National Technical University of Athens
School of Electrical and Computer Engineering
Division of Data Science and Machine Learning

Towards Understanding Privacy-Aware Artificial Intelligence

From Intuition to Application

Diploma Thesis

of

ANDREAS TRITSAROLIS

Supervisor: **Stefanos Kollias**
Professor

Co-Supervisors:
George Siolas
Laboratory Teaching Staff
Yannis Theodoridis
Professor

Athens, June 16, 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Data Science and Machine Learning

Towards Understanding Privacy-Aware Artificial Intelligence

From Intuition to Application

Diploma Thesis

of

ANDREAS TRITSAROLIS

Supervisor: **Stefanos Kollias**
Professor

Co-Supervisors:
George Siolas
Laboratory Teaching Staff
Yannis Theodoridis
Professor

Approved by the examination committee on June 16, 2022.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Stefanos Kollias	Yannis Theodoridis	George Stamou
Professor	Professor	Professor

Athens, June 16, 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Data Science and Machine Learning

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Andreas Tritsarolis
June 16, 2022

Περίληψη

Η Τεχνητή Νοημοσύνη (Artificial Intelligence – AI), και πιο συγκεκριμένα η Μηχανική Μάθηση (Machine Learning – ML), υιοθετείται ευρέως σε πολλές πτυχές της καθημερινότητάς μας, με τα δεδομένα να παίζουν καθοριστικό ρόλο στην επιτυχία της. Καθώς οι συσκευές Internet of Things (IoT) παράγουν τεράστιες ποσότητες δεδομένων με ιλλιγιώδεις ταχύτητες, προκύπτει μια πρόκληση όσον αφορά τα ζητήματα ιδιωτικότητας-απορρήτου και τους υπολογιστικούς πόρους. Ορισμένα δεδομένα ενχεδομένως να περιέχουν ευαίσθητες πληροφορίες (π.χ. ιατρικά αρχεία) και να μη δύνανται να είναι διαθέσιμα σε ανοικτά αποθετήρια, ενώ άλλα είναι τόσο ογκώδη που δεν μπορούν να χρησιμοποιηθούν σε ένα σύστημα κεντρικοποιημένης αρχιτεκτονικής για την εκπαίδευση ενός μοντέλου. Η Ομοσπονδιακή Μάθηση (Federated Learning – FL) προτάθηκε από την Google το 2016 [44, 45], προκειμένου να αντιμετωπίσει τις παραπάνω προκλήσεις. Εν ολίγοις, είναι μια προσέγγιση που επιτρέπει σε ένα μοντέλο να εκπαιδευτεί σε δεδομένα που δεν κατέχουμε και δεν μπορούμε να δούμε. Τα δεδομένα που δημιουργούνται από edge devices (π.χ. smartphone) αποθηκεύονται τοπικά και δεν κοινοποιούνται σε άλλους κόμβους στο δίκτυο ή τον κεντρικό διακομιστή [11, 42]. Αντίθετα, μόνο οι ενημερώσεις των τοπικών μοντέλων κοινοποιούνται και συγκεντρώνονται προκειμένου να δημιουργηθεί ένα ενιαίο μοντέλο. Σε αυτή τη διατριβή, κατανοούμε την Ομοσπονδιακή Μάθηση τόσο θεωρητικά όσο και αλγοριθμικά. Επιπλέον, συγκρίνουμε τα πλεονεκτήματα και τα μειονεκτήματά της με την κεντρικοποιημένη προσέγγιση, στο πλαίσιο της Πρόβλεψης Διαδρομής Πλοίου (Vessel Route Forecasting) και Κυκλοφοριακής Ροής (Vessel Traffic Flow Forecasting).

Λέξεις Κλειδιά

Μηχανική Μάθηση, Διατήρηση Απορρήτου-Ιδιωτικότητας, Ομοσπονδιακή Μάθηση, Αναλυτική Κινούμενων Δεδομένων, Πρόβλεψη Διαδρομής Πλοίου, Πρόβλεψη Κυκλοφοριακής Ροής Πλοίων

Σύνοψη

Η εξάπλωση των Internet of Things (IoT) συσκευών, όπως (βιο-)αισθητήρες, smartwatches, smartphones και GPS trackers, έχουν οδηγήσει στην παραγωγή τεράστιων ποσοτήτων ιατρικών, κινηματικών και πολλών άλλων τύπων δεδομένων. Η διαθεσιμότητα των εν λόγω δεδομένων έχει ζωτικό ρόλο στην επιτυχία των τεχνολογιών Μηχανικής Μάθησης (Machine Learning – ML), οι οποίες μπορούν να εκτελέσουν μια ποικιλία εργασιών που μερικές φορές μπορεί να υπερβαίνουν την ανθρώπινη ικανότητα [78]. Ωστόσο, τα δεδομένα που παράγονται από τα edge devices είναι εκ φύσεως ευαίσθητα (π.χ. ιατρικά αρχεία, πληροφορίες πλοίων κ.λπ.), με ουκ ολίγες φορές να είναι κατανεμημένα σε πολλά μέρη. Αυτές οι ιδιότητες θέτουν νέες προκλήσεις όσον αφορά την αποτελεσματική αποθήκευση, ανάλυση και εξαγωγή γνώσης από τέτοια δεδομένα.

Η κεντρικοποίηση των δεδομένων σε μια συγκεκριμένη τοποθεσία (π.χ. κέντρο δεδομένων) δύναται να καταστεί ιδιαίτερα περίπλοκη εργασία, λόγω του υψηλού κόστους αποθήκευσης/bandwidth (π.χ., ένας στόλος πλοίων τύπου AIS αναμένεται να παράγει πολλά TB δεδομένων σε καθημερινή βάση). Επιπλέον, λόγω κανονισμών όπως το GDPR[1], η συλλογή και κοινή χρήση ευαίσθητων δεδομένων μπορεί να γίνει αρκετά δύσκολη, αν όχι αδύνατη, αναγκάζοντας έτσι τα δεδομένα να υπάρχουν σε μεμονωμένες αποθήκες δεδομένων που διατηρούνται από τους αντίστοιχους ιδιοκτήτες/εταίρους. Εναλλακτικά, η ανάθεση της διαδικασίας εκπαίδευσης στα edge devices και/ή στις αποθήκες δεδομένων, έτσι ώστε κάθε εταίρος να μπορεί να χρησιμοποιήσει ένα μοντέλο ML χρησιμοποιώντας τα δικά του δεδομένα, μπορεί να επηρεάσει την απόδοση των μοντέλων, οδηγώντας είτε σε υποβέλτιστη απόδοση (π.χ. υπό-προσαρμογή) είτε σε μεροληπτική (biased) κατανομή στόχου (π.χ. υπέρ-προσαρμογή), ανάλογα με το μέγεθος του συνόλου δεδομένων και την κατανομή των χαρακτηριστικών, αντίστοιχα.

Προκειμένου να επιλυθούν οι παραπάνω προκλήσεις και να εκπαιδευτεί ένα μοντέλο Μηχανικής Μάθησης που δεν βασίζεται στη συλλογή όλων των δεδομένων σε μια κεντρική αποθήκευση, οι McMahan et al. [45] και Konečný et al. [33] προτείνουν την Ομοσπονδιακή Μάθηση (Federated Learning – FL), μια καινοτόμο μεθοδολογία Μηχανικής Μάθησης, όπου ένα κεντρικοποιημένο μοντέλο εκπαιδεύεται σε αποκεντρωμένα δεδομένα. Ενδελεχέστερα, κάθε edge device λαμβάνει ένα αρχικό μοντέλο από τον διακομιστή και προχωρά στην εκπαίδευσή του χρησιμοποιώντας τα αντίστοιχα (τοπικά) δεδομένα. Εν συνεχεία, όλα τα ενημερωμένα μοντέλα μεταφορτώνονται στον διακομιστή, δημιουργώντας έτσι ένα νέο, ενοποιημένο μοντέλο. Η επανάληψη της παραπάνω διαδικασίας για αρκετούς κύκλους, ενδέχεται να προκαλέσει σύγκλιση του εν λόγω μοντέλου, δημιουργώντας ένα ML μοντέλο το

[1] Προστασία δεδομένων. Ευρωπαϊκή Επιτροπή, https://ec.europa.eu/info/law/law-topic/data-protection_en, Τελευταία επίσκεψη: 01/07/2022

οποίο αποδίδει, τουλάχιστον, καλύτερα σε σχέση με το τι θα μπορούσε να μάθει κάθε μέρος από μόνο του. Ιδανικά, πρόκειται για μια προσέγγιση του ίδιου μοντέλου εάν εκπαιδευτεί με κεντρικοποιημένο τρόπο.

Χάρη στο FL, η αποκεντρωμένη φύση των δεδομένων διατηρείται, καθώς τα edge devices εκπαιδεύουν από κοινού ένα μοντέλο Μηχανικής Μάθησης, στέλνοντας μόνο τις ενημερώσεις (δηλαδή, τις παραγώγους) των μοντέλων στον κεντρικό διακομιστή. Εξαιτίας αυτού, κάθε εταίρος διατηρεί τον έλεγχο των δεδομένων του, καθώς, ουσιαστικά, τα δεδομένα δεν "φεύγουν" ποτέ από τη συσκευή, καθιστώντας δυσκολότερο για έναν εξωτερικό παρατηρητή να εξάγει οποιαδήποτε ευαίσθητη πληροφορία. Επιπλέον, μοιράζοντας τον φόρτο εργασίας της εκπαίδευσης σε πολλά edge devices, το FL επιτρέπει τη δημιουργία δυνητικά πιο έξυπνων μοντέλων, με μικρότερη λανθάνουσα καθυστέρηση συμπερασμού (inference latency), λιγότερη συνολική κατανάλωση ενέργειας, και, κατ' επέκταση, μικρότερο περιβαλλοντικό αντίκτυπο [52], διασφαλίζοντας παράλληλα το απόρρητο των δεδομένων.

Πέραν των παραπάνω πλεονεκτημάτων, επειδή το ενοποιημένο μοντέλο είναι κοινόχρηστο με όλα τα edge devices (π.χ., smartphones), δύναται να χρησιμοποιηθεί αμέσως, παρέχοντας μια εξατομικευμένη εμπειρία. Ένα χαρακτηριστικό παράδειγμα εφαρμογής FL είναι το Google Gboard [20]. Όταν το Gboard εμφανίζει μια προτεινόμενη λέξη, το smartphone αποθηκεύει τοπικά πληροφορίες σχετικά με το τρέχον περιβάλλον, καθώς και εάν όντως επιλέχθηκε η προτεινόμενη πρόταση ή όχι. Στη συνέχεια, χρησιμοποιώντας την προαναφερθείσα τεχνική, επεξεργάζεται το ιστορικό της συσκευής προκειμένου να προτείνει βελτιώσεις για τον επόμενο γύρο εκπαίδευσης του μοντέλου. Άλλα παραδείγματα από τη (σύγχρονη) καθημερινότητα περιλαμβάνουν τις εφαρμογές του Apple macOS/iOS, όπου το FL χρησιμοποιείται για την προστασία του απορρήτου των χρηστών, δημιουργώντας μοντέλα που στοχεύουν στην περαιτέρω βελτίωση της εμπειρίας χρήστη.

Λαμβάνοντας υπόψη τα παραπάνω, στόχος μας σε αυτή τη διατριβή είναι να διερευνήσουμε εφαρμογές τεχνητής νοημοσύνης με διασφάλιση του απορρήτου-ιδιωτικότητας, στο πλαίσιο της ανάλυσης δεδομένων ναυτιλίας. Συγκεκριμένα, επιλέγουμε μια πολύ κρίσιμη εργασία αναλυτικής, που ονομάζεται Πρόβλεψη Διαδρομής Πλοίων (Vessel Route Forecasting – VRF) και χρησιμοποιούμε μεθόδους FL προκειμένου να εκπαιδεύσουμε από κοινού ένα μοντέλο σε πολλαπλές αποθήκες δεδομένων (π.χ. πανεπιστήμια, εταιρείες, κ.λπ.), διασφαλίζοντας παράλληλα ότι δεν θα διαρρεύσουν ευαίσθητες πληροφορίες σε άλλους εταίρους εκτός του κατόχου των δεδομένων. Η εν λόγω εργασία είναι ιδιαίτερα σημαντική, καθώς δύναται να χρησιμοποιηθεί σε διάφορες πτυχές ασφάλειας της ναυτιλιακής κίνησης όπως, μεταξύ άλλων, αλιευτική προσπάθεια/πίεση, μελλοντικές συγκρούσεις, καθώς και συμπορευόμενα πρότυπα [68].

Άτυπα, δεδομένου ενός χρονικού διαστήματος Δt, στόχος μας είναι να προβλέψουμε τις μελλοντικές k τοποθεσίες ενός κινούμενου σκάφους μετά από χρόνο Δt. Το πρόβλημα που αντιμετωπίζουμε είναι αρκετά περίπλοκο καθώς, πέραν της εγγενούς δυσκολίας πρόβλεψης του μέλλοντος, πρέπει επιπλέον να ορίσουμε το πρωτόκολλο επικοινωνίας Ομοσπονδιακής Μάθησης, καθώς και μεθόδους για τη διασφάλιση του απορρήτου-ιδιωτικότητας των δεδομένων, διαδικασίες που δεν είναι προφανείς. Εξ όσων γνωρίζουμε, το πρόβλημα που στοχεύουμε να αντιμετωπίσουμε δεν έχει ακόμη αντιμετωπιστεί στη βιβλιογραφία.

Πολλές εφαρμογές που σχετίζονται με την κινητικότητα θα μπορούσαν να επωφεληθούν

από μια τέτοια εργασία ανάλυσης. Λόγω των μηχανισμών διασφάλισης του απορρήτου-ιδιωτικότητας, όχι μόνο τα edge devices (π.χ. πλοία τύπου AIS), αλλά και οι ιδιοκτήτες αποθηκών δεδομένων (π.χ. πανεπιστήμια) μπορούν να εκπαιδεύσουν από κοινού ένα μοντέλο Μηχανικής Μάθησης προκειμένου να δημιουργήσουν ένα "εξυπνότερο" μοντέλο, το οποίο εξατομικεύεται για να ταιριάζει στις ανάγκες κάθε εταίρου.

Εν κατακλείδι, η συνεισφορά μας αποτυπώνεται στα εξής:

- Παρέχουμε μια εις βάθος βιβλιογραφική ανασκόπηση σχετικά με την Κατανεμημένη και Ομοσπονδιακή Μάθηση, καθώς και μεθόδους για τη διασφάλιση του απορρήτου των δεδομένων.
- Υποδεικνύουμε εφαρμογές Τεχνητής Νοημοσύνης με διασφάλιση του απορρήτου στο πεδίο ανάλυσης κινηματικών δεδομένων, με έμφαση στην *Πρόβλεψη Διαδρομών Πλοίου*.
- Προτείνουμε το *FedVRF*, ένα γενικό πλαίσιο για την πρόβλεψη των μελλοντικών τοποθεσιών των πλοίων χρησιμοποιώντας Ομονσπονδιακή Μάθηση.
- Επιδεικνύουμε την αποτελεσματικότητα και την ευελιξία του *FedVRF* χρησιμοποιώντας πολλαπλά σύνολα δεδομένων πραγματικής κίνησης από τον ναυτιλιακό τομέα.
- Πειραματιζόμαστε περαιτέρω σχετικά με την αντιστάθμιση απώλειας ποιότητας/απόρρητου, σε σχέση με τα συστήματα Ομονσπονδιακής Μάθησης, καθώς και με τους μηχανισμούς διασφάλισης του απορρήτου, και δείχνουμε τα αποτελέσματά μας όσον αφορά την ακρίβεια πρόβλεψης.
- Εκμεταλλευόμαστε το *FedVRF* και αποδεικνύουμε τη χρηστικότητά του στο πλαίσιο της Πρόβλεψης Κυκλοφοριακής Ροής Πλοίων.

Η υπόλοιπη διατριβή είναι οργανωμένη ως εξής. Το κεφάλαιο 2 παρέχει μια ολοκληρωμένη βιβλιογραφική επισκόπηση σχετικά με την Κατανεμημένη και Ομοσπονδιακή μάθηση, καθώς και τεχνικές για τη διατήρηση του απορρήτου, ενώ στο Κεφάλαιο 3 ορίζουμε επίσημα το πρόβλημα της Πρόβλεψης Διαδρομών Πλοίων και παρέχουμε μια σύντομη βιβλιογραφική επισκόπηση, καθώς και τη μεθοδολογία μας σχετικά με την εκπαίδευση του μοντέλου *FedVRF* τόσο με Κεντρικοποιημένη, όσο και Ομοσπονδιακή Μάθηση. Επιπλέον, στο Κεφάλαιο 4 πειραματιζόμαστε διεξοδικά με το *FedVRF*, χρησιμοποιώντας τέσσερα σύνολα δεδομένων, που εντοπίζονται στον πραγματικό κόσμο και, συγκεκριμένα, στον ναυτιλιακό τομέα, και συζητάμε περαιτέρω τα ευρήματά μας σχετικά με τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μεθόδου εκμάθησης αναφορικά με την ακρίβεια πρόβλεψης. Τέλος, στο Κεφάλαιο 5 ολοκληρώνουμε τη διατριβή, δίνοντας παράλληλα κατευθύνσεις για μελλοντικές επεκτάσεις.

Abstract

Artificial Intelligence, and more specifically Machine Learning, is broadly adopted in many aspects of our daily lives, with data playing a crucial role in its success. While Internet of Things (IoT) devices generate massive amounts of data at high velocity, a challenge arises when privacy and computational resources are concerned. Some data may be quite sensitive (e.g., medical records) and cannot be openly available, while others are so voluminous that cannot be used in a centralized fashion to train a model. Federated Learning was proposed by Google in 2016 [44, 45] to address the aforementioned challenges. In a nutshell, it is an approach that allows a model to be trained on data we do not own and cannot see. The data generated by edge devices (e.g., smartphones) are stored locally and never shared with other nodes on the network or a central server [11, 42]. Instead, only model updates are shared and aggregated in order to construct a global model. In this thesis, we understand Federated Learning from both a theoretical and algorithmic perspective and compare its advantages and disadvantages to the centralized approach within the context of Vessel Route and Traffic Flow Forecasting.

