RoamML: Distributed Machine Learning at the Tactical Edge

Simon Dahdal*, Filippo Poltronieri*, Alessandro Gilli*, Mauro Tortonesi*, Roberto Fronteddu†, Raffaele Galliera†, Niranjan Suri†‡

* Distributed Systems Research Group, University of Ferrara, Ferrara, Italy
Email: {simon.dahdal, filippo.poltronieri, alessandro.gilli, mauro.tortonesi}@unife.it

† Florida Institute for Human & Machine Cognition (IHMC)
Email: {rfronteddu, rgalliera, nsuri}@ihmc.org

‡ US Army DEVCOM Army Research Laboratory (ARL), Adelphi, MD, USA
Email: niranjan.suri.civ@army.mil

***Abstract*—Machine learning is an effective methodology for enabling real-time data-driven decision-making in tactical scenarios, but its application in such scenarios raises many challenges due to data volume, unpredictable connectivity, and infrastructural challenges in edge environments. Furthermore, the need to perform training operations on remote powerful computing nodes might not be suited for tactical edge networks that often lack high bandwidth links, thus causing critical delays in assessing relevant information for decision-making. To overcome these challenges and enable machine learning at the tactical edge, this paper presents RoamML, a novel distributed continual learning approach tailored explicitly for the tactical edge. Built upon the foundational principle that "*moving the model is usually much cheaper than transferring extensive datasets*", RoamML seamlessly performs training operations by traversing network nodes, according to the *data gravity* concept. As RoamML encounters new data at each node, it continually trains on the encountered data to ensure that RoamML maintains an up-to-date and accurate model without transferring data to a centralized entity. Experimental results comparing RoamML with a baseline centralized machine learning solution show the potential of the proposed approach, which is capable of closely matching the accuracy of the baseline method.**

***Index Terms*—Tactical Networks, Distributed Machine Learning, Continual Learning, Data Gravity, Adaptive Machine Learning.**

I. Introduction

Data-driven situational awareness at the tactical edge is of paramount importance for contemporary military, safety, and emergency operations. It emphasizes making decisions backed by real-time data analysis, ensuring decisions are both timely and well-informed. With today's critical tactical scenarios, merely having information is insufficient; it must be efficiently processed, comprehended, and acted upon. Utilizing Data Analytics (DA) and Machine Learning (ML), it's feasible to sift through vast datasets quickly, identify patterns, and predict outcomes—imperative for crucial, time-sensitive decisions. In a prior paper [1], we examined the problem qualitatively, exploring data utilization in tactical operations, associated challenges, and the potential of DA and ML in refining the decision-making process.

However, implementing data-driven decision-making at the tactical edge presents many challenges. A pivotal concern is the enormous volume of data generated. While this volume promises rich insights, it also brings about significant logistical issues. Transferring such vast data for centralized processing and ML model training is both bandwidth-intensive and time-consuming, which could lead to decision-making delays. Moreover, tactical edge environments introduce unique challenges. Often situated in remote or disaster-impacted regions, unreliable networks, alongside the limited computational and storage capacities at the edge, hinder efficient data processing and model training [2], [3]. In disaster zones, compromised infrastructure can result in Denied, Degraded, Intermittent, and Limited (DDIL) networks [4]. Such conditions further disrupt data transfers, complicating the synchronization and dissemination of essential information. Achieving the full potential of data-driven situational awareness mandates both advanced analytics capabilities and strategic management of these challenges.

While certain machine learning frameworks propose solutions for training ML models in unpredictable and high-stakes situations, they are not without their complications. Take *Federated Learning* [5]–[9]. It allows ML models to train on large distributed datasets without centralizing the data. Sub-models are dispatched to nodes for training and then aggregated at a central server. Yet, federated learning's dependence on consistent network connections to all nodes of the tactical network and edge devices' limited computational power can be a challenge in areas with DDIL networks.

A possible alternative is *continual learning*: its primary goal is to incorporate new tasks while retaining knowledge of previous ones. Often referred to as sequential or incremental learning [10]–[12], this methodology updates the model iteratively with new input dataflows. Yet, in tactical contexts characterized by the challenges of edge environments, the typical centralized training associated with these approaches may prove impractical. The constraints of moving data and creating dataflows to a training node, combined with the complexities previously highlighted, make a traditional continual learning paradigm ill-suited for such scenarios.

