Resource Management in Next Generation MEC-Enabled Wireless Networks Using Machine Learning

By **Anne Catherine Nguyen**

A thesis submitted to the Faculty of Engineering
in partial fulfillment of the requirements for the degree of

Master of Computer Science
and Concentration Applied Artificial Intelligence

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

With the help of unmanned aerial vehicles (UAV) and multi-access edge computing (MEC), the Internet of Things (IoT) can provide valuable insights for a smart farm. The IoT devices are installed across the farmland and they monitor the land by performing image classification tasks. Some of the tasks include identifying a fire, monitoring the farmland for pest infestations, and identifying the growth stages of the crops. These tasks must be done frequently to provide the latest information, and taking too long to complete these tasks will lead to irrecoverable damages. Each task has a predetermined deadline. To ensure these computationally heavy tasks are completed on time, the IoT devices can offload these tasks to nearby hovering UAVs to perform the tasks on their behalf. One significant limitation of IoT devices and UAVs is that these devices rely on batteries. To provide support for these nodes that have finite energy, a MEC server is added as another computing resource to further alleviate the task load.

The wireless network is the cornerstone of the smart farm network, and it requires an algorithm to schedule the tasks. The decision-maker must consider the network's current energy capacity, computing resources, and deadlines for the tasks. This thesis considers a series of rule-based task scheduling algorithms as well as machine learning algorithms. The rule-based algorithms are based on round robin, or heuristics such as the UAV's current remaining energy and the networks' current computing queue. This thesis explores various Reinforcement learning (RL) techniques namely Q-Learning, Deep Q-Learning (DQL), Deep Risk-Sensitive (DRS), and DQL with Risk Quantification (RQ). These RL techniques jointly consider the energy and deadline limitations in their objective functions.

The simulation results show that the machine learning algorithms significantly outperformed the rule-based algorithms. The RL techniques that employed deep learning had lower percentages of tasks that exceeded their deadlines. In addition, the deep neural networks DNN helped decrease the

number of iterations required to achieve the optimal solution. This means that DQL and DRS were able to achieve the optimal solution faster than their tabular Q-Learning and Risk-Sensitive counterparts. In the simulation results, the majority of the tasks that did not meet their deadlines were fire detection tasks. This was because the fire tasks occurred frequently and had a shorter deadline. Unlike the other RL techniques, the RQ algorithm quantified the severity of each type of deadline violation and UAV battery levels in terms of damages. This enabled the agent to avoid actions that lead to severe damages. As a result, the RQ method was the only algorithm able to eliminate deadline violations that were fire detection tasks.

Acknowledgments

Firstly, I would like to thank Dr. Melike Erol-Kantarci for her guidance throughout my studies these past few years. She has opened up many opportunities for me and it has enriched my life immensely.

I would also like to thank Dr. Turgay Pamuklu for his mentorship and collaboration. His support and feedback have helped me grow as a researcher.

To my friends and colleagues at the Networked Systems and Communications Research Lab, thank you all. I am grateful to have been part of such a supportive community.

To my parents Ho Nguyen and Thanh Cao,

Thank you for being the foundation that enables me to soar. Your resilience inspires me.

Contents

List of Figures

List of Tables

List of Symbols

α	Risk Quantification's cost percentile
$\alpha^{B}_{\langle j,t \rangle}$	IoT task offload indicator
β	Scaling factor for current UAV's reward in Risk Quantification's reward function
δ	Mean uplink delay
$\Delta^{j_P}_{j'_R}$	Transmission delay between two computing resources
$\Delta^{P}_{\langle j,t \rangle}$	Task's computation delay
$\Delta^{Q}_{\langle j,t \rangle}$	Task's queuing delay
$\Delta_{\langle j,t \rangle}$	End-to-end delay
$\Delta_{j'}$	Computing resource's queue time
ϵ	Energy change threshold
η	Risk Quantification agent's CVaR value
Γ	Energy risk cost's scaling factor
$\gamma^{D}_{\langle j,t \rangle}$	Task deadline
$\mathbb{A}$	Action set for machine learning algorithms

$\mathbb{C}$	Deep Risk-Sensitive's risk function
$\mathbb{S}_{DQL}$	State set for Deep Q-Learning algorithm
$\mathbb{S}_{DRS}$	State set for Deep Risk-Sensitive algorithm
$\mathbb{S}_{Q}$	State set for Q-Learning algorithm
$\mathbb{V}$	Maximum number of network deadline violations
$\mathbb{V}$	Maximum number of total deadline violations allowed
$\mathbb{Z}$	Risk Quantification agent's set of costs
$\mathcal{C}$	Risk Quantification's cost function
$\mathcal{Q}$	Deep Risk-Sensitive's Q-value
$\mathcal{V}_j^L$	Deadline violation reward level
μ	Risk Quantification's reward value for an agent
Ω	Risk state set for Deep Risk-Sensitive
Θ	Scaling factor
Θ^D	Scaling factor for high-risk deadline violation
Θ^M	Scaling factor for mean uplink delay
Υ_j^B	UAV's full battery capacity
Υ_j^C	CPU energy consumption
Υ_j^L	Battery reward level
Υ_j^R	Remaining energy in a battery
Υ_U	UAV's remaining battery level

ζ Deep Risk-Sensitive agent's objective priority factor

$p^{+}_{\langle j,t\rangle j't'}$ Indicator for CPU allocation's first time interval

$p^{-}_{\langle j,t\rangle j't'}$ Indicator for CPU allocation's last time interval

$p_{\langle j,t\rangle j't'}$ CPU allocation indicator

R_{DQL} Reward function for Deep Q-Learning algorithm

R_{DRS} Reward function for Deep Risk-Sensitive algorithm

R_{Q} Reward function for Q-Learning algorithm

R_{RQ} Reward function for Risk Quantification algorithm

$v_{\langle j,t\rangle}$ Deadline violation indicator

W Energy consumption weight

$x_{\langle j,t\rangle j'}$ Task offload from IoT indicator

g Risk Quantification's energy cost function's growth factor

j UAV indicator

j' CPU indicator

k Task type indicator

l' MEC indicator

s Risk Quantification's fire task end-to-end delay risk growth value

t Time interval

w Risk Quantification's other tasks risk growth value

Abbreviations

1G First generation. 7

2G Second generation. 8

3G Third generation. 8

4G Fourth generation. 8

5G Fifth generation. 3, 7–9, 11, 20, 54

A3C Asynchronous Advantage Actor Critic. 24, 25

AoI Age of Information. 23, 26

CPU Central processing unit. xv, 19, 21, 28, 31, 34, 40, 47, 57

CVaR Conditional Value at Risk. xiii, 12–14, 24–26, 52

DNN Deep neural network. ii, 43, 44, 46, 48, 59–61

DQL Deep Q-Learning. ii, iii, xiv, xv, 4, 16, 22, 43, 44, 59, 60, 63, 66, 67, 69, 70, 72, 73, 76

