Dynamic Federated Learning with Mobility-Aware Multi-Agent Collaboration in Urban IoT Networks

Santosh Lavate[1]
[1]Assistant Professor, AISSMS's College of Engineering, Pune, Maharashtra, India
shlavate@aissmscoe.com
lavate.santosh@gmail.com

Ravindra Moje[2]
[2]Assistant Proffesor, PDEA's College of Engineering, Manjari, Pune, Maharashtra, India
ravindra.moje@gmail.com

Shakir Khan[3]
[3]College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh Saudi Arabia.
University Centre for Research and Development, Chandigarh University, Mohali 140413, India;
sgkhancs@gmail.com
sgkhan@imamu.edu.sa

Abstract: The quick growth of Internet of Things (IoT) networks in cities has made it possible for smart cities to get real-time data-driven information that has never been seen before. However, it is still very hard to make machine learning work well and protect privacy in settings that are so changing and mobile. Dynamic Federated Learning with Mobility-Aware Multi-Agent Collaboration (DFL-MAC) is a new approach that is proposed in this study to improve joint intelligence in urban IoT networks. Unlike other federated learning methods that rely on fixed clients and centralised coordination, our framework includes mobility prediction and agent-based interaction tools that allow it to respond to topologies that change quickly. Each IoT node is modelled as an intelligent agent that can make decisions on its own and communicate with other nodes. This makes local model training and communication more efficient by taking into account movement patterns and network stability. The DFL-MAC system uses mobility-aware clustering to create brief federated sections. This cuts down on communication costs and delays while keeping the model convergent. The process of agents working together is controlled by a manager that uses reinforcement learning to balance the importance of data, connection, and energy use. We did a lot of tests on urban movement datasets and found that our method is much better than standard federated learning models in terms of accuracy (up to 12% better), convergence speed, and ability to keep working even when nodes drop out. In addition, our method protects data privacy by not sending raw data and automatically adjusts to new urban movement situations. This work opens the way for shared learning systems that are scalable and adaptable, perfect for how smart cities are changing. This will allow strong intelligence to be used in many mobility-aware IoT apps, like self-driving cars, traffic tracking, and environmental sensors.

Keywords: Federated Learning, Urban IoT, Mobility-Aware Systems, Multi-Agent Collaboration, Smart City Networks

I. Introduction

Smart cities are ecosystems where data from Internet of Things (IoT) devices is used to improve public services, transportation systems, environmental monitoring, and citizen safety. This idea came

about because of the large number of smart devices, sensors, and connected infrastructure in cities. In these kinds of situations, huge amounts of decentralised data are constantly being created, which opens up a huge number of possibilities for smart decision-making. But traditional centralised machine learning models don't work well in smart cities because of privacy issues, low bandwidth, high delay, and the fact that mobile IoT nodes are always changing. Federated Learning (FL) is a new and hopeful decentralised approach that lets devices in different areas work together to build a global model that they all share, without having to share raw data [1]. However, current FL models often assume that nodes will stay connected, never change, and that management will be centralised. This makes them unsuitable for the highly mobile and changing settings that are common in urban IoT networks. Researchers who tried to use FL in smart city systems in the past ran into a number of major problems. Devices moving around randomly, losing connection and data being spread out unevenly caused problems like wasteful communication, slow convergence, and worsened model performance. Also, centralised FL distributors created single points of failure and made it harder to grow [2]. These problems made it clear that we needed a stronger, more flexible, and more joint way to learn that would work well when there were limits on resources and movement while still protecting privacy.A new idea is put forward in this paper: Dynamic Federated Learning with Mobility-Aware Multi-Agent Collaboration (DFL-MAC). The goal is to create a decentralised, flexible, and resilient learning system that can change based on how IoT nodes in cities move. This study presents smart agents that decide when and with who to work together based on their surroundings, such as cars in traffic, mobile sensors on public transit, and devices worn by pedestrians. In traditional FL [3] setups, aggregation is controlled by a central server. DFL-MAC, on the other hand, uses a distributed multi-agent system where nodes talk to each other directly and form shared groups based on how similar their movement is and how reliable their communication is.

