συμμετεχόντων ενδέχεται να διαφέρει λίγο ο αριθμός δεδομένων ανά κλάση και η ποσότητα στους διάφορους clients αφού η διαίρεση τους δεν μπορεί να είναι ακριβής ακέραιος αριθμός, ώστε να μοιραστούν απόλυτα ίσα. Ο δεύτερος τρόπος που χρησιμοποιήθηκε είναι η διαφορετική κατανομή ως προς τον αριθμό των δεδομένων ανά συμμετέχον σύστημα. Τα δεδομένα εδώ διαμοιράζονται με διαφορετική ποσότητα σε κάθε τοπικό σύστημα διατηρώντας όμως την συμμετοχή όλων των κλάσεων που αυτά να ανήκουν, με αριθμούς δεδομένων ανά κλάση πάλι με μικρές διαφορές μεταξύ τους για τον ίδιο λόγο με την πρώτη περίπτωση δεδομένου όμως του αριθμού των συνολικών δεδομένων που δίνονται στον κάθε client. Αυτό επιτυγχάνεται μέσω εφαρμογής της κατανομής Dirichlet, για την δημιουργία του αριθμού των δεδομένων που θα μοιραστεί στους διάφορους clients. Εδώ και πάλι αναφερόμαστε σε IID δεδομένα αφού μπορεί να διαφέρει ο αριθμός των δεδομένων που έχει ένας client σε σχέση με κάποιον άλλο αλλά και πάλι υπάρχουν όλες και οι ίδιες κλάσεις σε όλα τα συστήματα που παίρνουν μέρος στην ΟΜ. Η τρίτη περίπτωση αφορά δεδομένα non-

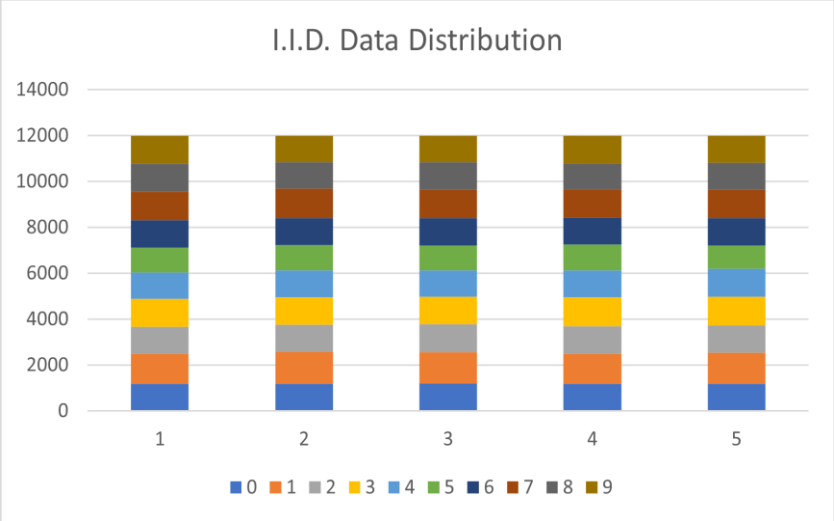
IID. Τα δεδομένα αυτά δημιουργούνται και πάλι με χρήση της κατανομής Dirichlet, ωστόσο πλέον όχι μόνο διαφέρει ο συνολικός αριθμός των δεδομένων αλλά επίσης διαφέρει ο αριθμός αυτών που υπάρχει σε κάθε μία από τις κλάσεις αλλά και ο αριθμός των ίδιων των κλάσεων δεδομένων που έχει το κάθε σύστημα. Έτσι πάμε σε ένα καθαρά πραγματικό σενάριο. Παρ' όλα αυτά, δεν πρέπει να ξεχνάμε ότι σε αυτή τη μελέτη το ενδιαφέρον είναι μόνο για cross-silo σενάρια. Γι' αυτόν τον λόγο, σε αυτή την διαδικασία διαμοιρασμού των δεδομένων, παρά τις ανομοιογένειες των δεδομένων που αναφέρθηκαν, δημιουργούνται αριθμοί έτσι ώστε να υπάρχει μια σχετική ισορροπία μεταξύ των συστημάτων, χωρίς να δημιουργηθούν μεγάλες ή ακραίες διαφορές τόσο στον αριθμό των διαφορετικών κλάσεων σε κάθε clients όσο και για την συνολική ποσότητα δεδομένων στον κάθε ένα. Για παράδειγμα δεν πρόκειται να μοιραστούν τα δεδομένα έτσι ώστε ένας client να έχει 10 κλάσεις δεδομένων και κάποιος άλλος 2 ή 3 αλλά όντως θα υπάρξουν οι διαφορές που αναφέρθηκαν στα δεδομένα απλώς σε μικρότερο βαθμό [11].

Με αυτούς τους τρόπους, προσομοιάζονται οι διαφορετικές περιπτώσεις συλλογής δεδομένων που μπορεί να υπάρχουν σε μία πραγματική εφαρμογή όπου ο αριθμός των δεδομένων ανά σύστημα ή τα είδη αυτών μπορεί να διαφέρουν αφού δεν υπάρχει βεβαιότητα, ότι κάθε ένα από τα συμμετέχοντα συστήματα θα είναι σε θέση να συλλέξει συγκεκριμένο αριθμό δεδομένων, ούτε πως θα έχει όλα ή και τα ίδια είδη των δεδομένων σε ίδια μεγέθη που πιθανώς να υπάρχουν σε κάποιο άλλο σύστημα.

Για όλες τις παραπάνω περιπτώσεις και πιο τεχνικά, στην υλοποίηση, ο διαμελισμός από τα συνολικά δεδομένα που περιέχουν τα datasets, επιτυγχάνεται με την δημιουργία λιστών που περιέχουν τους αύξοντες αριθμούς (ids) των εικόνων που θα έχει διαθέσιμες για εκπαίδευση το κάθε σύστημα. Αρχικά, δημιουργείται ένας αρχικός μονοδιάστατος πίνακας που περιέχει την κλάση της κάθε εικόνας, για όλες τις εικόνες που περιέχονται μέσα στο πακέτο δεδομένων που χρησιμοποιείται. Η επιλογή της λίστας αυτής με τις κλάσεις που αντιστοιχούν σε κάθε εικόνα και όχι με τις ίδιες τις εικόνες, γίνεται για ευκολία, καθώς οι εικόνες αποτελούν η κάθε μία ένα πίνακα μερικών διαστάσεων ενώ η κλάση της είναι απλά ένας αριθμός. Στην συνέχεια από αυτή τη λίστα δημιουργείται μία άλλη που έχει αριθμήσει την προηγούμενη και επομένως έχει όλους τους αύξοντες αριθμούς που αντιστοιχούν σε κάθε δεδομένο του κάθε dataset, που χρησιμοποιείται κάθε φορά. Τα περιεχόμενα αυτής της λίστας αναδιοργανώνονται σε τυχαίες θέσεις, και ύστερα χωρίζεται ανάλογα με τα μεγέθη που ειπώθηκαν παραπάνω. Δηλαδή ανάλογα την μέθοδο που χρησιμοποιείται τα δεδομένα είτε θα χωριστούν απλά ισομερώς σε n μέρη , όπου n ο αριθμός όλων των clients ($1^η$ περίπτωση, ομοιογενής κατανομή δεδομένων), είτε σε διαφορετικές ποσότητες για όλα τα δεδομένα και ανά κλάση για την $2^η$ και $3^η$ περίπτωση αντίστοιχα.

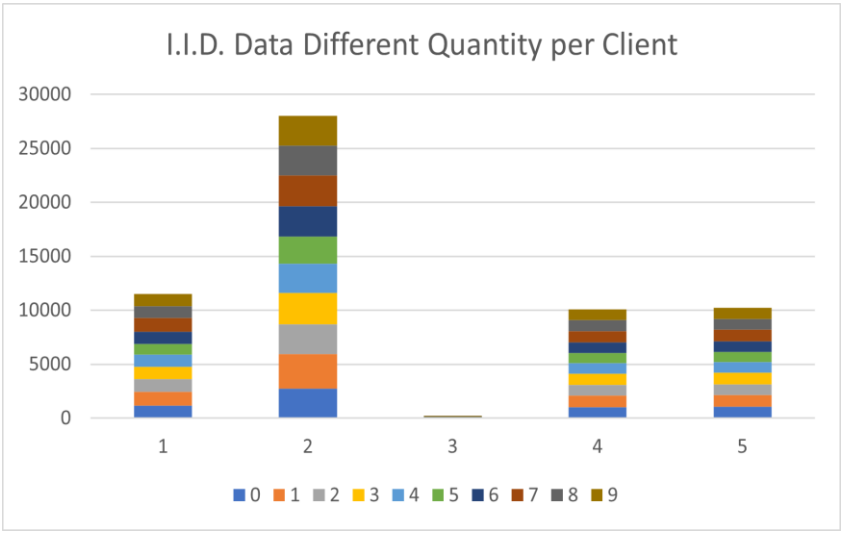
Συνολικά, η διαδικασία που γίνεται για την κάθε περίπτωση από αυτές που ειπώθηκαν, είναι ότι δημιουργείται για κάθε client μια λίστα που περιέχει μόνο τα ids των δεδομένων που θα έχει ο εξής client, και στην συνέχεια λαμβάνονται για κάθε client από το dataset, μόνο τα δεδομένα που του αναλογούν σύμφωνα με τα ids τους.

1ος Τρόπος διαμοιρασμού των δεδομένων – ομοιογενής κατανομή για 10 κλάσεις[4]-IID Data:



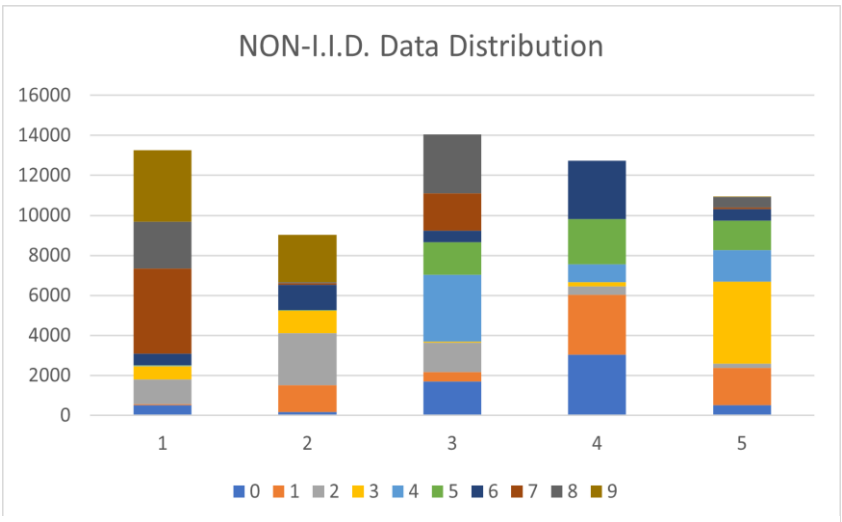
Διάγραμμα 3.4-1 IID Homogenous Distribution [4]

2ος Τρόπος διαμοιρασμού των δεδομένων – ομοιογενής κατανομή με διαφορετική ποσότητα δεδομένων ανά Client
IID Different Q:



Διάγραμμα 3.4-2 IID Different Quantity Distribution [4]

3ος Τρόπος διαμοιρασμού των δεδομένων – κατανομή με non-IID Δεδομένα non-IID:



Διάγραμμα 3.4-3 Non-IID Distribution [4]

[4] 0,1,2,3,4,5,6,7,8,9 οι διάφορες κλάσεις των δεδομένων.

ΚΕΦΑΛΑΙΟ 4 ° : Μοντέλα – Τεχνητά Νευρωνικά Δίκτυα

Για την υλοποίηση των πειραμάτων χρησιμοποιήθηκαν μοντέλα τεχνητών νευρωνικών δικτύων. Συγκεκριμένα δύο ειδών. Αρχικά χρησιμοποιήθηκε ένα συνελικτικό νευρωνικό δίκτυο, βασικής αρχιτεκτονικής και ένα Residual νευρωνικό δίκτυο που αναλύονται στην συνέχεια. Τα δύο αυτά είδη νευρωνικών μοντέλων αποτελούν δύο από τις πρώτες επιλογές και θεωρούνται τα πιο κατάλληλα για την επίλυση προβλημάτων ταξινόμησης εικόνων. Στην προσομοίωση που έγινε είναι δυνατό να διεξαχθούν οι συγκρίσεις αλγορίθμων συσσωμάτωσης σε σενάρια και για τα δύο μοντέλα, με ανάλογη επιλογή, με σκοπό την εξαγωγή καλύτερων συμπερασμάτων ως προς την λειτουργία και την αποδοτικότητα των αλγορίθμων.

4.1 Τεχνητά Νευρωνικά Δίκτυα

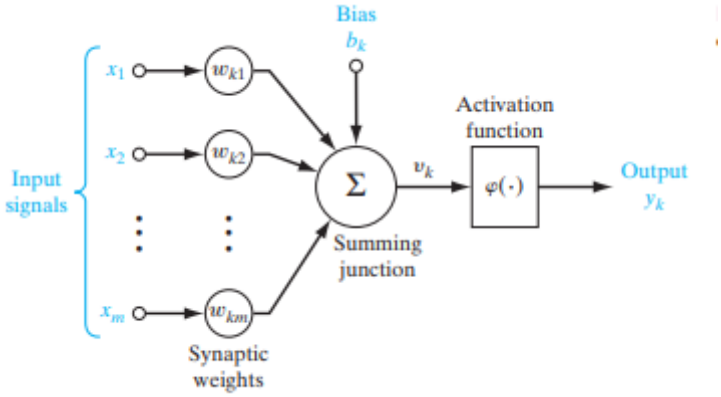
Τα τεχνητά νευρωνικά δίκτυα αποτελούν τον πιο χρησιμοποιούμενο και διαδεδομένο τύπο μοντέλου τεχνητής νοημοσύνης, και ειδικότερα μηχανικής μάθησης. Τα (τεχνητά) νευρωνικά δίκτυα είναι πλέον συνώνυμα της μηχανικής μάθησης και οι περισσότερες εφαρμογές της υλοποιούνται μέσω αυτών. Τα νευρωνικά δίκτυα αποτελούν το αποτέλεσμα της μελέτης του ανθρώπινου εγκεφάλου και του γεγονότος ότι όχι μόνο λειτουργεί τελείως διαφορετικά από έναν υπολογιστή αλλά και το ότι εκτελεί εξαιρετικά πολύπλοκες λειτουργίες, πολύ γρήγορα.

Αρχικά η βασική έμπνευση και λειτουργία των νευρωνικών δικτύων βασίζονται στους νευρώνες και πώς αυτοί παράγουν το αποτέλεσμα τους, μεταφέροντας πληροφορίες. Η μεταφορά αυτή πραγματοποιείται μέσω ηλεκτρικών σημάτων και το σήμα αυτό μεταδίδεται από τον ένα νευρώνα στον άλλο μέσω σύνδεσης των δύο αυτών νευρώνων. Η δημιουργία αυτής της σύνδεσης αποτελεί την διαδικασία της μάθησης σε ένα βιολογικό οργανισμό, με τις συνδέσεις που αξιοποιούνται περισσότερο να γίνονται ισχυρότερες ενώ αντίθετα οι λιγότερο χρήσιμες συνδέσεις εξασθενούν με τελικό στάδιο την αποσύνθεση τους [12].

Αυτές οι συνδέσεις και η μεταφορά του ηλεκτρικού σήματος γίνεται με την χρήση τριών κυρίαρχων μερών των νευρώνων. Τα μέρη αυτά είναι: οι δενδρίτες, το σώμα και ο άξονας του νευρώνα. Οι δενδρίτες λαμβάνουν τα σήματα των άλλων νευρώνων, μέσω μίας χημικής διαδικασίας η οποία αλλάζει την συχνότητα του σήματος αυτού. Στην συνέχεια, το σώμα είναι υπεύθυνο για την συνολική άθροιση των εισερχόμενων αυτών σημάτων. Ο άξονας του νευρωνικού κυττάρου αποτελεί το μέσο, από το οποίο τροφοδοτούνται με το τελικό σήμα που παράχθηκε οι επόμενοι νευρώνες. Το σήμα όμως που εξέρχεται από ένα νευρώνα θεωρείται ότι δεν είναι συνεχές αλλά είτε μεταφέρεται σήμα, είτε όχι. Η ερμηνεία αυτών των λειτουργιών των νευρώνων και της μάθησης ενέπνευσε στις αρχές της δεκαετίας του 1940 τους μαθηματικούς Warren McCulloch και Walter Pitts να σχεδιάσουν το πρώτο τεχνητό νευρωνικό δίκτυο μέσω ηλεκτρικών κυκλωμάτων με σκοπό να προσομοιωθεί η λειτουργία του εγκεφάλου, πυροδοτώντας έτσι την έναρξη της εξέλιξης και χρήσης των τεχνητών νευρωνικών δικτύων [12], [13].

Σε αντιστοιχία, ο τεχνητός νευρώνας έχει λειτουργίες που προσομοιάζουν αυτές του βιολογικού. Αρχικά, ένας τεχνητός νευρώνας έχει τα συναπτικά βάρη τα οποία επιτελούν παρόμοια λειτουργία με την αλλαγή συχνότητας των σημάτων που εισέρχονται που πραγματοποιείται στους δενδρίτες. Αποτελούν, δηλαδή ένα συντελεστή με τον οποίο θα πολλαπλασιαστεί το κάθε σήμα που εισέρχεται στον νευρώνα, δίνοντας έτσι «αξία» σε αυτό. Μετά έχει σειρά το τεχνητό σώμα του νευρώνα, ο κόμβος άθροισης που τελεί όμοια λειτουργία με αυτή του βιολογικού σώματος νευρώνα, αθροίζει δηλαδή τα όλα τα σήματα που έχουν επέλθει μετά τον πολλαπλασιασμό τους με τα συναπτικά βάρη. Επιπλέον, χρειάζεται άλλος ένας μηχανισμός ώστε να μιμηθεί την μεταφορά ή όχι του σήματος, δηλαδή τον έλεγχο μίας

δυαδικής λειτουργίας. Αυτό τον ρόλο έχει η συνάρτηση ενεργοποίησης που ελέγχει αν θα μεταδοθεί σήμα και πόσο θα είναι αυτό.



Εικόνα 4.1.1 Μοντέλο Τεχνητού Νευρώνα [14]

Συγκεντρωτικά σε ένα τεχνητό νευρώνα βρίσκονται:

- Συνάψεις, όπου η κάθε μία έχει το δικό της βάρος. Ειδικότερα, πολλαπλασιάζεται το βάρος της κάθε σύναψης w_n με το σήμα που έρχεται στην είσοδο της, όπου το βάρος αυτό ενισχύει ή υποβαθμίζει το σήμα αυτό. Μπορεί να πάρει θετική ή αρνητική τιμή.

- Αθροιστής, που αθροίζει όλα τα σήματα που εισέρχονται στον νευρώνα μετά τον πολλαπλασιασμό τους με από τα βάρη των συνάψεων.

- Συνάρτηση Ενεργοποίησης, καθορίζει αν θα μεταδοθεί σήμα από τον νευρώνα προς του άλλους και κυρίαρχα κανονικοποιεί το πλάτος του σήματος, οριοθετεί δηλαδή την τιμή του παραγόμενου σήματος [14].

Επιπλέον όπως φαίνεται και στην Εικόνα 4.1.1 χρησιμοποιείται και μία πόλωση η οποία προστίθεται μαζί με τα σταθμισμένα σήματα από τον αθροιστή. Αυτή είναι αναγκαία προκειμένου να γίνει μία «χειραγώγηση» της συνάρτησης ενεργοποίησης αφού μέσω αυτής τιμής μετατοπίζεται η συνάρτηση ενεργοποίησης ανάλογα με το επιθυμητό αποτέλεσμα.

Κατά την διάρκεια της εκπαίδευσης των νευρωνικών δικτύων, συνήθως οι παράμετροι που αλλάζουν είναι τα συναπτικά βάρη και η πόλωση (εκτός από συγκεκριμένες περιπτώσεις, όπου μπορεί να αλλάζει και η συνολική δομή τους). Αρχικά, τα βάρη και η πόλωση παίρνουν τυχαίες τιμές δοκιμάζονται και στην συνέχεια προσαρμόζονται με μηχανισμούς και αλγόριθμους βελτιστοποίησης, όπως ο πολύ συχνά χρησιμοποιούμενος αλγόριθμος SGD (Stochastic Gradient Descent), που περιγράφεται η λειτουργία του στο κεφάλαιο 5.1.

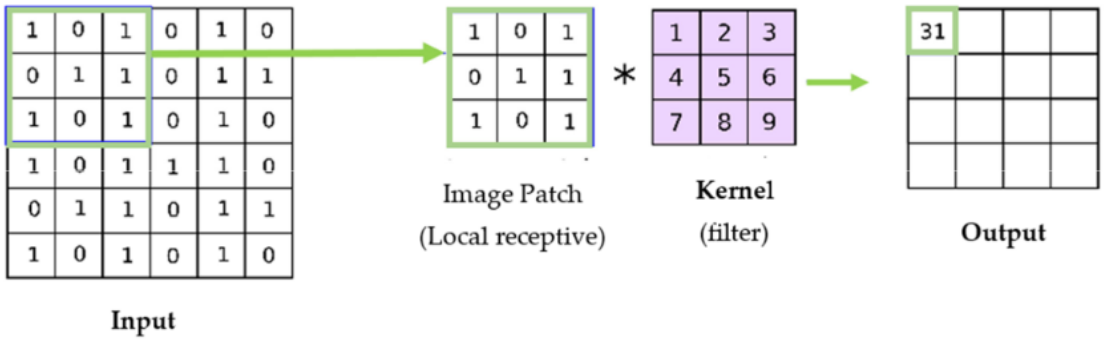
Τα νευρωνικά δίκτυα αποτελούνται από πολλούς νευρώνες. Οι νευρώνες αυτοί οργανώνονται σε κάθε δίκτυο σε στοιβάδες. Οι στοιβάδες έχουν να κάνουν με την οργάνωση των νευρώνων. Δηλαδή, τον ίδιο τον αριθμό των νευρώνων που θα υπάρχουν στην συγκεκριμένη στοιβάδα, τον αριθμό των συνάψεων στην είσοδο τους, την έξοδο τους ακόμα και την συνάρτηση ενεργοποίησης που θα χρησιμοποιούν. Έτσι οι στοιβάδες αυτές αναλύονται σε στοιβάδες εισόδου, εξόδου και τις κρυφές στοιβάδες που βρίσκονται μεταξύ των αυτών των δύο.

4.2 Συνελικτικό Νευρωνικό Δίκτυο

Τα Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ) ή αλλιώς Convolutional Neural Network (CNN) αποτελούν βασικό είδος νευρωνικών δικτύων σε εφαρμογές ανάλυσης εικόνων. Έχοντας αναπτυχθεί εδώ και δεκαετίες είναι ένα αναπόσπαστο κομμάτι σε προβλήματα υπολογιστικής μάθησης και τεχνητής νοημοσύνης. Η αρχιτεκτονική τους αποτελείται από πολλά επίπεδα νευρώνων περιέχοντας ωστόσο 3 βασικά επίπεδα. Τα 3 αυτά βασικά επίπεδα ενός συνελικτικού δικτύου είναι:

4.2.1 Επίπεδο Συνέλιξης

Το βασικότερο επίπεδο νευρώνων, απ' όπου παίρνουν και το όνομα τους τα συνελικτικά νευρωνικά δίκτυα είναι το επίπεδο συνέλιξης, στη διεθνή ορολογία Convolutional Layer. Το επίπεδο αυτό λειτουργεί εφαρμόζοντας φίλτρα (kernels) στα διανύσματα που εισάγονται. Αρχικά μέσω του φίλτρου και των



Εικόνα 4.2.1 Συνέλιξη [15]

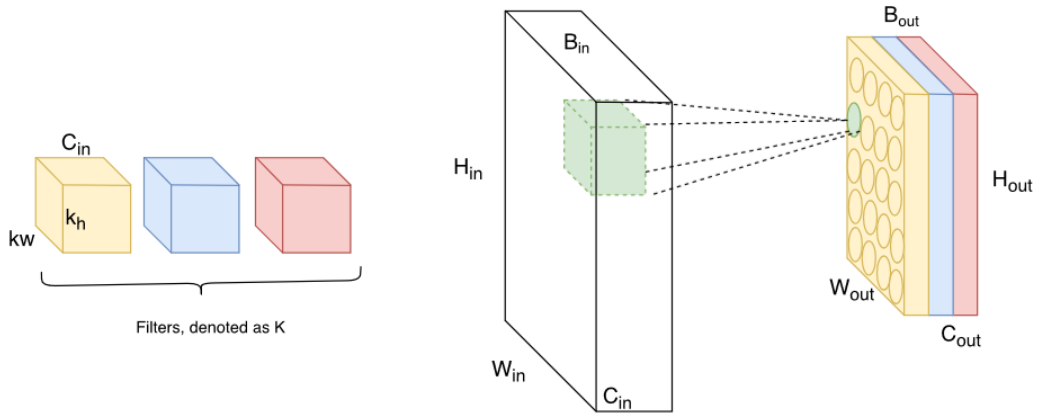
διαστάσεων του, οι οποίες ορίζονται εξαρχής, "επικαλύπτοντας" το διάνυσμα που έχει εισαχθεί, χωρίζει το διάνυσμα αυτό σε μικρότερων διαστάσεων, ίσων με του φίλτρου, διανύσματα. Στην συνέχεια πολλαπλασιάζονται τα στοιχεία του διανύσματος που έχει δημιουργηθεί με τους συντελεστές του φίλτρου ένα προς ένα και στην συνέχεια προσθέτονται με σκοπό να παραχθεί ένας συντελεστής στον τελικό πίνακα που δημιουργήθηκε πραγματοποιώντας δηλαδή πράξη εσωτερικού γινομένου (Εικόνα 4). Η ίδια διαδικασία θα πραγματοποιηθεί ξανά μέχρις ότου όλο το εισαγόμενο στο επίπεδο διάνυσμα να έχει επικαλυφθεί με βήμα μετακίνησης (stride) που ορίζεται. Για παράδειγμα σύμφωνα με την εικόνα 4, το πρώτο υπό-διάνυσμα που προκύπτει είναι το: $\begin{bmatrix}1 & 0 & 1\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix}$

και η πράξη που θα γίνει:

$$\begin{bmatrix}1 & 0 & 1\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix} = 1\cdot 1+0\cdot 2+1\cdot 3+0\cdot 4+1\cdot 5+1\cdot 6+1\cdot 7+0\cdot 8+1\cdot 9=31$$

ενώ το δεύτερο υπό-διάνυσμα με βήμα 1 είναι το: $\begin{bmatrix}0 & 1 & 0\\1 & 1 & 0\\0 & 1 & 0\end{bmatrix}$, και παρόμοια τα υπόλοιπα. Οι συντελεστές του κάθε φίλτρου ανήκουν στην διαδικασία της εκπαίδευσης και αλλάζουν ανά βήμα εκπαίδευσης.

Μέσω της συνέλιξης υπερτονίζονται τα σημεία των εικόνων που αλλάζει το μοτίβο. Δηλαδή επισημαίνονται τα όρια των διάφορων χαρακτηριστικών που μπορεί να υπάρχουν. Αυτό το αποτέλεσμα είναι πολύ χρήσιμο στην κατηγοριοποίηση των εικόνων αλλά και για την εύρεση διαφορετικών γνωρισμάτων μέσα σε μία εικόνα. Όταν αυτό το αποτέλεσμα τροφοδοτείται είτε σε επιπλέον στοιβάδες συνέλιξης είτε στις επόμενες στοιβάδες απλοποιείται αρκετά η ανάλυση των δεδομένων [15] [16] [17].