DRS Deep Risk-Sensitive. ii, iii, xiv, xv, 4, 44, 47, 49, 61, 67, 69, 70, 72, 73, 76

eMBB enhanced Mobile Broadband. 8

HEF Highest Energy First. 3, 36, 38, 65, 66, 76

HetNet Heterogeneous networks. 10, 23, 24

IoT Internet of Things. ii, xiii, 1, 3, 7–10, 19, 20, 23–28, 30–32, 34, 36, 38, 47, 56, 75

ITU International Telecommunication Union. 8

ITU-R Radiocommunication sector of the International Telecommunication Union. 8

KPI Key Performance Indicator. 54, 76

LTE Long-term evolution. 54

MDP Markov decision process. 23

MEC Multi-access edge computing. ii, xv, 7, 9, 19–25, 27, 30–33, 36, 37, 40, 42, 47, 54, 56, 63, 65, 66, 73, 75

mMTC Massive Machine Type Communication. 8

mmWave Millimeter waves. 8, 10, 11

NOMA Non-orthogonal multiple access. 21

QHEF Lowest Queue Highest Energy First. 3, 38, 63, 65, 66, 76

RL Reinforcement learning. ii, iii, 7–9, 11, 14, 15, 17, 20, 21, 23, 57, 76

RQ Risk Quantification. ii, iii, xiii–xv, 4, 49, 53, 61, 72–74, 76

RR Round Robin. 36, 63, 65, 66, 76

SDN Software-Defined Networking. 22

UAV Unmanned aerial vehicle. ii, iii, xiii–xv, 1–3, 7, 9, 19–23, 27, 28, 30–34, 36–38, 40, 42, 44, 45, 47, 49, 51–54, 56, 57, 62, 63, 65–69, 71, 73, 75–77

URLLC Ultra Reliable Low Latency Communication. 8

WBAN Wireless body area network. 25

Chapter 1

Introduction

1.1 Motivation

Agriculture is a domain that can leverage wireless networks and management tools to ensure that the farm is run efficiently. The farms' resources must be managed carefully to sustainably keep up with the demands of a growing population [1]. Smart farms are farms that incorporate technology such as the Internet of Things (IoT) to automatically monitor the state of the farm. They provide real-time reporting on the farm's soil condition, the growth of the crops, water levels, and livestock location [2]. They collect information through various types of sensors that monitor temperature, humidity, soil, fluid, and location [3]. Through wireless connectivity, the IoT sensors relay the latest status of the farm back to a central server. Once the central server has received the up-to-date date information from the sensors, precise decisions can be made. For example, turning on sprinklers in a targeted area where the soil was identified as dry.

Another useful tool for smart farming is unmanned aerial vehicles (UAV). They are deployed in the air overlooking the farm. UAVs are capable of acting as sensors and provide an aerial perspective on weed detection, pest identification, crop monitoring, and soil monitoring [4]. They can perform

tasks for the farm such as seed planting and pesticide spraying [5]. UAVs can also act as aerial base stations [6,7] and provide connectivity to the farmland.

The usage of artificial intelligence in conjunction with smart farm devices has proven to be useful in managing a farm. Due to their 3-dimensional positioning capabilities, UAVs are able to capture an accurate image from different angles. With the assistance of image classification, UAVs are useful for crop monitoring tasks where they can identify spots on plants [8] [9]. Yu et al. proposed a pest monitoring system that comprised UAVs with a mounted camera and ground sensors to monitor fields of crops [10]. The UAVs flew across the fields while capturing images and performing image classification to detect pest infestations on crops. With the assistance of artificial intelligence, smart farm networks can self-manage their resources to maintain efficient performance [3].

To accurately manage a smart farm, it requires the most up-to-date insights. This information can have various levels of urgency. For example, if the smart farm has a fire detection system, these fire-identifying tasks must be done as quickly as possible to prevent the fire from spreading and causing more damage. A growth monitoring system for crops can be another task in the smart farm, but the consequences of the system taking too long to identify the growth stage of a crop are not as dangerous as identifying a fire. The smart farm's equipment must perform a variety of tasks with differing deadlines.

In an environment with several limited resources and a series of time-sensitive tasks, task distribution is vital. Task distribution and scheduling have been studied in the fields of computing [11], mobile networks [12], and business project management [13]. It ensures that the tasks are computed such that they adhere to any preset constraints, while efficiently utilizing the available resources. Poor task distribution can lead to over-usage of certain resources while other resources remain idle and unused. The over-usage of some resources may lead to those resources failing to complete the tasks,

the resource no longer being available for usage, and the slow down of task completion. These situations can lead to even further damage.

The wireless network is the backbone of the entire smart farm operation which enables the farm to be efficiently managed. One of the limitations of the smart farm is that a lot of the components are battery-powered which means that the devices are operational for a finite period of time [14] [15]. In addition, the smart farm requires the UAVs to perform complex tasks on a limited battery power budget [16]. Many of the data collected from the smart farm are derived using convolutional neural network algorithms [17] [18]. IoT devices are limited in computing capacity and cannot perform every complex image recognition task. They need to offload computational tasks to nearby devices that have the processing capacity to execute the tasks. These image-processing tasks must be done in a time-critical manner since fire and pest detection are time-sensitive tasks. The wireless network's resources need to be energy-efficient to ensure that the network remains operational for as long as possible.

1.2 Contributions

This thesis explores several types of task distribution algorithms for a Fifth generation (5G) wireless network in a smart farm environment. It first takes a look at rule-based distribution algorithms. These algorithms employ fixed rules to determine where the tasks will be completed. The first rule-based algorithm introduced is round robin. Some of the rules are based on the known limitations of the system, these algorithms employ energy or computing resource heuristics to make their decisions. These algorithms are called Highest Energy First (HEF), and Lowest Queue Highest Energy First (QHEF). The contributions of this thesis are four reinforcement learning algorithms for task distribution in a smart farm environment. The first technique is a tabular Q-Learning method. This method employs Q-tables to determine which com-

puting resource in the network will compute the task. The second technique is a Deep Q-Learning (DQL) based algorithm that expands the state to include more transmission information. Unlike the first technique, it relied on Deep Learning models to predict the next destination of the task. The third algorithm is a Deep Risk-Sensitive (DRS) Q-Learning, where there is a separate state to determine the riskiness of the decision. This method employs deep learning to predict the possible reward and riskiness of each action. The final algorithm is the Risk Quantification (RQ) Q-Learning method where the riskiness of the action is quantified based on its potential damages. The damages are reflected in cost functions and a quantification measure was used to ensure that the agent does not select an action that leads to detrimental damages.