Keywords

Machine Learning, Privacy-Preservation, Federated Learning, Mobility Data Analytics, Vessel Route Forecasting, Vessel Traffic Flow Forecasting

Acknowledgements

First and foremost, I am extremely grateful to my co-supervisors, Prof. Yannis Theodoridis, and Dr. George Siolas, for their invaluable advice, continuous support, and patience during my M.Sc. thesis. Their immense knowledge and plentiful experience have encouraged me throughout my academic research and daily life.

Also, I greatly appreciate my supervisor, Stefanos Kollias for his marvelous supervision, and guidance throughout the period of my study.

Many thanks to all of the members of staff in the Artificial Intelligence and Learning Systems (AILS), and the Data Science (DataStories) Laboratory at National Technical University of Athens (NTUA) and University of Piraeus, respectively, for their kind support during my M.Sc. study. Also, I extend my thanks to all my friends and colleagues from NTUA for their time, advice, and moral support.

Last, but not least, my warm and heartfelt thanks go to my mother, for her unconditional, unequivocal, and loving support and hope she had given to me. Without that hope, this work would not have been possible. Thank you for all of your love and for always reminding me of the end goal.

Athens, June 2022

Andreas Tritsarolis

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The vast spread of IoT-enabled devices, such as (bio-)sensors, smartwatches, smartphones, and GPS trackers, has led to the production of vast amounts of medical, mobility, and several other types of data. The availability of such data is crucial to the success of Machine Learning (ML) technologies, which can perform a variety of tasks that may sometimes exceed human performance [78]. However, the data produced by the edge devices are by nature sensitive (e.g., health records, vessel information, etc.), and, more often than not, distributed across many parties. These properties pose new challenges in terms of efficient storage, analytics, and knowledge extraction out of such data.

Centralizing the data to a certain location (e.g., data center) may become quite a cumbersome task because of the high storage/bandwidth costs (e.g., AIS-enabled vessels are expected to generate several TBs of data on a daily basis). In addition, due to regulations such as GDPR[1], the collection and sharing of high sensitive data can become quite difficult, if not outright impossible, thus forcing the data to exist in isolated data silos maintained by the corresponding owners/parties. Alternatively, delegating the training process to the edge devices and/or data silos, so that each party can use an ML-based model using their own data, may impact the models' performance, with sub-optimal performance (e.g., under-fitting) or a biased target distribution (e.g., over-fitting), depending on the datasets' size and features' distribution, respectively.

In order to solve the aforementioned challenges, and train an ML-based model that does not rely on collecting all data to a centralized storage, McMahan et al. [45] and Konečný et al. [33] propose Federated Learning (FL), a novel ML paradigm, where a centralized model is trained on decentralized data. Figure 1.1 illustrates the proposed cross-device FL architecture. In particular, each edge device receives a seminal model from the server and proceeds to train it using its corresponding data. Afterwards, all updated models are uploaded to the server, where they are aggregated, thus producing a new model. Repeating the process for several cycles may eventually cause the global model to converge, producing an ML model that performs at least better than what each party can learn on its own, ideally an approximation of the same ML model if trained in a centralized fashion.

Using FL, the decentralized nature of the data is maintained, as the edge devices

[1] Data protection; European Comission, https://ec.europa.eu/info/law/law-topic/data-protection_en, Last visited: 07/01/2022

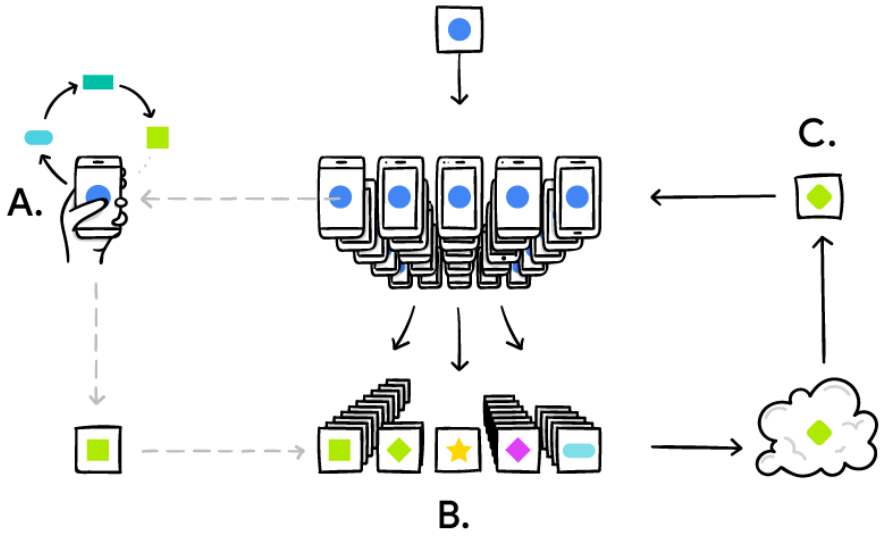

Figure 1.1. *Your phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated [source].*

collaboratively train an ML model, only sending weight updates (i.e., gradients) to the aggregation server. Because of that, every participant keeps control of its own data, as it essentially never "leaves" the device, therefore making it harder for an adversary to extract any sensitive information. Additionally, by sharing the training workload to multiple edge devices, FL allows for potentially "smarter" models, lower inference latency, less overall power consumption, and by extension, lighter environmental impact [52], all while ensuring data privacy.

In addition to the aforementioned advantages, because the aggregated global model is shared with all edge devices (i.e., smartphones), the improved model on your phone can be used immediately, therefore providing a personalized experience. A prime example of FL application in real-world scenarios is Google Gboard [20]. When Gboard shows a suggested query, the smartphone locally stores information about the current context and whether the suggestion was actually clicked or not. Afterwards, using FL, it processes that history on-device to suggest improvements to the next model iteration. Other real-world applications involve the applications of Apple macOS/iOS, where FL is used in order to protect user privacy, building models that are used to improve features to further enhance user experience.

Taking all the above into account, our aim in this thesis is to investigate privacy-aware AI applications within the context of maritime data analytics. In particular, we choose a very critical maritime analytics task, called *Vessel Route Forecasting (VRF)*, and employ privacy-preserving ML methods in order to collaboratively train an ML model across multiple data silos (e.g., universities, corporations, etc.), all while ensuring that no sensitive information will leak to other parties other than the owner of the data. VRF is critical

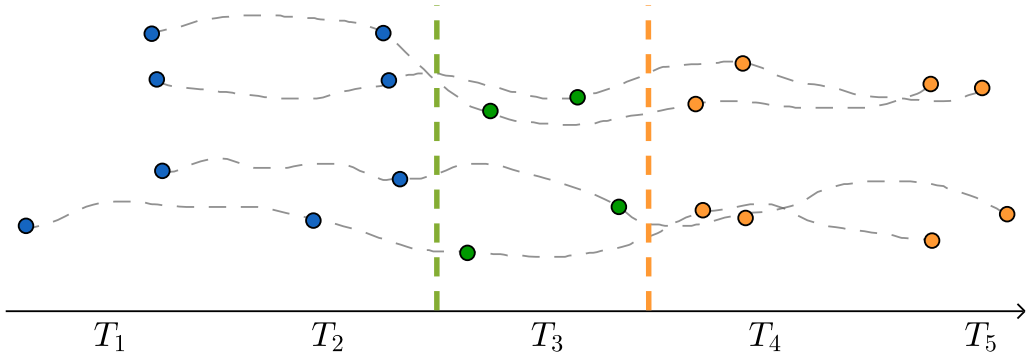

Figure 1.2. *Vessel Route Forecasting - blue, green, and orange graphs correspond to past, current, and predicted objects' locations*

because it can be used in various aspects of maritime mobility awareness, including, among others, fishing effort/pressure, future collisions, as well as co-movement patterns [68].

Informally, given a look-ahead time interval Δt, the goal is to predict the future k locations of a moving vessel after Δt time. Figure 1.2 illustrates such an example, where in blue and green, we have the information at hand (past and current locations, respectively), whereas in orange, we have the predicted routes. The problem we address is quite challenging since, apart from the inherent difficulty of predicting the future, we also need to define the FL communication protocol, as well as methods for ensuring data privacy, both of which are not straightforward procedures. To the best of our knowledge, the problem we aim to address has not been addressed in the literature yet.

1.1 Thesis Contribution

Several mobility-related applications could benefit from such an analytics task. Due to the privacy-preserving mechanisms in FL, not only edge devices (e.g., AIS-enabled vessels) but also data silo owners (e.g., universities) can collaboratively train an ML model in order to create a "smarter" predictive model, that, in addition, can be personalized to fit the needs of each participant. In a nutshell, our main contributions are the following:

- We provide an in-depth literature review regarding Distributed and Federated Learning, as well as methods for ensuring data privacy.
- We indicate privacy-aware AI applications within the mobile data analytics field, with emphasis on *Vessel Location Forecasting (VRF)*.
- We propose *FedVRF*, a framework for predicting the vessels' future locations using FL.
- We demonstrate the efficiency and versatility of *FedVRF* using several large-volume real-world data from the maritime mobility domain.

- We further experiment on the quality/privacy trade-off with respect to FL schemes, as well as privacy-preserving mechanisms, and demonstrate our results in terms of prediction accuracy.
- We exploit *FedVRF* and demonstrate its usability within the context of Vessel Traffic Flow Forecasting (VTFF).

1.2 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 provides a comprehensive overview of state-of-the-art works regarding Distributed and Federated Learning, as well as techniques for privacy preservation, while in Chapter 3, we formally define the *VRF* problem and provide a brief overview of related state-of-the-art works, as well as our methodology regarding the training of the *VRF* and *FedVRF* models in Centralized, and Federated Learning fashion. Additionally, in Chapter 4, we thoroughly experiment on *FedVRF* using four real-world datasets from the maritime domain and further discuss our findings, related to the pros and cons of each learning method, with respect to prediction accuracy. Finally, in Chapter 5, we conclude our thesis, giving hints for future work.

Chapter 2

From Distributed to Federated Learning - an Overview

Federated Learning is the intersection of multiple scientific discipline including, among others, Distributed Learning, Cryptography, and Data Ethics/Privacy. In this chapter, we discuss the related work behind the most important research topics of FL within the scope of this thesis, its advances in the form of state-of-the-art works, as well as its open problems.

2.1 Distributed Machine Learning

Distributed Machine Learning (DML), refers to multi-node ML models and systems that are designed to improve performance, preserve privacy, and scale to more training data and bigger models [16, 40, 66]. DML is divided into two main method families, namely scalability-oriented and privacy-oriented, where the produced solutions are either geared towards scalability, or privacy preservation, respectively. Recent advances on DML make ML tasks on big data feasible, scalable, flexible, and more efficient [78].

Existing works regarding scalability-oriented methods can be distinguished into three main sub-categories, depending on their training aspect, namely, Data, Model [26], and Hybrid parallelism [43], respectively.

Data-Parallel DML refers to a technique, where a dataset is split into multiple shards, which are afterwards processed through multiple replicas of the same model using different computing devices/workers, and communicating models' information in periodic intervals. Figure 2.1 illustrates an example of data-parallel DML, where three worker nodes are tasked with training a single ML model, and one master node (i.e. Parameter Server), where the workers' model parameters are aggregated in order to produce a single model. In particular, the training dataset is split into disjoint – ideally – independent and identically distributed (i.i.d) shards, which are sent to the workers in order to train an ML model using Stochastic Gradient Descent (SGD). After one batch (i.e. single SGD forward-backward pass), the models' weights w^i are sent to the master node where they are aggregated using weighted average (with respect to the number of observed training samples), thus generating a global model that is be sent to the workers for the next batch.

In data-parallel distributed learning there are two main approaches, namely, syn-

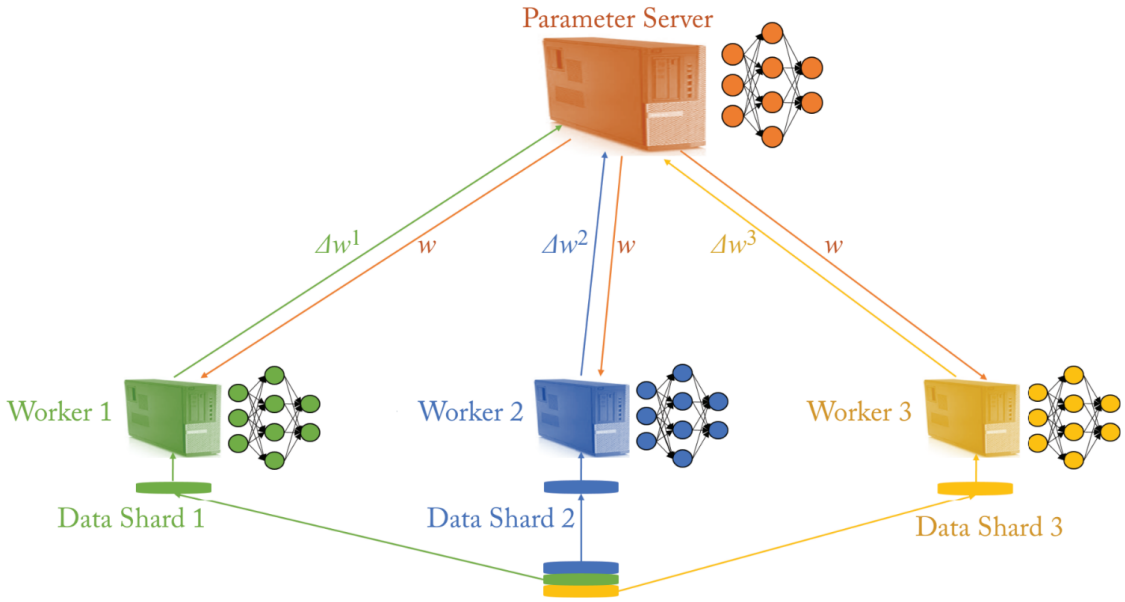

Figure 2.1. *Illustration of a Distributed Machine Learning (DML) system [78]*

chronous and asynchronous training, respectively. The former behaves as described in Figure 2.1, i.e., all workers train independently over data shards of the same dataset on replicas of the same model, and update its parameters after each training step. On the other hand, with asynchronous training, the aforementioned behaviour is maintained, with the difference that the workers do not necessarily need to constantly communicate the model updates with the parameter server after each training step.

This approach can naturally scale up well with increasing amounts of training data, which can no longer reside on a single machine. Data-parallel distributed learning is part of many ML frameworks, including the popular libraries PyTorch [37], and TensorFlow[1]. While the aforementioned training methodology is quite useful, finding the optimal synchronization strategy can become a cumbersome task. Towards this direction, Zhang et al. [83], propose AutoSync, a framework to automatically optimize synchronization strategies given model structures and resource specifications using low-shot data. In particular, by creating a search space from low-shot data collected in a few trial runs combined with a domain adaptive simulator, they discover synchronization strategies up to 1.6x better than manually optimized or fix-formed ones.

While the aforementioned approach is easily scalable with respect to processing power, as the ML models get larger and larger (e.g. BERT [9]), we may face the problem that the model cannot be loaded to a worker node, due to insufficient memory. This problem, while very rare with servers, it is not uncommon with entities such as smartphones or even a home PC.

Model parallelisation refers to a technique, where a model is split into multiple parts, and distributed to the computing devices (i.e. workers). Training takes place in serial function, where the forward/backward propagation involves communication of output

[1]Google, Distributed training with TensorFlow, https://www.tensorflow.org/guide/distributed_training. Last visited: 07-01-2020

from one device to another. While this approach reduces the memory footprint of the model, it drastically increases the communication requirements, therefore we usually resort to this approach only if the model cannot fit into a single node, not primarily to speed up the training process.

For instance, in Krizhevsky et al. [35] because the authors had two GPUs with 3GB each, they could not train a 60-million parameter Deep Convolutional Neural Network (CNN) model in data-parallel mode. Thus, they chose to split the model and distribute it among the two devices, and by cross-validation fine tune the communication scheme in order to maintain an acceptable fraction of the amount of computation.

Like data-parallel, tuning the communication scheme in model-parallel distributed learning can become a cumbersome task, especially with large ML models, with millions of parameters. Towards this direction, Jia et al. [25] propose OptCNN, a framework which employs model-parallel training on CNN models. By solving a graph search problem, based on different paralellisation schemes, they jointly optimize how each layer is distributed, increasing training throughput up to 2.2x over previous state-of-the-art parallelization strategies, achieving better scalability to multiple workers, while maintaining the performance of the original model.

Extending the previous work, Jia et al. [27] extend OptCNN, into a new framework, FlexFlow, by introducing a new search space of parallelization strategies that generalizes across different operator dimensions, namely, Sample, Operator, Attribute, and Parameter (SOAP), which describe how the training details (operators, training samples, samples' attributes, etc.) will be distributed. Employing the aforementioned architecture on six real-world benchmark datasets on two GPU clusters, they show that SOAP suggests better strategies achieving up to 3.3× over previous state-of-the-art approaches.

While data- and model- parallel are well-performing methods, they introduce a trade-off between space availability vs. communication/bandwidth capacity. Another line of research, in an effort to alleviate that trade-off, and provide further optimal training schemes, attempts to combine the merits of both data and model parallelization methods, thus producing a hybrid-parallel methodology.

For instance, Krizhevsky [34] propose a hybrid architecture, that trains a Convolutional Neural Network (CNN) using model parallelisation for its convolutional and data parallelisation for its fully-connected layers, respectively, for efficient scaling across multiple GPUs. In more advanced works, Low et al. [43] propose GraphLab, an ML framework which exploits the models' sparse structure and common computational patterns, enabling highly scalable models both in data and model dimension as well. In more advanced works, Wang et al. [73] reduce the problem of finding the optimal parallelisation strategy to finding the best tiling of partition tensors with the least overall communication cost. Within that context, they propose *SOYBEAN*, an ML framework which combines data and model parallelism, thus producing a training scheme that reduces the communication cost up to 4x, compared to pure data- or model- parallel schemes.