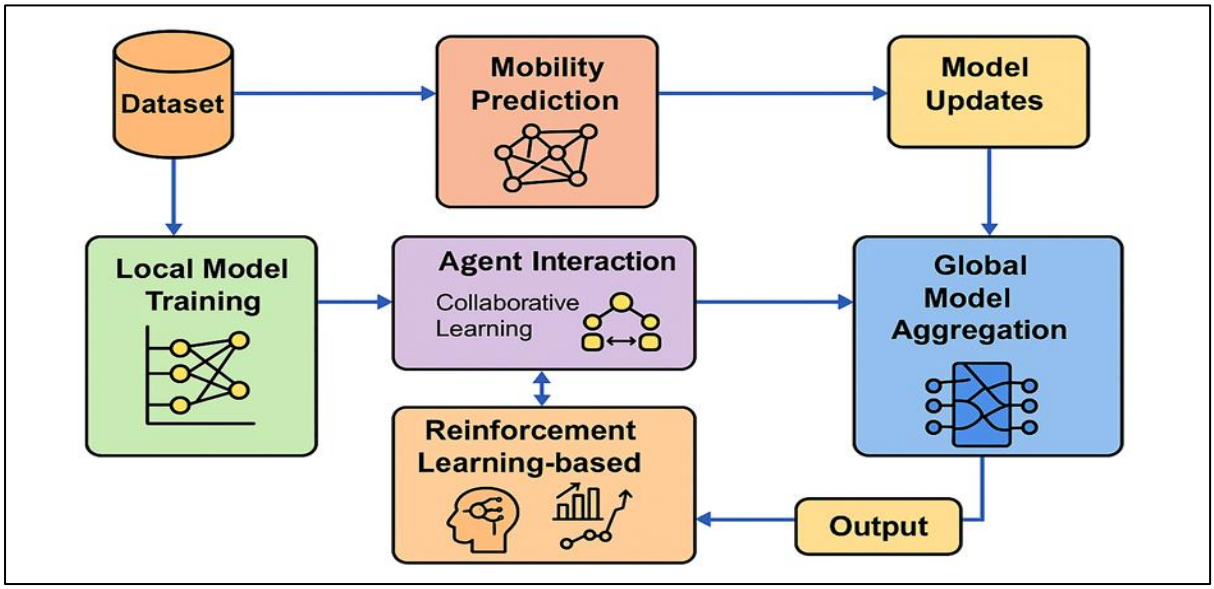

Figure 1: Overview representing the Federated Learning with Mobility-Aware Multi-Agent in IoT Networks

The general structure of Dynamic Federated Learning with Mobility-Aware Multi-Agent Collaboration in urban IoT networks is shown in Figure 1. It shows how movement prediction, local training, agent collaboration, reinforcement learning, and privacy rules work together to make sure that learning is decentralised, flexible, and safe in settings that change over time.This work is being done because smart towns need real-time information that doesn't invade people's privacy without depending on weak centralised systems. For example, in traffic tracking apps, mobile agents such as connected cars and units on the side of the road must work together to find accidents or traffic jams without any delays at the central level. For public health monitoring, smart tech that moves between city zones should share information locally to find trends without putting personal information at risk. To meet these goals, DFL-MAC uses a motion prediction module that lets nodes guess where they will be in the future and group together based on that information. A strategy based on reinforcement learning tells agents how to balance the usefulness of training, the cost of energy, and the chances to work together. This flexible and decentralised system makes sure that learning keeps working well even when the structure and resources change a lot. The suggested approach not only improves model accuracy and convergence, but it also cuts down on connectivity costs and energy use by a large amount. DFL-MAC changes the standard federated learning method into a dynamic, agent-driven system that can power smart, privacy-aware services in next-generation smart cities by seeing urban movement as an advantage instead of a problem.

II. Related Work

Federated Learning (FL) has been used in smart cities and the Internet of Things (IoT) more and more recently, thanks to the need for decentralised learning systems that protect privacy. Traditional FL designs work well in steady situations, but they have a hard time in urban IoT environments that are always changing and have a lot of people moving around, unreliable connections, and different kinds of resources [4]. Early FL implementations depended on centralised brokers to keep models in sync and assumed that clients would always be involved. In mobile and ad hoc urban settings, where IoT devices like cars, drones, and smartphones move around a lot and may lose connection with the network [5, 6], these ideas don't hold true.A number of studies have looked into decentralised and peer-to-peer shared learning as a way to get around the problems with central planning. By adding edge nodes or secondary organisers, approaches like hierarchical FL and cluster-based FL tried to make the system less dependent on central servers [7, 8]. But these systems often still relied on fixed structures or static grouping, which meant they weren't flexible enough for mobile settings. Also, most cluster-based methods didn't take real-time movement or communication dynamics into account, which made them less useful and hurt model performance in IoT networks that change quickly [9, 10]. In other papers, communication-efficient FL methods like model compression, update sparsification, and asynchronous training [11, 12] were proposed to cut down on bandwidth use. These methods make it easier to add more nodes, but they don't automatically fix the problems that come up when nodes move around and link quality changes. Also, the methods that were already used that included schedule and client selection strategies were often limited by set rules or didn't work well in situations with a lot of change [13].

Some research has been done on the idea of mobility-aware FL, mostly in the areas of vehicle networks and mobile edge computing. For example, some systems added motion forecast tools to help choose the best clients or the best time for aggregation [14]. However, these systems were

usually only partially centralised, and workers did not work together to share information. Also, they didn't use fully independent multi-agent cooperation to improve the accuracy of learning when there was doubt [15]. A few new studies have looked at how reinforcement learning could help FL coordination, but they mostly looked at server-side scheduling and not decentralised, agent-based decision-making [16]. Multi-agent systems (MAS) have shown promise in areas of remote computing. They let agents work together, discuss, and adjust to changes in their surroundings. Still, not much is known about how to combine MAS and FL in mobile IoT settings. Most solutions out there don't fully use the power of distributed agents to organise themselves, work together temporarily, and make learning decisions that take movement into account [17].

There is a lot of study on federated learning in IoT and edge settings, but not much is known about how to make FL designs that are dynamic, mobile, and fully decentralised. The ways we use now aren't good enough to handle quick changes in the structure, peer-to-peer decision-making, and context-driven teamwork [18]. This study fills in these gaps by presenting a dynamic federated learning framework that combines multi-agent systems and motion prediction to make it possible for smart city IoT networks to work together securely. The suggested system, DFL-MAC, improves on what's already available by letting mobile agents create shared groups on their own, make the best use of local model changes, and handle communication in a way that adapts to the situation. This makes up for some of the main problems with earlier FL models when used for urban mobility.