Publications

[P1] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Reinforcement Learning-Based Deadline and Battery-Aware Offloading in Smart Farm IoT-UAV Networks," in *ICC 2022 - IEEE International Conference on Communications*, pp. 189–194, 2022.

[P2] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Deep Reinforcement Learning for Task Offloading in UAV-Aided Smart Farm Networks," in *2022 IEEE Future Networks World Forum (FNWF)*, pp. 270–275, 2022.

[P3] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "To Risk or Not to Risk: Learning with Risk Quantification for IoT Task Offloading in UAVs," in *ICC 2023 - IEEE International Conference on Communications*, pp. 234–240, 2023.

1.3 Thesis Organization

The thesis is organized as follows. In this chapter, the purpose and environment of the thesis is introduced. Next, Chapter 2 establishes the background information used to understand the ideas presented in the thesis. Chapter 3 contains a literature review of the relevant studies to the topics introduced in this thesis. The system model is defined in Chapter 4. The proposed methods are introduced in Chapter 5. The simulation environment and re-

sults are presented in Chapter 6. Finally, the conclusion and future works are presented in Chapter 7.

Chapter 2

Background Theory

2.1 Introduction

The techniques presented in this thesis are task distribution algorithms for a 5G multi-access edge computing (MEC)-aided UAV and IoT smart farm network. The algorithms use reinforcement learning (RL) to decide where the task will be computed. One of the techniques employs financial risk evaluation methods as part of its learning process. This chapter will provide an introduction to wireless communication, financial risk evaluation techniques, and RL.

2.2 Wireless Communication

Wireless communication enables two or more devices to exchange data with one another without physical wires. Several types of mediums facilitate the exchange such as radio waves, microwaves, light, sonic, and electromagnetic induction. WiFi, radios, and Bluetooth are some examples of applications that employ wireless communication.

Another popular application of wireless communication is mobile networks. The First generation (1G) of mobile networks were deployed around

the 1980s, they used analog signals to support telephone communication. The usage of digital signals was introduced in the Second generation (2G) of mobile networks in the 1990s. This enabled text messaging and data transfer to be added to the list of services that mobile networks were capable of offering. The Third generation (3G) brought higher bandwidth capabilities and an international standardization of network protocols [19]. The theoretical downlink data rate for high-speed download packet access combined with multiple in multiple out is up to 84 Mbit per second [20]. The Fourth generation (4G) also increased the download and upload speeds from its predecessor. Long-term evolution (LTE) can have a downlink data rate of up to 300 Mbit per second and 75 Mbit per second for uplink [21]. The specifications for narrowband IoT were introduced in 3GPP's Release 13 [22].

2.2.1 5G

The next generation of wireless networks is 5G. It offers three types of services: enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communication (URLLC), and massive Machine Type Communication (mMTC) [23]. 5G networks employ millimeter waves (mmWave) which allow higher frequencies, which enables a high data rate. The Radiocommunication sector (ITU-R) of the International Telecommunication Union (ITU) set the minimum requirement for peak data rate at 20 Gbit per second for downlink and 10 Gbit per second for uplink [24]. The high data rate and low latency enable more computationally heavy real-time applications such as vehicle-to-vehicle communication [25], healthcare [26], and augmented reality [27]. Several machine learning algorithms have been employed to facilitate the services and applications offered by 5G [28] [29]. Algorithms such as convolutional neural networks and k-means clustering have been used for packet classification for efficient traffic management [30] [31]. RL has been used for scheduling network resources such as radio resources [32] [33].

2.3 Unmanned Aerial Vehicle - Multi-Access Edge Computing Network

In [34], Zeng et al. provided a survey on the benefits and challenges of integrating UAVs in a 5G and beyond network. One of the key traits of UAVs is their flexible 3-dimensional positioning. Although 5G offers a higher frequency than its predecessors, the mmWave has a shorter range. The 5G base stations will need to be placed close to the user, therefore the cell size will be smaller. UAV-mounted base stations can applied to provide better coverage as they can be easily moved and adjusted [35] [36].

Multi-access edge computing (MEC) provides computing services to users at the edge of the network. The survey papers [37–39] have identified that the use cases of MEC servers are computational offloading, caching, and smart cities. Furthermore, the authors in [40–42] have all proposed that in a 5G and beyond network, MEC devices can alleviate the UAV's workload through offloading. This results in the UAV conserving their energy as they do not have to compute as many computationally intensive tasks. In addition, UAV-MEC systems also lengthen the IoT devices' battery life through task offloading [43] because the IoT devices now have two sets of computing resources to offload their tasks. Therefore the addition of the MEC server promotes energy efficiency in the network as a whole.

2.4 Machine Learning in Wireless Networks

Machine learning is a branch of artificial intelligence that uses data and its own experience to teach itself to recognize patterns, make predictions, and select the best course of action. Machine learning consists of four approaches, supervised learning, unsupervised learning, semi-supervised learning, and RL. It has also been proven to be a useful tool for wireless networks. Sun et al. identified how various machine learning techniques were

used for managing resources, managing mobility, and localization in wireless networks [44]. Because machine learning techniques are adaptive and self-improving, they enable wireless networks to be self-organizing. Klaine et al. surveyed the different machine learning methods that enabled wireless networks to configure, optimize, and heal themselves automatically without human intervention [45]. Metha et al. reviewed machine learning techniques used in wireless sensor network applications [46]. IoT is another wireless network application that can leverage machine learning. Mahmood et al. introduced various use cases for machine learning algorithms in a 6G IoT network [47].

Supervised learning is a machine learning approach where the model learns from training data that is labelled. The authors in [48] and [49] employed neural networks for power management to maximize energy efficiency. Support vector machines were employed to predict localization for wireless sensor networks [50] and in an indoor setting [51] [52].

In unsupervised learning, the model's training data is not labelled. In Castro-Hernadez and Paranjape's work, the K-means algorithm was used to cluster time series data to predict user mobility at the cell edge to determine the optimal handover parameter in a heterogeneous network (HetNet) [53]. Similarly, Sinclair et al. also used unsupervised learning in [54] to optimize their handover process. They used an X-means self-organizing map to determine popular handover locations in a femtocell. Once these locations are defined, handovers are restricted to only those locations. In Chang et al.'s work, they used K-means in addition to a genetic algorithm to improve energy efficiency in a wireless sensor network [55]. Zhang et al. used K-means clustering to decipher the original signal from the received signal over an indoor mmWave channel [56].