2.2 Privacy-Preserving Learning

Distributed Learning can not only be used to scale up the training process of an ML model, but also integrate data from multiple sources. In many real-world scenarios, the data are distributed among many parties, for instance, corporations, hospitals, universities, etc. Because in some cases, the data may contain highly sensitive information, e.g., medical data, explicitly sharing them to outside sources is forbidden not only for ethical, but also for legal reasons due to regulations such as GDPR. In addition, sharing a (pre-)trained model on the aforementioned data, is not suggested, as recent works [22] have shown that parts of the training dataset can be recovered both from the weights of the model, as well as its gradients.

Therefore, in order to be able to either share data, or most commonly, ML models for distributed learning, it is necessary to ensure some privacy guarantees so as to protect our model from adversaries/outsiders. In general, a privacy-preserving distributed learning system must protect at least one of the following type of information [71], namely, input data, output predictions, model information (e.g. parameters), and identifiable information (e.g. which AIS location is emitted from which vessel). The popular tools for ensuring privacy can be sorted into two major categories, more specifically, obfuscation and encryption methods, respectively.

Obfuscation methods' primary aim is to modify the ML models' parameters in order to attain a certain level of privacy. Differential Privacy [13]) is a prime example of such method. It is a technique that makes possible for outside parties to collect and share aggregate sensitive information, while ensuring the privacy of individual users. Given the users' database X, a parallel database Y, and a randomized algorithm M with domain $\mathbb{N}^{|X|}$, we can ensure that M is (ϵ, δ)-differentially private if for all $S \subset Range(M)$ and for all $X, Y \in \mathbb{N}^{|X|}$ such that $\|X - Y\|_1 \leq 1$ we have that

$$Pr[M(X) \in S] \leq exp(\epsilon) * Pr[M(Y) \in S] + \delta \tag{2.1}$$

The aforementioned definition does not create differential privacy per se, however it is a measure of how much privacy is afforded by an aggregation query M. More specifically, it's a comparison between running M on a database X and a parallel database Y. A parallel database is defined as a copy of the database X, albeit without the records of a certain user. In a nutshell, it expresses that for each parallel database, the maximum distance between an aggregation query on X and Y, will be at most e^{ϵ}, with probability $1 - delta$. ϵ is a metric of privacy loss, and the smaller its value, the better privacy it ensures[2]. While setting ϵ and δ to zero may be tempting, absolute privacy comes at a cost of added noise, which impacts the convergence and accuracy of an ML model. Thus we need to compromise between privacy and model performance. For instance, Apple uses a privacy budget with ϵ up to 8 for general applications, and $\epsilon = 2$ for the Health application, respectively[3].

[2]For algorithms other than the Laplace mechanism, e.g. Gaussian, another similar metric is δ [13].

[3]Apple, Differential Privacy, https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf, Last visited: 27/01/2022

Differential Privacy has been used extensively in privacy-motivated ML models. From the survey of Zhang et al. [82] regarding collaborative Deep Learning and privacy-preserving mechanisms, we present some representative works. Song et al. [61] propose a variant of the popular Stochastic Gradient Descend (SGD), which ensures user privacy via Differential Privacy, called DP-SGD, which clips the gradients to a specific range, and injects noise to them, either via Gaussian or Laplacian distribution during training, so that the trained model ensures (ϵ, δ)-differential privacy. In a different direction from [61], Shokri et al. [60] propose another variant of SGD, called Distributed Selective Stochastic Gradient Descent (DSSGD), which allows the local model to selectively share a fraction of the parameters, thus avoiding information leakage, while maintaining a comparable performance with respect to centralized learning.

In a similar fashion, Sabater et al. [59] propose GOPA (GOssip noise for Private Averaging), a novel differentially private averaging protocol for ensuring user privacy at fully decentralized (e.g. peer-to-peer) learning environments. It relies on users exchanging (directly or through a server) correlated Gaussian noise in order to mask their private values without affecting the global average. This ultimately canceling noise is complemented by the addition of independent (non-canceling) Gaussian noise by each user. After extensive evaluation, they demonstrate its robustness to client (i.e. edge devices) dropouts, as well as its performance, which is on par with centralized ML training, all while maintaining a low communication overhead.

On the other hand, encryption methods' primary aim is to encode the ML models' parameters in a manner that it does not alter the distributed computation process, while in parallel not revealing sensitive information to other parties/adversarials. These methods include, but are not limited to, Secure Multi-Party Computation (SMPC) [79], and Homomorphic Encryption (HE) [56]. Within the scope of this thesis, we explore the related works on HE methods.

In brief, the edge devices in SMPC environment jointly compute a function from the private input by each party, without revealing such inputs to the other parties. SMPC allows us to compute functions of private input values so that each party learns only the corresponding function output value, but not input values from other parties.

Homomorphic Encryption on the other hand, encrypts the values in such way that certain algebraic computations can be performed over the cyphertext, without the need of decrypting it. Currently, the state-of-the-art in HE is from Gentry [17], where he proposed a HE scheme that is able to perform not only addition (as in the original thesis), but also multiplicative operations for unlimited number of times, with no computation loss.

Within PPML context, HE can be used in order to train an ML model so that the aggregation server cannot access its parameters, therefore dropping the need for an honest-but-curious server. The CryptoDL [21] framework is an HE-based approach for secure neural network training and inference. In the aforementioned framework, several activation functions are approximated using low-degree polynomials and mean pooling is used as a replacement for max pooling for the case of Convolutional Neural Networks.

Similarly, Juvekar et al [28] propose Gazelle, a scalable and low-latency system for secure neural network inference, using an intricate combination of HE and traditional

two-party computation techniques. By employing a simpler (compared to Linear HE in CryptoNets [18]) packed additively HE scheme, Gazelle supports very fast matrix-vector multiplications and convolutions over encrypted data.

2.3 Federated Learning

While Distributed Machine Learning (DML) can help us scale up the training process across multiple computational nodes, it can only be used on centralized data. Federated learning (FL) is a branch of DML which trains centralized models using decentralized data [44]. In comparison to DML, FL algorithms are fundamentally different and primarily geared towards data privacy. Table 2.1 illustrates the key differences between distributed and federated learning in two popular variants depending on the clients' type, namely, cross-silo and cross-device.

In particular, the most important differences between distributed and federated learning lies within the scale and distribution of the data, the reliability of the computing nodes, and the optimization schemes. In distributed learning, the data are centrally stored and i.i.d balanced across clients, while in federated learning the data are usually non-i.i.d, with them being either horizontally, or vertically partitioned, as they are generated locally and remain decentralized among the edge devices. Additionally, in FL the distribution scale is massive, spanning up to 10^{10} edge devices collaboratively training an ML model, while in distributed learning it is typically restricted to the available computing nodes within the cluster/data-center (up to 1000 clients). Another difference lies within the nodes' reliability, where in distributed learning are – virtually – always available, whereas in FL, and especially in the cross-device FL, only a fraction of clients are available at a given time instance, mainly due to variations in factors such as communication, bandwidth, resource allocation, etc.

Taking into account the characteristics of the data owners, we distinct two major FL variants, namely, cross-silo and cross-device FL [29]. Cross-device FL can be considered when the participating devices (clients) are typically large in number (up to 10^{10}) and have slow or unstable internet connections (c.f. Table 2.1). A principal motivating example for Federated Learning arises when the training data comes from users' interaction with mobile applications [33]. On the other hand, cross-silo FL can be considered when a relatively small group (usually $2 - 100$) of companies or organizations share a common incentive to collaboratively train an ML model based on their data, but cannot share them directly, due to either cost (e.g. centralize data to a certain location) or legal constraints (c.f. GDPR). Another key difference between cross-device and cross-silo FL, lies within the privacy requirements of the FL framework. In cross-device FL, data privacy is of the utmost importance, as the trained ML model will be available to virtually everyone, whereas in cross-silo FL, the trained ML model most likely be available for internal use among the participating parties, therefore the concerns about "virtually everyone" are less important in the life-cycle of the ML model.

With respect to the nodes' communication scheme, Rieke et al. [55] distinct three major categories, namely, client-server, peer-to-peer, and hybrid, as illustrated by Figure

	Data-center Distributed Learning	**Cross-silo Federated Learning**	**Cross-device Federated Learning**
Setting	Training a model on a large but "flat" dataset. Clients are compute nodes in a single cluster or datacenter.	Training a model on siloed data. Clients are different organizations (e.g. medical or financial) or geo-distributed datacenters.	The clients are a very large number of mobile or IoT devices.
Data distribution	Data is centrally stored and can be shuffled and balanced across clients. Any client can read any part of the dataset.	**Data is generated locally and remains decentralized.** Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	**A central orchestration server/service organizes the training**, but never sees raw data.	
Wide-area communication	None (fully connected clients in one datacenter/cluster).	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	
Data availability	All clients are almost always available.		Only a fraction of clients are available at any one time, often with diurnal or other variations.
Distribution scale	Typically 1 – 1000 clients.	Typically 2 – 100 clients.	Massively parallel, up to 10^{10} clients.
Primary bottleneck	Computation is more often the bottleneck in the datacenter, where very fast networks can be assumed.	Might be computation or communication.	Communication is often the primary bottleneck, though it depends on the task. Generally, cross-device federated computations use wi-fi or slower connections.
Addressability	Each client has an identity or name that allows the system to access it specifically.		Clients cannot be indexed directly (i.e., no use of client identifiers).
Client statefulness	Stateful; each client may participate in each round of the computation, carrying state from round to round.		Stateless; each client will likely participate only once in a task, so generally a fresh sample of never-before-seen clients in each round of computation is assumed.
Client reliability	Relatively few failures.		Highly unreliable; 5% or more of the clients participating in a round of computation are expected to fail or drop out (e.g., because the device becomes ineligible when battery, network, or idleness requirements are violated).
Data partition axis	Data can be (re-)partitioned arbitrarily across clients.	Partition is fixed. Could be example (horizontal) or feature (vertical) partitioned.	Fixed partitioning by example (horizontal).

Table 2.1. *Typical characteristics of federated learning settings vs. distributed learning in the datacenter (e.g. [7]). Cross-device and cross-silo federated learning are two examples of FL domains, but are not intended to be exhaustive. The primary defining characteristics of FL are highlighted in bold, but the other characteristics are also critical in determining which techniques are applicable. [29]*

2.2. Client-server architecture (Figure 2.2a), is quite similar to the architecture described in Chapter 2.1 (i.e., Data-center Distributed Learning), where multiple parties collaboratively train an ML model and via a central aggregation server, combine the acquired knowledge into a global model. On the other hand, peer-to-peer (Figure 2.2b) is quite different, as there is no need for a central server to aggregate the models' parameters. Each entity trains the same ML model using its local data, and communicates the updated pa-

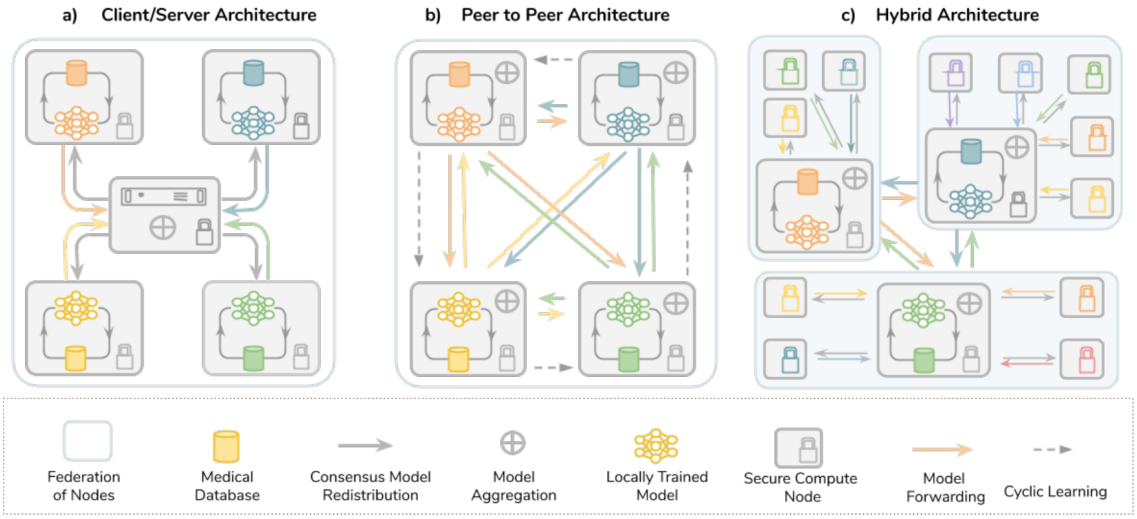

Figure 2.2. *Federated Learning Communication Architectures [55]*

rameters with the other workers using secure communication channels (e.g asymmetric cryptography).

Sending parameters from one entity to another can be conducted in two main ways, either via Cyclic, or Random transfer. In the cyclic transfer mode, the training collaborators are organized in a ring topology, where a worker receives the model parameters from its upstream worker, performs a training step using mini-batches from its local dataset, and sends the updated parameters to its downstream worker. This procedure is repeated until wither the ML model converges, or until the maximum allowed training time/steps is reached. Compared with the client-server, the advantage of peer-to-peer architecture is that it enables fully decentralized training, eliminating the need for an aggregation server, and the chance of leaking - potentially - sensitive information to it. However, there are several disadvantages, especially in the communication costs. For instance, in the cyclic transfer mode, since there is no central server, weight parameters are updated serially rather than in parallel batches, which takes more time to train a model [78].

Finally, in hybrid architecture (Figure 2.2c), a combination of the previously mentioned architectures is used, in order to create a global ML model from some training parties in peer-to-peer mode, which in their turn train a local ML model using multiple edge devices (i.e. computing entities) in client-server mode. An example of this method is the collaboration of several maritime organizations for training a global ML model for prediction the future locations of their fleet in decentralized fashion, by using ML models that are trained by the edge devices (i.e. AIS-enabled vessels) that belong to the organizations' corresponding fleet.

As far as the data distribution within the workers' data is concerned, we distinct two main variants, Horizontal and Vertical Federated Learning, illustrated at Figures 2.3, and 2.4, respectively. Horizontal Federated Learning (HFL) [29], can be used in cases where the participating parties' datasets overlap in feature but differ in sample space. For instance, two maritime organizations may have different vessel fleets, depending on their corresponding regions, albeit with very small overlap. However, their business models are

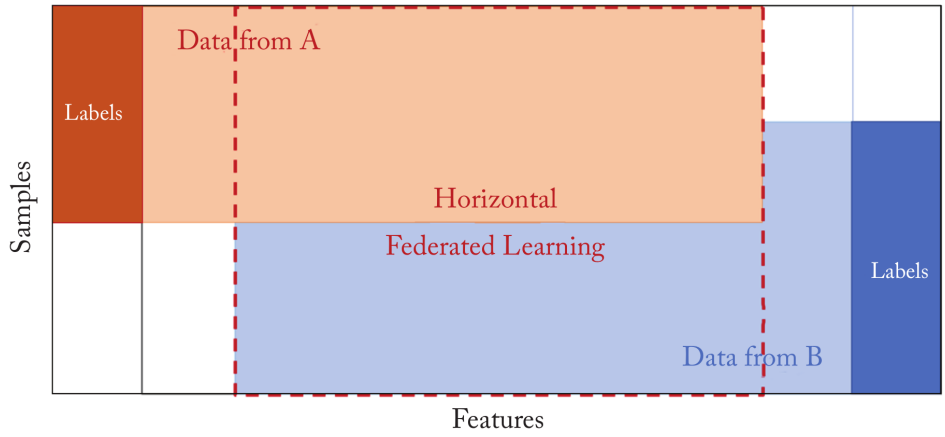

Figure 2.3. *Illustration of HFL, a.k.a. sample-partitioned federated learning [77].*

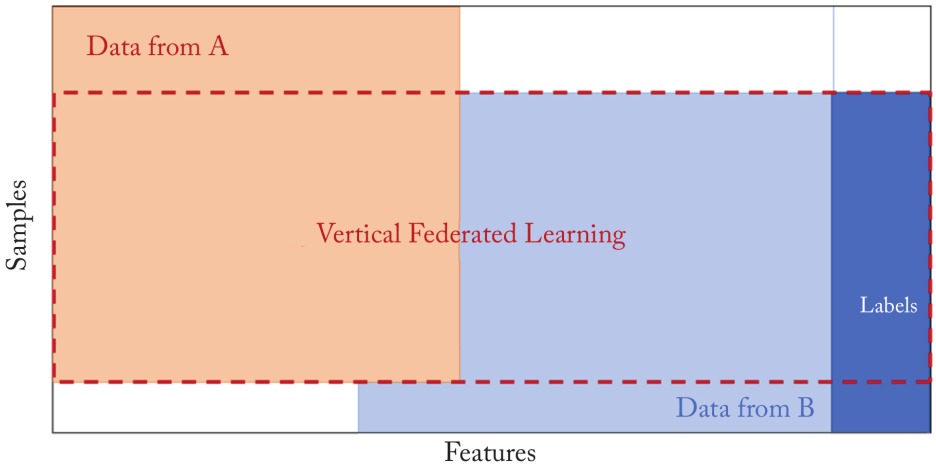

Figure 2.4. *Illustration of VFL, a.k.a. feature-partitioned federated learning [77].*

very similar. Hence, the datasets' feature spaces are similar. Formally, HFL is defined as follows:

$$X_i = X_j, Y_i = Y_j, I_i \neq I_j, \forall D_i, D_j, i \neq j \tag{2.2}$$

where (D_i, I_i), (D_j, I_j) are the datasets and user spaces of the i^{th}, and j_{th} party, respectively, with (X_i, Y_i) and (X_j, Y_j) being the corresponding feature and label spaces of the aforementioned parties. On the other hand, Vertical Federated Learning (VFL) can be used in cases where the participating parties' datasets differ in feature but overlap in sample space. For instance, maritime organizations may partner up with shipping companies in order to provide faster and safer routes for its fleet. Formally, VFL is defined as follows:

$$X_i \neq X_j, Y_i \neq Y_j, I_i = I_j, \forall D_i, D_j, i \neq j \tag{2.3}$$

where (D_i, I_i), (D_j, I_j) are the datasets and user spaces of the i^{th}, and j_{th} party, respectively, with (X_i, Y_i) and (X_j, Y_j) being the corresponding feature and label spaces of the aforementioned parties. In the cross-device setting the data is assumed to be par-

titioned by examples. In the cross-silo setting, in addition to partitioning by examples, partitioning by features is of practical relevance [29]. Finally, in cases where there are neither enough shared features nor samples among the training parties Federated Transfer Learning (FTL) [41] can be used in order to collaboratively train an ML model that transfers knowledge acquired among the parties to achieve better performance.