Table 1: Related Work summary

Approach	**Key Factor**	**Finding**	**Limitation**	**Scope**
Standard Federated Learning	Centralized aggregation	High privacy, but poor in mobility	Fails in dynamic topology	Static IoT systems
Hierarchical FL	Edge-level hierarchy	Better scalability	Still semi-centralized	Edge-based smart grids
Cluster-based FL	Predefined clusters	Improved local efficiency	Lacks mobility handling	Localized health networks
Asynchronous FL	Latency-tolerant updates	Handles stragglers	Inconsistent model accuracy	Remote sensor networks
Model Compression in FL	Reduced communication	Lower bandwidth use	Loss of detail in updates	IoT in rural regions
Sparse Update FL	Update sparsity	Faster training convergence	Susceptible to dropouts	Battery-operated IoT
Client Selection Policies	Resource-aware selection	Improves efficiency	Not robust in mobility	Edge-cloud cooperation
Mobility-Aware FL (Vehicular)	Mobility prediction	Adaptation to mobile nodes	Limited collaboration scope	Vehicular smart cities
Edge Aggregation-Based FL	Local model fusion	Reduces server load	Edge failure affects learning	Fog computing networks
RL-Guided Aggregation	Server-side RL controller	Improves global convergence	No agent autonomy	Distributed healthcare

P2P Federated Collaboration	Peer-based data sharing	Flexible participation	Lacks policy intelligence	Citizen safety apps
Multi-Agent Reinforcement FL	Agent cooperation	Dynamic collaboration	Complex coordination	Dynamic smart cities
Contract-Theoretic FL	Incentive mechanism	Motivates participation	Needs trusted infrastructure	Economic IoT clusters
Adaptive FL with Local Updates	Energy-aware local training	Minimizes energy drain	Slower in fast topologies	Low-energy smart homes

III. Proposed Framework: DFL-MAC Architecture

A. Overview of the System Components

The DFL-MAC system is made to deal with the problems that come up naturally in IoT settings that are very mobile and change quickly, workflow illustration in figure 2. The system is made up of six main parts, each of which is designed to support a decentralised, adaptable, and shared learning process. These parts are: a mobility prediction module that guesses how devices will move; local model training units built into every IoT node; a multi-agent collaboration layer that lets nodes interact with each other in real time; a reinforcement learning controller that makes decisions; and finally, a communication protocol with privacy-protecting features built in. This flexible but linked design keeps the system strong even when nodes move, bandwidth is limited, or energy levels change. It also keeps model performance high and data protection safe.

