FOGFLEET: Fog-Level Federated Transfer Learning for Adaptive Transport Mode Detection

Mahdieh Kamalian
University of Oslo
Oslo, Norway
mahdiehk@ifi.uio.no

Amir Taherkordi
University of Oslo
Oslo, Norway
amirhost@ifi.uio.no

Amir H. Payberah
KTH Royal Institute of Technology
Stockholm, Sweden
payberah@kth.se

Paulo Ferreira
University of Oslo
Oslo, Norway
paulofe@ifi.uio.no

***Abstract*—Transport Mode Detection (TMD) systems play a pivotal role in facilitating applications in transport, urban planning, and more. Exploiting the advancements in smartphone sensing capabilities, TMD systems have evolved for mobile applications with local classification on smartphones as a common approach. Yet, local approaches relying on centralized training raise privacy concerns due to the transmission of sensitive data (e.g., GPS logs) over the Internet. In this paper, we propose FOGFLEET, a novel Federated Transfer Learning (FTL) framework for TMD, addressing both privacy and performance concerns. Our approach relies on Federated Learning (FL) to train a global model on various datasets from different cities while employing transfer learning to adapt the global model to the specific characteristics of individual smartphones and cities. FOGFLEET relies on an architecture that integrates edge, fog, and cloud layers, with dedicated fog nodes for each city to simplify cross-silo federated learning. Experimental results demonstrate the effectiveness of the FOGFLEET framework in higher TMD accuracy by up to 20% than its comparable centralized approach. Furthermore, it outperforms the FL solutions reported in the literature with at least an 8% increase in accuracy. In this work, we also highlight the importance of sufficient training data for distributed training and discuss the impact of smartphone sensor qualities on the performance of TMD systems. Our work contributes to advancing TMD systems by providing an adaptive and privacy-preserving solution suitable for deployment in diverse urban environments and across various geographical locations.**

***Index Terms*—Transport Mode Detection, Federated Learning, Transfer Learning, Privacy, Accuracy, Fog Computing, Edge Computing**

I. INTRODUCTION

Transport Mode Detection (TMD) refers to systems capable of distinguishing between different modes of transportation, such as cars, buses, trains, etc. The knowledge obtained from TMD is central for many applications such as urban planning, transport, navigation, carbon emissions, air pollution, health, and entertainment [24]. The advancement of smartphones made them a practical platform for TMD [6], [10], [16]. Several researchers explored the smartphone's sensor data fitted to different Machine Learning (ML) algorithms for TMD [5], [19]. Such a TMD system follows a two-stage process including training and inference stages. In the training stage, an ML model (i.e., classifier) is trained based on the ground truth collected from the smartphone's sensors. In the inference stage, the trained model is integrated into a smartphone application for executing prediction requests.

The training stage can be performed using either a *centralized* or *distributed* approach. In a centralized approach, all the data required for training is gathered and processed by a central entity, such as a powerful server on the Cloud. The central entity has access to the entire dataset and performs training using the complete set of information. On the other hand, in a distributed approach, the dataset is divided into smaller subsets, and these subsets are distributed across multiple edge devices, like smartphones. Each device independently trains the model using only the subset of data it holds. The distributed training depends on the computational power of multiple devices, with each device having access to only a portion of the dataset.

Concerning the main goal for centralized or distributed training, centralized training tends to provide higher accuracy than the distributed approach as it can utilize the entire dataset, allowing the model to capture more comprehensive patterns and variations present in the data. However, the centralized approach raises privacy concerns, as it involves the storage and transmission of sensitive data such as GPS logs to a central entity. In contrast, distributed training approaches, such as Federated Learning (FL), prioritize privacy by keeping the data decentralized on edge devices [15]. While FL offers privacy advantages, it may suffer from lower accuracy compared to the centralized approach [7], [9]. Thus, there is a trade-off between accuracy and privacy when choosing between centralized and distributed training such as FL for TMD systems.

Many TMD systems based on FL mainly employ smartphones as FL clients, a setup commonly referred to as cross-device FL [8], [9], [12]. However, the non-IID (non-Independently and Identically Distributed) nature of smartphone data, characterized by limited variation across devices, poses a significant challenge for achieving optimal performance [9]. The heterogeneity in smartphones, containing differences in sensor quality, communication, and computational capabilities, further complicates the FL process. This challenge is worsened when only a few of these devices participate at a given time, potentially leading to biased training [14].

Beyond that, there are several differences in the public transport modes and their speed, acceleration pattern, road conditions, etc., in various locations (i.e., cities). Thus, the model trained based on data in one location can not generalize well in another location with various types of transport modes

or road conditions, etc. Many state-of-the-art solutions lack such generalization [13].

To overcome these challenges we propose a cross-silo Federated Transfer Learning (FTL) framework called FOGFLEET. In FOGFLEET, data is partitioned across different data silos, such as fog nodes deployed in various geographical locations. The core idea behind FOGFLEET is to create first a *global* FL model trained across multiple cities, each exhibiting unique transportation patterns and road conditions (i.e., the federated learning phase), then *fine-tune* the global model with data from individual smartphones in a new city (i.e., the transfer learning phase). Regarding the former phase, a fog node is introduced as the FL client per city, and data from multiple smartphones in the city is merged, thereby fog nodes help mitigate the non-IID nature of the data, leading to more robust and generalized models for TMD. The fine-tuning process adapts the global model to the specific characteristics of each smartphone's dataset, accounting for variations in sensor data quality, etc. By initializing a base model with pre-trained weights from the global model and selectively freezing certain layers, knowledge transfers from the global model to personalized models thus enabling efficient *adaptation* to the new city and smartphone data. Last but not least, FOGFLEET obtains an accuracy comparable to centralized approaches while leveraging fog nodes as the FL clients preserve the user's privacy city- or municipality-wide.

To summarise, the contributions of this work are as follows:

1) *Federated Transfer Learning (FTL) framework*: We propose a novel framework that integrates federated learning and transfer learning techniques to address the challenges of TMD across diverse locations. Furthermore, by combining federated learning with transfer learning, our framework offers an accurate and privacy-preserving TMD system for model training and adaptation.
2) *Personalized model adaptation*: Through the fine-tuning process using the TL technique, our solution enables the domain adaptation of a global model to individual smartphone datasets within a new city. By accounting for variations in sensor data quality and transport patterns, our approach produces personalized models customized to each smartphone, enhancing model performance.
3) *Evaluation of the FTL scenario versus a centralized scenario*: We compare our FTL solution with a centralized approach. In the centralized scenario, we apply centralized training with the dataset of each smartphone collected in three different cities. The experimental results suggest that despite distributed training, the FTL solution can achieve 1% to 20% higher accuracy than the centralized approach.

The rest of this article is structured as follows. Section II presents some related work with a comparison of our solution with both centralized and distributed state-of-the-art solutions. Section III details our solution, the underlying architecture of our solution, and the proposed classifier structure. In Section IV, we begin by describing the datasets used, the preprocessing methods applied, and the features selected. We then present the results of our experimental evaluation. Finally, Section V presents the conclusion.

II. RELATED WORK

Over the past few decades, TMD systems have evolved to facilitate different applications in various domains such as transport, healthcare, economy, energy, and entertainment [24].

The advancements in sensing computation, and storage capabilities of smartphones have provided a great opportunity to enhance TMD as mobile systems. Therefore, there has been a dominant trend toward utilizing smartphones for both data collection and inference in TMD systems. Within this context, local TMD approaches have emerged, where inference is performed directly on smartphones following a training stage [13].

