NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF COMPUTER SCIENCE

Optimization of Distributed Learning Methods

DIPLOMA THESIS

of

TAGARAKIS A. KONSTANTINOS

Supervisor: Panayiotis Tsanakas, Professor of NTUA

Athens, March 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Βελτιστοποίηση Μεθόδων Κατανεμημένης Μάθησης

Διπλωματικη Εργασια

του

ΤΑΓΑΡΑΚΗ Α. ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων: Παναγιώτης Τσανάκας, Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF COMPUTER SCIENCE

Optimization of Distributed Learning Methods

DIPLOMA THESIS

of

TAGARAKIS A. KONSTANTINOS

Supervisor: Panayiotis Tsanakas, Professor of NTUA

Approved by the examination committee on 27th March 2023.

(Signature)	*(Signature)*	*(Signature)*
. . . .	. . . .	. . . .
Panayiotis Tsanakas	George Matsopoulos	Andreas-Georgios Stafylopatis
Professor of NTUA	Professor of NTUA	Professor of NTUA

Athens, March 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Βελτιστοποίηση Μεθόδων Κατανεμημένης Μάθησης

Διπλωματικη Εργασια

του

ΤΑΓΑΡΑΚΗ Α. ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων: Παναγιώτης Τσανάκας, Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Μαρτίου 2023.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
....	....	....
Παναγιώτης Τσανάκας	Γεώργιος Ματσόπουλος	Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

. . . .
Κωνσταντίνος Ταγαράκης,
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

27η Μαρτίου 2023

Περίληψη

Είναι γνωστό ότι η πρόσφατη πρόοδος στον τομέα της μηχανικής μάθησης οφείλεται στον μεγάλο όγκο των διαθέσιμων δεδομένων. Ωστόσο, τα τελευταία χρόνια οι αυξανόμενες ανησυχίες για την προστασία των προσωπικών δεδομένων ώθησαν τις αρχές να περιορίσουν τη χρήση και τη διαβίβαση των δεδομένων που παράγονται από φυσικά πρόσωπα. Ταυτόχρονα, η μαζική ροή δεδομένων που παράγεται από το Διαδίκτυο των Πραγμάτων (Internet of Things-IoT) μπορεί να οδηγήσει σε υπερφόρτωση του δικτύου και σε αυξημένες ανάγκες για αποθηκευτική ικανότητα και υπολογιστική ισχύ. Καθώς η παραδοσιακή προσέγγιση της κεντρικοποιημένης μηχανικής μάθησης καταρρέει υπό αυτές τις συνθήκες, αναδύεται το μοντέλο της ομοσπονδιακής μηχανικής μάθησης (Federated Learning). Η κεντρική ιδέα της ομοσπονδιακής μάθησης είναι να μεταφερθεί η διαδικασία εκπαίδευσης στις τερματικές συσκευές των χρηστών, να εκπαιδευτούν πολλαπλά τοπικά μοντέλα και να παραχθεί ένα συνολικό μοντέλο από τον συνδυασμό τους. Η προσέγγιση αυτή αντιμετωπίζει τις παραπάνω προκλήσεις, αλλά νέες αναδύονται. Μια σημαντική πρόκληση που προκύπτει είναι η μεροληψία (bias) που εισάγουν οι τερματικές συσκευές η οποία οδηγεί σε ένα μη βέλτιστο μοντέλο σε σύγκριση με την παραδοσιακή κεντρικοποιημένη προσέγγιση (non identically and independently distributed data - non-iidness). Σκοπός της παρούσας διπλωματικής εργασίας είναι η διερεύνηση των βασικών αρχών του μοντέλου της ομοσπονδιακής μάθησης και της προαναφερθείσας μεγάλης πρόκλησης που το συνοδεύει. Τέλος, προτείνουμε έναν αλγόριθμο που μετριάζει αυτό το πρόβλημα και τον αξιολογούμε πραγματοποιώντας πειράματα σε ένα προσομοιωμένο περιβάλλον.

Λέξεις Κλειδιά

Μηχανική Μάθηση, Κατανεμημένη Μάθηση, Non-IIDness, Προσομοίωση, Βελτιστοποίηση.

Abstract

From medical imaging to predictive analytics, more and more applications nowadays are based on Artificial Intelligence-Machine Learning (AI/ML), to address complex problems. Although powerful, AI/ML techniques require enormous amounts of data in order to be trained. During the last decade, increasing privacy concerns require the authorities to restrict the use and transfer of data generated by individuals. At the same time, the massive stream of data produced and transmitted by the IoT (Internet of Things) devices can lead to network overload, increased demand for storage capacity and computational power. As the traditional approach of centralized machine learning (CL) struggles under these circumstances, the paradigm of Federated Learning (FL) emerges as an alternative. Unlike CL where the data processing task (training) occurs in a centralized entity (e.g. cloud server), in FL it is offloaded to the client devices (e.g. smartphones) and the central entity is only responsible to aggregate the produced local models. This approach tackles the above challenges but new ones come together. One major challenge is the bias that the client introduces which leads to a suboptimal model compared to a centralized approach. The aim of this thesis is to investigate the problem of bias in FL environments focusing on its impact on the model performance. On top, we present FedLoss, a novel bias mitigation algorithm, which is benchmarked in our dedicated Federated Learning simulation environment.

Keywords

Machine Learning, Federated Learning, Non-IIDness, Simulation, Optimization.

Ευχαριστίες

Αρχικά, ευχαριστώ τον καθηγητή μου, Παναγιώτη Τσανάκα, που μου έδωσε τη δυνατότητα να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα στα πλαίσια της διπλωματικής μου εργασίας. Έχοντας ολοκληρώσει αυτήν την προσπάθεια, δεν μπορώ παρά να ευχαριστήσω θερμά και τον υποψήφιο Διδάκτορα Γιώργο Δραϊνάκη για όλη την καθοδήγηση, την έμπρακτη βοήθεια και την ηθική υποστήριξη που μου προσέφερε σε κάθε στάδιο αυτής της εργασίας. Ένα μεγάλο ευχαριστώ οφείλω να πω και στην οικογένειά μου, σε όλους τους κοντινούς μου ανθρώπους, τους φίλους και συμφοιτητές που με στήριξαν, με τον έναν ή τον άλλον τρόπο, όλα αυτά τα χρόνια και που έδωσε ο καθένας λίγο από το χρώμα του στη ζωή μου σε αυτά τα κοπιαστικά χρόνια των σπουδών και όχι μόνο.

Αθήνα, Μάρτιος 2023
Κωστής Ταγαράκης

Contents

List of Figures

List of Tables

Εκτεταμένη Ελληνική Περίληψη

Κατά την τελευταία δεκαετία, η αφθονία των δεδομένων διαδραματίζει καθοριστικό ρόλο στην ταχεία ανάπτυξη εφαρμογών που βασίζονται στην Τεχνητή Νοημοσύνη-Μηχανική Μάθηση. Η Μηχανική Μάθηση περιλαμβάνει μια ευρεία γκάμα τεχνικών και αλγορίθμων που βασίζονται στα δεδομένα και επιτρέπουν στα συστήματα υπολογιστών να μαθαίνουν από την εμπειρία του παρελθόντος. Τα συστήματα αυτά τροφοδοτούνται με (μεγάλες ποσότητες) δεδομένων για την παραγωγή (εκπαίδευση) μοντέλων Μηχανικής Μάθησης, προκειμένου να κατανοήσουν πολύπλοκα πρότυπα και, ως εκ τούτου, να επιλύσουν προβλήματα που δεν μπορούν να αντιμετωπιστούν με αναλυτικές μεθόδους. Οι σχετικές εφαρμογές περιλαμβάνουν την ανάλυση βίντεο, την πρόβλεψη του καιρού, την πρόβλεψη της κατανάλωσης ενέργειας κ.λπ.

Τα δεδομένα παράγονται από ένα ευρύ φάσμα πηγών, συμπεριλαμβανομένων των μέσων κοινωνικής δικτύωσης, των κινητών τηλεφώνων, των συσκευών του Διαδικτύου των Πραγμάτων και των παραδοσιακών επιχειρησιακών συστημάτων. Παραδοσιακά, τα δεδομένα συλλέγονται και επεξεργάζονται (εκπαιδεύονται) κεντρικά σε ένα μόνο μηχάνημα ή σε ένα σύμπλεγμα μηχανημάτων, π.χ. σε ένα διακομιστή νέφους (Centralized Machine Learning - CL). Ωστόσο, από αυτή την κεντρικοποιημένη προσέγγιση προκύπτουν διάφορες προκλήσεις. Δεδομένου ότι τα δεδομένα μπορεί να περιέχουν ευαίσθητες πληροφορίες π.χ. ιατρικά αρχεία, τοποθεσία χρήστη, συνήθειες κ.λπ., η μεταφόρτωση δεδομένων σε μια κεντρική ή δημόσια οντότητα δεν είναι πάντα βιώσιμη. Από το 2018 και μετά, επιβάλλονται πρόσθετοι περιορισμοί από νέες νομοθετικές πράξεις για να καταστεί δυνατή η προστασία των δεδομένων και της ιδιωτικής ζωής. Το πιο αντιπροσωπευτικό παράδειγμα είναι ο γενικός κανονισμός για την προστασία των δεδομένων (ΓΚΠΔ) [3] της Ευρωπαϊκής Ένωσης. Εκτός από την προστασία της ιδιωτικής ζωής, η μεταφορά τεράστιου όγκου δεδομένων μπορεί να οδηγήσει σε υπερφόρτωση του δικτύου και αυξημένη ζήτηση για υποδομές αποθήκευσης.

Για την αντιμετώπιση των προαναφερθεισών προκλήσεων, προέκυψε η έννοια της ομοσπονδιακής μάθησης (Federated Learning). Η ομοσπονδιακή μάθηση παρουσιάστηκε για πρώτη φορά το 2016 [4] και έκτοτε έχει τραβήξει την προσοχή της επιστημονικής κοινότητας. Η βασική ιδέα πίσω από την ομοσπονδιακή μάθηση είναι η εκπαίδευση μοντέλων μηχανικής μάθησης σε αποκεντρωμένες πηγές δεδομένων, όπως κινητές συσκευές, έτσι ώστε τα δεδομένα να μην εκτίθενται στον κεντρικό διακομιστή. Ως εκ τούτου, οι συσκευές μεταφορτώνουν ένα μοντέλο μηχανικής μάθησης από τον κεντρικό διακομιστή, εκπαιδεύουν το μοντέλο τοπικά χρησιμοποιώντας τα δικά τους δεδομένα και στη συνέχεια μεταφορτώνουν το ενημερωμένο (τοπικό) μοντέλο πίσω στον διακομιστή. Στη συνέχεια, ο διακομιστής συγκεντρώνει τα τοπικά μοντέλα, δημιουργώντας ένα νέο (παγκόσμιο) μοντέλο μηχανικής μάθησης. Η διαδικασία επαναλαμβάνεται για αρκετούς κύκλους (γύρους), έως ότου επιτευχθεί σύγκλιση του μοντέλου. Το βασικό πλεονέκτημα της ομοσπονδιακής μάθησης είναι ότι τα δεδομένα παραμένουν στις συσκευές-πελάτες ανά πάσα στιγμή, επομένως διασφαλίζεται η ιδιωτικότητα. Αυτό σημαίνει ότι προσφέρει υψηλό επίπεδο προστασίας της ιδιωτικής ζωής, διασφαλίζοντας ότι τα δεδομένα παραμένουν τοπικά και δεν κοινοποιούνται σε τρίτους. Η προσέγγιση αυτή καθησυχάζει τα άτομα και τους οργανισμούς που διστάζουν να μοιραστούν τα δεδομένα τους με άλλους για λόγους ασφαλείας και προστασίας των προσωπικών τους δεδομένων. Επιπλέον, αυτό καθιστά πιο δύσκολο για τους χάκερ να αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα. Επίσης, μειώνεται το κόστος επικοινωνίας, δεδομένου ότι τα μοντέλα μηχανικής μάθησης είναι γενικά ελαφριά από άποψη μεγέθους δεδομένων, σε σύγκριση με τα πραγματικά δεδομένα.

Δυστυχώς, η ομοσπονδιακή μάθηση συνοδεύεται και από ορισμένα μειονεκτήματα. Το πρώτο είναι ότι βασίζεται στην διαθεσιμότητα των δεδομένων στις συσκευές των πελατών. Η μη διαθεσιμότητα δεδομένων μπορεί να προκαλέσει σοβαρή υποβάθμιση της απόδοσης του μοντέλου μηχανικής μάθησης. Επιπλέον, η μετάδοση/λήψη μοντέλων μεταξύ συσκευών-πελατών και ενός κεντρικού διακομιστή δημιουργεί πιθανούς κινδύνους ασφαλείας, όπως η πιθανότητα υποκλοπής των ενημερώσεων από έναν κακόβουλο παράγοντα ή η εισαγωγή πλαστών δεδομένων στη διαδικασία εκπαίδευσης. Από τη σκοπιά της μηχανικής λογισμικού, η ομοσπονδιακή μάθηση περιλαμβάνει πολλαπλές συσκευές που συνεργάζονται για την εκπαίδευση ενός μοντέλου. Σε τέτοια κλιμακωτά συστήματα, η αυξημένη πολυπλοκότητα δυσχεραίνει τη συντήρηση του λογισμικού, την αποσφαλμάτωση κ.λπ. Ωστόσο, αυτό που αναγνωρίζεται ως η αχίλλειος πτέρνα της ομοσπονδιακής μάθησης είναι η ετερογένεια των δεδομένων. Η μεροληψία εισάγεται στα δεδομένα των πελατών, λόγω των τάσεων, των παραλλαγών, των προτύπων, των συνηθειών κ.λπ. των χρηστών. Ως εκ τούτου, τα τοπικά μοντέλα που παράγονται στην πλευρά του πελάτη βασίζονται σε διαφορετικές κατανομές δεδομένων, οι οποίες δεν μπορούν να θεωρηθούν πανομοιότυπες ή ανεξάρτητες. Αυτή η μεροληψία μπορεί να προκαλέσει απόκλιση των τοπικών μοντέλων μηχανικης μάθησης και ταυτόχρονα να υποβαθμίσει την απόδοση του συνολικού μοντέλου (συνάθροιση των τοπικών μοντέλων).

Η μεροληψία που εισάγεται από τους πελάτες μεταφράζεται στην ομοσπονδιακή μάθηση σε ασυμμετρία στο σύνολο των πελατών. Οι βασικές μορφές ασυμμετρίας είναι οι εξής: 1. Ασυμμετρία στην ποσότητα των δεδομένων (Quantity skew). Σε αυτή την περίπτωση έχουμε πελάτες με σημαντικά μεγάλες διαφορές στις ποσότητες των δεδομένων που διαθέτει ο καθένας. 2. Ασυμμετρία στην κατανομή των ετικετών (Label skew). Κάθε πελάτης έχει τις δικές του προτιμήσεις με αποτέλεσμα κάθε πελάτης να έχει διαφορετική κατανομή ετικετών (labels) 3. Ασυμμετρία στην κατανομή των χαρακτηριστικών (features) των προτύπων (Feature skew). Κάθε χρήστης εισάγει τον δικό του θόρυβο με αντιπροσωπευτικό παράδειγμα την αναγνώριση χειρόγραφων ψηφίων όπου κάθε πελάτης έχει τον δικό του γραφικό χαρακτήρα.