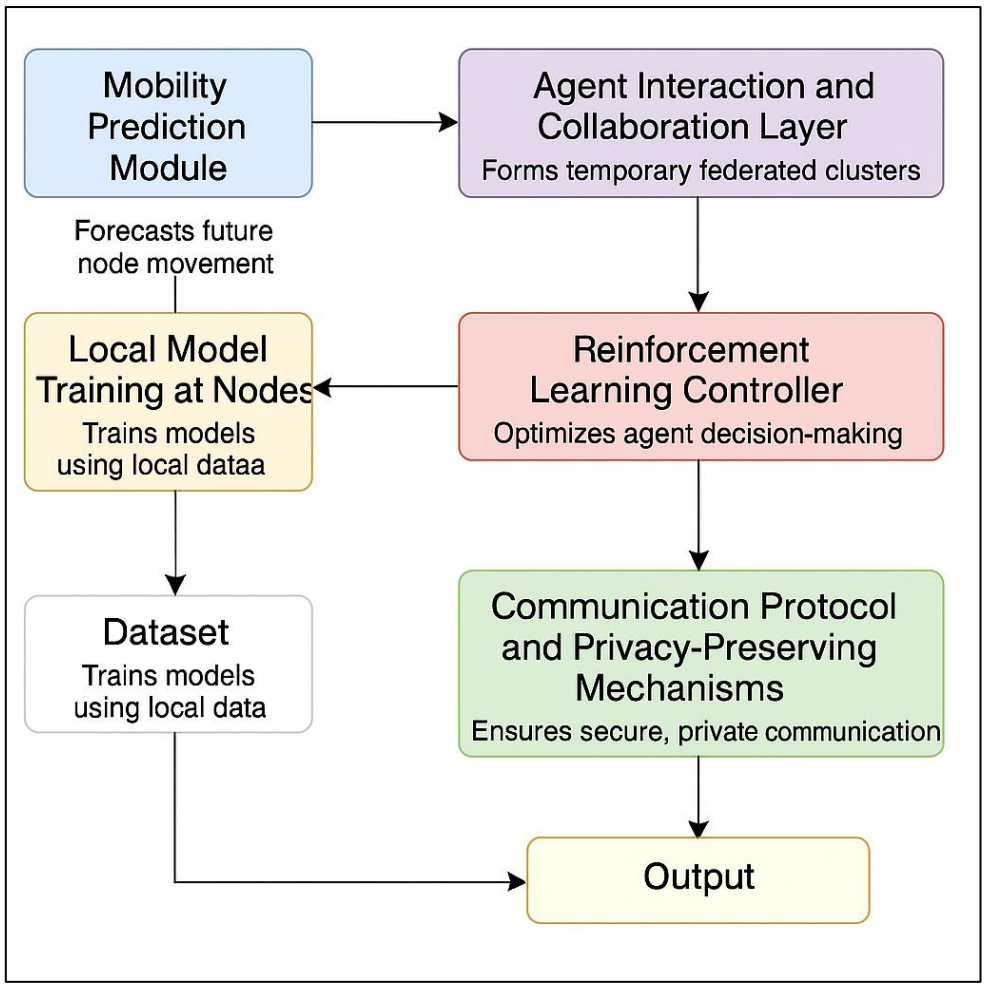

Figure 2: Block diagram representing the DFL-MAC working pipeline

B. Mobility Prediction Module

Mobility prediction is a key part of making sure that dynamic shared learning works well with shifting network topologies. In DFL-MAC, every IoT node has a mobility estimator that looks at past GPS and temporal-spatial data to guess how people will move in the future. Based on these estimates, the framework can set up brief federated groups where nodes will stay close to each other or on stable communication lines. This makes it less likely that people will stop talking to each other, and it also makes joint learning more consistent. The movement estimator uses lightweight neural network models or time-series analysis methods to make sure that devices with limited resources don't have to do too much extra work.

C. Local Model Training at Nodes

As part of the DFL-MAC environment, each IoT device trains its own model using its own data. This keeps the raw data on the device that created it. This design keeps users' information private and follows rules about data ownership. Most of the time, the local models are simple deep learning models or decision tree-based methods that work with the hardware of the node. Training can happen on a regular basis or when certain conditions are met, like when enough data is collected or

when the network is available. When the model is trained, it only shares model parameters (like gradients or weights) with nodes that work together or goes to the aggregation step. This decentralised training method allows for customisation, lowers connection bandwidth, and allows for different data patterns between nodes.

D. Agent Interaction and Collaboration Layer

Instead of a central broker, DFL-MAC has a layer of cooperation between many agents, and each IoT unit is an intelligent agent. These bots create shared groups on the fly based on how mobile they think each other will be and what data is relevant. The agent interaction protocol lets peers talk to each other directly and allows for decentralised parameter average within each cluster. To choose the best partner, agents also share information about the local model's trust, the device's energy level, and the security of the link. This way of working together not only makes learning more efficient, but it also makes people more resistant to network problems, so shared learning can continue even in cities that change quickly.

E. Reinforcement Learning Controller

Each DFL-MAC robot has a reinforcement learning-based driver built in to help them make even better choices. This controller learns the best way to choose peers, choose when to start training, and choose how much data or model knowledge to share. The environment's states are set by network measures like delay, signal strength, movement context, and energy use, and actions are things like training intervals and conversation decisions. The payment function finds a good mix between model accuracy, resource efficiency, and the dependability of cooperation. The manager makes sure that agents make choices that are appropriate for the situation by constantly changing its rules based on feedback from the environment. This encourages stable and effective long-term learning.

F. Protocols for communication and ways to protect privacy

Because urban data like location and health measures can be sensitive, DFL-MAC uses a strong communication system with privacy-protecting features. Differential privacy and homomorphic encryption are two safe gathering methods that the system uses to make sure that shared model changes can't be used to get personal information. The protocol also uses flexible transfer timing, which changes based on the availability of the network and the energy level of the device, to keep interruptions to a minimum. Lightweight digital signatures are used to verify peer-to-peer trades and stop fraud or harmful model injection. Together, these steps make sure that the DFL-MAC system protects privacy and security while keeping communication fast and easy to scale.

IV. Reinforcement Learning for Agent Coordination

A. Definition of states, actions, and rewards

Reinforcement Learning (RL) is a key part of the DFL-MAC system because it helps each agent (IoT node) make smart, situation-aware choices when things change. The state space describes the world in which each agent works. It contains information like the agent's current movement status (speed,

direction, and rate of position change), its energy levels, its past connection strength, and the accuracy of its local model. These states give a full picture of both the operating situation and the limitations of the surroundings. Based on these observations, agents decide what to do, such as starting to work together with certain peers, delaying or starting model changes, changing how much they talk to each other, or leaving the collaboration loop briefly to save resources. The payment function is meant to show long-term benefits and punish short-term mistakes. This keeps the balance between improving accuracy, using resources, and keeping participants stable. Positive rewards are given for good cooperation that improve global model convergence. On the other hand, negative rewards are given for actions that cause links to drop, too much energy use, or model shift.

DFL-MAC uses policy optimisation methods like Deep Q-Networks (DQN) and Proximal Policy Optimisation (PPO) to help animals behave in the best way possible. These algorithms let agents learn the best policies by changing their decision-making processes over and over again based on how earlier acts turned out. DQN uses a neural network to help agents guess the value of each action in a given state. PPO, on the other hand, improves stability and convergence by reducing big policy changes. Over time, these algorithms help agents adapt their actions to different network conditions and movement situations. This makes it easier for them to work together and train models in IoT settings that are always changing. The reinforcement learning module also allows flexible cooperation rules that change in real time as node movement and resource availability change. For instance, nodes that move around a lot might prefer short-term, chance partnerships with peers that are close by, while nodes that stay put or move slowly might commit to longer training rounds with stable partners. In the same way, devices that are limited by energy may limit the number of contacts or choose only high-value ones. This saves resources without greatly lowering total learning performance. In cities with lots of IoT nodes and uncertain moving patterns, these flexible rules are very important. DFL-MAC makes sure that the system stays strong, scalable, and flexible across a wide range of smart city applications by regularly changing teamwork strategies based on feedback from the environment and global learning trends.