Some approaches like *Gossip-based algorithms* were also presented in [13]. In such approaches, nodes within the network train the models on their local data without centralized coordination. These nodes exchange information, including

model parameters, gradients, or other relevant data, with a subset of randomly chosen neighboring nodes. Upon receiving this information, each node refines its model and persists in the gossip-driven iterative process. The primary goal is to build a global model that incorporates insights from distributed datasets across all nodes, iterating until a convergence is reached. While this decentralized approach bolsters resilience to failures, it also elevates the consumption of network resources. Specifically, in synchronous operations, nodes update models simultaneously and wait for data from their neighbors, leading to substantial communication overhead.

Finally, a non-traditional continual learning approach called *Random Walk learning* (RWL) [14], [15] is a decentralized learning approach where one node is activated at a time, updating the global model with its own data. This node then randomly chooses a neighboring node and passes the updated model to it. The process repeats until the model converges. While RWL reduces communication costs and power usage, it faces the main issue which is the extended training time and the unpredictability stemming from the total randomness in node selection, which can lead to inefficient or repetitive updates.

Therefore, there is the need for adaptable and efficient solutions specifically designed to overcome these challenges. To enable distributed machine learning at the tactical edge, this paper introduces **Roaming Machine Learning (RoamML)**, a distributed continual learning approach tailored for the tactical edge. Central to RoamML is the premise that *"moving the model is usually much cheaper than transferring extensive datasets"*. With data volumes expanding at the network's edge, transferring this data becomes complex, sometimes even prohibited by policies. Training models directly at data collection points can foster efficient resource utilization and faster insights.

RoamML is devised to navigate between network nodes, each serving as a distinct repository of data. Analogous to a traveler exploring different lands, RoamML views each node as a new opportunity to refine its knowledge, always on the hunt for fresh data sources to bolster its learning. As a safeguard, the RoamML agent periodically sends checkpoint models back to its destination nodes. These checkpoints serve as recovery points in case the agent encounters obstacles or becomes lost, ensuring the preservation of its learning journey up to that point.

Central to RoamML's operation is the Concept of **Data Gravity**. Instead of a fixed path, the model dynamically gravitates towards the most pertinent data sources within the network, ensuring it remains adaptive and continually evolves based on the available data's relevance. Large datasets exert a "gravitational pull" in the digital realm. RoamML agent, by design, is naturally drawn towards nodes with stronger data gravity, gauging the pull based on the size, relevance, and significance of the data within each node. Nodes with substantial or critical datasets invariably attract the model, guiding its training priorities. By harnessing data gravity, RoamML ensures a flexible and faster training process. Here, the importance of training on bigger datasets first proves critical. Larger datasets typically encompass a broader spectrum of information, thereby offering the model a richer, more diverse context to learn from. Also, having the possibility of training on vast datasets is not always available in the tactical and dynamic scenario, because of moving nodes, intermittent connections, and other complicating factors.

Our results validate the efficacy of the RoamML approach. We conducted a comparison between a baseline CNN model trained centrally on the full dataset and a RoamML model that underwent simulated continual training across various nodes, each containing sub-models derived from the primary dataset. Each subset created is a mini version of the original dataset, that is additionally imbalanced to simulate a realistic tactical scenario. The RoamML model's performance was very close to the baseline, underscoring the feasibility and the potential of this approach.

II. Roaming Machine Learning

RoamML is designed to navigate through various data access points within the tactical network and train the ML model. These data access points would allow the RoamML model to access data in geographical zones, clusters of nodes led by a gateway node, or individual nodes.

To simplify our preliminary approach to the challenge of distributed machine learning training at the tactical edge, we established a couple of fundamental assumptions. Firstly, The RoamML model begins its journey from a source node and completes its training cycle by returning to this same node. This source node is typically the HQ node, which either seeks updates on situational awareness in the tactical field or requires an ML model specifically trained for the prevailing scenario. Secondly, we operate under the presumption that a singular RoamML model roams within the tactical network, so it does not get duplicated in any way. Periodically, the RoamML agent dispatches checkpoint versions of the model back to the source node, ensuring continuous updates are relayed to the HQ. In this context, we must consider node mobility, introducing dynamic variables to the problem. Factors such as distance, latency, throughput, bandwidth, and new parameter such as Data Gravity Force (DGF) comes into play.

