FED-UP: Federated Deep Reinforcement Learning-based UAV Path Planning against Hostile Defense System

Alvi Ataur Khalil, and Mohammad Ashiqur Rahman
Analytics for Cyber Defense (ACyD) Lab, Florida International University, USA
akhal042@fiu.edu, marahman@fiu.edu

***Abstract*—In military operations, unmanned aerial vehicles (UAVs) have been heavily utilized in recent years. However, due to the antenna installment regulation, UAVs cannot be controlled by human operators in a restricted area. Hence, artificial intelligence (AI)-driven UAVs are the practical solution to this out-of-coverage problem. With the increased use of autonomous UAVs in military applications, defense systems are deployed to target and shoot down the enemy UAVs in operation. Thus, UAVs are needed to be trained, not only to achieve goals but also to avoid static and dynamic hostile defense systems. In this work, we propose FED-UP, a federated deep reinforcement learning (DRL)-based UAV path planning framework, that enables UAVs to carry out missions in a hostile environment with a dynamic defense system. The federated learning (FL) based training accelerates the reinforcement learning process and improves model performance. We additionally introduce significant reply memory buffer (SRMB) to quicken the training process more, by selecting the crucial experiences during the training period. The experimental results validate the efficiency of the proposed model in controlling UAVs in dynamic, hostile environments.**

***Index Terms*—Federated Learning, Deep Reinforcement Learning, Unmanned Aerial Vehicles, Path Planning**

I. INTRODUCTION

Due to its low price, versatility, and compact size, the unmanned aerial vehicle (UAV) has garnered considerable interest in both military and civilian industries [1]. In most of the real-world applications, UAVs must securely move between multiple sites in order to carry out specific tasks. As a result, a trustworthy and effective navigation system in diverse environments is crucial for these UAV applications. Despite tremendous progress in making UAV operation increasingly autonomous, UAV path planning in dynamic situations is still difficult since there is little time for it to avoid unforeseen flying obstructions like birds or other air-crafts [2]. Specifically, in military applications, where UAVs can not be controlled by human operators, due to jamming [3] and out-of-coverage issues, UAV path planning becomes an even more sophisticated problem. The military hostile environments include not only static defense systems like UAV fishing towers (through means of collision), but also deploy the enemy drones, that patrol through the environment. An efficient real-time trajectory planning is needed to dodge the enemy defense systems and complete various tasks in hostile environments.

In simple situations, the traditional path planning algorithms are effective at avoiding obstacles. However, with complex environments having dynamic obstacles, the traditional methods require recalculation to adjust for changes in the environment and completely fail when reassigning unknown goals. The reinforcement learning (RL) techniques, however, can adapt the paths almost immediately. In basic RL, an agent manages a Q-table for each state-action pair, that is suitable for simple small-scale environments (having a considerably limited number of states). However, as the environment gets sophisticated, resembling real-world hostile setups, Q-table size gets exponentially larger, making it infeasible to utilize.

To counter the above mention limitations of existing path planning solutions, we propose FED-UP, a **Fe**derated **D**eep RL (DRL)-based **U**AV **P**ath planning framework for real-time UAV trajectory design in hostile environments with static and dynamic defense systems. DRL offers exceptional feature learning capabilities, allowing it to process complicated, high-dimensional states and extract clear, useful feature information from them. The federated learning (FL) helps to speed up the exploration of the environment as the swarm UAVs parallelly interact with diverse states of the environment. Even with just one UAV of the swarm experiencing the goal completion and obstacle encounter, helps the training of the whole swarm through aggregation at the global model. To further accelerate the learning, we introduce significant reply memory buffer (SRMB), which controls the batch samples used for training the FED-UP model, prioritizing the significant state experiences. In summary, our contributions are as follows:

- We design and implement FED-UP, a novel UAV trajectory planning architecture for hostile environments.
- We introduce SRMB module for selective training of FED-UP, to accelerate the intelligent behaviour learning.
- We evaluate the proposed FED-UP framework with respect to goal completion, obstacle avoidance and traveled distance. The evaluation results show that FED-UP performs better than standard DRL, specially in dynamic hostile environments.

We go over the preliminary information in Section II. Section III discusses the related works. In Section IV we present our proposed FED-UP framework. We go over the frameworks' technical specifics in Section V. In Section VI, we explain the evaluation setup and present the empirical analysis and findings. Finally, we conclude the paper in Section VII.