Most previous local TMD systems employed a centralized ML or DL classifiers for training their system and then integrated the generated model for a local classification on the smartphone [3], [10], [11], [17]. In this setup, a model is trained on a central entity utilizing data gathered from diverse smartphone sensors such as accelerometer, and GPS log [13]. Accelerometer and GPS are the most common and informative sensors for TMD being able to capture the acceleration and speed patterns in transport vehicles [10], [19]. Accelerometers, in combination with magnetometers, introduce more accuracy in detection [3], [5], [23].

The centralized solutions despite their better performance give rise to notable privacy concerns, as it necessitates the transmission of sensitive user data, especially GPS over the Internet and storing it on a centralized entity for training.

For instance, Ferreira et al. [10] propose an ML solution for TMD named EdgeTrans, which primarily utilizes ML classifiers like Decision Tree (DT) and Random Forest (RF), choosing RF due to its superior accuracy. EdgeTrans relies on accelerometer and GPS data and employs a centralized cloud server for data storage and model training. However, the centralized nature of EdgeTrans raises significant privacy concerns. Moreover, the model achieves an average F1 score of 90% in detecting only five modes: car, bus, train, walk, and bicycle. Notably, the classifier struggles with a limited number of motorized modes and often confuses the car modality with the bus modality.

In another work, Wang et al. [11] introduce a deep learning classifier that employs a dual-stage Long Short-Term Memory (LSTM) network with an ensemble decision module for post-processing. This system includes a binary classifier for detecting elevator mode and a detailed classifier for distinguishing between bus, subway, High-Speed Railway (HSR), and non-motorized modes such as stationary and walking. The classifier achieved overall accuracies of 91.28% and 88.59% for stationary and walking modalities. However, the need for an ensemble decision module to smooth results in the time domain suggests initial inaccuracies or instability in classification outputs. Additionally, the system's complex structure could pose challenges during inference on smartphones.

In a comprehensive work, Wang et al. [3] provide an analysis of 15 various solutions using ML and DL classifiers on the Sussex-Huawei Locomotion-Transportation (SHL) dataset [19]. Their standout solution combines an RF and a fully connected Deep Neural Network (FC-DNN), achieving an average F1 score of 96% across 106 features. The study focuses on detecting transportation modes such as still, walk, run, bike, bus, car, train, and subway using data from accelerometers, gyroscopes, magnetometers, and GPS. Despite its success, the model faces significant challenges related to computational efficiency due to its complexity and extensive feature set. This could hinder deployment on standard smartphones, which typically have limited processing power and battery life. The research further encounters limitations due to the exclusive use of one type of Android-based smartphone (HUAWEI Mate 9), compounded by reported technical issues like sensor unavailability, irregular sampling rates, and sensor diversity, which may affect data quality and model robustness.

Due to the drawbacks of centralized training approaches including privacy, security, and the need for persistent Internet access between the smartphone and the centralized entity, some decentralized training, particularly FL approaches recently have been introduced [8], [9].

Mensah et al. [4] introduce an ensemble-based Federated Deep Neural Network (eFedDNN) utilizing FL. This classifier combines DL classifiers like LSTM, GRU, and 1D CNN as base learners and MLP as a meta-learner, enhancing TMD from GPS data without compromising user privacy. By employing an ensemble method, which integrates various local models trained on decentralized devices, the eFedDNN presents 84.1% overall accuracy in detecting transport modes of walking, biking, car, and public transit. The complexity of stacking these models limits the efficiency of execution on smartphones. Moreover, grouping all public transport modes into a single category and not presenting performance results for each mode separately restricts the classifier's performance and efficiency. The need for an ensemble decision module to refine the classification results further indicates that the initial outputs may not be sufficiently accurate or stable.

Yu et al., [8] present an FL framework with a maximum of 8 clients using a Convolutional Neural Network (CNN) model that obtained an overall accuracy of 67.52%. The main limitation of this work is achieving such a low accuracy compared to the centralized and local TMD works presented above. Furthermore, the structure of their model is too deep for a classification task of TMD, causing low accuracy (i.e., 67.13%) even when training the model with a centralized approach.

Cavalacante et al., [9], also present an FL framework using a Neural Network (NN) with an average accuracy of 80.6% for detecting only three transport modes including car, bus, and motorcycle. The limited number of modes restricts this system's efficiency and categorizes it as a non-fine-grained TMD system. The assessment outcomes outlined in this study indicate a decline in the performance of the distributed training approach relative to the centralized approach. The authors attribute this issue to an anticipated consequence of information loss during the aggregation of client models.

Huang et al., [2] present a Federated Learning-based Transport Mode Inference model with Privacy-Preserving Data Fusion (PPDF-FedTMI) that enhances privacy in TMD using GPS data to detect five modes of walk, bike, bus, car, and train. The model utilizes a fuzzy fusion algorithm and local differential privacy to safeguard user data and enhance performance on non-IID distributed datasets, achieving up to an 84% F1 score. However, the incorporation of complex techniques like fuzzy clustering and local differential privacy increases computational demands, potentially affecting efficiency with large-scale deployment. Additionally, the model's assumptions for handling non-IID data may not be universally applicable, limiting effectiveness in varied environments. Moreover, data collection from only Beijing may restrict the model's generalization.

Table I highlights the advantages of FOGFLEET by comparing it against both centralized and distributed state-of-the-art solutions. As demonstrated in the table, our FTL approach obtains an accuracy comparable to centralized approaches and surpasses other distributed approaches in performance. Furthermore, FOGFLEET offers superior privacy benefits over centralized approaches. This effectively addresses the trade-off between performance and privacy.

III. Architecture and Implementation

In this section, we provide an overview of our solution, FOGFLEET, and its architecture. As shown in Algorithm 1, our solution comprises two phases: FL and TL. For implementing FL and TL phases we rely on an architecture depicted in Fig. 1. According to this figure, our proposed architecture consists of three layers of edge, fog, and cloud. As shown in Fig. 1, the fog nodes are in Oslo, Rome, Stockholm, Tehran, and London. This setup results in a total of five FL clients (i.e., fog nodes), with the option to exclude one client from the FL process for experimentation purposes. The excluded client (i.e., the city's dataset) is used for the TL process (see Section IV).

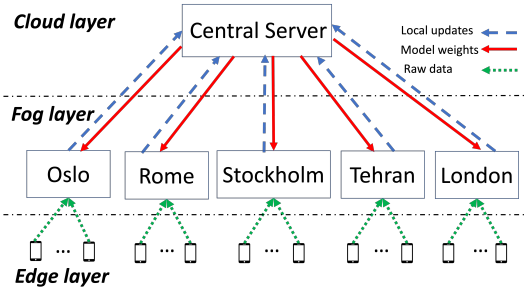


Fig. 1: FOGFLEET *architecture*

In the *Edge layer*, smartphones denoted as $S = \{s_1, ..., s_n\}$ serve as data collection devices through an Android-based application referred to as TMDetector [5]. In the *Fog layer*, each city is equipped with a dedicated fog node, which serves

TABLE I: Comparison of FOGFLEET with the State-of-the-Art (centralized or distributed) TMD Solutions

Study	Methodology	Privacy Concerns	Performance (Overall F1 Score or Accuracy)
Ferreira et al. [10]	Centralized ML with RF	High	90% (F1 score)
Wang et al. [11]	Centralized DL with LSTM	High	91% (accuracy)
Wang et al. [3]	Centralized ML+DL with RF+DNN	High	96% (F1 score)
Mensah et al. [4]	FL with LSTM+GRU+CNN+MLP	Low	84% (accuracy)
Yu et al. [8]	FL with CNN	Low	68% (accuracy)
Cavalcante et al. [9]	FL with NN	Low	81% (accuracy)
Huang et al., [2]	FL with LSTM	Low	84% (F1 score)
FOGFLEET	**FTL with MLP**	**Low**	**92%-97% (F1 score)[1]**

Note: ML=Machine Learning; DL=Deep Learning; FL= Federated Learning; RF=Random Forest; LSTM= Long-Short Term Memory; DNN=Deep Neural Network; GRU=Gated Recurrent Units; CNN= Convolutional Neural Network; MLP=Multilayer Perceptron; FTL=Federated Transfer Learning

[1] The overall F1 scores range from 92% to 97% across various datasets: 96.75% for Tehran, 94.8% for Rome, and 91.8% for Stockholm. Detailed evaluations can be found in Section IV.