On top of the aforementioned, Optimization is a yet another key difference between distributed and federated learning. In distributed learning, the global model is created by averaging the gradients of all local models, and performing SGD on the global model, an approach called *Gradient Averaging* [45]. On the other hand, in Federated Learning, mainly due to privacy issues relating to data leakage from gradients [22], another approach is preferred called *Model Averaging*, where the models' weights are averaged, instead of their gradients. In McMahan et al. [45] both methods are referred to as Federated Averaging (FedAvg).

Method	Advantage	Disadvantage
Gradient Averaging	Accurate gradient information Guaranteed convergence	Heavy communication Requires reliable connection
Model Averaging	Not bound to SGD Tolerance of update loss Infrequent synchronization	No guarantee of convergence Performance loss

Table 2.2. *Comparison between gradient averaging and model averaging [78]*

Table 2.2 summarizes the key differences between the two variants of FedAvg. In gradient averaging, the aggregation step is performed once per batch [34], the global model is guaranteed to converge, with rate correlated to the batches' size. While this approach produces reliable results, it requires a high communication bandwidth, as well as a near-constant connection with the clients, two hypotheses that may work with distributed learning, but cannot within FL. On the other hand Model Averaging, in order to address the availability/reliability issues within FL training environments, aggregates the models' *weights* once per training step. This approach, while it is more tolerant than Gradient Averaging, its infrequent synchronization, comes at a (great) performance loss, with no guarantee of convergence.

Following the same line of research, Reddi et al. [54] generalize the FedAvg algorithm in order to allow usage of adaptive optimization schemes, such as Adam [31], YOGI [81], and AdaGrad [12]. In particular, in the approach they propose the clients use SGD for local model training, while the aggregation server uses any of the aforementioned optimization schemes in order to update the parameters of the global model. In addition, by focusing on adaptive server optimization, they enable use of adaptive learning rates without increase in client storage or communication costs, while ensuring compatibility with cross-device FL [54].

Except network and communication efficiency, and client availability [33], another key challenge of federated optimization is the training parties' heterogeneity with respect to their local datasets [32]. In order to address the first two issues, FedAvg performs multiple local updates on the available clients before communicating to the server. While

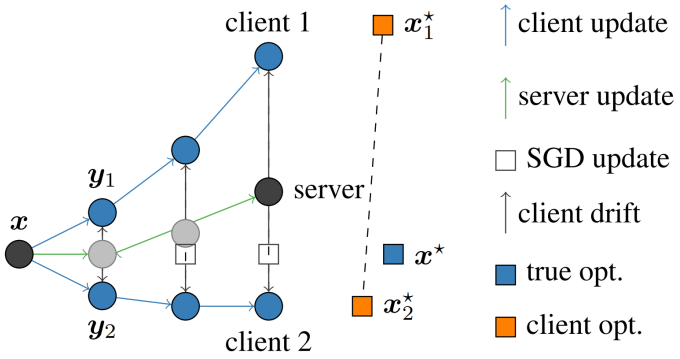

Figure 2.5. *Client-drift in FedAvg is illustrated for 2 clients with 3 local steps* ($N = 2, K = 3$). *The local updates* y_i *(in blue) move towards the individual client optima* x_i^* *(orange square). The server updates (in red) move towards* $\frac{1}{N}\sum_i x_i^*$ *instead of to the true optimum* x^* *(black square). [30]*

this approach works well with high convergence guarantees (in applications where the participating parties' datasets are homogeneous), when the clients are heterogeneous these guarantees fail to hold. By each step, the parties' locally fitted ML model will converge to different local optima, therefore introducing slow and unstable convergence to the global model, as Figure 2.5 illustrates. This phenomenon is better known as "client-drift", and in order to avoid its fewer local updates and/or smaller learning rates must be used, action which largely impact the convergence stability of FedAvg.

Towards this direction, Karimireddy et al. [30] acknowledge the aforementioned issue and propose a new federated optimization framework called SCAFFOLD, which uses control variates (variance reduction) in order to approximate an ideal unbiased update, therefore taking into account the "client-drift" in its local updates. By experimenting on various optimization settings, the authors prove that SCAFFOLD is resilient to client sampling (i.e. independent of the amount of client heterogeneity), and consistently outperforms FedAvg on non-convex experiments. Further following this line of research, another relevant approach is the q-FedAvg algorithm proposed by Li et al. [38], a novel optimization objective "inspired by fair resource allocation in wireless networks that encourages a more uniform accuracy distribution across devices in federated networks". While effective in cross-silo FL, the aforementioned methods are incompatible with cross-device FL as it requires clients to maintain state across rounds, a problem which the adaptive federated optimization algorithms at [54] aim to address.

Chapter 3

Maritime Analytics and the VRF Problem

3.1 Definitions and VRF Problem Formulation

Before we proceed to the actual formulation of the problem, let us provide some preliminary definitions.

Definition 3.1. *(Trajectory). A trajectory $T = \{p_1, \dots p_n\}$ of a moving object is considered as a sequence of timestamped locations, where n corresponds to the latest reported position $p_i \in T$, where $p_i = \{x_i, y_i, t_i\}$, with $1 \leq i \leq n$.*

Definition 3.2. *(Future Location Prediction). Given a trajectory T_i and a time interval Δt, the goal is to predict $p_i^{pred} = \{x_i^{pred}, y_i^{pred}, t_i^{pred}\}$ at timestamp $t_i^{pred} = t_i^{curr} + \Delta t$, where t_i^{curr} and t_i^{pred} correspond to current and predicted timestamps, respectively.*

Definition 3.3. *(Vessel Route Forecasting). Given a dataset D, a trajectory T_{ij} of the vessel v_j, a prediction horizon Δt and a number of transitions k, the goal is to train a data-driven model, which will predict the vessels' future k locations up to Δt with step s.*

If we recall Figure 1.2, its provides an illustration of Definition 3.3. More specifically, we know the movement of four objects from timestamps T_1 up to T_3. Our goal, given $t = 2$, and $s = 1$, is to predict the anticipated motion of these vessels until T_5, in cross-silo FL.

3.2 Related Work on VRF Methods

Despite the significant improvement of ML tools over the past decades, adapting this technology into the maritime industry is not a straightforward task. Distributed/Federated ML for solving extreme-scale streaming problems is a concept that is still in its research phases. Indicatively, overviews of the current state-of-the-art techniques in the field of Distributed ML are available in [72] and [2], while in Mohammadi et al. [46] present a survey on Deep Learning (DL) methods over Big Data and streaming data within the Internet of Things domain.

Considering the VRF problem, current status of state-of-the-art includes an adequate number of research works. More specifically, one line of work includes clustering-based prediction techniques. Such an approach was presented by Trasarti et al. [65] called MyWay. MyWay is a hybrid, pattern-based approach that utilizes individual patterns when

available, and when not, collective ones, in order to provide more accurate predictions and increase the predictive ability of the system. In a similar line of research, Petrou et al. [49, 50] utilize the work done by [63] on distributed subtrajectory clustering, in order to extract individual subtrajectory patterns from big mobility data. These patterns are subsequently utilized in order to predict the future location of the moving objects in parallel.

Specifically to the maritime domain, there are also works that leverage Neural Network (NN), and particularly Recurrent Neural Network (RNN)-based models [58]. RNNs are a popular method for trajectory prediction due to their powerful ability to fit complex functions, along with their ability of adjusting the dynamic behaviour as well as capturing the causality relationships across sequences.

Wang et al. [74] aiming at predicting the movement trend of vessels in the crowded port water of Tianjin port, proposed a vessel berthing trajectory prediction model based on bidirectional GRU (Bi-GRU) and cubic spline interpolation. Capobianco et.al. [5] provided a genuine DL approach for vessel trajectory prediction. Using a temporal window within an area of interest, and an encoder-decoder LSTM using the attention mechanism, they predict the vessels' future locations. Suo et al. [62] present an RNN-based model to predict vessel trajectories based on the DBSCAN [14] algorithm to derive main trajectories, and a symmetric segmented-path distance approach to eliminate the influence of a large number of redundant data and optimize incoming trajectories. Liu et al. [39] propose "Spatio-Temporal GRU", a trajectory classifier for modeling spatio-temporal correlations and irregular temporal intervals prevalently presented in spatio-temporal trajectories. More specifically, a segmented convolutional weight mechanism was proposed to capture short-term local spatial correlations in trajectories along with an additional temporal gate to control the information flow related to the temporal interval information.

3.3 Centralized vs. Federated Learning

In this chapter we present the proposed solution to the problem of *Vessel Route Forecasting* (VRF), and elaborate further towards the methodology used in the aforementioned FL scenarios.

In a nutshell, trajectories are a sequence of locations organized by time, therefore can be considered as time-series data [76] and thus techniques capable of handling sequential data and/or time series [57] can be applied. Over the past decades, the research interest on time-series forecasting has shifted to RNN-based models, with Gated Recurrent Units (GRU) being the newer generation of RNN, which has emerged as an effective technique for several learning problems, including sequential/temporal data applications [10].

While Long Short-Term Memory (LSTM) [23] is a quite popular RNN-based architecture, in our case however, GRU presents some interesting advantages over the LSTM. In particular, GRU networks are less complicated, easier to modify and - compared to LSTM - faster to train. Addidionally, GRU networks achieve better accuracy performance on route forecasting problems on various domains, such as maritime [62], aviation [19], and urban transportation [3]. Hence, in this thesis we follow the same direction and employ

a GRU-based model.

In brief, a GRU cell includes includes two gates, a reset gate which is used to decide "how much" past information to forget and an update gate which decides "how much" current information to add. Equations 3.1-3.4 briefly state the update rules for the employed GRU layer [6]. Additionally, details for the Back-Propagation Through Time (BPTT) algorithm, can be found in [75].

$$\mathbf{z}_k = \sigma(\mathbf{W}_{\tilde{\mathbf{p}}z} \cdot \tilde{\mathbf{p}}_k + \mathbf{W}_{hz} \cdot \mathbf{h}_{k-1} + \mathbf{b}_z) \tag{3.1}$$

$$\mathbf{r}_k = \sigma(\mathbf{W}_{\tilde{\mathbf{p}}r} \cdot \tilde{\mathbf{p}}_k + \mathbf{W}_{hr} \cdot \mathbf{h}_{k-1} + \mathbf{b}_r) \tag{3.2}$$

$$\tilde{\mathbf{h}}_k = \tanh(\mathbf{W}_{\tilde{\mathbf{p}}h} \cdot \tilde{\mathbf{p}}_k + \mathbf{W}_{hh} \cdot (\mathbf{r}_k * \mathbf{h}_{k-1}) + \mathbf{b}_h) \tag{3.3}$$

$$\mathbf{h}_k = \mathbf{z}_k \odot \mathbf{h}_{k-1} + (1 - \mathbf{z}_k) \odot \tilde{\mathbf{h}}_k \tag{3.4}$$

where $\mathbf{z}$ and $\mathbf{r}$ represent the update and reset gates, and $\tilde{\mathbf{h}}$ and $\mathbf{h}$ represent the intermediate memory and output, respectively. Additionally, $\tilde{\mathbf{p}}$ represents the input data, with $\mathbf{W}_*$, and $\mathbf{b}_*$ representing the weights and bias matrices of the GRU cell, respectively.

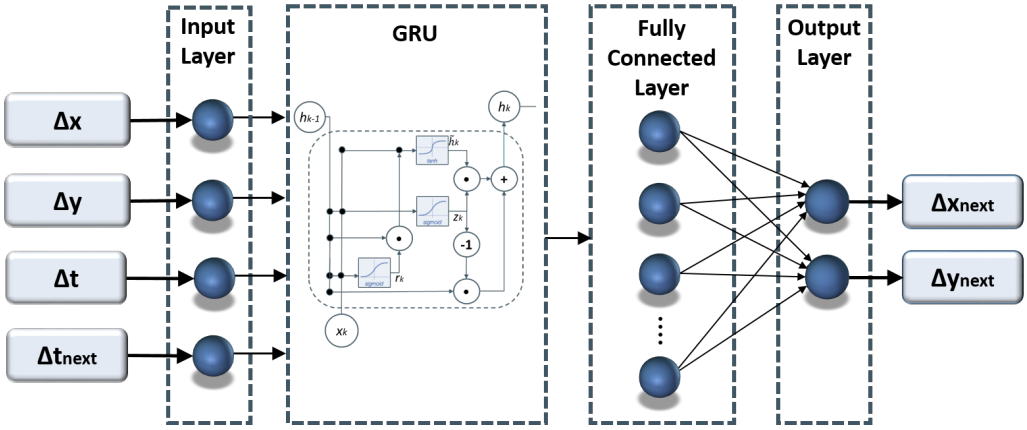

Figure 3.1. *GRU-based neural network architecture [68]*

To address the problem of Vessel Route Forecasting (VRF) in this thesis, we use the GRU-based model employed in [68] for predicting the future location of co-movement patterns in the maritime domain. Figure 3.1 illustrates the architecture of the employed ML model. More specifically, it consists of the following layers: a) an input layer of four neurons, one for each input variable, b) a single GRU hidden layer composed of 150 neurons, c) a fully-connected hidden layer composed of 50 neurons, and d) an output layer of two neurons, one for each prediction coordinate (longitude and latitude). The input variables consist of the differences in space (longitude and latitude), time, as well as the time horizon for which we want to predict the vessel's position. The differences are computed between consecutive points of each vessel. For the centralized training approach, we properly process and unify all available datasets into a single entity, which is afterwards used for training the ML model, using the Adam [31] optimization algorithm. On the other hand, in the cross-silo Distributed Learning environment, we create as many entities as available datasets, which are then used in order to train the ML model in

parallel using the FedAvg algorithm [45].

In contrast to the aforementioned ML paradigms, Cross-device Federated Learning (FL) instead of bringing the data to the model, i.e., centralizing the data to a single entity or using large data silos from various data owners, it brings the model to the data. More specifically, cross-device FL depends on various edge devices which are only aware of their own data, which are used in order to partially train an instance of the latest iteration of an ML model. Afterwards, this model is uploaded to a central (aggregation) server along with the rest - partially - updated models that are aggregated into the next iteration of the ML model, repeating the training cycle.

Within our FL environment, each "edge device" corresponds to the transmitted locations of a certain AIS vessel. Each "device" contains an instance of our proposed VRF model which is trained using only their corresponding data. For the aggregation of all VRF models, we use a variant of the FedAdam [54] algorithm, illustrated in Algorithm 3.1. In more detail, instead of SGD we use the Adam [31] optimizer for both on-device (local), and on-server (aggregation) training, in order to provide a more stable and robust convergence rate (compared to FedAvg). In Algorithm 3.1, the parameter τ corresponds to the *degree of adaptability* of the algorithm, with smaller values of τ representing higher degrees of adaptivity.

Algorithm 3.1: FedAdam

Initialization: $x_0, v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
for $t = 0, \ldots, T-1$ **do**
  Sample subset S of clients
  $x_{i,0}^t \leftarrow x_t$
  for *each client* $i \in S$ ***in parallel*** **do**
    for $k = 0, \ldots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t \leftarrow x_{i,k}^t - \eta_l g_{i,k}^t$
    end
    $\Delta_i^t \leftarrow x_{i,K}^t - x_t$
  end
  $\Delta_t \leftarrow \beta_1 \Delta_{t-1} + (1-\beta_1)\left(\frac{1}{|S|}\sum_{i\in S}\Delta_i^t\right)$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$
  $x_{t+1} \leftarrow x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$
end

Chapter 4

Use case: Application of our Approach over Real-world AIS datasets

In this chapter, we evaluate our VRF model on several centralized and federated learning schemes using three real-world maritime mobility datasets and present our experimental results.

4.1 Datasets and Preprocessing

In order to simulate all cross-silo federated learning scenarios, we use four large-scale real-world datasets from the maritime mobility domain. In particular, the first data source that we use is the "Piraeus AIS dataset for large-scale maritime data analytics" [70], referred to as the "Piraeus"[1] dataset. It consists of over 200 million AIS positioning messages from approximately 8,000 vessels (passenger boats, fisheries, cargo, containers, etc) within Saronic Gulf, Greece. The dataset ranges in time and space, as follows:

- Temporal range: May 9th, 2017 to December 26th, 2019 (≈2.5 years)
- Spatial range: longitude in [22.992, 24.031]; latitude in [37.437, 38.046]

The second data source that we use is the "Heterogeneous Integrated Dataset for Maritime Intelligence, Surveillance, and Reconnaissance" [53], referred to as the "Brest"[2] dataset. It consists of over 19 million AIS positioning messages from 5,055 vessels (passenger boats, fisheries, cargo, containers, etc) within Brest Bay, France. The dataset ranges in time and space, as follows:

- Temporal range: October 1st, 2015 to March 31st, 2016 (6 months)
- Spatial range: longitude in [-10.0, 0.0]; latitude in [45.0, 51.0]

In addition, the third data source that we use is the "Historical AIS data in Norwegian waters" [8], referred to as the "Norway"[3] dataset. It consists of over 362 million AIS positioning messages from 2,862 vessels (passenger boats, fisheries, cargo, containers,

[1]The dataset is publicly available at https://doi.org/10.5281/zenodo.5562629
[2]The dataset is publicly available at https://doi.org/10.5281/zenodo.1167594
[3]The dataset is publicly available at https://ais-public.kystverket.no/

etc) within Haugesund and Oslo, Norway. The dataset ranges in time and space, as follows:

- Temporal range: January 1st, 2019 to December 31st, 2019 (1 year)
- Spatial range: longitude in [4.09, 31.76]; latitude in [57.76, 71.38]

Finally, a fourth data source that we use is the so called "MarineTraffic"[4] dataset. It consists of over 5.5 million AIS positioning messages from 460 *passenger* vessels within Greece. The dataset ranges in time and space, as follows:

- Temporal range: January 1st, 2019 to December 31st, 2019 (1 year)
- Spatial range: longitude in [22.63, 28.03]; latitude in [34.96, 41.00]

Out of the aforementioned datasets (namely, Piraeus, Brest, Norway, and MarineTraffic) we selected a period of three months (January 1st – March 31st), which consists of 2,788,137, 3,744,412, 8,013,242, and 646,338 locations, from 1373, 1016, 335, and 177 vessels, respectively.