Το παραπάνω πρόβλημα αποτελεί βασική πρόκληση στην ομοσπονδιακή μάθηση και η αντιμετώπισή του αποτελεί αντικείμενο μελέτης για πολλούς ερευνητές. Οι προσπάθειες που γίνονται, από την επιστημονική κοινότητα, με σκοπό να μετριάσουν την επίδραση της ετερογένειας ταξινομούνται σε τρεις βασικές κατηγορίες. Στην πρώτη κατηγορία οι ερευνητές προσπαθουν να μετριάσουν το πρόβλημα ζητώντας από τους χρήστες να μοιραστούν ένα ποσοστό των δεδομένων τους π.χ 10% κατα το στάδιο της προ-επεξεργασίας (pre-processing) παραβιάζοντας, όμως, την βασική αρχή της ομοσπονδιακής μάθησης περι προστασίας της ιδιωτικότητας. Στην δεύτερη κατηγορία κατατάσσονται οι αλγόριθμοι που προσθέτουν κανόνες και αυξάνουν την πολυπλοκότητα στην μεριά του πελάτη με σκοπό να μην αποκλίνουν, πάνω από ένα όριο, τα τοπικά μοντέλα που παράγονται από το παγκόσμιο μοντέλο. Το βασικό πρόβλημα με αυτή την προσέγγιση είναι ότι ο πελάτης έχει περιορισμένους πόρους και υπολογιστικές δυνατότητες. Τέλος, στην τρίτη κατηγορία ανήκουν οι αλγόριθμοι που προσπαθούν να αντιμετωπίσουν το πρόβλημα στο στάδιο της συνένωσης

(Ensemble) των τοπικών μοντέλων. Η προσέγγιση αυτή σέβεται τα προστάγματα της ομοσπονδιακής μάθησης και τους περιορισμένους πόρους των χρηστών, ενώ παράλληλα είναι εύκολα υλοποιήσιμη. Για αυτό το λόγο, στην παρούσα διπλωματική επικεντρωθήκαμε στην τρίτη κατηγορία.

Ο αλγόριθμος που ανέδειξε τον χώρο της ομοσπονδιακής μάθησης ονομάζεται Federated Averaging και προσπαθεί να μετριάσει την μεροληψία των χρηστών λαμβάνοντας σε κάθε κύκλο εκπαίδευσης τον σταθμισμένο μέσο όρο των τοπικών μοντέλων που παράγονται από τους πελάτες. Τα τοπικά μοντέλα σταθμίζονται με βάση το πλήθος των δειγμάτων που κατέχει κάθε πελάτης. Θέλοντας να βελτιώσουμε τον Federated Averaging παρατηρήσαμε ότι σταθμίζει τα τοπικά μοντέλα μόνο με βάση την ποσότητα των δειγμάτων που κατέχουν - κάτι που τον κάνει ιδανικό σε ένα περιβάλλον με ασυμμετρία στην ποσοτητα των δεδομένων (Quantity skew). Η δική μας πρόταση είναι να σταθμίσουμε τα τοπικά μοντέλα που παράγονται με βάση την επίδοσή τους στο τοπικό τους σύνολο δεδομένων (local validation) . Η υπόθεση που κάνουμε είναι ότι τα μοντέλα που δεν τα πάνε καλά στον τοπικό έλεγχο αποκλίνουν περισσότερο από το μέχρι τώρα παγκόσμιο μοντέλο. Στο στάδιο της συνένωσης σταθμίζονται με βάση το αποτέλεσμα του τοπικού ελέγχου (loss) με στόχο να προωθηθούν τα μοντέλα που δυσκολεύονται περισσότερο στον τοπικό έλεγχο και να εμπλουτίσουμε το παγκόσμιο μοντέλο. Την παραπάνω μέθοδο την ονομάζουμε Fedloss.

Στην συνέχεια χρησιμοποιώντας το Flower (A Friendly Federated Learning Framework), ένα εργαλείο που βρίσκεται στην αιχμή του δόρατος στο πεδίο της ομοσπονδιακης μάθησης δημιουργήσαμε ένα περιβάλλον με σκοπό να ελέγξουμε την μέθοδό μας. Προσομοιώσαμε πέντε διαφορετικά σενάρια: ομοιογενές, ασυμμετρία στην κατανομή των ετικετών, ασυμμετρία στην ποσότητα των δεδομένων, ασυμμετρία στην κατανομή των χαρακτηριστικών και τέλος, ένα μεικτό σενάριο με ασυμμετρία στην κατανομή των ετικετών και στην κατανομή των χαρακτηριστικών. Σε αυτά τα περιβάλλοντα επιχειρήσαμε να εκπαιδεύσουμε ένα μοντέλο αναγνώρισης εικόνων χρησιμοποιώντας την μέθοδο που προτείνουμε (FedLoss) και την συγκρίνουμε με τις πλέον σύγχρονες μεθόδους (FedAvg, FedMedian, FedAvgM). Τέλος, παραθέτουμε και σχολιάζουμε τα αποτελέσματα μας ενώ προτείνουμε και μελλοντικές βελτιώσεις.

Chapter 1

Introduction

Over the last decade the abundance of data has been playing a key role in the rapid development of Artificial Intelligence-Machine Learning (AI/ML)-based applications. AI/ML involves a wide umbrella of data-driven techniques and algorithms that enable computer systems to learn from past experience. These systems are fed with (big amounts of) data to produce (train) ML models, in order to understand complex patterns and therefore solve problems that cannot be addressed by analytical methods. Relevant applications include video analytics, weather prediction, energy consumption prediction, etc.

Training data is generated from a wide range of sources, including social media, mobile phones, Internet of Things (IoT) devices and traditional enterprise systems. Traditionally, data is collected and processed (trained) ***centrally*** in a single machine or a cluster of machines e.g., in a cloud server (Centralized Machine Learning - CL). Several challenges emerge from this centralized approach, however. Since data may contain sensitive information e.g., medical records, user location, habits etc., uploading data to a central or a public entity is not always viable. From 2018 onwards, additional restrictions were imposed by new legislation acts to enable data protection and privacy. The most representative example is the general data protection regulation (GDPR) [3] by the European Union. Besides privacy, transferring enormous amounts of data may lead to network overloads and increased demand for storage infrastructure. As the ML task scales e.g., to TBytes of data, CL can cause computational bottlenecks at the server side as well as increased costs.

To tackle the above-mentioned challenges, the concept of Federated Learning (FL) has emerged. FL was introduced in 2016 [4] and has been drawing the attention of the scientific community ever since. The basic idea behind federated learning is to train machine learning models on ***decentralized*** data sources, such as mobile devices or edge devices, so that the data is not exposed to the central server. As such, the devices download an ML model from the central server, train the model locally using their own data, and then upload the updated (local) model back to the server. The server thereafter aggregates the local models, creating a new (global) ML model. The process is repeated for several cycles (rounds), until ML model convergence is reached. The key advantage of federated learning is that data remains in the client devices at all times, therefore privacy is ensured. That means it offers a high level of privacy protection by ensuring that data remains local and is not shared with third parties. This approach addresses the concerns of individuals and organizations that are hesitant to share their data with others due to privacy and security reasons. Furthermore, this makes it more difficult for hackers to gain access to sensitive data. Also, it manages to reduce the communication cost, since ML models are in general light-weight in terms of data size, compared to the actual data.

Unfortunately, federated learning also comes with some disadvantages and the first is that it relies on data being available at the edge devices, which may not always be the case. Data unavailability can cause serious degradation to the ML model's performance. Moreover, transmitting/receiving ML models between edge client devices and a central server creates potential security risks, such as the possibility of a malicious actor intercepting the updates or injecting fake data into the training process. From a software engineering perspective, federated learning involves multiple devices working together to train a model. In such scaled systems, the increased complexity hinders software maintenance, debugging, etc. What is identified as federated learning's achilles heel however is data heterogeneity. Bias is introduced in client data, due to user trends, variations, patterns, habits, etc. As such, local models produced at the client-side are based on different data distributions, which cannot be assumed identical or independent (***i.i.d.***). This bias can cause the local ML models to diverge and at the same time degrade the performance of the global ML model (aggregation of local ML models).

Object of this work

Several bias mitigation techniques have been proposed [5], however they disregard the key restrictions of FL e.g., client data exposure. In our work, we address the problem of bias in FL settings taking into account the fundamental restrictions imposed in FL systems. Initially, we perform a comparative analysis of existing state-of-the-art (SotA) bias mitigation algorithms. We then introduce FedLoss, our novel bias mitigation algorithm, which is benchmarked on our custom FL framework in several data distribution scenarios and datasets. Our results suggest that FedLoss outperforms SoTA algorithms e.g., FedAvg, FedAvgM, FedMedian in terms of accuracy by an average of 2%, across all bias scenarios. FL's vision is to harness the power and the wealth of client data but at the same time respect user privacy. Our work aims to identify existing research gaps in FL literature and support relevant research on the field.

Structure

The motivation about Federated learning was established in Section I, the following section is dedicated to detail the Federated learning framework and the main challenge of non-IIDness (Section II). In Section III we present our experimental novel algorithm and the simulation that we conducted in order to evaluate the algorithm. The results are presented in Section IV. Lastly, we conclude in Section V pointing also to future exploration.

Chapter 2

Theoretical Background

In this chapter, we are going to introduce the terminology and concept of Federated Learning (FL). Although, we assume that the reader is familiar with the basic Machine Learning concepts, for the sake of clarity we repeat these established concepts. For starters, we quote a definition of federated learning and the framework that we need in order to develop our thoughts. After that we are going to talk about the main challenges which arise in a federated learning setting. Finally, we discuss our approach to mitigate these challenges.

2.1 Machine learning

Artificial intelligence (AI) refers to a wide range of techniques that enable systems to learn and improve from past experience. Machine Learning (ML) is a subset of AI. As a concept, it captures an array of data-driven algorithms that operate without explicit programming. The theoretical basis of Machine learning is dated back in Kolmogorov–Arnold representation theorem [6], which states that a continuous multivariate function can be expressed on a compact set in terms of sums and compositions of a finite number of single variable functions. As such, a complex problem (task) can be broken down (and therefore addressed) in a large number of connected mathematical functions; the latter are defined as ***neurons***. The (numerical) connections between neurons are called ***weights***. The neurons together with their respective weights comprise the ***Neural Network (NN) model*** (or simply ***ML model***), as depicted in Fig 1 If X denotes the input feature vector of the ML model, the respective output vector Y can be then written as Y=**trans**(X)W, where **trans**(X) denotes the transpose of vector X and W the weights vector.

The process of ML is divided in two main phases: ***training*** and ***inference***. During the training (offline) phase, the ML model is fed with historical data, in order to learn the data patterns. Specifically, X and Y are known and the ML algorithm estimates the respective weight vector W, which in fact represents the relationships between the input (X) and output (Y) variables (see Figure 1). The estimation is performed via the minimization of a ***loss function***, which is a comparison metric between prediction and the ground truth. The process of minimizing the loss function (training) is the core part of the Ml algorithm. Modern ML algorithms utilize Gradient Descent [7], an optimization algorithm to find the minimum of the loss function. Every ML model has a set of hyper-parameters, which are tuned prior to the training phase (usually via test-runs. Relevant examples of hyper-parameters include:

- ***Epochs***: the number of complete passes through the training data.
- ***Batch size***: the number of training samples to work through before the model's internal parameters (weights) are updated.
- ***Learning rate***: it governs the pace at which an algorithm updates or learns the values of a parameter estimate.

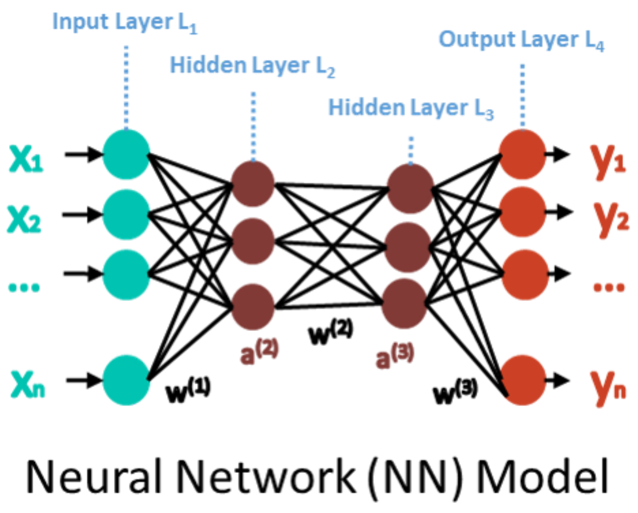

Figure 1: Neural Network Model.

Upon convergence (end of ML training), the ML model can be utilized for the inference phase. As such, new (unseen) data is processed by the ML model, to extract future predictions i.e., the input variable X vector and the weight vector W are known and therefore the output vector Y can be obtained. Note that unlike the training phase that is processing-heavy, inference does not require much processing capacity.

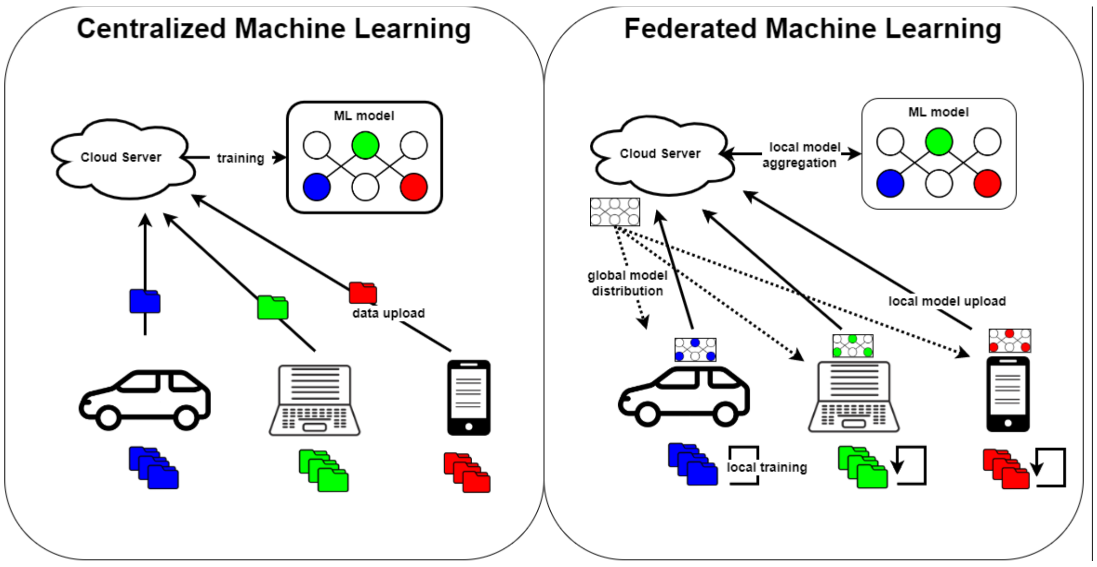

Figure 2: Centralized Machine Learning vs Federated Machine Learning.

In the traditional ML paradigm (see fig 2 Centralized Machine Learning - left part), the data from various sources is collected in a central entity e.g., a cloud server. It is thereafter divided into training, validation and inference dataset. The training dataset is used to train the ML model's parameters (weights). The validation dataset is used to control the training phase (convergence tests), perform hyper-parameter tuning, etc. Finally, the inference dataset (unseen data) is used in order to test (inference phase) the performance (accuracy) of the output ML model. The term "generalization" refers to the model's capability to adapt and react properly to previously unseen data. It examines how well a model can digest new data and make correct predictions. Generalization constitutes a key concept, when training an ML model. If a model is over-trained in the training dataset, it will be incapable of generalizing, therefore producing erroneous predictions under new data. The latter is known as ***overfitting***. The inverse (***underfitting***) is also true, which occurs when an ML model is trained with inadequate data. In cases of underfitting, the ML model fails to make accurate predictions even with the training data.

2.2 Federated learning framework

To begin with, we need a definition for the term of Federated Learning (FL). There have been many attempts in literature to define it and for the first time the term was introduced in 2016 by McMahan et al. [4] However, in this work we obtain the definition of Kairouz et. al. [8] which we reproduce in the lines that follow.

“Federated learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client’s raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.”

The process of FL is depicted in Fig 2 Each client (data generator source) e.g., a smartphone, a laptop, a sensor, a car, etc. receives a global model from the centralized (cloud) server. Each client then performs local training utilizing its own data and produces a local client ML model. The server thereafter collects all local models from the clients and merges (aggregates) them into an updated global ML model. This constitutes a ***training FL round***. Several rounds are usually required so that the global ML model reaches convergence. Note that during FL, client data always remains at the client-side and is never exposed to the server. As such, privacy is preserved, which is the key motivation of FL.

