Distributed federated and incremental learning for electric vehicles model development in Kafka-ML

Alejandro Carnero
ITIS Software
University of Malaga
Malaga, Spain
alexcarnero97@uma.es

Omer Waqar
University of the Fraser Valley
Abbotsford, British Columbia, Canada
omer.waqar@ufv.ca

Cristian Martín
ITIS Software
University of Malaga
Malaga, Spain
cristian@uma.es

Manuel Díaz
ITIS Software
University of Malaga
Malaga, Spain
mdiaz@uma.es

***Abstract*—With the increasing development and deployment of new systems for efficient and clean mobility, Electric Vehicles (EVs) are becoming more and more common among people. Those produce large amounts of data streams that need to be collected and analyzed to understand user needs and improve their performance. For this purpose, Artificial Intelligence (AI) techniques are playing a very important role. Within this context, Kafka-ML is a Machine Learning (ML) framework that enables the consumption and processing of data streams and allows the flexible management and deployment of neural networks throughout their entire life cycle. Kafka-ML can work with Distributed Neural Networks (DNN) which reduce latency and response times, perform incremental training over time allowing models to adapt to data on the fly, and carry out Federated Learning (FL) processes for this type of algorithms so a more robust global model can be created while maintaining data privacy and security, but all this separately. This work has considered the joint implementation of FL, for anonymous data sharing, incremental learning for continuous training of the models, and DNN for distribution of the models across different points on the map. All this applied within a Vehicle-to-everything (V2X) domain where EV usage and charge data can be shared to improve the user experience, as well as to better understand the behavior of this type of vehicles and their charging points to achieve savings, and how it affects people daily lives. An evaluation of the system related to this EV use case is presented to demonstrate the viability of the tool.**

***Index Terms*—electric vehicles, federated incremental machine learning, distributed neural networks, data streams, Kafka-ML**

I. Introduction

Due to the extensive usage of sensors and mobile apps, the Internet of Things (IoT) has become a research hotspot [1], and Machine Learning (ML) techniques like Deep Neural Networks (DNN) are tempting for managing IoT data sources [2]. The traditional technique of centralizing all accessible data and training new models each time there is an update is not sustainable, as current applications need to keep enormous volumes of data while models consume fresh data [3]. Additionally, the increasing prevalence of non-stationary data streams can lead to concept drift and the obsolescence of prediction models [4]. As a result, incremental learning procedures together with unsupervised labelling processes are crucial for these algorithms since they enable dynamic data adaptation. In turn, Federated Learning (FL) is a decentralized method that provides answers to problems with data access, security, and privacy [5]. By allowing users to save their data locally, FL protects user privacy, prevents data leakage, and enables the development of a larger, more reliable model through the sharing of information across the different working nodes [6].

In smart grids, Plug-in Electric Vehicles (PEVs) are considered components of the power demand and they are escalating in the market due to their energy-saving and environmental benefits. Also, the addition of a great number of PEVs to the current distribution network poses serious challenges to the power system [7]. To handle the PEV charging demand from both technical and financial points of view, aggregators include tools based on IoT technology, which can observe the users' historical behavior and estimate their travel behavior and the requested charging demand. Shortly, the increase in the share of PEV adoption will transform the PEV demand modeling framework into a big-data cyber-physical system [8].

However, there are some difficulties related to the development of PEVs that need to be overcome. On the one hand, both car owners and regulators pay more attention to data privacy and are unwilling to share data, which forms the isolated data island challenge. On the other hand, the incremental data generated from PEVs are massive and diverse. This being the case, the current FL or incremental learning frameworks can not effectively integrate this incremental PEVs data with existing ML models [9].

This work aims to develop an efficient solution for handling incremental data together with data privacy (FL processes) within the open-source ML framework Kafka-ML [10] while accommodating a shared and secure workspace for data exchange within the PEVs sector to overcome the existing challenges in this field. Kafka-ML is an open-source ML framework for the management and deployment, including training and inference, of neural networks based on data streams. This tool supports, on the one hand, the incremental training of neural networks over time, for both single and partitioned models, and on the other hand, FL processes, but

WINCOM 2024. DOI: https://doi.org/10.1109/WINCOM62286.2024.10654814

in an independent manner. For this reason, this work aims to develop the joint functionality of the two learning paradigms for EV usage and charge data. The validation considers a use case related to a PEV charging scenario to understand user behavior in this field of application.

The rest of the article is organized as follows. Section II presents the related work of this field of research. In Section III, the preliminary Kafka-ML architecture for federated incremental deployments is presented. Then, Section IV explains the motivation and the selected use case to do this project. Section V shows the first results of the validation of the framework. Finally, Section VI presents our conclusions and future directions of this work.