During the preprocessing stage, we drop erroneous records (i.e. GPS locations) based on a speed threshold $speed_{max}$ as well as stationary points with speed below $speed_{min}$. Afterwards, we organize the cleansed data into trajectories based on the temporal interval between two consecutive signals of the same vessel, given a threshold Δt, and drop those which consist of less than 10 points. In our experiments we set $speed_{max}$ = 50 knots, $speed_{min}$ = 1 knot, and Δt = 30 minutes, respectively. The rationale behind these thresholds stems from the characteristics of the dataset, which were unveiled after a statistical analysis of the distribution of *speed* and Δt between consecutive points of the same trajectory.

A map visualization of all four datasets used in our experimental study is illustrated in Figure 4.1. In particular, Figure 4.1a visualizes the transmitted AIS locations in Saronic Gulf, Greece on July 10th, 2018, while Figures 4.1b and 4.1c visualizes the transmitted AIS locations in Brest Bay, France, and Oslo, Norway on March 1st, 2016, and 2019, respectively. Last but not least, Figure 4.1d illustrates the transmitted AIS locations within (mostly) Aegean Sea on March 1st, 2019.

4.2 Experimental Setup

All conducted experiments on were implemented in Python3 (via Anaconda3[5] virtual environments). More specifically, the aforementioned models were implemented using PyTorch [48] and trained using Flower [4] for Federated Learning, via a single node equipped with 8 CPUs, 24 GB of RAM, and an Nvidia GTX 1050Ti with 4GB VRAM. Additionally, for Differential Private (DP) -enabled FL, we use a computational cluster that consists of the aforementioned node, and a MacBook Pro with 8 ARM CPUs and 16 GB of RAM.

[4]The dataset is kindly provided by MarineTraffic

[5]https://www.anaconda.com/

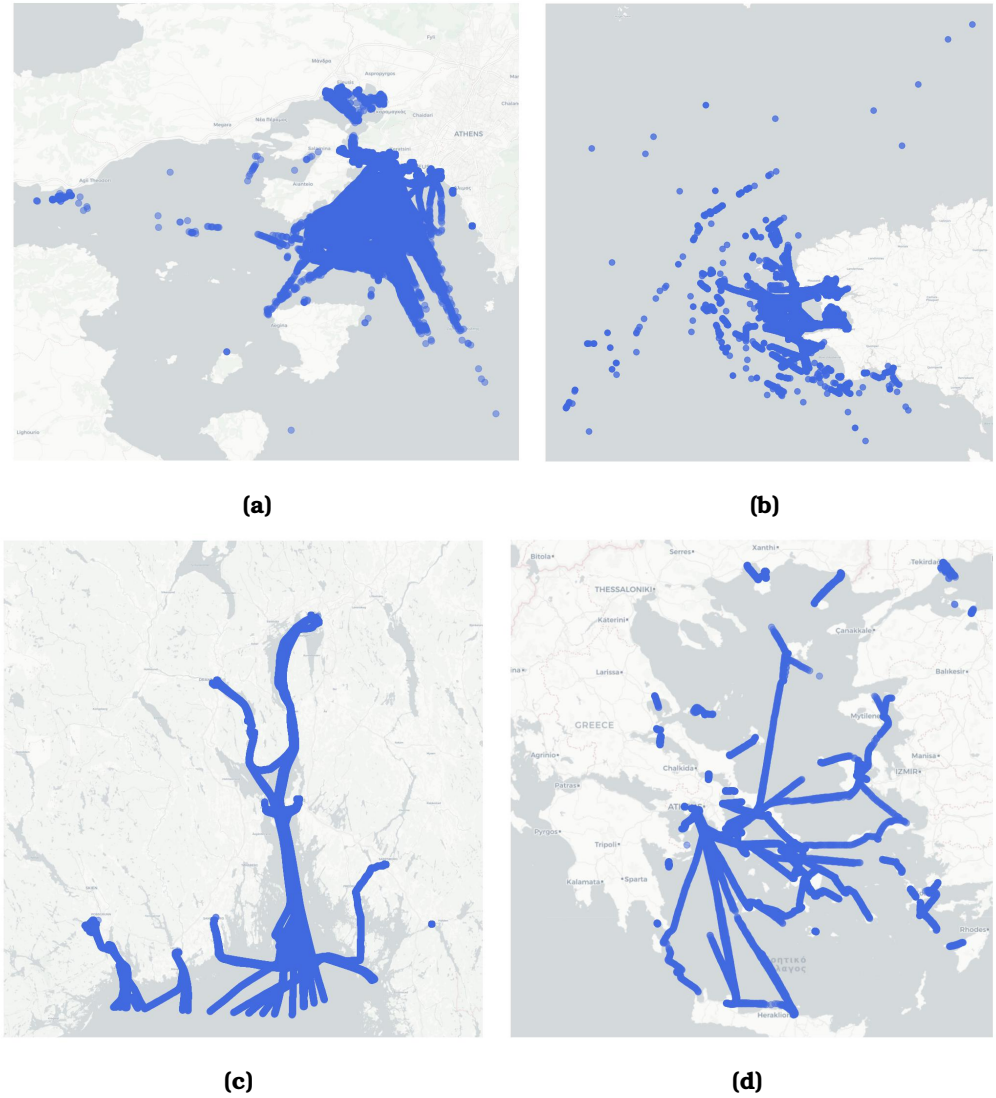

Figure 4.1. *Snapshots of (a) Piraeus; (b) Brest; (c) Oslo; and (d) MarineTraffic datasets.*

In the chapters that follow, we provide the experimental results of our study, comparing the VRF model presented in Chapter 3 trained in Centralized fashion against its FL variant, hereafter named "FedVRF", and assess its exploitation value towards short-term VRF and Vessel Traffic Flow Forecasting (VTFF).

4.3 Experimental Results

In this section, we train and evaluate our VRF model against its Centralized, and Cross-silo FL variants, and assess the privacy preservation of the latter using Differential Privacy.

4.3.1 Collaboration using Centralized ML

Suppose we have three partners, namely "Naval Academy Research Institute" (NARI), "Norwegian Coastal Administration" (NCA), and "University of Piraeus Research Center"

(UPRC), who provide the Brest, Norway, and Piraeus datasets, respectively, who aim to create a VRF model within the context of a research project. Having agreed on the architecture of the model, a baseline approach is to train three VRF instances (i.e., one for each of the aforementioned datasets/partners) in centralized fashion, and assess their performance against the other datasets, in order to converge on a single model.

After preparing the dataset using the procedure described at Chapter 4.1, we get 9,285, 5,922, and 15,262 trajectories from Brest, Norway and Piraeus datasets, respectively, which are split into training, validation and test sets with 70:20:10 split ratio. After training our VRF instances for 170 epochs (with early stopping) on each aforementioned dataset, Figure 4.2 illustrates the models' displacement error with respect to Δt for the test sets of each partners' dataset. At first, we observe that the VRF instance trained on the Piraeus dataset (c.f. Figure 4.2g,h,i), as expected (in the sense that it consists of a larger sample population), performs the best on all three datasets with displacement error ≈ 3.5 km on average for $\Delta t = 25-30$ min. on the Piraeus' test set, with Brest and Norway yielding ≈ 7.5 and ≈ 15 km on average for $\Delta t = 25-30$ min., on their corresponding datasets, respectively.

Further following this hypothesis, we expect, due to the population of the training set, that the VRF instance trained on the Norway and Brest datasets, respectively, will both perform the least, with the latter instance performing slightly better, consistently being between the Norway and Piraeus VRF instances.

Indeed, the model trained on the Norway dataset (c.f. Figure 4.2d,e,f) yields sub-optimal performance with ≈ 17 km on average for $\Delta t = 25-30$ min., on its corresponding dataset, and ≈ 9 and ≈ 4 km, on Brest and Piraeus datasets, respectively. Similarly, the displacement error of the VRF instance trained on the Brest dataset (c.f. Figure 4.2a,b,c) is consistently between the two aforementioned models, with ≈ 8 km on its corresponding dataset, and ≈ 15 and ≈ 3.8 km, on Norway and Piraeus datasets, respectively.

In general, we observe that both Norway, and Brest VRF instances constantly underperform, independent on which dataset are trained on, while the Piraeus' VRF instance consistently yields comparatively better predictions, i.e., has the smallest displacement error, on all aforementioned datasets. Therefore, in the collaboration scenario among NARI, NCA, and UPRC, the latter will be at an advantageous position since in whichever VRF instance, the displacement error for the Piraeus dataset remains (on average) consistently identical.

In more detail, Figure 4.3 illustrates the corresponding learning curves for each VRF model instance. We observe that the instances trained on Norway and Brest datasets appear to be underfit, while the instance trained on the Piraeus dataset appears to be well trained, conclusions which are reflected on their corresponding prediction performance, as Figure 4.2 illustrates. This behaviour is well justified, as the amount of training samples on the former two instances is insufficient for training a VRF model with good performance.

Because the participating partners are in agreement regarding privacy concerns, and in an effort to increase the training samples' population, a natural extension of the baseline approach is to unify all datasets into a single entity, in order to train a single VRF

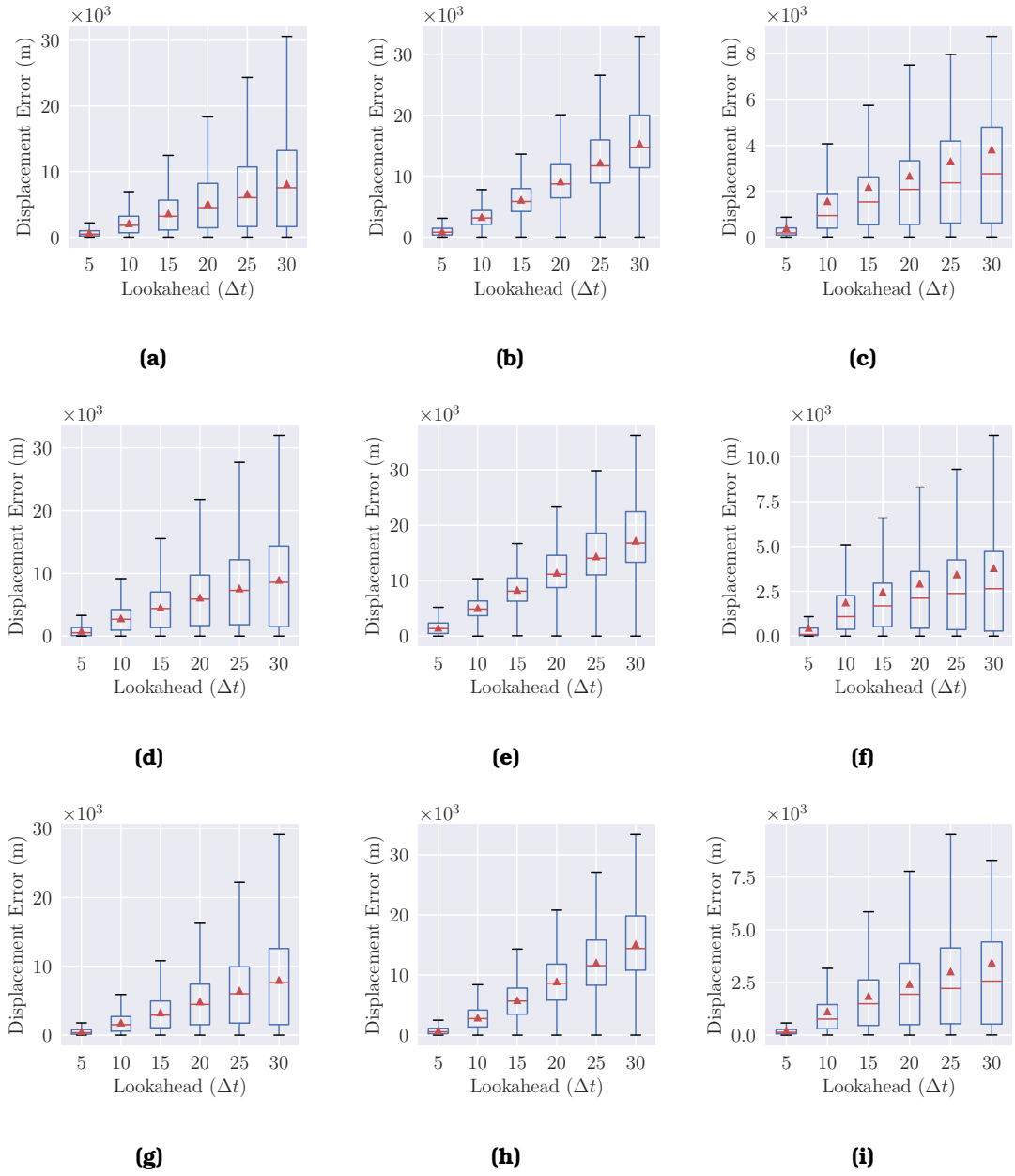

Figure 4.2. *Training three VRF instances in centralized fashion on Brest (a,b,c); Norway (d,e,f); and Piraeus (g,h,i) training set and assessing its displacement error on Brest (a,d,g); Norway (b,e,h); and Piraeus (c,f,i) test set.*

instance, which will potentially fit all parties' requirements. After training our "unified" model for 170 epochs (with early stopping), Figure 4.4 illustrates its displacement error on the Brest, Norway, and Piraeus datasets, respectively. We observe that, in all three datasets the aforementioned instance outperforms the previous instances, either significantly (≈ -3 km for $\Delta t = 25 - 30$ min.; c.f. Figure 4.4b vs. 4.2e) or marginally (≈ -0.1 and ≈ -0.2 km for $\Delta t = 25 - 30$ min.; c.f. Figures 4.4a vs. 4.2a, and 4.4c vs. 4.2i, respectively). Additionally, observing the learning curve of the model, we deduce that both the training and validation loss trajectory presents an underfitting behaviour, which is expected to some degree, mainly due to the datasets' heterogeneity, something that we

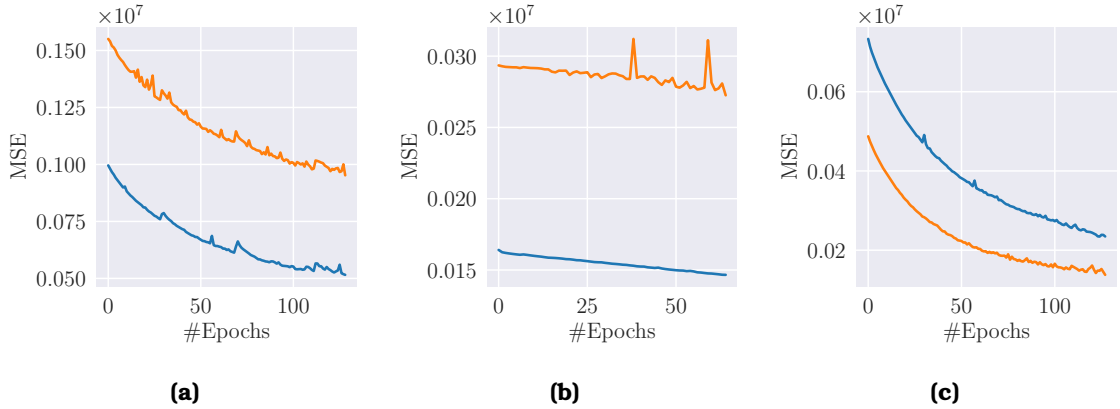

Figure 4.3. *Learning curves for (a) Brest; (b) Norway; and (c) Piraeus centralized VRF instances (blue and orange lines correspond to the training and validation sets, respectively).*

discuss in later sections.

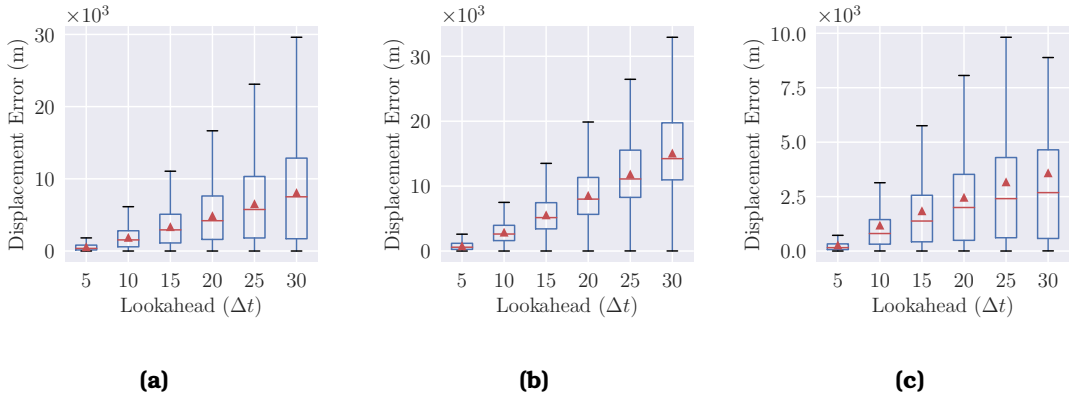

Figure 4.4. *Training a centralized VRF model on Brest, Norway, and Piraeus unified training set and assessing its displacement error on (a) Brest; (b) Norway; and (c) Piraeus test set.*

While the unified approach performs better than the baseline, in many real-world situations it is an unfeasible solution, mainly due to the increased costs in storage, bandwidth of transferring the data and setting up a unified database, as well as privacy concerns regarding, e.g., the transmitted locations of the partners' fleet. Therefore, in order to increase the training samples' population while ensuring a baseline privacy level, a natural extension of the Centralized ML (CML) approach, is to use DML techniques, such as FL in order to address the aforementioned issues.