The goal of the RoamML approach is to train the machine learning model by determining the optimal visiting order of the nodes. This order is designed to allow the RoamML model to reduce uncertainty and thereby maximize its performance (e.g., accuracy). It aims to accomplish this by training on as much relevant data as possible. Simultaneously, RoamML strives to minimize the overall time taken to improve the model's performance, complete the tour of the nodes, and return to the source node.

In the following subsections, we are going to present the variable, parameters, and heuristics in play, to give it a better context and explain the approach thoroughly.

A. Parameters & Variables

The RoamML model acts as a moving agent, operating within the framework of the Stochastic Orienteering Problem

(SOP) [16]. The graph nodes are defined as a set of moving nodes N in a tactical environment. Let's denote each node i as a tuple of time-dependent coordinates $(x_i(t), y_i(t))$. The nodes move around in space and time, making the entire scenario dynamic. The distance between two nodes i and j at time t is represented as $d_{ij}(t)$, reflecting the physical distance between nodes at a specific time t. Each node possesses a time-dependent data mass value $M_i(t)$, which signifies the volume of data available for the RoamML model to train at each point in time. The data transfer rate, or throughput, between nodes i and j at a given time t is denoted as $T_{ij}(t)$. This is subject to the maximum capacity of data transfer rate, or bandwidth, between these nodes, represented as $B_{ij}(t)$. Additionally, we consider the total delay or latency experienced by the RoamML model as it travels from node i to node j, denoted as $L_{ij}(t)$ and a coefficient of the computational capacity of the node C_j. The model's size at time t is denoted as $Size_{model}(t)$, which measures the model's dimension in terms of kilobytes.

B. Data Gravity Heuristic

In orienteering problems, the optimal selection of the next hop is a critical aspect of achieving efficient and effective solutions. Heuristic policies, such as the Nearest Neighbour Heuristics (NNH), provide a robust framework to navigate through these traveling problems. The primary notion behind NNH is its selection criteria - it opts for the nearest yet unvisited node as the next hop. Although the method may not guarantee globally optimal solutions, its simplicity and quick computational turnaround time compensate for potential sub-optimality

For the RoamML approach, We present the *Data-Gravity Heuristic*, an approach similar to the NNH, but instead of making decisions based on the distance between nodes, it relies on the impact of the Data Gravity Force of each node. The concept of data gravity, depicted in Fig. 1 as gravitational waves around nodes, pertains to the force exerted by each node, influencing the movement of the RoamML model. If a node remains unvisited, it generates a "gravitational pull" that attracts the RoamML model towards it. DGF reflects the volume of data that each node possesses and it falls to zero once the RoamML model visits the node and trains on its data, meaning there's no remaining "pull" as the model has no further use for the data in that node, if not to be used as a stepping stone to reach other nodes.

The importance of initiating training with bigger datasets is paramount in the realm of machine learning. Such datasets are generally more comprehensive, encompassing a wide and varied spectrum of information for the model to learn and adapt. This bigger context is instrumental in facilitating superior generalization capabilities for the ML model, enabling it to establish a solid and robust foundational knowledge base. This is crucial, as it sets the stage for subsequent refinement of the model's understanding, and can be further honed using more specific and potentially smaller datasets. More importantly, the opportunity to train on large datasets is not consistently accessible, especially in dynamic environments like the tactical edge. This is due to factors such as the mobility of nodes, intermittent connectivity, and other unpredictable network conditions, which can all serve as significant impediments to the seamless utilization of extensive datasets in the training process.