Algorithm 1: Pseudocode for federated learning and transfer learning phases (in the cloud server and fog nodes)

```
initialize_global_model()
for round = 1 to num_rounds do
    selected_clients = select_clients(F)
    foreach client f in selected_clients do
        local_model = received_global_model(client f)
        local_model = train_local_model(client f,
          local_model, D_f[f])
        update = compute_model_update(local_model)
        send_update_to_server(update)
    end
    global_model = aggregate_updates()
    evaluate_model(global_model)
end
terminate_fl_process()
for excluded city c in C do
    for each smartphone s in S do
        global_model = load_global_model_weights()
        personalized_model =
          transfer_learning(global_model,
          smartphone_dataset, D_s[s])
    end
end
terminate_tl_process()
phase1_global_model = federated_learning()
phase2_personalized_model =
  transfer_learning(phase1_global_model,
  new_city_dataset, smartphone_dataset)
```

as both a repository for local data storage and a processing node for FL and TL tasks. Therefore, for the given set of cities, $C = \{c_1, ..., c_n\}$, each city is assigned a dedicated fog server $F = \{f_1, ..., f_n\}$ serving as FL clients.

Under the framework of cross-silo FL presented in this work, the fog nodes offer several benefits over relying solely on smartphones for FL tasks: (i) fog nodes enhance FL by mitigating the non-IID data challenge by merging data from multiple smartphones; (ii) fog nodes create diverse and representative datasets that improve model generalization and reduce data irregularities; (iii) with fog nodes' superior computational capabilities compared to smartphones, fog nodes can perform advanced data processing, ensuring uniform training across devices; (iv) this approach minimizes the effects of smartphone variability. Thus, overall, fog nodes provide more robust and consistent datasets, increasing the effectiveness and reliability of the FL process. Furthermore, they enhance privacy by ensuring that GPS logs are not transmitted over the Internet to a cloud server for training.

Each client dataset denoted as D_f characterizes unique patterns of public transport, acceleration, deceleration, and road conditions specific to that city. The global model is derived through the aggregation of local models from selected clients. Therefore, the FL phase allows the global model to capture the variability and distinctions present in transport data across different cities. To ensure consistency in model training across all clients, we define a standardized set of class labels including stationary, walking, running, bicycling, car, train, tram, subway, bus, and ferry.

In the *Cloud layer*, the cloud server acts as a central server for FL to initialize a global model. Each client (i.e., fog node) receives the current global model from the central server and trains its local model using the available dataset. Then, each client sends the model updates (i.e., model parameters) resulting from local training to the central server. On the central server, model aggregation techniques[1] (e.g., FedAvg) are applied to update the global model.

Note that relying solely on fog nodes (a decentralized FL) instead of a central cloud server could introduce challenges in terms of managing and coordinating the distributed training process effectively. The central cloud server also provides a starting point for model training by initializing the global model and distributing it to the participating clients. Therefore, the global model can capture the variability present in transport

[1] In federated learning, aggregation refers to the process of combining model updates from multiple participating devices (e.g., edge devices) to create a global model. During the training process, each device computes its local model update using its local data and sends this update to a central server or aggregator. The aggregator then combines these updates using a specified aggregation algorithm to generate a new global model [15].

data across different locations, providing a robust basis for subsequent adaptation.

FOGFLEET leverages the Flower FL framework [22] to implement FL and provide a client-server communication infrastructure, enabling coordination between clients and the central server. The central server aggregates the received model updates using the FedAvg algorithm that computes the weighted average of the model updates (i.e., model parameters) from all participating clients to generate a global model. The updated global model is then sent back to the clients for the next round of training. This process iterates for multiple rounds (e.g., 10 rounds) until convergence is met. Throughout the training process, a convergence metric such as accuracy is monitored to assess model performance and convergence [14].

Then, the fog nodes generate personalized models for each smartphone based on the global model generated by the cloud server. We employ the TL technique to personalize models tailored to individual smartphones' characteristics and cities, enhancing model performance. In the TL phase, we fine-tune the global model on a new dataset obtained from a smartphone represented as D_s (cf. Section IV). This fine-tuning process adapts the global model to the specific characteristics of each smartphone's dataset in the new city, taking into account variations in sensor data quality and transport patterns. Note that the fine-tuning process does not influence the global model training since it is performed independently on the new dataset after the global model has been trained and finalized in the FL phase.

Note that for any new smartphone s_{n+1} within a new city of c_{n+1}, the same TL phase can be applied to personalize the base model with the new dataset of $D_{s_{n+1}}$.

A. Classifier structure

In this section, we describe the neural network model structure employed in this study. One of the requirements of a local TMD system is to perform the inference directly on smartphones [13]. Therefore, it is essential to adopt a model that is lightweight enough to perform inference efficiently on smartphones. To meet this requirement, we opt for a shallow Multi-Layer Perceptron (MLP) model. The structure of the MLP model utilized in this study is presented in Table II. The adopted MLP model includes an input layer, three hidden layers, two dropout layers, and an output layer. The global model (i.e., base model) can infer a total of 10 transport modes, 4 non-motorized modes of stationary, walking, running, and bicycling, and 6 modes of motorized modes including car, train, tram, subway, bus, and ferry. Thus in the output layer, the *NUM_CLASSES* corresponds to 10 class labels. We use this structure for each client during the FL process. To make the comparison possible when fitting a model for each smartphone in the centralized scenario, we also use this structure (see Section IV-D).

For the TL phase, we rely on the structure presented in Table III. As shown in this table, we freeze all the base model layers (without the output layer) and add two more hidden layers on top of them. After the initial training, we fine-tune the model with all layers, including those from the base model. The model is then recompiled and trained again with a low learning rate to adapt further to the specific task.

TABLE II: MLP Model Summary for FL phase

Layer	**Parameters**
Input	Shape: *INPUT_SHAPE*
Dense (Hidden Layer 1)	Units: 128, Activation: ReLU[1]
Dropout (Dropout Layer 1)[2]	Rate: 0.10
Dense (Hidden Layer 2)	Units: 128, Activation: ReLU
Dropout (Dropout Layer 2)	Rate: 0.10
Dense (Hidden Layer 3)	Units: 128, Activation: ReLU
Output	Units: *NUM_CLASSES*, Activation: Softmax[3]

[1] ReLU, short for Rectified Linear Unit, is a simple yet effective activation function that outputs the input value directly if it is positive, and zero otherwise [20].

[2] The Dropout layer is used for regularization. It randomly drops some nodes in a Neural Network during training [20].

[3] The Softmax function converts the weighted sum values into probabilities to produce predictions for *NUM_CLASSES* corresponding to each class in the classification task. Therefore, the Softmax function employs a standardization of the outputs to ensure that they collectively sum up to represent one of the classes [20].