4.3.2 Collaboration using Federated ML

Having created our three VRF instances, and trained them using CML, we proceed to use FL techniques in order to train a unified model, that is able to fit all partners' requirements (e.g., accurate prediction) without the need to centralize the datasets, and potentially compromise the users' privacy.

The server which previously was used to store the partners' datasets and the unified

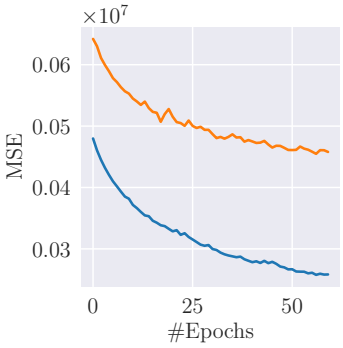

Figure 4.5. *Learning curve for the unified centralized VRF instance (blue and orange line corresponds to "train" and "dev" sets, respectively).*

VRF model, now will serve as an aggregation node, which receives the updated parameters of the partners' VRF models (i.e., local model), and aggregates them into a single entity (i.e., global model – FedVRF).

For training we use the FedAdam algorithm, as presented in Chapter 3.3 and the FLOWER [4] framework for instantiating the aggregation server, as well as the models' FL workers and local VRF models. After 170 epochs, Figure 4.6 illustrates the performance of the FL model in Brest, Norway, and Piraeus datasets. Compared to the displacement error of the centralized models (c.f. Figures 4.2a,e,i), we observe that the prediction error of FedVRF increased drastically (≈ 1 and ≈ 0.5 km) on Brest and Piraeus datasets, while it is marginally close (≈ 0.1 km) on Oslo dataset, respectively, for $\Delta t = 25 - 30$ min..

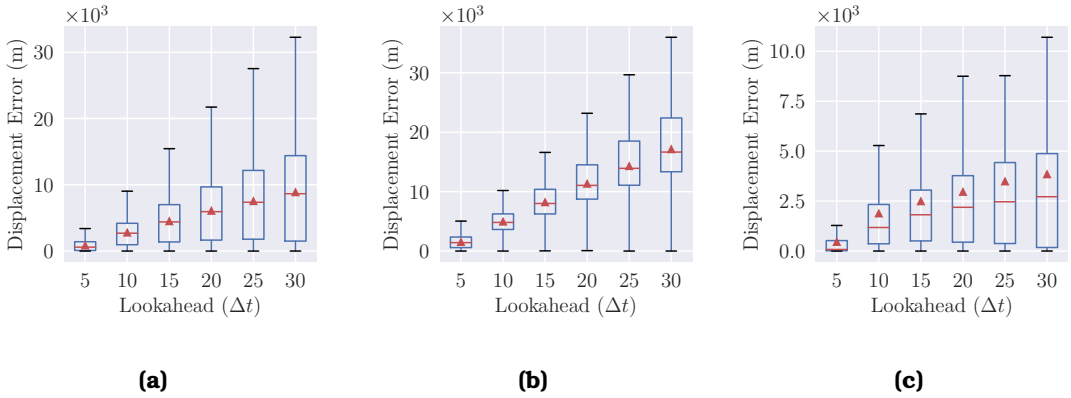

Figure 4.6. *Training a VRF model using FL on Brest, Norway and Piraeus training set and assessing its displacement error on (a) Brest; (b) Norway; and (c) Piraeus test set.*

In order to further understand the reason behind this behaviour, Figure 4.7 illustrates the learning curve of the partners' local VRF instance, compared to the learning curve of FedVRF. In general, We observe that in all three datasets, namely Brest, Norway, and Piraeus, the loss of the local models diverges from the global model by a large margin, a behaviour which is observed throught the training process, better known as "client-drift".

The main cause behind "client-drift" lies within the participating parties' heterogeneity. In particular, Figure 4.8 illustrates the Probability Density Function (PDF) of Brest,

Norway, and Piraeus datasets, respectively. We observe that the PDFs of Brest and Piraeus datasets seem to follow a unimodal distribution, which is easier for the VRF model to adapt to, as reflected by the rapidly downward trend in their corresponding learning curves at Figures 4.3a and 4.3c, respectively. On the other hand, the multimodal PDF of the Oslo dataset, introduces a high level of heterogeneity, which renders difficult the training process for the VRF model, something which is not only reflected at Figure 4.3b but also in the overall training process of FedVRF (c.f. Figure 4.6), where the aggregation (averaging) process of FedAdam (and FedAvg, in general), inhibits the training process.

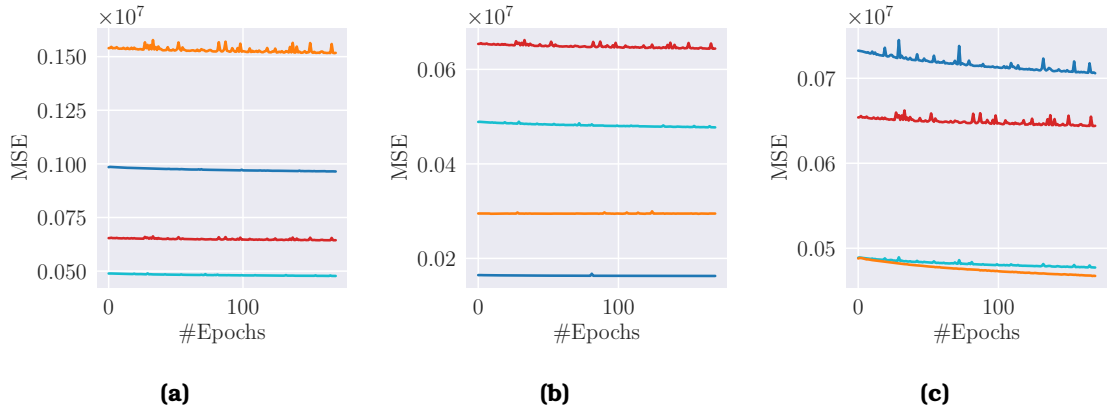

Figure 4.7. *Learning curve for (a) Brest; (b) Norway; and (c) Piraeus local VRF workers compared to FedVRF. Blue/orange lines correspond to the workers' training and validation loss, while cyan/red lines correspond to the training and validation loss of FedVRF, respectively.*

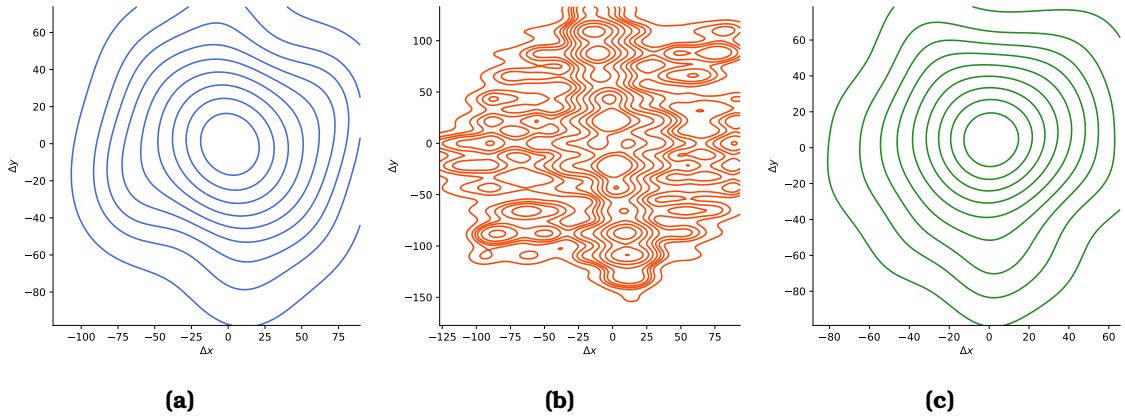

Figure 4.8. *Probability density functions (PDFs) of Δx and Δy of (a) Brest; (b) Norway; and (c) Piraeus datasets.*

Related works regarding "client-drift" (c.f. Chapter 2) recommend using fewer local updates and/or smaller learning rates. These action(s) however, largely impact the convergence stability of FedAvg (and its variants), as well as the learning "capacity" of the FL model, as Figure 4.7 illustrates for learning rate $\eta = 10^{-3}$. In order to properly address "client-drift" we use a branch of FL, called Personalized Federated Learning (PerFL), which can properly alleviate the aforementioned issue by ensuring a more uniform accuracy dis-

tribution across training parties [64].

4.3.3 Addressing Client Drift

As demonstrated in the previous section, models trained using FedAvg (or similar variants - e.g., FedAdam) on heterogeneous datasets, are prone to client drift, with its severity level being correlated to the amount of client heterogeneity (c.f. Figure 4.8). In order to effectively address that issue, we use a variant of FedAvg from the PerFL method family, named qFedAvg [38], which is illustrated at Algorithm 4.1. In a nutshell, it emphasizes on training parties (as q increases) with higher local empirical losses $F_k(w)$, thus imposing more uniformity to the training accuracy distribution and potentially creating a model flexible enough to fit all parties' needs.

In our experiments, we use a variant of qFedAvg in which the selected devices, update their corresponding weights using Adam [31], instead of SGD, tuned using their default values set at [4]. After 170 epochs, Figure 4.9 illustrates the performance of the PerFL model in Brest, Norway, and Piraeus datasets, respectively. In general, we observe that PerFL greatly outperforms the FL model, yielding smaller displacement errors up to ≈ 2.5 km on the Norway dataset, and up to ≈ 1 and ≈ 0.7 km on Piraeus and Brest datasets, respectively, for $\Delta t \in [5, 30)$ min.

Algorithm 4.1: qFedAvg, adapted from [38]

Input: $K, E, T, q, 1/L, \eta, w^0, p_k, k = 1, \ldots, m$
for $t = 0, \ldots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all selected devices
  Each selected device k updates w^t for E epochs of **Adam** on F_k with step size η, to obtain $\hat{w}_k^{t+1}$
  Each selected device k computes:
    $\Delta w_k^t = L(w^t - \hat{w}_k^{t+1}$
    $\Delta_k^t = F_k^q(w^t)\Delta w_k^t$
    $h_k^t = qF_k^{q-1}(w^t)\|\Delta w_k^t\|^2 + LF_k^q(w^t)$
  Each selected device k sends Δ_k^t and h_k^t back to the server
  Server updates w^{t+1} as:
    $$w^{t+1} = w^t - \frac{\sum_{k \in S_t} \Delta_k^t}{\sum_{k \in S_t} h_k^t}$$
end

Compared to the baseline CML approach (c.f. Figure 4.2a,e,i), we observe similar results on Norway dataset, while on Brest and Piraeus datasets, the performance is marginally worse, with displacement error up to 0.15 and 0.1 km larger, on average, respectively. Additionally, compared to the unified CML approach (c.f. Figure 4.4), we observe marginally worse results, as well, with displacement error up to 0.2 km larger, on average, on all datasets.

Further focusing on the learning curve of the partners' local models vs. the global (personalized) FedVRF model (c.f. Figure 4.10), we observe that the training/validation loss

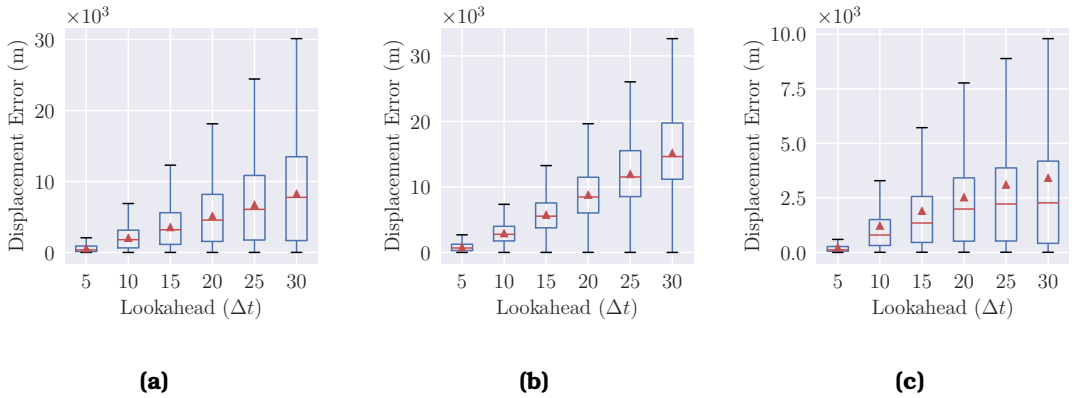

Figure 4.9. *Training a VRF model using PerFL on Brest, Norway and Piraeus training set and assessing its displacement error on (a) Brest; (b) Norway; and (c) Piraeus test set.*

curves of the Brest and Piraeus follow the same downward trend as their corresponding CML models (c.f. Figure 4.3), albeit with some added noise, mainly due to the aggregation (averaging) process. Additionally, we observe that the local models' learning curves are closer to the curves of personalized FedVRF, therefore preserving minimal (compared to FL) "client-drift".

Particularly, in the case of the Norway dataset, we observe an upward trend with increasing oscillations (i.e. noise), indicating a slight underfitting issue on that particular VRF worker. This however is expected, mainly due to the complex PDF of the Norway dataset (c.f Figure 4.8), and further demonstrates the advantage of PerFL, as the introduced noise of this worker is smoothed out in the global model, thus not allowing the datasets' heterogeneity to inhibit the learning process.

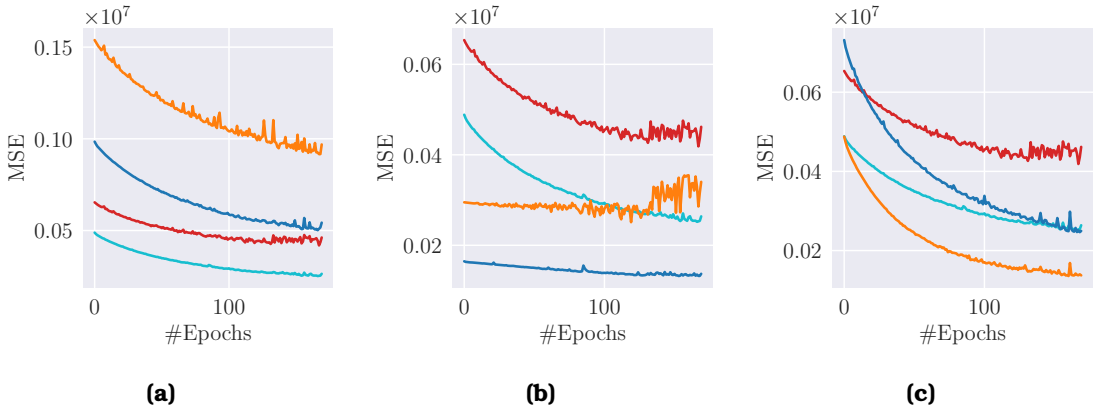

Figure 4.10. *Learning curve for (a) Brest; (b) Norway; and (c) Piraeus local VRF workers compared to personalized FedVRF. Blue/orange lines correspond to the workers' training and validation loss, while cyan/red lines correspond to the training and validation loss of personalized FedVRF, respectively.*

4.3.4 The Privacy Preservation Trade-off

While FL ensures (to a certain degree) user privacy, this assumption only holds when the aggregation server is considered to be honest-but-curious, or in other words, it has complete access to the models' weights/gradients (curious), but does not leak them (honest) to any participant either from inside, or from outside the federation cluster (e.g. an adversary). If the above assumption cannot hold, then FL may fail to ensure user privacy as the weights/gradients can be reverse engineered in order to recover (a part of) the dataset [22]. Towards this direction, we can either encode (e.g., via Homomorphic Encryption), or obfuscate the parties' models, in order to decrease the probability of data leakage.

In the scope of this thesis, we make use of Differential Privacy [13] (DP), as implemented in the Opacus [80] framework, in order to obfuscate the models' parameters by adding noise sampled from a Gaussian distribution to their corresponding parameters prior to sending them to the aggregation server. Practically, in this way, the training parties can have more control over their data, with the aggregation server still being capable of training the model, as the added noise tends to cancel out during the aggregation phase[6] [51].

After training our model using qFedAvg for 170 epochs with DP, Figure 4.11 illustrates the distribution of the displacement error on the Brest, Norway, and Piraeus datasets, respectively. We observe that, compared to non-DP PerFL, the added noise significantly increased the prediction error in all datasets, with average perturbation up to 2.5 km for the Norway dataset, and 0.8 km for Brest and 0.4 km Piraeus datasets, respectively.