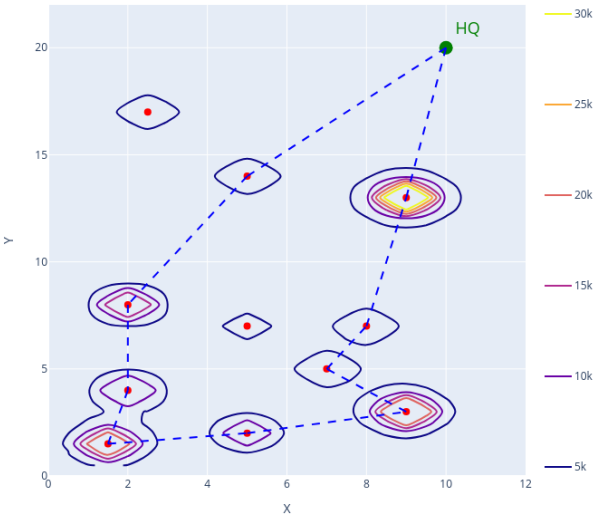


Fig. 1. The Data Gravity Force heuristic in choosing the path

The gravity force for a given node can be formulated as follows. The force that node i exerts on the RoamML model at time t is directly proportional to the mass of data of the node, denoted as $M_i(t)$. so the next hop of the node is determined by the node's data mass. The explained Heuristic is illustrated in Algorithm 1.

Algorithm 1 Data Gravity Heuristic

Initialize:
Set of unvisited nodes: $Nodes$
Current time: t

function CALCULATEDATAGRAVITY($node_i, t$)
    $M_i(t) \leftarrow$ Mass of data of node i at time t
    $DGF_i \leftarrow F_{gravity}(M_i(t))$ ▷ Init. Data Gravity Force
    return DGF_i

function SELECTNEXTHOP($nodes$)
    $selectedNode \leftarrow$ null
    $maxDGF \leftarrow 0$
    for $node$ in $nodes$ **do**
        $DGF \leftarrow$ CalculateDataGravity($node, t$)
        if $DGF > maxDGF$ **then**
            $maxDGF \leftarrow DGF$
            $selectedNode \leftarrow node$
    return $selectedNode$

The previously introduced heuristic can be extended to encompass a broader range of scenarios, resulting in a path

policy that relies on the overall attractiveness of each node. This enhancement is achieved by considering various factors, including data mass, the throughput of connections between nodes, the average latency of communication between nodes, and the computational capacity of individual nodes. As the RoamML model is required to undergo data training once, the Data Gravity Force exerted by a previously visited node i diminishes to zero. To represent this concept, an indicator variable I_i is assigned to each node i, where I_i is set to 1 if the node has been visited and 0 if it hasn't. With these considerations in mind, we proceed to define the gravity force, denoted as $F_{gravity}(i, j, t)$ as follows:

$$L_{ij}(t) = \frac{Size_{model}(t)}{B_{ij}(t)} \tag{1}$$

$$F_{\text{gravity}}(i, j, t) = \left((1 - I_j) \cdot M_j(t) + \frac{T_{ij}(t)}{L_{ij}(t)}) \right) \cdot C_j \tag{2}$$

In the next Section, we are going to present the simulated experimental evaluation of the RoamML approach.

III. Experimental Evaluation

To show the validity of the RoamML approach, we ran a simulation experiment utilizing realistically distributed datasets across a tactical environment to train an ML model. We selected the MNIST dataset for our experiments due to its prevalent use as a benchmark, as well as for reproducibility purposes. This dataset comprises handwritten digit images, with labels as integers from 0 to 9. The dataset was partitioned into training and testing subsets, containing 60,000 and 10,000 samples, respectively. We applied some data augmentation to enhance the model's ability to generalize from the training data and to add a level of difficulty to the learning process.

Our first step is to establish a performance baseline for our machine learning model. To achieve this, we trained numerous deep neural networks and ultimately opted for a convolutional neural network (CNN) architecture, which is shown in Fig. 2. This CNN was trained on the entire dataset in a centralized manner. The centralized approach to training is ideal, as it allows the model to learn from the entire dataset at once, which can lead to a more robust and well-generalized model. With this configuration, the CNN achieved 98.4% testing accuracy, successfully setting our intended performance baseline.

Then, to assess the RoamML approach, we divided the complete dataset into 20 distinct sub-datasets. Each sub-dataset simulated a node in our tactical network. Each sub-dataset had a deliberately imbalanced distribution of data samples, emphasizing one class out of the ten available (digits 0 to 9). Such a design imitates real tactical situations where nodes might specialize in collecting certain data types, leading to imbalanced distributions network-wide. In Fig. 3, we illustrate a snippet of the data division into subsets for each node of the tactical scenario. From the three out of twenty subsets shown in this figure, we can observe the data distribution across all classes of numbers. This visualization highlights the number of samples of each class and more clearly shows the imbalance in the subsets.

