On Vessel Location Forecasting and the Effect of Federated Learning

Andreas Tritsarolis *Department of Informatics University of Piraeus* Piraeus, Greece andrewt@unipi.gr

Nikos Pelekis *Department of Statistics & Insurance Science University of Piraeus* Piraeus, Greece npelekis@unipi.gr

Konstantina Bereta *Kpler* Athens, Greece kbereta@kpler.com

Dimitris Zissis *Department of Product & Systems Design Engineering University of the Aegean* Syros, Greece

dzissis@aegean.gr

Yannis Theodoridis *Department of Informatics University of Piraeus* Piraeus, Greece ytheod@unipi.gr

Abstract—The wide spread of Automatic Identification System (AIS) has motivated several maritime analytics operations. Vessel Location Forecasting (VLF) is one of the most critical operations for maritime awareness. However, accurate VLF is a challenging problem due to the complexity and dynamic nature of maritime traffic conditions. Furthermore, as privacy concerns and restrictions have grown, training data has become increasingly fragmented, resulting in dispersed databases of several isolated data silos among different organizations, which in turn decreases the quality of learning models. In this paper, we propose an efficient VLF solution based on LSTM neural networks, in two variants, namely Nautilus and FedNautilus for the centralized and the federated learning approach, respectively. We also demonstrate the superiority of the centralized approach with respect to current state of the art and discuss the advantages and disadvantages of the federated against the centralized approach. *Index Terms*—Machine Learning, Privacy Preservation, Federated Learning, Mobility Data Analytics, Vessel Location Fore-

I. INTRODUCTION

Vessel Location Forecasting (VLF) is a critical task in the maritime domain because it can be used in several aspects of maritime mobility awareness, including, among others, fishing effort/pressure forecasting [\[1\]](#page-8-0), traffic flow management [\[2\]](#page-8-1), future collision avoidance [\[3\]](#page-8-2), as well as co-movement pattern discovery [\[4\]](#page-8-3). Informally, given a look-ahead time interval Δt _{next}, the goal is to predict the future location of a moving vessel s_j at Δt _{next} time after current timestamp $t_i^{s_j}$. Figure [1](#page-1-0) illustrates such an example, where given the vessels' current routes (black solid lines), VLF predicts their corresponding future locations (green points).

Nevertheless, the vast spread of IoT-enabled devices, such as sensors, smartwatches, smartphones, and GPS trackers, has led to the production of vast amounts of data, including mobility data. The availability of this volume of data is crucial to the success of Machine Learning (ML) technologies,

which can perform a variety of tasks that may sometimes exceed human performance [\[5\]](#page-8-4). Nevertheless, the information generated by the edge devices inherently consists of sensitive data and is frequently dispersed among numerous entities. These characteristics present novel challenges regarding the effective storage, analysis, and extraction of valuable insights from such data $[6]$, $[7]$.

Centralizing data to a certain location (e.g., data center) may become quite a cumbersome task because of the high storage/bandwidth costs (e.g., commercial maritime traffic systems monitor thousands of vessels per day, receiving several TBs of AIS information). In addition, sharing data entails several risks including disclosure of commercial information, trade secrets, and customer personal information^{[1](#page-0-0)}. Therefore, in some domains, collecting and sharing data may become quite difficult, if not outright impossible, thus forcing data owners to store them in isolated data silos. 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 datasets, 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. [\[8\]](#page-8-7) and Konečný et al. [\[9\]](#page-8-8) propose the Federated Learning (FL) paradigm, where a centralized model is trained on decentralized data. In particular, each edge device (or data silo, depending on the problem architecture) 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.

¹c.f., for instance, the General Data Protection Regulation (GDPR) in European Union: [https://ec.europa.eu/info/law/law-topic/data-protection](https://ec.europa.eu/info/law/law-topic/data-protection_en)_en.

To appear in the Proceedings of the 25th IEEE International Conference on Mobile Data Management (MDM), June 24 - June 27, 2024, Brussels, Belgium. ©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

casting

Repeating the process for several cycles may eventually cause the global model to converge, producing an ML model that competes, in terms of quality, the (local) model that each party can learn on its own [\[5\]](#page-8-4).

Using FL, the decentralized nature of the data is maintained, as the edge devices / data silos 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 / silo, therefore making it harder for an adversary to extract sensitive information. Distributing the training workload to multiple edge devices / silos, FL allows for potentially "smarter" models, lower inference latency, less overall power consumption, and by extension, lighter environmental impact [\[10\]](#page-8-9).

Fig. 1. Predicting the future locations of three moving objects (e.g., vessels in the maritime domain) s_1 , s_2 and s_3 . Given the vessels' current routes up to $t_{ni}^{s,j}$ (solid lines), VLF is used to predict vessels' future locations (green points) at three discrete timestamps, $t_n^{s_j} + \Delta t_{\text{next}} t_n^{s_j} + \Delta t'_{\text{next}}$, and $t_n^{s_j} + \Delta t'_{\text{next}}$ respectively.

In this paper, we propose Nautilus, an efficient VLF solution based on Neural Networks (NN), in particular the Long Short-Term Memory (LSTM) model, which will be shown (in Section [IV\)](#page-4-0) to outperform current state-of-the-art [\[11\]](#page-8-10) in the vast majority of cases at a prediction horizon Δt_{next} up to 60 min. Moreover, with the FL paradigm in mind, we extend Nautilus to an FL-oriented architecture, called FedNautilus, geared towards collaboratively training the LSTM model across multiple data silos (i.e., vessel traffic controllers, fleet owners, maritime ICT industries, etc.).