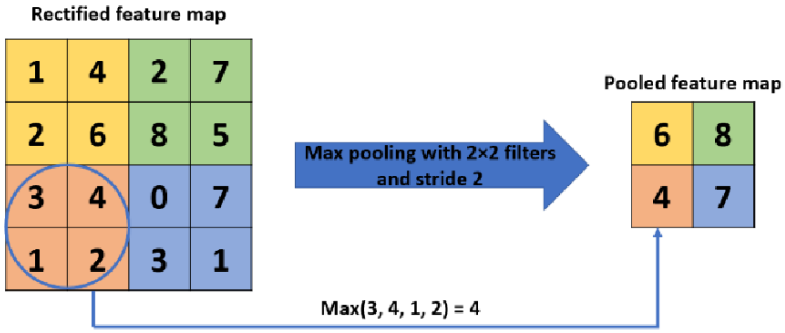


Εικόνα 4.2.2 Συνέλιξη σε πολλές διαστάσεις [19]

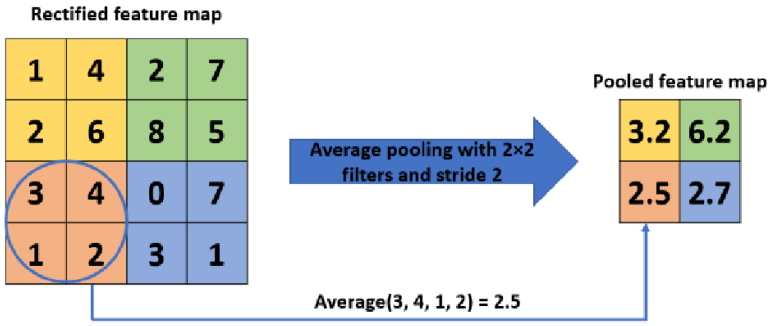
4.2.2 Επίπεδο Ομαδοποίησης/ Συγκέντρωσης

Το δεύτερο πιο κοινό επίπεδο νευρώνων στα συνελικτικά νευρωνικά δίκτυα στα αγγλικά Pooling Layer είναι υπεύθυνο για να μειώσει τον αριθμό των συντελεστών και επομένως να απλοποιήσει το διάνυσμα που εισέρχεται σε αυτό. Λειτουργεί και αυτό βασιζόμενο σε φίλτρα. Με παρόμοια διαδικασία με το επίπεδο συνέλιξης τα φίλτρα ορισμένων διαστάσεων επικαλύπτουν όλο το εισερχόμενο διάνυσμα με δεδομένο και πάλι βηματισμό.

Τα πιο διαδεδομένα είδη του επιπέδου είναι τα max-polling και average-pooling. Όπως μαρτυρούν και τα ονόματα τους το max-polling καταχωρεί στο νέο διάνυσμα που δημιουργείται το μέγιστο συντελεστή από τους συντελεστές που λαμβάνονται υπόψη με βάση τις διαστάσεις του φίλτρου (Εικόνα 4.2.3). Αντίθετα το average-pooling καταχωρεί τον μέσο όρο των συντελεστών, ξανά με βάση τις διαστάσεις του φίλτρου (Εικόνα 4.2.4) [16] [18].



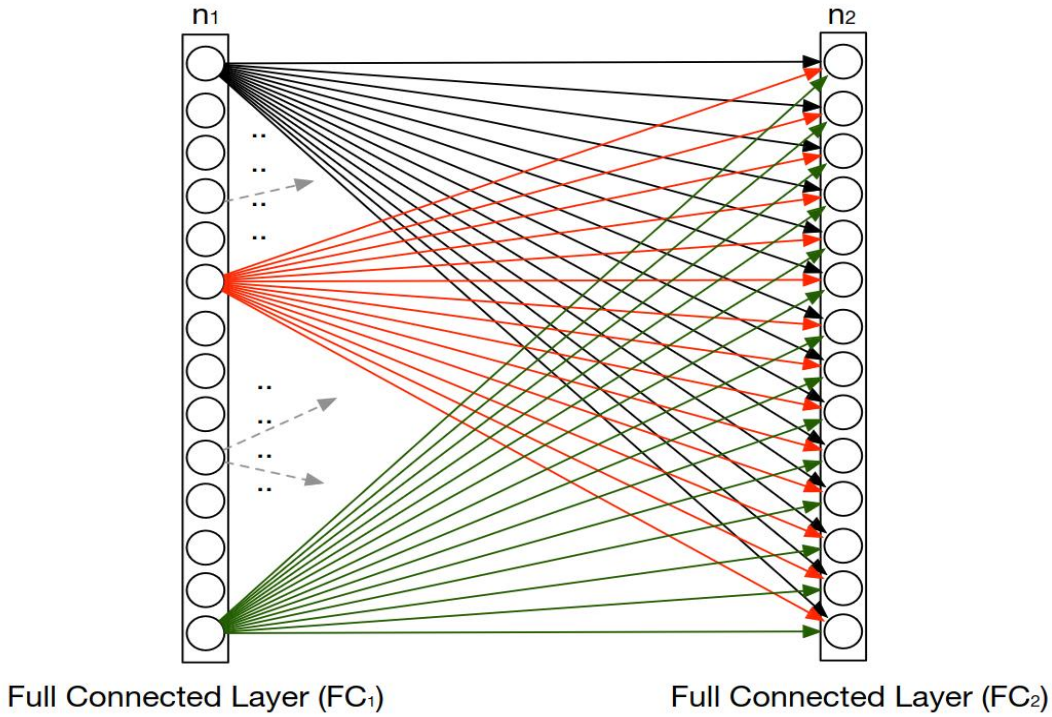
Εικόνα 4.2.3 Max Pooling [18]



Εικόνα 4.2.4 Average Pooling [18]

4.2.3 Πλήρως Συνδεδεμένο Επίπεδο

Στο επίπεδο αυτό, που στα αγγλικά λέγεται Fully Connected layer, κάθε κόμβος νευρώνα είναι άμεσα συνδεδεμένος με έναν εξωτερικό νευρώνα του επόμενου επιπέδου, Συνήθως εφαρμόζεται μια γραμμική σχέση σαν συνάρτηση ενεργοποίησης (Linear) δημιουργώντας έτσι ένα αποτέλεσμα της μορφής $x2 = w * x1 + b$, όπου το w αποτελεί συντελεστή εκπαίδευσης και το b (bias), την πόλωση. Συνήθης επίσης είναι και η σχετική γραμμική σχέση (ReLu) όπου έχει την ίδια συμπεριφορά με την γραμμική συνάρτηση κάνοντας 0 όμως όλες τις τιμές που είναι μικρότερες ή ίσες του 0 [19] [17].



Εικόνα 4.2.5 Πλήρως Συνδεδεμένα Επίπεδα [19]

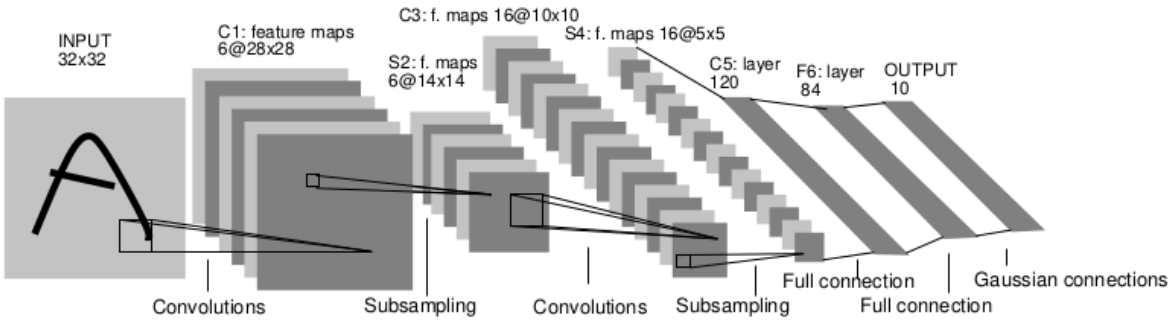
4.2.4 Δομή για Simple CNN:

Ειδικότερα, το μοντέλο που χρησιμοποιήθηκε αποτελεί ένα από τα πιο απλά συνελικτικά νευρωνικά δίκτυα, βασισμένο στο LeNet-5 (Εικόνα 4.2.6) ένα νευρωνικό δίκτυο που προτάθηκε για πρώτη φορά το 1989 [20]. Η αρχιτεκτονική του νευρωνικού δικτύου (SimpleCNN), αποτελείται από 6 επίπεδα και είναι η εξής:

Επίπεδο 1: 2 Διαστάσεων Συνέλιξη με είσοδο τον αριθμό των καναλιών της κάθε εικόνας (1 για εικόνες σε κλίμακα του γκρι και 3 για έγχρωμες εικόνες), διαστάσεις φίλτρου (kernel) 5x5 και 6 κανάλια εξόδου.
Επίπεδο 2: 2 Διαστάσεων Max Pooling, με μέγεθος φίλτρου 2x2 και βήμα 2
Επίπεδο 3: 2 Διαστάσεων Συνέλιξη με είσοδο τον αριθμό των καναλιών της εξόδου του πρώτου επιπέδου (6) που έχουν περάσει από το 2 επίπεδο και έχουν πλέον μέγεθος 14x14 το κάθε ένα, διαστάσεις φίλτρου (kernel) 5x5 και 16 κανάλια εξόδου

Στην συνέχεια ακολουθούν τα πλήρως συνδεδεμένα επίπεδα:

Επίπεδο 4: Πλήρως Συνδεδεμένο επίπεδο με 400 νευρώνες εισόδου και 120 εξόδου.
Επίπεδο 5: Πλήρως Συνδεδεμένο επίπεδο με είσοδο 120 νευρώνες εισόδου από το επίπεδο 5 και 84 εξόδου.
Επίπεδο 6: Πλήρως Συνδεδεμένο επίπεδο με 84 νευρώνες εισόδου από το επίπεδο 5 εξόδου ίσο με τον αριθμό των κλάσεων που ομαδοποιούμε τα δεδομένα.



Εικόνα 4.2.6 Αρχιτεκτονική του LeNet-5 [20]

Στον παρακάτω Πίνακας 4.2-1 υπάρχει η απόδοση του πρώτου μοντέλου, του συνελικτικού νευρωνικού δικτύου που δημιουργήθηκε σύμφωνα με τα παραπάνω. Οι αποδόσεις αυτές είναι αποτέλεσμα των προβλέψεων του μοντέλου σε σχέση με τις πραγματικές κλάσεις των δεδομένων που ανήκουν στα testing data για κάθε ένα από τα τρία dataset που χρησιμοποιήθηκαν. Παρουσιάζονται ενδεικτικά οι αποδόσεις μετά από 5, 10 και 20 επαναλήψεις εκπαίδευσης (epochs) και για μέγεθος παρτίδων δεδομένων (batch-size) 10.

Πίνακας 4.2-1 Ποσοστιαία Απόδοση Συνελικτικού Νευρωνικού Δικτύου

EPOCHS	MNIST	CIFAR10	CIFAR100
5	98.51	61.83	23.29
10	98.88	64.20	28.45
20	98.97	63.61	29.61

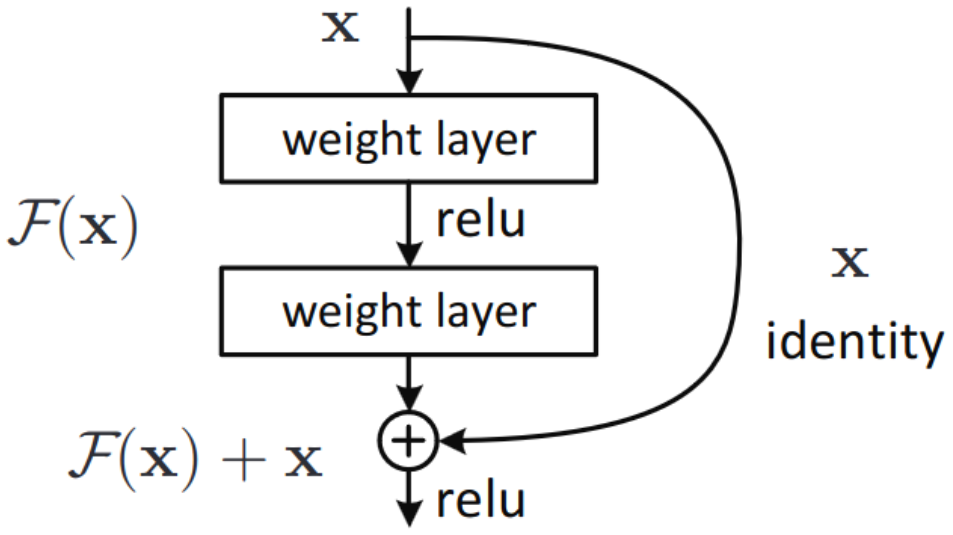
4.3 Residual Neural Network

Το δεύτερο είδος νευρωνικού μοντέλου που χρησιμοποιήθηκε είναι το Residual Neural Network (ResNet). Ουσιαστικά πρόκειται για υποκατηγορία των συνελικτικών δικτύων και έχουν και αυτά ως βασικό συστατικό το επίπεδο συνέλιξης που ειπώθηκε στο 4.2.1. Η εισαγωγή στην αρχιτεκτονική αυτή έγινε το 2015 [21].

Η αφορμή για την εύρεση της αρχιτεκτονικής αυτής ήταν ότι στα πολυεπίπεδα συνελικτικά δίκτυα όταν εφαρμόζονταν παραπάνω από 19 επίπεδα η απόδοση τους έπεφτε ραγδαία. Αυτό είναι αποτέλεσμα του προβλήματος "vanishing gradients" δηλαδή της "εξαφάνισης" - ελαχιστοποίησης των παραγώγων που χρησιμοποιούνται από τους αλγόριθμους βελτιστοποίησης κατά την εκπαίδευση των δικτύων. Έτσι οι συντελεστές/βάρη των μοντέλων δεν αλλάζουν καθιστώντας την διαδικασία της εκπαίδευσης εξαιρετικά δύσκολη. Συνεπώς το μοντέλο δεν "μαθαίνει" πλέον από τα δεδομένα που του παρέχονται επηρεάζοντας σημαντικά την απόδοση του [21].

4.3.1 Residual Block

Η λύση που δόθηκε γι' αυτό είναι οι συνδέσεις παράκαμψης/ skip connections, δημιουργώντας τα Residual Blocks (Εικόνα 4.3.1). Αυτά αποτελούνται από 2 επίπεδα συνέλιξης στα οποία η είσοδος που τροφοδοτείται σε αυτά περνά αυτούσια και στην έξοδο τους, με την τελική έξοδο του block να είναι το άθροισμα της εισόδου με την έξοδο των επιπέδων συνέλιξης. Με τον τρόπο αυτό σε περίπτωση που εμφανιστεί το πρόβλημα vanishing gradients που αναφέρθηκε και προκαλέσει να γίνουν οι αλλαγές των βαρών μηδενικές ή σχεδόν μηδενικές, η συνάρτηση αλλαγής στην έξοδο του block δεν θα είναι ποτέ 0 αφού θα προστεθεί με την είσοδο του. Αυτό επιτρέπει την εισαγωγή απεριόριστου αριθμού επιπέδων στο δίκτυο αφού εξαλείφεται ο κίνδυνος εμφάνισης του προβλήματος αυτού - θεωρητικά, αφού μπορεί να προκύψουν άλλα προβλήματα όπως overfitting κ.ά. - [21].



Εικόνα 4.3.1 Residual Block [21]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Εικόνα 4.3.2 ResNet Architectures [21]

4.3.2 Δομή για ResNet18:

Ειδικότερα, το μοντέλο που χρησιμοποιήθηκε είναι το ResNet18, ένα Residual Νευρωνικό Δίκτυο με 18 επίπεδα. Η αρχιτεκτονική που ακολουθήθηκε περιγράφεται αναλυτικά στην εργασία [21], και φαίνεται στην παραπάνω εικόνα, στην στήλη για τα 18 επίπεδα.

Επίπεδο 1: 2 Διαστάσεων Συνέλιξη με είσοδο τον αριθμό των καναλιών της κάθε εικόνας (1 για εικόνες σε κλίμακα του γκρι και 3 για έγχρωμες εικόνες), διαστάσεις φίλτρου (kernel) 7x7, βηματισμό (stride) 2.

Επίπεδο 2: 2 Διαστάσεων Max Pooling, με μέγεθος φίλτρου 3x3 και βήμα 1

Στην συνέχεια τα επίπεδα με τα Residual Blocks:

Επίπεδο 3: 2 x Res Block, που περιέχουν: 2 υπό-επίπεδα, με το κάθε ένα να είναι 2 Διαστάσεων Συνέλιξη με φίλτρο 3x3 και έξοδο 128 κανάλια.

Επίπεδο 4: 2 x Res Block, που περιέχουν: 2 υπό-επίπεδα, με το κάθε ένα να είναι 2 Διαστάσεων Συνέλιξη με φίλτρο 3x3 και έξοδο 256 κανάλια.

Επίπεδο 5: 2 x Res Block, που περιέχουν: 2 υπό-επίπεδα, με το κάθε ένα να είναι 2 Διαστάσεων Συνέλιξη με φίλτρο 3x3 και έξοδο 64 κανάλια.

Επίπεδο 6: 2 x Res Block, που περιέχουν: 2 υπό-επίπεδα, με το κάθε ένα να είναι 2 Διαστάσεων Συνέλιξη με φίλτρο 3x3 και έξοδο 512 κανάλια.

Τέλος, τα:

Επίπεδο 7: 2 Διαστάσεων Average Pooling, με έξοδο διάνυσμα διαστάσεων 1x1 για 512 διανύσματα εισόδου από το επίπεδο 6.

Επίπεδο 8: Πλήρως Συνδεδεμένο επίπεδο με 512 νευρώνες εισόδου από το επίπεδο 7 και εξόδου ίσο με τον αριθμό των κλάσεων που ομαδοποιούμε τα δεδομένα.

Επιπλέον, θα πρέπει να αναφερθεί πως τόσο στην εσωτερική αρχιτεκτονική του Residual Block ανάμεσα στα συνελικτικά επίπεδα, όσο και του γενικότερου Residual δικτύου, πριν το επίπεδο 2, χρησιμοποιήθηκαν επίπεδα δισδιάστατης κανονικοποίησης (BatchNorm 2d) και σχετικής γραμμικής σχέσης (ReLu). Αυτό έγινε καθώς βελτιώνουν την απλοποιούν και βελτιώνουν την διαδικασία της εκπαίδευσης του δικτύου.

Στον παρακάτω Πίνακας 4.3-1, όπως και με το πρώτο μοντέλο, φαίνεται η απόδοση του δεύτερου μοντέλου, του residual νευρωνικού δικτύου που δημιουργήθηκε σύμφωνα με τα παραπάνω. Οι αποδόσεις είναι και πάλι, αποτέλεσμα των προβλέψεων του μοντέλου σε σχέση με τις πραγματικές κλάσεις των δεδομένων που ανήκουν στα testing data για κάθε ένα από τα τρία dataset που χρησιμοποιήθηκαν. Παρουσιάζονται και για αυτό το δίκτυο, ενδεικτικά οι αποδόσεις μετά από 5, 10 και 20 επαναλήψεις εκπαίδευσης (epochs) και για μέγεθος παρτίδων δεδομένων (batch-size) 10.

Πίνακας 4.3-1 Ποσοστιαία Απόδοση ResNet Νευρωνικού Δικτύου

EPOCHS	MNIST	CIFAR10	CIFAR100
5	99.27	73.50	41.67
10	99.14	75.71	44.42
20	99.45	76.78	44.52

ΚΕΦΑΛΑΙΟ 5 ° : Αλγόριθμοι Συσσωμάτωσης

Το θέμα της συγκεκριμένης εργασίας αποτελούν οι αλγόριθμοι συσσωμάτωσης, η σύγκριση τους και η ανάδειξη του σημαντικού ρόλου τους στην διαδικασία της Ομοσπονδιακής Μάθησης. Οι αλγόριθμοι αυτοί αποτελούν απαραίτητο και αναπόσπαστο κομμάτι της ΟΜ καθώς ορίζουν την συμπεριφορά του κεντρικού συστήματος και ενορχηστρώνουν την λειτουργία των τοπικών συστημάτων, ως προς τον τρόπο που θα γίνει η βελτίωση, η συλλογή και η ενσωμάτωση των παραμέτρων και των συντελεστών των τοπικών μοντέλων, που εκπαιδεύονται στα συστήματα που συμμετέχουν και χρησιμοποιούνται για την παραγωγή και την βελτίωση του κεντρικού μοντέλου. Οι αλγόριθμοι αυτοί ελέγχουν ολόκληρη την διαδικασία της εκπαίδευσης, αλλά και της επικοινωνίας, της λήψης και της συσσωμάτωσης των βαρών των μοντέλων αποτελώντας έτσι ένα από τα κυρίαρχα μέρη της Ομοσπονδιακής Μάθησης. Μέσω της εργασίας που επιτελούν, επιτρέπεται η παράλληλη εκπαίδευση των μοντέλων στα διάφορα συστήματα που συμμετέχουν στην διαδικασία και ουσιαστικά παρέχεται η ιδιωτικότητα των δεδομένων που τροφοδοτούνται σε κάθε τοπικό μοντέλο, το οποίο αποτελεί και το σημαντικότερο πλεονέκτημα της Ομοσπονδιακής Μάθησης. Ακόμα, επηρεάζουν άμεσα και σημαντικά τόσο την απόδοση αλλά και την επίδοση ολόκληρης της διαδικασίας, κάτι που γίνεται εμφανές μέσω της σύγκρισης τους.

Για να γίνει λόγος για αλγόριθμους συσσωμάτωσης, που ουσιαστικά αποτελούν τον αλγόριθμο βελτιστοποίησης του κεντρικού μοντέλου, με δεδομένα που προέρχονται από τα τοπικά θα πρέπει να γίνει αναφορά στον βασικό αλγόριθμο βελτιστοποίησης, τον Stochastic Gradient Descent (SGD).

Επιπλέον για να γίνει ανάλυση των αλγορίθμων θα πρέπει να οριστούν οι όροι:

w_t: *Βάρη του μοντέλου στον γύρο t*
$w_t{}^k$: *Βάρη του μοντέλου στον γύρο t στον client k*
K: *Συνολικός αριθμός clients που συμμετέχουν στην ΟΜ*
C: *Μέρος των clients που επιλέγονται σε κάθε γύρο*
η: *Βήμα μάθησης (learning rate)*
n: *Συνολικός αριθμός δεδομένων που συμμετέχει στην εκπαίδευση*
n_k: *Αριθμός τοπικών δεδομένων στον client k*
$f(w)$: *Συνάρτηση σφάλματος για* (x_t, y_t)
B: *Μέγεθος minibatch, τοπικής παρτίδας δεδομένων (batch)*
E: *Αριθμός Epochs*

5.1 Stochastic Gradient Descent

Ο αλγόριθμος Stochastic Gradient Descent (SGD) αποτελεί θεμελιώδες κομμάτι σε πολλές εφαρμογές υπολογιστικής μάθησης αφού αποτελεί έναν από τους πιο κοινούς αλγόριθμους βελτιστοποίησης μοντέλων κατά την διαδικασία της εκπαίδευσης τους. Σκοπός είναι η ελαχιστοποίηση της συνάρτησης του σφάλματος. Ξεκινώντας θα πρέπει να αναφερθεί ο αλγόριθμος Gradient Descent (κάθοδος βασισμένη στην κλίση) που αναπτύχθηκε από τον Cauchy το 1847 [22]. Σύμφωνα με τον αλγόριθμο αυτό σε κάθε επανάληψη τα βάρη του μοντέλου θα ενημερώνονται σύμφωνα με τον εξής τύπο:

$w_{t+1} = w_t - \eta \nabla_{\mathrm{w}} f(w_t)$,
[23], [24].

Ωστόσο, συνήθως, και ειδικότερα στην υπολογιστική μάθηση σκοπός δεν είναι να βελτιώσουμε την συμπεριφορά για μία μεταβλητή του μοντέλου παρά για ένα σύνολο μεταβλητών ή μία ομάδα του συνόλου των δεδομένων (batch). Επομένως σκοπός είναι η ελαχιστοποίηση μιας αντικειμενικής συνάρτησης F(w) της μορφής: $\frac{1}{n}\sum_{i=1}^{n} f_i(w)$. Άρα ο αλγόριθμος προσαρμόζεται και γίνεται: $w_{t+1} = w_t - \eta \nabla_w(\frac{1}{n}\sum_{i=1}^{n} f_i(w_t))$, [25].

Μία εξέλιξη του αλγόριθμου αυτού αποτελεί ο Stochastic Gradient Descent (SGD). Όπως γίνεται αντιληπτό και από το όνομα, ο αλγόριθμος αυτός εισάγει και το κριτήριο της τυχαιότητας. Σε αυτή την παραλλαγή του, ο αλγόριθμος δεν χρειάζεται να υπολογιστεί για κάθε δεδομένο ή για κάθε παρτίδα, αλλά επιλέγεται τυχαία ένα σημείο και υπολογίζεται το βήμα βελτιστοποίησης σύμφωνα με το τύπο του Gradient Descent ανά κάθε μία επανάληψη. Η παραλλαγή αυτή έχει επικρατήσει εξαιτίας του πολύ μειωμένου υπολογιστικού φόρτου που όπως γίνεται εύκολα αντιληπτό απαιτείται σε σχέση με τον κλασικό αλγόριθμο. Στην περίπτωση εκπαίδευσης ενός μοντέλου, με χρήση του SGD, το τυχαίο αυτό σημείο εκφράζεται μέσω της τυχαίας επιλογής μικρής παρτίδας δεδομένων (mini-batch), έτσι ώστε να γίνει βελτιστοποίηση του μοντέλου αυτού [26], [27], [28].

Όπως είναι γνωστό σκοπός ενός αλγόριθμου βελτιστοποίησης, είναι η εύρεση του ολικού μέγιστου ή ελάχιστου, για προβλήματα ελαχιστοποίησης όπως αυτό για το οποίο γίνεται λόγος, δηλαδή της εύρεσης του μικρότερου παράγοντα της συνάρτησης σφάλματος που υπάρχει στην εκπαίδευση μοντέλων υπολογιστικής μάθησης. Ένα πολύ συνηθισμένο πρόβλημα που αντιμετωπίζουν όλοι οι αλγόριθμοι και μέθοδοι βελτιστοποίησης είναι η «παγίδευση» τους σε τοπικά ελάχιστα ή μέγιστα χάνοντας έτσι το ολικό. Σύμφωνα με αυτά που έχουν ειπωθεί, αν ένας αλγόριθμος βρίσκεται κάποια χρονική στιγμή σε ένα τοπικό ακρότατο της συνάρτησης, είναι δύσκολο να αφήσει αυτό το σημείο αφού με την κλίση της συνάρτησης που χρησιμοποιεί φαίνεται ότι τα σημεία που βρίσκονται κοντά σε αυτό, (η απόσταση των σημείων είναι ανάλογη με τον βαθμό μάθησης), είναι χειρότερα από αυτό στο οποίο ήδη βρίσκεται.

Ένας παράγοντας που μπορεί να χρησιμοποιηθεί μαζί με τον SGD, κάτι ιδιαίτερα σύνηθες στην εκπαίδευση νευρωνικών δικτύων, είναι το momentum («ορμή»). Μέσω αυτού μπορούν να αντιμετωπιστούν σε αρκετές περιπτώσεις προβλήματα όπως το παραπάνω, ενισχύοντας έτσι την απόδοση και την ταχύτητα της διαδικασίας της εκπαίδευσης. Ωστόσο αυτό δεν είναι βέβαιο, με μελέτες να διαφοροποιούνται ως προς το πότε η χρήση το, και σε τι συνδυασμό των άλλων παραμέτρων, ευνοεί την εκπαίδευση, με κυρίαρχο ρόλο να έχουν το μέγεθος παρτίδας δεδομένων (batch size) και ο βαθμός μάθησης (learning rate) [29].