978-3-903176-51-5 © 2022 IFIP

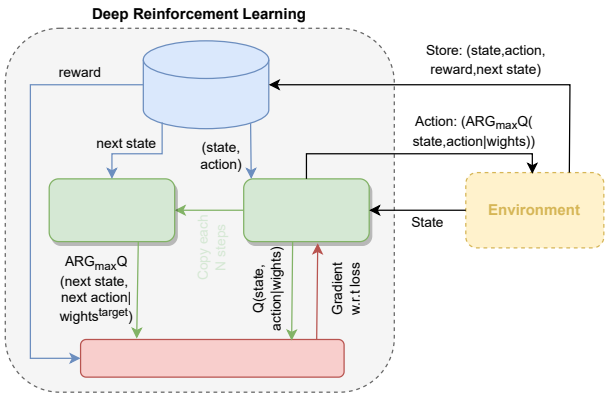


Fig. 1. Deep Reinforcement Learning Architecture.

II. Background

This section presents some preliminary concepts that will help explain the framework later.

A. UAV path-planning

Path planning is the process of determining the best route between a source and a destination, and it is one of the most critical problems that need to be investigated in the field of UAVs. The primary goal of UAV path planning is to create a cost-effective flight path that satisfies the UAV performance criteria with a low chance of being destroyed during the flight [4]. The basic path planning problem includes specific routes to choose from for reaching the goal, while in the case of UAVs, the problem is more sophisticated. The UAVs have to plan a trajectory that is collision-free, at the same time, cost-effective.

B. Deep Reinforcement Learning

A promising method for autonomously learning complicated behaviors from limited sensor observations is DRL. Although a significant portion of DRL research has concentrated on video game applications and simulated control, which have nothing to do with the constraints of learning in real environments, DRL has also shown promise in making it possible for physical robots to learn complex skills in the real-world. Consequently, as the real-world ties directly to how a human learns, it becomes an ideal domain for evaluating DRL algorithms [5]. Unlike the basic RL, where an agent manages a Q-table, DRL agent maintains a neural network, called Q-network as the learning mechanism. The Q-network based learning can effectively perform exceptionally well in real-world-based complex environments. The system diagram of a DRL architecture is presented in Figure 1. For training the Q-network, the agent's experiences are saved as training data samples into a storage called Reply Memory Buffer. To provide stability to the action decision from the Q-network, an additional supplemental neural network is added to the DRL framework, called the target network. The weights from the Q-network are copied to the target network after a certain number of episode steps. The target network predicts the future Q-values for the next states, which are used to calculate the loss of the Q-network's prediction.

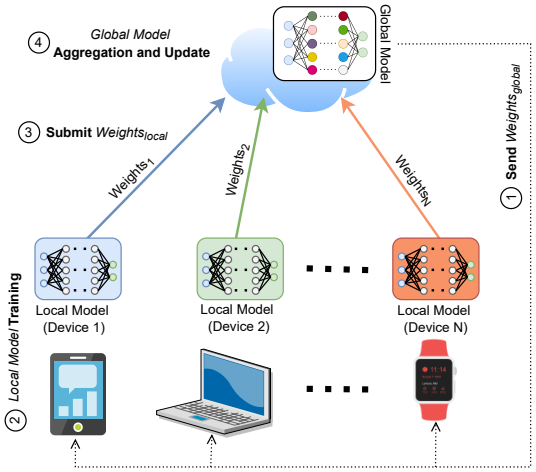


Fig. 2. Federated Learning Architecture.

C. Federated Learning

Data and computation resources are now often dispersed across end-user devices, different areas, or corporations. Laws or regulations prevent the aggregated or direct sharing of distributed data and computing resources among various areas or organizations for machine learning tasks. FL is an effective approach for utilizing distributed computing and data resources to collaboratively train machine learning models. FL also abides by the rules and regulations to ensure data security and privacy [6]. The basic goal of FL is to do a collaborative on-device training of a single machine learning model without disclosing the raw training data to any other parties [7]. The basic steps of an FL architecture is shown in Figure 2, where the cloud server holds the global machine learning model to be trained. In the first iteration, the random weights of the global model is $Sent$ to the end devices, each having a local model. The end devices $Train$ their respective local models with their private data and then $Submit$ the local model's weight to the global model. Finally, the global model $Aggregates$ the weights and $Updates$ it's weights. The updated global model's weights again $Sent$ to the end devices, each of which $Updates$ their local models and $Trains$ it with the private local data. These steps are repeated till the end of iterations.