Then, we have to mention that depending on the size and the kind of clients we have two categories of Federated Learning: the ***Cross-Device*** and the ***Cross-Silo*** setting. In the first case we have a relatively large number of clients ($>10^6$) with limited resources and unreliable networks. For example, clients can be mobile phones, IoT devices, sensors, wearables or even autonomous vehicles. These clients are restricted due to: 1) their computational capabilities 2) their mobility (and therefore availability) 3) energy constraints e.g., power sources such as batteries and 4) their connectivity, which relies on the communication infrastructure i.e., WiFi/4G/5G networks. On the other hand, in Cross-Silo, clients refer to big data hubs such as banks, hospitals, companies, countries, etc. In this case, we have a relatively small size of clients (2-100), their computational resources are significantly better, they use reliable communication networks and usually drain power from the electrical grid (which is generally assumed stable).

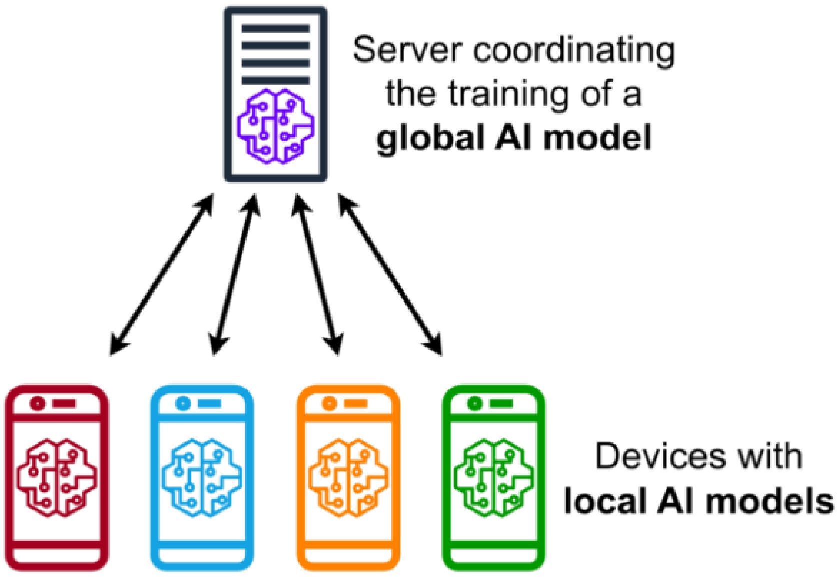

Figure 3: Cross-Device setting: million of devices with relatively little data which are considered to be unreliable clients. [1]

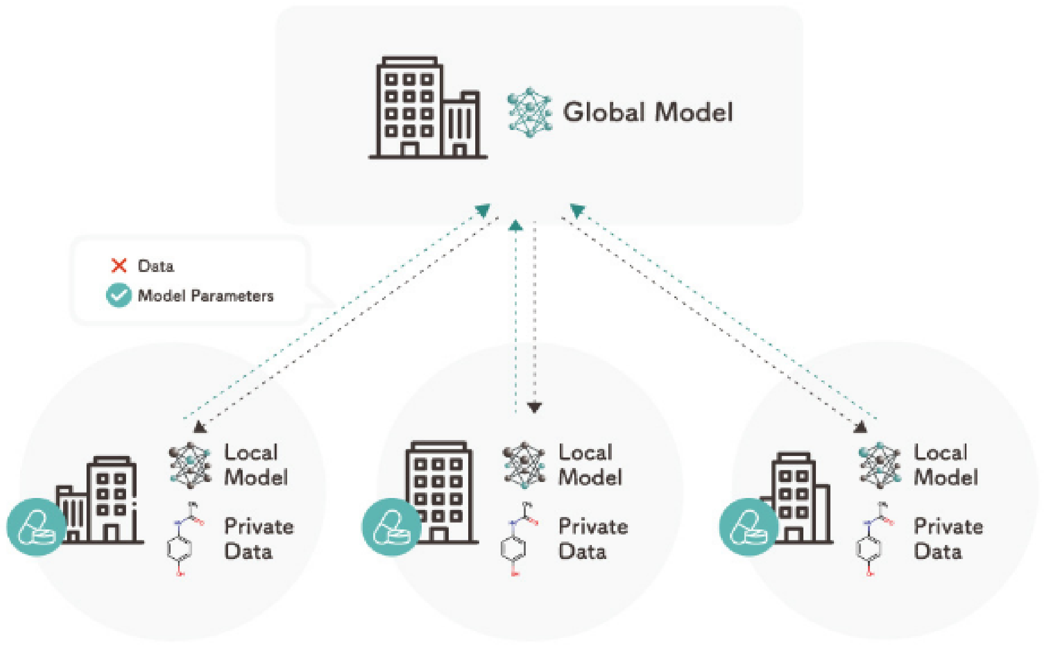

Figure 4: Cross-Silo setting: few data hubs, such as hospitals, banks etc. which are considered to be reliable clients. [2]

Having said this, we have to mention that in both cases we rely on the same Federated Learning algorithm. This algorithm is our next topic.

In this section, an outline of the Federated Learning algorithm is presented, followed by a formal definition. Our algorithm consists of four basic steps:

1. The server broadcasts an initial (untrained) global model to the clients.
2. The clients perform local training, based on their acquired (stored) data, each producing a local model.
3. Each client returns (upload) a trained local model.
4. The server collects the local models, aggregates them to an updated global model and repeats step 1. until convergence is reached.

The first step is the initialization phase, which only occurs once. In this step the server selects the clients that meet certain predefined criteria e.g., have adequate data availability or connectivity. By default, the server performs a random sampling to select the clients for the next FL round. After that the selected clients download the global model and they start the local training. When they complete their local process, the local models are sent back to the server. The most common strategy is for the server to wait for all clients to upload the model and then to proceed to the next step. This is referred to as synchronous FL. Asynchronous FL schemes are also proposed [9], but are beyond the scope of this study. Last but not least is the aggregation step. In the aggregation step the server merges the local models into a global one. The formal definition [10] of FL is analyzed in the following section:

```
Generic Federated Learning Algorithm
Input : N, C , T, E
Output: $w_{TE}$
    Initialize $w_0$
    for each round $t \in \{0, E, 2E, \ldots, (T-1)E\}$ do
        m ←max(CxN, 1)
        $I_t$ ← (random set of m clients)
        for each client $i \in I_t$ in parallel do
            $w^i_{t+E}$ ← CLIENT-UPDATE($w_t$)
        end for
        $w_{t+E}$ ← AGGREGATION($w^1_{t+E},\ldots,w^N_{t+E}$)
    end for
    return $w_{TE}$
```

We assume the following notation: w^i is parameters of the model for the client i , N is the number of clients, C is the fraction of sampling, T is the number of rounds, E is the local epochs.

2.3 Non-Independent and Identically Distributed

Despite the fact that the Federated Learning setting enables the continuous development of Machine Learning models, it sets strict restrictions in regards to accessing raw data i.e., the clients dataset. Due to the distributed nature of Federated Learning (data is split among the clients), data heterogeneity is introduced. As a result, clients end up producing biased models in general. This fact leads us to say that the I.I.D assumption that maybe holds in a centralized environment tears down in a federated world. Before we dive into the non-IIDness we will try to introduce the I.I.D concept for completeness.

To start with, I.I.D stands for Independent and Identically Distributed and it is the scientific definition of randomness in statistics. To avoid a technical definition for the terms "***Independent***" and "***Identically***", we are going to introduce them with the classic coin toss experiment example. Let's assume that we toss a coin. If we get "heads" on the first trial, the probability of getting "heads" or "tails" in the next trial doesn't change. Note here that we don't care about the fairness of the coin, it doesn't have to be 50-50. It can be 60-40 or 70-30 for "heads"-"tails", respectively. Regardless of the number of previous tosses, the next toss has no dependency from the past. As such, the data distributed is assumed "Independent" i.e., each experiment (toss) has no dependency on previous experiments. Now, in our example the term "identically" means that in each coin toss the probabilities to get heads or tails doesn't change over time. To avoid the confusion with the term of Independency imagine that you toss a fair coin and suddenly after some repetitions someone changes your coin with an unfair one. Then, the underlying mechanism that generates the data (Heads or Tails) has changed. In this case the identically distributed assumption doesn't hold. [11]

Although our intention is to avoid in depth analysis the rest of this paragraph is dedicated in the explanation of fundamental terms such as loss function, empirical risk and the connection between I.I.D assumption and machine learning. Initially, the IID assumption is closely related to the concept of empirical risk in machine learning. Successively, empirical risk is the average loss of a model over a dataset. In machine learning, the goal is to learn a function f that maps inputs x to outputs y, given a set of training data {(xi, yi)}. The empirical risk of a model f is defined as the average loss $L(f(x_i), y_i)$ over the training data: $R(f) = \frac{1}{n}\sum_i L(f(x_i), y_i)$ a where n is the number of samples in the training data. A loss function, also known as a cost function used in machine learning to evaluate the performance of a model by measuring the difference between predicted and actual values. The IID assumption is often made when defining the empirical risk. Specifically, we assume that the training data {(xi, yi)} are independent and identically distributed samples from an unknown probability distribution P(X, Y), which means that the samples are drawn independently from the same distribution. By minimizing the empirical risk over a class of functions F, we can find a model f* that best fits the training data. However, if the IID assumption is violated, the empirical risk may not be a good estimate of the expected risk, and the model may overfit or underfit the data (see also Section 1). In Federated Learning the IID assumption collapses by nature. Each client cannot be expected to collect samples from the same distribution as the rest of the clients, due to seasonality, client mobility, localities, user habits, etc. Therefore the concept of non-IIDness is introduced, which poses a great challenge for the ML task.

Clearly, the term non-IIDness means that IID assumption doesn't hold. In a federated setting, non-IIDness shows up with the form of data heterogeneity which exists in many ML applications and distributed learning settings. Let assume that we have a supervised learning task with feature x and label y. Each client is randomly selected from a pool. We denote Pi and Pj the local data distributions of client i and client j participating in federated learning. Then, data heterogeneity can be expressed as the difference between Pi and Pj for different clients i and j. Using the above notations, we split the data heterogeneity in federated learning in the following categories.[12]

- **Label distribution skew**: Label distribution means that the label Pi(y) distribution of different clients is different. For example, each hospital has its own specialization so it is possible to have more records from patients with specific kinds of diseases.

- **Quantity skew**: Even though the data distributions among the clients can be consistent, the size of the local datasets varies across the clients (in the general case). For example, a client collecting images from a smartphone in an urban environment may acquire twice the amount of samples compared to another client in a rural environment.

- **Feature distribution skew**: Feature distribution skew describes the variation of the distribution of the feature Pi(x) from client to client. The most representative example is the handwritten digits. Imagine that you have to train a model to recognize digits in a federated setting. Each client (person) is expected to have a unique handwriting style.

- **Mixed skew:** When several biases are introduced, a combination of the above-mentioned cases is presented, which is defined as a mixed skew.

This is not an exhaustive list; we have only presented the basic types. One honorable mention is the temporal skew which refers to the skew in distribution under temporal data, including spatio-temporal data and time-series data [13]. Relevant examples include weather data, network statistics e.g., data-rate and energy consumption data.

2.4 The FederatedAveraging Algorithm

In this section we introduce the fundamental aggregation algorithm, Federated Averaging (FedAvg), as presented in [2]: The basic idea of FedAvg is in the aggregation step where the server collects the local models and the size of the training set from each client and performs a weighted averaging to produce the updated global model. Mathematically speaking let's say that w^i_t is the model trained by the client i in the t round and n_i is the size of the respective dataset. For the round t the server has sampled I_t clients (C*N). The updated global model is

$$w_{t+E} = \sum_{i \in I_t} \frac{n_i}{m_t} {w_t}^i \qquad \text{where } m_t = \sum_{i \in I_t} n_i$$

Now we replace the above equations with the term AGGREGATION in the generic federated learning algorithm and the FedAvg is produced.

Federated Averaging Algorithm

Input: N: number of clients, C: Fraction of sampling, T: Number of rounds, E: local epochs

Output: w_{TE}

```
Initialize $w_0$
for each round t ∈ {0, E, 2E, …, (T-1)E} do
m ← max(C*N, 1)
$I_t$ ← (random set of m clients)
    for each client i ∈ $I_t$ in parallel do
    $w^i_{t+E}$ ← CLIENT-UPDATE($w_t$)
    end for
$m_t = \sum_{i \in I_t} n_i$
$w_{t+E} = \sum_{i \in I_t} \frac{n_i}{m_t} {w_t}^i$

end for
return $w_{TE}$
```

2.5 Beyond FedAvg

Over the last decade the scientific community that works in Federated Learning have been developing numerous variants of the Federated Averaging algorithm or algorithms that add every kind of complexity in their logic to mitigate the problem of non-IIDness.

We categorize these approaches based on which phase of the federated process they intervene. As a result, we have three categories: Pre-processing, In-processing, Post-processing (see Figure 4). FL algorithms which add complexity at the beginning of the round before the server broadcasts the model to the clients for training belong in the first category. The most common approach in this category is data sharing (via encryption mechanism) between clients. For example, XorMixFL[14] a privacy-preserving XOR based mixup data augmentation technique where the core idea is to collect other devices' encoded data samples that are decoded only using each device's own data samples. The problem with **these approaches is that sharing data violates the philosophy of Federated Learning**. Privacy is a key concept and the main motivation of FL, therefore such methods are not practical in realistic FL deployments e.g., to train medical data.

The next category is in-processing where an algorithm modifies the local ML task or the server's selection algorithm. For instance FAVOR [15], an experience-driven control framework that intelligently chooses the client devices to participate in each round of federated learning to counterbalance the bias introduced by non-IID data and to speed up convergence At FAVOR framework they use a profiling technique based on model weights and then a Reinforcement Learning mechanism based on deep Q-learning proposes the most suitable clients in order to maximize the validation accuracy and minimize the communication rounds. One more example for this category is the FedProx[16] which is based on FedAvg but it adds a proximal term to the client in order to restrain local updates to the global model. Then there is a family of adaptive algorithms that estimates drift across clients and mitigates it. e.g AdaBest[17], SCAFFOLD[18]. The main **disadvantage of in-processing methods is that they add computational cost in the client-side**. In FL, clients are by nature limited by their computational capacity and resources e.g., battery, bandwidth, etc. Every mechanism that introduces further complexity in the clients may not be sustainable in an actual (scaled) FL environment e.g., in smart homes with several IoT devices, in connected vehicles, in industrial sensor networks, etc.

In the post-processing approach, the FL algorithm tries to mitigate the non-IIDness in the aggregation step i.e., on the server side. Relevant examples include FedAvg, FedAvgM[19] and FedMedian[20]. We are going to talk about these algorithms in the next chapter. The post-processing methods add complexity in the server (which in general is not limited by its resources, since it resides in the cloud) at the same time preserving the privacy of the local datasets. As a result, **they are the most suitable methods to mitigate non-IIDness, taking into consideration practical implementations** and the deployment of FL in real environments.

Having identified this gap, from this section onwards we focus on the post-processing methods. As such, our suggested algorithm and the SotA algorithms we compare against belong to this class of algorithms.