Αυτό που κάνει η παράμετρος του momentum είναι να εισάγει στην ανανέωση των συντελεστών όχι μόνο την κλίση της συνάρτησης σφάλματος εκείνη την στιγμή αλλά και παλαιότερες κλίσεις που έχουν υπολογιστεί σε προηγούμενους γύρους εκπαίδευσης. Η επιρροή της παλαιότερης κλίσης στον νέο συντελεστή ελέγχεται μέσω του συντελεστή ρ όπως φαίνεται και από τον αλγόριθμο του SGD με momentum στην Εικόνα 5.1.1,

Algorithm SGD and SGDM

1: **Input:** the loss function $\ell(w, z)$, the initial point $\boldsymbol{w}_1 \in \mathbb{R}^d$, the batch size b, learning rates $\{\eta_t\}_{t=1}^T$, $\boldsymbol{m}_0 = 0$, and momentum hyperparameters $\{\mu_t\}_{t=1}^T$.
2: **For** $t = 1 \to T$:
3: Sample a mini-batch of data B_t with size b
4: Calculate stochastic gradient $\nabla f_{B_t}(w_t) = \frac{1}{b}\sum_{z \in B_t} \ell(w_t, z)$
5: Update $\boldsymbol{m}_t \leftarrow \mu_t \boldsymbol{m}_{t-1} + \nabla f_{B_t}(\boldsymbol{w}_t)$
6: Update $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta_t \boldsymbol{m}_t$
7: **End For**

Εικόνα 5.1.1 Stochastic Gradient Descent (with Momentum) [29]

5.2 Federated Stochastic Gradient Descent (FedSGD)

Η εφαρμογή του αλγόριθμου βελτιστοποίησης SGD σε ρύθμιση Ομοσπονδιακής Μάθησης. Ο Federated Stochastic Gradient Descent - FedSGD, αποτελεί τον πρώτο αλγόριθμο συσσωμάτωσης με εφαρμογή στην Ομοσπονδιακή Μάθηση. Ο τρόπος που δουλεύει είναι σχετικά απλός. Αρχικά το στοχαστικό του μέρος αφορά την τυχαία επιλογή ενός μέρος των τοπικών συστημάτων clients C, αν πρόκειται για επιλογή με στοχαστικό κομμάτι αλλιώς μιλάμε για μη στοχαστική διαδικασία (non-stochastic gradient descent) όπου επιλέγονται όλοι οι clients που είναι διαθέσιμοι με το $C = 1$. Από πλευράς client, αυτό που γίνεται είναι ότι υπολογίζεται η μέση κλίση $gk = \nabla_w F_k(w_t) = \nabla_w(\frac{1}{n_k}\sum_{i=1}^{n_k} f_i(w_t))$ από όλα τα τοπικά δεδομένα n_k που έχει ο εξής client k για το τοπικό μοντέλο που του αντιστοιχεί. Από πλευράς κεντρικού συστήματος, θεωρείται ένα δεδομένο η: $learning\ rate$, αθροίζονται οι αυτές οι κλίσεις των clients, και ανανεώνεται το κεντρικό μοντέλο. Το κεντρικό μοντέλο ανανεώνει του συντελεστές του – βάρη με την διαφορά των υπαρχόντων συντελεστών, από το γινόμενο του βαθμού μάθησης με το συνολικό άθροισμα του γινομένου των εκάστοτε κλίσεων που υπολογίστηκαν σε client και του ποσοστού των τοπικών δεδομένων σε σχέση με τα συνολικά των clients που συμμετέχουν κάθε φορά: $w_{t+1} = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$. Έτσι όπως εύκολα γίνεται αντιληπτό ουσιαστικά κάθε client που συμμετέχει, εφαρμόζει μία φορά βελτιστοποίηση, μέσω gradient descent στο τοπικό μοντέλο του και ο server παράγει ένα σταθμισμένο μέσο όρο από τα τοπικά μοντέλα που παράγονται [1].

5.3 Federated Averaging (FedAvg)

Ο πλέον βασικότερος αλγόριθμος Ομοσπονδιακής Μάθησης. Αποτελεί, το σημείο αναφοράς για όλες τις έρευνες, τις συγκρίσεις, τις δοκιμές και είναι ο αλγόριθμος που στηρίζονται όλες οι προσπάθειες δημιουργίας νέων αλγορίθμων, που έχουν σκοπό να καλύψουν τις αδυναμίες του. Αρχικά προτάθηκε στην πρώτη δημοσίευση που εισήγαγε και ιδέα της Ομοσπονδιακής Μάθησης το 2016 από ερευνητές της Google [1], και αποτελεί ουσιαστικά εξέλιξη του FedSGD. Η μεγάλη δημοφιλία του έγκειται στο γεγονός πως πρόκειται για έναν αρκετά απλό αλγόριθμο, με σχετικά μικρές απαιτήσεις επικοινωνίας μεταξύ του server και των clients ενώ παράλληλα έχει ικανοποιητική απόδοση, σε συνάρτηση πάντα με την απλότητα του.

Ξεκινώντας, γίνεται αρχικοποίηση του γενικού μοντέλου από τον server. Επιλέγεται ένα ποσοστό από τους συνολικά διαθέσιμους clients και διανέμεται σε αυτούς ένα αντίγραφο του κεντρικού μοντέλου. Σε αυτούς τους clients, γίνεται τοπική εκπαίδευση. Εκπαιδεύεται δηλαδή το κάθε δίκτυο που είναι σε κάθε server, με τα δεδομένα που υπάρχουν στο τοπικό αυτό σύστημα. Στην συνέχεια, γίνεται συγκέντρωση των εκπαιδευμένων τοπικών μοντέλων και ενημερώνεται το κεντρικό μοντέλο στο server, με το μέσου μοντέλου από αυτά που συλλέχθηκαν.

Ωστόσο, αυτό που περιεγράφηκε μόλις είναι η ίδια γενική διαδικασία με τον FedSGD. Αυτό που κάνει τον FedAvg να διαφέρει είναι οι πολλαπλές βελτιστοποιήσεις του τοπικού μοντέλου πριν αυτό σταλεί στο server. Αναλυτικότερα, επιλέγεται ένα μέρος του συνολικού αριθμού των clients που θα συμμετέχουν σε κάθε γύρο στην διαδικασία της εκπαίδευσης, C. Αφού δοθεί ο σταθερός αυτός αριθμός, σε κάθε γύρο επιλέγονται τυχαία ποιοι θα είναι αυτοί οι $C * K$ clients που θα συμμετάσχουν και θα κάνουν εκπαίδευση του μοντέλου τους. Επιπλέον θα πρέπει να καταχωρηθούν ακόμα δύο σταθεροί αριθμοί. Αυτοί είναι ο αριθμός των επαναλήψεων της εκπαίδευσης σε όλα τα τοπικά δεδομένα, ο αριθμός δηλαδή των epochs και το μέγεθος της τοπικής παρτίδας δεδομένων, B. Έτσι το τοπικό μοντέλο σε κάθε επιλεγμένο client θα εκπαιδευθεί $E * \frac{n_k}{B}$ φορές, προτού αποσταλεί στο server για βελτίωση του κεντρικού μοντέλου που στην συνέχεια θα αποσταλεί πάλι σε $C * K$ clients, για να ακολουθήσει η ίδια διαδικασία μέχρι να συμπληρωθεί ο αριθμός των γύρων επικοινωνίας (communication rounds) που έχει ορισθεί [1].

Συγκεντρωτικά, όπως φαίνεται και στην εικόνα 7.3:

Από πλευράς server, γίνεται αρχικοποίηση του κεντρικού μοντέλου,
για κάθε γύρο επικοινωνίας:

- Επιλέγονται τυχαία $C * K$ clients

για κάθε επιλεγμένο client:

- Γίνεται ενημέρωση του τοπικού μοντέλου μέσω της εκπαίδευσης των τοπικών μοντέλων στους clients.
- Αθροίζονται όλα τα τοπικά δεδομένα από όλους τους clients, ώστε να βρεθεί ο συνολικός αριθμός των δεδομένων n.
- Ενημερώνονται τα βάρη του κεντρικού μοντέλου αθροίζοντας τον σταθμισμένο μέσο όρο των βαρών των τοπικά εκπαιδευμένων μοντέλων με βάση τα δεδομένα του εκάστοτε client ως προς τα συνολικά δεδομένα των όλων clients που συμμετέχουν στον εκάστοτε γύρο επικοινωνίας.

Σε κάθε ενημέρωση τοπικού μοντέλου σε κάθε client, χωρίζονται τα τοπικά δεδομένα σε παρτίδες σύμφωνα με το μέγεθος της παρτίδας B
Για κάθε epoch (μέχρι E):

Για κάθε τοπική παρτίδα δεδομένων:

- Τα βάρη ενημερώνονται μέσω του αλγόριθμου βελτιστοποίησης SGD, όπως αναλύθηκε στο κεφάλαιο 5.1. $[w_{t+1} = w_t - \eta \nabla_w f(w_t)]$

Επιστρέφονται τα νέα βάρη του μοντέλου στο Server.

Algorithm `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **in parallel do**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$m_t \leftarrow \sum_{k \in S_t} n_k$
$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in \mathcal{B}$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Εικόνα 5.3.1 FederatedAveraging Algorithm [1]

5.4 Federated Algorithm with Proximal Term (FedProx)

Ο FedProx αποτελεί έναν αλγόριθμο, που έχει σκοπό να αντιμετωπίσει την ετερογένεια που υπάρχει σε περιβάλλοντα υπολογιστικής μάθησης. Ο FedProx καλείται να διαχειριστεί την ανομοιογένεια των δεδομένων. Συγκεκριμένα οι ιδιότητες του αξιοποιούνται κυρίως όταν θα πρέπει να γίνει διαχείριση και μάθηση σε non-IID δεδομένα. Ωστόσο, το κυρίαρχο χαρακτηριστικό του που τον κάνει να διαφέρει, είναι ότι γίνεται προσπάθεια διαχείρισης και της διαφορετικότητας των ίδιων των συστημάτων που συμμετέχουν στην συνολική διαδικασία. Τα συστήματα αυτά μπορεί να διαφέρουν τόσο για τις υπολογιστικές τους δυνατότητες όσο και για τις δυνατότητες επικοινωνίας τους με το κεντρικό σύστημα - server. Μεγαλύτερη εφαρμογή για τέτοιος αλγόριθμους, υπάρχει σε μεγαλύτερη ανομοιογένεια, κυρίαρχα δηλαδή σε σενάρια cross-device, ωστόσο όχι μόνο δεν μπορεί να αποκλειστεί αυτό το φαινόμενο σε cross-silo σενάρια, αλλά είναι και αρκετά σύνηθες, με μικρότερες ωστόσο διαφορές μεταξύ των συστημάτων και αρκετά σπάνιων διακοπών και προβλημάτων επικοινωνίας με τον server.

Όπως, αναφέρθηκε ο βασικός στόχος είναι η αντιμετώπιση της ανομοιογένειας των δεδομένων και των ίδιων των συστημάτων. Έτσι ο FedProx εισάγει την έννοια του Proximal term («εγγύς όρος»): $\frac{\mu}{2}\|w - w_t\|^2$. Όπου $\|w - w_t\|^2$ η 2-νόρμα διανύσματος που υπολογίζει την Ευκλείδεια απόσταση μεταξύ δύο διανυσμάτων, που αναπαρίστανται από τους συντελεστές του τοπικού μοντέλου με το μοντέλο που στάλθηκε από το server για τον γύρο t. Ακόμα ο συντελεστής μ είναι ένας δεδομένος όρος και υπάρχει για να ελέγχει την ισχύ του Proximal Term. Ο όρος αυτός συμβάλει στην καταπολέμηση της ανομοιογένειας περιορίζοντας τις τοπικές ανανεώσεις ώστε να είναι κοντά στο κεντρικό μοντέλο χωρίς να χρειάζεται να γίνει ειδική εξατομίκευση του κάθε client ως προς το πόσο μπορεί να βελτιώσει το μοντέλο του για να μην «ξεφύγει» από το κεντρικό (π.χ. με όριο των epochs) και ενσωματώνει ομαλά τα διαφορετικά δεδομένα των τοπικών μοντέλων [30]. Έτσι διαφοροποιείται η συνάρτηση σφάλματος σε κάθε client. Πλέον, οι clients προσπαθούν να ελαχιστοποιήσουν την συνολική συνάρτηση αθροίσματος: $f(w) + \frac{\mu}{2}\|w - w_t\|^2$, όπου $f(w)$ η κλασσική συνάρτηση σφάλματος του τοπικού μοντέλου, όπως χρησιμοποιείται συνήθως, μαζί πλέον με τον proximal term που αναφέρθηκε. Ακόμα πρέπει να αναφερθεί η ύπαρξη άλλη μίας μεταβλητής της γ, η οποία ελέγχει το μέγεθος της εργασίας του κάθε client. Η μεταβλητή αυτή έχει σημαντικό ρόλο στην λειτουργία του FedProx, όταν συμμετέχουν συστήματα διαφορετικής υπολογιστικής ικανότητας, επιτρέποντας μεγαλύτερη ευελιξία στη διαδικασία της εκπαίδευσης στα συστήματα αυτά και θα μπορούσε να παραλληλιστεί εν μέρη με το learning rate για κάθε σύστημα.

Ο FedProx θα μπορούσε να χαρακτηριστεί ως μία ειδική περίπτωση του FedAvg. Αν αναλογιστεί κάποιος πως αν η μεταβλητή μ γίνει ίση με το 0 και άρα ο proximal term μηδενιστεί και ο συντελεστής γ διατηρηθεί σταθερός, αντίστοιχα με το learning rate, σε όλους τους clients και τους γύρους της εκπαίδευσης, δηλαδή δεν ληφθεί καθόλου υπόψη η ανομοιογένεια των συστημάτων τότε ουσιαστικά πρόκειται για τον FedAvg [30].

Επιπλέον, θα πρέπει να αναφερθεί η ευελιξία που προσφέρει ο FedProx στην εκπαίδευση του τοπικού μοντέλου στην αυτούσια μορφή του, χωρίς δηλαδή ιδιαίτερων εξατομικεύσεων. Εξαιτίας της διαδικασίας που ακολουθείται, με σκοπό την ελαχιστοποίηση της συνολικής συνάρτησης, που αναφέρθηκε, επιτρέπει την βελτιστοποίηση του τοπικού μοντέλου, δηλαδή την ελαχιστοποίηση της $f(w)$, για κάθε client, με οποιονδήποτε αλγόριθμο βελτιστοποίησης προτιμάται από τον ίδιο τον client. Έτσι παρέχεται η δυνατότητα χρησιμοποίησης άλλου αλγόριθμου και όχι μόνο μέσω του SGD, που χρησιμοποιείται από τον FedAvg και άρα αποτελεί την επιλογή για κάθε συγκεντρωτικό αλγόριθμο Ομοσπονδιακής Μάθησης που βασίζεται πάνω σε αυτόν.

Algorithm FedProx

Input : $K, T, \mu, \gamma, w_0, N, p_k, k = 1, \ldots, N$
for $t = 1, 2, \ldots, T$ **do**
    Server selects a subset S of K clients at random (each client k is chosen with probability p_k)
    Server sends w_t to all chosen clients
    Each chosen client $k \in S$ finds a w_{t+1}^k which is a γ_t^k-inexact minimizer of: $w_{t+1}^k \approx \text{argmin}_w \, h_k(w; w_t) = f(w) + \frac{\mu}{2}\|w - w_t\|^2$
    Each client $k \in S$ sends w_{t+1}^k back to the server
    Server aggregates the w's as $w_{t+1} = \frac{1}{K}\sum_{k \in S_t} w_{t+1}^k$
end for

Εικόνα 5.4.1 FedProx Algorithm

5.5 Federated Normalized Averaging (FedNova)

Άλλος ένας αλγόριθμος συσσωμάτωσης για συστήματα Ομοσπονδιακής Μάθησης είναι ο Federated Normalized Averaging (FedNova). Αιτία της δημιουργίας του FedNova, όπως και άλλων αλγορίθμων αποτέλεσε και πάλι η ανομοιογένεια των συστημάτων των clients και η τοπική σύγκλιση που παρουσιάζεται στους clients από τα διαφορετικά τοπικά σύνολα δεδομένων και τις διαφορετικές ικανότητες τους. Δηλαδή, κάθε τοπική εκπαίδευση στους διάφορους clients, προσπαθώντας να βελτιώσει το μοντέλο, το κάνει όσο καλύτερο μπορεί, το κάθε ένα από αυτά σε τοπικά καλύτερες λύσεις για τα δεδομένα που έχει. Ωστόσο, με την ετερογένεια που μπορεί να υπάρχει στα δεδομένα (non-IID) και τις ικανότητες των συστημάτων, μπορεί μεν να παρουσιάζεται καλύτερο το τοπικό μοντέλο στα πλαίσια της εκπαίδευσης (training) και ίσως και της τοπικής επαλήθευσης (testing), αλλά αυτό δρα αρνητικά στο κεντρικό μοντέλο υποβαθμίζοντας την απόδοση του όταν αυτή θα πρέπει να επαληθευθεί σε ένα φάσμα δεδομένων όλων των συστημάτων που συμμετέχουν στην ΟΜ [31], [32].

Έτσι για να δοθεί λύση στο κύριο αυτό πρόβλημα της ΟΜ προτάθηκε ο FedNova [33]. Ο συγκεκριμένος αλγόριθμος προκειμένου να αντιμετωπίσει τα προβλήματα που αναφέρθηκαν, εισάγει την κανονικοποίηση σε κάθε ανανέωση του κεντρικού μοντέλου. Δηλαδή, αυτός ο αλγόριθμος σε αντίθεση με τον FedAvg ανανεώνει το κεντρικό μοντέλο με τον κανονικοποιημένο μέσο όρο των κλίσεων των τοπικών μοντέλων. Η κανονικοποίηση αυτή γίνεται ως προς τον αριθμό των τοπικών ανανεώσεων των κλίσεων αυτών.

Ξεκινώντας, γίνεται ξανά αρχικοποίηση του κεντρικού για το μοντέλο του server. Στην συνέχεια για κάθε γύρο επικοινωνίας επιλέγεται ξανά ένα μέρος των διαθέσιμων clients για να συμμετάσχει στον αντίστοιχο γύρο. Σε κάθε client υπολογίζονται οι κλίσεις του μοντέλου για τα τοπικά δεδομένα σε αντίστοιχο αριθμό Epochs και μεγέθους τοπικών παρτίδων όπως και στον FedAvg, αλλά πλέον στην συσσωμάτωση για την δημιουργία και την ανανέωση κάθε φορά του κεντρικού μοντέλου λαμβάνεται ο μέσος όρος των κανονικοποιημένων κλίσεων των μοντέλων. Συγκεκριμένα για κάθε ανανέωση του τοπικού μοντέλου, λαμβάνονται οι κλίσεις που παρέχονται από τον αλγόριθμο τοπικής βελτιστοποίησης. Οι κλίσεις αυτές πλέον διαιρούνται με την πρώτης τάξεως νόρμα ενός μη αρνητικού διανύσματος a το οποίο ορίζεται μέσω της ποσότητας των συνολικών κλίσεων που λήφθηκαν, δηλαδή το πλήθος των ανανεώσεων που έγιναν στο τοπικό μοντέλο. Αυτό το πλήθος ορίζεται μέσω μίας μεταβλητής τ, η οποία είναι αριθμεί τις ανανεώσεις αυτές.

Αναλυτικότερα, το κεντρικό μοντέλο ανανεώνεται σύμφωνα με την σχέση:

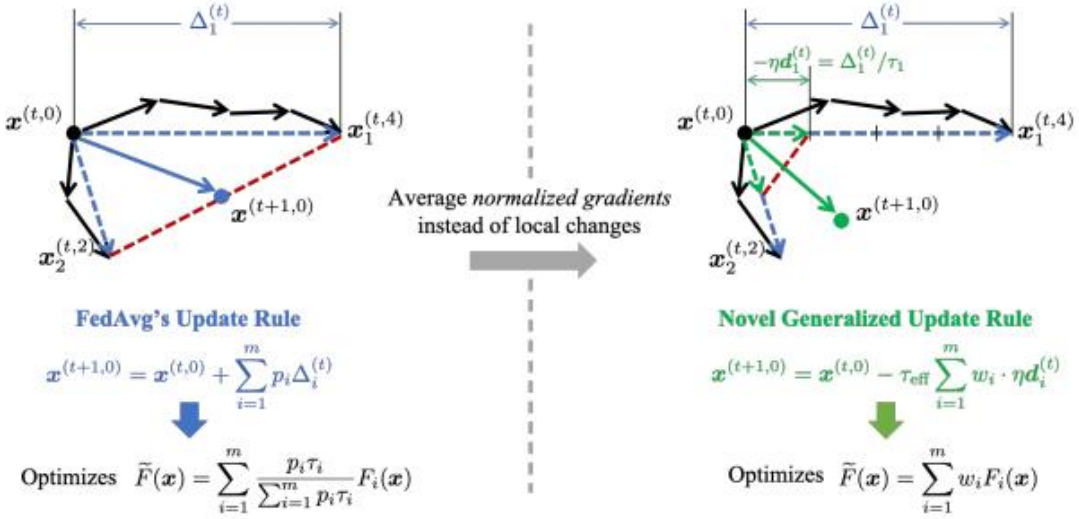
$$x^{t+1} - x^t = -\tau_{eff} \sum_{k=1}^{k} w_k * \eta d_k^t$$

Όπου: x: οι συντελεστές του κεντρικού μοντέλου στη χρονική στιγμή t και,

τ_{eff}: μεταβλητή που αριθμεί τα βήματα ανανέωσης του τοπικού μοντέλου, και μπορεί να διαμορφωθεί η τιμή της από το server, αν χρειάζεται.

w_k: τα βάρη του τοπικού μοντέλου όπως αυτά είναι πριν την ανανέωση τους μέσω του τοπικού αλγόριθμου βελτιστοποίησης του μοντέλου και πολλαπλασιαστούν δηλαδή με τις κλίσεις που υπολογίστηκαν.

d_k^t: οι κανονικοποιημένες κλίσεις, δηλαδή οι κλίσεις των παραμέτρων των μοντέλων όπως υπολογίζονται πάντα διαιρεμένες με το διάνυσμα a.



Εικόνα 5.5.1 Σύγκριση της ανανέωσης των παραμέτρων των τοπικών μοντέλων (μαύρα βέλη) και του κεντρικού μοντέλου (μπλε/πράσινο) βέλος ανάμεσα σε FedAvg και FedNova [31]

Επιπλέον ισχύουν:

$$\tau_{eff} = -\sum_{k=1}^{k} p_k * \|a_i\|, \quad w_k = -\frac{p_k * \|a_k\|}{\sum_{k=1}^{k} p_k * \|a_k\|}, \quad d_k^t = \frac{G_k^t * a_k}{\|a_k\|}$$

Και

$p_k = n_k/n$,

G_k^t: διάνυσμα που περιέχει τις κλίσεις των παραμέτρων την στιγμή t.

Ένα από τα κυρίαρχα χαρακτηριστικά του FedNova είναι η ευελιξία που προσφέρει στην επιλογή τοπικού αλγόριθμου ανανέωσης των μοντέλων. Ακόμα και στην αρχική δημοσίευση του γίνεται λόγος για ελευθερία επιλογής και πιο ειδικά αναφέρονται οι παραμετροποιήσεις που μπορεί να δεχθεί για τον SGD, τον SGD με momentum, κ.ά., αλλά ιδιαίτερη μνεία αξίζει στο ότι μπορεί να χρησιμοποιηθεί και παράλληλα με την χρήση στον SGD του proximal term που εισήγαγε ο αλγόριθμος FedProx.

Ανάλογα με την επιλογή αυτή αλλάζει και ο υπολογισμός του διανύσματος a [31]. Ενδεικτικά, και όπως χρησιμοποιήθηκε στην προσγείωση που υλοποιήθηκε αν γίνεται χρήση του SGD με momentum τότε το a υπολογίζεται:

$$\|a_k\| = \left[\tau - \rho * \frac{1-\rho^{\tau}}{1-\rho}\right] / (1-\rho)$$

Με ρ τον συντελεστή του momentum, 5.1, και όταν γίνεται χρήση απλού SGD χωρίς momentum γίνεται: $\|a_k\| = \tau$

ΚΕΦΑΛΑΙΟ 6 ° : Απόδοση και Σύγκριση των Αλγορίθμων Συσσωμάτωσης

Όπως αναφέρεται και στον τίτλο ακόμα, βασικός σκοπός της παρούσας διπλωματικής εργασίας είναι η σύγκριση των αλγορίθμων συσσωμάτωσης. Οι αλγόριθμοι, μεταξύ των οποίων έγινε σύγκριση είναι αυτοί που αναφέρθηκαν στο προηγούμενο κεφάλαιο. Η σύγκριση τους έγινε με τα αποτελέσματα και τους βαθμούς επίδοσης του κεντρικού μοντέλου που δημιουργείται μέσω αυτών και σε όλα τα δεδομένα για testing του κάθε dataset. Όπως είναι γνωστό οι διαφορετικές διατάξεις του συνολικού περιβάλλοντος που προσομοιώνονται είναι σε σενάρια cross-silo. Η σύγκριση μεταξύ τους, αφορά προφανώς τέτοια σενάρια. Συνεπώς, στις συγκρίσεις οι πειραματικές εφαρμογές αποτελούνται από χαμηλό αριθμό συστημάτων που συμμετέχουν στην διαδικασία της εκπαίδευσης. Ο αριθμός αυτός συγκεκριμένα είναι 15 τοπικά συστήματα. Επιπλέον, επειδή σε αυτές τις περιπτώσεις τα συστήματα που χρησιμοποιούνται έχουν δεδομένες συνθήκες συνδεσιμότητας και επικοινωνίας, και γνωστές υπολογιστικές ικανότητες, οι οποίες συνήθως είναι ικανοποιητικές προς υψηλές, χρησιμοποιήθηκαν τα τοπικά συστήματα όπως ορίζεται από τον αριθμό αυτών που επιλέχθηκαν σε κάθε επανάληψη της πειραματικής εφαρμογής, χωρίς πιθανότητα αδυναμίας και εγκατάλειψης από την συνολική διαδικασία κάποιου συστήματος.

Η ιδιωτικότητα των δεδομένων του εκάστοτε τοπικού συστήματος, που είναι το κυρίαρχο ζήτημα της Ομοσπονδιακής Μάθησης, προστατεύεται αφού δεν υπάρχει μεταφορά των δεδομένων καθαυτών. Επιπλέον, προστατεύονται τα δεδομένα που αφορούν όχι την εκπαίδευση αλλά και τους παραμέτρους των μοντέλων μέσω τεχνικών που αναφέρθηκαν στο κεφάλαιο 2.4 (differential privacy κ.λπ.). Ωστόσο, στις συγκρίσεις που πραγματοποιήθηκαν στα πλαίσια της συγκεκριμένης μελέτης δεν έγινε εφαρμογή αυτών των τεχνικών με προσθήκη θορύβου στα δεδομένα που μεταφέρονται μεταξύ client και server, τους συντελεστές δηλαδή και τις βοηθητικές μεταβλητές των μοντέλων και έτσι μεταφέρονται αυτούσια.