TABLE III: MLP Model Summary for TL phase with Frozen Layers

Layer	**Parameters**
Input	Shape: *INPUT_SHAPE*
Base Model	Outputs from global model up to the output layer
Dense (Hidden Layer 1)	Units: 128, Activation: ReLU
Dropout (Dropout Layer 1)	Rate: 0.10
Dense (Hidden Layer 2)	Units: 128, Activation: ReLU
Dense (Output Layer)	Units: *NUM_CLASSES*, Activation: Softmax
Frozen Layers	
Base Model	all layers (up to the output layer)
Fine-Tuning	
All Layers	Including base model

The base model layers, up to the output layer, are frozen during training. After the initial training, the model undergoes a fine-tuning process where all layers, including those from the base model, are trainable. The model is then recompiled and trained again to adapt further to the specific task.

The hyperparameters used in our experiments were carefully selected by trial and error tests that were inspired by the Grid Search method. We first predefined a set of values for each hyperparameter, such as the number of units (e.g., 16, 32, 64, 128, 512, 1024), number of layers (e.g., 2, 3, 4), and various batch sizes, optimizers and learning rates. We then trained the MLP model with the combinations of these predefined hyperparameter values and evaluated each configuration based on performance metrics, such as validation loss. The best-performing configuration was selected based on these tests.

For instance, the learning rate for the global model is set to 0.01, while for the fine-tuned model, a slightly lower learning rate of 0.009 is chosen. We employed custom optimizers for each model, with Adam optimizer [2] for the global model

[2]The Adam optimizer presented by Kingma [18] is a commonly used optimization algorithm in DL, especially for training NNs.

and Nadam optimizer [3] for the fine-tuned model. To prevent overfitting and ensure efficient training, we incorporated early stopping with a patience of 5 epochs and a learning rate scheduler that reduces the learning rate by a factor of 0.5 if the validation loss does not improve for 2 consecutive epochs during the initial training phase. For the fine-tuning phase, a more conservative learning rate scheduler was applied, reducing the learning rate by a factor of 0.05 under similar conditions. Given that the optimal number of epochs varies across experiments, we utilize early stopping to automatically determine when the model has completed training instead of relying on manual adjustment. The patience parameter, indicating the number of epochs, represents a complete pass of the training data through the algorithm without improvement. Once this threshold is reached, training is terminated [20]. Each model is trained for a maximum of 30 epochs with a batch size of 64.

IV. Experimental evaluation

In this section, we begin by detailing the datasets utilized for evaluation, describing the preprocessing techniques employed, and identifying the features selected for analysis. Subsequently, we discuss the performance outcomes of the experiments, which are elaborated upon in Section IV-D.

A. Datasets

As mentioned before, we used an Android-based application called TMDetector mobile application to gather data [5]. TMDetector reads data from accelerometer and magnetometer sensors at a sampling rate of 1Hz, while data from GPS is captured every 10 seconds. This adjustment in GPS sampling rate is due to its higher battery usage compared to the other sensors. Preliminary experiments indicated that this sampling rate suffices for maintaining system accuracy. Data collection is conducted across four distinct cities: Oslo, Rome, Stockholm, and Tehran utilizing the same application to capture a wide range of transport modes, road conditions, and related patterns across the urban areas of these cities.

Data collection within each city involved 10 various participants using different smartphone models, including Sony Xperia XZ2 Compact, Sony Xperia XZS, Samsung Galaxy S21, Motorola Nexus 6, Xiaomi Poco F3, Xiaomi Mi Mix 3, Google Pixel 5a, Xiaomi Mi 10 5G, Samsung Galaxy S10, and Motorola Moto G20. The smartphones used vary in age from 1 to 9 years and run on Android versions ranging from 7 to 13. This diversity covers a broad range of both Android versions and device ages. During data collection, users carried their smartphones in diverse positions such as bags, backpacks, front or back pockets, etc., with instructions to refrain from shaking the devices to prevent data noise. The distribution of datasets within each city varied depending on participants' involvement. We made our collected datasets publicly available on GitHub for TMD research purposes [1].

[3]The Nadam optimizer is an extension of Adam, introduced by Dozat [21]. It combines Adam with Nesterov momentum.

Furthermore, we utilized the SHL dataset, a publicly available dataset [3], [19], collected in London. We refer to the SHL dataset as the London dataset in this work. It is worth noting that each city's dataset is hosted on a fog node dedicated to that specific location. London dataset comprises data from three users, each carrying four HUAWEI Mate 9 smartphones. The smartphones for collecting the dataset were affixed to different parts of the users' bodies, including the torso, hip, bag, and hand. We selected accelerometer, magnetometer, and GPS data from this dataset same as other cities.

B. Preprocessing

We perform data cleaning and noise filtering to rectify sensing and user errors. When gathering the ground truth data using TMDetector, users specify the start and end of a trip. Additionally, they assign a class label, denoting the transport mode. According to our experience, users may inadvertently forget to terminate their trips, resulting in invalid samples, or assign an incorrect modality to a trip. For example, the trip speed indicates movement but the user has annotated the trip as stationary. To address these issues, we employ data filtering to eliminate invalid trips or those labeled with an erroneous modality. Furthermore, data lacking logs from the GPS sensor at the desired sampling rate are filtered out and excluded. Such occurrences typically arise when the smartphone is heavily shielded or when the user is situated in a covered area like a tunnel.

Following filtering and preprocessing, the duration of the Oslo dataset comprises nearly 32 hours, while the Rome dataset contains 56 hours of data. Similarly, the Tehran dataset comprises 58 hours of data, and the Stockholm dataset encompasses 69 hours. As for the London dataset, sensor data was initially sampled at 100Hz. We downsampled it to 1Hz, aligning with our desired sampling rate. Therefore, the London dataset, after our filtering and reformatting processes, spans nearly 14 hours.

C. Feature selection

An MLP can learn the most informative and suitable representation of the raw data during the learning process. Therefore, we introduce our feature set presented in Table IV from the raw sensor data.

Smartphones are equipped with tri-axial accelerometers and magnetometers, resulting in measurements being relative to the device itself rather than to the earth. Although it is feasible to determine earth-relative orientation when the smartphone is stationary, accuracy diminishes when the device is in motion [17]. To address the impact of changes in smartphone orientation during motion, in addition to the tri-axial values, we calculate the magnitude values for both acceleration and magnetic field according to the formula presented in (1).

$$A_{\text{magnitude}} = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{1}$$

We apply min-max normalization to our data to mitigate the issues associated with uneven scales of input features,

TABLE IV: List of Features

Feature Name	Description
accX	X-coordinate of the acceleration
accY	Y-coordinate of the acceleration
accZ	Z-coordinate of the acceleration
lat	Latitude from GPS
lon	Longitude from GPS
acc	Accuracy from GPS
magX	X-coordinate of the magnetic field
magY	Y-coordinate of the magnetic field
magZ	Z-coordinate of the magnetic field
accMagnitude	Acceleration Magnitude calculated based on (1)
magMagnitude	Magnetometer Magnitude calculated based on (1)
distance	Distance calculated based on two consecutive GPS data points
speed	Speed calculated based on two consecutive GPS data points

such as slow convergence or getting stuck in local minima. Min-max normalization scales the features to a fixed range (usually between 0 and 1), ensuring that each feature contributes proportionately to the learning process. The cost of having this bounded range is that we will end up with smaller standard deviations, which can suppress the effect of outliers [20]. Our initial tests with min-max, Robust[4], and Z-score normalization[5]. suggest that min-max scaling performs the best for our solution. In (2) we present the formula to normalize our datasets. The normalization process is done before the datasets are used to train and test the MLP model.

$$x_{\text{normalized}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{2}$$

D. Performance results

This section presents the performance results obtained from three sets of experiments across two scenarios. We utilize the datasets mentioned in Section IV-A for these experiments. In each experiment, one city dataset out of the five is excluded from the FL process. Subsequently, the excluded city dataset is partitioned into subsets based on the smartphones used for data collection within that city known as that specific smartphone dataset. Thereafter, we fine-tune the global model using each smartphone dataset. We selected smartphone datasets collected in three cities of Rome, Tehran, and Stockholm (which have a larger dataset compared to Oslo and London) to evaluate the fine-tuning phase.