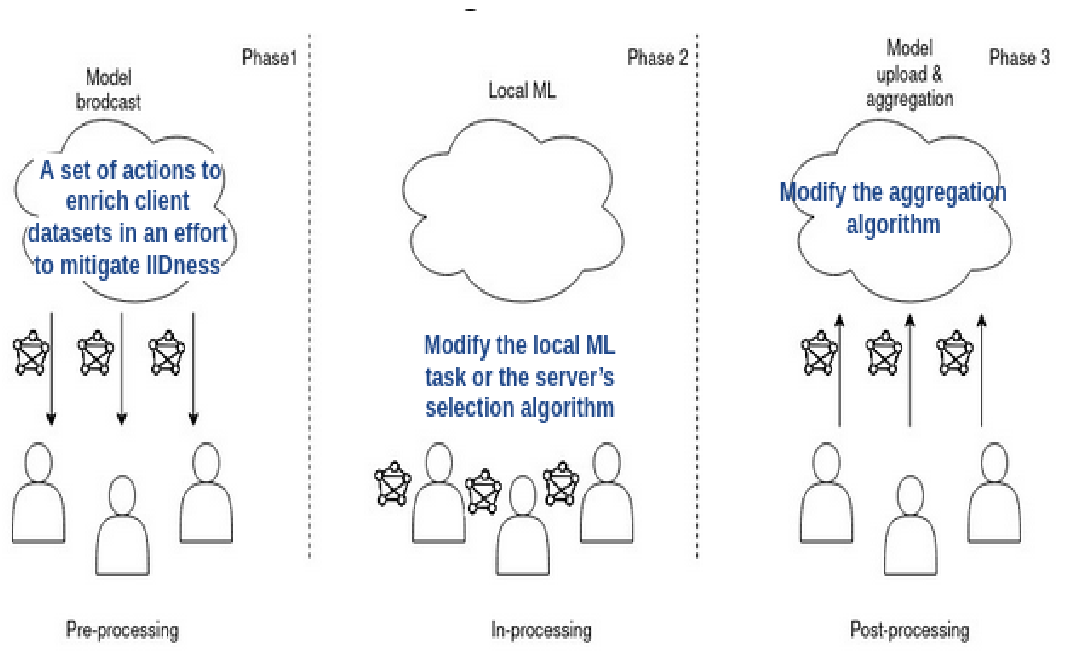
Phase1
Model brodcast
A set of actions to enrich client datasets in an effort to mitigate IIDness
Pre-processing
Phase 2
Local ML
Modify the local ML task or the server's selection algorithm
In-processing
Model upload & aggregation
Phase 3
Modify the aggregation algorithm
Post-processing

Figure 5: Categorization of federated learning methods based on the phase that each method intervenes.

Chapter 3

Methodology

3.1 Proposal

In this work we introduce our novel algorithm, which is a variant of FedAvg based on a validation loss. The FL setting is assumed as follows: Each client is assumed to collect data from the environment, which are locally stored. The acquired client data constitutes the client dataset. Each client dataset is split in 70% training set -10% local model validation dataset -20% global model validation dataset. In each training round, the server broadcasts the global model to several (randomly selected) clients. The selected clients train the global model, using their training dataset. When they finish their local training, they evaluate their local model in the local validation dataset. We denote l^i_{t+E} the validation loss of the i^{th} client. The clients upload their models w^i_{t+E} and the respective losses where the server aggregates them into a global (updated) one. Note that the default FL algorithm (FedAvg) aggregates the local models in a weighted averaging manner, where the weights are proportional to the client dataset size. In our solution, the aggregation step is a weighted averaging of the local models, where the weights are proportional to their respective losses (taken from the validation step). ***Intuition:*** this extra validation step before the aggregation gives a measure of how well the model fits on the dataset and by weighting them with the loss we favor the clients that fit less and we promote the local models that were trained to datasets with different samples and distributions in comparison the already datasets that the global model have already seen. We denote this method as ***Fedloss***.

```
Fedloss
Input: N: number of clients, C: Fraction of sampling, T: Number of rounds, E: local epochs
Output: $w_{TE}$
Initialize $w_0$
for each round $t \in \{0, E, 2E, \dots, (T-1)E\}$ do
 $m \leftarrow \max(C*N, 1)$
 $I_t \leftarrow$ (random set of m clients)
         for each client $i \in I_t$ in parallel do
         $w^i_{t+E} \leftarrow$ CLIENT-UPDATE($w_t$)
```

```
        $l^i_{t+E}$ ←CLIENT-VALIDATION($w^i_{t+E}$) # in 10%
        end for
    $m_t = \sum_{i \in I_t} l^i_{t+E}$
    $w_{t+E} = \sum_{i \in I_t} \frac{l^i_{t+E}}{m_t} w^i_{t+E}$
end for
return $w_{TE}$
```

3.2 Heterogeneity Simulation

In the previous chapter we talked about the fact that the IID assumption in general does not hold in an FL environment. In order to test our proposed nonIIDness mitigation algorithm, we need to simulate the heterogeneity which occurs in a setting like this. For that reason, we have created a benchmarking framework to simulate the basic types of skewness that we have mentioned in chapter 2: label skew, quantity skew, feature skew and a mixed skew scenario. On top, we simulate an ideal scenario, where the IID assumption holds. Our simulation strategy is based on the work of Li et al. [21].

IID scenario: At first, we created a homogenous partition by splitting our dataset into equal sizes with consistent distributions. As such, client datasets are similar in size and are drawn from the same distribution. Note that the (ideal) IID scenario is rarely found in an actual system. However, we have included the scenario in our simulations for completeness as well as to provide a fair comparison between our proposed algorithm and SotA alternatives.

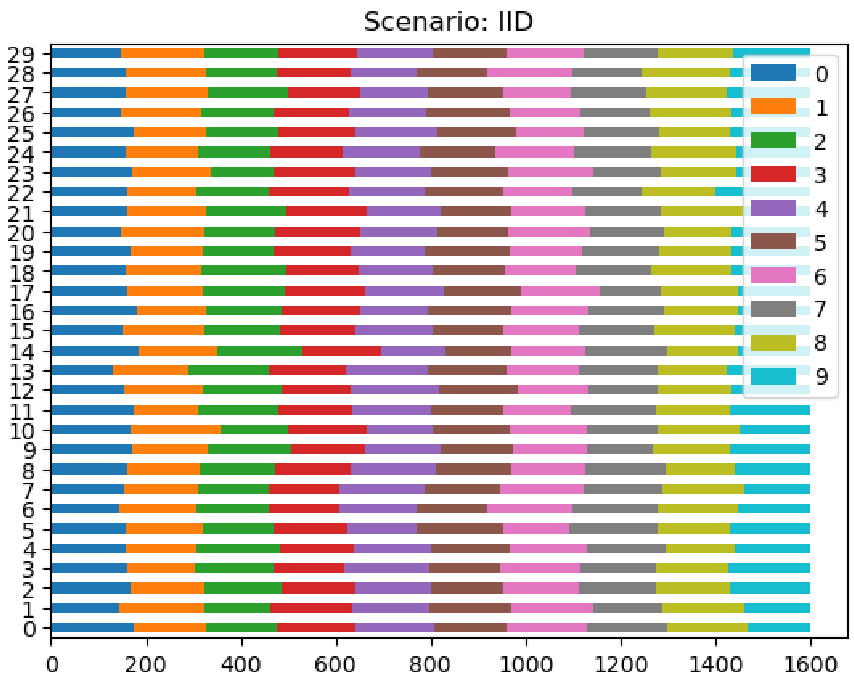

Figure 6: Clients' distribution in IID setting.

Label Distribution Skew: In order to simulate this kind of heterogeneity the authors [21] proposed two different methods. The first is quantity-based label imbalance where each client owns a fixed number of labels. For example, if we were addressing a digit-classification task, such as MNIST [22] and assumed a total of 10 clients, we would assign one digit/label per client. The second method is distribution-based label imbalance; according to this, each client is allocated a proportion of the samples of each label based on the Dirichlet distribution. Following the authors [21] we denote Dir(β) the Dirichlet distribution and β the concentration parameter. The advantage of this approach is that you can control the level of imbalance by varying β. In this work, we preferred the second method to simulate this type of skewness, since it models label distribution skewness in a more realistic way. An example of label distribution skewness is depicted in Figure 7 (for β = 0.5).

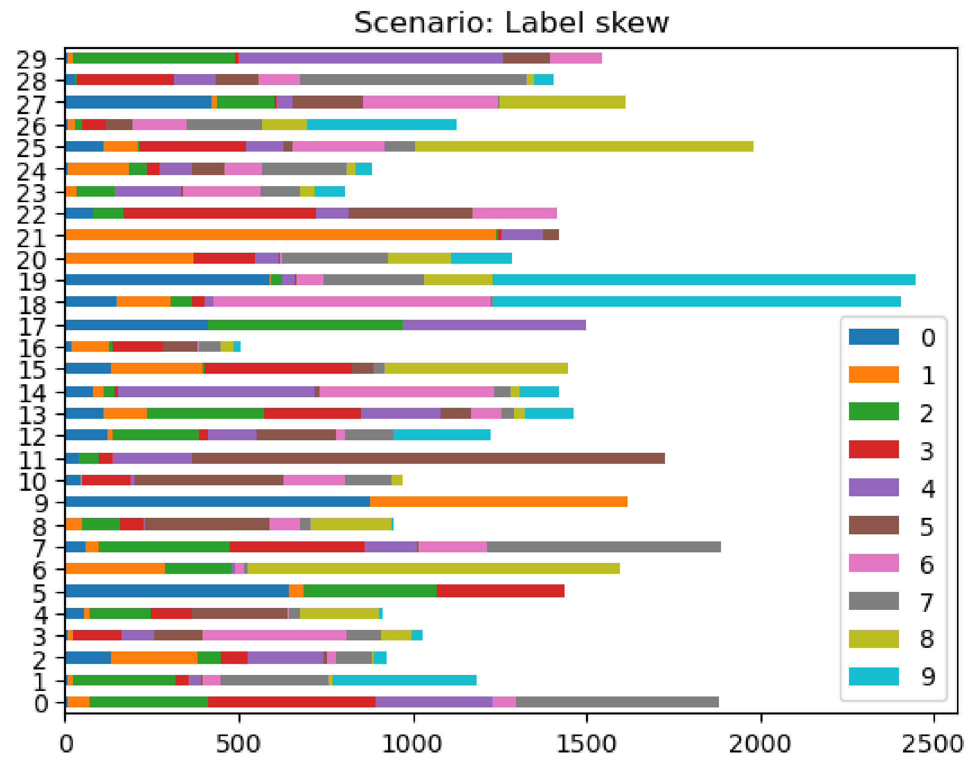

Figure 7: Clients' distribution in label skew setting.

Quantity skew: In this scenario, we preserve the data distribution among the clients, whilst varying the respective local dataset sizes. To simulate this, we use Dirichlet distribution to allocate different amounts of data samples into each client. Like distribution-based label imbalance setting we can use the concentration parameter β to control the level of imbalance. Figure 7 depicts an example of quantity skew.

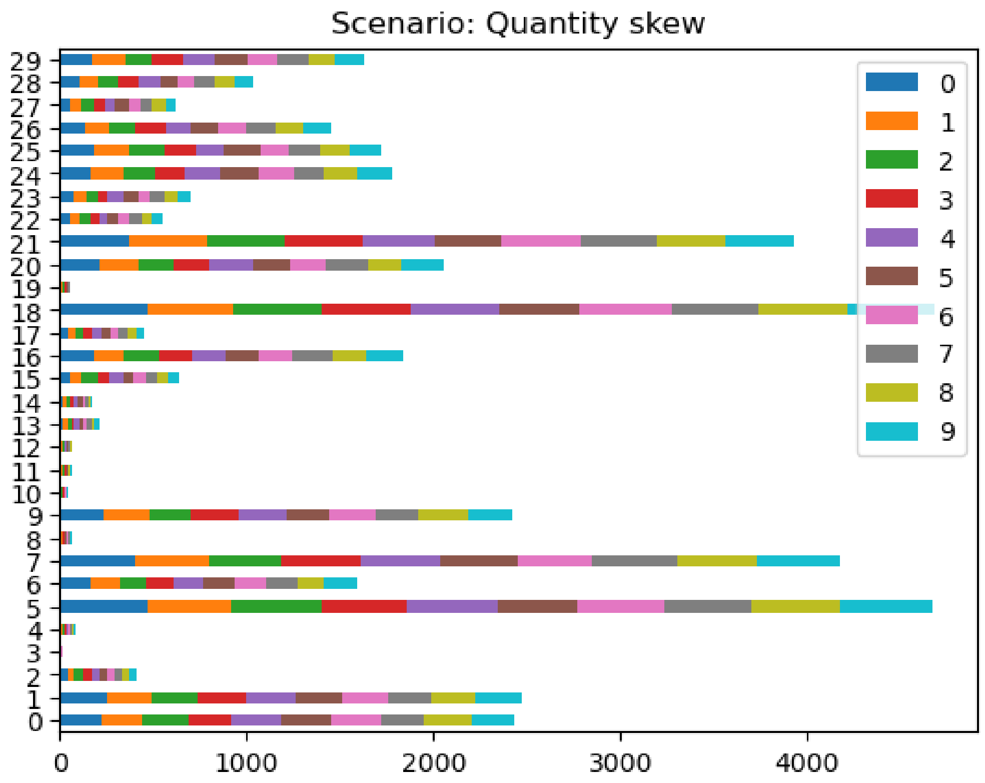

Figure 8: Clients' distribution in quantity skew setting.

Feature Distribution Skew: In order to achieve this skewness there are three different ways: Noise-based feature imbalance, Synthetic feature imbalance, Real-world feature imbalance. We use noise-based feature imbalance where we divide the whole dataset into multiple clients randomly and then we add different levels of Gaussian noise for each client. More specifically, in each client i noise is added to its data, based on a Gaussian distribution with a mean value of 0 and a variance value of $\sigma i/N$, where σ stands for the noise level, i is the i-th client, and N is the population of clients. By changing σ, the feature dissimilarity can be increased or decreased. Relevant examples are depicted in Figures 9, 10

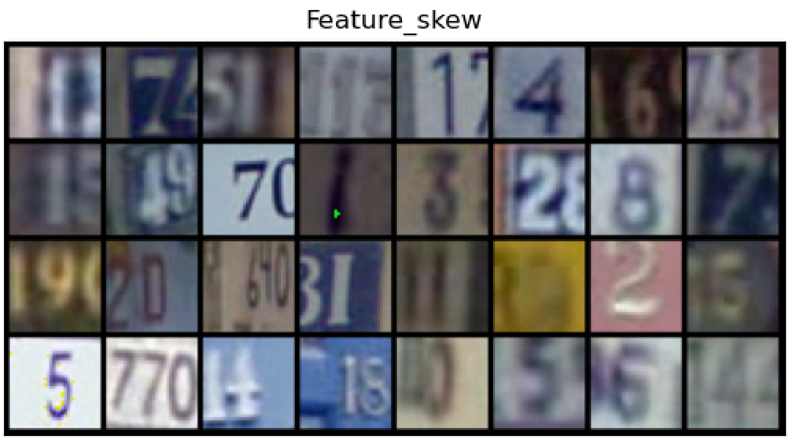

Figure 9: Local dataset of client 1.

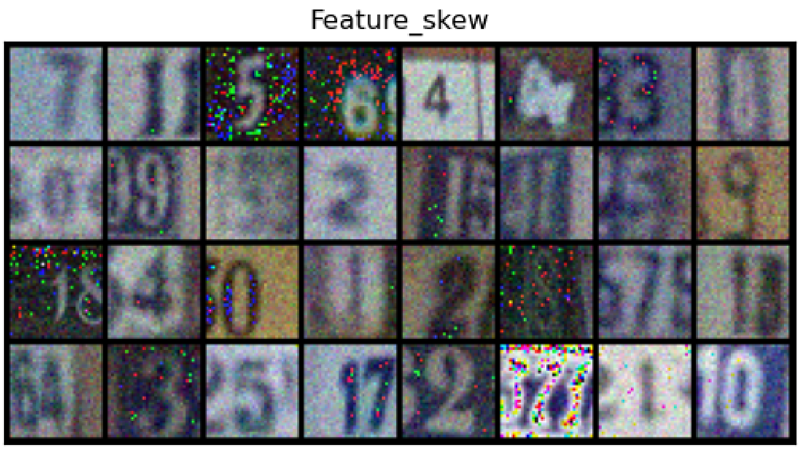

Figure 10: Local dataset of client 20.

Mixed skew: A mixed skew scenario can be any combination of the above basic types of skewness. Our mixed skew setting was created by using the label skew scenario and feature distribution skew. Mixed skew represents scenarios that are closer to reality, since they capture various types of biases introduced in the clients.