III. Related Works

With the advancement toward UAV utilization in military operations, autonomous UAV control in a hostile environment has become a very important research area. To mention some of the recent works related to UAV trajectory design in hostile environments, Han et al. proposed a satellite-assisted UAV path planning in [3], where the satellite builds a situational map of the hostile environment using the data uploaded by the UAVs. Siemiatkowska et al. introduced a mixed-integer linear programming (MILP) based framework for UAV swarms mission in a hostile environment, where they utilized EO/IR camera images and synthetic aperture radar (SAR) based detection mechanism for potential dangerous objects [8]. On the other hand, for maritime-based hostile environments, Kim et al. proposed a social learning particle

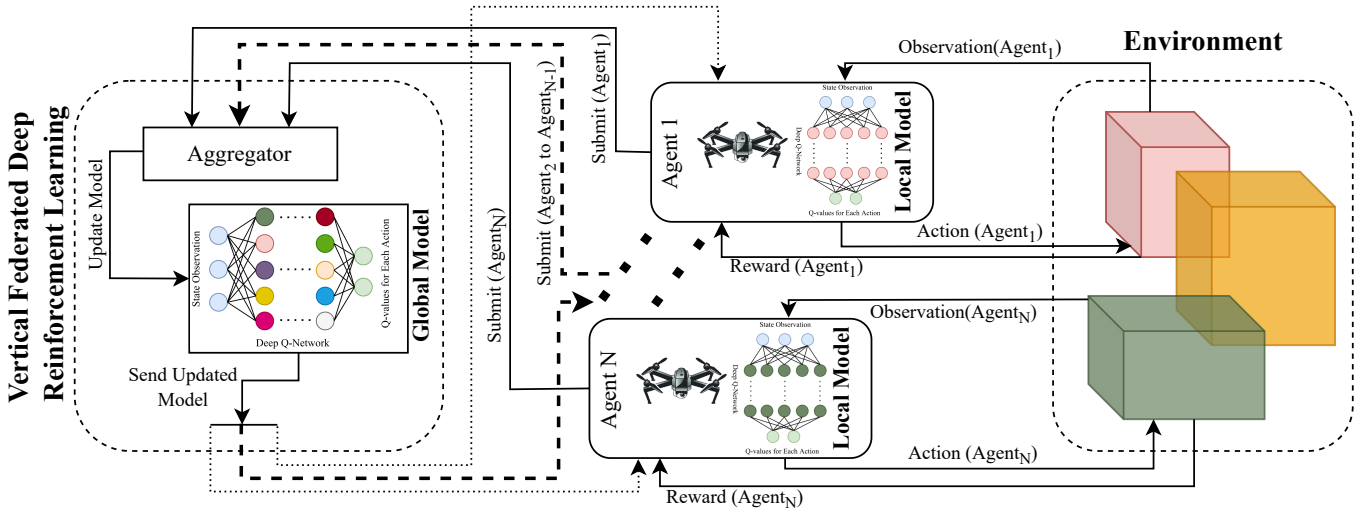


Fig. 3. Proposed FED-UP framework.

swarm optimization approach in [9], for optimal operation management of UAV swarms. For the deployment of UAVs in a hostile environment, Han et al. proposed a congestion game to model the interaction among the UAV swarms in [10], where each UAV independently adjusts the location and realizes the position control. Wen et al. proposed an online path planning framework for UAVs hostile environment in [11], where they model static threats based on an intuitionistic fuzzy set (A-IFS) and dynamic threats based on a pursuit-evasion game. All of these papers assumed the environment state is static after the trajectory or deployment plan has been formulated.

RL-based UAV path planning solutions have become increasingly popular due to their effectiveness in unknown settings. Khalil et al. proposed a novel economic trading based RL-framework for co-operative UAV path-planning in non-dynamic environments [12]. Alpdemir et al. proposed a comprehensive probabilistic radar behavior model environment that complies with MDP, as well as a framework that combines a fundamental RL algorithm with a particular type of transfer learning [13]. Papoudakis et al. evaluate and compare three different classes of multi-agent DRL (MARL) algorithms in [14], to establish the notion that the performance of DRL strongly depends on the environment properties. In order to successfully solve the issue of the DRL's neural network's erroneous prediction at the early stage of training, Xie et al. offered an action selection technique that combines the present reward R value with the Q value [1]. Yan et al. introduced a quick situation assessment approach in [15] that converts sequential scenario maps of the global environmental states into dynamic adversary threats. Wan et al. developed a motion control framework based on actor-critic architecture that can perform dual-channel roll and speed control by anticipating the desired steering angle and analyzing the probability of a collision [16]. The UAV can fly safely on its own in dynamic, uncertain environments with the help of the controller. These works, however, consider that the UAV agents have the global observation of the dynamic obstacles to formulate a global awareness, which hardly resembles the real-world hostile environment. Consequently, we propose FED-UP framework considering partial awareness of the environment, resembling sensor-based observation of the UAVs. We experiment with different levels of observation capability with the layer (L) parameter, which is discussed in the later sections.