II. Related work

This section presents a set of recent articles that are related to incremental and FL to better understand the state-of-the-art of these research topics.

In [11], Nan Yang et al. introduced a novel framework called FedMAE, addressing the training of large-scale distributed unlabeled datasets in FL, being the first to apply Transformer [12] in this area of research. Their pre-training model only contains a one-block encoder and a one-block decoder and uses a masking strategy for training data, so it is easy to deploy in various lightweight clients. Their pre-training models using FL can be trained on different clients asynchronously without average aggregation. Pre-trained models are then cascaded in the server to build a multi-block downstream model. Theoretical analysis and experimental results on image reconstruction and classification show that their FedMAE achieves a considerable boost compared to previous methods. However, this work does not consider an incremental learning process of their ML models over time.

Also, in [13], Donald Shenaj et al. introduced a new setting combining continual and FL. They allow each client to follow its continual learning path asynchronously with other clients, while the server orchestrates the distributed training to safely aggregate the different learned models coming from the streams of information locally acquired by the clients. They tackle this novel task using prototype-based learning, a representation loss, fractal pre-training, and a modified aggregation policy. The overall objective of their method is to end up with a prediction model that has assimilated all the tasks knowledge acquired by individual clients, avoiding catastrophic forgetting inherent in the incremental local task progression. Nevertheless, this project does not envisage working with partitioned neural networks.

In turn, to address models catastrophic forgetting of old classes and limitation of communication cost when using large-scale models in FL, in [14], Chenghao Liu et al. proposed a novel framework called Federated Enhanced Transformer (FedET), which simultaneously achieves high accuracy and low communication cost. Specifically, FedET uses Enhancer, a tiny module, to absorb and communicate new knowledge, and applies pre-trained Transformers combined with different Enhancers to ensure high precision on various tasks. To address local forgetting caused by new classes of new tasks and global forgetting brought by non-independent and identically distributed (IID) class imbalance across different local clients, they proposed an Enhancer distillation method to modify the imbalance between old and new knowledge and repair the non-IID problem. Despite this, their framework has not been tested and evaluated within the PEVs sector.

On the other hand, in [15], Yuchen Zhao et al. proposed an activity recognition system that uses semi-supervised FL, wherein clients conduct unsupervised learning on autoencoders with unlabelled local data to learn general representations, and a cloud server conducts supervised learning on an activity classifier with labeled data. Their experimental results show that using a long short-term memory autoencoder and a Softmax classifier, the accuracy of their proposed system is higher than that of both centralized systems and semi-supervised FL using data augmentation. The accuracy is also comparable to that of supervised FL systems. Meanwhile, they demonstrate that their system can reduce the number of needed labels and the size of local models, and has faster local activity recognition speed than supervised FL does. Even so, this system does not support FL processes for DNN.

Meanwhile, in [16], Haowen Lin et al. focus on a new scenario for cross-silo FL, where data samples of each client are partially labeled. They borrow ideas from semi-supervised learning methods where a large amount of unlabeled data is utilized to improve model accuracy despite limited access to labeled examples. They propose a new framework dubbed SemiFed that unifies two dominant approaches for semi-supervised learning: consistency regularization and pseudo-labeling. SemiFed first applies advanced data augmentation techniques to enforce consistency regularization and then generates pseudo labels using the model predictions during training. SemiFed takes advantage of the federation so that for a given image, the pseudo-label holds only if multiple models from different clients produce a high-confidence prediction and agree on the same label. Nonetheless, this work lacks the ability to incrementally train their models over time as the data changes and new labels appear.

In addition, in [17], Souad Ajjaj et al. propose a new approach for attack detection in Vehicular Ad hoc Networks (VANETs) based on incremental online ML. Their approach uses data collected from the monitoring of the VANET nodes behavior in real-time and trains an online model using incremental online learning algorithms. More specifically, this research addresses the detection of black hole attacks that pose a significant threat to the Ad hoc On Demand Distance Vector routing protocol. Finally, the performance of two online incremental classifiers are assessed and compared. However, this work does not consider the use of

FL techniques for its application case, which could be very beneficial.

Therefore, to address the specific challenge related to PEVs in this work, our open-source ML framework Kafka-ML offers a flexible development environment that combines FL of deep neural networks to share anonymous information while preserving data privacy, incremental training processes for them over time to adapt to new data as it changes, and partitioning and distribution of these algorithms along different charging points of these vehicles to study their whole behavior, save costs, and improve the user experience.

III. KAFKA-ML ARCHITECTURE

In this work, we redesigned the architecture of our open-source ML framework Kafka-ML to allow federated and incremental training of DNN. On the one hand, the implementation and deployment of DNN offer advantages such as latency reduction thanks to early outputs and distributed processing load [18]; on the other hand, performing FL processes allows maintaining data security and privacy while creating a larger and more robust model thanks to the work of different computing nodes [19]; and finally, performing incremental training over time allows models to dynamically adapt to new data within changing environments [20].