Οι συγκρίσεις που γίνονται δείχνουν τις διαφορές στην απόδοση των κεντρικών μοντέλων που παράγονται ως αποτέλεσμα Ομοσπονδιακής Μάθησης με χρήση των διαφορετικών αλγορίθμων που αναφέρθηκαν, και όχι με τις αντίστοιχες αποδόσεις των μοντέλων που παράγονται με την κλασσική μέθοδο υπολογιστικής μάθησης, κεντρικοποιημένα δηλαδή με την εκπαίδευση να γίνεται σε ένα σύστημα (Πίνακας 4.2-1 και Πίνακας 4.3-1). Η σύγκριση αυτή μεταξύ των δύο αυτών μορφών μάθησης ως προς την απόδοση τους δεν έχει λόγο να πραγματοποιηθεί, αφού όπως γίνεται εύκολα αντιληπτό, όχι μόνο είναι πολύ δύσκολο να παρουσιαστούν με Ομοσπονδιακή Μάθηση, τα επίπεδα απόδοσης της κεντρικοποιημένης υπολογιστικής μάθησης αλλά πολλές φορές ακόμα και να πλησιάσουμε αυτή την απόδοση αποτελεί πολύ μεγάλη πρόκληση. Αντίθετα οι συγκρίσεις που πραγματοποιούνται μεταξύ των αλγορίθμων γίνονται σε διαφορετικά κάθε φορά σενάρια, τα οποία διαφοροποιούνται, μέσω αλλαγών των παραμέτρων τους. Οι παράμετροι αυτοί αποτελούνται από τους συνολικά διαθέσιμους clients όπου μοιράζονται τα συνολικά δεδομένα, τον αριθμό αυτών που τελικά επιλέγονται σε κάθε γύρο, τον αριθμό των τοπικών επαναλήψεων εκπαίδευσης των μοντέλων epochs, το μέγεθος των τοπικών πακέτων δεδομένων – batch size, τον αριθμό των γύρων επικοινωνίας και το μοντέλο που χρησιμοποιείται. Επίσης, σημαντικό ρόλο όπως έχει δειχθεί παίζουν και τα δεδομένα που χρησιμοποιούνται και η κατανομή τους στα τοπικά συστήματα. Αν πρόκειται για IID/non-IID και τι είδους διαφοροποιήσεις υπάρχουν σε αυτά., Εξίσου σπουδαίο ρόλο έχουν και τα ίδια τα συστήματα. Παρότι οι συγκρίσεις αφορούν cross-silo σενάρια, μπορεί να υπάρχουν διαφορές στις ικανότητες των συστημάτων. Οι διαφορές αυτές σε καμία περίπτωση δεν πλησιάζουν τις διαφορές που μπορεί να υπάρχουν σε cross-device περιπτώσεις, αλλά η ύπαρξη των «μικρών» αυτών διαφορών για cross-silo είναι υπαρκτές και αρκετά συνηθισμένες, όπως ήδη αναφέρθηκε και πιο πριν. Ωστόσο,

εξαιτίας του μικρού μεγέθους αυτών των διαφορών στα cross-silo σενάρια και καθώς σκοπός είναι αρχικά μια πιο γενική μελέτη των διαφορών απόδοσης των αλγορίθμων, δεν έγινε προσομοίωση των διαφορών αυτών των συστημάτων αφού θα περιέπλεκε αρκετά την διαδικασία.

Οι μετρήσεις μπορούν να γίνουν με αλλαγές των παραμέτρων:

- Μοντέλου – [Simple CNN/ ResNet18]
- Δεδομένων – [MNIST/ CIFAR10/ CIFAR100]
- Κατανομής των Δεδομένων – [IID Ομοιογενής/ IID Διαφορετική Ποσότητα ανά Client/ Non-IID]
- Αριθμού Γύρων Επικοινωνίας
- Αριθμού Συνολικών Clients
- Ποσοστού Επιλεγμένων Clients σε κάθε γύρο
- Αριθμού Επανάληψης Εκπαίδευσης - Epochs
- Μεγέθους Παρτίδας - Batch size Δεδομένων Εκπαίδευσης

Αρχικά υπάρχει η επιλογή του μοντέλου, μεταξύ των δύο διαθέσιμων επιλογών, και των δεδομένων, με τα διαθέσιμα datasets, προκειμένου να υπάρχει μια καλή εικόνα της Ομοσπονδιακής Μάθησης, σε διαφορετικές συνθήκες. Στην συνέχεια τους συντελεστές που αλλάζουν την δομή του ευρύτερου συστήματος της ΟΜ και βοηθούν στην παρατήρηση τόσο της συνολικής, αλλά και τοπικής ανά διαστήματα, ανάλυσης της απόδοσης και της συμπεριφοράς των αλγορίθμων. Αρχικά την κατανομή των δεδομένων. Από τους πιο βασικούς παράγοντες διαφοροποίησης και δυσκολίας της ΟΜ γενικότερα. Αποτελεί πολύ σημαντικό παράγοντα στην σύγκριση των αλγορίθμων, αφού όπως αναφέρθηκε και στην ανάλυση τους, αποτελεί κυρίαρχο πρόβλημα και παίζει σημαντικό ρόλο στην ανάπτυξη τους με πολλούς από αυτούς να στοχεύουν στην αντιμετώπιση της ανομοιογένειας που προκαλείται από την κατανομή που εφαρμόζεται. Όμως σημαντική είναι και η επίδοση των αλγορίθμων σε ομοιογενή δεδομένα, ειδικότερα όταν πρόκειται για σενάρια cross-silo, όπου η πιθανότητα είναι σημαντική και για τις δύο περιπτώσεις δεδομένων. Στην συνέχεια, η παράμετρος των γύρων επικοινωνίας που θα πραγματοποιηθούν έχει αυξημένο ρόλο αφού, τόσες φορές θα ανανεωθεί το γενικό μοντέλο, και δίνει ευκαιρία συμμετοχής σε clients που ίσως δεν έχουν συμμετάσχει ήδη, αν πρόκειται για περιπτώσεις που δεν επιλέγεται σε κάθε γύρο το σύνολο τους που είναι και το πιο σύνηθες. Οι αριθμοί των συνολικών clients και του δείγματος τους που συμμετέχει σε κάθε γύρο, προφανώς συμβάλουν ιδιαίτερα, αφού μεταβάλουν τον όγκο των δεδομένων προσφέροντας επιπλέον στοιχεία που μπορεί να είναι αρκετά σημαντικά, συνολικά ή ανά γύρο δυσκολεύοντας ή μη όμως την διαδικασία. Ο αριθμός των επαναλήψεων εκπαίδευσης – epochs, που γίνονται είναι μια σημαντική τροποποίηση με τα τοπικά συστήματα στα cross-silo σενάρια να είναι ικανά τελέσουν έναν ικανοποιητικό αριθμό, ο οποίος θα πρέπει να συγκρατηθεί, ώστε να μην υπάρξουν αρνητικές επιπτώσεις της αυξημένης εκπαίδευσης των μοντέλων όπως ισχύει γενικά στην εκπαίδευση μοντέλων και νευρωνικών δικτύων (φαινόμενο overfitting κλπ.). Τέλος, το μέγεθος παρτίδας – batch size των δεδομένων που

χρησιμοποιούνται κατά την διάρκεια κάθε epoch τοπικής εκπαίδευσης φαίνεται μέσα από μελέτες [34], να επηρεάζει την απόδοση των αλγορίθμων.

Όπως φαίνεται και από τις επιλογές των αλλαγών που φαίνονται στην παραπάνω λίστα, αν δοκιμαστούν όλες θα υπάρξει ένα πολύ μεγάλο πλήθος συγκρίσεων. Αυτό είναι πολύ θετικό προκειμένου να ληφθούν ασφαλή συμπεράσματα. Παρότι όμως οι συγκρίσεις ιδανικά θα γινόντουσαν με αυτές τις επιλογές, στην παρούσα εργασία δεν πραγματοποιούνται όλες οι αλλαγές. Πραγματοποιούνται συγκεκριμένες αλλαγές με παραδοχή κάποιων σταθερών μεταβλητών που αναφέρονται στην συνέχεια, διατηρώντας όμως τα σενάρια που υλοποιούνται αντιπροσωπευτικά για την καταλληλόλητα εφαρμογής του κάθε αλγόριθμου.

Δεδομένων αυτών, οι προσομοιώσεις που θα πραγματοποιηθούν για τους αλγόριθμους FedAvg, FedProx και FedNova θα έχουν κοινές και σταθερές τις εξής παραμέτρους:

Αριθμού Συνολικών Clients [15]

Ποσοστού Επιλεγμένων Clients σε κάθε γύρο [0.7]

Αριθμού Επανάληψης Εκπαίδευσης - Epochs [10]

Μεγέθους Παρτίδας - Batch size Δεδομένων Εκπαίδευσης [32]

Η εξαγωγή συμπερασμάτων γίνεται δοκιμάζοντας και λαμβάνοντας υπόψη όλα τα παραπάνω.

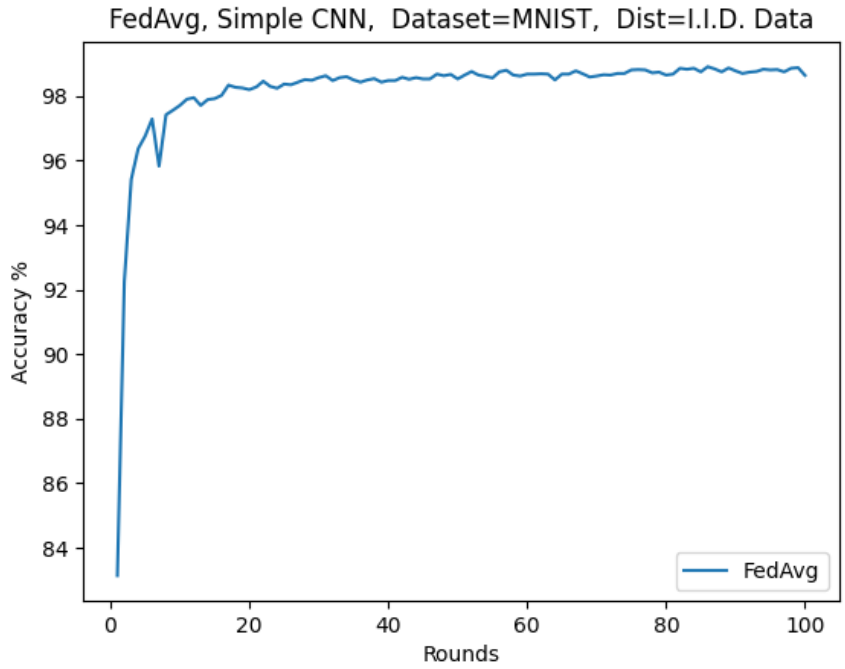
Αποτυπώνονται μέσω διαγραμμάτων και πινάκων η απόδοση του κάθε αλγόριθμου. Στα διαγράμματα αυτά, φαίνεται η απόδοση του γενικού μοντέλου, σε σχέση με τους γύρους επικοινωνίας του server με τους clients. Αρχικά όπως ειπώθηκε και νωρίτερα, σαν μέτρο απόδοσης, μπορεί να ληφθεί η απόδοση του μοντέλου μέσω κλασσικής υπολογιστικής μάθησης.

Όπως φαίνεται στη συνέχεια, και για τους τρεις αλγόριθμους που υλοποιήθηκαν για το Dataset CIFAR100, δεν μπορούν να ληφθούν ασφαλή αποτελέσματα λόγω της μικρής απόδοσης του απλού μοντέλου. Στην ουσία έχουμε ένα αρκετά χαμηλό ανώτατο threshold, κάτι το οποίο δεν δίνει ενδεικτικές τιμές ούτε για την μέση απόδοση αλλά ούτε και για την διακύμανση των επιδόσεων. Γι’ αυτό το λόγο χρησιμοποιείται και το δεύτερο μοντέλο (ResNet 18), για να ληφθούν αποτελέσματα για τους αλγόριθμους, για το Dataset CIFAR100.

6.1 FedAvg – Αποτελέσματα

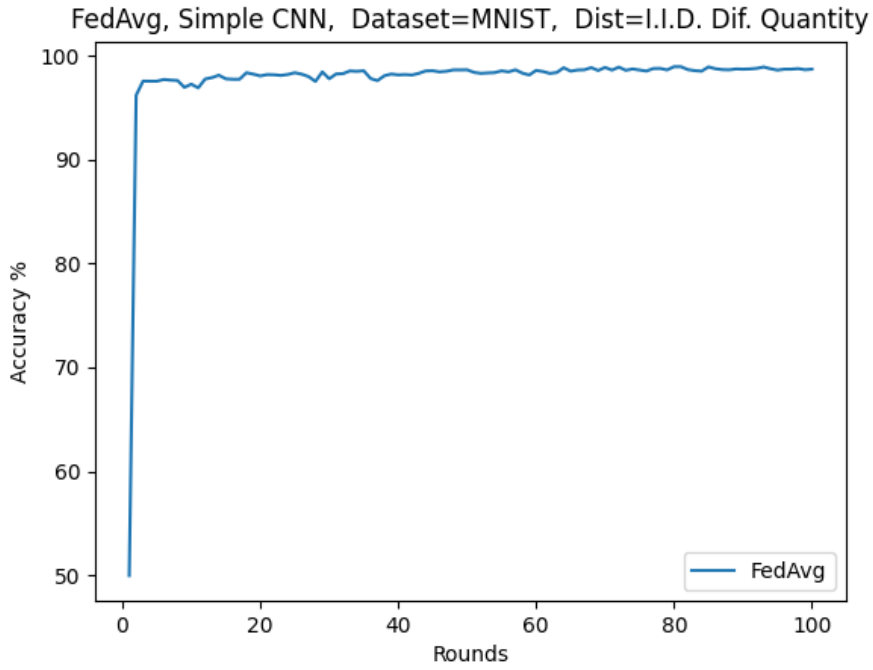
MNIST – Simple CNN (Μέγιστη Επίδοση 98.97):

Για IID κατανομή:



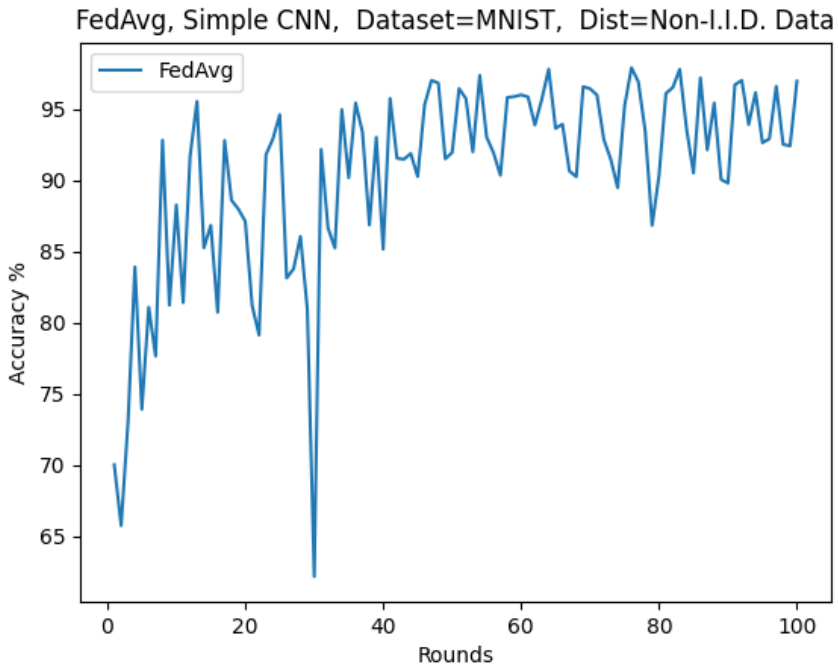
Διάγραμμα 6.1-1 FedAvg, MNIST, SimpleCNN, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.1-2 FedAvg, MNIST, SimpleCNN, IID Dif. Quantity Data

Για Non-IID κατανομή:



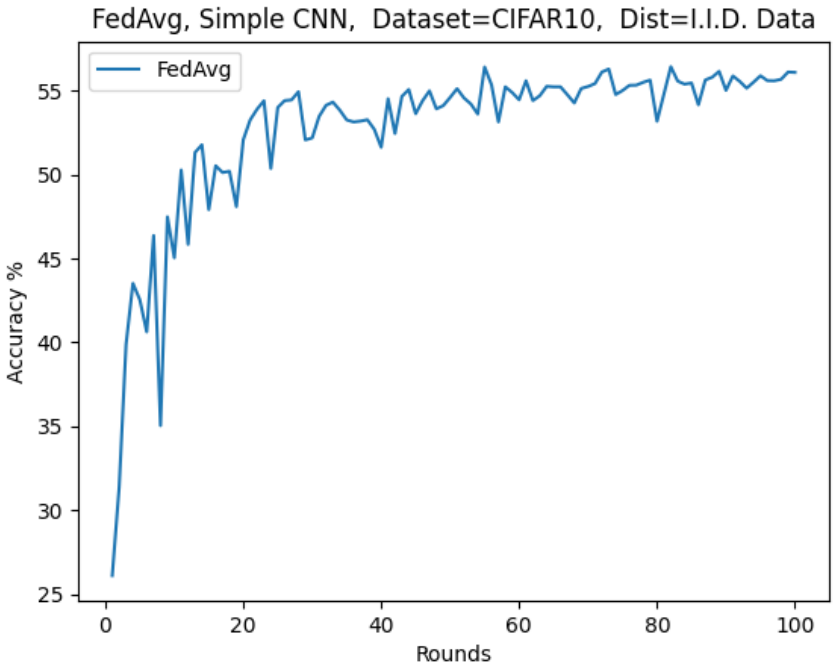
Διάγραμμα 6.1-3 FedAvg, MNIST, SimpleCNN, Non-IID Data

Πίνακας 6.1-1 FedAvg, MNIST, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	98.91	98.95	97.91
Μέση (Μετά από 15 epochs)	98.61	98.46	91.95
Διακύμανση (Μετά από 15 epochs)	0.03	0.098	29.84

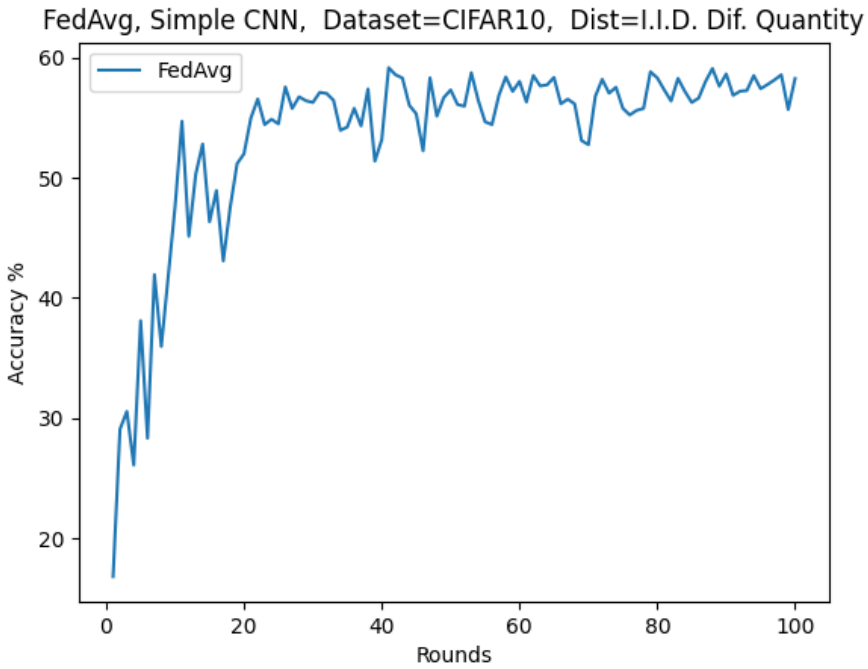
CIFAR10 – Simple CNN (Μέγιστη Επίδοση 64.2):

Για IID κατανομή:



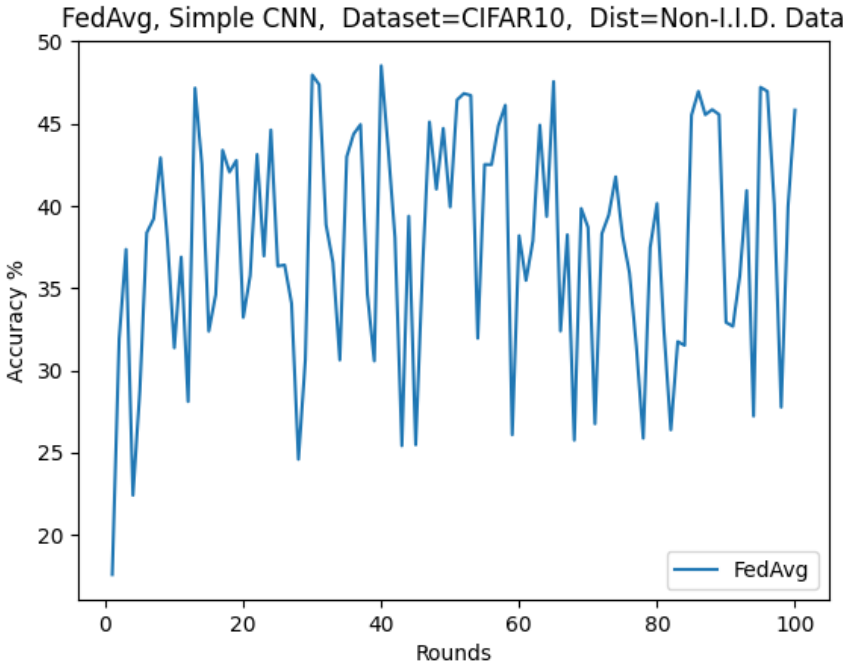
Διάγραμμα 6.1-4 FedAvg, CIFAR10, SimpleCNN, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.1-5 FedAvg, CIFAR10, SimpleCNN, IID Dif. Quantity Data

Για Non-IID κατανομή:



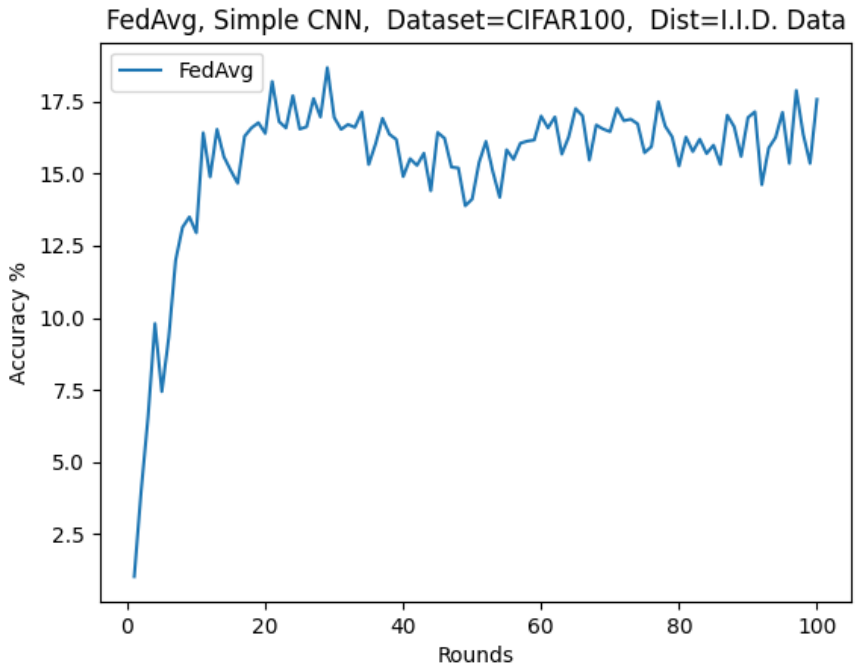
Διάγραμμα 6.1-6 FedAvg, CIFAR10, SimpleCNN, Non-IID Data

Πίνακας 6.1-2 FedAvg, CIFAR10, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	56.42	59.14	48.53
Μέση (Μετά από 15 epochs)	54.35	56.08	38.37
Διακύμανση (Μετά από 15 epochs)	2.47	6.81	44.46

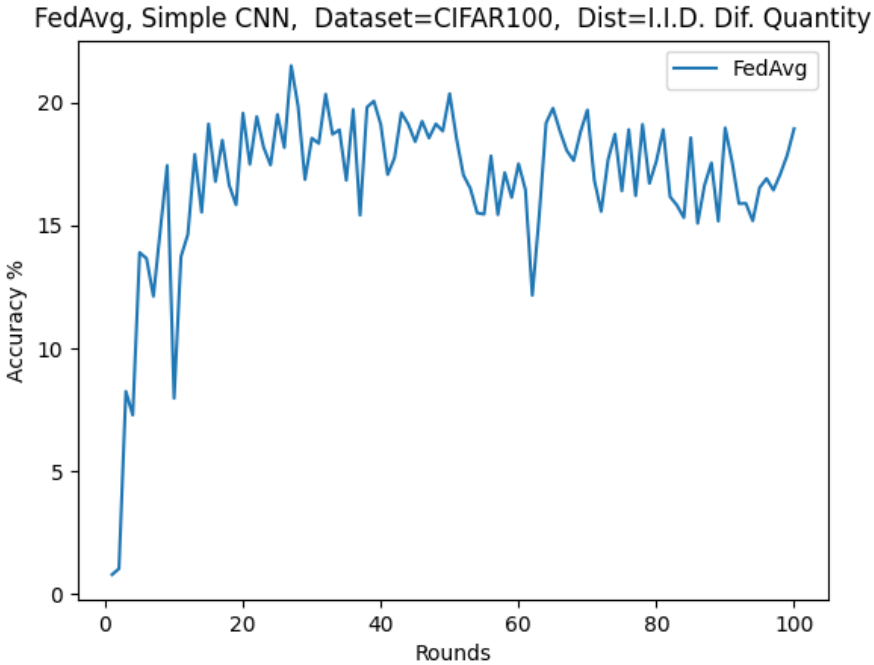
CIFAR100 – Simple CNN (Μέγιστη Επίδοση 29.61):

Για IID κατανομή:



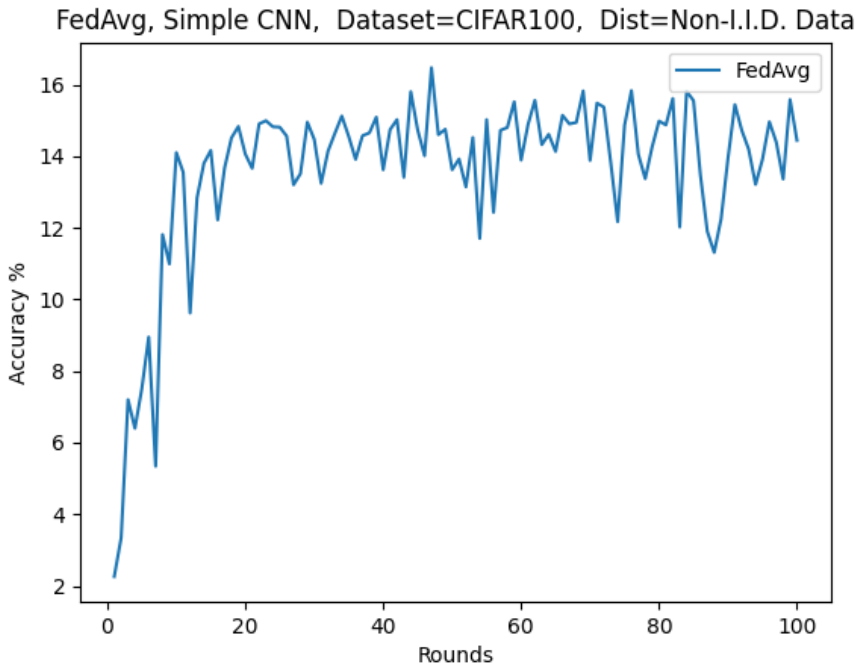
Διάγραμμα 6.1-7 FedAvg, CIFAR100, SimpleCNN, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.1-8 FedAvg, CIFAR100, SimpleCNN, IID Dif. Quantity Data

Για Non-IID κατανομή:



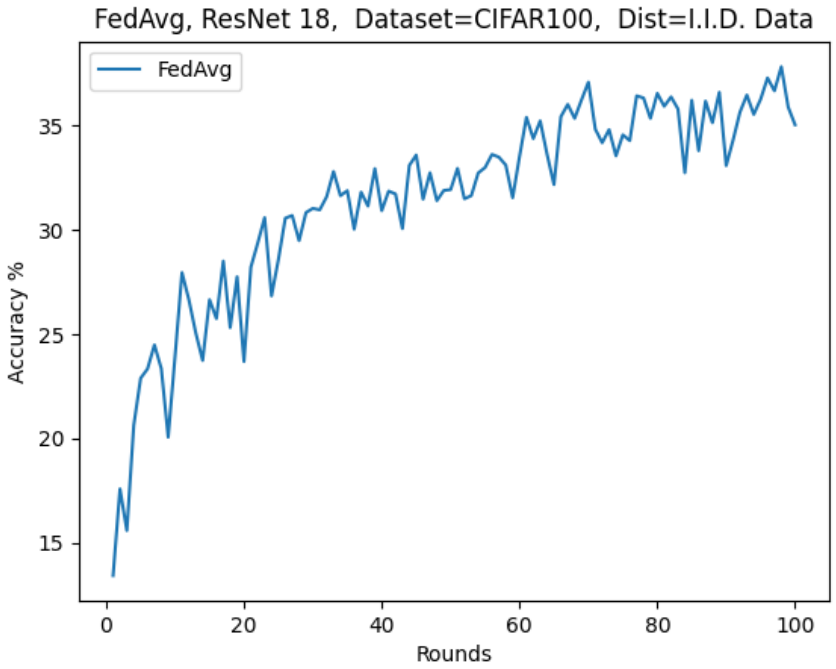
Διάγραμμα 6.1-9 FedAvg, CIFAR100, SimpleCNN, Non-IID Data