3.3 Experimental setup

3.3.1 Datasets and federated settings

Our experiments are conducted in two image-classification (public) datasets that are widely used in literature: SVHN [23], CIFAR10 [24]. The Street View House Numbers (SVHN) Dataset (see Figure 11 - left part) is a dataset based on images from real-world settings, used for ML and in particular object recognition algorithms. Images refer to house numbers in Google Street View images. It contains a total of 10 classes, one for each digit. As such, digit '1' has a label value of 1, '9' has a label value of 9 and '0' has a label value of 10. SVHN contains a total of 73257 digits for training and 26032 digits for testing. CIFAR-10 (see Figure 11 - right part) is also an object recognition dataset that consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. The following object classes can be found in the dataset: "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship" and "truck". Note that the classes are mutually exclusive i.e., there is no overlap between automobiles and trucks.

Figure 11: SVHN[23] & CIFAR10 [24] datasets.

For each dataset we simulate 5 different scenarios: IID, Label skew, Quantity skew, Feature skew, Mixed skew (that is a combination of Label skew and Feature skew). For the federated learning setting, the server selects 5 out of 30 clients during each communication round in line with the results presented in [25], with T = 50 total rounds for all methods.

In order to evaluate our proposed method we compare against existing SotA bias mitigation algorithms: FedAvg[4], FedMedian [20], FedAvgM [19]. FedAvg refers to the default FL aggregation algorithm, as analyzed in Section 2 The main idea behind FedMedian is that instead of averaging the local models, it calculates the median model. The main advantage of FedMedian is the robustness against the outliers. On the other hand, FedAvg with momentum (FedAvgM) is a variant of FedAvg, which utilizes the momentum technique at the aggregation step. Specifically, FedAvg updates the weights via $w \leftarrow w - \Delta w$, where Δw is the weighted average of the local updates. The method of FedAvgM adds momentum at the server, by computing $v \leftarrow \beta v + \Delta w$, and update the model with $w \leftarrow w - v$.

3.3.2 Model Architecture

To address the above-mentioned ML tasks, we use a Convolutional Neural Network (CNN), being the most suitable model for image-classification tasks[26]. CNN is a well-known deep machine learning architecture that accepts an image as input, assigns importance (learnable weights and biases) to various aspects in the image, therefore enabling differentiation between images i.e., image classification.

Our custom model was developed using two 5x5 convolution layers, followed by 2x2 max pooling (the first with 6 channels and the second with 16 channels) and two fully connected layers with ReLU activation (the first with 120 units and the second with 84 units). To tune our custom model (prior to the actual training task), we conducted test-runs using grid search. The respective results are shown in the Appendix. The optimal values obtained are the following: Adam optimizer with a learning rate of 0.001 (both datasets), local epochs = 10 (both datasets) and batch size = 64 and 128 for SVHN and CIFAR10, respectively.

3.3.3 Evaluation

We follow a federated evaluation by splitting the local datasets into 80% training dataset and 20% test dataset. In each round we perform an evaluation step after the aggregation step where we broadcast the updated model to the clients and they evaluate the model on the test dataset. We are utilizing the accuracy metric i.e., the percentage of successfully classified images to the total number of classified images, as our evaluation performance indicator. Our accuracy metric has a maximum value of 1 (all images successfully classified) and a minimum value of 0 (no image successfully classified). We would like to take into consideration the size of the local dataset that's why we weighting the local accuracy with the respective dataset size before we take the average. In the evaluation step we assume full participation of the clients i.e., all clients perform inference/testing on their local datasets, regardless of their participation in the training process.

3.3.4 Environment

For our purposes we utilize the Flower framework [27] which gives us the ability to simulate a Federated Learning setting in a single machine. Flower also provides implementations of different strategies which we use for our baseline comparison strategies. Furthermore, we make use of FedLab [28] which implements the basic non-IID partitions (NIID-Bench[21]). The rest of the stack is Python and Pytorch. Our gear: a cloud VM with 30 virtual cores and 200 GB ram.

Chapter 4

Results

For each setting we conducted 5 runs and we took the mean average. Due to the partial sampling of the clients, the training process for most of the scenarios becomes unstable so for our analysis we split the results into 5 segments with 10 rounds per segment and at the end of each segment we compare the algorithms. We mark with bold the method with the highest accuracy.

4.1 Homogeneous

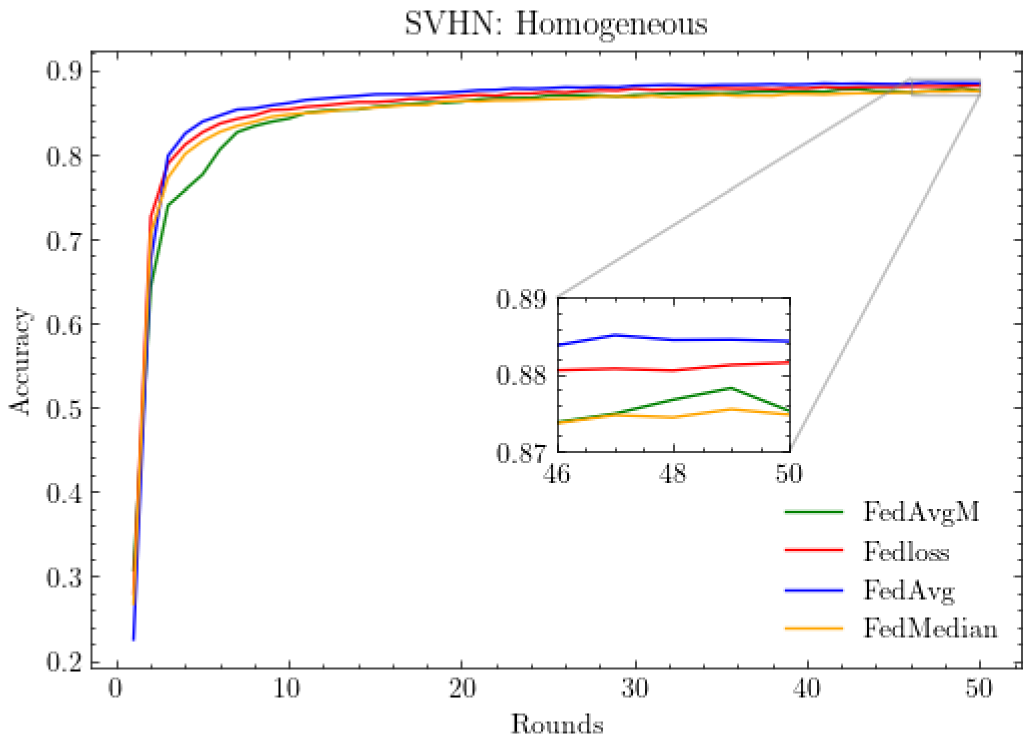

Figure 12: Results for the SVHN task in a homogeneous setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	**86.1% ±0.5%**	**87.4% ±0.5%**	**88.1% ±0.5%**	**88.3% ±0.4%**	**88.4% ±0.5%**
FedAvgM	84.2% ± 1.9%	86.2% ±1.0%	87.0% ±0.4%	87.5% ±0.5%	87.5% ±0.6%
FedMedian	84.7% ±1.4%	86.3% ±0.8%	86.8% ±0.7%	87.2% ±0.6%	87.5% ±0.7%
Fedloss	85.3% ±1.0%	86.9% ±0.5%	87.8% ±0.4%	87.8% ±0.5%	88.2% ±0.4%

Table 1: Results of the methods per 10 rounds in a homogeneous setting for the SVHN task.

The first simulation is on the SVHN task in a homogeneous setting. Undoubtedly in this scenario the FedAvg outperforms the rest of the algorithms which scores top accuracy 88.5% at the 47th round. At the same time, Fedloss beats FedMedian and FedAvgM in each segment.

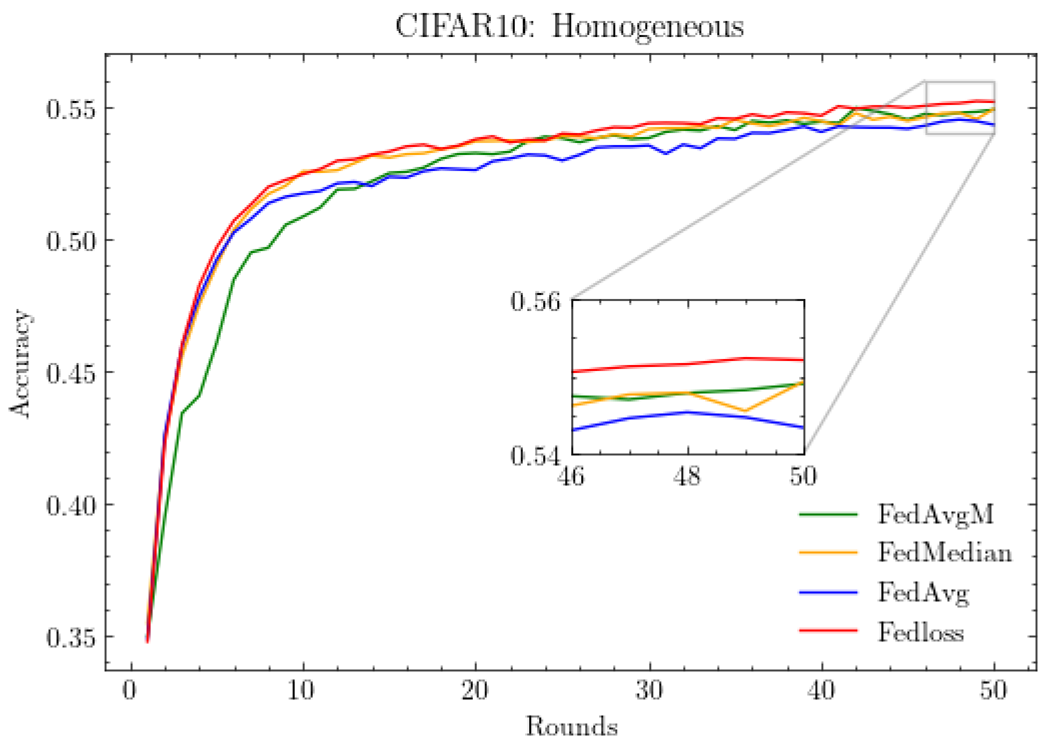

Figure 13: Results for the CIFAR10 task in a homogeneous setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	51.7% ±0.3%	52.6% ±0.8%	53.6% ±0.3%	54.1% ±0.5%	54.3% ±0.4%
FedAvgM	50.9% ± 1.3%	53.3% ±1.4%	53.8% ±1.2%	54.5% ±1.2%	54.9% ±1.3%
FedMedian	**52.6% ±1.3%**	53.7% ±1.2%	54.2% ±1.3%	54.5% ±1.7%	54.9% ±1.7%
Fedloss	52.5% ±1.6%	**53.8% ±1.0%**	**54.4% ±0.9%**	**54.7% ±1.0%**	**55.2% ±1.0%**

Table 2: Results of the methods per 10 rounds in a homogeneous setting for the CIFAR10 task.

For the CIFAR10 task it seems that in the 10th round the FedMedian with mean accuracy 52.6% outperforms the other algorithms. Fedloss in second place scores 52.5%. At the end of the next 10 rounds, Fedloss comes first with 53.8% with mean accuracy while the FedMedian achieves 53.7% while FedAvM surpass FedAvg with 53.3% and 52.6% respectively. In the 30th round Fedloss remains at the top with

54.7% mean accuracy, FedMedian follows with 54.2% In the 40th round Fedloss keeps the first position with 54.7% as the FedMedian ties with FedAvM with 54.5%. At the final round (50) Fedloss finished with 55.2%, FedAvgM and FedMedian shared second place with 54.9%. For the homogeneous scenario Fedloss appears to perform at least as well as the baselines which allow us to proceed the evaluation of the method in more complex (and therefore realistic) non-iid settings.

4.2 Label skew

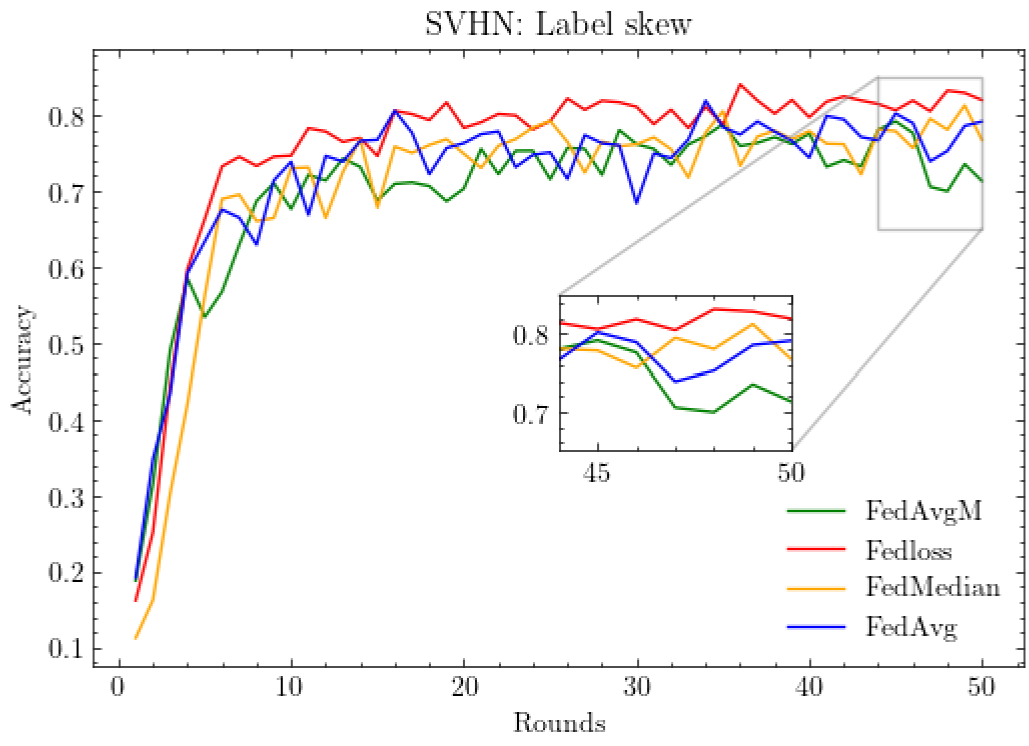

Figure 14: Results for the SVHN task in label skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	73.8% ±2.3%	76.3% ±6.3%	68.4% ±4.6%	74.4% ±5.2%	79.1 ±3.4%
FedAvgM	67.6% ± 5.0%	70.3% ±5.3%	76.2% ±5.8%	77.6% ±4.2%	71.3% ±2.6%
FedMedian	73.0% ±6.0%	74.9% ±8.3%	76.1% ±2.7%	77.9% ±5.1%	76.7% ±5.5%
Fedloss	**74.7% ±2.1%**	**78.3% ±4.2%**	**81.1% ±3.6%**	**79.7% ±7.6%**	**82.0% ±1.1%**

Table 3: Results of the methods per 10 rounds in label skew setting for the SVHN task.

Unlike the homogenous case, in this setting we can see that the training process presents higher variations. Fedloss manages to mitigate the bias effect if we compare its peaks and valleys against the rest of the algorithms.In order to enhance our analysis we introduce the best accuracy for each algorithm. FedAvg's top accuracy is 82.9% at the 34th round, FedAvgM's is 79.2% at the 45th round, FedMedian's is 81.3% at 49th round. We observe that the Fedloss outshines the baseline algorithms throughout the entire process and achieves top accuracy.