The approach described in this paper meets the industrial needs of vessel tracking and maritime intelligence companies, like Kpler. Kpler owns the largest terrestrial network of AIS receivers worldwide as well as MarineTraffic^{[2](#page-1-1)}, one of the best known platforms for real-time vessel tracking, which is based on AIS. However, AIS is a collaborative maritime reporting system, so sometimes vessels might become untraceable via AIS, for the following reasons: (a) equipment malfunction, (b) the vessel navigates into an area that is out of AIS coverage, (c) the vessel has its AIS transponder switched-off (e.g., when engaging in illegal activities). VLF gives us the opportunity to estimate the future location(s) of a vessel given the last known positions.

The main motivation behind the FL approach is that since AIS data are collected using a de-centralized network of AIS receivers, scattered all over the world, the use of FL allows for reducing the data that needs to be transferred from the receivers to the centralized server, enabling the creation of local VLF models which are then aggregated into a global model in federated fashion. In addition, due to the diversity of the participating parties' datasets, the global model can also extract knowledge of certain behaviours (e.g., tight maneuvering) that may not be available to all parties. Nevertheless, the FL problem in our setting is challenging since, apart from the inherent difficulty of forecasting, we also need to define the FL communication protocol (e.g., FedAvg [\[8\]](#page-8-7)), which is not a straightforward procedure at all.

In summary, the main contributions of this work are as follows:

- We propose an efficient VLF architecture based on LSTM, in two variants, namely Nautilus and FedNautilus, for the centralized and the (cross-silo) federated learning approach, respectively.
- We demonstrate the efficiency of the proposed architecture, in terms of prediction accuracy in short-term prediction horizon (up to 60 min.), using three largevolume real-world maritime AIS datasets.
- We study the effect of FL on the task at hand, i.e., the performance of FedNautilus with respect to FL aggregation hyper-parameters.

The rest of this paper is organized as follows: Section [II](#page-1-2) discusses related work. Section [III](#page-3-0) formulates the problem at hand and presents the proposed (Fed)Nautilus architecture in its two variants (centralized vs. federated learning). Section [IV](#page-4-0) presents our experimental study, where it is shown that our solution outperforms state-of-the-art. Finally, Section [V](#page-8-11) concludes the paper, also giving hints for future work.

II. RELATED WORK

A. Vessel Location Forecasting

Considering the VLF problem, current state-of-the-art includes an adequate number of research works. More specifically, one line of work includes clustering-based prediction techniques. Petrou et al. [\[12\]](#page-9-0) utilize the work done by [\[13\]](#page-9-1) on distributed subtrajectory clustering, in order to extract individual subtrajectory patterns from big mobility data; these

²MarineTraffic: Global Ship Tracking Intelligence.<www.marinetraffic.com>

patterns are subsequently utilized in order to predict the future location of the moving objects in parallel. In a more recent work, Zygouras et al. [\[14\]](#page-9-2) introduce EnvClus^{*}, a novel datadriven framework which performs trajectory forecasting via a mobility graph which models vessels' most likely movements among two ports.

Wang et al. [\[15\]](#page-9-3) 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. [\[16\]](#page-9-4) provided an NN-based approach for vessel trajectory prediction. In particular, they predict the vessels' future locations using a temporal window within an area of interest, and an encoder-decoder LSTM using the attention mechanism.

Suo et al. [\[17\]](#page-9-5) present an RNN-based model to predict vessel trajectories based on the DBSCAN [\[18\]](#page-9-6) clustering algorithm to derive main trajectories, and a symmetric segmentedpath distance approach to eliminate the influence of a large number of redundant data and optimize incoming trajectories.

Liu et al. [\[19\]](#page-9-7) propose "Spatio-Temporal GRU", a trajectory classifier for modeling spatio-temporal correlations and irregular temporal intervals prevalently presented in spatiotemporal trajectories. More specifically, a segmented convolutional weight mechanism was proposed to capture shortterm local spatial correlations in trajectories along with an additional temporal gate to control the information flow related to the temporal interval information.

Most recently, Chondrodima et al. [\[11\]](#page-8-10) propose a novel LSTM-based VLF framework, specially designed for handling vessel data by addressing some major GPS-related obstacles, such as variable sampling rate and sparse trajectories. Moreover, in order to improve the predictive power of VLF, they propose a novel trajectory data augmentation method based on the well-known Douglas-Peucker line simplification algorithm [\[20\]](#page-9-8). This work is considered, to the best of our knowledge, the state of the art in (short-term) VLF achieving an accuracy error around 2 km in 30 min. prediction horizon.

B. Federated Learning

While distributed ML [\[21\]](#page-9-9) can help us scale up the training process across multiple computational nodes, it can only be used on centralized data. On the other hand, FL trains centralized models using decentralized data [\[22\]](#page-9-10), as such, FL algorithms are primarily geared towards data privacy.

Considering the characteristics of the data owners, we distinguish two major FL variants, namely, cross-silo and cross-device FL [\[23\]](#page-9-11). 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 connection; a principal motivating example arises when the training data comes from users' interaction with mobile applications [\[24\]](#page-9-12). On the other hand, cross-silo FL can be considered when a relatively small group (usually 2–100) of organizations share a common incentive to collaboratively train an ML model based on their data, but cannot share them directly, due to

Fig. 2. 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 green) move towards $\frac{1}{N} \sum_i x_i^*$ instead of to the true optimum x^* (black square; [\[25\]](#page-9-13)).

e.g, storage cost or legal constraints. 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.