Πίνακας 6.1-3 FedAvg, CIFAR100, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	18.68	21.50	16.47
Μέση (Μετά από 15 epochs)	16.22	17.68	14.31
Διακύμανση (Μετά από 15 epochs)	0.82	2.64	1.07

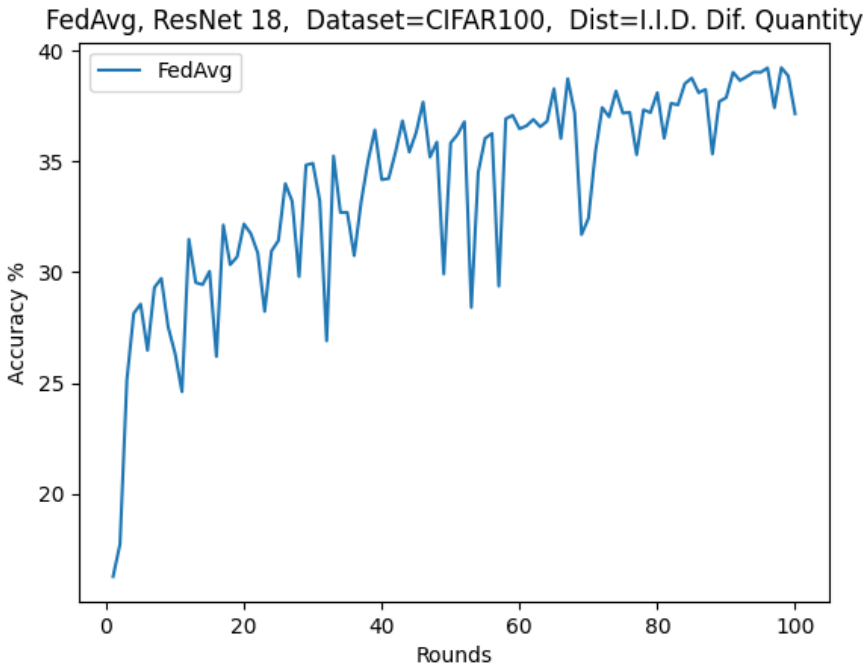
CIFAR100 – ResNet 18 (Μέγιστη Επίδοση 44.52):

Για IID κατανομή:



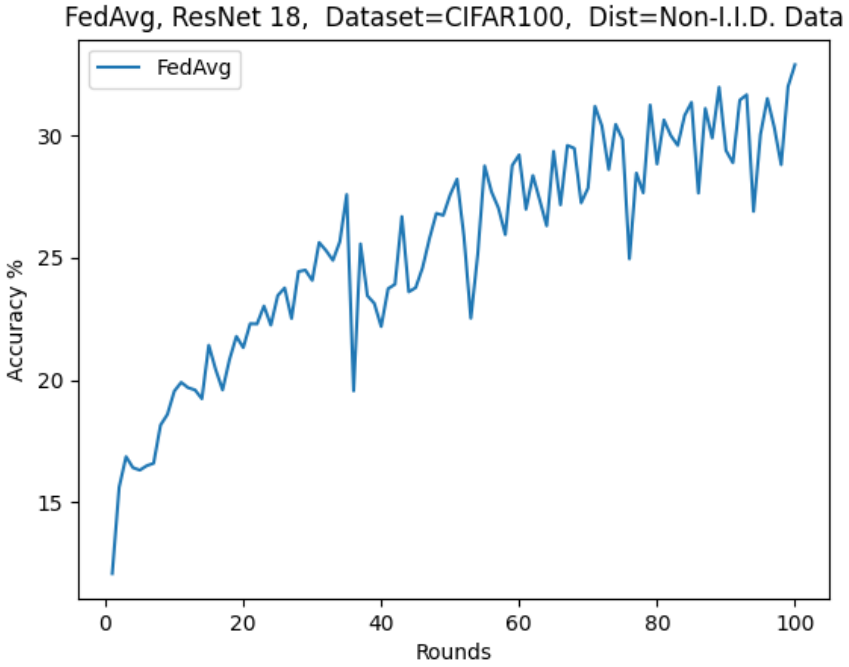
Διάγραμμα 6.1-10 FedAvg, CIFAR100, ResNet18, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.1-11FedAvg, CIFAR100, ResNet18, IID Dif. Quantity Data

Για Non-IID κατανομή:



Διάγραμμα 6.1-12 Διάγραμμα 6.1 11FedAvg, CIFAR100, ResNet18, non-IID Data

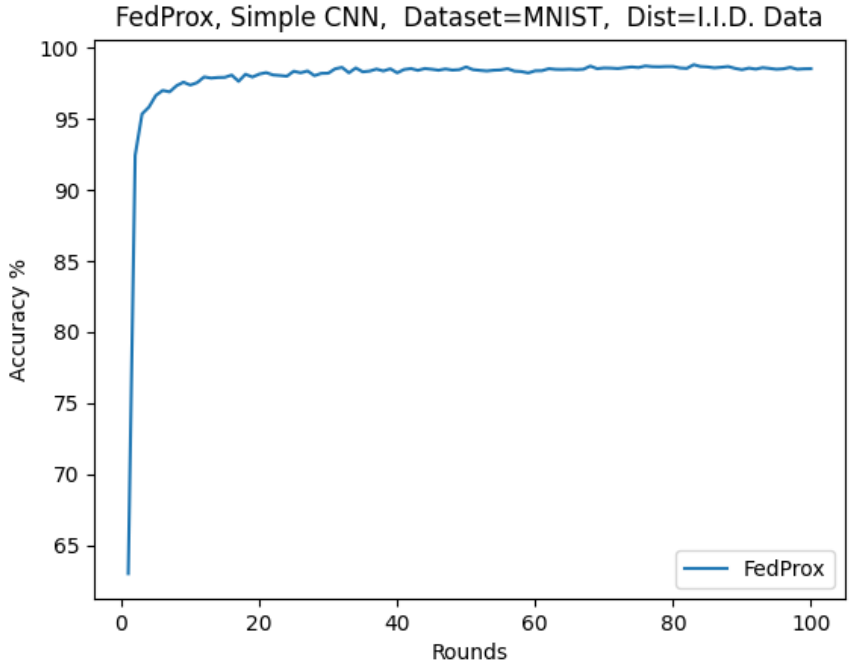
Πίνακας 6.1-4 FedAvg, CIFAR100, ResNet18

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	37.84	39.24	32.91
Μέση (Μετά από 15 epochs)	33.00	35.25	26.83
Διακύμανση (Μετά από 15 epochs)	8.56	9.82	10.95

6.2 FedProx – Αποτελέσματα

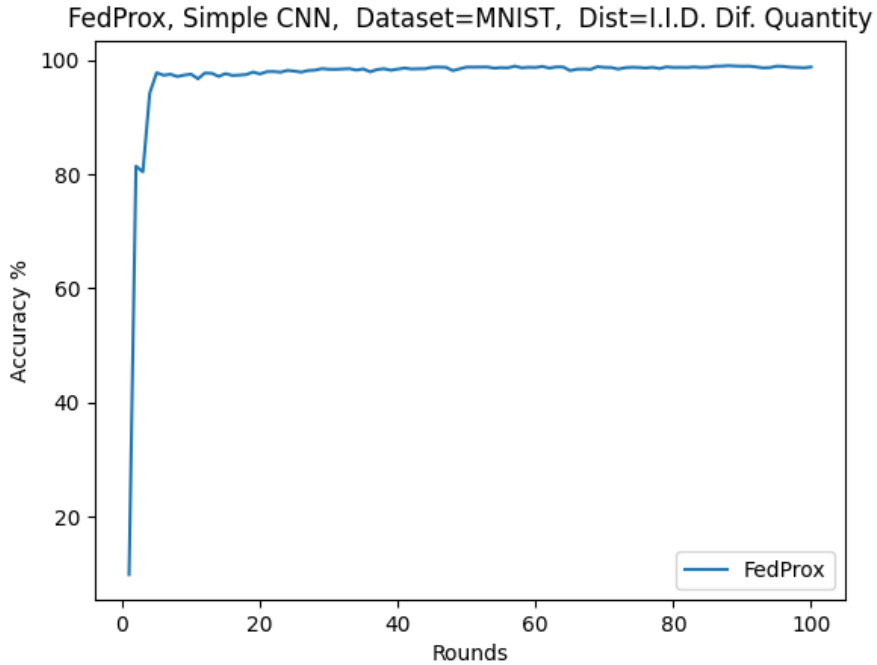
MNIST – Simple CNN (Μέγιστη Επίδοση 98.97):

Για IID κατανομή:



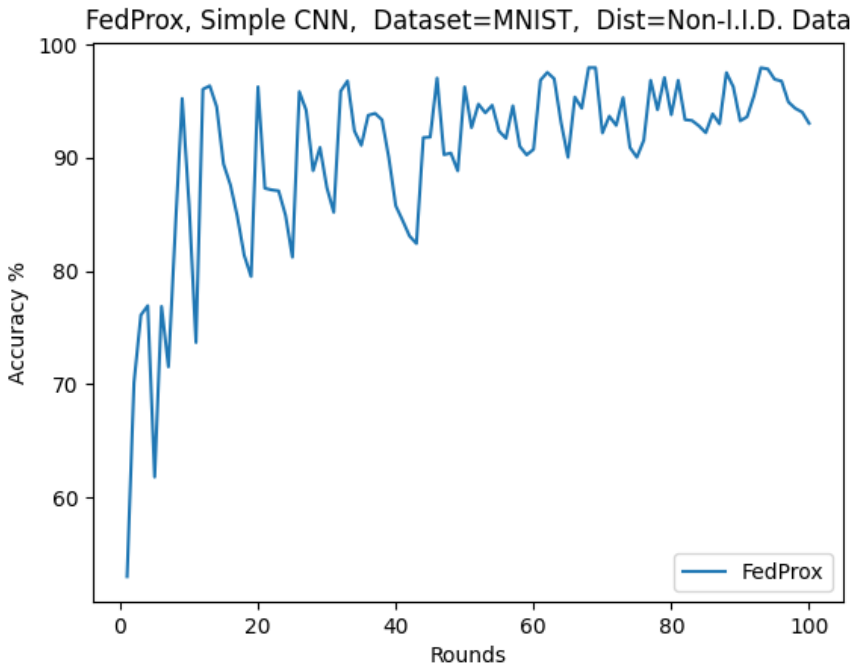
Διάγραμμα 6.2-1 FedProx, MNIST, SimpleCNN, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.2-2 FedProx, MNIST, SimpleCNN, IID Dif. Quantity Data

Για Non-IID κατανομή:



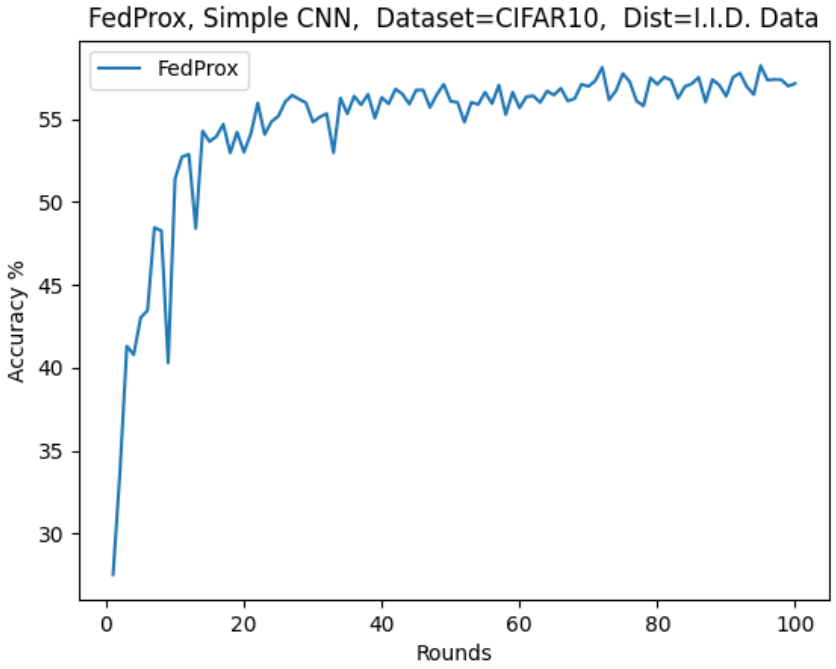
Διάγραμμα 6.2-3 FedProx, MNIST, SimpleCNN, Non-IID Data

Πίνακας 6.2-1 FedProx, MNIST, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	98.79	99.06	97.97
Μέση (Μετά από 15 epochs)	98.42	98.55	92.26
Διακύμανση (Μετά από 15 epochs)	0.04	0.13	18.81

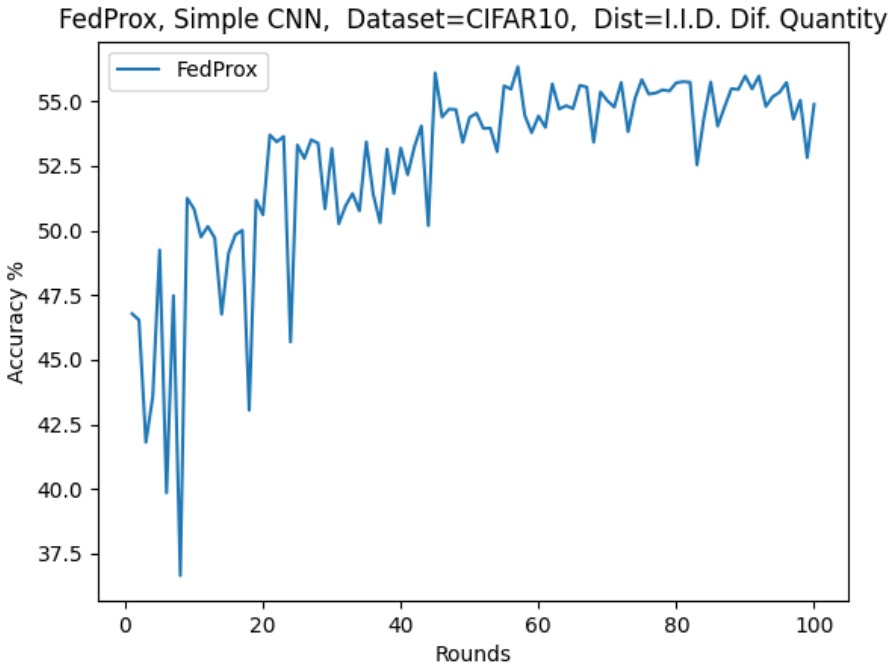
CIFAR10 – Simple CNN (Μέγιστη Επίδοση 64.2):

Για IID κατανομή:



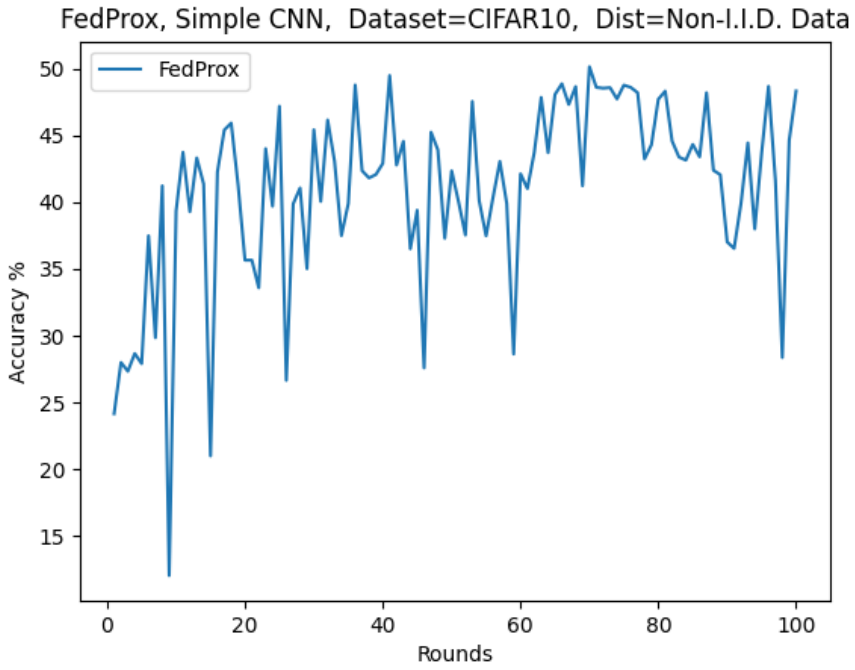
Διάγραμμα 6.2-4 FedProx, CIFAR10, SimpleCNN, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.2-5 FedProx, CIFAR10, SimpleCNN, IID Dif. Quantity Data

Για Non-IID κατανομή:



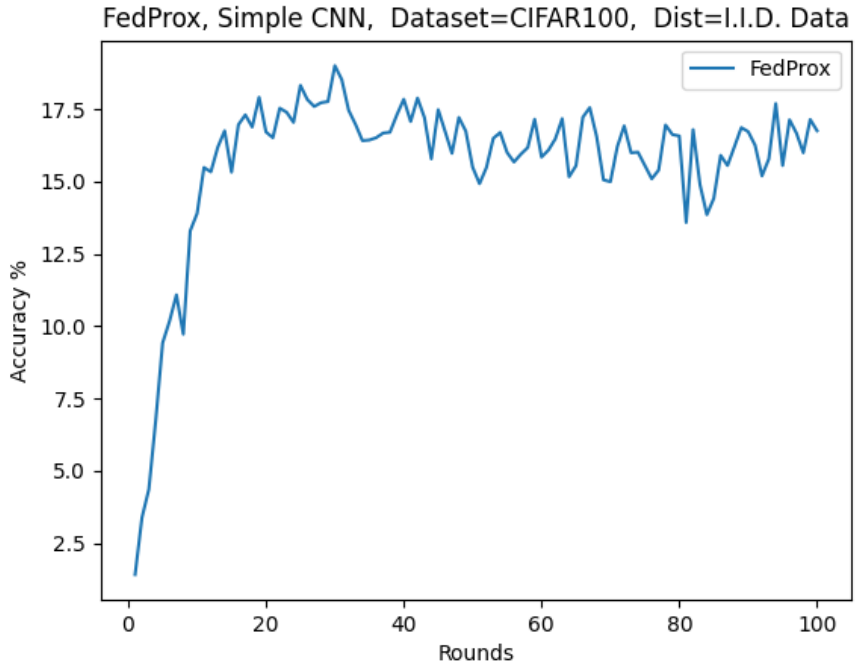
Διάγραμμα 6.2-6 FedProx, CIFAR10, SimpleCNN, Non-IID Data

Πίνακας 6.2-2 FedProx, CIFAR10, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	58.23	56.34	50.14
Μέση (Μετά από 15 epochs)	56.23	53.74	42.53
Διακύμανση (Μετά από 15 epochs)	1.23	5.03	26.75

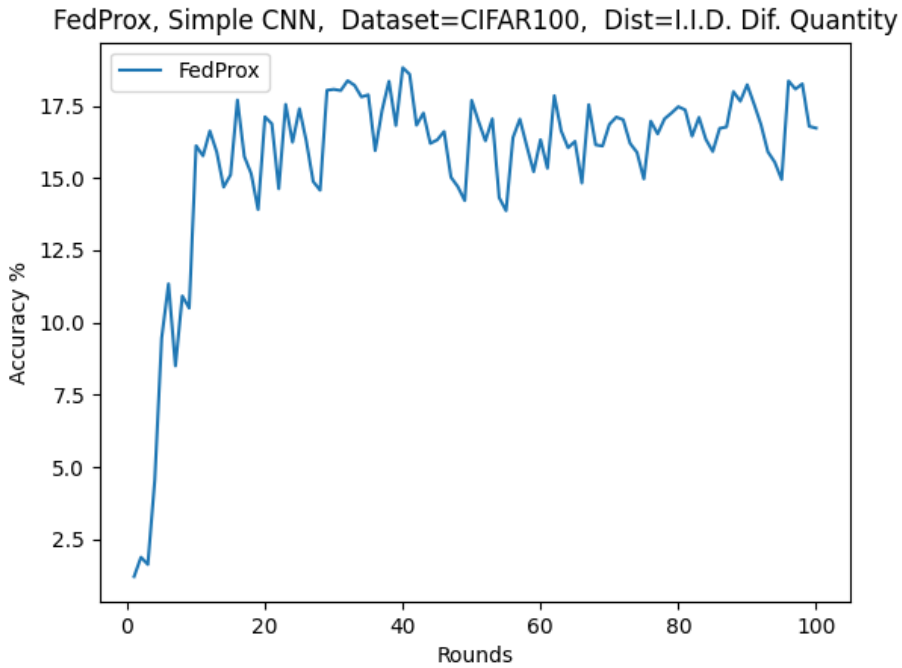
CIFAR100 – Simple CNN (Μέγιστη Επίδοση 29.61):

Για IID κατανομή:



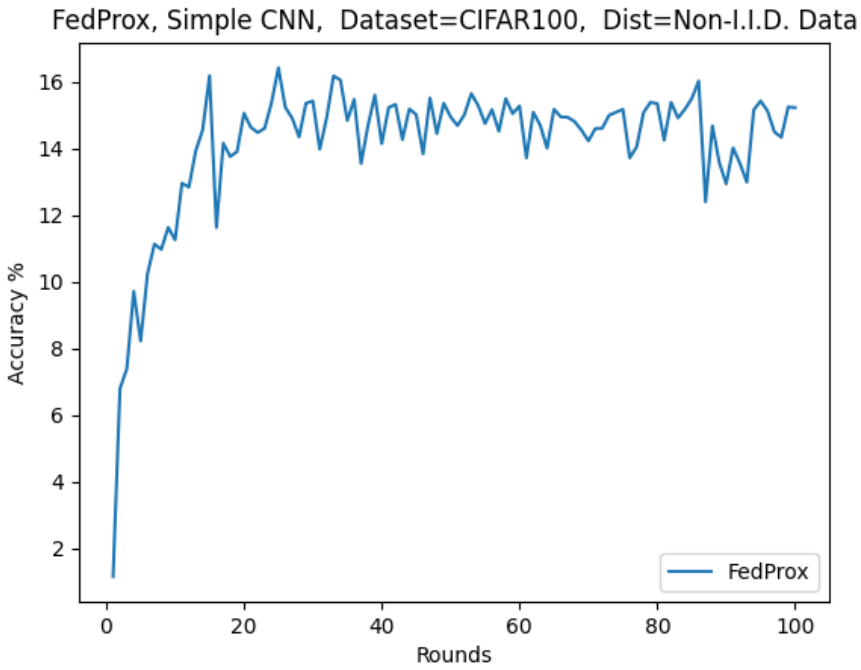
Διάγραμμα 6.2-7 FedProx, CIFAR100, SimpleCNN, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.2-8 FedProx, CIFAR100, SimpleCNN, IID Dif. Quantity Data

Για Non-IID κατανομή:



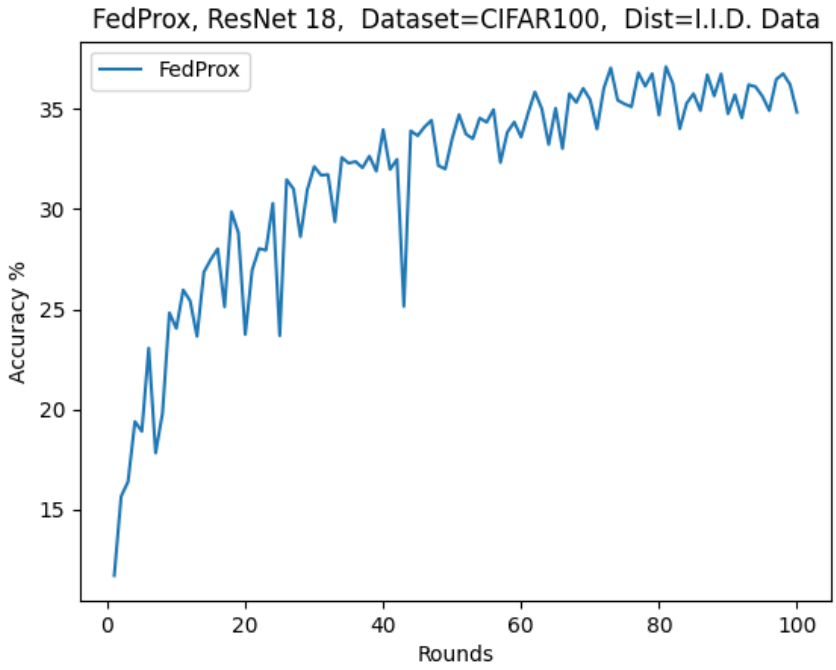
Διάγραμμα 6.2-9 FedProx, CIFAR100, SimpleCNN, Non-IID Data

Πίνακας 6.2-3 FedProx, CIFAR100, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	19.00	18.82	16.43
Μέση (Μετά από 15 epochs)	16.50	16.64	14.74
Διακύμανση (Μετά από 15 epochs)	1.00	1.37	0.64

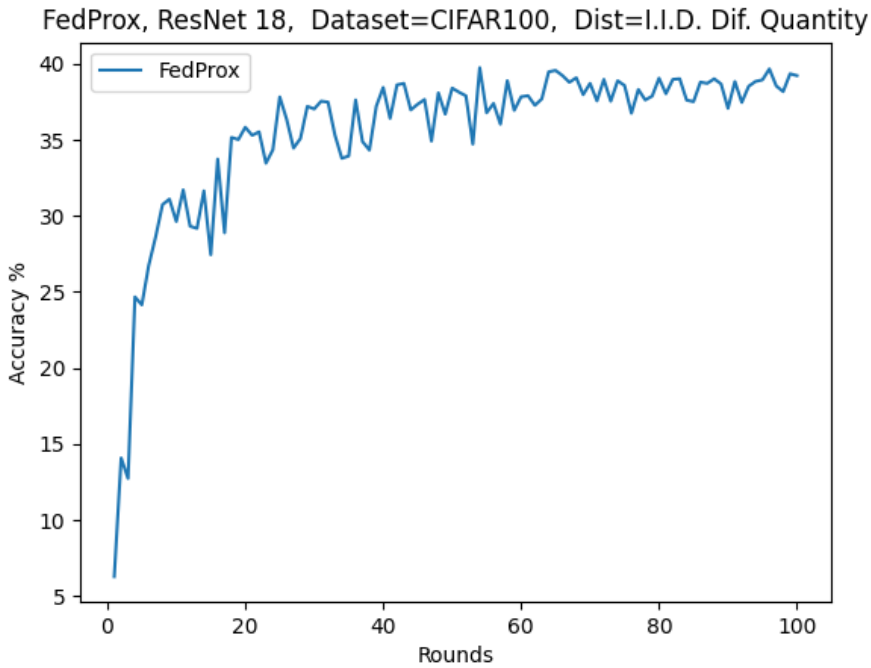
CIFAR100 – ResNet 18 (Μέγιστη Επίδοση 44.52):

Για IID κατανομή:



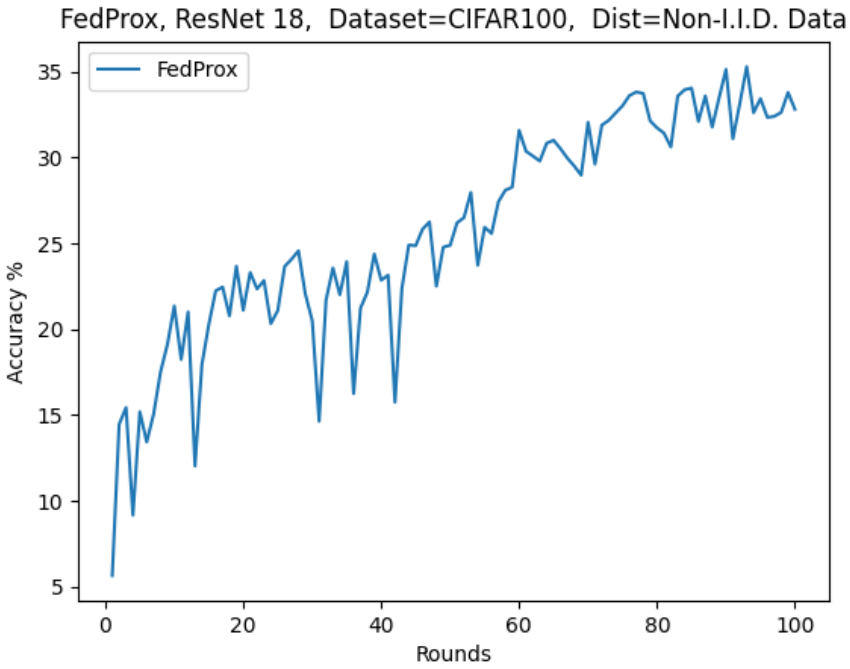
Διάγραμμα 6.2-10 FedProx, CIFAR100, ResNet18, IID Data

Για IID κατανομή με διαφορετική ποσότητα δεδομένων ανά Clients:



Διάγραμμα 6.2-11 FedProx, CIFAR100, ResNet18, IID Dif. Quantity Data

Για Non-IID κατανομή:



Διάγραμμα 6.2-12 FedProx, CIFAR100, ResNet18, Non-IID Data

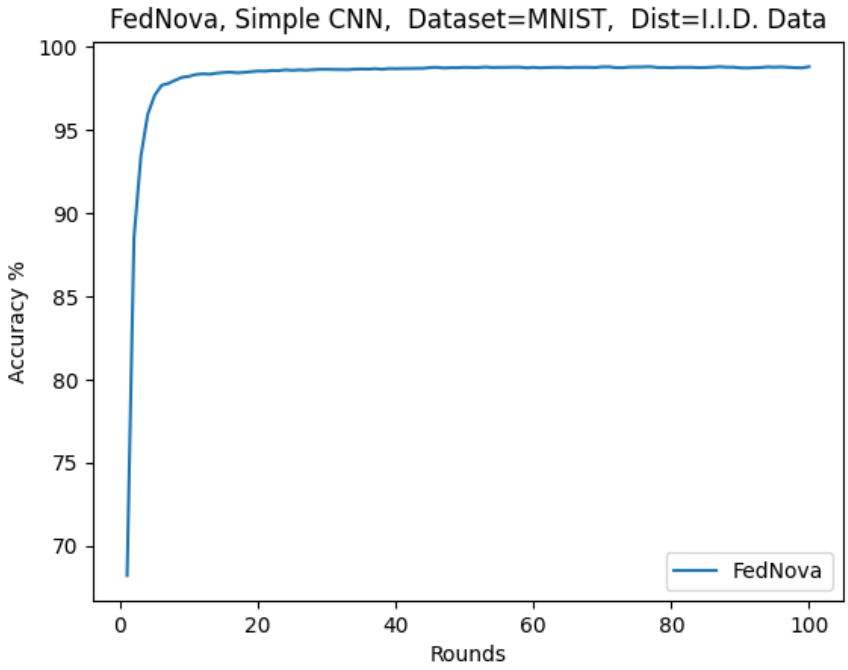
Πίνακας 6.2-4 FedProx, CIFAR100, ResNet18

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	37.09	39.74	35.29
Μέση (Μετά από 15 epochs)	33.30	37.49	27.54
Διακύμανση (Μετά από 15 epochs)	9.52	3.44	25.70

6.3 FedNova – Αποτελέσματα

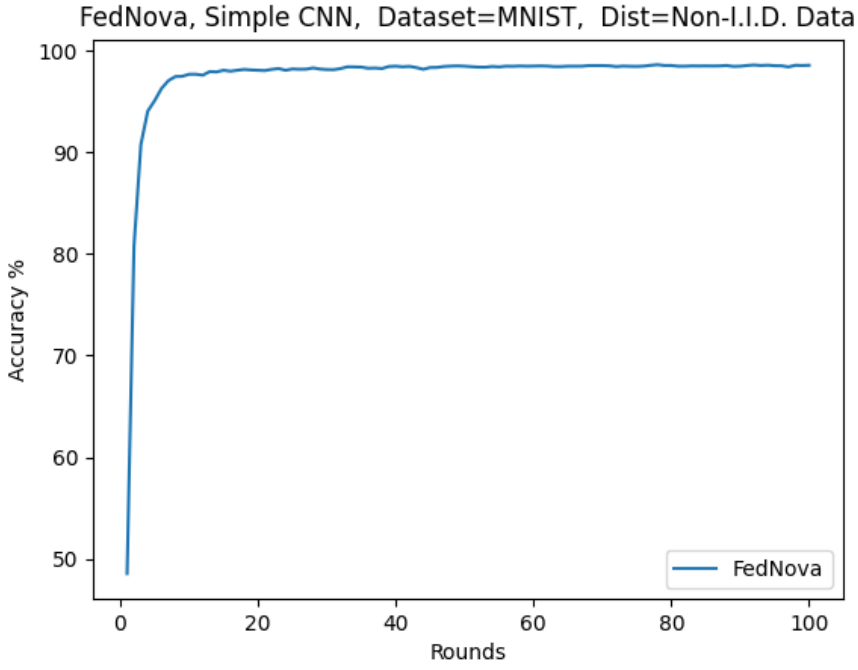
MNIST – Simple CNN (Μέγιστη Επίδοση 98.97):

Για IID κατανομή:



Διάγραμμα 6.3-1 FedNova, MNIST, SimpleCNN, IID Data

Για Non-IID κατανομή:



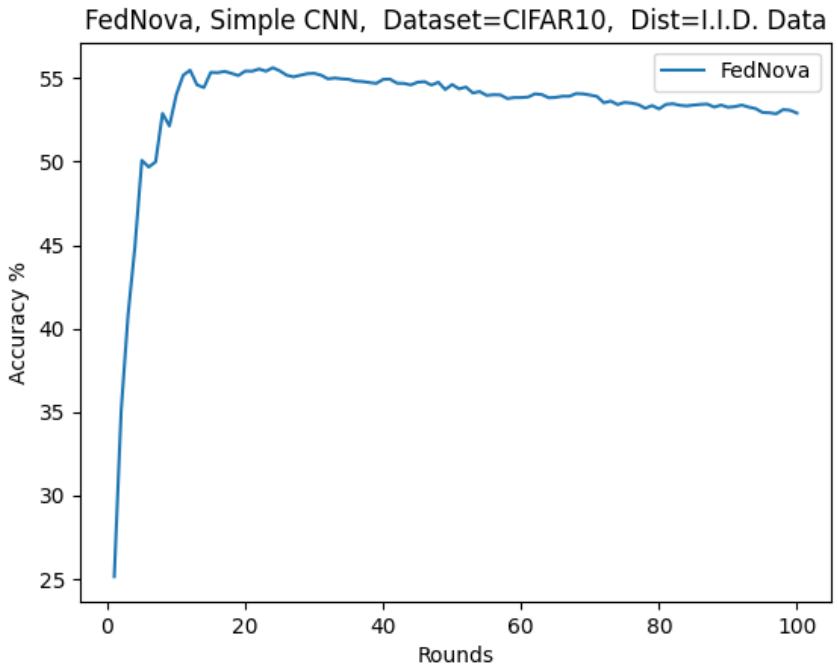
Διάγραμμα 6.3-2 FedNova, MNIST, SimpleCNN, Non-IID Data

Πίνακας 6.3-1 FedNova, MNIST, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	98.81	-	98.63
Μέση (Μετά από 15 epochs)	98.71	-	98.40
Διακύμανση (Μετά από 15 epochs)	0.007	-	0.02

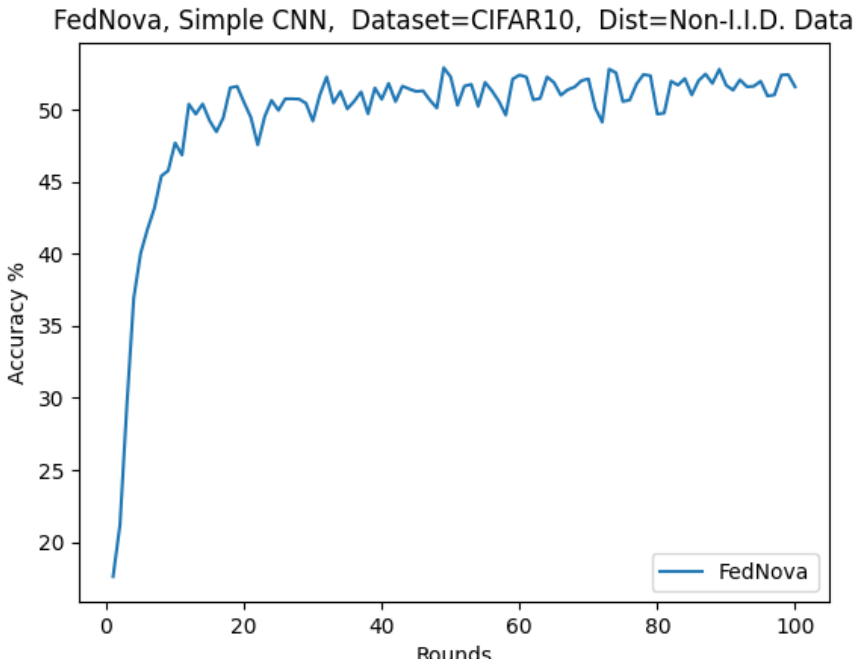
CIFAR10 – Simple CNN (Μέγιστη Επίδοση 64.2):

Για IID κατανομή:



Πίνακας 6.3-2 FedNova, CIFAR10, SimpleCNN, IID Data

Για Non-IID κατανομή:



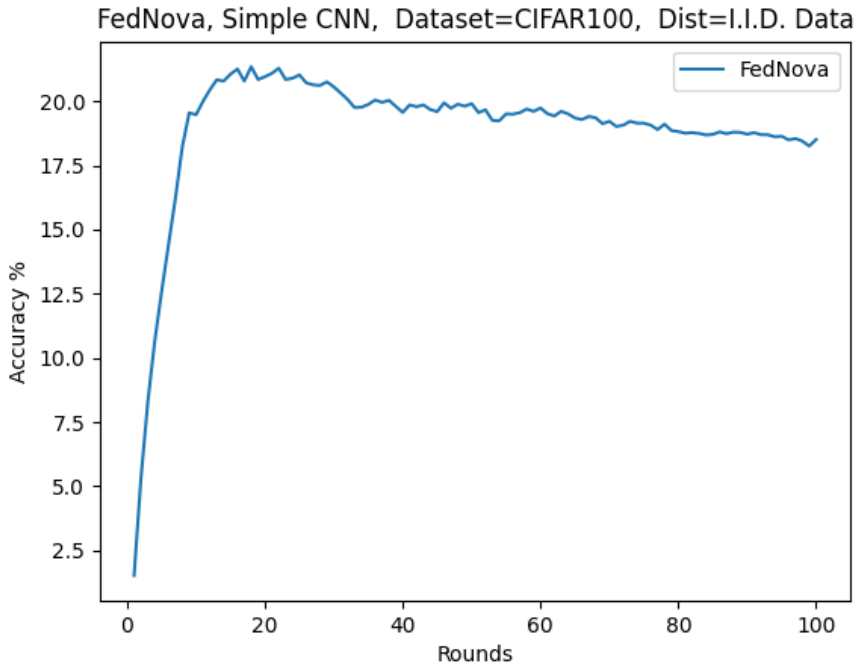
Πίνακας 6.3-3 FedNova, CIFAR10, SimpleCNN, Non-IID Data

Πίνακας 6.3-4 FedProx, CIFAR10, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	55.63	-	52.93
Μέση (Μετά από 15 epochs)	54.18	-	51.16
Διακύμανση (Μετά από 15 epochs)	0.63	-	1.10

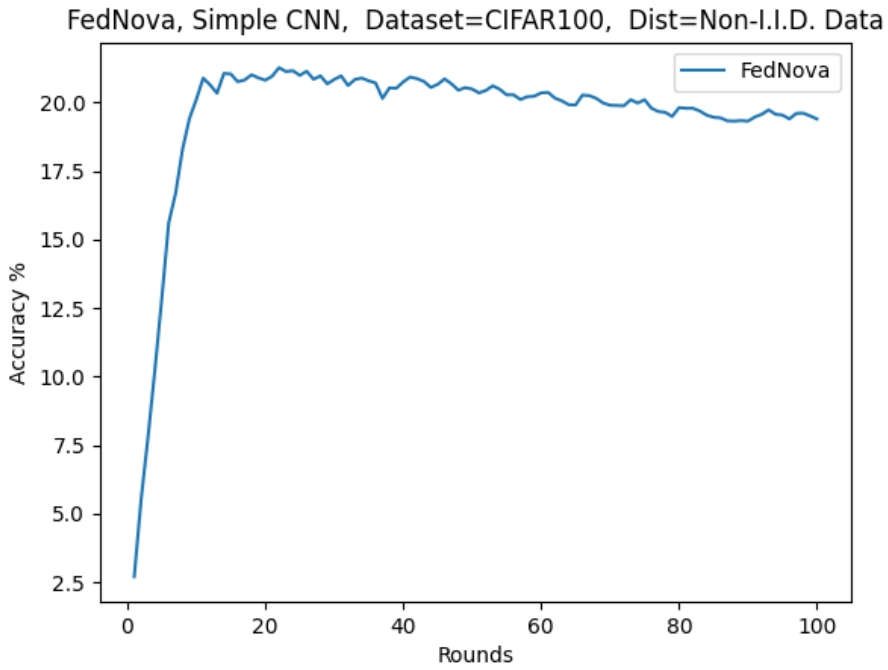
CIFAR100 – Simple CNN (Μέγιστη Επίδοση 29.61):

Για IID κατανομή:



Διάγραμμα 6.3-3 FedNova, CIFAR100, SimpleCNN, IID Data

Για Non-IID κατανομή:



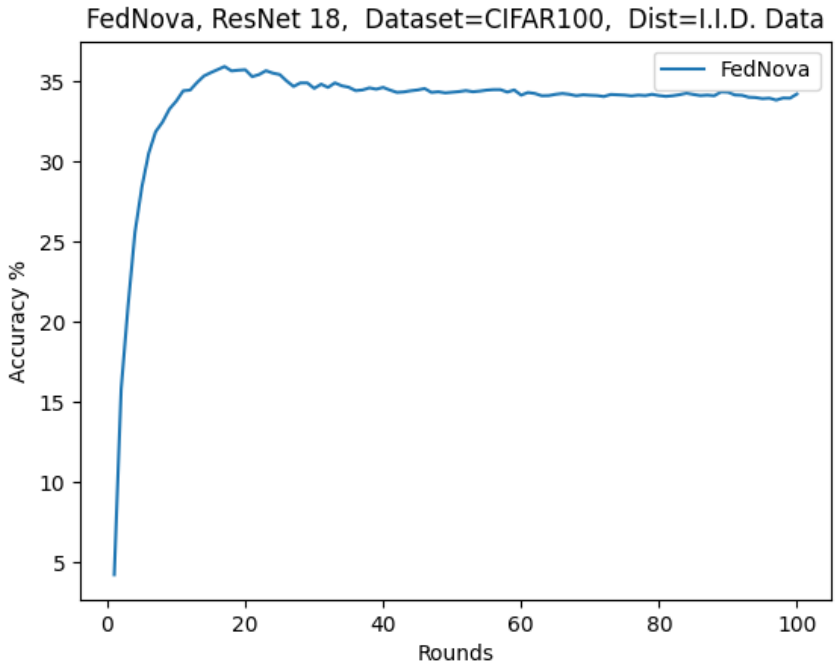
Διάγραμμα 6.3-4 FedNova, CIFAR100, SimpleCNN, Non-IID Data

Πίνακας 6.3-5 FedNova, CIFAR100, SimpleCNN

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	21.34	-	21.27
Μέση (Μετά από 15 epochs)	19.57	-	20.24
Διακύμανση (Μετά από 15 epochs)	0.60	-	0.31

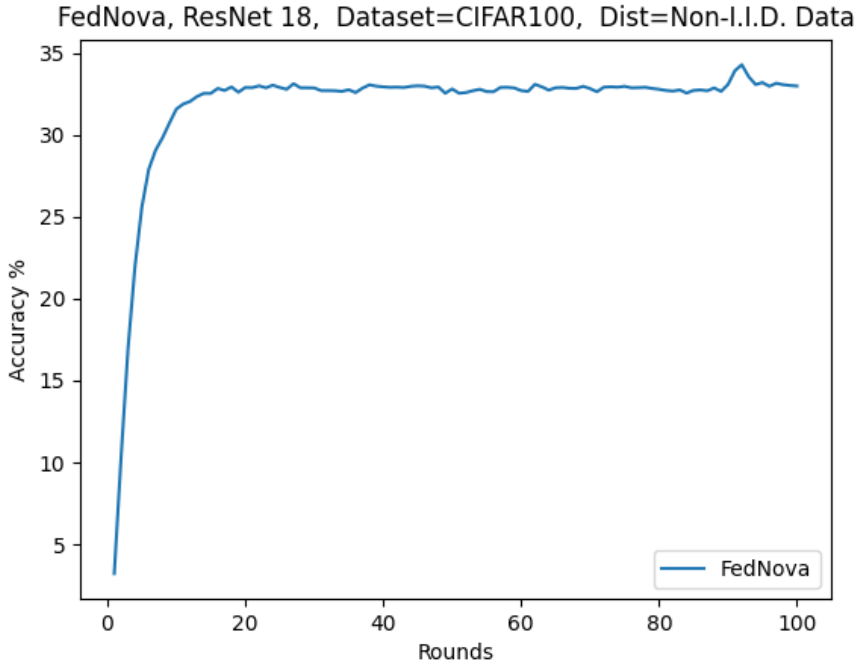
CIFAR100 – ResNet 18 (Μέγιστη Επίδοση 44.52):

Για IID κατανομή:



Διάγραμμα 6.3-5 FedNova, CIFAR100, ResNet18, IID Data

Για Non-IID κατανομή:



Διάγραμμα 6.3-6 FedNova, CIFAR100, ResNet18, Non-IID Data

Πίνακας 6.3-6 FedNova, CIFAR100, ResNet18

Επίδοση	IID	IID Dif. Quantity	Non – IID
Υψηλότερη	35.91	-	34.29
Μέση (Μετά από 15 epochs)	34.45	-	32.89
Διακύμανση (Μετά από 15 epochs)	0.23	-	0.06

6.4 Ανάλυση Αποτελεσμάτων – Συγκρίσεις

Μετά, την παρουσίαση των αποτελεσμάτων μπορούν να εξαχθούν τα εξής συμπεράσματα σχετικά με τις αποδόσεις των αλγορίθμων που μελετήθηκαν και δοκιμάστηκαν.
Και για τους τρείς αλγόριθμους μπορεί να ειπωθεί με ασφάλεια, πως η επίδοση του κεντρικού μοντέλου, όπως είναι φυσικό, αφού ανάλογη είναι και η απόδοση του μοντέλου που χρησιμοποιείται, χειροτερεύει όσο αυξάνεται η πολυπλοκότητα των δεδομένων εκπαίδευσης. Με την γενικότερη επίδοση και των τριών να είναι καλύτερη στο dataset MNIST και να φθίνει για το CIFAR10 και να μειώνεται ακόμη περισσότερο για το CIFAR100.

Αρχικά, για τον FedAvg, τον αλγόριθμο του οποίου, στην λειτουργία του στηρίζονται οι άλλοι. Παρατηρείται μία ικανοποιητική συμπεριφορά της απόδοσης του κεντρικού μοντέλου που παράγεται όταν τα δεδομένα είναι όμοια κατανεμημένα σε κλάσεις στους clients τόσο όσο όταν έχουν ίδιο (ή αρκετά κοντινό) συνολικό αριθμό δεδομένων δηλαδή για IID και IID με διαφορετική ποσότητα, με την δεύτερη περίπτωση να δυσχεραίνει την σταθερότητα του μοντέλου, όχι όμως απαραίτητα και την απόδοση αφού σε κάποιες περιπτώσεις παρουσιάστηκε μεγαλύτερη μέγιστη απόδοση και κυριότερα και μέση απόδοση του κεντρικού συστήματος διατηρώντας την έτσι σε ικανοποιητικό βαθμό σε σχέση με την πρώτη. Η συμπεριφορά αυτή, όμως χειροτερεύει δραστικά στην τρίτη περίπτωση, όπου έχουμε ανομοιογενή δεδομένα, non-IID. Βλέπουμε μεγάλη μείωση της απόδοσης, τόσο στις τιμές της μέγιστης και της μέσης τιμής, αλλά βλέπουμε και μεγάλη αστάθεια με την τιμή της διακύμανσης να ανεβαίνει σε οριακά μη επιτρεπτά και ασφαλή για χρήση επίπεδα. Ωστόσο, δεδομένης της ιδιαιτερότητας που έχει ως ο αλγόριθμος που αποτελεί το επίπεδο αναφοράς για όλους τους άλλους αποτελεί τον πιο απλό και εύκολα υλοποιήσιμο αλγόριθμο. Επιπλέον, αυτό κάνει το κόστος επικοινωνίας των clients με τον server, να είναι το ελάχιστο δυνατό.

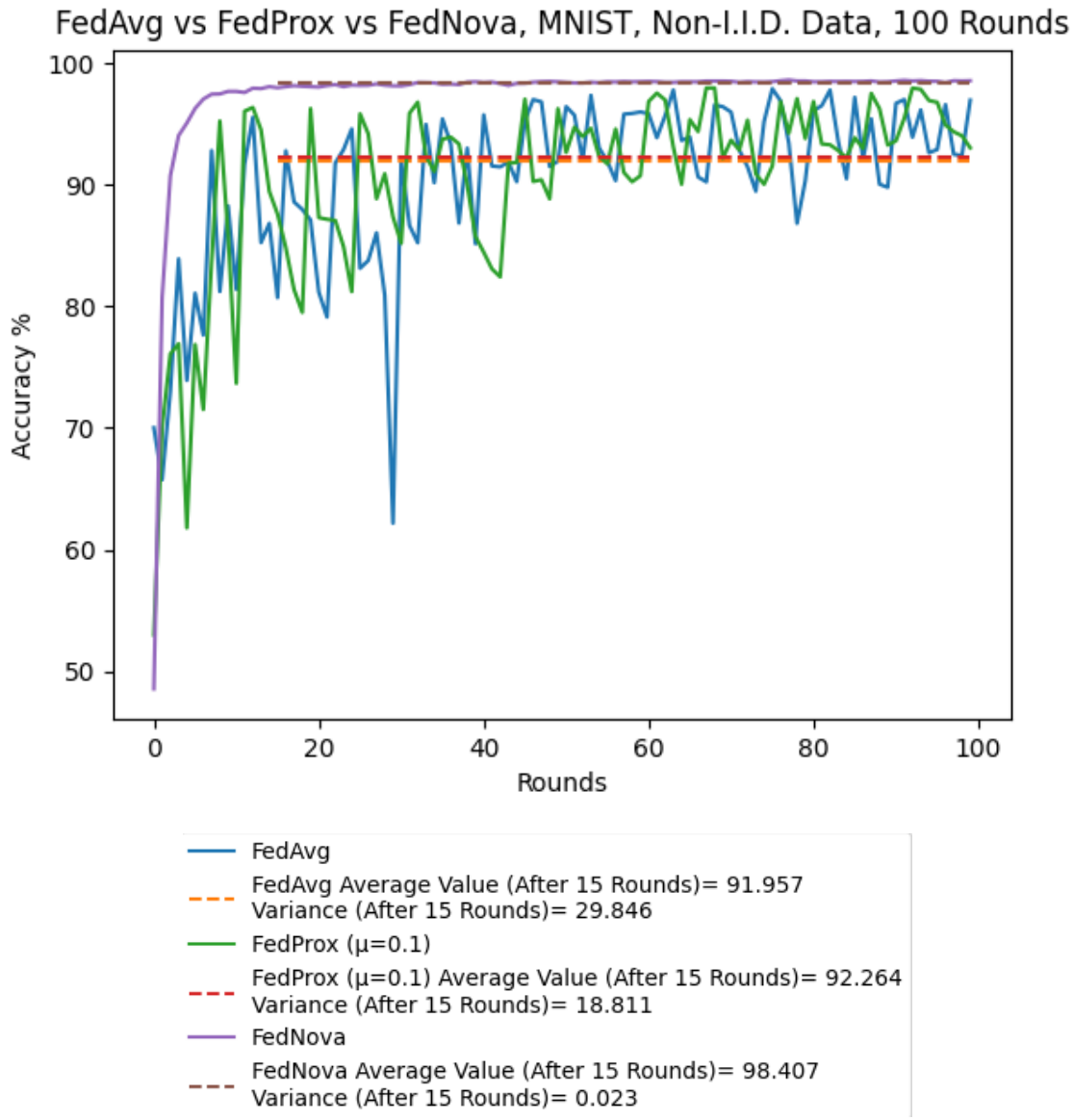
Στην συνέχεια, για τον FedProx, παρατηρείται άμεσα και εύκολα πως πρόκειται για μια αποδοτική αναβάθμιση του FedAvg. Έχει παρόμοια συμπεριφορά, αλλά με μικρότερες αποκλίσεις από την καλύτερη επίδοση που επιτυγχάνεται από το κεντρικό μοντέλο. Ξανά συναντάται η καλύτερη συμπεριφορά σε IID και ίσα κατανεμημένα δεδομένα. Για IID με διαφορετική ποσότητα, ξανά, υπάρχουν ικανοποιητικά αποτελέσματα ίσα και καλύτερα από αυτά του FedAvg. Η μεγάλη διαφορά των δύο όμως φαίνεται στη χειρότερη κατανομή δεδομένων αυτή για non- IID. Η επίδοση του FedProx, όπως είναι φυσικό είναι πολύ χειρότερη σε σχέση με τις άλλες κατανομές αλλά αρκετά καλύτερη από αυτή του FedAvg. Παρατηρούνται μεγαλύτερες τιμές τόσο για την μέγιστη όσο και για την μέση απόδοση του κεντρικού, αλλά οι μεγάλες διαφορές υπάρχουν στην σταθερότητα της απόδοσης του κεντρικού μοντέλου που παράγει ο FedProx με την διακύμανση της απόδοσης στα σενάρια να είναι κοντά στη μισή σε σχέση με αυτής του FedAvg. Ακόμα, ο FedProx αποτελεί και αυτός έναν απλό και επικοινωνιακά αποδοτικό αλγόριθμο, αφού εισάγει τον Proximal Term, όπως αναλύθηκε στο 5.4, αλλά σε κάθε ανανέωση του τοπικού μοντέλου χωρίς να υπάρχει κάποια επιπλέον πληροφορία που θα πρέπει να μεταδοθεί από τον client στον server και αντίθετα, πέρα από μόνο την μεταβλητή μ που είναι απλά ένας δεκαδικός αριθμός με σχεδόν μηδενικό μέγεθος (και αν χρησιμοποιηθεί η μεταβλητή γ για κάθε client).