As explained in [19], the FL process in Kafka-ML is a novel mechanism to carry out distributed and asynchronous FL considering flexible models and data. Several constraints can be used to define the minimum number of data to train on, the types of data, the shapes of the input data to the model, etc. Once the models are deployed in Kafka-ML, their architecture, and the initial weights are published in a specific Apache Kafka topic. In this way, federated clients can always know whether a new model is available. When detected, federated clients also check if there is data available in a local Kafka topic to train that model, in case this data matches the constraints previously defined for them. If the data constraints match, the federated Kafka-ML client deploys a worker task to perform the training with the corresponding model and data. Once trained, that task sends the new model weights to the main Kafka-ML training task via another specific Kafka topic for further aggregation with the global model. Once the weights have been added, the main training component republishes the new weights in the corresponding topic so that federated clients can continue with the following iterations. Each of these iterations constitutes a round of aggregation, and this process is repeated until all of them are completed. Figure 1 shows a sequence diagram describing the whole procedure.

The FL process has been divided into two distinct parts. The first refers to the internal work of our Kafka-ML platform, and the second refers to the work carried out within the federated nodes. The new functions within the main Kafka-ML module consist of generating and sending the data standardization, to specify those models within the system that comply with that structure and will therefore perform the federated training and the federated Kafka topics for the exchange of model weights. Then, as long as all the aggregation rounds are not completed, the model is sent to the federated nodes and waits to receive the new weights obtained from the training of these devices, to subsequently aggregate them in the global model. Once all the aggregation phases are completed, the resulting metrics are parsed and sent to the Kafka-ML back-end. On the other hand, the work done in the federated nodes consists of waiting to receive the last version of the model and loading it into memory. As this is incremental training over time with a flexible dataset size, each iteration of the training is performed with a new batch of data. This is the reason why, when deploying a model, the user has to set a timeout to receive each of them. So as soon as the new batch is received, the incremental training of the model is performed, and if there is no further data within the time limit, the training finishes, the metrics are saved, and the new weights are sent back to the main federated Kafka-ML task for aggregation with the global model. Finally, if the version of the model is received as -1, the FL process is completed. Algorithm 1, for the previously explained working process within the Kafka-ML platform, and Algorithm 2, for the operation of the federated client in charge of carrying out the training, describe the complete learning process of the federated incremental deployment for both single and distributed DNN. For simplicity, variables and function parameters have been omitted.

Algorithm 1 Workflow algorithm of federated incremental training at Kafka-ML platform

```
get_models()
if distributed then
    configure_distributed_models()
end if
generate_and_send_data_standardization()
generate_federated_kafka_topics()
while rounds < agg_rounds do
    send_model_to_federated_devices()
    aggregation = False
    while !aggregation && new_weights do
        receive_weights()
        set_weights()
        aggregate_model()
        rounds += 1
        aggregation = True
    end while
end while
parse_metrics()
send_final_metrics()
```

The structure of Kafka-ML utilizes a microservice infrastructure, composed of various elements crafted with specific responsibilities. These elements are encapsulated as Docker containers, providing improved isolation and adaptability. The deployment of Kafka-ML and its elements, along