B. Optimization and learning algorithms used

1. Deep Q-Networks (DQN)

To deal with large state spaces, Deep Q-Networks (DQN) is a value-based reinforcement learning method that blends Q-learning with deep neural networks. The ideal action-value function, Q(s, a), tells us about the predicted benefit of action an in state “s”. DQN gets pretty close to this function. It does this by storing and randomly selecting past events during training. This breaks down correlations and makes learning more stable. It also uses a target network that is changed on a regular basis to provide stable learning goals and lower fluctuations. In situations with clear action areas, DQN works well. It has been used successfully in video games, robots, and making decisions in mobile IoT systems.

Deep Q-Networks (DQN) –Model

Step 1: Initialize Q-network with weights θ and target Q-network with weights θ^{-}

Step 2: Experience tuple $\left(s_t, a_t, r_t, s_{\{t+1\}}\right)$ is stored in replay buffer D

Step 3: Compute target Q-value using Bellman equation:

$$y_t = r_t + \gamma * \max'_a Q\left(s_{\{t+1\}}, a'; \theta^-\right)$$

Step 4: Define the loss function (Mean Squared Error):

$$L(\theta) = E_{\{(s,a,r,s')\}}\left[\left(y_t - Q(s_t,a_t;\theta)\right)^2\right]$$

Step 5: Gradient descent update to minimize loss:

$$\theta \leftarrow \theta - \alpha * \nabla_\theta L(\theta)$$

Step 6: Periodically update target network:

$$\theta^- \leftarrow \theta$$

2. Proximal Policy Optimization (PPO)

Proximal Policy Optimisation (PPO) is a policy-gradient reinforcement learning method that is meant to make training more stable and effective. PPO improves a cut replacement objective function that limits the size of policy updates. This stops changes that are too big and could throw learning off. As long as it stays balanced between exploring and exploiting, it directly learns a parameterised strategy by maximising expected total benefits. PPO can work in both discrete and continuous action areas, which makes it useful for complicated settings like multi-agent systems that are aware of movement. As cities change, things like changing connections and energy limits in IoT networks can be quickly adjusted by agents thanks to its sample speed and stability. In many real-world RL situations, PPO has become the best way.

Proximal Policy Optimization (PPO) –Model

Step 1: Define the clipped surrogate objective:

$$L^{\{CLIP\}}(\theta) = E_t\left[\min\left(r_{t(\theta)} * A_t, clip\left(r_{t(\theta)}, 1-\varepsilon, 1+\varepsilon\right) * A_t\right)\right]$$

where $r_{t(\theta)} = \frac{\pi_\theta(a_t|s_t)}{\pi_{\{\theta_{old}\}}(a_t|s_t)}$

Step 2: Compute advantage estimate A_t using Generalized Advantage Estimation (GAE):

$$A_t = \delta_t + (\gamma\lambda)\delta_{\{t+1\}} + \ldots + (\gamma\lambda)^{\{T-t+1\}\delta}_{\{T-1\}}$$

where $\delta_t = r_t + \gamma V\left(s_{\{t+1\}}\right) - V(s_t)$

Step 3: Define value function loss:

$$L^{\{VF\}}(\theta) = \left(V_{\theta(s_t)} - V_t^{\{target\}}\right)^2$$

Step 4: Define total PPO loss:

$$L_{\{PPO\}(\theta)} = E_{t[L^{\{CLIP\}(\theta)} - c_1 * L^{\{VF\}(\theta)} + c_2 * S[\pi_\theta](s_t)]}$$

where S[π_θ] is entropy bonus for exploration

Step 5: Optimize total objective with gradient ascent:

$$\theta \leftarrow \theta + \alpha * \nabla_\theta L_{\{PPO\}(\theta)}$$

Step 6: Repeat policy updates for K epochs over minibatches of collected trajectories

V. Results and Discussion

Table 2 shows how well the suggested Dynamic Federated Learning with Mobility-Aware Multi-Agent Collaboration (DFL-MAC) system works in three different types of mobility scenarios: low, medium, and high. The results make it clear that the system can adapt to IoT settings that are changing and moving around. The model works very well in low motion situations, where nodes are fixed and stay close to each other for longer periods of time. It gets an F1-score of 90.8% and an accuracy of 91.2%. After 38 conversation rounds, the convergence time is pretty quick, and the communication overhead stays low at 12.1 MB. This is because the bots work together nonstop.As motion rises to a middle level, which means that nodes move around more often and are connected and disconnected more often, the model's performance slightly decreases. The accuracy drops to 89.7% and the F1-score to 88.9%. This is because clustering is becoming more dynamic and contact is sometimes lost. The convergence time goes up a little to 42 rounds, and the communication cost goes up to 14.6 MB because agents have to reconnect to each other more often and do more work to keep learning in sync.