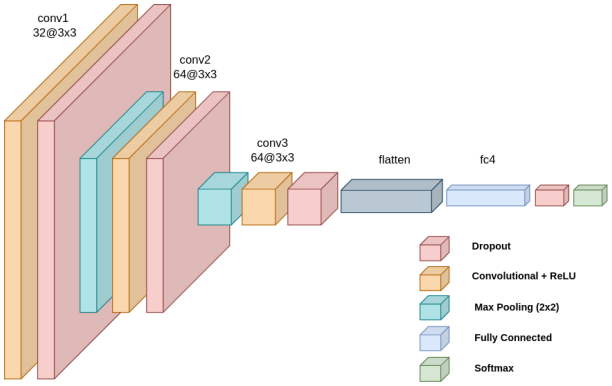


Fig. 2. CNN used in the experiments

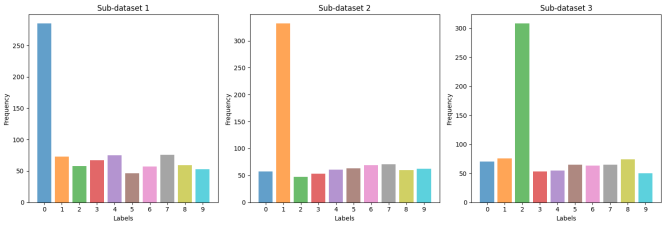


Fig. 3. Snippet of 3 out of the 20 subsets divided in the 20 nodes

Continual Learning was applied to these sub-datasets, sequenced by their sizes following the Data Gravity concept. This sequencing aimed to simulate the RoamML model's behavior, moving from one node to another during training. By training the model on these sequentially ordered sub-datasets, we sought to gauge the algorithm's adaptability and performance in real-world-like situations. The number of training epochs for each sub-dataset was randomly selected within the range of 20 to 30.

All experimental results are illustrated in Fig. 4. Please note that the green dashed vertical lines in all the sub-figures indicate the conclusion of each training phase on a particular subset and the beginning of the training on the next subsets. In other words, they indicate the end of each training phase on one node and the movement to another one.

Fig 4(a) shows the training and validation accuracy over time. In this sub-figure, we observe a global incremental training trend from the first subset of data to the last. This is reflected in the increasing trend of accuracy values, highlighting the CNN's ability to improve its performance as more data is incorporated through training phases. By the end of the training process, we observed a validation accuracy of 97%.

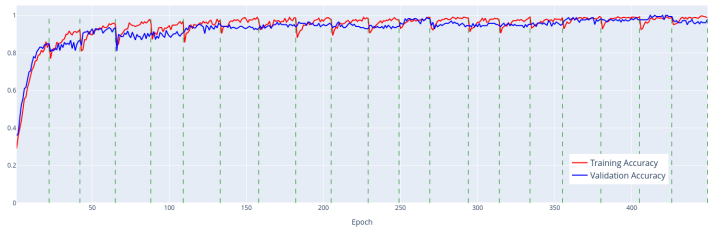
Fig 4(b) illustrates the training and validation loss over time. Similar to the accuracy, this sub-figure indicates a consistent trend throughout the training phases. As the training progresses from one subset of data to the next, we can see that the loss generally decreases, signifying that the model is effectively learning and optimizing its weights to minimize the error between its predictions and actual labels. By the end of the training process, we observed a validation loss of 0.028.