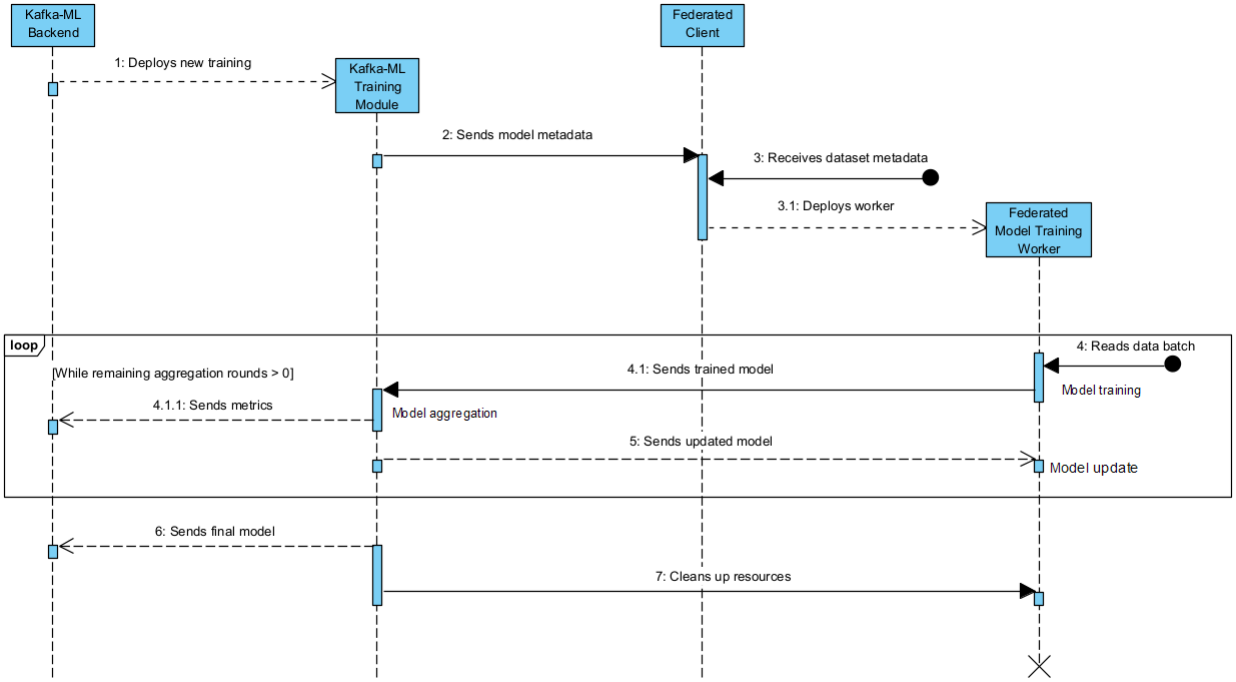


Fig. 1: Federated model training process in Kafka-ML

with node clusters for distributed and production setups, is orchestrated by Kubernetes. Kafka-ML is an open-source project for deploying and managing the pipeline of ML models, with its implementation, configurations, Kubernetes deployment files, and examples available on our GitHub repository[1]. In Figure 2, an overview of the Kafka-ML architecture is presented, with newly introduced components highlighted in blue and modified components in orange. The control logger, as usual, manages control messages and sends them to the back-end. Additionally, Kubernetes takes on the responsibility of managing Apache Kafka and Zookeeper as Docker containers, which are essential components needed by Kafka.

On the one hand, the back-end module manages the operation of Kafka-ML data and initiates the deployment of required elements in Kubernetes upon request, such as training tasks. The primary modifications that will be added to the back-end focus on facilitating the integration between the three ML paradigms to provide joint functionality, which impacts directly the deployment formulation within the framework. Additionally, new functionalities have been developed to execute the incremental federated training phase of distributed models. On the other hand, the front-end module continues to provide the essential web interface for users to connect with Kafka-ML and access its full range

[1]https://github.com/ertis-research/kafka-ml

Algorithm 2 Workflow algorithm of federated incremental training at federated Kafka-ML client

```
while model_message do
    read_model()
    load_model()
    while !timeout && data do
        get_data_batch()
        incremental_training()
    end while
    save_metrics()
    send_model_and_metrics()
    if version == −1 then
        break
    end if
end while
```

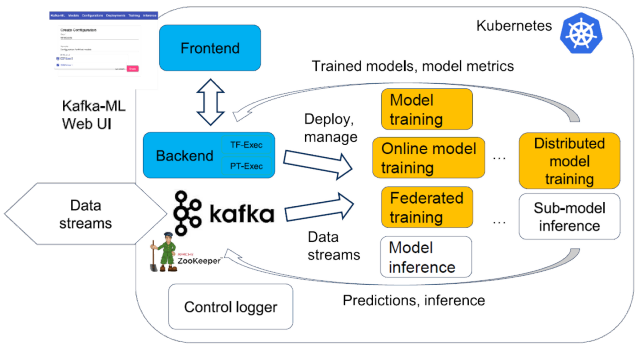


Fig. 2: Overview of the new Kafka-ML architecture and its components deployed and managed in Kubernetes clusters. Orange component: modified; blue component: added.

of features. This module interfaces with the REST API established in the back-end, and the key adjustments that will be introduced in this new version aim to enhance the specialized forms within the framework pipeline, handling the management and support of incremental federated deployments for DNN. However, as the federated incremental deployment has no impact on this stage of the pipeline, the inference component will stay the same for both single and distributed models.

IV. Use case: dynamic electric vehicle learning and model development

As the global shift towards sustainable mobility gains momentum, PEVs have emerged as a central component of this transition, offering a cleaner and more environmentally friendly mode of transportation. However, with the increasing prevalence of PEVs comes a substantial influx of data streams, encompassing diverse aspects of vehicle performance, charging infrastructure, and user behavior. Effectively managing and interpreting this wealth of data is critical for optimizing PEV's performance and understanding its impact on daily lives.