In semi-supervised learning, the model's training data consists of both labelled and unlabelled data. In Huang et al.'s work, they wanted to classify wireless interference sources for an industrial IoT application [57]. They

identified that labelled data was scarce meanwhile unlabelled data was easier to obtain. They used temporal ensembling to label the unlabelled data, and then they trained a convolutional neural network based on the newly labelled data and the original labelled data. Semi-supervised learning can also be used in localization applications. Traditional localization methods require that the data be geo-tagged to accurately determine the location of the cellular device. Chakraborty et al. presented a probabilistic semi-supervised approach that used a maximum-likelihood estimation to label the unlabelled, and an expectation maximization model to predict the localization of cellular devices [58]. Camelo et al. introduced a semi-supervised deep autoencoder model to identify the technology that is trying to access a shared radio spectrum [59].

Unlike supervised, unsupervised, and semi-supervised learning, RL is a machine learning approach that does not require pre-existing training data. The learning model learns from its own experience and exploration. RL has many wireless applications. Iturria-Rivera et al. introduced two RL algorithms to optimize the handover policy for a dual connectivity urban network [60]. The objective of both algorithms is to improve latency for the user equipment in the network. Yao et al. proposed a UK-means deep RL algorithm for optimal beam formation and resource allocation for a 5G mmWave network [61]. Their algorithm initially clustered the users based on their location, then they used deep RL to direct radio resources to the clusters. Sarikhani et al. deployed deep RL for optimal spectrum management [62]. Their algorithm enabled two types of users to efficiently share a frequency band in a radio network. Saeidian et al. presented a deep RL algorithm to manage the downlink transmit power of all cells in a 5G urban macro network [63]. This thesis also investigates using RL for optimal resource management but will focus on energy and computing resources, and explores a risk-sensitive approach to RL.

2.5 Risk

The financial domain has offered many different risk evaluation methods. A risk is defined as a potential negative return in a financial setting. The risk evaluation metrics evaluate the potential losses for different investment portfolios from their previous returns. When the portfolio makes a profit during a period, then the return is positive. Conversely, if the portfolio incurs a loss, then the return for that period is negative. There are several methods used to measure risks, such as the Sharpe ratio, the Sortino ratio, and the Conditional Value at Risk (CVaR). Each method evaluates the risk of the portfolio based on the portfolio's set of previous returns. From the risk measurements, the investors can compare investments and make decisions on whether to invest given the investor's risk tolerance.

The Sharpe ratio [64] measures the risk of an investment portfolio by dividing the increase in return by the volatility of the portfolio. Assume an investor is considering investing in portfolio i and would like to calculate i's Sharpe ratio (S_i). First, it subtracts the return of a known safe investment (R_s) from the portfolio's expected return (R_i). Then the difference between the two portfolios is divided by the standard deviation of portfolio i's returns (σ_i).

$$S_i = \frac{R_i - R_s}{\sigma_i} \tag{2.1}$$

The difference between the two portfolios ($R_i - R_s$) indicates how much more the investor can gain from investing in portfolio i versus the safe investment s, and σ_i indicates the volatility of portfolio i's returns. Because σ_i is the standard deviation of all of portfolio i's returns, a high σ_i value indicates that portfolio i's returns are sparse. This indicates high volatility. Whereas a low σ_i value indicates that most of the returns are closer to the mean. A high Sharpe ratio indicates that portfolio i's return is significantly higher than the safe investment's return, and investment i's returns are not

very volatile. Therefore when comparing portfolios, a portfolio with a high Sharpe ratio is preferred [65].

The Sortino ratio [66] is an extension of the Sharpe ratio, but it has a different denominator. Instead of dividing by the standard deviation of all of portfolio i's returns, it uses the standard deviation of portfolio i's negative returns. Similarly to the Sharpe ratio, a high Sortino ratio is preferred because it indicates that portfolio i has a higher return than the safe portfolio, and portfolio i's negative returns do not have a large range.

CVaR is a method of evaluating risk by evaluating the average loss at a given percentile. CVaR was introduced by Rockafellar and Uryasev as a way to evaluate investment portfolios based on their expected shortfalls [67]. For example, given the losses of portfolio i, CVaR is the mean of the highest α percentile of portfolio i's losses. This enables investors to ensure that $(1-\alpha)$ percent of the portfolio's losses are below the CVaR.

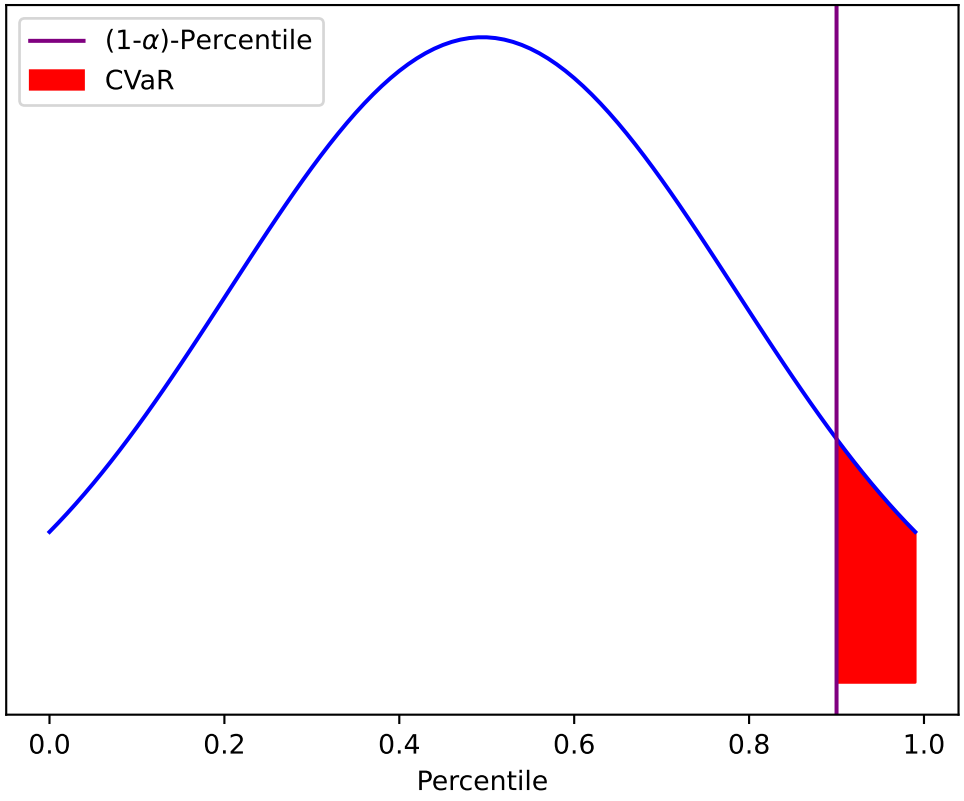


Figure 2.1: CVaR example

An investor can compare two portfolios by setting a fixed α level and calculating the corresponding CVaR values for both portfolios. In this case, the portfolio with a lower CVaR is preferred because it indicates that the portfolio's losses below the (1-α) percentile are below the CVaR.