IV. FRAMEWORK

In this section, we introduce the proposed FED-UP framework, which is presented in Figure 3. We specifically utilize the vertical federated RL concept, as we are considering that a collaborating swarm of UAVs deployed in the same environment together will simultaneously explore different states of the environment. Each of the UAV agents will have its own local RL model, which will be trained through K consecutive episodes and reinitialized to replicate the global federated model at the beginning of the next $(K+1)$-th episode. The observation vector size is exactly the same for all the agents, and so is the action space. Each UAV agent will be deployed at a different state in the three-dimensional environment, and they will interact and explore the surrounding states. The ϵ parameter (discussed later in Section V) will control the exploring and exploitation behavior, which will be constant for all the agents. Through the steps of an episode, the experiences of the UAV agents will be stored in the respective memory buffers of the agents (in our model, each agent has two distinct memory buffers, discussed later in Section V). At the end of each episode, the local Q-network of an agent is trained with a random batch of experience tuples from the memory buffer of the agent. The detailed structure of the proposed DRL mechanism-based learning of an agent is shown in Figure 5. The Q-network will be consulted for an agent's exploitation actions. At the end of each K episodes, the UAV agents will submit the current weights of the respective Q-networks to the vertical federated DRL (VF-DRL) module, where an Aggregator module performs the weights and biases aggregation from all the local models. The global model is

TABLE I
LIST OF NOTATIONS

Symbol	Definition
S	Set of states
s	Current state
s'	Next state
A	Set of actions
R	Reward function
r	Current reward
L	Agent's observable number of layers
K	FL model update interval (episodes)
N	Target model update interval (steps)
B	Batch size of Q-network training samples
T	Transition probability
π	Policy
ϵ	Exploration parameter
γ	Discount factor
ϕ	Sorting parameter
σ	Sampling rate
δ	ϕ value increment interval

updated with the aggregated weights and biases, and later sent to each of the UAV agents, when the $(K+1)$-th episode begins. The structure of the global model is exactly identical to the local models. The vertical nature of the FL drives the exploration faster. This accelerates the training process of the global model and helps in further improving the model performance in a relatively less number of episodes. Table I contains the definition of the notation, which are utilized in the modeling of the framework.

V. TECHNICAL DETAILS

In this section, we discuss the formulation of the path-planning in a hostile environment problem and the Markov Decision Process (MDP)-based design of the DRL solution. Then, we introduce the SRMB concept to accelerate the learning of the agents. Later, we discuss the VF-DRL module in detail. Finally, we discuss the hostile defense systems.

A. Modeling DRL-based Path Planning

We model the problem as a Markov game, which is the generalization of MDP. Generally, DRL MDP is a five-tuple, consisting of the states, actions, transition probability, reward, and discount factor. However, we modeled a further sophisticated MDP for the complex hostile environment we are considering in this work. Formally our MDP can be presented by a nine-tuple: $\{$S, A, T, R, $\pi, \epsilon, \gamma, \phi, \sigma\}$. We will discuss each element of the tuple in the subsequent sections:

1) States (S): The set of states represent all the possible observation an agent can have in the environment. In our case, as we are considering partially observable MDP, the agents' observation at a particular time step includes one/more layers of cubes around the agent. To be more specific, if the number of layers is one, that means there is one layer of cubes around the agent, which results in twenty-seven cubes. This concept effectively simulates the agents' partial view of the whole environment. It also lets the agent to be prepared when there is an obstacle or goal nearby. It can effectively observe the point of interest if the point has reached the surrounding layers. We also assume the goal locations are known to the agent at the start of the episode, so agents can plan the trajectory towards the goals while dodging the obstacles in real-time. Consequently, the observation also includes the three coordinate distances from the goals.

2) Actions (A): The set of actions represent the activity an agent can perform to interact with the environment. As we are considering a discrete environment with cubes, the movements are also discrete (and not continuous angular movement). So there are eleven actions an agent can perform, which include going up, going down, going forward, going backward, going left, going right, going diagonally in four directions, and no movement (hovering).

3) Transition (T): Transition probability represents the probability of transitioning from one state to another state. In our model, the probability is dependent on the current state and the L number of previous states, where L is the number of surrounding layers (observable by the agent) chosen for that environment. That is because the agent has knowledge of the surrounding L cubes to perform a suitable action.