Except network and communication efficiency and client availability [\[9\]](#page-8-8), another key challenge of federated optimization is the training parties' heterogeneity with respect to their local datasets [\[24\]](#page-9-12). In order to address the first two issues, FedAvg [\[8\]](#page-8-7) performs multiple local updates on the available clients before communicating to the server. While 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](#page-2-0) illustrates. This phenomenon is known as "client-drift", and in order to avoid it, fewer local updates and/or lower learning rates must be used, actions that have a significant impact on the convergence stability of FedAvg.

Towards this direction, the authors in [\[25\]](#page-9-13) 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 considering 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 nonconvex experiments. Further following this line of research, FedProx [\[26\]](#page-9-14) presents an extension to FedAvg which adds a regularization term in the clients' cost function in order to restrict local updates to be closer to the initial (global) model. In a similar fashion, the qFedAvg algorithm [\[27\]](#page-9-15) introduces a novel optimization objective inspired by fair

Fig. 3. The proposed Nautilus architecture.

resource allocation in wireless networks that encourages a more uniform accuracy distribution across devices in federated networks. Closest to our work, however in the urban domain, the authors in [\[28\]](#page-9-16) propose ATPFL, a framework for predicting pedestrian trajectories, combining FL with Automated Machine Learning (AutoML). Additionally, FAHEFL [\[29\]](#page-9-17) is an FL-based algorithm geared towards user/model privacy (via Homomorphic Encryption) for vehicle trajectory prediction and behaviour classification.

To the best of our knowledge, our work is the first in the literature that experimentally evaluates centralized vs. federated VLF approaches over diverse (with respect to location and activity) maritime data silos, providing insightful results.

III. PROBLEM FORMULATION AND PROPOSED METHODOLOGY

In this section, we present the proposed Nautilus solution and describe it under both centralized and federated learning approaches.

A. Problem Definition

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

Definition 1. *(Trajectory). A trajectory* $T = \{p_1, \ldots, p_n\}$ *of a moving object is defined as a sequence of timestamped locations,* $p_i = \{x_i, y_i, t_i\}, 1 \le i \le n$ *.*

Definition 2. *(Vessel Location Forecasting). Given a dataset* D *of moving vessels' trajectories, a trajectory* T ^s *and the type* s_{true} *of vessel s, and a prediction horizon* Δt_{next} *, the goal is to train a data-driven model over* D*, which will be able to pre*dict the vessels' future location $p_{n+1}^s = \{x_{n+1}^s, y_{n+1}^s, t_{n+1}^s\}$ at timestamp $t_{n+1}^s = t_n^s + \Delta t_{next}$.

If we recall Figure [1,](#page-1-0) it provides an illustration of Definition [2.](#page-3-1) More specifically, we know the movement of three moving objects up to $t_n^{s_j}$. Our goal, given Δt_{next} , is to predict the anticipated location of these vessels at $t_n^{s_j} + \Delta t_{\text{next}}$.

B. The proposed Nautilus architecture

To address the VLF problem, we propose an extension of the LSTM-based model employed in [\[11\]](#page-8-10) for predicting the future location of moving vessels in the short-term, which considers their kinematic characteristics and their corresponding type; Figure [3](#page-3-2) illustrates the architecture of the proposed model. More specifically, Nautilus architecture consists of the following layers: a) an input layer of six neurons, one for each input variable, b) a single LSTM hidden layer composed of 350 neurons, c) an embedding layer with six dimensions for vectorizing the vessel's type, d) a Dropout layer with probability $p = 0.25$ for model regularization, e) a fullyconnected hidden layer composed of 150 neurons, and f) an output layer of two neurons, one for each output variable.

The input variables consist of the differences (i.e., deltas) in the Cartesian space ($\Delta x, \Delta y$), speed Δv and course $\Delta \phi$ over ground, current time Δt_{curr} , as well as the temporal horizon Δt_{next} for which we want to predict the vessels' future location. On the other hand, the output variables consist of the differences in space (Δx_{next} , Δy_{next}) of the predicted locations with respect to the current locations.

With respect to the employed LSTM, each cell includes three gates, namely, "forget", "input", and "output", which are responsible for what information we are going to drop from the cell state c_{t-1} , store from the cell state c_{t-1} , and output to the next cell state c_t , respectively. Eqs. [1–](#page-3-3)[6](#page-3-4) briefly state the update rules for the employed LSTM layer [\[30\]](#page-9-18). Additionally, details for the Back-Propagation Through Time (BPTT) algorithm, can be found in [\[31\]](#page-9-19).

$$
\mathbf{f}_t = \sigma \left(\mathbf{W}_f x_t + \mathbf{R}_f h_{t-1} + \mathbf{b}_f \right) \tag{1}
$$

$$
\mathbf{i}_t = \sigma \left(\mathbf{W}_i x_t + \mathbf{R}_i h_{t-1} + \mathbf{b}_i \right) \tag{2}
$$

$$
\mathbf{o}_t = \sigma \left(\mathbf{W}_o x_t + \mathbf{R}_o h_{t-1} + \mathbf{b}_o \right) \tag{3}
$$

$$
\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c x_t + \mathbf{R}_c h_{t-1} + \mathbf{b}_c\right) \tag{4}
$$

$$
\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{\tilde{c}}_{t-1} \tag{5}
$$

$$
\mathbf{h}_t = \mathbf{g}_t \odot \tanh(\mathbf{c}_t) \tag{6}
$$

TABLE I STATISTICS OF THE DATASETS USED IN OUR EXPERIMENTAL STUDY, AFTER THE PREPROCESSING PHASE (WITH TWO DIFFERENT TRAJECTORY SEGMENTATION THRESHOLDS, $t_max = 30$ MIN. AND 60 MIN., RESPECTIVELY).