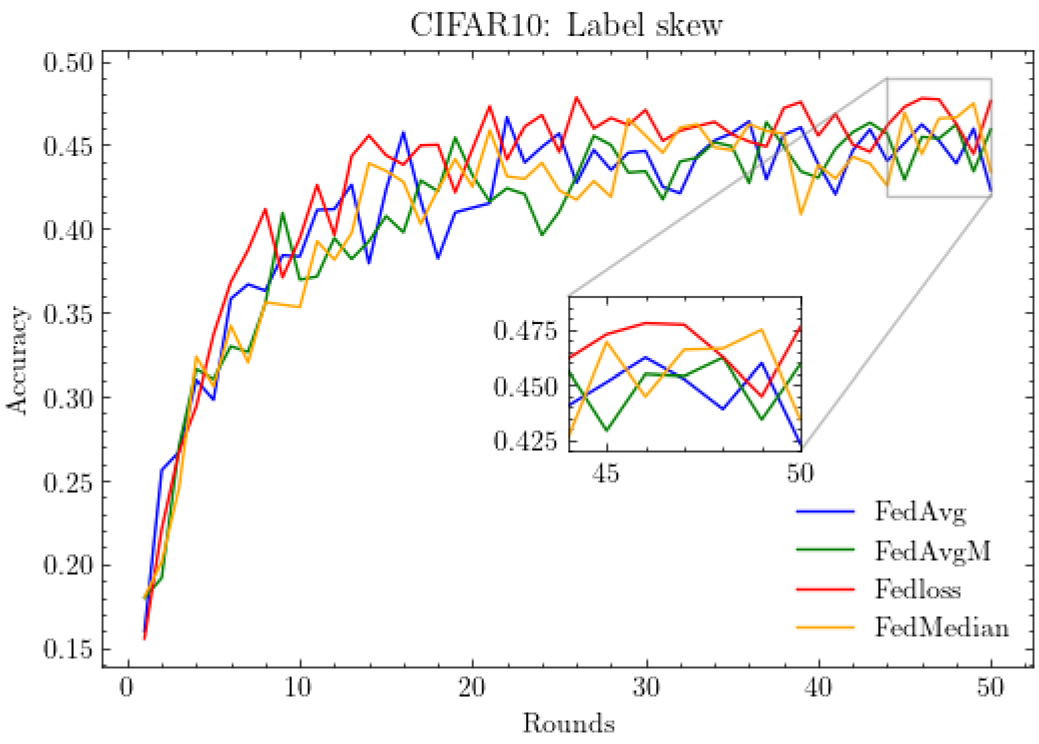

Figure 15: Results for the CIFAR10 task in label skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	38.3% ±3.6%	41.2% ±2.6%	44.6% ±4.4%	43.9% ±4.2%	42.3 ±8.4%
FedAvgM	37.0% ± 5.7%	43.2% ±2.2%	43.4% ±3.4%	43.0% ±5.3%	45.9% ±3.1%
FedMedian	35.3% ±6.0%	42.5% ±5.7%	45.6% ±4.0%	43.8% ±3.9%	43.4% ±5.3%
Fedloss	**39.4% ±5.4%**	**44.8% ±2.5%**	**47.1% ±1.8%**	**45.6% ±3.7%**	**47.6%±1.8%**

Table 4: Results of the methods per 10 rounds in label skew setting for the CIFAR10 task.

Similar to the preceding case, Fedloss beats the baseline algorithms in the majority of the training rounds. In this setting Fedloss's best accuracy is 47.8% at round 46, FedAvg's is 46.7% at round 22, FedAvgM's is 46.4% at round 37 while FedMedian's is 47.5% at round 49. FedAvg doesn't take into account the label skewness and it fails to retain high accuracy during the training process. In this scenario the validation loss of Fedloss gives us a measure of the skewness distance between the global model that the client downloads and the local model that produces after each training step. By aggregating the local models based on the validation loss we promote the models that have greater “distance” with the current global model so the next global model to be “closer” to these clients. This is a clear demonstration that our initial intuition analyzed in Sec 3.1, allows Fedloss to mitigate non-IIDness more efficiently, compared to SotA algorithms, for the specific settings.

4.3 Quantity skew

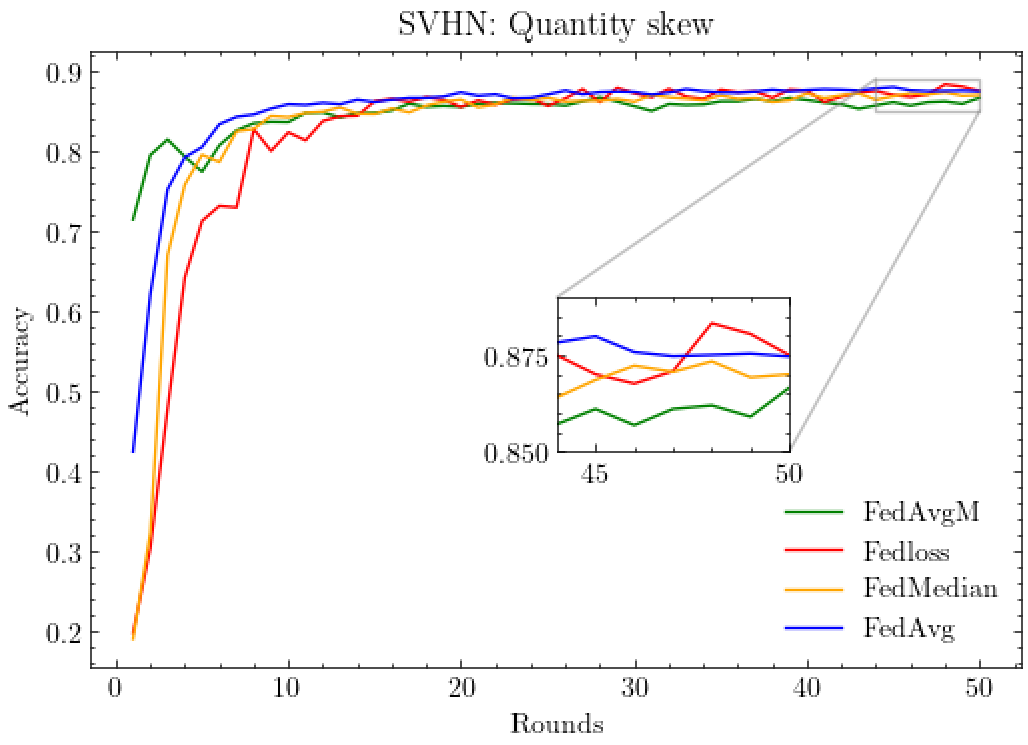

Figure 16: Results for the SVHN task in quantity skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	**85.9% ±1.8%**	**87.3% ±0.6%**	**87.4% ±0.4%**	**87.6% ±0.7%**	**87.5% ±0.4%**
FedAvgM	83.6% ± 1.7%	85.5% ±0.9%	85.6% ±1.2%	86.4% ±0.8%	86.7% ±0.5%
FedMedian	84.2% ±1.4%	86.4% ±0.5%	86.2% ±1.7%	87.2% ±0.6%	87.0% ±0.5%
Fedloss	82.3% ±4.7%	85.5% ±2.2%	87.2% ±1.8%	87.5% ±0.7%	**87.5%±1.0%**

Table 5: Results of the methods per 10 rounds in quantity skew setting for the SVHN task.

As expected, in this scenario FedAvg excels in both accuracy and stability. Despite the fact that Fedloss for the 10 first rounds falls behind, after the 20th round it achieves similar accuracy with FedAvg. Note though that quantity skew (an assumption that FedAvg is based on) is not a realistic setting in most FL environments. In any case, a future extension of our algorithm can be investigated to address this rare case as well, by e.g., taking into consideration the client data samples during aggregation (similar to FedAvg).

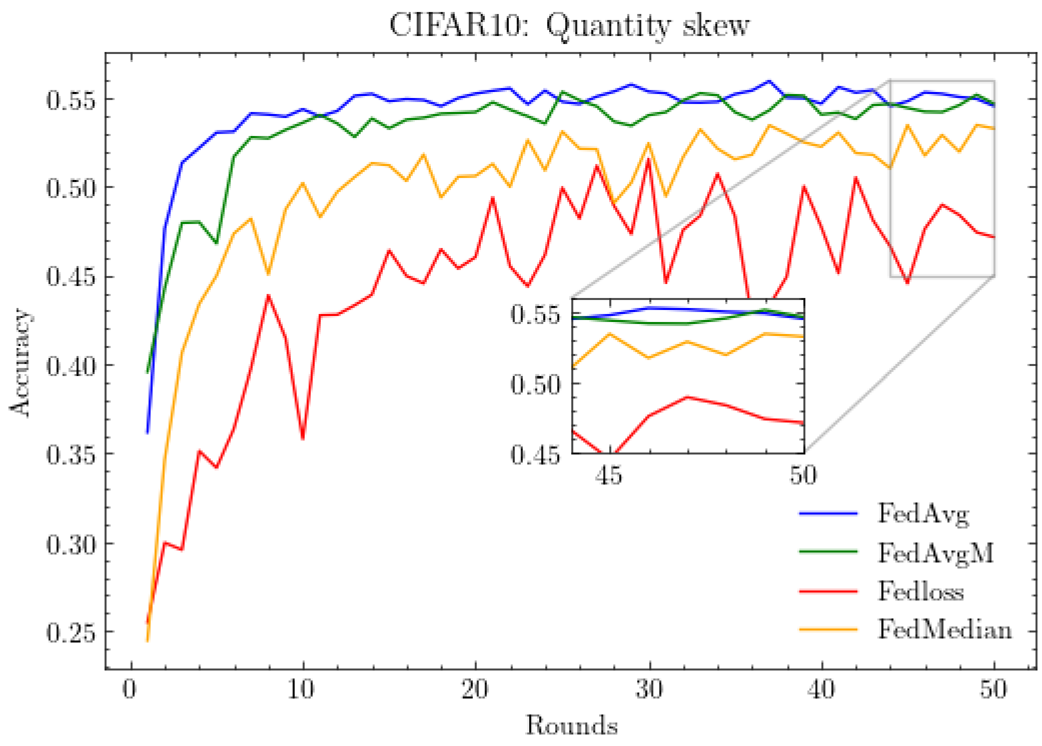

Figure 17: Results for the CIFAR10 task in quantity skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	**54.4% ±1.8%**	**55.2% ±1.0%**	**55.3% ±1.0%**	**54.7% ±0.9%**	54.5% ±0.7%
FedAvgM	53.6% ± 1.0%	54.2% ±2.0%	54.0% ±1.3%	54.1% ±0.9%	**54.7% ±0.9%**
FedMedian	50.2% ±1.9%	50.6% ±2.0%	52.4% ±2.0%	52.2% ±1.2%	53.3%±0.9%
Fedloss	35.8%±13.4%	46.0% ±7.0%	51.6% ±4.0%	47.7% ±9.1%	47.2%±10.1%

Table 6: Results of the methods per 10 rounds in quantity skew setting for the CIFAR10 task.

FedAvg and FedAvgM dominate in this scenario, too as expected. FedAvg and its variant (FedAvgM) are both designed for this non-IID scenario. FedAvg aggregates the local model using their respective local dataset size. On the contrary Fedloss takes no action against this setting. This scenario is not the usual case but it points out Fedloss's weakness. An important note is that FedAvg is stable under quantity skewness compared to FedMedian and Fedloss.

4.4 Feature skew

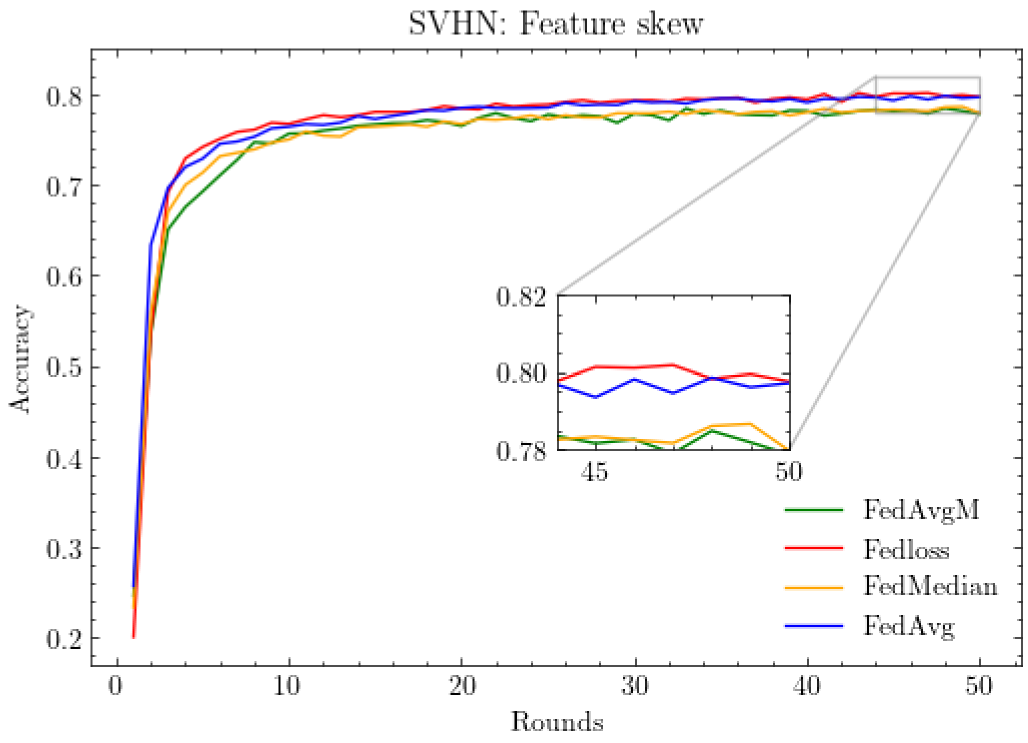

Figure 18: Results for the SVHN task in feature skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	76.4% ±1.2%	**78.5% ±0.5%**	**79.3% ±0.4%**	79.1% ±0.4%	79.7% ±0.6%
FedAvgM	75.7% ± 0.4%	76.5% ±1.5%	77.9% ±0.8%	78.2% ±0.5%	77.9% ±1.2%
FedMedian	75.0%±1.6%	76.7% ±1.1%	77.9% ±0.9%	78.0% ±0.8%	78.0%±0.7%
Fedloss	**76.8%±0.7%**	**78.5% ±0.6%**	**79.3% ±0.6%**	**79.5% ±0.5%**	**79.8%±0.7%**

Table 7: Results of the methods per 10 rounds in feature skew setting for the SVHN task.

In this scenario, we observe that Feature skewness doesn't affect the stability of the training process compared to Label skewness. Secondly, we see that the FedAvg and

Fedloss perform similarly, outperforming the other two methods. Furthermore, we notice that Fedloss converges faster until round 18.