2.6 Reinforcement Learning

RL is a machine learning technique where an agent is interacting in a changing environment. The agent performs actions that also influence changes in

the environment. After each action, the agent is evaluated based on how their action changed the environment. The evaluation is reflected as a reward. The objective of RL is for the agent to learn the series of actions that will lead to the highest long-term reward. There are five components in RL, the agent, action, state, environment, and reward. The agent performs an action in the environment and the action impacts the environment. The environment reacts and changes the state of the agent. After the action is performed in the environment, the agent is given a reward. The agent has a predetermined objective, and the reward is a response given based on how well the actions of the agent will achieve the objective. The reward and new state are fed to the agent and the agent performs a new action.

2.6.1 Q-Learning

Q-Learning is a type of RL where each action is measured by a metric called Q-values. These values predict the long-term value of taking action a at a given state s. It follows the Bellman's equation,

$$Q^{new}(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r + \gamma max(Q(s_{t+1}, a))), \tag{2.2}$$

where $Q(s_t, a_t)^{new}$ is the new Q-value for the state-action pair (s_t, a_t), α is the learning rate, r is the reward for action a_t, $Q(s_t, a_t)$ is the former Q-value, γ is the discount factor, and $max(Q(s_{t+1}, a_t))$ is the estimation of the Q-value for the next state s_{t+1}. α is a value between 0 and 1 that determines the level at which the agent values the new Q-value's prediction versus the old Q-value. γ is a value between 0 and 1 that indicates the significance of future rewards.

The Q-values for every state-action pair are stored in a Q-table. At first, the Q-table is empty and the agent takes random actions to build experience. This is considered to be the agent's exploration phase. After the agent takes

an action, the Q-table gets updated with Q-values based on the Bellman Equation. Once the agent has built up enough experience, the agent can use its previous experience to evaluate the action. Once the agent has run out of possible actions to perform or the agent has reached its goal, the episode is complete. All of the rewards that the agent received for every action are summed up. The more episodes the agent goes through, the better decisions they make, the episodic rewards increase, and the agent gets closer to the optimal solution. Eventually, the episodic reward will reach a point where it cannot increase. This is called convergence and it indicates that the agent has found the optimal solution.

Epsilon Greedy Policy

Epsilon greedy is an action selection policy. The agent uses this policy every time makes an action selection. In epsilon greedy, a random number between 0 and 1 is generated. If the number is less than the epsilon value, then the agent will select a random action. The agent will otherwise select the action based on the argmax of the Q-values. The epsilon value determines the probability of the agent exploring versus exploiting its previous experiences. At the beginning of the training period, epsilon is high. As the episodes go by, the epsilon value decreases and the agent relies more on the Q-values.

2.6.2 Deep Q-Learning

DQL is an extension of Q-Learning, but instead of using a Q-table to determine the Q-values for each state-action pair, they employ deep learning to predict the Q-values [68]. As the agent goes through a series of actions and encounters different scenarios, the neural networks train themselves with new experiences to get more precise in predicting the Q-values. To learn to predict the Q-values, the agent employs two neural networks and an experience replay. The experience replay is a buffer that contains tuples of the previous

state, action, and reward. The first neural network is called the target network. This network is trained using a random sample of experiences from the experience replay. It learns from the experiences and adjusts the parameters to accurately predict the Q-values. The second neural network is the main network. This network is used to generate the Q-values for each possible action for a state-action pair. Unlike the target network whose parameters are constantly changing, the main network's parameters remain fixed for a certain number of iterations, then the main network will copy the target network's latest parameters. The usage of experience replay and target networks stabilizes learning. One of the vulnerabilities of Q-Learning is that every action influences the Q-values of the subsequent actions after every timestamp. This makes Q-Learning vulnerable to abrupt Q-value changes especially due to unexplored actions, or inaccurate estimation. The experience replay buffer aids the agent in providing a more accurate Q-value estimation because the target neural network is trained using samples of existing experiences and uses those experiences to estimate the Q-value of unexplored actions. This minimizes the risk of an over or underestimation of Q-values because it will be based on previous experience. Also, the main network's parameters staying fixed means that it will not be influenced by sudden surges and downfalls of Q-values. In addition, the neural networks allow a larger state space without being bound by physical disk space like the Q-Learning method which is bounded because it must store all the Q-values for all state-action pairs.

2.6.3 Risk-Sensitive Reinforcement Learning

Risk-Sensitive RL involves training an agent to avoid known risky behaviors. In traditional RL algorithms, the agent has one objective, maximizing long-term rewards. The agent does not take into consideration the riskiness of an action, therefore it may take risky actions to achieve its objective. In real-world scenarios, these actions may lead to severe damage to the agent that can put it in a state that renders it unable to recover. A risk-sensitive agent

is not just trying to maximize the long-term reward but is aware of the risk associated with each action it takes to reap the rewards. After each action is executed, not only is the reward calculated, but the risk is also calculated. The agent is now trying to find a series of actions that lead to maximizing long-term reward, but also minimizing risk.

Chapter 3

Literature Review

3.1 Energy Minimization

Energy efficiency is an important consideration in wireless networks because many of the network components are battery-powered. Yang et al.'s objective was to minimize the total power consumption in a UAV-MEC network [69]. They defined four subproblems finding the optimal user association, power control, computation capacity allocation, and UAV placement. The subproblems were solved iteratively. There have been several studies that have the objective of lowering energy consumption and delay in a farm environment. In [70], Zhao et al. proposed using a network that consists of multiple hovering UAVs and Multi-access edge computing (MEC) devices to aid farm monitoring. The UAVs provided: connectivity between the IoT devices and a central processing unit (CPU) that can perform the image classification task. The UAVs have mounted CPUs and can compute the image processing tasks themselves or they can forward the task to a nearby MEC device to compute the task. Zhao et al. aimed to increase the throughput by considering the delay critical tasks in their smart farm environment [70].

There are several UAV trajectory planning approaches to minimize energy consumption. The authors in [71–74] present solutions that use trajectory

planning for UAVs to offload tasks from IoT to minimize the task completion delay as well as energy consumption in an IoT monitoring environment.

Yu et al. present utilizing UAVs to provide access to out-of-reach MEC services to IoT devices [75]. Their proposed algorithm positions the UAVs in a way that minimizes service delays and maximizes the UAV's energy efficiency. They use convex approximation to find the optimal solution to their problem.