In the PEVs sector, we want to design an infrastructure in which the generation and sharing of data streams from various PEVs and their charging stations are continuously learning once new transactions are carried out. The other important parts of the system are the PEVs, their charging points, and the federated ML models. The approach is to create a work process that starts when the cars are left charging at the stations so that all their data is downloaded there, processed, and used to train federated models incrementally. Then, all the information from different charging points would be aggregated into a global and more robust model placed in a fog/edge computation node for the study and prediction of different aspects that have a certain degree of interest. Thanks to this system, all entities related to PEVs that want to join this anonymous data-sharing plan will be able to do so and acquire relevant information about the vehicle's performance, charging infrastructure, and user behavior. Currently, we do not use any encryption mechanism for the preservation of security and privacy of sensitive data related to PEVs within the FL process, but it is considered as a future work. On the other hand, as shown in Section V, our Kafka-ML platform is able to handle the scalability issue when increasing the number of simultaneous requests produced by different users. The final goal would be to study and predict the average charging time per vehicle according to its characteristics, the most economically affordable charging hours, the most efficient routes to nearby charging stations based on traffic and electricity prices, etc. Figure 3 shows a schematic of a general FL framework for secure data exchange between different entities.

In parallel, this study directs its focus toward our Kafka-ML framework, an ML tool tailored for real-time data

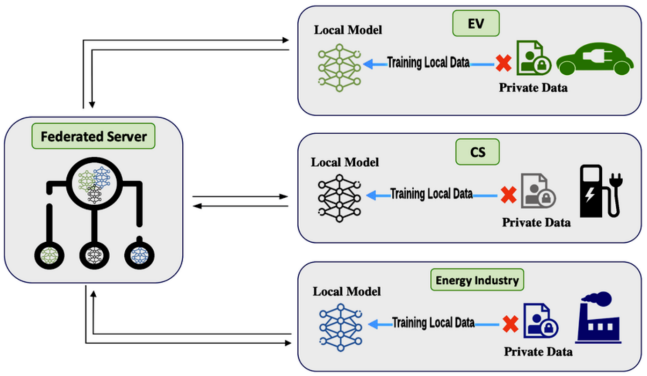


Fig. 3: General FL scheme for secure data exchange [21]

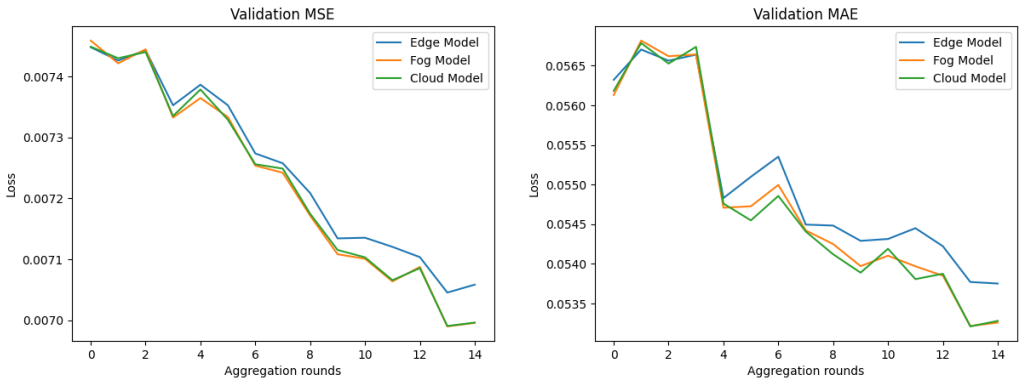
streams. The motivation lies in enhancing Kafka-ML capabilities to seamlessly integrate key ML tasks, such as federated and incremental training of DNN jointly, thus empowering it to efficiently process and derive valuable insights from the varied and dynamic data generated by PEVs. This would create a central access point for interested members from which to work with all the data from different cars and different charging stations, as well as to deploy new models to join the network. By bridging this gap, we aim to not only optimize PEV functionality but also contribute to the broader landscape of AI-driven solutions for sustainable and efficient mobility.

V. Evaluation

The work developed within our Kafka-ML framework allows users to perform incremental federated training deployments over time for their partitioned ML models, in a collaborative manner. For the first validation of this new joint functionality, three tests were carried out to evaluate two different scenarios within the PEVs sector: on the one hand, the occupancy level of this type of vehicle in different charging points, and on the other hand, the specific time that these vehicles take to charge their batteries. This has been possible thanks to the use of a dataset from Palo Alto (CA, EE. UU) that collects almost 260,000 charging records from different users throughout the city[2]. This dataset has been cleaned and preprocessed appropriately, leaving as input to the models the charging station ID, occupancy/charging start date, charging port type and number, vehicle connector type, and user ID.