Τέλος, για τον αλγόριθμο FedNova, είναι πολύ ορατή η διαφορά σε σχέση με του δύο προηγούμενους αλγόριθμους. Με την χρήση αυτού του αλγόριθμου παρατηρείται μια εξαιρετικά καλύτερη απόδοση κυρίαρχα όσον αφορά την σταθερότητα του μοντέλου του server που παράγεται. Όπως και για τους προηγούμενους αλγόριθμους η απόδοση είναι πολύ καλή για IID κατανομή. Ο FedNova συγκλίνει στην καλύτερη τιμή συνήθως πιο γρήγορα από τους άλλους, κάτι που μπορεί να τον κάνει αρκετά πιο αποδοτικό αφού δεν θα υπάρχει η ανάγκη για άσκοπους γύρους επικοινωνίας. Συνολικά, υπάρχει σε αρκετά σενάρια καλύτερη επίδοση και στις μέγιστες τιμές που επιτυγχάνονται αλλά αυτό που ξεχωρίζει είναι οι τιμές της μέσης απόδοσης και της διακύμανσης της απόδοσης για τα σενάρια non-IID. Η σταθερότητα

που προσφέρει είναι πολύ καλύτερη από αυτή του FedProx και προφανώς και του FedAvg. Ωστόσο, όπως είναι εύκολα παρατηρήσιμο και από τα διαγράμματα πιο πάνω δεν υπάρχει για κανένα από τα τρία datasets, το σενάριο με κατανομή με διαφορετική ποσότητα δεδομένων. Αυτό συμβαίνει καθώς, τουλάχιστον στην υλοποίηση της προσομοίωσης που πραγματοποιήθηκε για τον σκοπό της παρούσας εργασίας, το κεντρικό μοντέλο που εξήγαγε ο FedNova παρουσίαζε εξαιρετικά μεγάλο overfitting σε δεδομένα των clients που είχαν τον μεγαλύτερο συνολικό όγκο δεδομένων, με την απόδοση του να πέφτει σε ακραία χαμηλά επίπεδα. Μικρή μείωση της απόδοσης με την αύξηση των rounds παρατηρείται σε σενάρια IID και σε non-IID που όπως έχει αναφερθεί στο 3.4 έχει μικρότερες διαφορές στην ποσότητα των δεδομένων από τη δεύτερη κατανομή, κάτι που μπορεί όμως να οφείλεται στην πολύ γρήγορη σύγκλιση στην καλύτερη τιμή που αναφέρθηκε και όχι απαραίτητα σε παρόμοιο πρόβλημα με αυτό της κατανομής με διαφορετική ποσότητα δεδομένων. Ακόμα ο FedNova έχει μεγαλύτερη πολυπλοκότητα και επικοινωνιακό κόστος σε σχέση με τους δύο προηγούμενους αφού όπως εξηγήθηκε και στο κεφάλαιο 5.5, εισάγει επιπλέον υπολογισμούς και πρέπει να γίνει η μεταφορά των κανονικοποιημένων κλίσεων και του διανύσματος α, με το οποίο γίνεται η κανονικοποίηση.

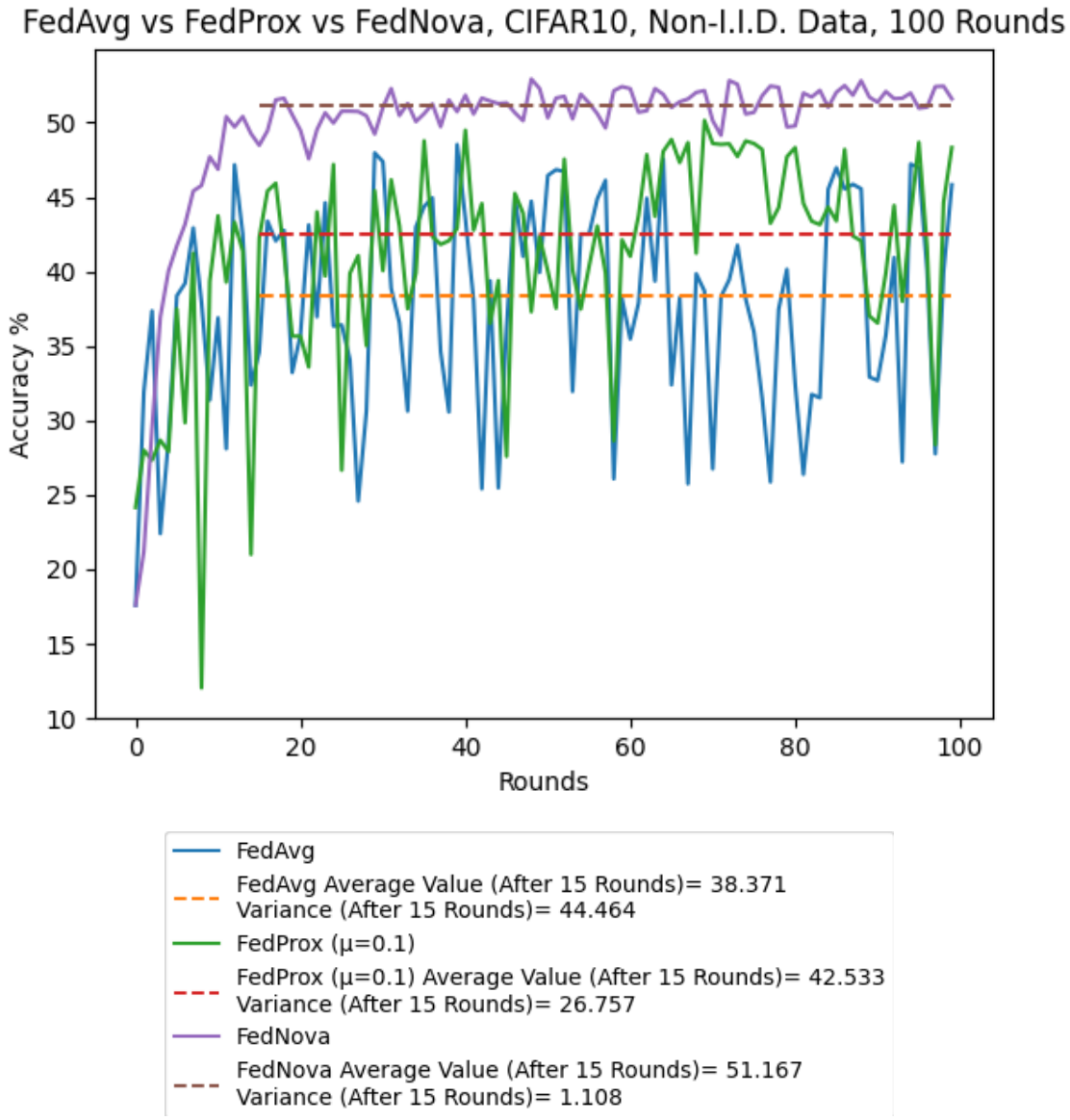
Συγκεντρωτικά και για κατανομή non-IID, αφού για IID και οι τρεις αλγόριθμοι έχουν ικανοποιητική απόδοση:

Dataset MNIST – Simple CNN:



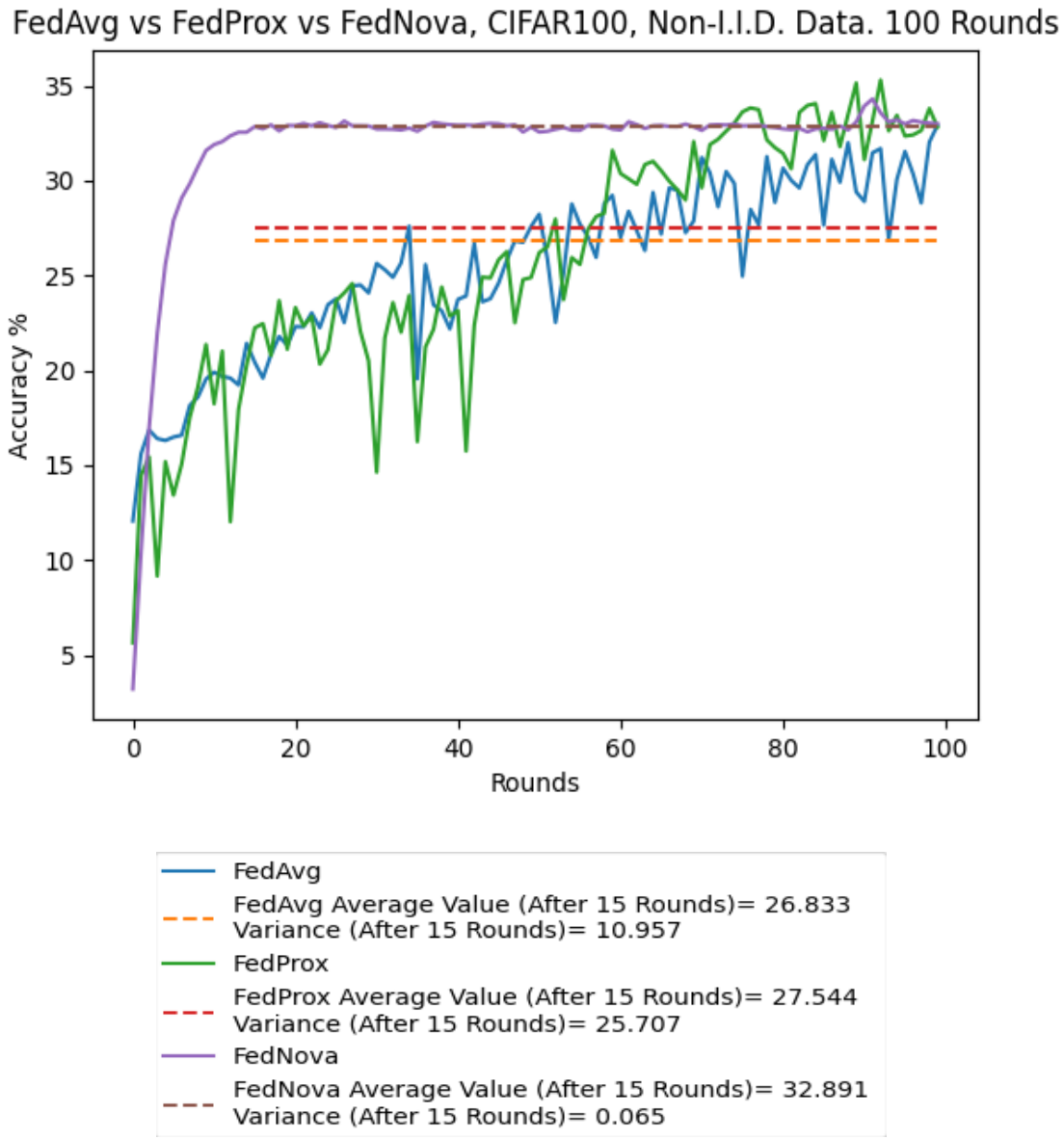
Διάγραμμα 6.4-1 FedAvg vs FedProx vs FedNova, MNIST, Non-IID Data

Dataset CIFAR10 – Simple CNN:



Διάγραμμα 6.4-2 FedAvg vs FedProx vs FedNova, CIFAR10, Non-IID Data

Dataset CIFAR100 – ResNet18:



Διάγραμμα 6.4-3 FedAvg vs FedProx vs FedNova, CIFAR100, Non-IID Data

ΚΕΦΑΛΑΙΟ 7 ° Συμπεράσματα – Βελτιώσεις

Από τις συγκρίσεις για cross-silo σενάρια που υλοποιήθηκαν μέσω προσομοίωσης, φαίνεται ότι η επιλογή ενός αλγορίθμου ως του επικρατέστερου και του καλύτερου θα πρέπει να κριθεί για την εκάστοτε εφαρμογή. Αυτό συμβαίνει αφού όπως φάνηκε και από τις μετρήσεις ο FedNova έχει την καλύτερη συμπεριφορά για την απόδοση του κεντρικού μοντέλου αλλά, σύμφωνα με τα αποτελέσματα που λήφθηκαν μέσω της συγκεκριμένης υλοποίησης που έγινε φαίνεται ακατάλληλος όταν πρόκειται για υλοποιήσεις με IID δεδομένα με διαφορετική ποσότητα δεδομένων ανά client.

Αν πρόκειται για τέτοιο σενάριο η καλύτερη επιλογή μετά από τον FedNova, φαίνεται να είναι η υλοποίηση με την χρήση του FedProx. Η χρήση του FedAvg, προτείνεται για δοκιμαστικές υλοποιήσεις και οπωσδήποτε για αρχικές εφαρμογές Ομοσπονδιακής Μάθησης σε σενάρια που δεν έχουν υλοποιηθεί ξανά με αυτό το είδος υπολογιστικής μάθησης.

Επιπλέον, θα πρέπει να αναφερθεί πως τα παραπάνω ισχύουν μόνο όσον αφορά, την απόδοση και την συμπεριφορά του κεντρικού μοντέλου. Θα πρέπει να ληφθεί υπόψη για κάθε εφαρμογή ποιος αλγόριθμος αξίζει να υλοποιηθεί δεδομένης της πολυπλοκότητας που έχει, με τους FedProx και FedAvg, να προτείνονται για ευκολότερη υλοποίηση και για λιγότερο υπολογιστικά κόστος και κόστος μεταφοράς δεδομένων.

Οι μετρήσεις που λήφθηκαν στην παρούσα εργασία αποτελούν μία βάση για την κατανόηση και την επιλογή αλγορίθμου συσσωμάτωσης σε εφαρμογές Ομοσπονδιακής Μάθησης, ωστόσο οι υλοποιήσεις επιδέχονται αρκετές βελτιώσεις για να εξαχθούν ασφαλέστερα συμπεράσματα. Αρχικά οι περιορισμοί που λήφθηκαν για τις δοκιμές, επηρεάζουν την εγκυρότητα των αποτελεσμάτων, αφού δεν μπορεί να υπάρχει βεβαιότητα ότι δεν θα ήταν διαφορετικά αν λαμβάνονταν άλλες επιλογές. Θα πρέπει να εκτελεστούν δοκιμαστικά σενάρια με κύριες διαφοροποιήσεις την επιλογή διαφορετικών μοντέλων υπολογιστικής μάθησης και διαφορετικών, αλγόριθμων βελτιστοποίησης των τοπικών μοντέλων διαφορετικών συνόλων δεδομένων, με αυξημένες δυνατότητες αυτής της επιλογής στον FedNova και ιδιαίτερα ακόμα και τον συνδυασμό του με τον Proximal Term του FedProx.

Επιπλέον, σημαντικές είναι και οι υλοποιήσεις με διαφορετική επιλογή δεδομένων και διαφορετικών συστημάτων που συμμετέχουν στην Ομοσπονδιακή Μάθηση, με την διαφοροποίηση αυτή να γίνεται μέχρι τον βαθμό που επιτρέπει η φύση των cross-silo σεναρίων. Επίσης, δοκιμαστικές υλοποιήσεις θα πρέπει να γίνουν έτσι ώστε να κριθεί η απόδοση των αλγορίθμων και με την εισαγωγή θορύβου στα δεδομένα που μεταφέρονται δηλαδή τους συντελεστές των μοντέλων αφού είναι αποτελεί πολύ σημαντικό κομμάτι ασφαλείας σε πραγματικές υλοποιήσεις

Ιδιαίτερα σημαντικές βελτιώσεις έτσι ώστε να γίνει σωστή κρίση των αλγορίθμων και της συνολικής τους απόδοσης αποτελούν η σωστή καταγραφή του χρόνου που απαιτείται προκειμένου να παραχθεί και να ανανεωθεί το κεντρικό μοντέλο και η πραγματική και απόλυτη μέτρηση του όγκου των δεδομένων που μεταφέρονται από το κεντρικό σύστημα στα τοπικά και αντίθετα.

Προτείνονται ακόμα και οι διαφορετικές υλοποιήσεις των ίδιων αλγορίθμων και σε παρόμοια σενάρια προκείμενου να επαληθευτεί ή όχι η εγκυρότητα των αποτελεσμάτων που παρουσιάστηκαν.

Τέλος, θα πρέπει να ενθαρρύνεται η πραγματοποίηση περισσότερων ερευνητικών μελετών αλλά και υλοποιήσεων για τον ευρύτερο τομέα της Ομοσπονδιακής Μάθησης. Αφού πλέον η υπολογιστική νοημοσύνη βρίσκεται στο επίκεντρο της τεχνολογίας, η Ομοσπονδιακή Μάθηση αποτελεί ένα πολλά υποσχόμενο τομέα της, που όμως παρουσιάζει δυσκολίες στην εφαρμογή της και η εύρεση απαντήσεων και λύσεων σε προβλήματα που δημιουργούνται σε αυτή είναι ακόμα αρκετά απαιτητική.

Βιβλιογραφία – Αναφορές

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017. [Online].

[2] S. Russell and P. Norvig, "Artificial Intelligence: a modern approach, 4th US ed," 2021. [Online].

[3] A. P. Engelbrecht, Computational intelligence: an introduction, John Wiley & Sons, 2007.

[4] A. V. Joshi, "Machine learning and artificial intelligence," 2020. [Online].

[5] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated machine learning: Concept and applications," 2019. [Online].

[6] C. Huang, J. Huang and X. Liu, "Cross-silo federated learning: Challenges and opportunities," 2022. [Online].

[7] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," 2021. [Online].

[8] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang and Q. Yang, "Vertical federated learning," 2022. [Online].

[9] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," 2020. [Online].

[10] Q. Li, Y. Diao, Q. Chen and B. He, "Federated learning on non-iid data silos: An experimental study," 2022. [Online].

[11] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos and Y. Khazaeni, "Federated Learning with Matched Averaging," 2020. [Online]. Available: https://openreview.net/forum?id=BkluqlSFDS.

[12] A. Thakur and A. Konde, "Fundamentals of neural networks," 2021. [Online].

[13] L. V. Fausett, Fundamentals of neural networks: architectures, algorithms and applications, Pearson Education India, 2006.

[14] S. Haykin, Neural networks and learning machines, Pearson India Education Services Pvt. Ltd, 2021.

[15] A. W. Salehi, S. Khan, G. Gupta, B. I. Alabduallah, A. Almjally, H. Alsolai, T. Siddiqui and A. Mellit, "A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope," 2023. [Online].

[16] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015. [Online].

[17] J. Wu, "Introduction to convolutional neural networks," 2017. [Online].

[18] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," 2020. [Online].

[19] W. Ma and J. Lu, "An equivalence of fully connected layer and convolutional layer," 2017. [Online].

[20] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," 1998. [Online].

[21] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," 2016. [Online].

[22] C. Lemaréchal, "Cauchy and the gradient method," 2012. [Online].

[23] J. Lu, "Gradient descent, stochastic optimization, and other tales," 2022. [Online].

[24] B. Polyak, Introduction to Optimization, 2020.

[25] L. Bottou, "Stochastic gradient descent tricks," 2012. [Online].

[26] S.-i. Amari, "Backpropagation and stochastic gradient descent method," 1993. [Online].

[27] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," 2013. [Online].

[28] J. Konečn\`y, H. B. McMahan, D. Ramage and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016. [Online].

[29] J. Fu, B. Wang, H. Zhang, Z. Zhang, W. Chen and N. Zheng, "When and Why Momentum Accelerates SGD: An Empirical Study," 2023. [Online].

[30] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, "Federated optimization in heterogeneous networks," 2020. [Online].

[31] J. Wang, Q. Liu, H. Liang, G. Joshi and H. V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, vol. 33, 2020, pp. 7611-7623.

[32] Y. Shi, Y. Zhang, Y. Xiao and L. Niu, "Optimization Strategies for Client Drift in Federated Learning: A review," 2022. [Online].

[33] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," 2020. [Online].

[34] J. Reyes, L. Di Jorio, C. Low-Kam and M. Kersten-Oertel, "Precision-weighted federated learning," 2021. [Online].

Παράρτημα Α

Ο κώδικας για την υλοποίηση της προσομοίωσης υπάρχει και στο GitHub repository: https://github.com/leonmaz/Thesis_Federated_Learning_Algorithms

fed_simulation/
├── main.py
├── utils.py
├── models.py
├── data_splitter.py
├── datasets.py
├── plot.py

main.py

```
import copy
import numpy as np
import torch
import logging
import os
import torch.optim as optim
import torch.nn as nn
import matplotlib.pyplot as plt
from plot import plot
from models import SimpleCNN, ResNet
from datasets import *
from utils import *
from data_splitter import *
from mlxtend.plotting import plot_confusion_matrix
import time

def models_init(n_clients, args):

    nets = {net_i: None for net_i in range(n_clients)}

    if args.model == "simple_cnn":
        if args.dataset in ("mnist"):
            net = SimpleCNN(img_channels=1, num_classes=10,
                            fc1=nn.Linear(16 * 4 * 4, 120))
        elif args.dataset in ("cifar10"):
            net = SimpleCNN(img_channels=3, num_classes=10)
        elif args.dataset in ("cifar100"):
            net = SimpleCNN(img_channels=3, num_classes=100)

    elif args.model == "resnet_18":
        if args.dataset in ("mnist"):
            net = ResNet(img_channels=1, num_classes=10)
        elif args.dataset in ("cifar10"):
            net = ResNet(img_channels=3, num_classes=10)
        elif args.dataset in ("cifar100"):
```

```
36              net = ResNet(img_channels=3, num_classes=100)
37
38      for net_i in range(n_clients):
39          nets[net_i] = net
40
41      return nets
42
43  def train_net(alg, id, net, train_data,
44                test_data, epochs, lr, device="cpu", mu=None,
45                global_net=None):
46      logger.info(
47          'Training Network of Client %s at device: %s'
48          % (id+1, device))
49
50      train_acc = accuracy(net, train_data, device, False)
51      test_acc, conf_matrix = accuracy(net, test_data,
52                                       device,
53  get_confusion_matrix=True)
54
55      logger.info('--- Pre-Training Training Accuracy: {}'
56                  .format(train_acc))
57      logger.info('--- Pre-Training Test Accuracy: {}'
58                  .format(test_acc))
59
60      optimizer = optim.SGD(filter(lambda p: p.requires_grad,
61                                   net.parameters()), lr=lr,
62                            momentum=args.rho,
63                            weight_decay=args.reg)
64
65      criterion = nn.CrossEntropyLoss().to(device)
66
67      if type(train_data) == type([1]):
68          pass
69      else:
70          train_data = [train_data]
71
72      tau = 0
73      for epoch in range(epochs):
74          cnt = 0
75          epoch_loss_collector = []
76          for tmp in train_data:
77              for batch_id, (x, target) in enumerate(tmp):
78                  cnt += 1
79                  x, target = x.to(device), target.to(device)
80                  optimizer.zero_grad()
81                  x.requires_grad = True
82                  target.requires_grad = False
83                  target = target.long()
84
85                  out = net(x)
86                  loss = criterion(out, target)
87
88                  if alg == 'fedprox':
89                      # for fedprox
90                      global_weights = list(
91                          global_net.to(device)
```

```
                        .parameters())
                    fed_prox_reg = 0.0
                    for param_index, param in enumerate(
                            net.parameters()):
                        fed_prox_reg += ((mu / 2) * torch.norm(
                            (param -
                             global_weights[param_index]))**2)
                    loss += fed_prox_reg

                loss.backward()
                optimizer.step()

                tau += 1

                epoch_loss_collector.append(loss.item())
        epoch_loss = sum(
            epoch_loss_collector) / len(
                epoch_loss_collector)
        logger.info('Epoch: %d, Loss: %f' % (
            epoch+1, epoch_loss))

    if alg == 'fednova':
        global_model.to(device)
        global_model_param = global_model.state_dict()
        net_param = net.state_dict()
        a_i = (tau - args.rho * (1 - pow(args.rho, tau)) /
               (1 - args.rho)) / (1 - args.rho)
        # ---FedNova for SGD Momentum ||ai|| = [τ − ρ(1 − ρ^τ )/(1 −
ρ)] /(1 − ρ)
        d_i = copy.deepcopy(global_model.state_dict())
        for param in d_i:
            d_i[param] = torch.true_divide(
                global_model_param[param]-net_param[param], a_i)
            # ---FedNova Normalized Gradients: Δ(i) = −ηG(i)*a(i) =
            # (net_param(t) - net_param(t-1),
            # net_param(t) = global_model_param)

    train_acc = accuracy(net, train_data,
                         get_confusion_matrix=False, device=device)
    test_acc, conf_matrix = accuracy(net,
                                     test_data,
get_confusion_matrix=True, device=device)

    logger.info('----- Training Accuracy: %f' % train_acc)
    # logger.info('----- Test Accuracy: %f' % test_acc)

    net.to(device)
    logger.info('*!Training Completed!*')

    if alg == 'fednova':
        return train_acc, test_acc, conf_matrix, a_i, d_i
    else:
        return train_acc, test_acc, conf_matrix, None, None

def local_train_net(id, selected, global_model,
                    args, dataid_map, device="cpu"):
```

```
148     avg_acc = 0.0
149     dataids = dataid_map[id]
150 
151     net = nets[id]
152     net.to(device)
153 
154     train_dl_local, test_dl_local, _, _ = dataloader(
155         args.dataset, args.datadir,
156         args.batch_size_tr, args.batch_size_test, dataids)
157 
158     trainacc, testacc, conf_matrix, a_i, d_i = train_net(
159         args.alg, id, net,
160         train_dl_local, test_dl_local,
161         args.epochs, args.lr, args.device,
162         args.mu, global_model)
163 
164     avg_acc += testacc
165 
166     avg_acc /= len(selected)
167     nets_list = list(nets.values())
168 
169     if args.alg == 'centralized':
170         return testacc, conf_matrix
171     elif args.alg == 'fednova':
172         return nets_list, a_i, d_i
173     else:
174         return nets_list
175 
176 
177 if __name__ == '__main__':
178     start_time = time.time()
179 
180     args = get_args()
181     n_rounds = args.n_rounds
182     mkdirs(args.logdir)
183     mkdirs(args.modeldir)
184 
185     info = str(args.alg) + ', ' + str(
186         args.dataset) + ', Model=' + str(
187         args.model) + ', Data Distribution=' + str(args.dist)
188 
189     if args.log_file_name is None:
190         path = info
191     else:
192         path = args.log_file_name
193 
194     dir = os.path.join(args.logdir, path)
195     mkdirs(dir)
196 
197     device = torch.device(args.device)
198 
199     # Create Log File
200 
201     if args.log_file_name is None:
202         args.log_file_name = 'log_file--' + info
203     log_path = args.log_file_name+'.log'
```

```
204     logging.basicConfig(
205         filename=os.path.join(dir, log_path),
206         format='%(asctime)s %(levelname)-8s %(message)s',
207         datefmt='%m-%d %H:%M', level=logging.DEBUG, filemode='w')
208 
209     logger = logging.getLogger()
210     logger.setLevel(logging.DEBUG)
211 
212     if args.alg == 'centralized':
213         info = 'CENTRALIZED LEARNING \n   Dataset: ' + str(
214             args.dataset) + '\n   Model: ' + str(
215             args.model) + '\n   Epochs: ' + str(
216             args.epochs)
217         logger.info(info)
218         n_clients = 1
219         print(f"Number of clients: {
220             n_clients} --- Centralized Training")
221     else:
222 
223         logger.info('Algorithm: ' + str(
224             args.alg) + '\nDataset: ' + str(
225             args.dataset) + '\nModel: ' + str(
226             args.model) + '\nData Distribution: ' + str(
227             args.dist) + '\nClients: ' + str(
228             args.n_clients) + '\nClients Sample: ' + str(
229             args.sample) + '\nRounds: ' + str(
230             args.n_rounds) + '\nEpochs: ' + str(
231             args.epochs) + '\nBatch Size Tr: ' + str(
232             args.batch_size_tr) + '\nDevice: ' + args.device)
233         n_clients = args.n_clients
234 
235         logger.info("-" * 50)
236         logger.info("Partitioning " + str(args.dataset) +
237                     " Data For " + str(n_clients) + " Clients")
238         print(f"Number of clients: {n_clients}")
239 
240     y_train, dataid_map = split_data(
241         args.dataset, args.datadir, args.dist, n_clients, args.alpha)
242     n_classes = len(np.unique(y_train))
243     print(f"Number of classes of dataset: {n_classes}")
244 
245     for i in range(n_clients):
246         dataids = dataid_map[i]
247 
248     train_dl_global, test_dl_global, train_ds_global, test_ds_global=
249 dataloader(
250         args.dataset, args.datadir,
251         args.batch_size_tr, args.batch_size_test, None)
252     # global model data => Full datasset => Dataids = None
253 
254     with Spinner():
255 
256         if args.alg == 'centralized':
257             logger.info("Centralized Learning")
258             nets = models_init(1, args)
259             id = 0
```

```
260             selected = np.arange(n_clients)
261             global_model = None
262             test_acc, conf_matrix = local_train_net(
263                 id, selected, global_model,
264                 args, dataid_map, device)
265             logger.info('Model Accuracy: ' + str(test_acc))
266             logger.info("-" * 50)
267 
268         else:
269             logger.info("Initializing Nets")
270             nets = models_init(args.n_clients, args)
271             global_models = models_init(1, args)
272             global_model = global_models[0]
273 
274             global_model.to(device)
275             global_param = global_model.state_dict()
276 
277             for net_id, net in nets.items():
278                 net.load_state_dict(global_param)
279 
280             global_accuracy_per_round = []
281 
282             global_param = global_model.state_dict()
283 
284             a_i_list = [0] * n_clients
285             d_i_list = [copy.deepcopy(global_model.state_dict())
286                         for i in range(n_clients)]
287             d_i_round = copy.deepcopy(global_model.state_dict())
288             for i in range(n_clients):
289                 for param in d_i_list[i]:
290                     d_i_list[i][param] = 0
291             for param in d_i_round:
292                 d_i_round[param] = 0
293 
294             for round in range(args.n_rounds):
295                 logger.info("Communication Round: " + str(round+1))
296 
297                 arr = np.arange(args.n_clients)
298                 np.random.shuffle(arr)
299                 selected = arr[:int(args.n_clients * args.sample)]
300                 if len(selected) < 1:
301                     print('Selected Clients are less than 1')
302                     exit()
303                 logger.info(
304                     "Clients Participating in Round: " + str(
305                         selected))
306                 print(selected)
307 
308                 for id in selected:
309                     nets[id].load_state_dict(global_param)
310 
311                     if args.alg == 'fedavg':
312                         global_model_tr = None
313 
314                     else:
315                         global_model_tr = global_model
```

```
316
317                     if args.alg == 'fednova':
318                         _, a_i_list[id], d_i_list[id] =
319 local_train_net(id, selected,
320 global_model_tr, args, dataid_map, device)
321                     else:
322                         local_train_net(
323                             id, selected, global_model_tr, args,
324                               dataid_map, device)
325
326                 # update global model
327                 updated_global_model = global_model.state_dict()
328                 total_data = sum(
329                     [len(dataid_map[r]) for r in selected])
330                 # m_t ← Σ(k∈St nk) (FedAvg)
331                 p = [len(
332                     dataid_map[r])/total_data for r in selected]
333                 # p ← n(i)/n_total (n(i):data in i)
334
335                 if args.alg == 'fednova':
336                     t_eff = 0.0
337                     for id in range(len(selected)):
338                         nets[selected[id]].to(device)
339                         nets[selected[id]].load_state_dict
340                         net_para = nets[selected[id]].state_dict()
341                         t_eff = t_eff + a_i_list[selected[id]]*p[id]
342
343                     d_i_round =
344 copy.deepcopy(global_model.state_dict())
345                     for param in d_i_round:
346                         d_i_round[param] = 0.0
347
348                     for i in range(len(selected)):
349                         d_para = d_i_list[i]
350                         for param in d_para:
351                             d_i_round[param] += d_para[param] * p[i]
352
353                     print("a_i= ", a_i_list)
354                     print("p= ", p)
355                     print("t_eff= ", t_eff)
356
357                     for param in updated_global_model:
358                         updated_global_model[param] = \
359                             updated_global_model[param] - \
360                             t_eff * d_i_round[param]
361
362 global_model.load_state_dict(updated_global_model)
363
364                 else:
365                     for id in range(len(selected)):
366                         nets[selected[id]].to(device)
367                         nets[selected[id]].load_state_dict
368                         net_param = nets[selected[id]].state_dict()
369                         if id == 0:
370                             for param in net_param:
371                                 global_param[param] = \
```

```
372                                 net_param[param] * p[id]
373                         else:
374                             for param in net_param:
375                                 global_param[param] += \
376                                 net_param[param] * p[id]
377
378                     global_model.load_state_dict(global_param)
379
380                 train_acc = accuracy(global_model,
381                             train_dl_global,
382                             device=device,
383                             get_confusion_matrix=False)
384                 test_acc = accuracy(global_model,
385                             test_dl_global,
386                             device=device,
387                             get_confusion_matrix=False,)
388                 logger.critical(
389                     '-- Global Model Train Accuracy: %f' % train_acc)
390                 logger.critical(
391                     '-- Global Model Test Accuracy: %f' % test_acc)
392
393                 global_accuracy_per_round.append(test_acc)
394                 logger.critical(global_accuracy_per_round)
395
396             logger.critical('Global Number of Training Data: %d' %
397                             len(train_ds_global))
398             logger.critical('Global Number of Testing Data: %d' %
399                             len(test_ds_global))
400             logger.critical(global_accuracy_per_round)
401
402             train_acc = accuracy(global_model,
403                         train_dl_global,
404                         device=device,
405                         get_confusion_matrix=False)
406             test_acc, conf_matrix = accuracy(global_model,
407                                 test_dl_global,
408                                 device=device,
409                                 get_confusion_matrix=True)
410
411             logger.info('--Global Model Tr Accuracy:%f'% train_acc)
412             logger.info('--Global Model Test Accuracy: %f'%test_acc)
413             logger.critical("%s seconds"%(time.time()-start_time))
414             logger.info("-" * 50)
415             plot(args.alg, global_accuracy_per_round, args.model,
416                  args.dataset, args.n_rounds, args.dist)
417
418     # Confusion Matrix of Global Model
419     fig1, ax = plot_confusion_matrix(conf_matrix)
420     fig1.suptitle('Confusion Matrix of Global Model')
421     fig1.savefig(dir + '/conf_matrix.png')
422     # plt.show()
```