	Spatial Range	Temporal Range	# Records	#Vessels (i.e., distinct IDs)	#Trajectories	Sampling Rate (sec.) (min; avg; max)	#Points per Trajectory (min; avg.; max.)
Brest	$lon \in [-9.71, -1.09]$	$2015 - 10 - 01$ –	4,408,217 (30 min.)	$1,654$ (30 min.)	14,418 (30 min.)	$10; 26; 1,800$ (30 min.)	20; 306; 13,638 (30 min.)
	$lat \in [45.00, 50.24]$	2016-03-31	4,504,533 (60 min.)	1,895 (60 min.)	13,155 (60 min.)	10; 36; 3,600 (60 min.)	20; 342; 15,557 (60 min.)
Piraeus	$lon \in [23.02, 23.80]$	$2019 - 01 - 01$	1,903,582 (30 min.)	1,321 (30 min.)	12,895 (30 min.)	$10; 35; 1,800$ (30 min.)	20; 148; 5,659 (30 min.)
	$lat \in [37.50, 38.04]$	2019-03-30	1,951,356 (60 min.)	1,324 (60 min.)	$10,588$ (60 min.)	10; 48; 3,600 (60 min.)	20; 184; 5,659 (60 min.)
Aegean	$lon \in [24.46, 26.59]$	$2018 - 11 - 01$ –	1,216,691 (30 min.)	$2,901$ (30 min.)	8,143 (30 min.)	11; 165; 1,800 (30 min.)	20; 149; 1,302 (30 min.)
	$lat \in [36.08, 39.49]$	2018-11-30	1,226,136 (60 min.)	$2,907(60 \text{ min.})$	7,741 (60 min.)	11; 169; 3,600 (60 min.)	20; 158; 1,600 (60 min.)

where h_t is the hidden state at time t, c_t (\tilde{c}_t) is the (intermediate) cell state at time t, x_t is the input at time t, h_{t-1} is the hidden state of the LSTM cell at time $t - 1$ or the initial hidden state at time 0, and \mathbf{i}_t , \mathbf{f}_t , \mathbf{g}_t , \mathbf{o}_t are the input, forget, cell, and output gates, respectively. W, R and b are the input weight matrices, the recurrent weight matrices and the bias terms, respectively. σ is the sigmoid function and \odot is the Hadamard product.

For the centralized training approach, we train our model using the Adam [\[32\]](#page-9-20) optimization algorithm with learning rate $\eta = 10^{-4}$ and the Root Mean Squared Error (RMSE) loss function for 100 epochs. To prevent over-fitting, we use the well-known early stopping [\[33\]](#page-9-21) mechanism with a patience of 10 epochs.

Our model extends [\[11\]](#page-8-10) by dividing its architecture into two parts based on the type of mobility information: internal and external. The former is related to the input of the recurrent layer, which encodes vessels' trajectories, whereas the latter enriches the projection of the vessels' input trajectory at a fully connected layer. In terms of internal information, we include additional factors related to vessels' movement, such as Δv and $\Delta\phi$. Furthermore, in terms of external information, we enrich the projection of the input trajectory with information related to the type of the vessel via a separate embedding layer that is jointly trained with the model.

C. Extending to FedNautilus

Figure [4](#page-5-0) illustrates the FL workflow of FedNautilus. In general, we have N clients, each of which trains a (local)FedNautilus instance with their own data (stored in separate data silos). Afterwards, they share the parameters (i.e., weights) of their local models to the aggregation server, which generates the (global)FedNautilus model and returns it to the participating clients. Moreover, in order to tailor (local)FedNautilus' decisions to the clients' needs, we fine-tune the (local)FedNautilus model for a – usually – small number of epochs, a technique known as Personalized Federated Learning (PerFL) [\[34\]](#page-9-22), which will be explained further in the following paragraphs.

In our FL environment, each data silo corresponds to the transmitted locations of a certain AIS-enabled fleet. Each silo contains an instance of our Nautilus model, which is trained using only their respective data. For training the local model

instances, we use the same optimization algorithm with the centralized approach, while for the optimization of the global model we use the FedProx [\[26\]](#page-9-14) algorithm for 70 FL rounds with $\mu_{prox} = 10^{-3}$, empirically selected based on the values listed at [\[26\]](#page-9-14) and overall training progress of the corresponding (local)Nautilus models.

Furthermore, to further address the client-drift issue, we exploit on Personalized Federated Learning (PerFL) [\[34\]](#page-9-22). PerFL is a branch of FL that aims to address these issues by customizing the global model for each client in the federation. In a nutshell, it aims to leverage the collective wisdom of clients' data in order to create models that are tailored to the data distributions of individual clients. In this paper, we follow the approach described in [\[35\]](#page-9-23), which is to fine-tune the global model for a certain number of epochs (10 in our case) using the clients' corresponding datasets, and extend it by using early stopping [\[33\]](#page-9-21) mechanism with a patience of 3 epochs, in order to avoid over-fitting.

IV. EXPERIMENTAL STUDY

In this section, we evaluate our VLF model on various centralized and federated learning schemes using three realworld maritime mobility datasets, and present our experimen-tal results^{[3](#page-4-1)}.

A. Experimental Setup, Datasets and Preprocessing

In our experimental study, we use three large-scale realworld maritime AIS datasets, referred to as "Brest"[4](#page-4-2) , "Piraeus"[5](#page-4-3) , and "Aegean"[6](#page-4-4) , respectively. Figure [5](#page-5-1) provides a visualization of these datasets on the map. All conducted experiments were implemented in Python. More specifically, the aforementioned models were implemented using PyTorch^{[7](#page-4-5)} and trained using Flower^{[8](#page-4-6)} for FL, via a GPU cluster equipped with 2 Nvidia A100 GPUs, 128 CPUs, and 1TB of RAM.