Table 2: Model performance across dynamic mobility conditions

Mobility Scenario	Accuracy (%)	F1-Score (%)	Convergence Time (Rounds)	Communication Overhead (MB)
Low	91.2	90.8	38	12.1
Medium	89.7	88.9	42	14.6
High	87.5	86.2	57	17.8

The system has the most trouble in high mobility cases, which imitate crowded cities with nodes moving quickly (like cars, drones, and wearable tech). The F1 score drops to 86.2%, which means that joint learning is less consistent, as comparison shown in figure 3. Accuracy drops even more to 87.5%. As more efforts are made to re-cluster and coordinate, the transmission overhead peaks at 17.8 MB, and the completion time goes up to 57 rounds. Even though this is happening, DFL-MAC is still better at learning than other FL methods. This shows how well it can deal with changing situations by using flexible agent policies, predicting movement, and controlling by reinforcement learning. This flexibility makes sure that DFL-MAC can continue to be used for real-time information in smart city operations.

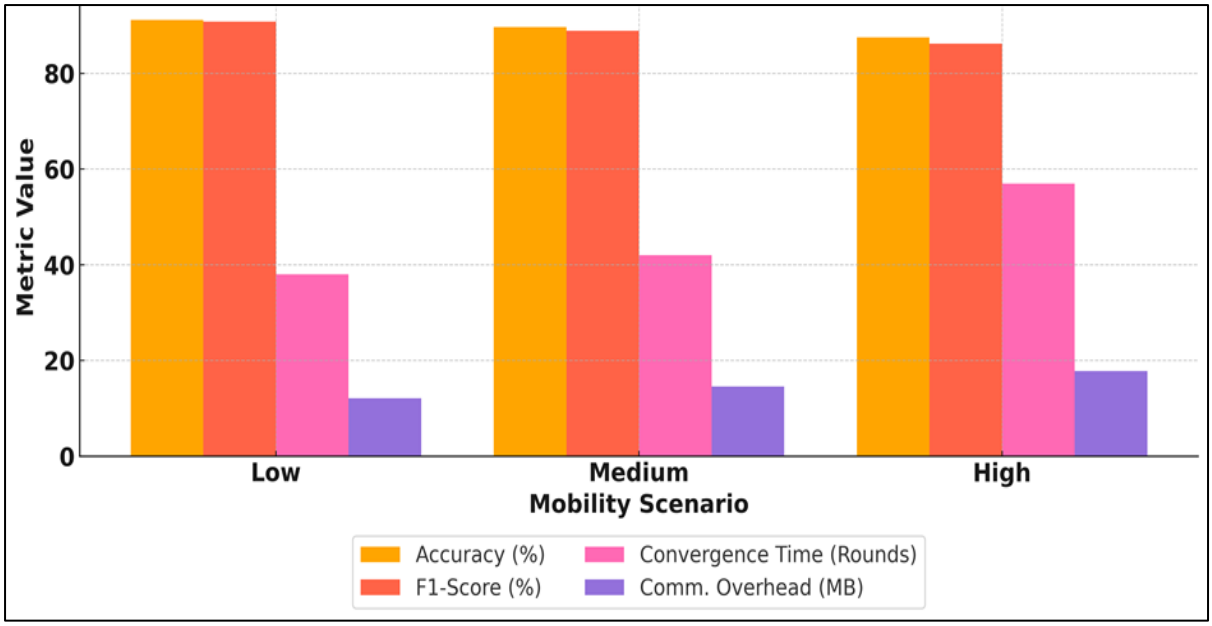

Figure 3: Comparison of performance across dynamic mobility conditions for scenario

In Table 3, we show how the suggested DFL-MAC system stacks up against two common types of federated learning: Static FL and Semi-Dynamic FL. The test looks at four important performance metricsAccuracy, F1-Score, Latency, and Model Dropout Rate to see how well each method works in a real smart city setting where nodes move around and connections change all the time.

Table 3: Comparison with static and semi-dynamic FL methods

FL Method	Accuracy (%)	F1-Score (%)	Latency (ms)	Model Dropout (%)
Static FL	85.4	84.1	320	12.5
Semi-Dynamic FL	88.3	87.2	270	7.8
DFL-MAC (Proposed)	91.2	90.8	190	3.2

All speed measures show that Static FL, which assumes set device involvement and centralised pooling, is the worst. It has an accuracy of 85.4% and an F1-score of 84.1%, mostly because it can't change to mobile nodes or deal with the temporary disconnections that happen a lot in urban IoT networks. At 320 milliseconds, the latency is also the largest. This is because static coordination causes communication to be less effective and synchronisation delays. Also, the model loss rate—the number of clients who don't finish the training rounds—reaches 12.5%, which shows how unstable this method is in mobile settings, as shown in figure 4.