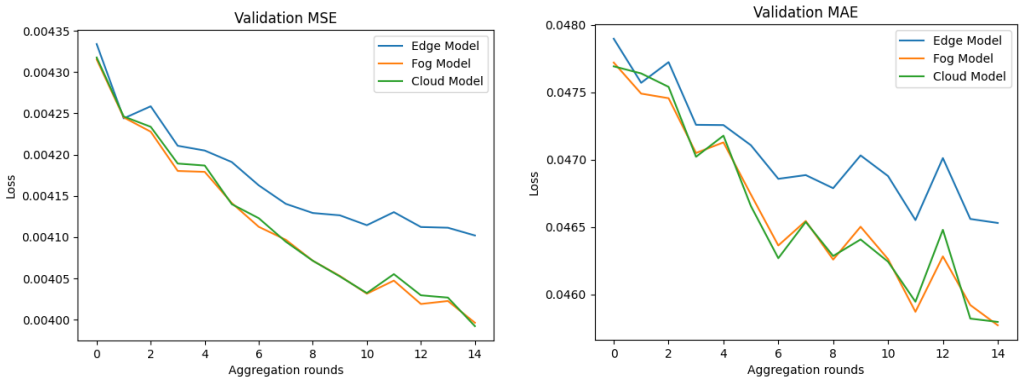
For both scenarios, a partitioned LSTM neural network using TensorFlow[3]/Keras[4] has been implemented to study the different temporal correlations present in the dataset. In this case, the neural network has been divided into three blocks of layers with early outputs [22] emulating the Edge-Fog-Cloud architecture that can resemble the vehicle-station-

[2]https://data.cityofpaloalto.org/dataviews/257812/electric-vehicle-charging-station-usage-july-2011-dec-2020/
[3]https://www.tensorflow.org/
[4]https://keras.io/



(a) MSE (b) MAE

Fig. 4: Incremental validation metrics in occupancy level sub-models



(a) MSE (b) MAE

Fig. 5: Incremental validation metrics in charging time sub-models

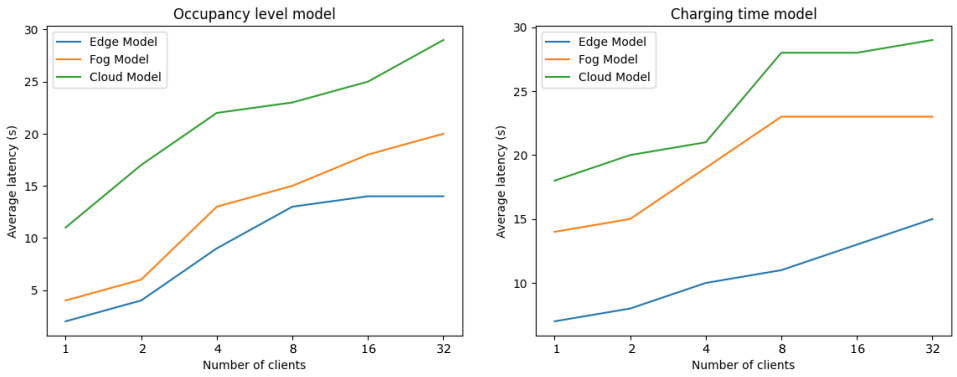
cloud use case, respectively. The parameters that have been defined to study the behavior of the sub-models were the Mean Squared Error (MSE) for the loss and the Mean Absolute Error (MAE) as a metric, which also indicates the difference between two continuous variables, both in the validation phase. For the dataset, 80% of the data has been used for training and the remaining 20% for validation. Lastly, the training configuration used was 256 as batch size, the Adam optimizer, a learning rate of 0.001, and a total of 50 epochs. For the incremental configuration, a maximum time window of one minute has been defined to wait for new data, and for the federated configuration, a total of fifteen aggregation rounds and the federated averaging strategy[5] (FedAvg) have been chosen to aggregate the weights. In addition, a latency-related test has been performed in the inference phase of both distributed models to see how they behave varying the number of clients making requests.

The tests have been carried out on our on-premises cluster of 7 servers. Each machine has an Intel(R) Xeon(R) Gold 6230R CPU with two NVIDIA(R) Tesla(R) V100 GPUs and 384 GB of RAM. Each one of the seven machines runs Kubernetes v1.21.6 and Docker 20.10.8 on top of Ubuntu 20.04.3 LTS. A Kubernetes master was deployed in one node, while the remaining six are Kubernetes workers. One of the machines runs a virtual machine with identical software characteristics, and this one is enabled as a Kubernetes master, while the rest of the machines are Kubernetes workers. The streaming data sent divides the training data set into equivalent parts (fifty thousand data each in this case), and they are sent one by one, progressively leaving intervals of 200 seconds between them to simulate an interrupted but continuous arrival of data in time. The client that sent the information and where the results were measured was a PC with an Intel(R) Core(TM) i9-10900K CPU and 64 GB of RAM.