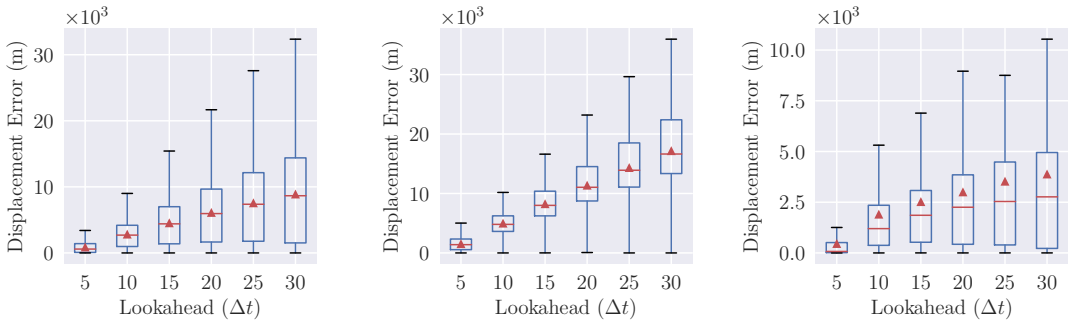

Figure 4.11. *Training a Differentially Private (DP) VRF model using PerFL on Brest, Norway and Piraeus training set and assessing its displacement error on (a) Brest; (b) Norway; (c) and Piraeus (c) test set.*

Further focusing on the models' learning curves at Figure 4.12 we observe immediately that the added noise severely impact the convergence of the global model, presenting not only higher levels of client drift (compared to non-DP PerFL), but also an extreme case of

[6]Facebook AI, Introducing Opacus: A high-speed library for training PyTorch models with differential privacy, https://ai.facebook.com/blog/introducing-opacus-a-high-speed-library-for-training-pytorch-models-with-differential-privacy/, Last visited: 2022/05/10

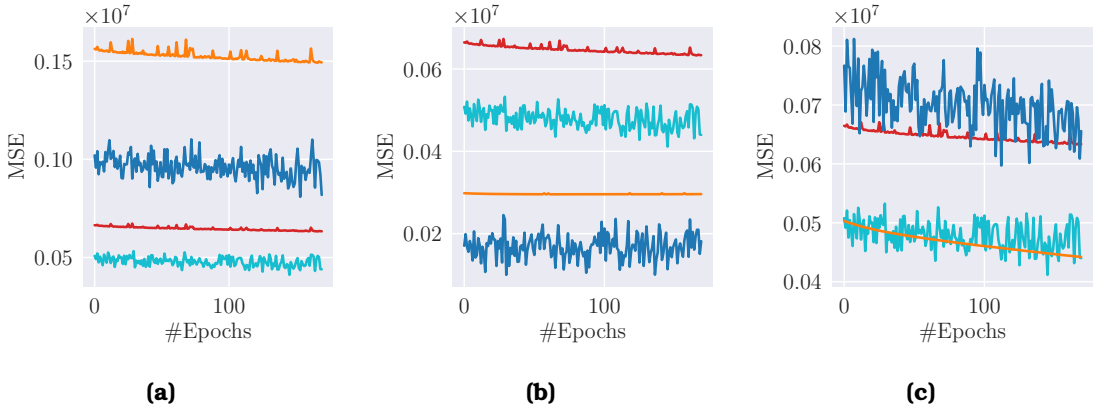

Figure 4.12. *Learning curve for DP-enabled (a) Brest; (b) Norway; and (c) Piraeus local VRF workers compared to personalized FedVRF. Blue/orange lines correspond to the workers' training and validation loss, while cyan/red lines correspond to the training and validation loss of personalized FedVRF, respectively.*

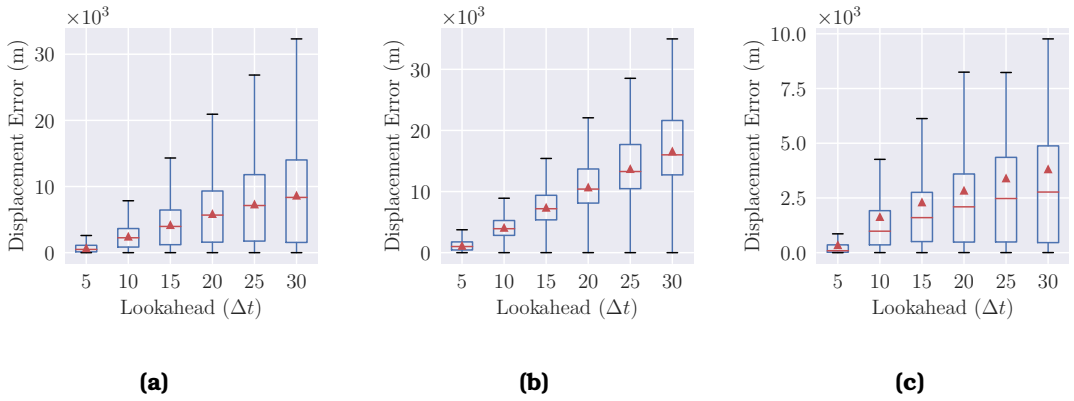

Figure 4.13. *Training a Differentially Private (DP) VRF model using PerFL on pretrained Brest, Norway and Piraeus corresponding CML model and training set and assessing its displacement error on (a) Brest; (b) Norway; and (c) Piraeus test set.*

underfitting, as deduced by the curves' "plateauing" behaviour. In an effort to improve the aforementioned phenomena, one technique, presented by Papernot et al. [47] is to use pretrained ML model instances, and via DP, produce a private model.

In our case, we use the VRF instances that were trained on Brest, Norway, and Piraeus datasets using CML, and through DP-enabled qFedAvg, train a privacy-aware global model that potentially can outperform the previous FedVRF instance. After 170 FL rounds, Figure 4.13 illustrates the performance of the global model on all three aforementioned datasets. While we indeed outperformed the non-pretrained DP-enabled FedVRF model, the difference between the two models is quite marginal (up to 0.5 km), especially when compared to the personalized FedVRF instance, where the displacement error difference is up to 2, 0.7, and 0.4 km on the Norway, Brest, and Piraeus datasets, respectively.

Comparing the models' learning curves (c.f. Figure 4.14), we observe an interesting phenomenon. In general, the learning curves of local and global models are closer compared to the non-pretrained variant (indicating less client drift; c.f. Figure 4.12).

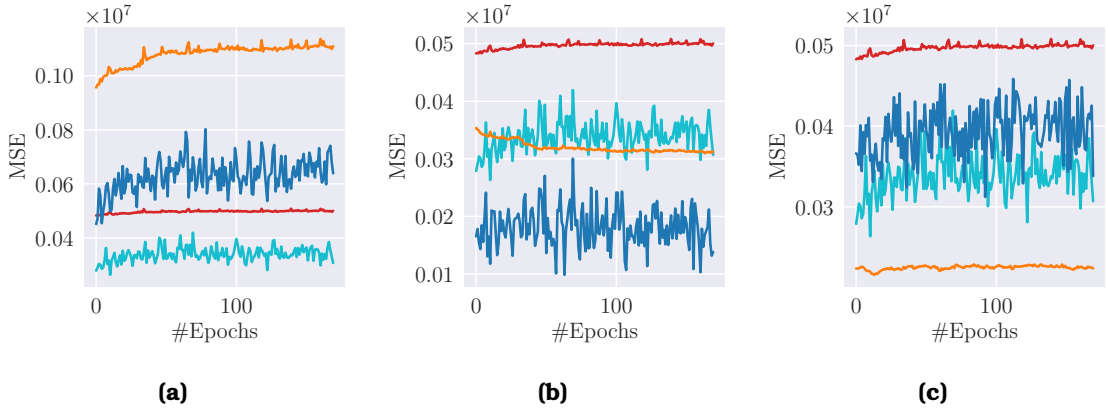

Figure 4.14. *Learning curve for DP-enabled (a) Brest; (b) Norway; and (c) Piraeus local VRF workers (/w pretrained CML models) compared to personalized FedVRF. Blue/orange lines correspond to the workers' training and validation loss, while cyan/red lines correspond to the training and validation loss of personalized FedVRF, respectively.*

However, especially in the Brest and Oslo datasets, the training loss follows an upward trend, instead of an downward one.

While in other ML workflows/frameworks, this could indicate a potentially major issue in the training process, in our case it is expected, since the models have been separately pretrained, thus they have converged on different local minima. Due to the "Fairness" mechanism of qFedAvg, the participants' training losses are combined in order to increase the fairness/uniformity of models' performance while maintaining their corresponding average performance, therefore some local losses (e.g. Brest) tend to increase, and others (e.g. Oslo) to decrease.

Another experiment, revolves around the trade-off between prediction accuracy and privacy budget (ϵ). In DP, we have two additional hyperparameters, namely ϵ, and δ (c.f. Equation 2.1). As defined in Chapter 2.2, ϵ is the privacy parameter which can be controlled by the data analyst to maintain the trade-off between privacy and accuracy, and δ is the probability of information accidentally being leaked [1].

Regarding δ, a rule of thumb is to be less than the inverse of the size of the training dataset[7]. With that into account we choose delta to the inverse of the upper order of magnitude of the samples' population[8] $\left(\text{e.g., \#samples} = 1024 \leq 10^5; \delta = 1/10^5\right)$, or in other words $\delta = 10^{-5}$, 10^{-6}, and 10^{-5} for Brest, Norway, and Piraeus datasets, respectively. On the other hand, for ϵ there are some values that we can use as a reference (c.f. Chapter 2.2), as well as works towards approximating it [24, 36], within the scope of this thesis we will use a baseline approach in which we set ϵ to the tightest value of total privacy spent among the three workers. During the previous experiment, where the workers had - virtually - unlimited privacy budget ($\epsilon = \infty$), the total privacy budget spent was $\epsilon = 162.2$, 247.683, 110.692 for the Brest, Norway, and Piraeus datasets, respectively, thus we set

[7] "What does epsilon=1.1 really mean? How about delta?", Opacus FAQ, https://opacus.ai/docs/faq#what-does-epsilon11-really-mean-how-about-delta, Last visited: 07/04/2022

[8] "Building an Image Classifier with Differential Privacy", GitHub, https://github.com/pytorch/opacus/blob/main/tutorials/building_image_classifier.ipynb, Last visited: 07/04/2022

$\epsilon = 110$.

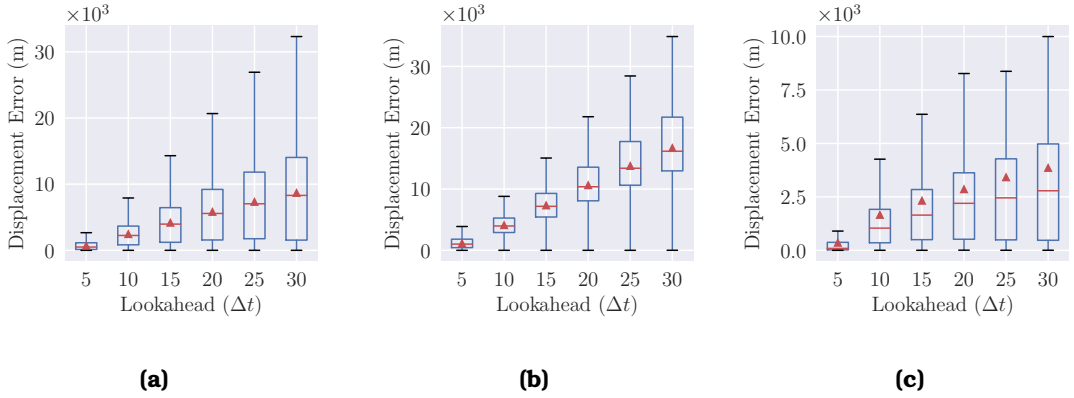

Figure 4.15. *Training a (ϵ, δ) Differentially Private (DP) VRF model using PerFL on pretrained Brest, Norway and Piraeus corresponding CML model and training set and assessing its displacement error on (a) Brest; (b) Norway; and (c) Piraeus test set.*

After 170 epochs, Figure 4.15 illustrates the displacement error distribution of the global DP-enabled FedVRF model on all three aforementioned datasets. Comparing it to the previous DP-enabled FedVRF experiment (c.f. Figure 4.13), restricting the privacy budget ϵ does little impact on the prediction error ($\approx$ 0.1 km for $\Delta t = 25 - -30$ min.), thus placing (ϵ, δ)-DP- between plain DP-, and pretrained DP-enabled FedVRF. Similarly, the models' corresponding learning curves at Figure 4.16 show similar behaviour as the pretrained DP-enabled FedVRF ($\epsilon = \infty$; c.f. Figure 4.13), albeit with slightly higher introduced noise (due to the restricted privacy budget), but with less client drift, therefore in general, we successfully traded-off model accuracy for higher user privacy.

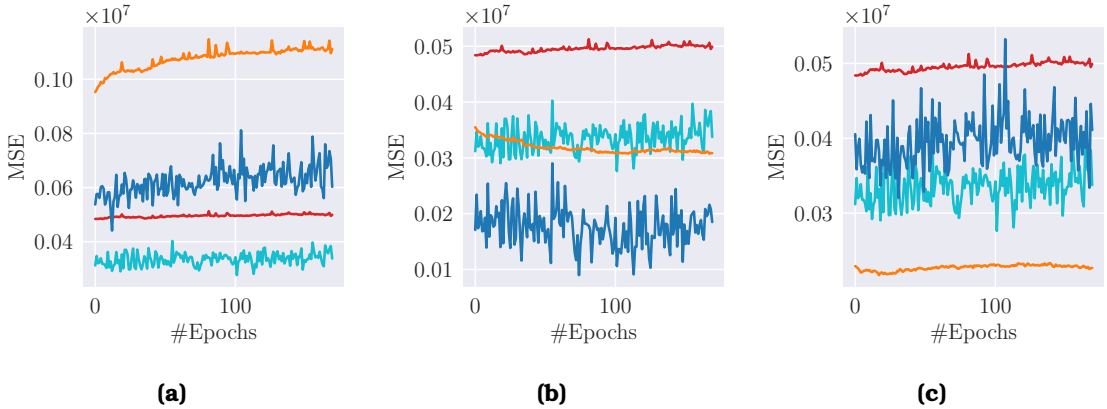

Figure 4.16. *Learning curve for (ϵ, δ) DP-enabled (a) Brest; (b) Norway; and (c) Piraeus local VRF workers (/w pretrained CML models) compared to personalized FedVRF. Blue/orange lines correspond to the workers' training and validation loss, while cyan/red lines correspond to the training and validation loss of personalized FedVRF, respectively.*

4.4 Discussion and Exploitation

Towards the real-world assessment of FedVRF, Figure 4.17 illustrates the predicted trajectory of a randomly selected cargo vessel in Norway and Piraeus, as well as a Tanker vessel in Brest dataset, respectively. We observe that in all three cases the personalized FedVRF consistently outperforms all other VRF variants by a large margin, including the CML ones, which are close to the PerFL solution following a similar trajectory, albeit with larger deviation from the actual route.

Additionally, the added noise within the DP-enabled FedVRF models, renders them unable to properly predict the vessels' future route, being relatively accurate for the first $\Delta t \approx 5$ minutes, and afterwards making a "U-turn" before returning to where the trajectory began. Similar results can also be found in plain FedVRF, where the "client-drift" has greatly impact the vessels' future route prediction, by deviating (and making the "U-turn") almost at the first $\Delta t \approx 5$ minutes.

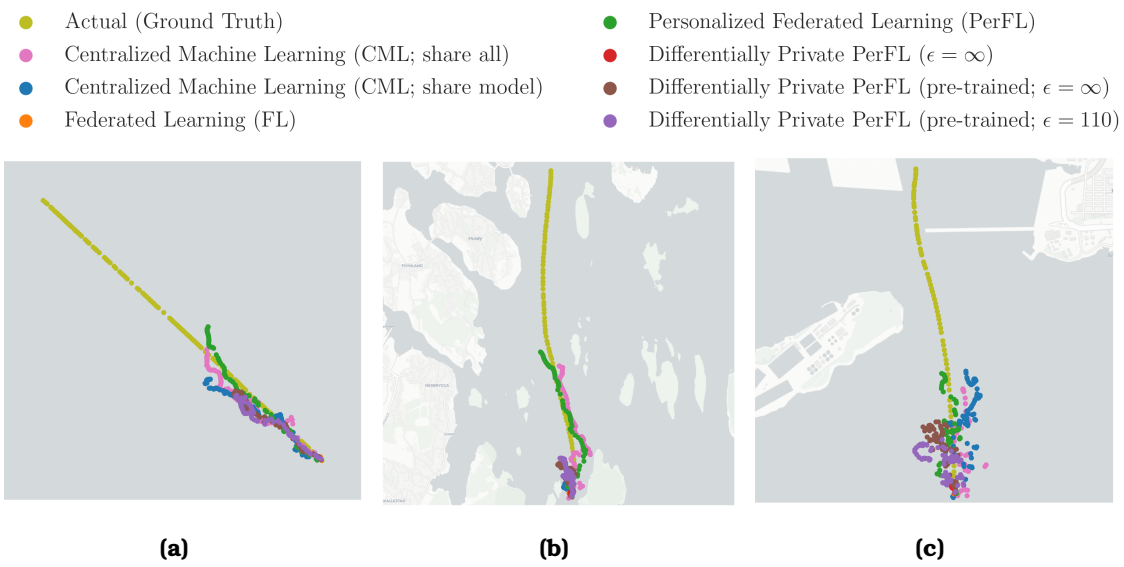

Figure 4.17. *Predicting the trajectory of a vessel up to $\Delta t = 15$ min. using all (Fed-)VRF variants on (a) Brest, (b) Norway, and (c) Piraeus dataset.*

Towards the exploitation of FedVRF on maritime traffic control, Figure 4.18 illustrates the traffic density within the Piraeus dataset for a period up to $\Delta t = 15$ min. Focusing on popular Cargo/Tanker and Passenger vessels' routes, we observe that compared to the actual traffic density, both CML VRF models anticipate increased traffic on passenger routes, e.g., from Piraeus to Aegina, as well as increased intra-port traffic, either from official or unofficial ports, such as the Tankers'/Cargos' anchorage [69] south-east of Salamina.

On the other hand, the FedVRF model anticipates higher intra-port traffic, with moderate emphasis on popular passenger routes, and less focus on the cargos' anchorages. Similarly, the personalized FedVRF model predicts increased traffic flow on popular vessels' routes, ports, and anchorages, albeit with an increased interest within the inner Saronic Gulf, where higher flow is anticipated. Finally, the DP-enabled FedVRF solutions,

all provide similar results to personalized VRF, with slight variations on the predicted flow within passengers' routes. These findings may trigger domain experts into further investigating these occurences and reach some meaningful conclusions.