Ghdiri et al. provided a cluster-based approach for deadline-aware task computation for a UAV-IoT network [76]. The objective of their proposed solution is to find the optimal trajectory for the UAVs to collect data from IoT devices that are energy efficient for the UAV and respect the IoT data's deadlines. Firstly, their algorithm clusters the IoT, and each cluster head is a point that the UAV has to reach. Next, the algorithm determines the UAV and the most energy-efficient way to visit the cluster heads.

Using RL to manage wireless network resources to optimize performance is widely studied across many different next generation wireless network applications. In [77], Elsayed et al. surveyed the challenges and opportunities of AI in 5G networks and beyond. They introduced multiple applications for RL in wireless networks such as energy management. Furthermore, Khoramnejad et al. [78] proposed a deep RL approach to solve a joint optimization problem consisting of maximizing computation and minimizing energy consumption for a 5G and beyond network through offloading. Their network also makes use of MEC servers as a processing unit to assist their network in computing-intensive tasks. Chen et al. also proposed a deep RL task for resource allocation to minimize energy consumption in a MEC system that supports augmented reality [79].

3.2 Task offloading

Xu et al. solved a task offloading problem in a Non-orthogonal multiple access (NOMA) heterogeneous network with a MEC server [80]. The objective of the presented algorithm was to reduce energy consumption for all users. The objective jointly considered task offloading, local CPU frequency scheduling, power control, and subchannel resource allocation. They divided the problem into two sub-problems, namely offloading and resource allocation. Their proposed algorithm solved these two sub-problems via mixed integer linear programming. This thesis is also proposing a task-offloading algorithm to reduce energy consumption and the proposed network also utilizes a MEC server. Unlike the network proposed in this paper, the proposed network has UAVs with finite power supplies which adds a different element to the power consumption.

In [81] they introduced a multi-tier structure as a task-offloading solution for next generation wireless networks. In their proposed structure there are three layers, the cloud, the network edge, and the UEs. The network edge is between the cloud and the UEs. The advantage of this structure is that the low-latency tasks can be done at the edge server, meanwhile, the tasks that are computationally intensive and do not require low latency can be done by the cloud server. Wang et al. propose multiple types of network applications that could benefit from the multi-tier structure as well as future areas of research [81].

Utilizing RL for task-offloading is another emerging topic in UAV-based networks. Ebrahim et al. [82] provided a tradeoff between delay and energy by optimizing offloading decisions using deep RL. Sacco et al. [83] aimed to reduce the required information and training time for an RL-based task-offloading solution. Zhao et al. [84] studied on multi-agent TD3 approach to address their continuous action space problem. Yang et al. [85] focused on service cost minimization by optimizing the joint task-offloading and time-division multiple access-based channel allocation decisions.

To address the growing demand for internet usage, Kiran et al. [86] introduce a software-defined network (SDN) with MEC servers that use Q-Learning to optimize the network's delay time. The SDN has three layers: the control layer, the edge computing layer, and the infrastructure layer. They present a Q-Learning algorithm to allocate resources in the SDN network to match the UE with the best resource in the SDN, based on the UE's requirement. The paper demonstrated that the Q-Learning algorithm where the agents were cooperative and shared their Q-values provided the best resource allocation solution. This cooperative solution fits their application because it is a centralized system where there is a central location that knows all the Q-values. Our Q-Learning solutions use agents that are independent from one another.

Alfakih et al. presented a DQL method to allocate MEC-assisted network resources to mobile users in a cyber-physical-social system [87]. Their system is used to detect traffic violations. The mobile device is the agent and must decide the destination where the task will be performed. The optimal solution will send the task to a destination that will reduce energy consumption and delay. The task can be done by the mobile device itself, or sent to the nearest edge server, an adjacent server, or a remote cloud. This paper is solving a similar problem to ours but for a different application. They also use DQL to find a task distribution solution that will minimize energy consumption and delay. The agent is the mobile user which is also the end user. Unlike [87], the system presented in this thesis assigns the UAV to be the agent and the middleman.

Zhang et al. also proposed a DQL algorithm for task offloading in a wireless network that employs a MEC server [88]. They present a task-offloading algorithm for an autonomous vehicle network that needs to process a large amount of sensory data for smart navigation. They use MEC servers to alleviate some of the processing, however, they need to schedule the task to efficiently utilize the MEC servers and ensure that the processing tasks are

successfully delivered to the MEC servers. Their task-offloading algorithm needs to select the best MEC server to perform the task sent from the vehicle in a given time frame.

In another study, Zhou et al. included satellite computation and communication in their IoT-UAV network [89]. The objective of their work is to find an optimal task scheduling policy that reduces latency while considering the energy constraint of the UAVs. They modelled their problem as a constrained Markov decision process (MDP) and provided a deep RL solution.

Likewise, Akbari et al. [90] introduced a deep RL algorithm in an industrial IoT environment. The algorithm aimed to find an optimal placement and scheduling policy for virtual network functions to minimize the Age of Information (AoI) and cost. The age of information (AoI) is calculated on the receiver's end, and it is the time progressed since the most recently received packet was generated. Ideally, the AoI should be as low as possible, to ensure that the receiver has received the most updated information from the IoT. In a delay-sensitive dynamic environment such as industrial IoT, minimizing the latency is crucial.

3.3 Financial Risk Measurement for Resource Management

In wireless networks, financial concepts like risk can be used to evaluate and model behaviors to find an algorithm that avoids undesirable behaviors. Apandi et al. [91] use the Sharpe ratio as a decision-making parameter for base stations to use to find an optimal way to find user associations to send subcarriers. Apandi et al. presented a decision-making algorithm that employed the Sharpe Ratio to aid the base station in determining the appropriate user association and subcarrier allocation for downlink users in HetNet. In HetNets, there are base stations with different levels of transmit power to

provide coverage for an area. For a user to receive service in the area, it must be associated with a nearby base station, and the base station will allocate subcarriers to the user. Traditionally, the user associates with the nearest base station with the highest transmit power. This causes an imbalance between the usage of base stations in the network. Base stations with high transmit powers, such as macro base stations, were over-utilized, meanwhile, the base stations with low transmit powers were available. In their version of the Sharpe Ratio, the numerator is the mean of the UE's achievable rates, and the denominator is the standard deviation of the achievable rates. The objective is to determine the distribution of subcarrier allocation that will maximize the Sharpe Ratio by increasing the mean user rate and lowering the standard deviation of the user rate. A lower standard deviation would indicate more fairness among the base stations because there is less discrepancy between the user rates. Each base station has a finite number of subcarriers, and a subcarrier can only be assigned to one user. All of the users within the area of a base station will broadcast their achievable user rate to the base station. Each base station's subcarrier will calculate the probability that this user association will maximize the Sharpe Ratio. Finally, the subcarrier selects the user by selecting the user with the highest probability.