A. *PEVs occupancy level model*

This test compares the validation performance of the first distributed model which is in charge of studying the total occupancy level of PEVs at the different charging stations.

[5]https://www.educative.io/answers/what-is-federated-averaging-fedavg



(a) Occupancy level model

(b) Charging time model

Fig. 6: Average inference latency response with different numbers of clients

On the one hand, Figure 4a shows the MSE loss progression. As can be seen, the three sub-models tend to have values very close to 0.0070, which does not show a high dispersion of errors. On the other hand, Figure 4b shows the progression of the MAE metric. According to this graph, the three sub-models tend to have values close to 0.053. This value indicates a MAE quite accurate, being in our problem an average difference of approximately 0.050 seconds in the total vehicle occupancy time at the charging stations.

B. PEVs charging time model

This second test tries to evaluate the validation behavior of the second distributed model in charge of studying the charging time of PEVs also in the different charging stations. For the case of the MSE, shown in Figure 5a, the progressions of the sub-models for this metric tend to have values very close to 0.0040, which does not show again a high dispersion of errors. For the case of MAE, shown in Figure 5b, the sub-models tend to have values close to 0.046. In this case, the test also obtains an approximate mean absolute difference of 0.050 seconds for the prediction of the total charging time of the PEVs, which can be considered quite accurate for our problem.

The results of the graphs of this second test show how the first sub-model in the distributed chain, corresponding to the Edge, obtains slightly higher values for both metrics. This is normal since, in the overall computation, we have fewer layers of the network, which makes it less precise.

C. Inference latency

This test aims to measure the average response latency of each of the two partitioned models as the number of concurrent clients and thus the number of requests increases. In this case, the configuration chosen for the test was a single Kafka broker, one partition per topic, and one replica per sub-model. Regarding the clients, tests were carried out with 1, 2, 4, 8, 16, and 32 clients. On the one hand, Figure 6a shows the results for the model predicting the total occupancy of the PEVs, while Figure 6b shows the results of the prediction model for the total charging time of these vehicles. It is important to highlight the relationship in response times that exists between the sub-models. For both models, the average response latency of the Edge sub-model is lower than the average latency of the Fog sub-model. At the same time, the average response latency of the Fog sub-model is lower than the average latency of the Cloud sub-model. This is achieved by the early outputs of both partitioned neural networks, which allow for faster predictions further down the distributed chain as the input data has to pass through fewer layers of the network. This last point is quite important and beneficial in the case of critical applications. In our case, this could be likened to deploying the Edge sub-model directly in a smart PEV, so that we get the fastest possible predictions directly in our vehicle. If the prediction is not sufficiently accurate, we would then feed the data to the next sub-model, which would be the Fog deployed in a charging station, giving a slightly slower result. Finally, if the Fog prediction is not accurate enough either, we would move the data to the third level, which would be the sub-model deployed in the Cloud, obtaining the slowest prediction of all.

VI. Conclusions and Future Work

The global transition toward sustainable mobility is marked by the increasing prominence of PEVs as an eco-friendly mode of transportation. However, this shift brings forth a substantial challenge in managing the wealth of data streams generated by PEVs, encompassing aspects such as vehicle performance, charging infrastructure, and user behavior.

Traditional methods of data analysis prove inadequate for the distributed and incremental nature of PEVs-generated data. Recognizing this, the integration of AI and ML techniques emerges as a necessity to effectively interpret and manage the voluminous, real-time, and dynamic data associated with PEVs. Unlike conventional vehicles, PEVs offer

a unique data landscape that requires advanced analytics for optimal performance. By harnessing AI capabilities, stakeholders can gain real-time insights that facilitate informed decision-making, leading to improvements in PEV technology and infrastructure.

With all this in mind, this paper proposes an open workspace for dealing with this typology of data coming from the PEVs sector. Our open-source framework Kafka-ML focuses on managing ML and AI pipelines through data streams in production scenarios. In this work, we have proposed an enhancement of the tool features to enable the joint functionality of three different ML paradigms within it, which are: 1) the partitioning and distribution of neural networks that allow for reduced latency through early outputs; 2) the incremental training of such algorithms that allows models to dynamically adapt to the data as it changes over time; and finally, 3) FL processes that preserve data security and privacy while forming a more robust overall model. In this way, we would be able to perform federated training deployments of distributed neural networks indefinitely in time, which would be very useful in the field of PEVs due to the characterization of their data and environment.

The next major step in this work will be a more thorough evaluation of the tool. Also, we want to consider the possibility of dynamically partitioning neural network models to be able to test different configurations of them flexibly. In addition, we want to add an encryption mechanism for the preservation of security and privacy of sensitive data related to PEVs within the FL process. Lastly, the implementation of models trained with the ML techniques studied in this project and their subsequent deployment in real PEVs within the automotive sector to check their performance.