³The corresponding source code used in our experimental study is available at:<https://github.com/DataStories-UniPi/Nautilus>

⁴The dataset is publicly available at [https://doi.org/10.5281/zenodo.11675](https://doi.org/10.5281/zenodo.1167594) [94.](https://doi.org/10.5281/zenodo.1167594)

⁷PyTorch: An Imperative Style, High-Performance Deep Learning Library. <https://pytorch.org>

⁸Flower: A Friendly Federated Learning Framework.<https://flower.dev>

^{.&}lt;br>⁵The dataset is publicly available at [https://doi.org/10.5281/zenodo.55626](https://doi.org/10.5281/zenodo.5562629) [29.](https://doi.org/10.5281/zenodo.5562629)

⁶The (proprietary) dataset has been kindly provided by Kpler for research purposes, in the context of project MobiSpaces [\(https://mobispaces.eu\)](https://mobispaces.eu).

Fig. 4. Extending to FedNautilus - Federated Learning Workflow

Fig. 5. Snapshot of (a) Brest; (b) Piraeus; and (c) Aegean dataset, after the preprocessing phase.

Our data (pre-)processing stage consists of two phases. The first phase is a process consistently followed in the literature due to the noise, irregularity of sampling rate, etc. that are typical in AIS datasets. The second phase (data preparation for model training) on the other hand, uses the output of the first phase as its input and is performed for ML-specific purposes, i.e., to feed its output into our (Fed)Nautilus architecture.

In particular, the preprocessing phase includes (i) record deduplication, in which we eliminate duplicate AIS messages based on their unique ID (Maritime Mobile Service Identity - MMSI) and time of transmission, and (ii) dropping vessels with invalid ID with respect to the first three digits of MMSI that correspond to the country of origin (Maritime Identification Digits - MID) [\[16\]](#page-9-4). In order to focus our model on the "big picture" of maritime mobility, and avoid the effect of vessels' micromovements, as an effect of the unstable sampling rate of AIS^9 AIS^9 , we also subsample the vessels' trajectories, so that the minimum temporal difference between two points is set to Δt min = 10 seconds.

Because this operation may drop several – insignificant –

AIS messages, out of the resulting trajectories we drop the ones with less than $min_pts = 20$ points. Afterwards, we drop the AIS erroneous messages due to invalid speed (i.e., speed over speed $max = 50$ knots) or stationarity (i.e., speed less than speed_ $min = 1$ knot). Finally, we perform a trajectory segmentation task when a successive pair of points with a temporal difference higher than t max is detected (keeping, also in this case, the property of at least *min pts* points per trajectory). In our study, we experimented with two different thresholds, $t_{max} = 30$ min. and 60 min., in order to evaluate the prediction accuracy of our model up to $\Delta t_{\text{next}} = 30$ min. and $\Delta t_{\text{next}} = 60$ min., respectively. Table [I](#page-4-7) illustrates the statistics of the datasets (spatial and temporal range, number of locations, trajectories, etc.) after the above mentioned steps.

Towards training the model in order to account the trajectories' variable length as well as their temporal dependencies, we take the first order difference (slope) of the kinematic features (i.e., location, speed, course) for each trajectory, and then apply a semi-overlapping sliding window with length (i.e., number of transitions) from $length_min = 18$ up to $length_max = 32$ transitions. Since the aim is to predict the next location, we select as label the next transition from each window. Finally, to account for the unit difference in the time series variables, we normalize their corresponding values

⁹MarineTraffic - How often do the positions of the vessels get updated on MarineTraffic?, [https://help.marinetraffic.com/hc/en-us/articles/217631867](https://help.marinetraffic.com/hc/en-us/articles/217631867-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic-) [-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic-.](https://help.marinetraffic.com/hc/en-us/articles/217631867-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic-) Last visited at Jan. 20, 2024.

by subtracting the mean of each feature and dividing by its variance (z-score standardization).

In the two sections that follow, we provide the experimental results of our study, comparing the two variants, centralized and federated, of the proposed (Fed)Nautilus framework presented in Section [III,](#page-3-0) also with respect to the current state-of-the-art [\[11\]](#page-8-10). The performance comparison is based on a popular prediction quality measure, namely the Final Displacement Error (FDE) defined in Eq. [7.](#page-6-0)

$$
FDE = \frac{1}{n} \sum_{i} \sqrt{\frac{\left(\Delta x_i^{pred} - \Delta x_i^{true}\right)^2 + \left(\Delta y_i^{pred} - \Delta y_i^{true}\right)^2}{\left(\Delta y_i^{pred} - \Delta y_i^{true}\right)^2}}
$$
(7)

B. Experimental Results on Nautilus

In this section, we demonstrate the results of our experimental study on Nautilus. Following the preprocessing outlined in Section [IV-A,](#page-4-8) we acquire 265,525, 109,906, and 70,201 sliding windows from the Brest, Piraeus, and Aegean datasets, respectively. These windows will be used to train the model. They are divided into training, validation, and test sets using a split ratio of 50:25:25% (1 fold) with respect to the datasets' temporal span (c.f., Table [I\)](#page-4-7). After training our VLF instances, Table [II](#page-7-0) illustrates the performance of Nautilus on the datasets' test set with respect to FDE. We observe that our solution outperforms the state-of-the-art method [\[11\]](#page-8-10) in the majority of cases; in particular, it is the clear winner on the Brest and Piraeus datasets, especially when both models are trained over the $t_{max} = 60$ min. trajectory segmentation variant of datasets, while it appears to be a balance between the two models on the Aegean dataset.