CVaR has been used to manage wireless networks. Authors have identified some undesirable behavior in their network they wish to avoid such as high energy consumption, and they use CVaR to avoid such behavior because CVaR gives us the ability to define a lower bound for the worst percentile of cases. Optimizing using CVaR ensures that the majority of cases will be better than the CVaR.

In Wang's work [92], they introduce a risk-aware Asynchronous Advantage Actor Critic (A3C) task-offloading algorithm that aims to minimize the total latency for their fiber wireless heterogeneous network (FiWi HetNet). This network consists of IoT devices that connect wirelessly to one another and MEC servers. The wireless channel is noisy and there is potential for

packet drops. The MEC servers connect to a single edge-cloud server via fiber optic lines. The IoT devices have a series of tasks to perform and they can perform the tasks locally or offload the tasks to the edge cloud server. The authors define risk as exceeding the task's delay requirements due to packet loss or overload of the edge cloud server. They use CVaR to evaluate the probability that the task's latency exceeds the delay requirement. The CVaR was one of the constraints in their A3C task-offloading decision-making algorithm.

In Ling's work [93], they introduce a block coordinate descent algorithm to find the optimal task-offloading policy for a wireless body area network (WBAN). Their network consists of several body sensors and a multi-access edge computing server. The body sensors collect and send signals that contain physiological information. They send the signal to an access point and the access point relays the signal to the MEC server to perform analysis. The access point also can send energy to the energy-constrained sensors. According to the paper, the sensors follow a "frame-based protocol". The frame is then further divided into time slots. In each time slot, the sensor can either offload the task to the access point or receive energy from the access point. The objective of the proposed algorithm is to schedule the sensors' activities for each time slot, such that the tasks' latency is minimized and respecting the energy constraints. They defined their problem using a "distributionally robust chance-constrained problem" where their energy is constrained [93]. The authors used CVaR to define the greatest lower bounds for their energy constraints.

In Zhou's work [94], they presented an IoT monitoring system, which consists of an IoT device and a receiver. The IoT device surveys a physical process, and it then generates and sends packets containing information on the status of the physical process to a receiver. The IoT device sends the packet via a noisy wireless channel. Upon successfully receiving the packet, the receiver will send feedback to the IoT device through a noiseless chan-

nel. The authors used the packet's AoI to measure the novelty of the status update. The IoT device must decide whether or not to send a status update to the receiver. Transmitting a packet consumes energy, so the device does not want to send a status update too often, however, if it waits too long in between updates, the receiver will not have the most up-to-date information on the status. The paper proposed a risk-aware dynamic programming algorithm to determine if the IoT device should send a status packet to the receiver. The algorithm jointly minimizes total energy cost, the AoI of the packets, and the CVaR of the packets' AoI. They considered long AoIs to be a risk and used CVaR to measure the average of longest AoIs at a given percentile. If they can minimize the CVaR of the AoIs, they can ensure that the majority of the AoIs will be below the CVaR.

In Hoiles' work [95], they present two policies that use CVaR to estimate content demand in femtocell networks. The objective of the paper is to have a policy that will reduce the delay in delivering YouTube content to users in a femtocell network. To reduce the delay, they cache the content in local servers based on demand. To accurately select the server, they needed to use a CVaR-based prediction function. CVaR was used to determine the uncertainty when it came to the content request prediction. Through using CVaR, they were able to provide a prediction with a "probabilistic guarantee" [95].

Chapter 4

System Model

The farm environment consists of UAVs hovering over farmland. They are fixed in the air hovering above a set of IoT devices deployed across the land. The IoT devices can perform monitoring tasks like image classification to detect fire and pests, and monitor crop growth. To have accurate image classification, the current state of the art requires transformer-based techniques [96]. For detecting agriculture-based images, convolutional neural networks yielded the best results [97]. These techniques are computationally heavy and require a lot of memory space. Therefore to alleviate some of the computational burdens, the IoT devices offload the tasks to the nearby hovering UAVs. The UAVs are equipped with resources that have higher computing capacities than the IoT. Similar to the IoTs the UAVs have limited power and computing resources therefore they also can offload their tasks. The UAVs can offload to another UAV or a MEC node.

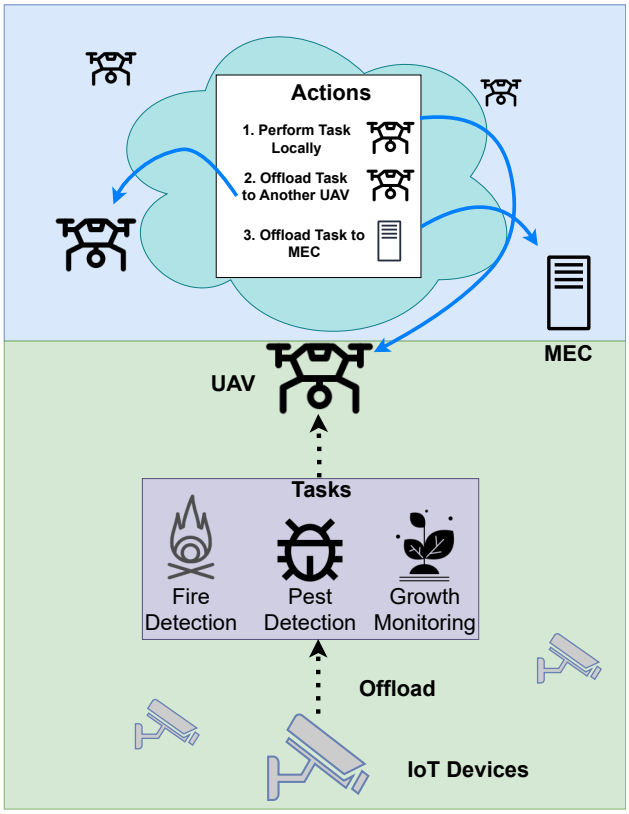


Figure 4.1: System model of smart farm environment

4.1 Farm Environment

4.1.1 IoT

The IoT devices are deployed on the ground throughout the farmland. They monitor the crops by periodically taking snapshots of the crops and performing image classification tasks. They run on batteries and have limited computing power. To extend their batteries' longevity, they offload the image classification tasks to the closest UAV. Every task offloaded by the IoT ($\alpha^{B}_{\langle j,t \rangle} = 1$) at the time interval t, will be allocated to a CPU in the network