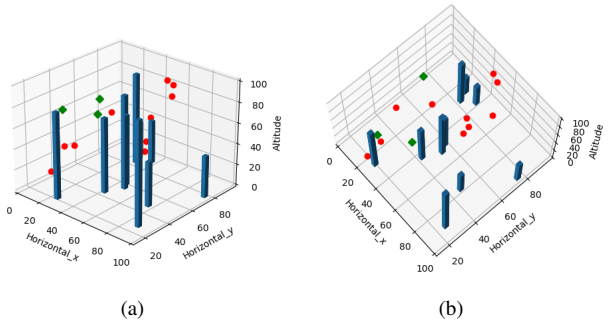
4) Reward (R): The rewards define the reward function that dictates the agent's learning process. The reaching of goal points awards the agent rewards while ending up on an obstacle penalizes the agent. Again, there is a movement penalty for each action taken that does not make the agent reach a goal or obstacle location.

5) Policy (π): Policy is the learning of agents, by interacting and exploring the environment; that is, which action an agent will take, given a particular state. In our case, it is simply the Q-network's weight and biases that determine the action an agent will take for a given observation of the state.

6) Exploration Parameter (ϵ): The exploration parameter controls the action behavior of the agent. It is primarily set close to 1, which means that there is an almost certain probability that the agent will take random actions to interact with the environment as exploration behavior. Those actions will shape the learning of the Q-network. As the episode progresses with additional steps, the ϵ value is decreased, so is the probability of random action. The less the value of ϵ, the more the probability that the agent will perform a action by consulting the Q-network, which is the exploitation behavior.

7) Discount Factor (γ): The discount factor determines the RL agents' level of concern about rewards in the distant future in comparison to those in the near future. The value of this parameter ranges from zero to one. The agent will only learn about actions that result in an immediate reward if γ is set to zero, making it fully myopic. A reward R in future, that occurs after N steps, will be discounted by a factor of N (i.e., the reward will be R^N).

8) Sorting Parameter (ϕ): This parameter controls the insertion of the experiences into the SRMB. The value of this parameter is set from 0 to 1, where a zero value means only the experiences where an agent ends up on a goal/obstacle are stored in SRMB, while none of the layer encounters are considered significant. The value can be incremented to one in L steps, where L is the number of layers. Let, δ represents



(a) (b)

Fig. 4. The simulated three dimensional environment for a random episode, (a) From x-y-z axis perspective, (b) From nearly x-y axis perspective. In these figures, the blue towers are the static defense system and the red spheres are the dynamic hostile defense system. The green diamonds represent the goals.

one increment for the ϕ value, then δ can be calculated using the following equation:

$$\delta = \frac{\phi_{max} - \phi_{min}}{L} \quad (1)$$

Here, ϕ_{max} and ϕ_{min} are set to 1 and 0, respectively. The first increment from zero means the experiences where a goal/obstacle is encountered in the first neighboring layers are stored, while the final increment up to one means all the experiences where a goal/obstacle is even in the most distant surrounding layer are stored.

9) Sampling Rate (σ): This parameter controls the sampling rate of the experience memories, which are selected for batch-wise training of the Q-Network. This parameter can take values from 0 to 1, where 0 means all the samples of the training batch are taken from the general Reply Memory Buffer, and 1 means all the samples of the training batch are taken from the SRMB. We experiment with different values of σ to present the effect of the SRMB on the performance of FED-UP model.

B. Selective Training with SRMB

In this section, we introduce the proposed SRMB concept, that will aid the training of the Q-network in a faster way. Generally, in the case of DRL, as discussed in Section II, the Q-network's training data is simply the experience tuples from the RL agent's exploration. In each time step, the agent's current state, current action, current reward, next state, and completion flag (indicating goal completion state) are stored into a memory buffer as a single tuple. The reply memory buffer has a predefined memory size. As an episode terminates, a random batch of tuples are selected from the memory buffer to train the Q-network. However, this randomization can hardly ensure that the Q-network is served with the memory samples that will aid the learning of intelligent behavior. So we are proposing SRMB, which will help to train the model in a controllable manner by storing the most significant memory tuples. Each agent will have both the Reply Memory Buffer and the SRMB, where the sampling technique (with parameter *sampling rate*, σ) and sorter (with *sorting parameter*, ϕ) will control the effect of SRMB. The function and properties of σ and ϕ are already discussed in the previous section. Figure