For instance, when we opt to exclude the Rome dataset from the FL phase, we divide the Rome dataset into five subsets based on the smartphones used for data collection: specifically, Sony Xperia XZ2 Compact, Sony Xperia XZS, Samsung Galaxy S21, Motorola Nexus 6, and Xiaomi Mi 10. These subsets are then referred to as smartphone datasets. Thereafter, we fine-tune the global model with each smartphone dataset.

As mentioned before, we perform two scenarios for each experiment. In the first scenario, known as the centralized scenario we evaluate the performance result of the MLP model defined in Table II by training this model centrally with each smartphone dataset. In the second scenario, called the FTL scenario, we first use the MLP structure presented in Table II for training the global model (i.e., FL phase). Then we fine-tune the global model with each smartphone dataset (TL phase) according to the structure presented in Table III. Through these two scenarios, we intend to show that our solution performance outperforms the centralized training approach while preserving privacy.

1) Experiment 1: Personalized models for Rome smartphones: As previously mentioned, from the Rome dataset, we selected a subset dedicated to specific smartphones. Thereafter, we conducted tests under both centralized and FTL scenarios. Table V presents the performance results obtained for various smartphones in both centralized and FTL scenarios. Each scenario comprised four trials, with different numbers of samples ranging from approximately 1 hour (3600 samples) to 4 hours (14400 samples). We conducted these experiments with four different trials to accommodate the varying sizes of each smartphone dataset, ensuring that each trial included smartphones with at least 4 hours of data collection.

TABLE V: F1 score values for centralized and FTL Scenarios (Rome dataset)

Smartphone	F1 Score			
	1 hour	2 hours	3 hours	4 hours
Centralized Scenario				
Samsung Galaxy S21	0.72	0.75	0.78	0.79
Sony Xperia XZ2 Compact	0.79	0.90	0.92	0.92
Sony Xperia XZS	0.75	0.87	0.91	0.92
Motorola Nexus 6	0.85	0.92	0.95	0.95
Xiaomi Mi 10	0.84	0.90	0.92	0.92
FTL Scenario				
Samsung Galaxy S21	0.85	0.87	0.88	0.90
Sony Xperia XZ2 Compact	0.93	0.93	0.96	0.96
Sony Xperia XZS	0.89	0.89	0.94	0.96
Motorola Nexus 6	0.93	0.93	0.96	0.96
Xiaomi Mi 10	0.94	0.94	0.95	0.96

As shown in Table V, F1 scores tend to be higher in the FTL scenario compared to the centralized scenario for most smartphones and time durations. This indicates that the FTL solution outperforms the traditional centralized approach. The improvement in F1 scores suggests that leveraging local data and fine-tuning models for personalizing the models contributes positively to the overall classification performance. In this table, the Samsung Galaxy S21 tends to have lower F1 scores compared to other smartphones in both scenarios, especially in the centralized scenario. This variability could be attributed to differences in sensor precision, or data quality among smartphones.

[4]Robust scaling, scales data in a way that is less sensitive to outliers. This is achieved by using robust statistics, such as the median and interquartile range [25].

[5]Z-score scaling, is a technique used to transform data so that it has a mean of zero and a standard deviation of one. This is achieved by subtracting the mean of the feature from each data point and then dividing by the standard deviation of the feature. The resulting data distribution has a mean of zero and a standard deviation of one, allowing for easier comparison between different features with different scales [25].

2) Experiment 2: Personalized models for Tehran smartphones: Same as experiment 1, we conduct four trials for centralized and FTL scenarios for the Tehran dataset. Table VI presents the performance results obtained for various smartphones in centralized and FTL scenarios. Similar to the previous experiment, the F1 scores in the FTL scenario tend to be higher compared to the centralized scenario for most smartphones and time durations. This suggests that exploiting the FTL framework leads to improved classification performance compared to the centralized approach, consistent with findings from the previous experiment.

TABLE VI: F1 score values for centralized and FTL Scenarios (Tehran dataset)

Smartphone	**F1 Score**			
	1 hour	**2 hours**	**3 hours**	**4 hours**
Centralized Scenario				
Samsung Galaxy S21	0.72	0.80	0.84	0.88
Sony Xperia XZ2 Compact	0.86	0.86	0.88	0.90
Sony Xperia XZS	0.88	0.86	0.90	0.95
Motorola Nexus 6	0.86	0.91	0.94	0.95
FTL Scenario				
Samsung Galaxy S21	0.92	0.92	0.93	0.95
Sony Xperia XZ2 Compact	0.94	0.95	0.96	0.96
Sony Xperia XZS	0.95	0.97	0.98	0.98
Motorola Nexus 6	0.92	0.92	0.96	0.98

3) Experiment 3: Personalized models for Stockholm smartphones: Table VII represents the F1 score values for the Stockholm dataset repeating the four trials performed in the previous experiments. Similar to experiments 1 and 2, the FTL scenario outperforms the centralized scenario considering the F1 score values. Samsung Galaxy S10 consistently exhibits the highest F1 scores among all smartphones in both scenarios, indicating its superior performance in classifying transport modes. In contrast, F1 score values for the Samsung Galaxy S21 compared to other smartphones are lower in both scenarios. Furthermore, Sony Xperia XZS shows a noticeable improvement in F1 score in the FTL scenario compared to the centralized scenario. These variabilities highlight the influence of smartphone characteristics, and sensor quality on classification performance.

Assessing all the tables presenting the F1 score values for Rome, Tehran, and Stockholm presented in Tables V, Table VI and Table VII, suggests that with higher number of samples on each smartphone, the performance results can improve. The aforementioned findings underscore the significance of having a sufficient volume of training data and suggest that the current F1 score values are heavily dependent on the quantity of training samples. Furthermore, consistently lower F1 score values for Samsung Galaxy S21 and higher F1 score values for Motorola Nexus 6 among the other smartphones in all the experiments for Rome, Tehran, and Stockholm datasets, imply the necessity for defining a personalized model for each smartphone according to the variations in their sensor precision.