To delve deeper into our analysis, we evaluated the uncertainty associated with the trained model's predictions. More specifically, a good uncertainty estimate of an ML model refers to the degree of confidence that the model has in its predictions. This is crucial for understanding how reliable the model's predictions are under various circumstances. While Deep Neural Networks (DNNs) are potent predictors, they often lack inherent uncertainty metrics. Accurately gauging this uncertainty is vital, particularly when misjudged predictions can be catastrophic. Bayesian Neural Networks (BNNs) are common for deriving uncertainty from deep models but are computationally taxing. Contrastingly, *Monte Carlo (MC) Dropout* [17] offers a simpler solution. Initially, a regularization technique to mitigate overfitting, dropout nullifies a fraction of input units randomly during each update in training. MC Dropout interprets dropout as an approximate Bayesian inference. Hence, for uncertainty quantification, we retained dropout during inference. Conducting 100 forward passes with active dropout (illustrated in Fig. 2), resulted in varied predictions, providing an estimated uncertainty measure. So, for each given input sample of the test set, the spread of the 100 predictions is a measure of the model's uncertainty. The standard deviation across these predictions is used as an approximation for the model's uncertainty: the higher the standard deviation, the greater the model's uncertainty about that particular prediction.

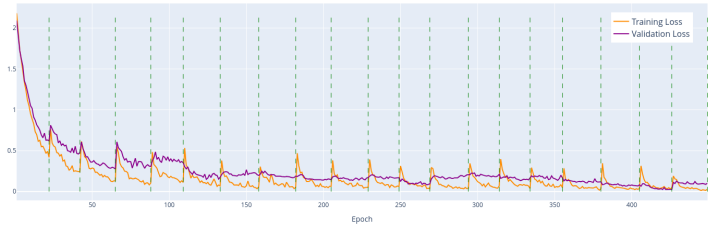
In Fig 4(c) we can observe the uncertainty of classification of each class of numbers of the test set, we can also observe a global tendency of all numbers uncertainty to decrease post the ending of the entire training phase.

Lastly, we summarize the results in a final Fig 4(d), a global measure of the model's test accuracy and uncertainty across the test set. The shown mean uncertainty is calculated by taking the mean of all standard deviations of all classes (shown in Fig. 4(c)) after each subset training phase. It is known that the model's uncertainty, also known as epistemic uncertainty, can be reduced if we train our model on a larger dataset containing richer information. However, in our case, the mean uncertainty results indicate not only that training the model on a larger dataset reduces uncertainty, but also that continually feeding data to train the model reduces the model's uncertainty by the end of the training sessions. These conclusions can be observed at the level of specific class numbers in Fig. 4(c), and more generally at the entire model's level by observing the mean uncertainty estimate.

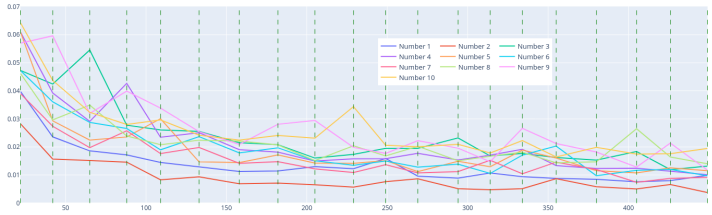
In conclusion, we coupled the uncertainty estimation with the testing accuracy. We observed that the final accuracy on the test set, after each subset training phase, reached 97.7% by the conclusion of the entire training phase. The testing accuracy of the RoamML approach achieves performance that is very close to that of our centralized baseline model, thereby proving the viability of the approach.



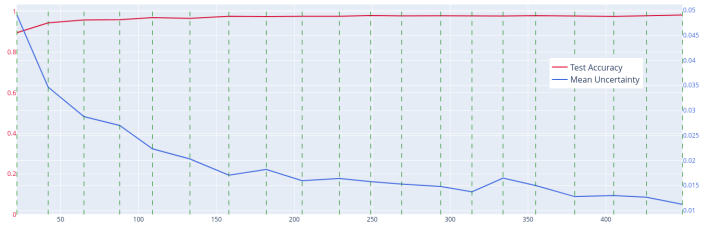
(a) Training accuracy



(b) Training loss



(c) Model uncertainty



(d) Testing accuracy and uncertainty

Fig. 4. RoamML training & evaluation: (a) Accuracy, (b) Loss (c) Uncertainty of all classes (d) Test accuracy and Mean of model uncertainty; The green dashed vertical lines indicate the conclusion of each training phase on a particular subset

IV. Future work

While the preliminary results of the RoamML model are promising, further development and refinement in several pivotal areas are essential. A critical focus for future research is the enhancement of privacy preservation within the RoamML framework. Despite its ability to train directly on nodes, potential privacy issues may arise if sensitive data is accessed. Incorporating techniques such as differential privacy could allow the model to learn from data without exposing sensitive information, bolstering RoamML's viability where data privacy is paramount.