Acknowledgment

This work is funded by the Spanish projects Grant TSI-063000-2021-116 ('5G+TACTILE_2: Digital vertical twins for B5G/6G networks') funded by MICIU/AEI/10.13039/501100011033/ and by 'European Union NextGenerationEU/PRTR', Grant TED2021-130167B ('GEDIER: Application of Digital Twins to more sustainable irrigated farms'), funded by MICIU/AEI/10.13039/501100011033/ and by 'European Union NextGenerationEU/PRTR', Grant CPP2021-009032 ('ZeroVision: Enabling Zero impact wastewater treatment through Computer Vision and Federated AI') funded by MICIU/AEI/10.13039/501100011033/ and by 'European Union NextGenerationEU/PRTR', and Grant PID2022-141705OB-C21 ('DiTaS: A framework for agnostic compositional and cognitive digital twin services') funded by MICIU/AEI/10.13039/501100011033/ and by 'FEDER'. This project has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement EVOLVE No 101086218. This work was also supported by funding from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] G. Fortino, C. Savaglio, G. Spezzano, and M. Zhou, "Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[2] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, 2018.

[3] T. J. Park, K. Kumatani, and D. Dimitriadis, "Tackling dynamics in federated incremental learning with variational embedding rehearsal," 2021.

[4] J. L. Lobo, J. Del Ser, A. Bifet, and N. Kasabov, "Spiking neural networks and online learning: An overview and perspectives," *Neural Networks*, 2020.

[5] N. Yang, D. Yuan, C. Z. Liu, Y. Deng, and W. Bao, "Fedil: Federated incremental learning from decentralized unlabeled data with convergence analysis," 2023.

[6] M. yuan Zhu, Z. Chen, K. fan Chen, N. Lv, and Y. Zhong, "Attention-based federated incremental learning for traffic classification in the internet of things," *Computer Communications*, 2022.

[7] H. Jahangir, H. Tayarani, A. Ahmadian, M. A. Golkar, J. Miret, M. Tayarani, and H. O. Gao, "Charging demand of plug-in electric vehicles: Forecasting travel behavior based on a novel rough artificial neural network approach," *Journal of Cleaner Production*, 2019.

[8] H. Jahangir and C. Konstantinou, "Plug-in electric vehicles demand modeling in smart grids: A deep learning-based approach: Wip abstract," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, 2021.

[9] Y. Lei, S. L. Wang, M. Zhong, M. Wang, and T. F. Ng, "A federated learning framework based on incremental weighting and diversity selection for internet of vehicles," *Electronics*, 2022.

[10] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, "Kafka-ml: Connecting the data stream with ml/ai frameworks," *Future Generation Computer Systems*, 2022.

[11] N. Yang, X. Chen, C. Z. Liu, D. Yuan, W. Bao, and L. Cui, "Fedmae: Federated self-supervised learning with one-block masked auto-encoder," 2023.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[13] D. Shenaj, M. Toldo, A. Rigon, and P. Zanuttigh, "Asynchronous federated continual learning," 2023.

[14] C. Liu, X. Qu, J. Wang, and J. Xiao, "Fedet: A communication-efficient federated class-incremental learning framework based on enhanced transformer," 2023.

[15] Y. Zhao, H. Liu, H. Li, P. Barnaghi, and H. Haddadi, "Semi-supervised federated learning for activity recognition," 2021.

[16] H. Lin, J. Lou, L. Xiong, and C. Shahabi, "Semifed: Semi-supervised federated learning with consistency and pseudo-labeling," 2021.

[17] S. Ajjaj, S. El Houssaini, M. Hain, and M.-A. El Houssaini, "Incremental online machine learning for detecting malicious nodes in vehicular communications using real-time monitoring," *Telecom*, 2023.

[18] A. Carnero, C. Martín, D. R. Torres, D. Garrido, M. Díaz, and B. Rubio, "Managing and deploying distributed and deep neural models through kafka-ml in the cloud-to-things continuum," *IEEE Access*, 2021.

[19] A. J. Chaves, C. Martín, and M. Díaz, "Towards flexible data stream collaboration: Federated learning in kafka-ml," *Internet of Things*, 2024.

[20] A. Carnero, C. Martín, G. Jeon, and M. Díaz, "Online learning and continuous model upgrading with data streams through the kafka-ml framework," *Available at SSRN 4681565*, 2024.

[21] Z. Teimoori and A. Yassine, "A review on intelligent energy management systems for future electric vehicle transportation," *Sustainability*, 2022.

[22] D. R. Torres, C. Martín, B. Rubio, and M. Díaz, "An open source framework based on kafka-ml for distributed dnn inference over the cloud-to-things continuum," *Journal of Systems Architecture*, 2021.