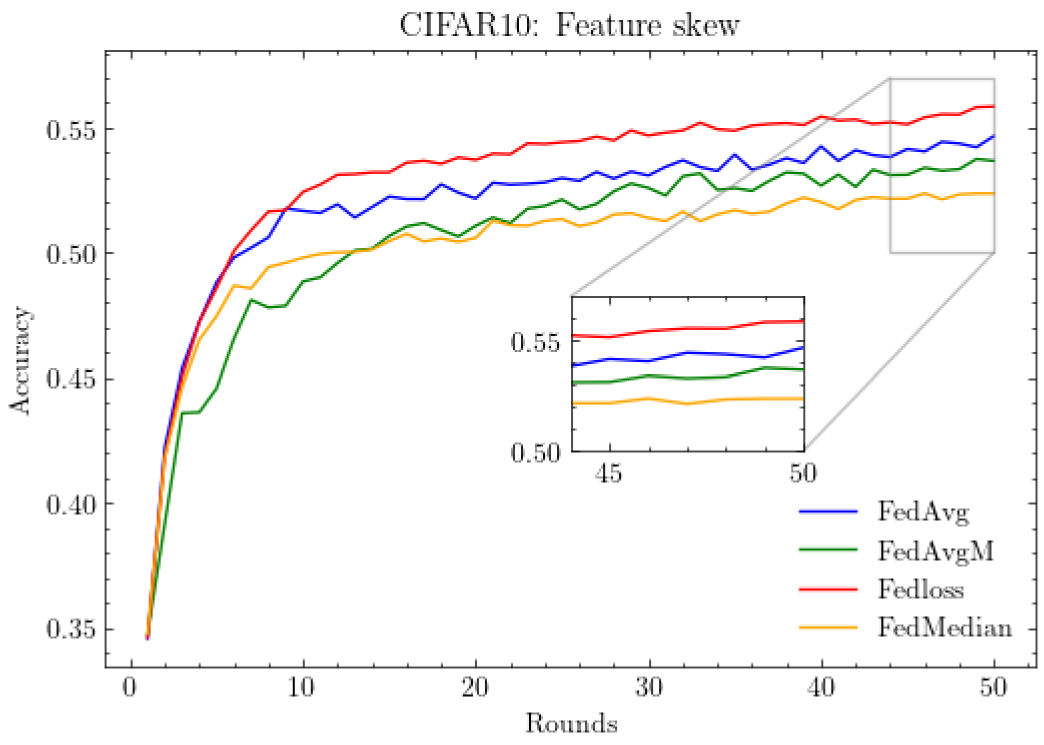

Figure 19: Results for the CIFAR10 task in feature skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	51.7% ±1.9%	52.2% ±1.1%	53.1% ±1.5%	54.3% ±1.2%	54.7% ±1.2%
FedAvgM	48.8% ± 2.2%	51.1% ±2.1%	52.6% ±1.9%	52.7% ±2.2%	53.7% ±1.8%
FedMedian	49.8% ±2.0%	50.6% ±1.9%	51.4% ±2.0%	52.0% ±1.8%	52.4%±1.5%
Fedloss	**52.4%±2.1%**	**53.7% ±2.3%**	**54.7% ±2.8%**	**55.5% ±2.3%**	**55.9%±2.1%**

Table 8: Results of the methods per 10 rounds in feature skew setting for the feature task.

For the first eight rounds FedAvg and Fedloss converge at the same rate. After the 10th round it is clear that FedAvg retreats. Also, Fedloss and FedMedian appear to have more stable behavior. The key take-away point is therefore that Fedloss achieves a general better performance by promoting the local models that have been trained at the most difficult datasets, during aggregation..

4.5 Mix skew

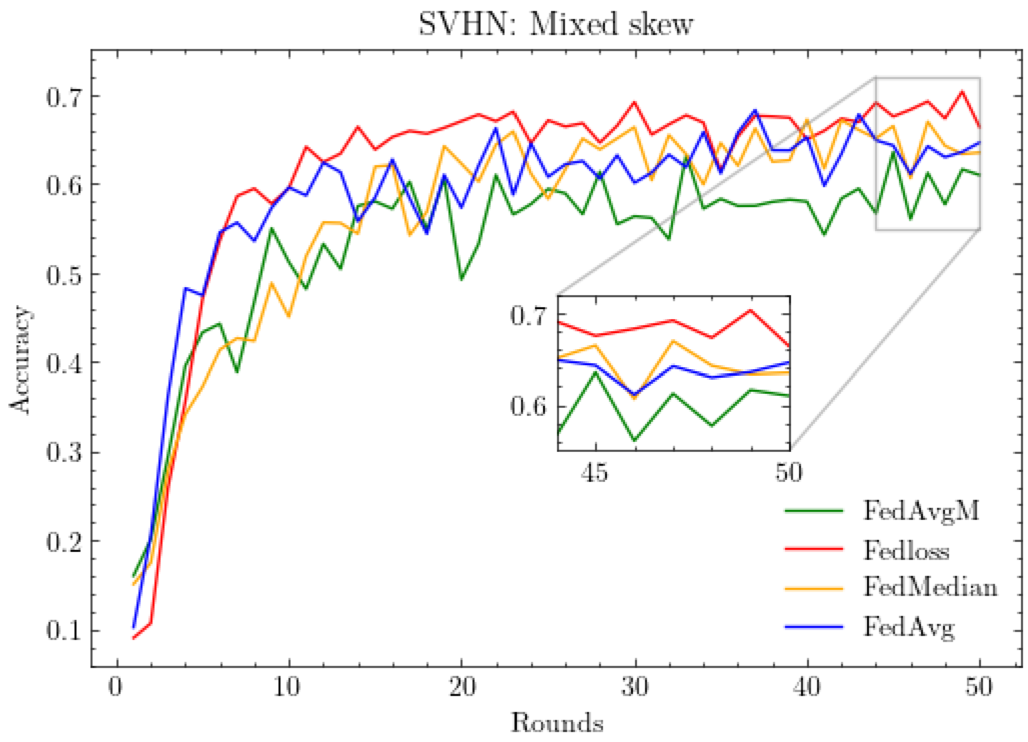

Figure 20: Results for the SVHN task in mixed skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	**59.6% ±4.9%**	57.3%±11.8%	60.1% ±6.2%	65.2% ±7.8%	64.6% ±5.6%
FedAvgM	51.2% ± 6.2%	49.2% ±9.9%	56.4% ±6.0%	58.0% ±9.0%	61.0% ±3.7%
FedMedian	45.1%±20.0%	62.2% ±6.3%	66.4% ±3.5%	**67.2% ±1.6%**	63.5%±6.3%
Fedloss	59.5%±4.9%	**67.0% ±4.1%**	**69.2% ±2.1%**	65.0% ±6.4%	**66.4%±3.3%**

Table 9: Results of the methods per 10 rounds in mix skew setting for the SVHN task.

In the mix skew setting we combined label skew and feature skew in order to evaluate our proposed method in an even more challenging scenario. At the first 10 rounds FedAvg performs better than the others but for the rest of the rounds Fedloss surpasses the baselines. FedAvg scores top accuracy 68.3% at the 37th round, FedAvgM scores 63.6% at the 45th round, FedMedian's is 67.2 at the 40th round, Fedloss outperforms the other with 70,4% at the 40th round.

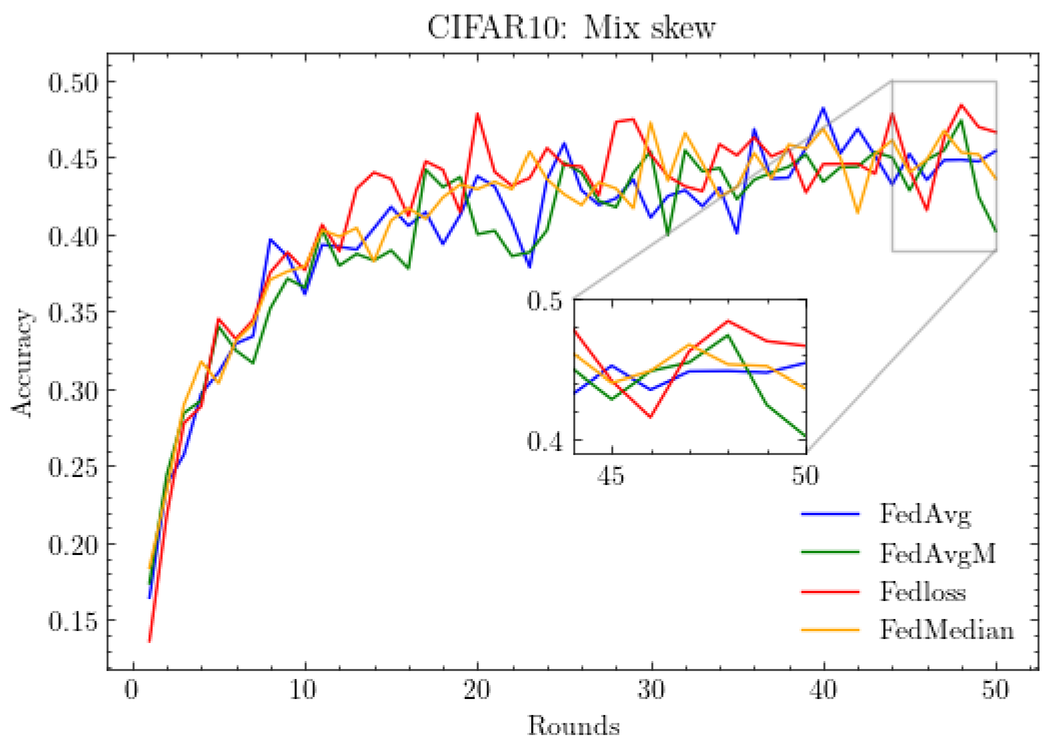

Figure 21: Results for the CIFAR10 task in mixed skew setting.

Mean accuracy ± standard deviation

Algo\Round	10	20	30	40	50
FedAvg	36.3% ±4.0%	44.2%±4.0%	41.0% ±2.5%	**47.8% ±3.7%**	45.6% ±5.4%
FedAvgM	36.9% ± 3.6%	39.9% ±1.8%	45.4% ±3.3%	43.1% ±1.8%	39.8% ±7.0%
FedMedian	**38.1%±2.2%**	43.1% ±2.3%	**47.5% ±3.8%**	47.1% ±2.8%	43.6%±5.5%
Fedloss	38.0%±6.7%	**48.0% ±2.1%**	45.4% ±5.2%	44.7% ±5.3%	**46.8%±1.1%**

Table 10: Results of the methods per 10 rounds in mix skew setting for the CIFAR10 task.

In our last scenario with CIFAR10 under mix skew the limits are indistinguishable and no method clearly outperforms the others. However, throughout the training process we see that Fedloss performs at least as well as the others and achieves the best accuracy 48.4% at the 50th round while the others achieve: FedAvg 47.8% at 40th round, FedAvgM 47.5% at 48th round, FedMedian 47.5% at 30th round.

The above experiments provide strong evidence that Fedloss is a method that can perform well under most of these challenging scenarios. Note that on average Fedloss outperforms all other algorithms by 2%, across all scenarios and datasets. Feature skew and label skew are the settings whereFedloss excels, while a quantity skew setting can be proved a very challenging task for the method. CIFAR10 seems to be a much more complex task compared to the SVHN for all algorithms in the Federated Learning setting.

Chapter 5

Conclusions

5.1 Summary

To sum up, this thesis investigates the emerging field of Federated Learning, a promising Machine Learning paradigm which enables the development of Machine Learning models in a distributed manner. The original motivation for the development of Federated learning is to address the increasing privacy concerns. Initially, we outlined FedAvg, the fundamental algorithm of Federated learning. Secondly we illustrated the main challenge that accompanies the Federated learning framework: bias introduced by data asymmetry i.e., Non-IIDness. An in-depth analysis was given as to how non-IID is expressed in an FL setting, as well as how bias degrades the performance of FL models. The basic non-IID scenarios were presented.

In this context we introduced Fedloss, our novel bias mitigation algorithm. Fedloss was benchmarked in a custom FL framework based on Flower, an FL simulation environment, in order to evaluate our proposed method under various non-iid settings. Our results against State-of-the-Art FL mitigation algorithms suggest that Fedloss consistently outperforms existing solutions, while preserving the strict privacy requirement of FL and in parallel without introducing any complexity in the resource-constrained client devices.

5.2 Future work

For future research, we would like to investigate the combination of Fedloss with FedAvg, as a means of harnessing the benefits from both algorithms. Also, more experiments in different ML tasks such as text prediction, speech recognition etc. should be conducted in order to validate the generalization of our results. Another direction is the deployment of our solution in an actual environment, to therefore study its behavior in a cross-device setting with unreliable clients.

Appendix

Tuning

In order to set the hyperparameters we performed a grid search for both tasks in IID settings. The grid: learning rate {0.01, 0.001, 0.0001}, batch size {32, 64, 128} local epochs {1, 10, 25}