Focusing on the results of Nautilus, and comparing the two variants (i.e., 30 min. vs. 60 min. trajectory segmentation threshold), we observe that the 30 min. variant yields slightly better results on the Brest dataset, while the opposite happens on the Piraeus and Aegean datasets. A first conclusion out of this is that the segmentation threshold is not a critical factor for the quality of the prediction; the original assumption that by setting t ma $x = 30$ min. at the preprocessing phase would lead to consistently better performance for predictions up to $\Delta t_{\text{next}} = 30$ min. was not confirmed in practice. This behaviour can partially be attributed to the increase in the population of the train set, in terms of target lookaheads, which may introduce a regularization effect to the model.

C. Experimental Results on FedNautilus

Moving to the FL setting, let us assume that the participating partners are in agreement to exchange the parameters of each partner's VLF model (i.e., local model), and aggregate them into a single entity (i.e., global model) using the FedProx algorithm [\[26\]](#page-9-14) with $\mu_{prox} = 10^{-3}$, as presented in Section [III-B.](#page-3-5) After 70 rounds, Table [III](#page-7-1) (rows titled "FedNautilus") illustrates the performance of the (global) FedNautilus model over the test set of the datasets at hand. Compared to the prediction error of our centralized solution (c.f. Table [II\)](#page-7-0), we

Fig. 6. Learning curve for (a) Brest, (b) Piraeus, and (c) Aegean local FedNautilus instances compared to the global FedNautilus model. Solid blue and orange lines correspond to participants' training and validation loss, respectively, while dashed green and red lines correspond to the global FedNautilus training and validation loss, respectively.

observe that the prediction quality on all three datasets of the global FedNautilus model has been decreased with respect to the centralized results.

In order to further understand the reason behind this behaviour, Figure [6](#page-6-1) illustrates the learning curve of the partners' local VLF instances, compared to the learning curve of the global FedNautilus model. In general, we observe that the training/validation loss of the Aegean (c.f., Figure [6c\)](#page-6-2) local models diverges from the global FedNautilus model, a behaviour which is observed throughout the training process, better known as "client-drift".

The main cause behind "client-drift" lies within the participating parties' heterogeneity. In particular, Figure [7](#page-8-12) illustrates the Kernel Density Estimation (KDE) of the first and second principal components of the three datasets. We observe that the estimated distributions of Brest and Piraeus datasets follow a - seemingly - unimodal distribution, while on the other hand, the multimodal distribution of the Aegean dataset, introduces a high level of heterogeneity, which inhibits the training process of global FedNautilus model from matching the participants' performance target(s).

By adjusting the μ_{prox} parameter, we can be either more "strict" or more "relaxed" with participating data silos whose contributions (i.e., model weights) greatly deviate with respect to the aggregated (global) model. Table [IV](#page-9-24) illustrates the performance of (global)FedNautilus for μ_{prox} =

TABLE II PREDICTION ERROR (FDE; METERS) OF NAUTILUS VS. STATE-OF-THE-ART PER DATASET (LESS IS BETTER).

		(0,5]	(5, 10]	(10, 15]	(15, 20)	(20, 25)	(25, 30)	(30, 35)	(35, 40)	(40, 45]	(45, 50)	(50, 55]	(55, 60]
Brest	VLF-LSTM [11] (30 min.) Nautilus (30 min.)	13 12	308 302	602 579	802 811	1073 1139	1316 1220	N/A N/A	N/A N/A	N/A N/A	N/A N/A	N/A N/A	N/A N/A
	VLF-LSTM [11] (60 min.) Nautilus (60 min.)	14 13	366 353	781 714	1042 893	1210 1032	1427 1380	2926 2654	2493 2296	2699 2119	1987 1897	3991 3627	4167 4177
Piraeus	VLF-LSTM [11] (30 min.) Nautilus (30 min.)	14 14	157 154	348 353	438 442	713 685	551 601	N/A N/A	N/A N/A	N/A N/A	N/A N/A	N/A N/A	N/A N/A
	VLF-LSTM [11] (60 min.) Nautilus (60 min.)	15 15	155 144	332 334	436 409	570 537	406 332	672 639	478 353	871 723	1458 1369	1518 1346	2620 2179
egean	VLF-LSTM [11] (30 min.) Nautilus (30 min.)	41 43	201 198	477 494	505 501	1036 902	1764 1976	N/A N/A	N/A N/A	N/A N/A	N/A N/A	N/A N/A	N/A N/A
	VLF-LSTM [11] (60 min.) Nautilus (60 min.)	43 45	190 188	499 482	348 457	1010 1053	1302 1132	967 918	2360 2684	1460 1373	2950 2860	8369 9402	5124 2848

TABLE III

PREDICTION ERROR (FDE; METERS) OF NAUTILUS VS. FEDNAUTILUS PER DATASET (LESS IS BETTER).

 $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}.$

Combined with the insight given by Figure [7,](#page-8-12) we observe that the Piraeus dataset has the most increase in performance, presenting its best at $\mu_{prox} = 1.0$, with $\mu_{prox} = 10^{-3}$ being a close contender. On the other hand, we observe that the Brest and Aegean datasets present an overall increase in FDE up to 25 min., presenting their best at $\mu_{prox} = 10^{-1}$ and $\mu_{prox} = 10^{-3}$, respectively. In other words, the more we increase μ_{prox} , the more FedProx tends to benefit the Piraeus dataset, and ignore the Brest and Aegean datasets, due to their increasing complexity in terms of KDE.