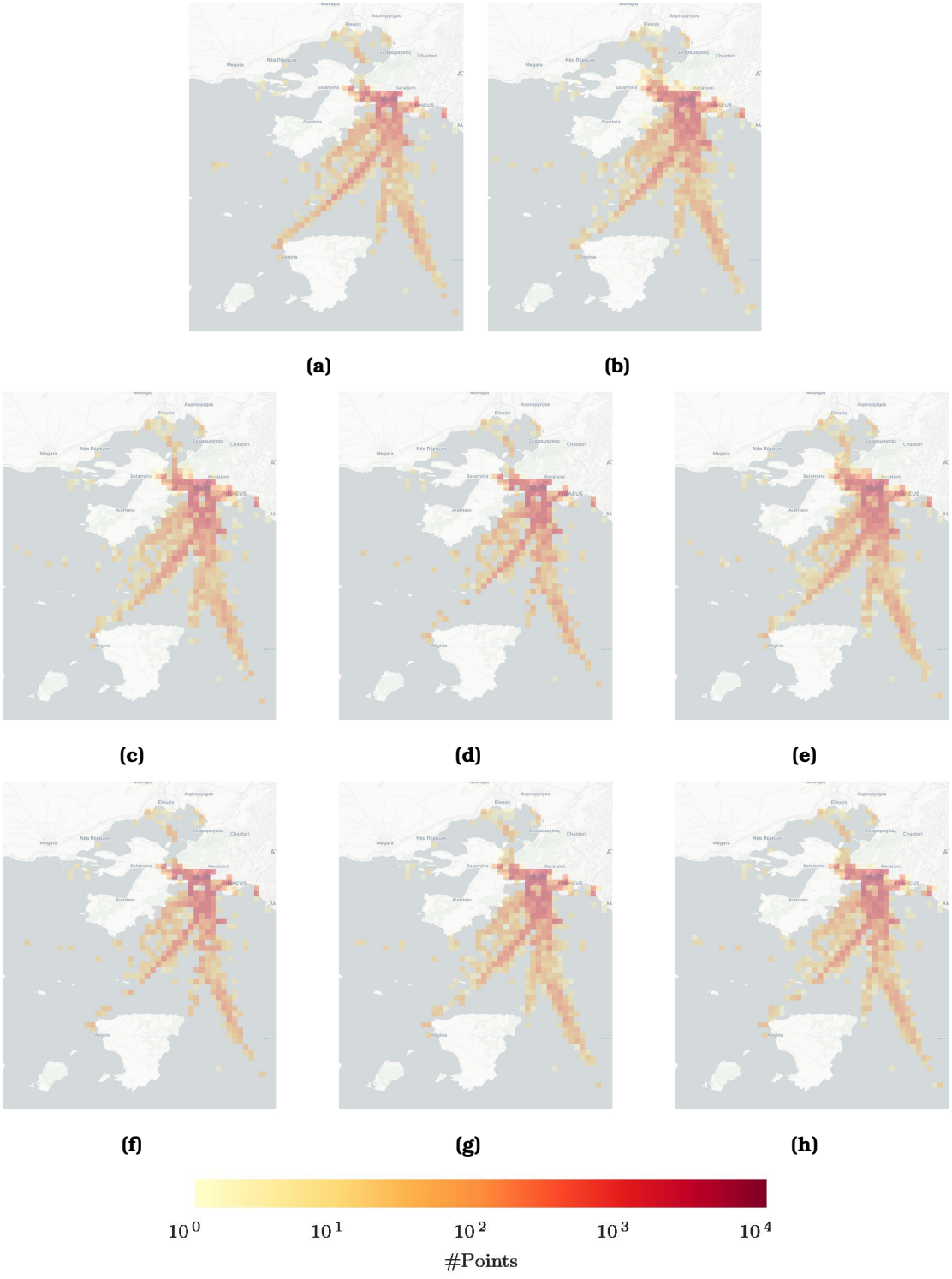

Figure 4.18. *Predicting maritime traffic flow up to $\Delta t = 15$ min. using all (Fed-)VRF variants, namely, (b) CML (share all), (c) CML (share model), (d) FL, (e) PerFL, (f) DP-PerFL ($\epsilon = \infty$), (g) DP-PerFL (pretrained; $\epsilon = \infty$), and (h) DP-PerFL ($\epsilon = 110$) compared to (a) actual traffic flow on Piraeus dataset.*

Chapter 5

Conclusions and Future Work

In this thesis, we studied Federated Learning (FL) from both a theoretical and algorithmic perspective, and compared its advantages and disadvantages to the centralized approach, based on the task of Vessel Route Forecasting (VRF). Our experimental study on four real-world AIS datasets demonstrates the advances and open problems of FL, as well as the advantages of Personalized Federated Learning (PerFL) over highly heterogeneous datasets'. In the near future, we aim to further optimize the architecture of the VRF model, in order to decrease its displacement error, and render it suitable not only for short-term, but also long-term prediction as well. Additionally, we aim to further experiment on PerFL algorithms by fine-tuning the existing algorithms as well as implementing newer algorithms, such as [15]. Moreover we aim to further exploit on privacy-preservation mechanisms, by adding more data silos, in order to further leverage the properties of Differential Privacy (DP). Finally, we aim to extend the applications' scope of FedVRF into the scope of maritime transportation safety, and more specifically in Vessel Collision Risk Assessment (VCRA) [67], therefore shifting to Federated Vessel Collision Risk Assessment (FedVCRF).

Bibliography

[1] Aitsam, M. Differential privacy made easy. *CoRR abs/2201.00099* (2022).

[2] Ben-Nun, T., and Hoefler, T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR) 52*, 4 (2019), 1–43.

[3] Benterki, A., Judalet, V., Maaoui, C., and Boukhnifer, M. Long-term prediction of vehicle trajectory using recurrent neural networks. In *IECON* (2019), IEEE, pp. 3817–3822.

[4] Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., and Lane, N. D. Flower: A friendly federated learning research framework. *CoRR abs/2007.14390* (2020).

[5] Capobianco, S., Millefiori, L. M., Forti, N., Braca, P., and Willett, P. Deep learning methods for vessel trajectory prediction based on recurrent neural networks. *IEEE Trans. Aerosp. Electron. Syst. 57*, 6 (2021), 4329–4346.

[6] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP* (2014), ACL, pp. 1724–1734.

[7] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A. W., Tucker, P. A., Yang, K., and Ng, A. Y. Large scale distributed deep networks. In *NIPS* (2012), pp. 1232–1240.

[8] Desjardins, J. Historical ais data in norwegian waters. https://ais-public.kystverket.no/ais-download. Last Visited: 2022/02/01.

[9] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)* (2019), Association for Computational Linguistics, pp. 4171–4186.

[10] Dey, R., and Salem, F. M. Gate-variants of gated recurrent unit (GRU) neural networks. In *Proceedings of the IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (2017), pp. 1597–1600.

[11] Dinh, C. T., Tran, N. H., Nguyen, M. N. H., Hong, C. S., Bao, W., Zomaya, A. Y., and Gramoli, V. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Trans. Netw. 29*, 1 (2021), 398–409.

[12] Duchi, J. C., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res. 12* (2011), 2121–2159.

[13] Dwork, C., and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci. 9*, 3-4 (2014), 211–407.

[14] Ester, M., Kriegel, H., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD* (1996), AAAI Press, pp. 226–231.

[15] Fallah, A., Mokhtari, A., and Ozdaglar, A. E. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *NeurIPS* (2020).

[16] Galakatos, A., Crotty, A., and Kraska, T. Distributed machine learning. In *Encyclopedia of Database Systems (2nd ed.)*. Springer, 2018.

[17] Gentry, C. Fully homomorphic encryption using ideal lattices. In *STOC* (2009), ACM, pp. 169–178.

[18] Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K. E., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML* (2016), vol. 48 of *JMLR Workshop and Conference Proceedings*, JMLR.org, pp. 201–210.

[19] Han, P., Wang, W., Shi, Q., and Yang, J. Real-time short-term trajectory prediction based on gru neural network. In *Proceedings of the 38th Digital Avionics Systems Conference (DASC)* (2019), pp. 1–8.

[20] Hard, A., Kiddon, C. M., Ramage, D., Beaufays, F., Eichner, H., Rao, K., Mathews, R., and Augenstein, S. Federated learning for mobile keyboard prediction, 2018.

[21] Hesamifard, E., Takabi, H., and Ghasemi, M. Cryptodl: Deep neural networks over encrypted data. *CoRR abs/1711.05189* (2017).

[22] Hitaj, B., Ateniese, G., and Pérez-Cruz, F. Deep models under the GAN: information leakage from collaborative deep learning. In *CCS* (2017), ACM, pp. 603–618.

[23] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computation 9*, 8 (1997), 1735–1780.

[24] Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., and Roth, A. Differential privacy: An economic method for choosing epsilon. In *CSF* (2014), IEEE Computer Society, pp. 398–410.

[25] Jia, Z., Lin, S., Qi, C. R., and Aiken, A. Exploring hidden dimensions in accelerating convolutional neural networks. In *Proceedings of the 35th International Conference on Machine Learning* (2018), vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 2274–2283.

[26] Jia, Z., Zaharia, M., and Aiken, A. Beyond data and model parallelism for deep neural networks. In *Proceedings of Machine Learning and Systems* (2019), A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, pp. 1–13.

[27] Jia, Z., Zaharia, M., and Aiken, A. Beyond data and model parallelism for deep neural networks. In *MLSys* (2019), mlsys.org.

[28] Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. P. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security Symposium* (2018), USENIX Association, pp. 1651–1669.

[29] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Found. Trends Mach. Learn. 14*, 1-2 (2021), 1–210.

[30] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. SCAFFOLD: stochastic controlled averaging for federated learning. In *ICML* (2020), vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 5132–5143.

[31] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)* (2015).

[32] Konečný, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR abs/1610.02527* (2016).

[33] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *CoRR abs/1610.05492* (2016).

[34] Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *CoRR abs/1404.5997* (2014).

[35] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (2012), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25.

[36] Lee, J., and Clifton, C. How much is enough? choosing ϵ for differential privacy. In *ISC* (2011), vol. 7001 of *Lecture Notes in Computer Science*, Springer, pp. 325–340.

[37] Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., and Chintala, S. Pytorch distributed: Experiences on accelerating data parallel training. *CoRR abs/2006.15704* (2020).

[38] Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *ICLR* (2020), OpenReview.net.

[39] Liu, H., Wu, H., Sun, W., and Lee, I. Spatio-temporal GRU for trajectory classification. In *ICDM* (2019), IEEE, pp. 1228–1233.

[40] Liu, T., Chen, W., and Wang, T. Distributed machine learning: Foundations, trends, and practices. In *WWW (Companion Volume)* (2017), ACM, pp. 913–915.

[41] Liu, Y., Chen, T., and Yang, Q. Secure federated transfer learning. *CoRR abs/1812.03337* (2018).

[42] Lo, S. K., Lu, Q., Wang, C., Paik, H., and Zhu, L. A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Comput. Surv. 54*, 5 (2021), 95:1–95:39.

[43] Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., and Hellerstein, J. M. Graphlab: A new framework for parallel machine learning. *CoRR abs/1006.4990* (2010).

[44] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS* (2017), vol. 54 of *Proceedings of Machine Learning Research*, PMLR, pp. 1273–1282.

[45] McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. Federated learning of deep networks using model averaging. *CoRR abs/1602.05629* (2016).

[46] Mohammadi, M., Al-Fuqaha, A., Sorour, S., and Guizani, M. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials 20*, 4 (2018), 2923–2960.

[47] Papernot, N., Thakurta, A., Song, S., Chien, S., and Erlingsson, Ú. Tempered sigmoid activations for deep learning with differential privacy. In *AAAI* (2021), AAAI Press, pp. 9312–9321.

[48] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS* (2019), pp. 8024–8035.

[49] Petrou, P., Nikitopoulos, P., Tampakis, P., Glenis, A., Koutroumanis, N., Santipantakis, G. M., Patroumpas, K., Vlachou, A., Georgiou, H. V., Chondrodima, E., Doulkeridis, C., Pelekis, N., Andrienko, G. L., Patterson, F., Fuchs, G., Theodoridis, Y.,

and Vouros, G. A. ARGO: A big data framework for online trajectory prediction. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019* (2019), pp. 194–197.

[50] Petrou, P., Tampakis, P., Georgiou, H. V., Pelekis, N., and Theodoridis, Y. Online long-term trajectory prediction based on mined route patterns. In *MASTER@ECML-PKDD 2019* (2019), pp. 34–49.

[51] Qardaji, W. H., Yang, W., and Li, N. Differentially private grids for geospatial data. In *ICDE* (2013), IEEE Computer Society, pp. 757–768.

[52] Qiu, X., Parcollet, T., Fernández-Marqués, J., de Gusmão, P. P. B., Beutel, D. J., Topal, T., Mathur, A., and Lane, N. D. A first look into the carbon footprint of federated learning. *CoRR abs/2102.07627* (2021).

[53] Ray, C., Dreo, R., Camossi, E., Jousselme, A.-L., and Iphar, C. Heterogeneous integrated dataset for maritime intelligence, surveillance, and reconnaissance. *Data in Brief 25* (2019), 104–141.

[54] Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *ICLR* (2021), OpenReview.net.

[55] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K. H., Ourselin, S., Sheller, M. J., Summers, R. M., Trask, A., Xu, D., Baust, M., and Cardoso, M. J. The future of digital health with federated learning. *CoRR abs/2003.08119* (2020).

[56] Rivest, R. L., Adleman, L., and Dertouzos, M. L. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press* (1978), 169–179.

[57] Rossi, A., Barlacchi, G., Bianchini, M., and Lepri, B. Modelling taxi drivers' behaviour for the next destination prediction. *IEEE Trans. Intell. Transp. Syst. 21*, 7 (2020), 2980–2989.

[58] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323* (1986), 533–536.

[59] Sabater, C., Bellet, A., and Ramon, J. Distributed differentially private averaging with improved utility and robustness to malicious parties. *CoRR abs/2006.07218* (2020).

[60] Shokri, R., and Shmatikov, V. Privacy-preserving deep learning. In *Allerton* (2015), IEEE, pp. 909–910.

[61] Song, S., Chaudhuri, K., and Sarwate, A. D. Stochastic gradient descent with differentially private updates. In *GlobalSIP* (2013), IEEE, pp. 245–248.

[62] Suo, Y., Chen, W., Claramunt, C., and Yang, S. A ship trajectory prediction framework based on a recurrent neural network. *Sensors 20*, 18 (2020), 5133.

[63] Tampakis, P., Pelekis, N., Doulkeridis, C., and Theodoridis, Y. Scalable distributed subtrajectory clustering. In *2019 IEEE International Conference on Big Data (Big Data)* (2019), IEEE, pp. 950–959.

[64] Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–17.

[65] Trasarti, R., Guidotti, R., Monreale, A., and Giannotti, F. Myway: Location prediction via mobility profiling. *Inf. Syst. 64* (2017), 350–367.

[66] Trask, A. *Grokking Deep Learning*, 1st ed. Manning Publications Co., 2019.

[67] Tritsarolis, A., Chondrodima, E., Pelekis, N., and Theodoridis, Y. Vessel Collision Risk Assessment using AIS Data: A Machine Learning Approach. In *MBDW* (2022), ACM, pp. 170–173.

[68] Tritsarolis, A., Chondrodima, E., Tampakis, P., and Pikrakis, A. Online co-movement pattern prediction in mobility data. In *EDBT/ICDT Workshops* (2021), vol. 2841 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[69] Tritsarolis, A., Kontoulis, Y., Pelekis, N., and Theodoridis, Y. Masec: Discovering anchorages and co-movement patterns on streaming vessel trajectories. In *SSTD* (2021), ACM, pp. 170–173.

[70] Tritsarolis, A., Kontoulis, Y., and Theodoridis, Y. The piraeus ais dataset for large-scale maritime data analytics. *Data in Brief 40* (2022), 107782.

[71] Vepakomma, P., Swedish, T., Raskar, R., Gupta, O., and Dubey, A. No peek: A survey of private distributed deep learning. *CoRR abs/1812.03288* (2018).

[72] Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J. S. A survey on distributed machine learning. *ACM Computing Surveys (CSUR) 53*, 2 (2020), 1–33.

[73] Wang, M., Huang, C., and Li, J. Unifying data, model and hybrid parallelism in deep learning via tensor tiling. *CoRR abs/1805.04170* (2018).

[74] Wang, S., Cao, J., and Yu, P. S. Deep learning for spatio-temporal data mining: A survey. *CoRR abs/1906.04928* (2019).

[75] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (1990), 1550–1560.

[76] Xue, H., Huynh, D. Q., and Reynolds, M. SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction. In *WACV* (2018), IEEE Computer Society, pp. 1186–1194.

[77] YANG, Q., LIU, Y., CHEN, T., AND TONG, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. 10*, 2 (2019), 12:1–12:19.

[78] YANG, Q., LIU, Y., CHENG, Y., KANG, Y., CHEN, T., AND YU, H. *Federated Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019.

[79] YAO, A. C. How to generate and exchange secrets (extended abstract). In *FOCS* (1986), IEEE Computer Society, pp. 162–167.

[80] YOUSEFPOUR, A., SHILOV, I., SABLAYROLLES, A., TESTUGGINE, D., PRASAD, K., MALEK, M., NGUYEN, J., GOSH, S., BHARADWAJ, A., ZHAO, J., CORMODE, G., AND MIRONOV, I. Opacus: User-friendly differential privacy library in pytorch. *CoRR abs/2109.12298* (2021).

[81] ZAHEER, M., REDDI, S. J., SACHAN, D. S., KALE, S., AND KUMAR, S. Adaptive methods for nonconvex optimization. In *NeurIPS* (2018), pp. 9815–9825.

[82] ZHANG, D., CHEN, X., WANG, D., AND SHI, J. A survey on collaborative deep learning and privacy-preserving. In *DSC* (2018), IEEE, pp. 652–658.

[83] ZHANG, H., LI, Y., DENG, Z., LIANG, X., CARIN, L., AND XING, E. P. Autosync: Learning to synchronize for data-parallel distributed deep learning. In *NeurIPS* (2020).

List of Abbreviations

CML	Centralized Machine Learning
CNN	Convolutional Neural Network
DL	Deep Learning
DML	Distributed Machine Learning
DP-SGD	Differentially Private Stochastic Gradient Descent
DSSGD	Distributed Selective Stochastic Gradient Descent
FedAvg	Federated Averaging
FedVRF	Federated Vessel Route Forecasting
FL	Federated Learning
FTL	Federated Transfer Learning
GDPR	General Data Protection Regulation
GRU	Gated Recurrent Unit
HE	Homomorphic Encryption
HFL	Horizontal Federated Learning
I.I.D	Independent and Identically Distributed
LSTM	Long Short-Term Memory
ML	Machine Learning
NN	Neural Network
PDF	Probability Density Function
PerFL	Personalized Federated Learning
PPML	Privacy-Preserving Machine Learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMPC	Secure Multi-Party Computation
VFL	Vertical Federated Learning
VRF	Vessel Route Forecasting
VTFF	Vessel Traffic Flow Forecasting