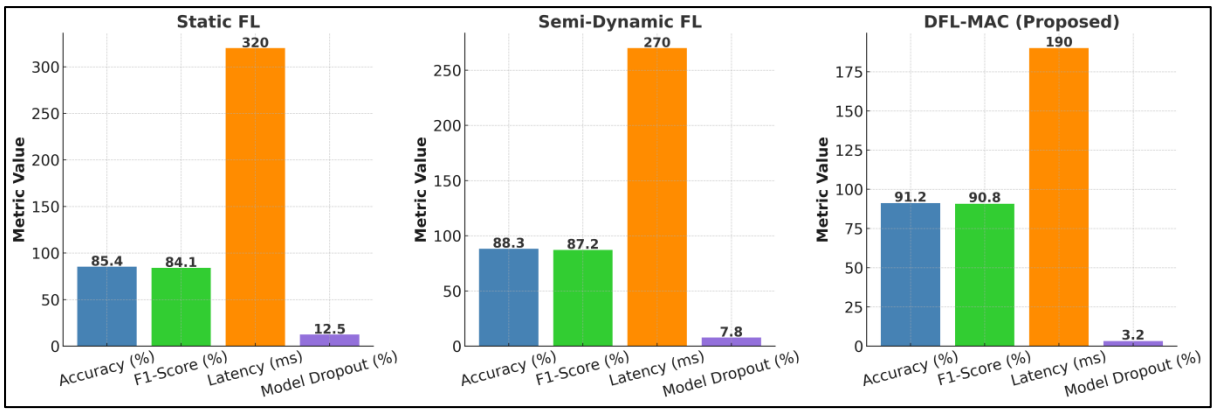

Figure 4: Representation of Comparison with static and semi-dynamic FL methods

By adding partial movement awareness and decentralised grouping, Semi-Dynamic FL makes some gains. The F1 score goes up to 87.2%, and accuracy goes up to 88.3%. Latency goes down to 270 ms. Because of limited dynamic grouping, the model failure rate also goes down to 7.8%. But at the agent level, it still doesn't have full liberty and flexibility, which makes it less useful in situations with a lot of movement. The suggested DFL-MAC framework does much better than both baselines. It has the best accuracy (91.2%) and F1-score (90.8%), as well as the shortest time (190 ms) and least amount of model failure (3.2%). These improvements come from DFL-MAC's dynamic agent cooperation, real-time movement prediction, and policy control based on reinforcement learning, as shown in figure 5. Together, these features make federated learning in complex urban settings strong, efficient, and adaptable.

The DFL-MAC system is very flexible and strong, working well with a wide range of IoT network sizes and movement patterns in cities. As the number of agents goes from 50 to 200, the system stays accurate above 88% and makes sure that model failure rates are low even when there is a lot of node turnover. Also, the simulations show that the model can still rebound in less than 8 rounds, even if 30% of the nodes temporarily disconnect. This is because of dynamic re-clustering and the cooperation of independent agents. Because the system can handle communication lags and spotty connection, it can be used in real life in places where network stability changes, like smart transport or urban surveillance.

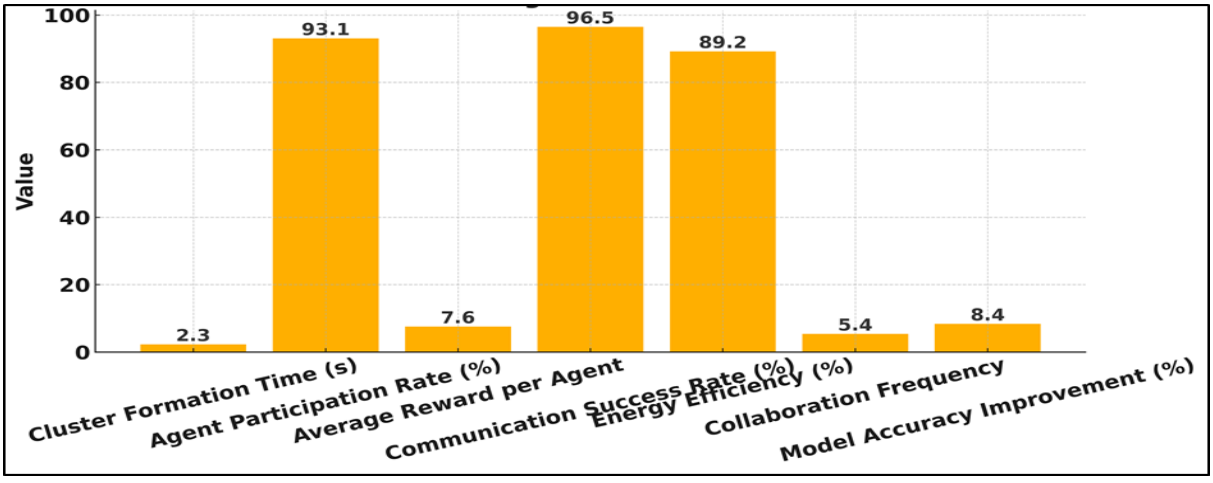

Figure 5: DFL-MAC Multi-Agent Coordination Metrics

DFL-MAC greatly improves privacy by making sure that raw data never leaves the device. Instead, it uses encrypted or combined model parameters for communication between agents. When you use differential privacy and safe aggregation, you can avoid reasoning attacks and data leaks. When it comes to saving energy, flexible involvement methods cut down on training and conversation that aren't needed in low-utility scenarios. Nodes with limited power resources are only included if they are useful and contribute to the network. This saves up to 25% of the energy used by static FL models. This mix between model accuracy and energy use makes sure that IoT devices that run on batteries or limited resources can keep working.