Focusing on the FedNautilus instances trained with μ_{prox} = 10[−]³ (more or less, the same observations hold the other values of μ_{prox} used in our experimental study), Table [III](#page-7-1) (rows titled "(Per)FedNautilus") illustrates the performance of the global personalized FedNautilus instance, (Per)FedNautilus, on the three datasets. Compared to the centralized and the "global" federated approaches (rows titled "Nautilus" and "(global)FedNautilus", respectively, in Table [III\)](#page-7-1), it appears that personalization clearly improved the prediction accuracy of our model, not only against the global FL model but also against the (local) models trained upon the partners' datasets. This is a key result of our performance study that confirms the value of the FL paradigm.

D. A note on the communication cost

As mentioned in Section [I,](#page-0-1) FL offers significant improvements in communication costs compared to centralized ML,

Fig. 7. Kernel Density Estimation (KDE) of $1st$ and $2nd$ principal components of (a) Brest; (b) Piraeus; and (c) Aegean dataset.

especially when dealing with large datasets. Assume a hypothetical scenario in which, rather than training an ML-based model in a centralized (isolated) manner, each data silo pools their own datasets and trains a centralized (unified) ML-based model. The combined size of the Brest, Piraeus, and Aegean datasets is up to 4.15 GB. The server sends the updated model (6.7 MB, of which 2.2 MB is the size of the model parameters) to participating data silos at each epoch, for a total of 6.7 * 3 $*$ 70 = 1.37 GB for 70 epochs in the worst-case scenario (i.e., the early stopping mechanism is not activated). As a result, the communication cost of the centralized training process is 5.52 GB.

On the other hand, FL trains the model locally on the devices or servers where the data is stored. In this case, only the updated model parameters, 2.2 MB in total, must be sent from each data silo to the central server and vice versa. As a result, the total communication cost for each training round in FL is 6.6 MB (3 silos * 2.2 MB) * 2 (send/receive), or 0.90 GB for 70 FL rounds. This represents a significant reduction in communication cost of 84% compared to the centralized approach. As a result, FedNautilus can be considered a viable solution for not only data silos but also edge devices with limited bandwidth resources.

V. CONCLUSION

In this paper, we proposed (Fed)Nautilus, a VLF framework extending current state-of-the-art [\[11\]](#page-8-10), we trained it in two variants, following the centralized ML and the FL paradigms, respectively, and assessed the pros and cons of each approach. In particular, through an extensive experimental study on three real-world AIS datasets, we demonstrated the efficiency of Nautilus in comparison with related work. We also evaluated the benefits and open problems of the FL-based solution, FedNautilus, as well as the usefulness of personalization over heterogeneous datasets.

Following the research guidelines of the emerging Mobility Data Science era [\[36\]](#page-9-25), in the near future, we aim to further adjust the architecture of the FedNautilus model by incorporating additional external factors, such as weather conditions and itinerary information. In a parallel line of research, we aim to transfer the FedNautilus model from the cross-silo to the cross-device paradigm in order to assess the balance between prediction quality and communication cost, based on the preliminary findings about the latter, which were discussed in Section [IV-D.](#page-7-2) In the long-term, we aim to address the clientdrift issue of FedNautilus by either fine-tuning the existing algorithms, or implementing newer ones, such as [\[37\]](#page-9-26).

ACKNOWLEDGMENT

This work was supported in part by the Horizon Framework Programme of the European Union under grant agreement No. 101070279 (MobiSpaces; [https://mobispaces.eu\)](https://mobispaces.eu). In this work, Kpler provided the Aegean AIS dataset and the requirements of the business case.

REFERENCES

- [1] P. Tampakis, E. Chondrodima, A. Tritsarolis *et al.*, "i4sea: a big data platform for sea area monitoring and analysis of fishing vessels activity,' *Geo-spatial Information Science*, vol. 25, no. 2, pp. 132–154, 2022.
- [2] P. Mandalis, E. Chondrodima, Y. Kontoulis *et al.*, "Machine learning models for vessel traffic flow forecasting: An experimental comparison, in *Procedings of the 23rd IEEE International Conference on Mobile Data Management (MDM)*, 2022.
- [3] A. Tritsarolis, E. Chondrodima, N. Pelekis *et al.*, "Vessel collision risk assessment using AIS data: A machine learning approach," in *Proceedings of the 23rd IEEE International Conference on Mobile Data Management (MDM)*, 2022.
- [4] A. Tritsarolis, E. Chondrodima, P. Tampakis *et al.*, "Predicting comovement patterns in mobility data," *GeoInformatica*, 2022.
- [5] Q. Yang, Y. Liu, Y. Cheng *et al.*, *Federated Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019.
- [6] D. Zissis, K. Chatzikokolakis, G. Spiliopoulos *et al.*, "A distributed spatial method for modeling maritime routes," *IEEE Access*, vol. 8, pp. 47 556–47 568, 2020.
- [7] G. Spiliopoulos, K. Chatzikokolakis, D. Zissis *et al.*, "Knowledge extraction from maritime spatiotemporal data: An evaluation of clustering algorithms on big data," in *Proceedings of IEEE International Conference on Big Data*, 2017.
- [8] H. B. McMahan, E. Moore, D. Ramage *et al.*, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.
- [9] J. Konečný, H. B. McMahan, F. X. Yu et al., "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.
- [10] X. Qiu, T. Parcollet, J. Fernández-Marqués et al., "A first look into the carbon footprint of federated learning," *CoRR*, vol. abs/2102.07627, 2021.
- [11] E. Chondrodima, N. Pelekis, A. Pikrakis *et al.*, "An efficient LSTM neural network-based framework for vessel location forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 4872–4888, 2023.