TABLE VII: F1 score values for centralized and FTL Scenarios (Stockholm dataset)

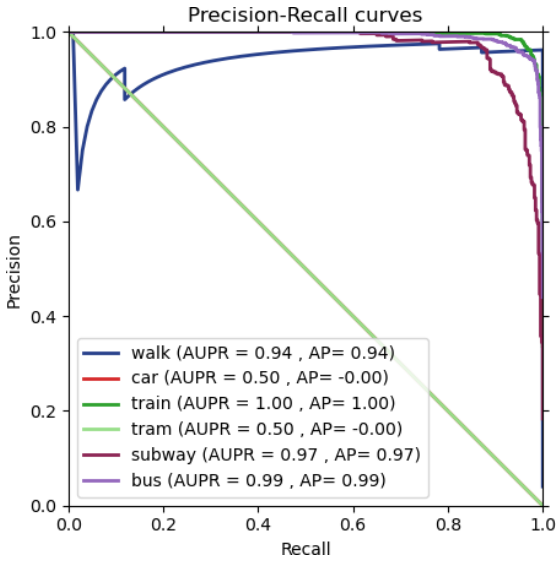
Smartphone	**F1 Score**			
	1 hour	**2 hours**	**3 hours**	**4 hours**
Centralized Scenario				
Samsung Galaxy S21	0.71	0.74	0.75	0.75
Sony Xperia XZ2 Compact	0.81	0.87	0.89	0.89
Sony Xperia XZS	0.74	0.80	0.80	0.84
Motorola Nexus 6	0.85	0.90	0.92	0.92
Samsung Galaxy S10	0.94	0.95	0.96	0.96
FTL Scenario				
Samsung Galaxy S21	0.76	0.81	0.81	0.82
Sony Xperia XZ2 Compact	0.87	0.91	0.92	0.93
Sony Xperia XZS	0.88	0.91	0.91	0.91
Motorola Nexus 6	0.86	0.93	0.94	0.96
Samsung Galaxy S10	0.95	0.97	0.97	0.97

To delve deeper into understanding the impact of smartphone characteristics and sensor quality on classification performance, we focus our analysis on two specific smartphones from Experiment 3: Samsung Galaxy S10 and Sony Xperia XZS. These smartphones are chosen due to their interesting behavior across both centralized and FTL scenarios. In this analysis, we utilize precision-recall and ROC (Receiver Operating Characteristic) curves [6]. These curves allow us to evaluate the performance improvements from the centralized scenario to the FTL scenario for each smartphone, specifically examining each class individually.

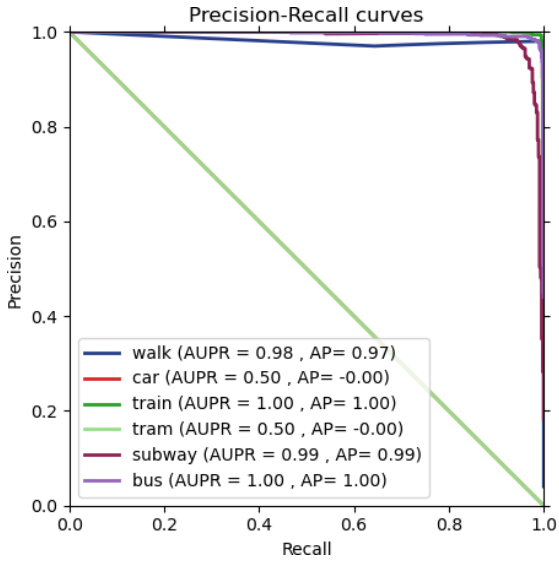
Fig. 2 depicts precision-recall and ROC curves for four available modes (i.e., walk, train, subway, and bus) in the Samsung Galaxy S10 dataset with 4 hours duration. Unavailable modes in this dataset are shown with a steady line. As shown in the figure, there are noticeable improvements in Average Precision (AP) observed in part (b) compared to part (a) and in part (d) compared to part (c). Additionally, AUROC values in part (d) are closer to 1 for most transport modes compared to part (c), exhibiting enhanced discriminative power in the FTL scenario.

Fig. 3 depicts precision-recall and ROC curves for five available modes (i.e., walk, train, tram, subway, and bus) in the dataset of Sony Xperia XZS with 4 hours duration. The unavailable mode of the car modality is shown with a steady line in this figure. A comparison between parts (a) and (b) in this figure, reveals a notable increase in AP for the train modality by 6%. Additionally, there is a 5% increase in AP values across the walk, subway, and bus modalities from the centralized scenario to the FTL scenario. Furthermore, the

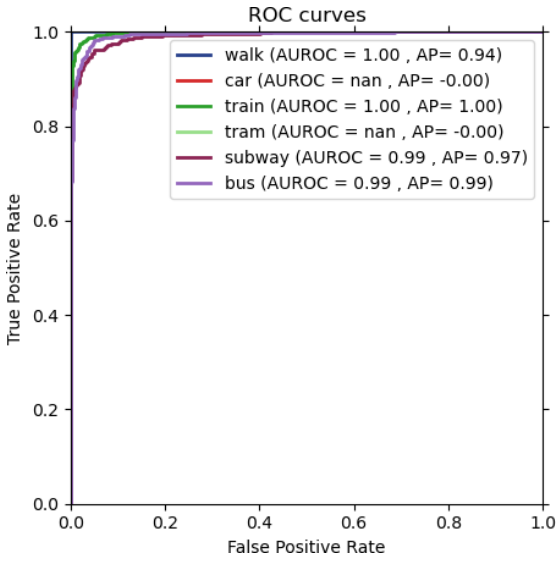
[6]Precision-recall curves demonstrate a model's ability to maintain high precision and recall, especially useful in datasets with imbalanced classes. The Receiver Operating Characteristic (ROC) curves highlight the balance between true positive and false positive rates, providing key performance insights. The Area Under the Precision-Recall Curve (AUPR) and the Area Under the Receiver Operating Characteristic (AUROC) quantitatively assess these aspects, respectively. A high AUPR (close to 1) indicates effective identification of positive instances with minimal false positives, while a high AUROC (close to 1) suggests strong differentiation between positive and negative instances. .



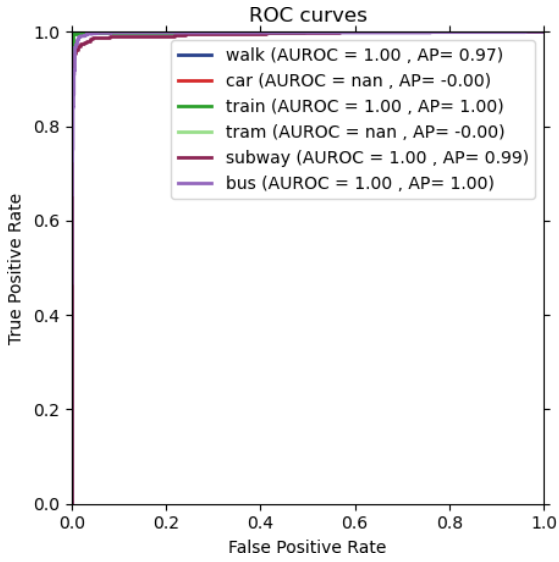
(a) Precision-recall curves for centralized scenario



(b) Precision-recall curves for FTL scenario



(c) ROC curves for centralized scenario



(d) ROC curves for FLT scenario

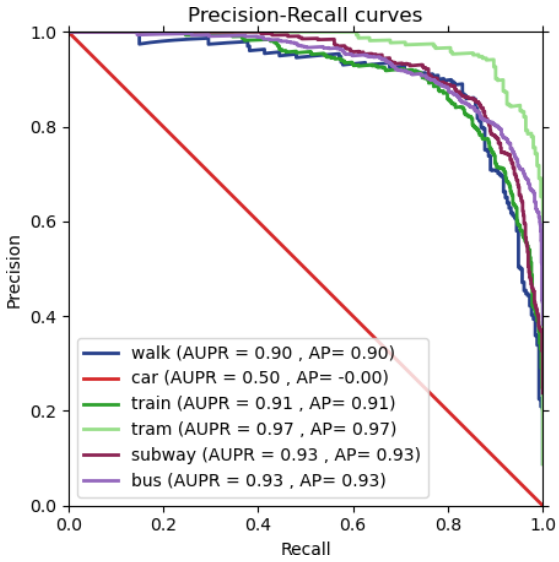
Fig. 2: Experiment 3, Trial 4 hours: Precision-recall and ROC curves of centralized and FTL scenarios for Samsung Galaxy S10

comparison between part (a) and part (b) demonstrates higher AUPR values across all transport modes. Similarly, in part (d), AUROC values are closer to 1 compared to part (c), indicating the superiority of the FTL scenario to the centralized scenario in distinguishing between different transport modes.