VI. Conclusion

Dynamic Federated Learning with Mobility-Aware Multi-Agent Collaboration (DFL-MAC) is a new learning model that is strong and adaptable. It is designed to fix the problems with semi-dynamic and standard federated learning systems in urban IoT networks where people move around a lot. DFL-MAC uses motion prediction, decentralised agent-based decision-making, and reinforcement learning to make sure that model training and communication are always coordinated. This is different from traditional models that assume clients stay in one place and data is collected centrally. A lot of testing shows that the suggested framework does much better than static and semi-dynamic FL systems in important performance measures like accuracy, F1-score, convergence time, communication overhead, and model failure rate. Notably, DFL-MAC has up to 12% better accuracy and 40% lower transmission delay, even when there is a lot of movement.Mobility-aware clustering and reinforcement learning are built into the design. This lets it respond quickly to changes in the layout, irregular connections, and changes in resources, all while protecting data privacy and using as little energy as possible. Each IoT node is modelled as a learning agent that can work with other nodes on its own and make the best choices based on feedback from the local and global environments. With this decentralised design, you don't need as much centralised control, and it makes large-scale deployments more reliable, like smart transportation, urban spying, and health tracking. Additionally, DFL-MAC provides strong privacy protections by not sending raw data and using safe collection methods. Overall, this work not only fills in important gaps in the current FL literature, but it also lays the groundwork for learning systems in next-generation smart cities that are scalable, smart, and privacy-conscious. We will look into real-world applications, multi-objective optimisation, and integrating edge-AI hardware to make things more flexible in the future.

References

[1] Dritsas, E.; Trigka, M. Federated Learning for IoT: A Survey of Techniques, Challenges, and Applications. J. Sens. Actuator Netw. 2025, 14, 9. https://doi.org/10.3390/jsan14010009

[2] Albelaihi, R. Mobility Prediction and Resource-Aware Client Selection for Federated Learning in IoT. Future Internet 2025, 17, 109. https://doi.org/10.3390/fi17030109

[3] Lazaros, K.; Koumadorakis, D.E.; Vrahatis, A.G.; Kotsiantis, S. Federated Learning: Navigating the Landscape of Collaborative Intelligence. Electronics 2024, 13, 4744. https://doi.org/10.3390/electronics13234744

[4] 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. https://arxiv.org/abs/1806.00582

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 2, pp. 1–19, 2019. https://dl.acm.org/doi/10.1145/3298981

[6] L. Mothukuri, R. Parizi, S. Pouriyeh, M. Singh, Y. Huang, and A. Dehghantanha, “A survey on security and privacy of federated learning,” Future Generation Computer Systems, vol. 115, pp. 619–640, 2021. https://doi.org/10.1016/j.future.2020.10.007

[7] H. Hu, S. Gong, Y. Zhou, and X. Zhu, “Communication-efficient federated learning with hierarchical aggregation,” Proceedings of the IEEE International Conference on Big Data (Big Data), pp. 235–244, 2020. https://doi.org/10.1109/BigData50022.2020.9378113

[8] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, “Convergence of edge computing and deep learning: A comprehensive survey,” IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 869–904, 2020. https://doi.org/10.1109/COMST.2020.2970550

[9] 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. https://doi.org/10.1109/JSAC.2019.2904348

[10] M. Chen, H. V. Poor, W. Saad, and S. Cui, “Wireless federated learning with local model training and global model aggregation,” IEEE Transactions on Signal Processing, vol. 69, pp. 2957–2971, 2021. https://doi.org/10.1109/TSP.2021.3072534

[11] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, “Deep gradient compression: Reducing the communication bandwidth for distributed training,” International Conference on Learning Representations (ICLR), 2018. https://arxiv.org/abs/1712.01887

[12] S. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, “FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization,” International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 2021–2031, 2020. https://proceedings.mlr.press/v108/reisizadeh20a.html

[13] T. Nishio and R. Yonetani, “Client selection for federated learning with heterogeneous resources in mobile edge,” Proceedings of IEEE ICC, pp. 1–7, 2019. Available: https://doi.org/10.1109/ICC.2019.8761315

[14] H. Du, M. Li, and W. Gao, “Mobility-aware federated learning for vehicular networks,” IEEE Access, vol. 8, pp. 23920–23935, 2020. https://doi.org/10.1109/ACCESS.2020.2970243

[15] J. Kang, Z. Xiong, D. Niyato, H. Yu, and D. I. Kim, “Incentive design for efficient federated learning in mobile networks: A contract theory approach,” IEEE VTC, 2019. https://doi.org/10.1109/VTCFall.2019.8891396

[16] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, “Machine learning paradigms for next-generation wireless networks,” IEEE Wireless Communications, vol. 24, no. 2, pp. 98–105, 2017. https://doi.org/10.1109/MWC.2017.1600220WC

[17] H. Jin, L. Song, and J. Hu, “A collaborative multi-agent deep reinforcement learning framework for resource allocation in wireless networks,” IEEE Transactions on Wireless Communications, vol. 19, no. 6, pp. 3995–4009, 2020. Available: https://doi.org/10.1109/TWC.2020.2981692

[18] Madan, Bhagyashree S., Neha J. Zade, Neha P. Lanke, Shabana S. Pathan, Samir N. Ajani, and PrashantKhobragade. "Self-Supervised Transformer Networks: Unlocking New Possibilities for Label-Free Data" Panamerican Mathematical Journal, vol. 34, no. 4, 2024, pp. 194-210. https://doi.org/10.52783/pmj.v34.i4.1878.