TABLE IV PREDICTION ERROR (FDE; METERS) OF (GLOBAL)FEDNAUTILUS ACROSS DIFFERENT VALUES OF μ_{prox} (LESS IS BETTER).

		(0,5]	(5, 10]	(10, 15]	(15, 20)	(20, 25)	(25, 30)	(30, 35]	(35, 40]	(40, 45]	(45, 50)	(50, 55)	(55, 60)
Brest	$\mu_{prox} = 0.0001$	17	393	754	1079	1168	1574	2755	2524	2307	2146	3549	4289
	$\mu_{prox} = 0.001$	17	375	729	1033	1058	1579	2715	2480	2285	2222	3296	4308
	$\mu_{prox} = 0.01$	17	383	787	1059	1083	1584	2741	2430	2309	2206	3464	4187
	$\mu_{prox} = 0.1$	17	376	728	1035	1037	1556	2682	2437	2156	2167	3381	4206
	$\mu_{prox} = 1.0$	17	391	800	1070	1153	1660	2831	2502	2329	2556	3401	4216
Piraeus	$\mu_{prox} = 0.0001$	25	258	390	560	666	676	812	844	1220	1634	1547	2766
	$\mu_{prox} = 0.001$	24	239	387	534	654	675	867	772	1238	1555	1708	2767
	$\mu_{prox} = 0.01$	23	236	386	529	653	586	777	718	1140	1678	1503	2395
	$\mu_{prox} = 0.1$	24	249	405	552	661	593	760	770	1184	1614	1584	2531
	$\mu_{prox} = 1.0$	24	234	365	504	626	583	791	656	1024	1568	1488	2592
Aegean	$\mu_{prox} = 0.0001$	561	1532	3519	3595	3680	8144	8304	6039	4390	7654	19453	30318
	$\mu_{prox} = 0.001$	528	1334	2976	3293	3109	8808	9400	4296	3236	7585	18786	21302
	$\mu_{prox} = 0.01$	582	1440	3005	3555	3267	8158	9314	5295	4582	5997	16759	32592
	$\mu_{prox} = 0.1$	567	1491	3368	3743	4303	8483	9679	5815	3780	8207	19034	34912
	$\mu_{prox} = 1.0$	552	1425	3123	3311	3416	7924	8562	5097	4205	4870	15212	27838

- [12] P. Petrou, P. Nikitopoulos, P. Tampakis *et al.*, "ARGO: A big data framework for online trajectory prediction," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases (SSTD)*, 2019.
- [13] P. Tampakis, N. Pelekis, C. Doulkeridis *et al.*, "Scalable distributed subtrajectory clustering," in *Proceedings of the IEEE International Conference on Big Data*, 2019.
- [14] N. Zygouras, A. Troupiotis-Kapeliaris, and D. Zissis, "Envclus*: Extracting common pathways for effective vessel trajectory forecasting," *IEEE Access*, vol. 12, pp. 3860–3873, 2024.
- [15] S. Wang, J. Cao, and P. S. Yu, "Deep learning for spatio-temporal data mining: A survey," *CoRR*, vol. abs/1906.04928, 2019.
- [16] S. Capobianco, L. M. Millefiori, N. Forti *et al.*, "Deep learning methods for vessel trajectory prediction based on recurrent neural networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 4329–4346, 2021.
- [17] Y. Suo, W. Chen, C. Claramunt *et al.*, "A ship trajectory prediction framework based on a recurrent neural network," *Sensors*, vol. 20, no. 18, 2020.
- [18] M. Ester, H. Kriegel, J. Sander *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.
- [19] H. Liu, H. Wu, W. Sun *et al.*, "Spatio-temporal GRU for trajectory classification," in *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2019.
- [20] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [21] T. Liu, W. Chen, and T. Wang, "Distributed machine learning: Foundations, trends, and practices," in *Proceedings of the 26th International Conference on World Wide Web Companion (WWW)*, 2017.
- [22] B. McMahan, E. Moore, D. Ramage *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS*, 2017.
- [23] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021.
- [24] J. Konečný, H. B. McMahan, D. Ramage et al., "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016.
- [25] S. P. Karimireddy, S. Kale, M. Mohri *et al.*, "SCAFFOLD: stochastic controlled averaging for federated learning," in *Proceedings of the*

37th International Conference on Machine Learning (ICML), vol. 119. PMLR, 2020, pp. 5132–5143.

- [26] T. Li, A. K. Sahu, M. Zaheer *et al.*, "Federated optimization in heterogeneous networks," in *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys)*, 2020.
- [27] T. Li, M. Sanjabi, A. Beirami *et al.*, "Fair resource allocation in federated learning," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.
- [28] C. Wang, X. Chen, J. Wang et al., "ATPFL: automatic trajectory prediction model design under federated learning framework," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [29] M. Han, K. Xu, S. Ma et al., "Federated learning-based trajectory prediction model with privacy preserving for intelligent vehicle," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 10 861–10 879, 2022.
- [30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [31] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, 2015.
- [33] L. Prechelt, "Early stopping - but when?" in *Neural Networks: Tricks of the Trade (2nd ed.)*, ser. Lecture Notes in Computer Science. Springer, 2012, vol. 7700, pp. 53–67.
- [34] A. Z. Tan, H. Yu, L. Cui *et al.*, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9587–9603, 2023.
- [35] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *Proceedings of the International Conference on Communications (ICC)*, 2020.
- [36] M. Mokbel, M. Sakr, L. Xiong *et al.*, "Mobility data science: Perspectives and challenges," *ACM Transactions on Spatial Algorithms and Systems*, 2024.
- [37] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.