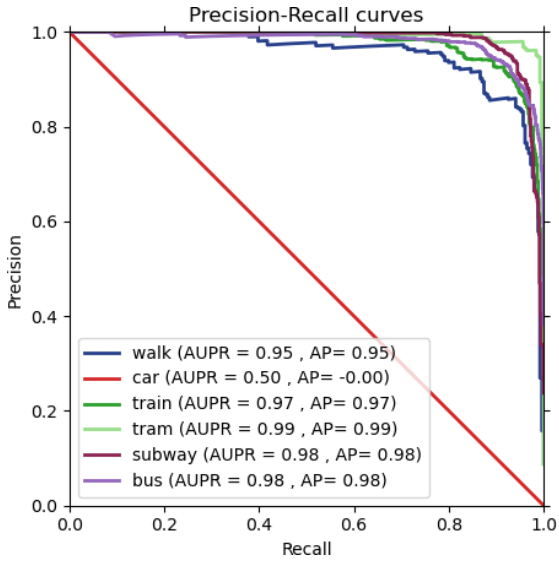
The performance improvements are particularly noticeable when the dataset duration is restricted to 1 hour, as shown in Fig. 4. This figure depicts precision-recall and ROC curves for five available modes (i.e., walk, train, tram, subway, and bus) in the dataset of Sony Xperia XZS with a 1-hour duration. In this figure, we specifically observe a 13% increase in AP for the tram and subway modalities when comparing part (a) to part (b). This highlights the effectiveness of the FTL solution, which can quickly personalize models for new smartphones across diverse locations, even with a small amount of training data, given the precision of smartphone sensors.

V. Conclusion

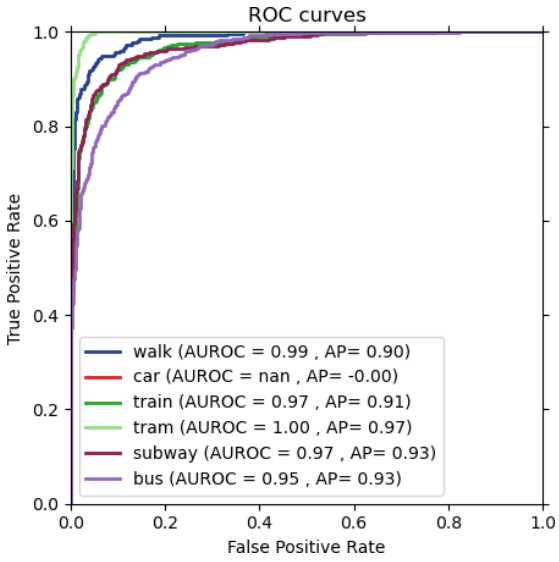
In this work, we presented FogFLeeT relying on a novel FTL solution that efficiently adapts an MLP model to new cities for TMD while leveraging the diverse data available from multiple smartphones. We deployed our FTL solution with a fog-based architecture, exploiting the fog nodes as the federated learning clients. In this way, we not only preserve the user's privacy (city-wide) but also achieve better performance results than a centralized scenario. By fine-tuning a global



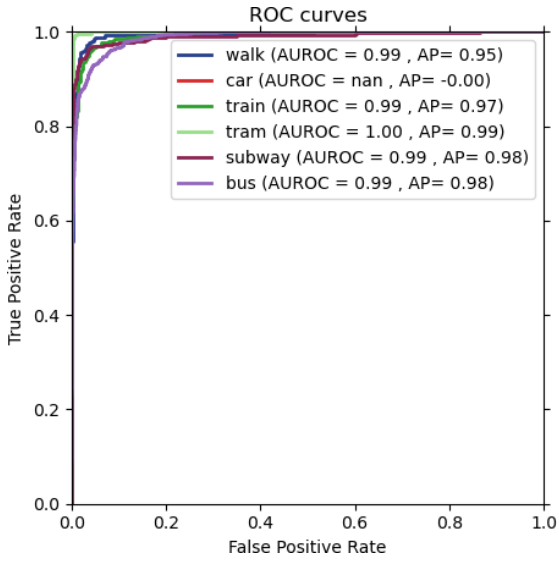
(a) Precision-recall curves for centralized scenario



(b) Precision-recall curves for FTL scenario



(c) ROC curves for centralized scenario



(d) ROC curves for FLT scenario

Fig. 3: Experiment 3, Trial 4 hours: Precision-recall and ROC curves of centralized and FTL scenarios for Sony Xperia XZS

model with data from individual smartphones in a new city, we demonstrated significant improvements in model performance, even with limited training data (e.g., 1 hour). Depending on the smartphone sensor precision, data quality, and time duration our solution obtained a minimum of 1% and a maximum of 20% higher F1 score values when comparing it to a centralized scenario. Our solution not only accommodates the varying characteristics of transport modes across different cities but also highlights the effectiveness of personalized models in achieving high accuracy. This emphasizes the potential of our FTL solution for rapidly deploying accurate TMD systems in new locations, eventually contributing to more adaptive and privacy-preserving TMD systems.

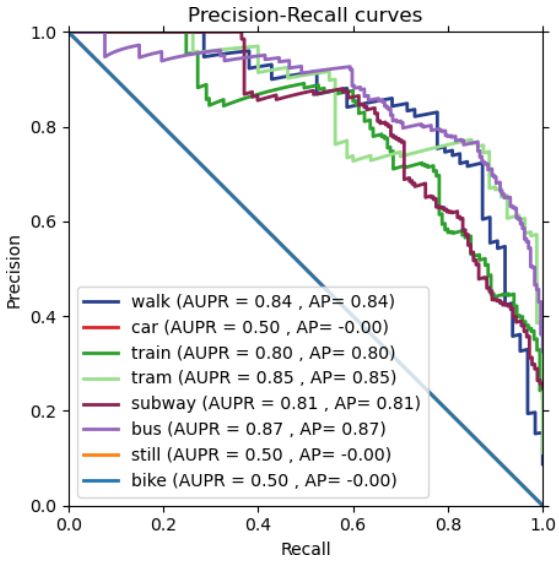
VI. ACKNOWLEDGEMENT

This work was partly supported by the Norwegian Research Council under Grants 262854/F20 (DILUTE project) and 322473 (AirQMan project).

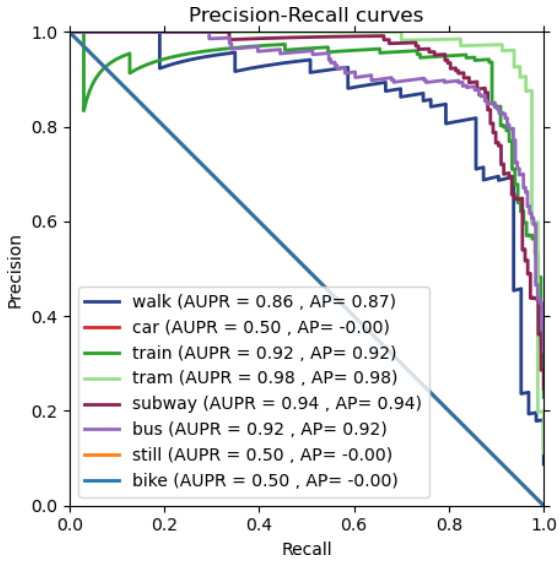
REFERENCES

[1] M. Kamalian, and P. Ferreira, and A. Taherkordi *TMDetector Sample Datasets from Oslo, Rome, Tehran, Stockholm, Github https://github.com/mahdieh-ka/TMDetector-sample-data*