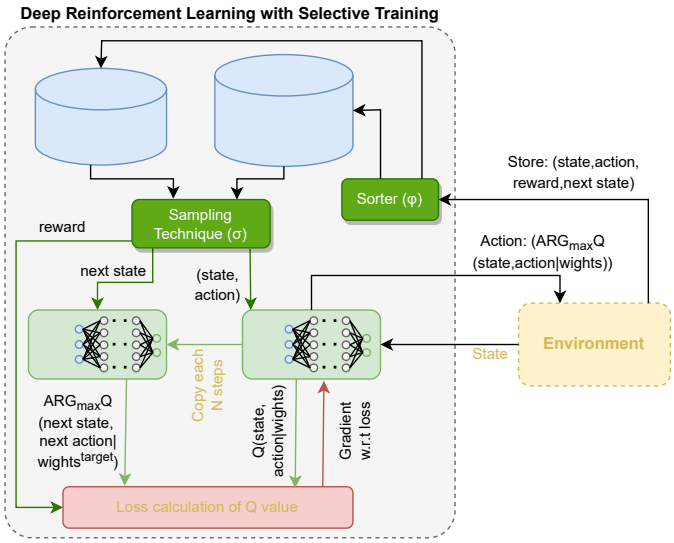


Fig. 5. Improved Deep Reinforcement Learning Architecture with SRMB.

5 presents the block diagram of the improved DRL training with SRMB.

Sorter: Unlike the general DRL, where the reply memory buffer directly receives the memory tuples, in our case the sorter receives all the memories. Then it sorts the memories into SRMB according to ϕ parameter. Whichever memories are not stored in the SRMB are sent to the Reply Memory Buffer for storing.

Sampling Technique: This module is responsible for preparing the batch samples for Q-network training. According to the sampling rate parameter, this module takes $B \times \sigma$ samples from the SRMB, where B is the size of a single training batch. Consequently, the rest $B \times (1-\sigma)$ samples are taken from the Reply Memory Buffer.

C. Federated Learning for Swarm Intelligence

The FL technique is primarily utilized for training a machine model from distributed data sources while maintaining data security. We found that the inherit distributed learning structure of FL boosts the performance of the swarm-based learning models, as the global model gets fitted with all the diverse experiences from different swarm agents. Similar to the standard FL, the proposed vertical FL happens in four repeating steps (presented in Figure 2). At the beginning of the first episode, a global model is initialized with random weights and biases, and sent to all the agents. Then agents initialize their local Q-network model with the received weights and biases. After each K episodes, the agents submit their current Q-network's weights and biases to the VF-DRL module, where an aggregator model is responsible for processing the received network models. We experiment with three different aggregation algorithms, which will be discussed later in this section. The global model is then updated with the aggregated weights and biases. Finally, the global model is sent again to all the agents at the beginning of $(K+1)$-th episode, and the agents update their local Q-network copying the weights

and biases received from the VF-DRL module. Now, we will briefly introduce and discuss the three aggregation algorithms we experimented with:

- FedSGD: In federated stochastic gradient descent [17], a random portion of the swarm UAVs are selected by the aggregator, and their local Q-networks' gradients are averaged and utilized to create a gradient descent step for the global model.
- FedAvg: In the federated averaging [18] technique, the aggregator averages the weights and biases of all the local Q-network models from the agents to update the global model.
- FedMA: In the federated match averaging [19] technique, the layers are processed separately by the aggregator, where only the nodes with comparable weights are merged, and the global model's nodes are updated.

We experiment with all three algorithms in order to decide on the appropriate one for this application. After observing the average reward value of 2000 episodes, the $FedAvg$ algorithm was found to be the most fitting option for this application. All the experiments (discussed in Section VI) are performed having $FedAvg$ as the aggregation algorithm.

D. Modeling of Hostile Defense System

For designing an environment with a hostile defense setup, we introduced three types of defense systems:

1) Static Defense System (SDS): These defense systems are simulated by designing static towers with different heights at different locations in the environment. In Figure 4, the blue towers represent these defense systems.

2) Dynamic Fixed Route Defense System (DFRDS): These defense systems are simulated by objects moving back and forth in the environment in a fixed route. There are three kinds of routes: horizontal movement with variable x-axis values and fixed y-z axis values, horizontal movement with variable y-axis values and fixed x-z axis values, and vertical movement with variable z-axis values and fixed x-y axis values. In Figure 4, the red spheres represent these defense systems. This defense system resembles guarding UAVs.