By design, RoamML is poised to navigate networks comprising multiple domains, each potentially representing different organizations or geographical locations with unique data. Merging RoamML's cross-domain capabilities with the intra-domain training efficiency of federated learning could yield

a major advancement, enabling efficient, privacy-preserving, and dynamic learning in multi-domain environments.

Furthermore, further research is needed to understand the impact of heterogeneous data distributions, such as non-iid and iid data, on both continual learning [18] and on the concept of data gravity. Finally, empirical evaluation of RoamML in real-world or emulated scenarios is a critical step towards practical application, offering some insights into its performance and potential impact across various domains. We plan to create an edge-deployable platform, influenced by relevant concepts from the industrial domain such as the work in [19]. The platform will not only facilitate the training of ML models over the tactical edge but it will also support the deployment of ML-powered applications.

V. Conclusions

In this paper, we introduced the RoamML approach, highlighting its potential to facilitate the training of machine learning models at the tactical edge. We detailed the fundamental aspects of this approach, emphasizing the significance of the "data gravity" concept as a guiding system for the RoamML model. Our experimental evaluation demonstrated that the RoamML approach can achieve performance comparable to centralized methods.

A RoamML model is capable of autonomously navigating this landscape, training robust and adaptable models that can learn from a diverse range of data sources and adapt to fluctuating network conditions. This innovative approach has the potential to revolutionize our understanding and training of machine learning in dynamic, high-stakes, and data-rich environments such as the tactical edge.

Acknowledgements

This work has been partially supported by the Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 - Next Generation EU (NGEU).

References

[1] S. Dahdal, F. Poltronieri, M. Tortonesi, C. Stefanelli, and N. Suri, "A data mesh approach for enabling data-centric applications at the tactical edge," in *2023 International Conference on Military Communications and Information Systems (ICMCIS)*, 2023, pp. 1–9.

[2] J. Perazzone, M. Dwyer, K. Chan, C. Anderson, and S. Brown, "Enabling machine learning on resource-constrained tactical networks," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022, pp. 932–937.

[3] M. Fogli, G. Pingen, T. Kudla, S. Webb, N. Suri, and H. Bastiaansen, "Towards a cots-enabled federated cloud architecture for adaptive c2 in coalition tactical operations: A performance analysis of kubernetes," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, 2021, pp. 225–230.

[4] L. Campioni, F. Poltronieri, C. Stefanelli, N. Suri, M. Tortonesi, and K. Wrona, "Enabling civil–military collaboration for disaster relief operations in smart city environments," *Future Generation Computer Systems*, vol. 139, pp. 181–195, 2023.

[5] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2017.

[6] S. M. Mathews and S. A. Assefa, "Federated learning: Balancing the thin line between data intelligence and privacy," 2022.

[7] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[8] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[10] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," 2023.

[11] S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. O. Stanley, J. Clune, and N. Cheney, "Learning to continually learn," 2020.

[12] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2022.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[14] A. B. Ardic, H. Seferoglu, S. El Rouayheb, and E. Koyuncu, "Random walking snakes for decentralized learning at edge networks," in *2023 IEEE 29th International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2023, pp. 1–6.

[15] G. Ayache, V. Dassari, and S. E. Rouayheb, "Walk for learning: A random walk approach for federated learning from heterogeneous data," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, p. 929 – 940, 2023.

[16] T. C. Thayer and S. Carpin, "An adaptive method for the stochastic orienteering problem," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4185–4192, 2021.

[17] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," 2016.

[18] M. F. Criado, F. E. Casado, R. Iglesias, C. V. Regueiro, and S. Barro, "Non-iid data and continual learning processes in federated learning: A long road ahead," *Information Fusion*, vol. 88, pp. 263–280, 2022.

[19] R. Venanzi, S. Dahdal, M. Solimando, L. Campioni, A. Cavalucci, M. Govoni, M. Tortonesi, L. Foschini, L. Attana, M. Tellarini, and C. Stefanelli, "Enabling adaptive analytics at the edge with the bi-rex big data platform," *Computers in Industry*, vol. 147, p. 103876, 2023.