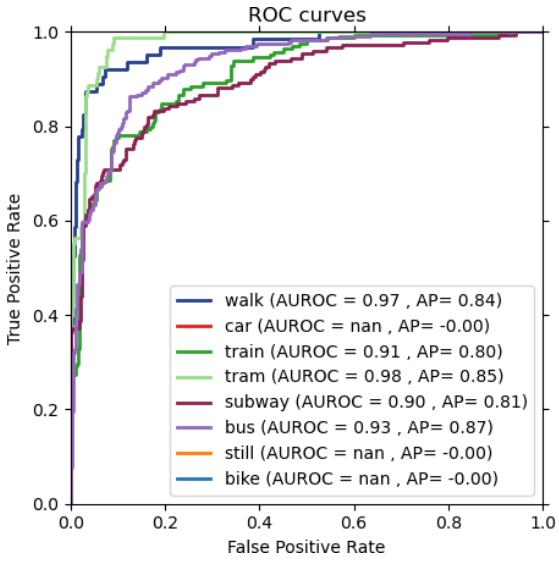
[2] Huang, Qihan and Zhang, Jing and Zeng, Zuanyang and He, Ding and Ye, Xiucai and Chen, Yi *PPDF-FedTMI: A Federated Learning-based Transport Mode Inference Model with Privacy-Preserving Data Fusion*, *Simulation Modelling Practice and Theory*, vol. 129, p. 102845, 2023, Elsevier.



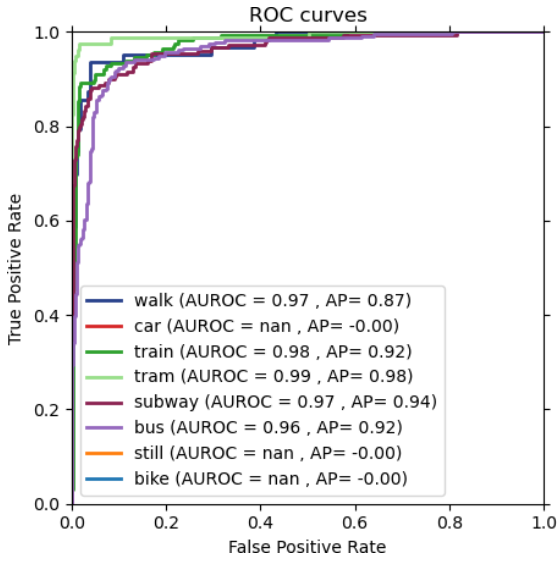
(a) Precision-recall curves for centralized scenario



(b) Precision-recall curves for FTL scenario



(c) ROC curves for centralized scenario



(d) ROC curves for FLT scenario

Fig. 4: Experiment 3, Trial 1 hour: Precision-recall and ROC curves of centralized and FTL scenarios for Sony Xperia XZS

[3] Wang, Lin and Gjoreski, Hristijan and Ciliberto, Mathias and Lago, Paula and Murao, Kazuya and Okita, Tsuyoshi and Roggen, Daniel, *Summary of SHL challenge 2023: Recognizing locomotion and transportation mode from GPS and motion sensors*, *Adjunct Proceedings of the 2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing & the 2023 ACM International Symposium on Wearable Computing, 575–585, 2023.*

[4] eFedDNN: Ensemble based federated deep neural networks for trajectory mode inference, *Mensah, Daniel Opoku and Badu-Marfo, Godwin and Al Mallah, Ranwa and Farooq, Bilal*, *2022 IEEE International Smart Cities Conference (ISC2)*, pp. 1–7, 2022, IEEE

[5] M. Kamalian and P. Ferreira, "FogTMDetector-Fog Based Transport Mode Detection using Smartphones," in *2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC)*, pp. 9–16, 2022, IEEE.

[6] M. Oplenskedal, A. Taherkordi, and P. Herrmann *DeepMatch: deep matching for in-vehicle presence detection in transportation*, *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems (DEBS 2020)*, Montreal, Quebec, Canada, 2020.

[7] M. A. Fauzi, B. Yang, and B. Blobel, "Comparative analysis between individual, centralized, and federated learning for smartwatch based stress detection," *Journal of Personalized Medicine*, vol. 12, no. 10, pp. 1584, 2022, MDPI.

[8] F. Yu, Z. Xu, Z. Qin, and X. Chen, "Privacy-preserving federated learning for transportation mode prediction based on personal mobility data," *High-Confidence Computing*, vol. 2, no. 4, pp. 100082, 2022, Elsevier.

[9] I. C. Cavalcante, R. I. Meneguette, R. H. Torres, L. Y. Mano, V. P. Gonçalves, J. Ueyama, G. Pessin, G. D. Amvame Nze, and G. P. Rocha Filho, "Federated system for transport mode detection," *Energies*, vol. 15, no. 23, pp. 9256, 2022, MDPI.

[10] P. Ferreira, C. Zavgorodnii, and L. Veiga, "edgeTrans-Edge transport mode detection," *Pervasive and Mobile Computing*, vol. 69, p. 101268, 2020, Elsevier.

[11] Wang, Sixian and Yao, Shengshi and Niu, Kai and Dong, Chao and Qin, Cheng and Zhuang, Hongcheng, *Intelligent Scene Recognition Based on Deep Learning*, vol. 9, pp. 24984–24993, IEEE Access, 2021, IEEE.

[12] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated

learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.
[13] Mahdieh Kamalian, Paulo Ferreira, Eric Jul, *A survey on local transport mode detection on the edge of the network*, *Applied Intelligence*, vol. 52, no. 14, pp. 16021-16050, 2022, Springer.
[14] P. Qi, D. Chiaro, A. Guzzo, M. Ianni, G. Fortino, and F. Piccialli, "Model aggregation techniques in federated learning: A comprehensive survey," *Future Generation Computer Systems*, 2023, Elsevier.
[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.
[16] Alessio D Marra, Henrik Becker, Kay W Axhausen, Francesco Corman, *Developing a passive GPS tracking system to study long-term travel behavior*, *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 348-368, 2019, Elsevier.
[17] Bryan D. Martin, Vittorio Addona, Julian Wolfson, Gediminas Adomavicius, Yingling Fan, *Methods for Real-Time Prediction of the Mode of Travel Using Smartphone-Based GPS and Accelerometer Data*, *Sensors*, vol. 17, no. 9, article number: 2058, 2017, URL: http://www.mdpi.com/1424-8220/17/9/2058, ISSN: 1424-8220, DOI: 10.3390/s17092058.
[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[19] L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen, "Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset," *IEEE Access*, vol. 7, pp. 10870–10891, 2019, IEEE, doi: 10.1109/ACCESS.2019.2890793.
[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.
[21] T. Dozat, "Incorporating Nesterov momentum into Adam," *Dozat, Timothy*, 2016.
[22] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, et al., "Flower: A friendly federated learning research framework," arXiv preprint arXiv:2007.14390, 2020.
[23] K.-Y. Chen, R. C. Shah, J. Huang, and L. Nachman, "Mago: Mode of transport inference using the hall-effect magnetic sensor and accelerometer," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1–23, 2017, ACM New York, NY, USA.
[24] Kashif Naseer Qureshi, Abdul Hanan Abdullah, *A survey on intelligent transportation systems*, *Middle-East Journal of Scientific Research*, vol. 15, no. 5, pp. 629-642, 2013.
[25] M. Kuhn and K. Johnson, *Applied Predictive Modeling*, vol. 26, Springer, 2013.