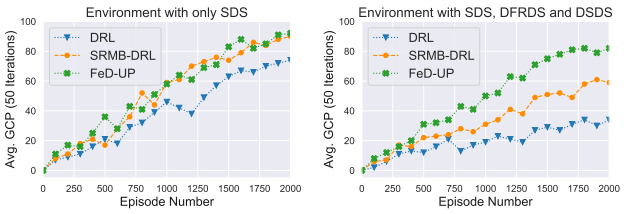
3) Dynamic Stochastic Defense System (DSDS): These defense systems are simulated by objects moving randomly in the environment, without any fixed route or path. In Figure 4, the red spheres represent these defense systems too. This defense system resembles surveying and exploring UAVs.

VI. Experimental Analysis

In this section, first, we introduce the different evaluation metrics utilized to validate the performance of the FED-UP. Then, we evaluate and analyze the framework's performance with respect to the metrics introduced.

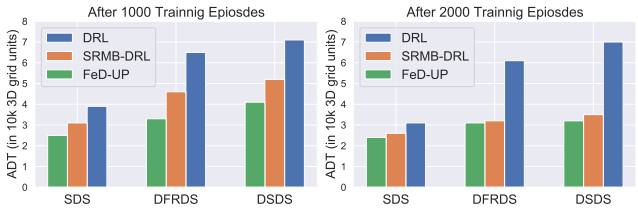
A. Evaluation Metrics

This section defines the three metrics used for evaluating the performance of FED-UP framework.



(a) (b)

Fig. 6. Average GCP from 50 iterations by the UAV swarms, with DRL, SRMB-DRL, and FED-UP, having (a) only SDS obstacles in the environment, and (b) having SDS, DFRDS, and DSDS obstacles in the environment.



(a) (b)

Fig. 7. ADT of the UAV agents, during training with DRL, SRMB-DRL, and FED-UP, after (a) 1000 episodes, and (b) 2000 episodes, with environment with SDS, DFRDS and DSDS.

Goal Completion Percentage (GCP): This metric defines what percentage of goal set has been visited or processed. As a result, the GCP can be described as follows:

$$GCP = \frac{\#\ of\ goals\ completed}{\#\ of\ total\ goals\ in\ the\ goal\ set} \times 100$$

Obstacles Encounter Count (OEC): This metric defines how many times hostile entities has been encounter. This is the summation of total hostile encounter. OEC can be defined as:

$$OEC = \sum_{i \in Agents} count_i$$

Here, $count_i$ is the number of times i-th agent has encountered an obstacle, and $Agents$ refers to the set of all agents.

Average Distance Travelled (ADT): This metric defines the average amount of distance the agents have to travel to reach their goal points. Ideally, the smaller the ADT, the more efficient the path planning technique. ADT is defined as follows:

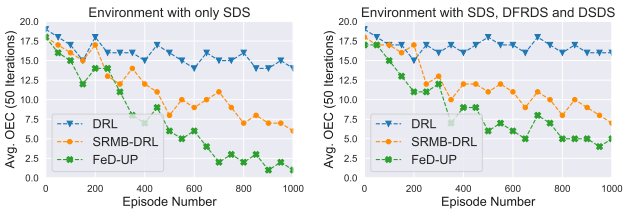
$$ADT = \frac{1}{size(Agents)} \sum_{i \in Agents} d_i$$

Here, d_i is the distance that i-th agent had to travel to complete its goal set, and $Agents$ refers to the set of all agents.

B. Evaluating the Performance of FED-UP

In this section, we evaluate the performance of FED-UP framework, by presenting a comparative analysis with DRL and SRMB-DRL (DRL with SRMB, but no FL) with respect to the metrics discussed in the previous section.

1) Goal completion capability in different hostile setups: In this part, we discuss the average GCP of agent swarms with different learning techniques, in environments with progressively higher difficulty, as presented in Figure 6. From Figure 6(a), it is seen that with only static obstacles, all



(a) (b)

Fig. 8. Average OEC from 50 iterations by the UAV swarm, with DRL, SRMB-DRL, and FED-UP, having (a) only SDS obstacles in the environment, and (b) having SDS, DFRDS, and DSDS obstacles in the environment.

the techniques perform almost similarly. However, as the dynamic obstacles are introduced in the environment (in Figure 6(b)), there arises visible performance differences among the methods. Although the average GCP goes down for all the methods, FED-UP performs substantially better than SRMB-DRL and DRL (completing around 80% goals). The SRMB-DRL technique, even without FL, does perform better than the standard DRL method, completing around 60% goals in the dynamic environment.