utils.py

```python
import os
import torch
import argparse
import sys
import time
import threading
import numpy as np

from sklearn.metrics import confusion_matrix


def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('--model', type=str, default='simple_cnn',
        help='neural network used in training -- options:
simple_cnn/resnet_18 (default: simple_cnn)')
    parser.add_argument('--dataset', type=str, default='cifar10',
        help='dataset used for training -- options:
mnist/cifar10/cifar100 (default: cifar10)')
    parser.add_argument('--epochs', type=int, default=10,
        help='number of local epochs')
    parser.add_argument('--n_clients', type=int, default=15,
        help='number of clients in FL app (default: 15)')
    parser.add_argument('--alg', type=str, default='fedavg',
        help='fl algorithm -- options: fedavg/fedprox')
    parser.add_argument('--n_rounds', type=int, default=100,
        help='number of maximum communication rounds')
    parser.add_argument('--dist', type=str, default='iid',
        help='the data partitioning strategy -- options:
iid/iid_dif_q/non_iid (default: iid)')
    parser.add_argument('--batch_size_tr', type=int, default=32,
        help='input batch size for training (default: 32)')
    parser.add_argument('--batch_size_test', type=int, default=32,
        help='input batch size for testing (default: 32)')
    parser.add_argument('--lr', type=float, default=0.01,
        help='learning rate (default: 0.1)')
    parser.add_argument('--datadir', type=str, required=False,
default="./data/",
        help="Data directory")
    parser.add_argument('--reg', type=float, default=1e-5,
        help="L2 regularization strength")
    parser.add_argument('--logdir', type=str, required=False,
default="./logs/",
        help='Log directory path')
    parser.add_argument('--modeldir', type=str, required=False,
default="./models/",
        help='Model directory path')
    parser.add_argument('--alpha', type=float, default=0.5,
        help='Parameter for the dirichlet distribution for data
partitioning')
    parser.add_argument('--device', type=str, default='cuda:0',
        help='The device to run the program')
```

```
53      parser.add_argument('--log_file_name', type=str, default=None,
54          help='The log file name')
55      parser.add_argument('--mu', type=float, default=0.01,
56          help='The mu parameter for fedprox')
57      parser.add_argument('--rho', type=float, default=0,
58          help='Parameter controlling the momentum SGD')
59      parser.add_argument('--sample', type=float, default=0.7,
60          help='Sample ratio for each communication round')
61      args = parser.parse_args()
62      return args
63
64
65  def mkdirs(dirpath):
66      try:
67          os.makedirs(dirpath)
68      except Exception as _:
69          pass
70
71  def accuracy(model, data, device, get_confusion_matrix):
72
73      true_labels, pred_labels = np.array([]), np.array([])
74
75      if type(data) == type([1]):
76          pass
77      else:
78          data = [data]
79
80      total = 0
81      correct = 0
82
83      with torch.no_grad():
84           for item in data:
85              for _, (x, target) in enumerate(item):
86                  x, target = x.to(device), target.to(device)
87                  out = model(x)
88                  _, pred_label = torch.max(out.data, 1)
89
90                  total += x.data.size()[0]
91                  correct += (pred_label == target.data).sum().item()
92
93                  if device == "cpu":
94                      pred_labels = np.append(
95                          pred_labels, pred_label.numpy())
96                      true_labels = np.append(
97                          true_labels, target.data.numpy())
98                  else:
99                      pred_labels = np.append(
100                         pred_labels, pred_label.cpu().numpy())
101                     true_labels = np.append(
102                         true_labels, target.data.cpu().numpy())
103
104     if get_confusion_matrix:
105         conf_matrix = confusion_matrix(true_labels, pred_labels)
106         return correct/float(total), conf_matrix
107     return correct/float(total)
108
```

```
class Spinner:
    busy = False
    delay = 0.1

    @staticmethod
    def spinning_cursor():
        while 1:
            for cursor in '|/-\\': yield cursor

    def __init__(self, delay=None):
        self.spinner_generator = self.spinning_cursor()
        if delay and float(delay): self.delay = delay

    def spinner_task(self):
        while self.busy:
            sys.stdout.write(next(self.spinner_generator))
            sys.stdout.flush()
            time.sleep(self.delay)
            sys.stdout.write('\b')
            sys.stdout.flush()

    def __enter__(self):
        self.busy = True
        threading.Thread(target=self.spinner_task).start()

    def __exit__(self, exception, value, tb):
        self.busy = False
        time.sleep(self.delay)
        if exception is not None:
            return False

def get_client_class_summary(train_label, client_id):
    class_sum = {}
    data_sum = {}

    for client, data_id in client_id.items():
        unq, unq_cnt = np.unique(
          train_label[data_id], return_counts=True)
        tmp = {int(unq[i]): int(unq_cnt[i]) for i in range(len(unq))}
        class_sum[client] = tmp
        data_sum[client] = len(data_id)
    return class_sum, data_sum
```

models.py

```
# Simple CNN -> Simple Convolutional Neural based on LeNet-5
(https://ieeexplore.ieee.org/document/726791)
# --> Accuracy  MNIST =~ 98%, CIFAR10  =~ 64%, CIFAR100 =~ 29%

# ResNet -> ResNet18 Residual Neural Network
(https://arxiv.org/abs/1512.03385)
# --> Accuracy: MNIST =~ 99%, CIFAR10  =~ 76%, CIFAR100 =~ 44%

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch import Tensor
from typing import Type

class SimpleCNN(nn.Module):
    def __init__(self, img_channels: int = 3, num_classes: int = 10,
fc1=None):
        super(SimpleCNN, self).__init__()
        # Conv2d(input image channel (3 channels
        # (CIFAR - R G B) or 1 (MNIST - Grayscale),
        # 6 output channels, 5x5 square convolution kernel)
        #  -> [28X28x6]
        self.conv1 = nn.Conv2d(img_channels, 6, 5)
        # 2x2 maxpooling (kernel_size=2, stride=2) -> [14x14x6]
        self.pool = nn.MaxPool2d(2, 2)
        # Conv2d(6 input, 16 output, 5x5 conv kernel) -> [10x10x16]
        self.conv2 = nn.Conv2d(6, 16, 5)

        # Linear layers y = Wx + b
        # (fully connected layers to 10 output classes)

        if fc1 == None:
            self.fc1 = nn.Linear(16 * 5 * 5, 120)
        else:
            self.fc1 = fc1
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, num_classes)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)  # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


class BasicBlock(nn.Module):
    def __init__(
            self,
            in_channels: int,
            out_channels: int,
            stride: int = 1,
            expansion: int = 1,
```

```
56              downsample: nn.Module = None) -> None:
57          super(BasicBlock, self).__init__()
58
59          self.expansion = expansion
60          self.downsample = downsample
61
62          self.conv1 = nn.Conv2d(
63              in_channels,
64              out_channels,
65              kernel_size=3,
66              stride=stride,
67              padding=1,
68              bias=False
69          )
70          self.bn1 = nn.BatchNorm2d(out_channels)
71          self.relu = nn.ReLU(inplace=True)
72          self.conv2 = nn.Conv2d(
73              out_channels,
74              out_channels*self.expansion,
75              kernel_size=3,
76              padding=1,
77              bias=False
78          )
79          self.bn2 = nn.BatchNorm2d(out_channels*self.expansion)
80
81      def forward(self, x):
82          identity = x
83          out = self.conv1(x)
84          out = self.bn1(out)
85          out = self.relu(out)
86          out = self.conv2(out)
87          out = self.bn2(out)
88          if self.downsample is not None:
89              identity = self.downsample(x)
90          out += identity
91          out = self.relu(out)
92          return out
93
94  class ResNet(nn.Module):
95      def __init__(
96              self,
97              img_channels: int,
98              num_classes: int,
99              num_layers: int = 18,
100             block=BasicBlock) -> None:
101         super(ResNet, self).__init__()
102         if num_layers == 18:
103             # 2 Blocks per layer (4 layers = 2, 2, 2, 2 blocks)
104             layers = [2, 2, 2, 2]
105             self.expansion = 1
106
107         self.in_channels = 64
108         self.conv1 = nn.Conv2d(
109             in_channels=img_channels,
110             out_channels=self.in_channels,
111             kernel_size=7,
```

```
            stride=2,
            padding=3,
            bias=False
        )
        self.bn1 = nn.BatchNorm2d(self.in_channels)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2,
padding=1)

        self.layer1 = self._make_layer(
            block, 64, layers[0])
        self.layer2 = self._make_layer(
            block, 128, layers[1], stride=2)
        self.layer3 = self._make_layer(
            block, 256, layers[2], stride=2)
        self.layer4 = self._make_layer(
            block, 512, layers[3], stride=2)

        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(512*self.expansion, num_classes)

    def _make_layer(
        self,
        block: Type[BasicBlock],
        out_channels: int,
        blocks: int,
        stride: int = 1
    ) -> nn.Sequential:
        downsample = None
        if stride != 1:
            downsample = nn.Sequential(
                nn.Conv2d(
                    self.in_channels,
                    out_channels*self.expansion,
                    kernel_size=1,
                    stride=stride,
                    bias=False
                ),
                nn.BatchNorm2d(out_channels * self.expansion),
            )
        layers = []
        layers.append(
            block(
                self.in_channels,
                out_channels,
                stride,
                self.expansion,
                downsample
            )
        )
        self.in_channels = out_channels * self.expansion
        for i in range(1, blocks):
            layers.append(block(
                self.in_channels,
                out_channels,
                expansion=self.expansion
```

```
168             ))
169         return nn.Sequential(*layers)
170 
171     def forward(self, x: Tensor) -> Tensor:
172         x = self.conv1(x)
173         x = self.bn1(x)
174         x = self.relu(x)
175         x = self.maxpool(x)
176         x = self.layer1(x)
177         x = self.layer2(x)
178         x = self.layer3(x)
179         x = self.layer4(x)
180         x = self.avgpool(x)
181         x = torch.flatten(x, 1)
182         x = self.fc(x)
183         return x
184 
185 if __name__ == '__main__':
186     # torch.save(SimpleCNN, 'models/simple_model.pth')
187     SimpleCNN = SimpleCNN(3, 10)
188     print("SimpleCNN (
189           e.g.: 3 imput channels and 10 outputs classes):", SimpleCNN)
190     ResNet18 = ResNet(3, 18, BasicBlock, 10)
191     print("ResNet18 (
192           e.g.: 3 imput channels and 10 outputs classes):", ResNet18)
```

data_splitter.py

```python
import numpy as np
import logging
from datasets import *
from utils import *
from torchvision import transforms
from torch.utils.data import DataLoader
from datasets import CIFAR10_DATASET, MNIST_DATASET, CIFAR100_DATASET

def split_data(dataset, datadir, dist, n_clients, alpha):

    if dataset == 'mnist':
        datadir = datadir + "/mnist/"
        train_labels = load_mnist_data(datadir)
        n_train = len(train_labels[0])
        train_labels = train_labels[0]
    elif dataset == 'cifar10':
        datadir = datadir + "/cifar10/"
        train_labels = load_cifar10_data(datadir)
        n_train = len(train_labels)
    elif dataset == 'cifar100':
        datadir = datadir + "/cifar100/"
        train_labels = load_cifar100_data(datadir)
        n_train = len(train_labels)

    print(f"Total number of training data: {n_train}")

    if dist == "iid":  # homo
        ids = np.random.permutation(n_train)
        # randomly shuffle n_train
        batch_ids = np.array_split(ids, n_clients)
        dataids_map = {i: batch_ids[i] for i in range(n_clients)}

    elif dist == "iid_dif_q":
        # diferrent quantity in each cliennt
        ids = np.random.permutation(n_train)
        min_size = 0
        K = 10
        # K: num_of_classes
        if dataset == 'cifar100':
            K = 100
        while min_size < K:
            props = np.random.dirichlet(np.repeat(alpha, n_clients))
            props = props/props.sum()
            min_size = np.min(props*len(ids))
        props = (np.cumsum(props)*len(ids)).astype(int)[:-1]
        batch_ids = np.split(ids, props)
        dataids_map = {i: batch_ids[i] for i in range(n_clients)}

    elif dist == "non_iid":
        min_size = 0
        min_require_size = 1
        # minimum number of data classes in each client
        K = 10
        # K: num_of_classes
        if dataset == 'cifar100':
```

```
            K = 100

        N = n_train
        dataids_map = {}

        while min_size < min_require_size:
            id_batch = [[] for _ in range(n_clients)]
            for k in range(K):
                id_k = np.where(train_labels == k)[0]
                np.random.shuffle(id_k)
                props = np.random.dirichlet(
                    np.repeat(alpha, n_clients))
                # Balance
                props = np.array(
                    [p * (len(idx_i) < N / n_clients)
                        for p, idx_i in zip(props, id_batch)])
                props = props / props.sum()
                props = (
                    np.cumsum(props) * len(id_k)).astype(int)[:-1]
                id_batch = [id_i + id.tolist()
                    for id_i, id in zip(id_batch, np.split(id_k,
props))]
                min_size = min([len(id_i) for id_i in id_batch])

        for i in range(n_clients):
            np.random.shuffle(id_batch[i])
            dataids_map[i] = id_batch[i]

    class_sum, data_sum = get_client_class_summary(
        train_labels, dataids_map)
    logger = logging.getLogger()
    logger.info('Class Sum per Client: ' + str(class_sum) +
                '\nNumber of Data per Client: ' + str(data_sum))
    print(
        f"Class Sum per Client: {class_sum}\nNumber of Data per
Client: {data_sum}")

    return (train_labels, dataids_map)

def dataloader(dataset, datadir, train_bs, test_bs, dataids=None):

    if dataset == 'mnist':
        dl_obj = MNIST_DATASET
        transform = transforms.Compose(
            [transforms.ToTensor()])
    elif dataset == 'cifar10':
        dl_obj = CIFAR10_DATASET
        transform = transforms.Compose(
            [transforms.ToTensor(),
             transforms.Normalize(mean=[0.507, 0.486, 0.440],
                                  std=[0.267, 0.256, 0.276])])
    elif dataset == 'cifar100':
        dl_obj = CIFAR100_DATASET
        transform = transforms.Compose(
            [transforms.ToTensor(),
             transforms.Normalize(mean=[0.507, 0.486, 0.440],
```

```
112                                     std=[0.267, 0.256, 0.276])])
113
114     train_ds = dl_obj(datadir,
115                       dataids,
116                       transform=transform,
117                       train=True,
118                       download=True)
119
120     test_ds = dl_obj(datadir,
121                      train=False,
122                      transform=transform,
123                      download=True)
124
125     train_dl = DataLoader(dataset=train_ds,
126                           batch_size=train_bs,
127                           shuffle=True,
128                           drop_last=False)
129
130     test_dl = DataLoader(dataset=test_ds,
131                          batch_size=test_bs,
132                          shuffle=False,
133                          drop_last=False)
134
135   return train_dl, test_dl, train_ds, test_ds
```

datasets.py

```
import argparse
import numpy as np
from torchvision import datasets
from torch.utils.data import Dataset
from PIL import Image

def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('--dataset',
                        type=str, required=False,
                        default="cifar10",
                        help="Dataset (mnist, cifar10, cifar100)")
    args = parser.parse_args()
    return args

# Only training data is needed to make partitions
def load_mnist_data(datadir):
    mnist_train = datasets.MNIST(datadir, train=True, download=True)
    mnist_test = datasets.MNIST(datadir, train=False, download=True)
    train_data = mnist_train.data
    train_labels = np.array(mnist_train.targets)
    test_labels = np.array(mnist_test.targets)

    return (train_labels, test_labels)

def load_cifar10_data(datadir):
    cifar10_train = datasets.CIFAR10(datadir, train=True,
download=True)

    train_labels = np.array(cifar10_train.targets)

    # y_train = np.array(y_train)
    # return (cifar10_train)
    return (train_labels)

def load_cifar100_data(datadir):

    cifar100_train = datasets.CIFAR100(datadir, train=True,
download=True)

    train_labels = np.array(cifar100_train.targets)
    # print(train_labels)

    return (train_labels)


if __name__ == '__main__':

    args = get_args()

    if args.dataset == "mnist":
        datadir = "./data/mnist/"
        load_mnist_data(datadir)
    elif args.dataset == "cifar10":
        datadir = "./data/cifar10/"
```

```
56          load_cifar10_data(datadir)
57      else:
58          datadir = "./data/cifar100/"
59          load_cifar100_data(datadir)
60
61      print(f'Training Dataset Downloaded {args.dataset}')
62
63
64  class MNIST_DATASET(Dataset):
65
66      def __init__(self, root, dataids=None, train=True, transform=None,
67  download=False):
68          self.root = root + "/mnist/"
69          self.dataids = dataids
70          self.train = train
71          self.download = download
72          self.transform = transform
73          self.data, self.target = self.__build_dataset__()
74
75      def __build_dataset__(self):
76          cifar10_dataobj = datasets.MNIST(self.root, self.train,
77  self.download)
78          data = cifar10_dataobj.data
79          target = np.array(cifar10_dataobj.targets)
80          if self.dataids is not None:
81              data = data[self.dataids]
82              target = target[self.dataids]
83          return data, target
84
85      def __getitem__(self, index):
86          data, target = self.data[index], self.target[index]
87          # doing this so that it is consistent with all other datasets
88          # to return a PIL Image
89          data = Image.fromarray(data.numpy(), mode='L')
90          if self.transform is not None:
91              data = self.transform(data)
92          return data, target
93
94      def __len__(self):
95          return len(self.data)
96
97  class CIFAR10_DATASET(Dataset):
98
99      def __init__(self, root, dataids=None, train=True, transform=None,
100 download=False):
101
102         self.root = root + "/cifar10/"
103         self.dataids = dataids
104         self.train = train
105         self.download = download
106         self.transform = transform
107         self.data, self.target = self.__build_dataset__()
108
109     def __build_dataset__(self):
110         cifar10_dataobj = datasets.CIFAR10(
111             self.root, self.train, self.download)
```

```
112         data = cifar10_dataobj.data
113         target = np.array(cifar10_dataobj.targets)
114         if self.dataids is not None:
115             data = data[self.dataids]
116             target = target[self.dataids]
117         return data, target
118 
119     def __getitem__(self, index):
120         data, target = self.data[index], self.target[index]
121         # print("cifar10 img:", data)
122         # print("cifar10 target:", target)
123         if self.transform is not None:
124             data = self.transform(data)
125         return data, target
126 
127     def __len__(self):
128         return len(self.data)
129 
130 class CIFAR100_DATASET(Dataset):
131 
132     def __init__(self, root, dataids=None, train=True, transform=None,
133 download=False):
134 
135         self.root = root + "/cifar100/"
136         self.dataids = dataids
137         self.train = train
138         self.download = download
139         self.transform = transform
140 
141         self.data, self.target = self.__build_dataset__()
142 
143     def __build_dataset__(self):
144 
145         cifar10_dataobj = datasets.CIFAR100(
146             self.root, self.train, self.download)
147 
148         data = cifar10_dataobj.data
149         target = np.array(cifar10_dataobj.targets)
150 
151         if self.dataids is not None:
152             data = data[self.dataids]
153             target = target[self.dataids]
154 
155         return data, target
156 
157     def __getitem__(self, index):
158 
159         data, target = self.data[index], self.target[index]
160 
161         if self.transform is not None:
162             data = self.transform(data)
163 
164         return data, target
165 
166     def __len__(self):
167         return len(self.data)
```

plot.py

```
import matplotlib.pyplot as plt
import argparse
from utils import mkdirs


def plot(alg, global_accuracy_per_round, model, dataset, n_rounds,
dist):
    plot_name = str(alg) + ', ' + str(model) + ', ' + ' Dataset=' + \
        str(dataset) + ', ' + ' Dist=' + str(dist)  # 'plot name'

    dir = './plots/' + str(plot_name)
    mkdirs(dir)

    filepath = (dir + '/' + str(plot_name)+'.txt')
    f = open(str(filepath), 'w')

    x1 = range(1, n_rounds+1)
    y1 = global_accuracy_per_round
    y1 = [item * 100 for item in y1]

    line_avg = sum(y1)/len(y1)
    max_value = max(y1)
    max_epoch = y1.index(max(y1))
    line_avg_list = [line_avg] * len(y1)
    var = variance(y1, line_avg_list)

    # After 15 epochs
    y_st = y1[15:len(y1)]
    line_avg_15 = sum(y_st)/len(y_st)
    max_value_15 = max(y_st)
    max_epoch_15 = y_st.index(max(y_st))
    line_avg_list_15 = [line_avg_15] * len(y_st)
    var_15 = variance(y_st, line_avg_list_15)

    f.write("Max Value: " + str(max_value) +
            " at epoch: " + str(max_epoch) + "\n")
    f.write("Average Value: " + str(line_avg) + "\n")
    f.write("Variance: " + str(var) + "\n \n")

    f.write("After 15 Epochs (16 ~ end): " + "\n")

    f.write("Max Value: " + str(max_value_15) +
            " at epoch: " + str(max_epoch_15) + "\n")
    f.write("Average Value: " + str(line_avg_15) + "\n")
    f.write("Variance: " + str(var_15))
    f.close()

    # plotting the line 1 points
    plt.plot(x1, y1, label=alg)

    plt.xlabel('Rounds')
    plt.ylabel('Accuracy %')
    plt.title(str(plot_name))
    plt.legend()
```

```
    plt.savefig(dir + '/' + str(plot_name)+'.png')
    plt.show()

def variance(y, avg_list):
    dif = [a - b for a, b in zip(y, avg_list)]  # y[i]-avg_list[i]]
    list = [x**2 for x in dif]
    var = (sum(list)/(len(y)-1))
    return var

def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('--model', type=str, default='simple_cnn',
        help='model used in training - options:simple_cnn/resnet_18')
    parser.add_argument('--dataset', type=str, default='cifar10',
        help='dataset for training - options:
mnist/cifar10/cifar100')
    parser.add_argument('--alg', type=str, default='fedavg',
        help='fl algorithm - options: fedavg/fedprox')
    parser.add_argument('--n_rounds', type=int, default=100,
        help='number of communication rounds')
    parser.add_argument('--dist', type=str, default='iid',
        help='data partitioning - options: iid/iid_dif_q/non_iid')
    args = parser.parse_args()
    return args


if __name__ == '__main__':
    args = get_args()
    if args.alg == 'fedavg':
        alg = 'FedAvg'
    elif args.alg == 'fedprox':
        alg = 'FedProx'
    elif args.alg == 'fednova':
        alg = 'FedNova'

    if args.model == 'simple_cnn':
        model = 'Simple CNN'
    elif args.model == 'resnet_18':
        model = 'ResNet 18'

    if args.dataset == 'mnist':
        dataset = 'MNIST'
    elif args.dataset == 'cifar10':
        dataset = 'CIFAR10'
    elif args.dataset == 'cifar100':
        dataset = 'CIFAR100'

    if args.dist == 'iid':
        dist = 'I.I.D. Data'
    elif args.dist == 'iid_dif_q':
        dist = 'I.I.D. Dif. Quantity'
    elif args.dist == 'non_iid':
        dist = 'Non-I.I.D. Data'

    plot(alg, None, model, dataset, 100, dist)
```