CIFAR10

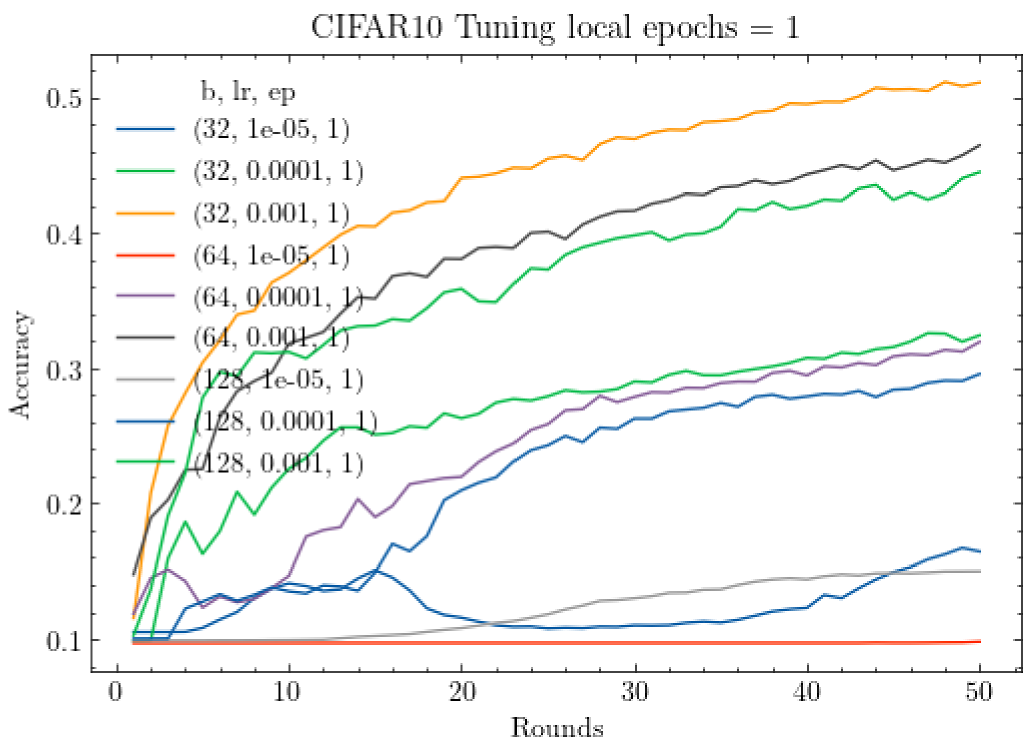

Figure 22: Tuning CIFAR10 local epochs = 1

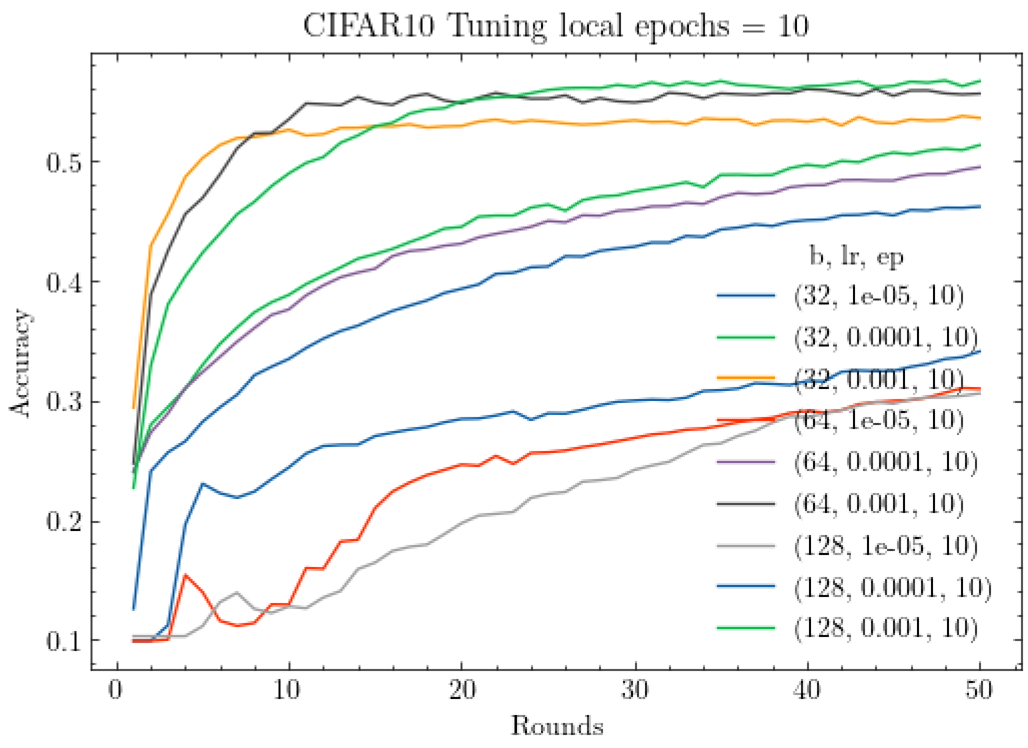

Figure 23: Tuning CIFAR10 local epochs = 10

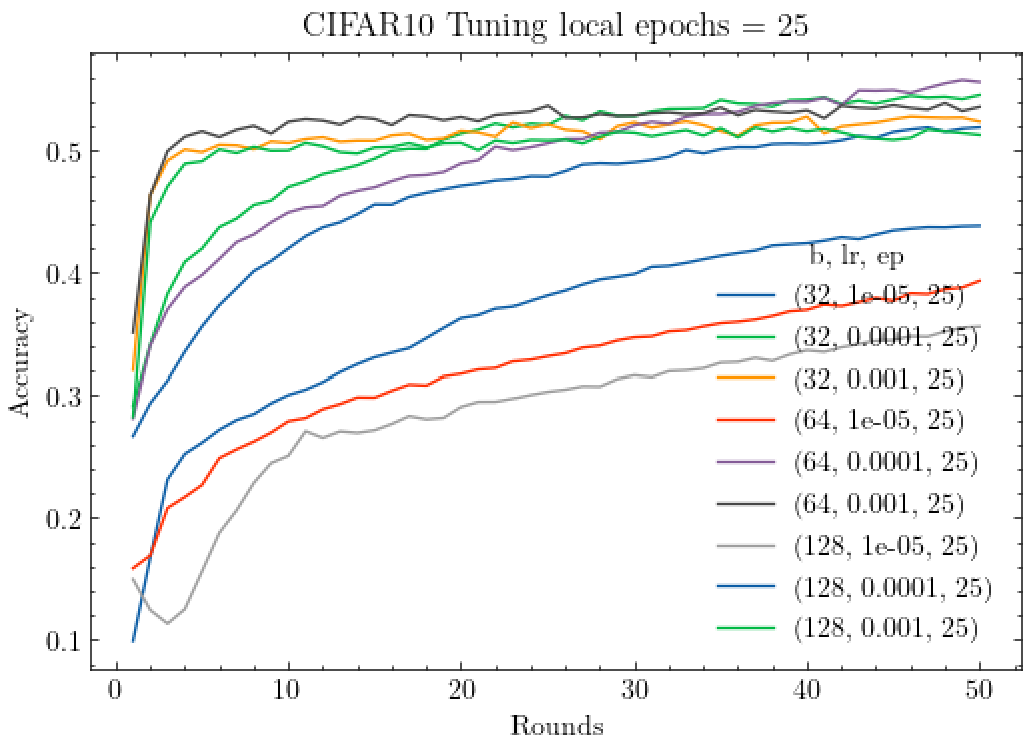

Figure 24: Tuning CIFAR10 local epochs = 25

SVHN

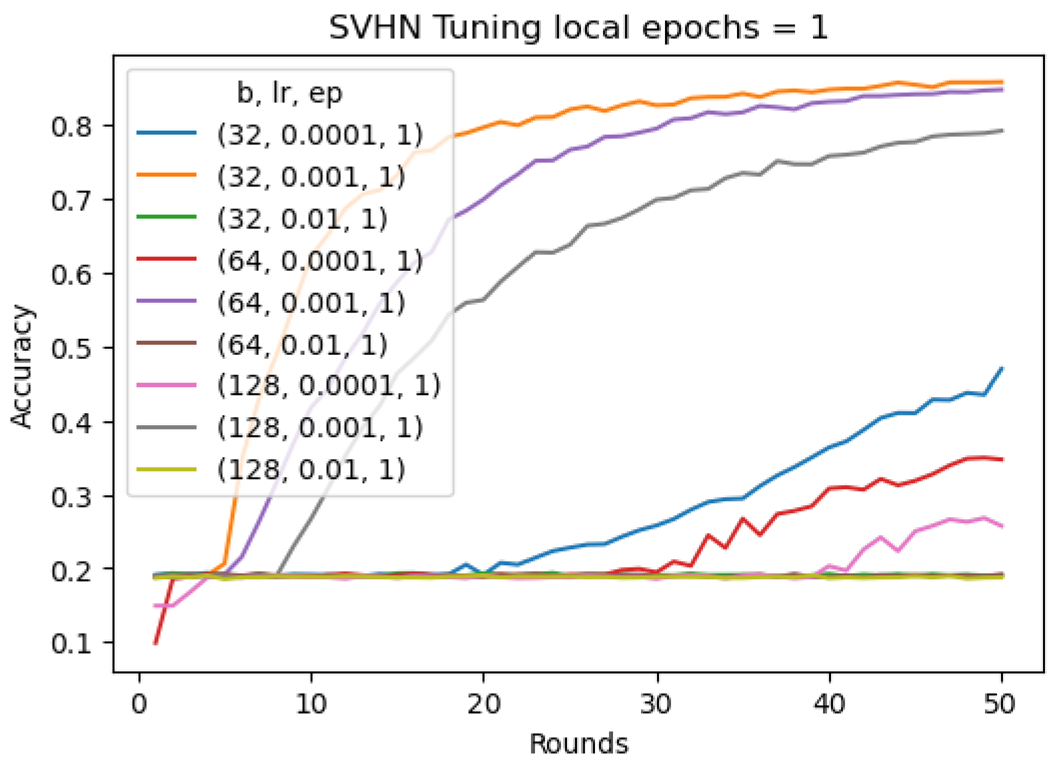

Figure 25: Tuning SVHN local epochs = 1

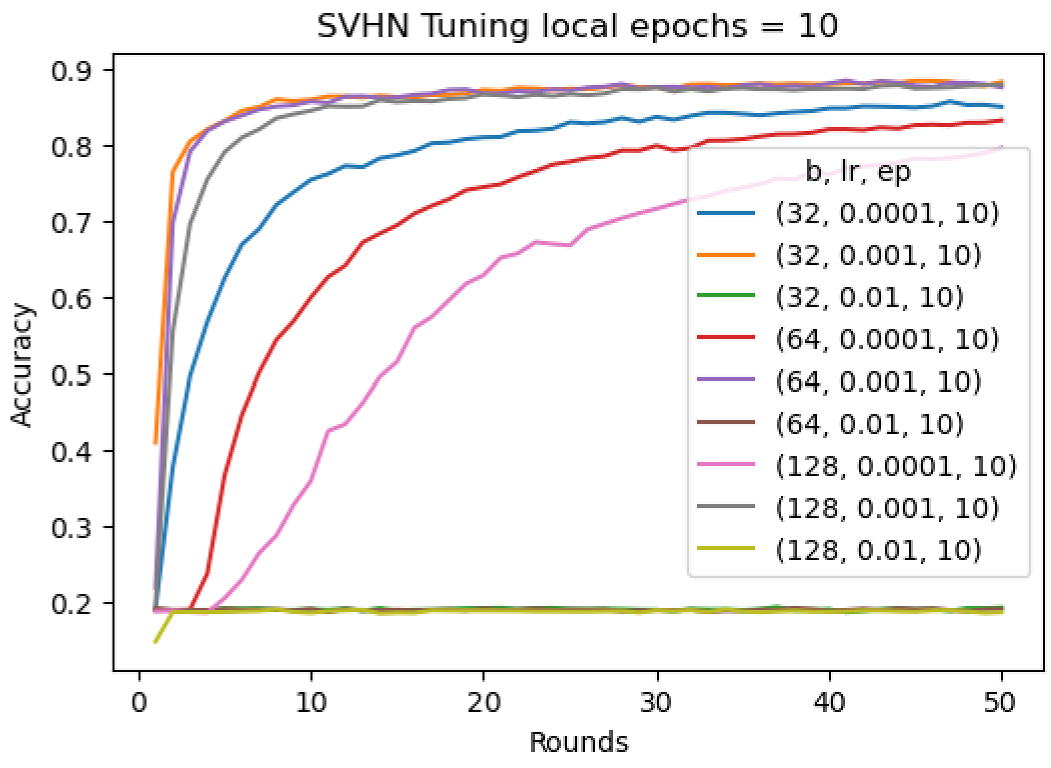

Figure 26: Tuning SVHN local epochs = 10

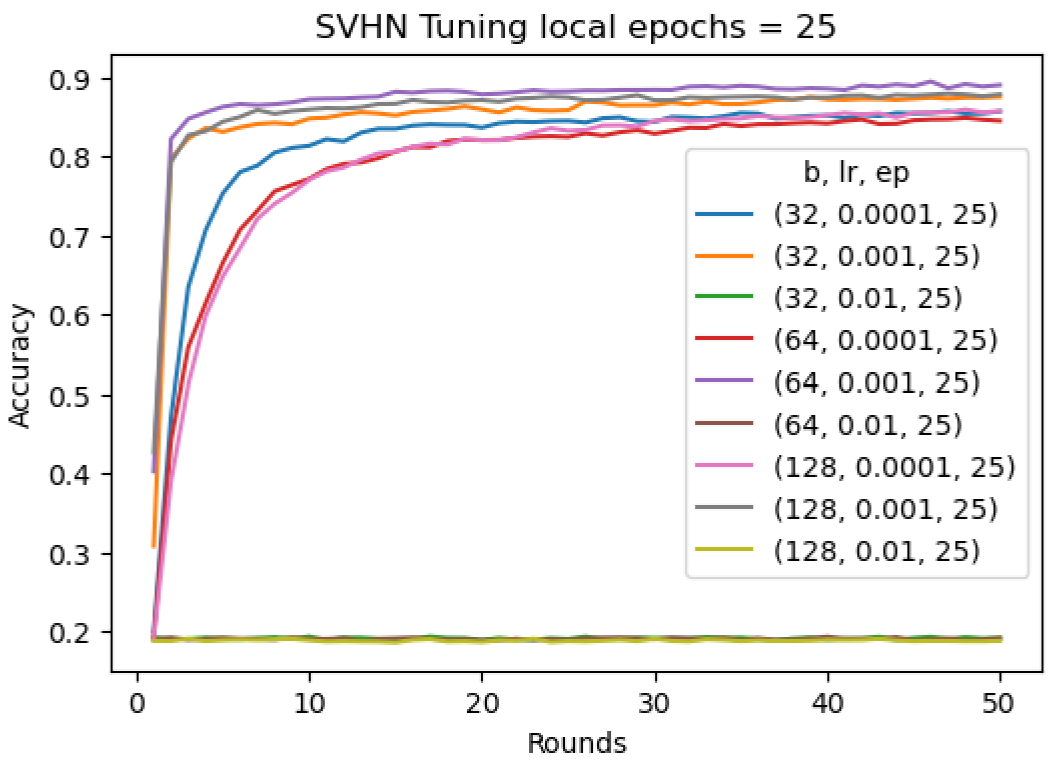

Figure 27: Tuning SVHN local epochs = 25

Bibliography

[1] "Centralized_federated_learning_protocol." https://upload.wikimedia.org/wikipedia/commons/1/11/Centralized_federated_learning_protocol.png (accessed Mar. 06, 2023).

[2] Business Wire, "Federated learning (Graphic: Business Wire)," *Businesswire*. https://mms.businesswire.com/media/20220208005049/en/1349465/5/img_296918_2.jpg?download=1 (accessed Mar. 06, 2023).

[3] "General Data Protection Regulation (GDPR) – Official Legal Text," *General Data Protection Regulation (GDPR)*. https://gdpr-info.eu/ (accessed Mar. 15, 2023).

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data." arXiv, Jan. 26, 2023. Accessed: Mar. 06, 2023. [Online]. Available: http://arxiv.org/abs/1602.05629

[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," 2018, doi: 10.48550/arXiv.1806.00582.

[6] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition"," *Dokl Akad Nauk SSSR*, vol. 114, no. 5, pp. 953–956, 1957.

[7] S. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4–5, pp. 185–196, 1993.

[8] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning." arXiv, Mar. 08, 2021. Accessed: Feb. 02, 2023. [Online]. Available: http://arxiv.org/abs/1912.04977

[9] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous Federated Learning on Heterogeneous Devices: A Survey." arXiv, Aug. 11, 2022. Accessed: Mar. 29, 2023. [Online]. Available: http://arxiv.org/abs/2109.04269

[10] A. B. Mansour, G. Carenini, A. Duplessis, and D. Naccache, "Federated Learning Aggregation: New Robust Algorithms with Guarantees." arXiv, Jul. 18, 2022. Accessed: Feb. 12, 2023. [Online]. Available: http://arxiv.org/abs/2205.10864

[11] S. Chandran, "Significance of I.I.D in Machine Learning," *Medium*, Jun. 06, 2021. https://medium.datadriveninvestor.com/significance-of-i-i-d-in-machine-learning-281da0d0cbef (accessed Mar. 07, 2023).

[12] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-IID data in Federated Learning," *Future Gener. Comput. Syst.*, vol. 135, pp. 244–258, Oct. 2022, doi: 10.1016/j.future.2022.05.003.

[13] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated Learning on Non-IID Data: A Survey." arXiv, Jun. 12, 2021. Accessed: Feb. 02, 2023. [Online]. Available: http://arxiv.org/abs/2106.06843

[14] M. Shin, C. Hwang, J. Kim, J. Park, M. Bennis, and S.-L. Kim, "XOR Mixup: Privacy-Preserving Data Augmentation for One-Shot Federated Learning." arXiv, Jun. 09, 2020. Accessed: Mar. 15, 2023. [Online]. Available: http://arxiv.org/abs/2006.05148

[15] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Toronto, ON, Canada: IEEE, Jul. 2020, pp. 1698–1707. doi: 10.1109/INFOCOM41043.2020.9155494.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks." arXiv, Apr. 21, 2020. Accessed: Mar. 15, 2023. [Online]. Available: http://arxiv.org/abs/1812.06127

[17] F. Varno, M. Saghayi, L. Rafiee Sevyeri, S. Gupta, S. Matwin, and M. Havaei, "AdaBest: Minimizing Client Drift in Federated Learning via Adaptive Bias

Estimation," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., in Lecture Notes in Computer Science, vol. 13683. Cham: Springer Nature Switzerland, 2022, pp. 710–726. doi: 10.1007/978-3-031-20050-2_41.

[18] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning." arXiv, Apr. 09, 2021. Accessed: Mar. 15, 2023. [Online]. Available: http://arxiv.org/abs/1910.06378

[19] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification." arXiv, Sep. 13, 2019. Accessed: Mar. 08, 2023. [Online]. Available: http://arxiv.org/abs/1909.06335

[20] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates." arXiv, Feb. 25, 2021. Accessed: Mar. 08, 2023. [Online]. Available: http://arxiv.org/abs/1803.01498

[21] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated Learning on Non-IID Data Silos: An Experimental Study." arXiv, Oct. 28, 2021. Accessed: Jan. 17, 2023. [Online]. Available: http://arxiv.org/abs/2102.02079

[22] "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges." http://yann.lecun.com/exdb/mnist/ (accessed Mar. 09, 2023).

[23] "The Street View House Numbers (SVHN) Dataset." http://ufldl.stanford.edu/housenumbers/ (accessed Mar. 14, 2023).

[24] "CIFAR-10 and CIFAR-100 datasets." https://www.cs.toronto.edu/~kriz/cifar.html (accessed Mar. 14, 2023).

[25] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC*, IEEE, 2019.

[26] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN*, IEEE, 2018.

[27] D. J. Beutel *et al.*, "Flower: A Friendly Federated Learning Research Framework." arXiv, Mar. 05, 2022. Accessed: Mar. 22, 2023. [Online]. Available: http://arxiv.org/abs/2007.14390

[28] D. Zeng, S. Liang, X. Hu, H. Wang, and Z. Xu, "FedLab: A Flexible Federated Learning Framework." arXiv, Apr. 22, 2022. Accessed: Mar. 14, 2023. [Online]. Available: http://arxiv.org/abs/2107.11621