2) Optimal travel distance in different hostile environments: Figure 7 presents the ADT of the UAV swarms, being trained with different methods. With 1000 training episodes, it is observed in Figure 7(a) that FED-UP agents require minimal distance to be traveled to complete the goal set in every kind of hostile setup. Then, in Figure 7(b) we see SRMB-DRL improves the most and gets close to the performance of FED-UP (around 25k 3D grid units), while there is little improvement in the latter. This is because the training performance of FED-UP with respect to ADT becomes saturated after 1000 episodes. Even after 2000 episodes, there is minimal improvement observed for the standard DRL.

3) Obstacle avoidance capability in different hostile setups: In this part, we evaluate the FED-UP framework with respect to the OEC performance in progressively harder hostile environments (Figure 8). With only SDS, FED-UP performed substantially better than both DRL and SRMB-DRL, as presented in Figure 8(a). However, as the dynamic obstacles are introduced in the environment (Figure 8(b)), the performance of SRMB-DRL becomes comparable to FED-UP. That is because both of the techniques utilize the SRMB, while the slight advantage for the FED-UP comes from the VF-DRL module. The overall performance of both methods degrades with a tougher hostile environment.

VII. CONCLUSION

In this work, we present an FL-based improved DRL method for UAV path planning in hostile environments with static and dynamic defense systems. Our evaluation results have shown that the proposed FED-UP model outperforms the standard DRL method in both the simplistic environment setup (only SDS) and complex environment setups (with DFRDS and DSDS). We observe that the proposed model training is remarkably faster than DRL in terms of training episodes. Moreover, there is a significant improvement in performance in terms of obstacles avoidance (68% less OEC with dynamic defense), goal completion (113% greater GCP with dynamic defense), and minimizing travel distance (57% less ADT with dynamic defense). In our future work, we will focus on further optimizing the SRMB and VF-DRL modules and experiment with even more sophisticated defense systems (including tracker and follower UAVs).

REFERENCES

[1] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments," *IEEE Access*, vol. 9, pp. 24 884–24 900, 2021.

[2] Z. Ma, C. Wang, Y. Niu, X. Wang, and L. Shen, "A saliency-based reinforcement learning approach for a uav to avoid flying obstacles," *Robotics and Autonomous Systems*, vol. 100, pp. 108–118, 2018.

[3] C. Han, A. Liu, K. An, H. Wang, G. Zheng, S. Chatzinotas, L. Huo, and X. Tong, "Satellite-assisted uav trajectory control in hostile jamming environments," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3760–3775, 2021.

[4] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, 2020.

[5] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[6] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: A survey," *Knowledge and Information Systems*, pp. 1–33, 2022.

[7] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[8] B. Siemiatkowska and W. Stecz, "A framework for planning and execution of drone swarm missions in a hostile environment," *Sensors*, vol. 21, no. 12, p. 4150, 2021.

[9] J. Kim, H. Oh, B. Yu, and S. Kim, "Optimal task assignment for uav swarm operations in hostile environments," *International Journal of Aeronautical and Space Sciences*, vol. 22, no. 2, pp. 456–467, 2021.

[10] C. Han, A. Liu, K. An, G. Zheng, and X. Tong, "Distributed uav deployment in hostile environment: A game-theoretic approach," *IEEE Wireless Communications Letters*, vol. 11, no. 1, pp. 126–130, 2021.

[11] N. Wen, X. Su, P. Ma, L. Zhao, and Y. Zhang, "Online uav path planning in uncertain and hostile environments," *International journal of machine learning and cybernetics*, vol. 8, no. 2, pp. 469–487, 2017.

[12] A. A. Khalil, A. J. Byrne, M. A. Rahman, and M. H. Manshaei, "Replanner: Efficient uav trajectory-planning using economic reinforcement learning," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2021, pp. 153–160.

[13] M. N. Alpdemir, "Tactical uav path optimization under radar threat using deep reinforcement learning," *Neural Computing and Applications*, vol. 34, no. 7, pp. 5649–5664, 2022.

[14] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Comparative evaluation of cooperative multi-agent deep reinforcement learning algorithms," *arXiv: 2006.07869*, 2020.

[15] C. Yan, X. Xiang, and C. Wang, "Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments," *Journal of Intelligent & Robotic Systems*, 2020.

[16] K. Wan, X. Gao, Z. Hu, and G. Wu, "Robust motion control for uav in dynamic uncertain environments using deep reinforcement learning," *Remote sensing*, vol. 12, no. 4, p. 640, 2020.

[17] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5332–5344, 2020.

[18] S. Ek, F. Portet, P. Lalanda, and G. Vega, "Evaluation of federated learning aggregation algorithms: application to human activity recognition," in *2020 ACM International Symposium on Wearable Computers*, 2020.

[19] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv preprint arXiv:2002.06440*, 2020.