Table 4.1: Summary of the Smart Farm Notations

Sets	**Size**	**Description**
$t \in \mathcal{T}$	T	Set of time intervals
$j \in \mathcal{J}$	J	Set of UAVs
$l \in \mathcal{L}$	L	Set of MEC devices
$j' \in \mathcal{J}^+$	J+L	Set of CPUs
$k \in \mathcal{K}$	K	Set of task types
UAV Parameters	**Range**	**Description**
$\Delta_{j'}$	$\mathbb{R}$	CPU's current queue time
$\Delta^{j_P}_{j'_R}$	$\mathbb{R}$	Transmission delay between computing resources
Task Parameters	**Domain**	**Description**
$\gamma^D_{\langle j,t \rangle}$	$\mathbb{R}$	Deadline
$\alpha^B_{\langle j,t \rangle}$	$\{0,1\}$	Indicator for task offloading from IoT
$x_{\langle j,t \rangle j'}$	$\{0,1\}$	Indicator that task's computation destination is j'
$p_{\langle j,t \rangle j't'}$	$\{0,1\}$	Indicator that task is being processed by j' at time interval t'
$p^+_{\langle j,t \rangle j't'}$	$\{0,1\}$	Indicator that task is being processed by j' for the first time at time interval t'
$p^-_{\langle j,t \rangle j't'}$	$\{0,1\}$	Indicator that task has finished being processed by j' at time interval t'
$v_{\langle j,t \rangle}$	$\{0,1\}$	Deadline violation indicator
Delay Parameters	**Range**	**Description**
$\Delta_{\langle j,t \rangle}$	$\mathbb{R}$	Task's end-to-end delay
$\Delta^Q_{\langle j,t \rangle}$	$\mathbb{R}$	Task's queuing delay
$\Delta^P_{\langle j,t \rangle}$	$\mathbb{R}$	Task's computation delay
Energy Parameters	**Values**	**Description**
Υ^B_j	$\mathbb{N}$	Battery capacity
Υ^C_j	$\mathbb{R}$	CPU energy consumption
Υ^R_j	$\mathbb{R}$	Remaining energy in a battery

Eq. (4.1). $x_{\langle j,t\rangle j'}$ is equal to one when the CPU j' processes the task. $p_{\langle j,t\rangle j't'}$ is equal to one when the task is processed in the time interval t'

$$\sum_{t'=t}^{T} \sum_{j' \in \mathcal{J}^+} p_{\langle j,t\rangle j't'} * x_{\langle j,t\rangle j'} = \alpha^B_{\langle j,t\rangle} * \delta^P_{\langle j,t\rangle}, \quad \forall j \in \mathcal{J}, \forall t \in \mathcal{T} \tag{4.1}$$

4.1.2 UAV

The UAVs hover in a fixed position over the farmland in the air. They collect offloaded tasks from the IoT devices and other neighbouring UAVs. The UAVs contain computing resources that can perform computationally heavy tasks. They can also offload tasks to any UAVs or MEC devices. When the UAVs receive a task from an IoT device, they have three options. They can perform the task themselves. They can offload the task to another UAV. The neighbouring UAVs are also limited by their battery power and computing queue. Finally, the UAV can offload the task to a MEC device.

The ping-pong effect occurs when a task is offloaded from node to node and never actually gets executed. To avoid the ping-pong effect, Eq. (4.2) enforces a constraint where if a UAV has received a task that was offloaded from another UAV, the last UAV must perform the task. It cannot offload the task to another node.

$$\sum_{j' \in \mathcal{J}^+} x_{\langle j,t\rangle j'} \leq 1, \qquad \forall j \in \mathcal{J}, \forall t \in \mathcal{T} \tag{4.2}$$

Eq. (4.3) indicates that a UAV must complete the entire task by itself.

A single task cannot be divided amongst multiple UAVs or MEC devices.

$$\sum_{j' \in \mathcal{J}^+} p_{\langle j,t \rangle j' t'} \leq 1, \qquad \sum_{j' \in \mathcal{J}^+} p^+_{\langle j,t \rangle j' t'} \leq 1, \qquad (4.3)$$
$$\sum_{j' \in \mathcal{J}^+} p^-_{\langle j,t \rangle j' t'} \leq 1, \qquad \forall j \in \mathcal{J}, \forall t' \in \mathcal{T}, \forall t \in \mathcal{T}$$

With a three-dimensional range of motion, UAVs offer a wide range of services. They can provide line-of-sight connectivity between the IoT devices, and other UAVs or MEC devices. They can also provide computing capabilities. However, UAVs are limited by their battery capacities. If the UAV is performing too many image classification tasks, their hover time will be greatly reduced.

4.1.3 MEC

The MEC device is a computing resource that remains fixed on the ground. It has higher computing power than the UAVs, therefore the tasks are computed faster. It also has unlimited power because there is an assumption that it is connected to the power grid. The MEC device cannot offload a task, it must perform all the tasks that it has received.

Eq. (4.4) constrains the CPU of the UAVs and MECs' devices such that they can only perform one task in one time interval t.

$$\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} p_{\langle j,t \rangle j' t'} \leq 1, \qquad \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} p^+_{\langle j,t \rangle j' t'} \leq 1, \qquad (4.4)$$
$$\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} p^-_{\langle j,t \rangle j' t'} \leq 1, \qquad \forall j' \in \mathcal{J}^+, \forall t' \in \mathcal{T}$$

Furthermore, Eq.(4.5) indicates that the CPUs cannot start a new task before they have completed their current task.

$$p_{\langle j,t \rangle j' (t'+1)} = p_{\langle j,t \rangle j' t'} + p^+_{\langle j,t \rangle j' (t'+1)} - p^-_{\langle j,t \rangle j' (t'+1)} \qquad (4.5)$$

Eq.(4.6) restricts the start time of the computing devices so that they can only begin processing a task at the beginning of the simulation.

$$p_{\langle j,t\rangle j'(0)} = p^{+}_{\langle j,t\rangle j'(0)} \tag{4.6}$$

The UAVs and MEC devices cannot redo a completed task as specified in Eqs. (4.7) and (4.8).

$$\sum_{t' \in \mathcal{T}} p^{+}_{\langle j,t\rangle j't'} \leq 1, \qquad \forall j \in \mathcal{J}, \forall t \in \mathcal{T}, \forall j' \in \mathcal{J}^{+} \tag{4.7}$$

$$\sum_{t' \in \mathcal{T}} p^{-}_{\langle j,t\rangle j't'} \leq 1, \qquad \forall j \in \mathcal{J}, \forall t \in \mathcal{T}, \forall j' \in \mathcal{J}^{+} \tag{4.8}$$

4.1.4 Communication Relationship in IoT-UAV-MEC Network

As shown in Figure 4.2 The UAVs are able to reach any other UAV in the network. All UAVs are capable of reaching the MEC devices to offload their tasks. The IoT devices are only able to offload their tasks to a nearby UAV.

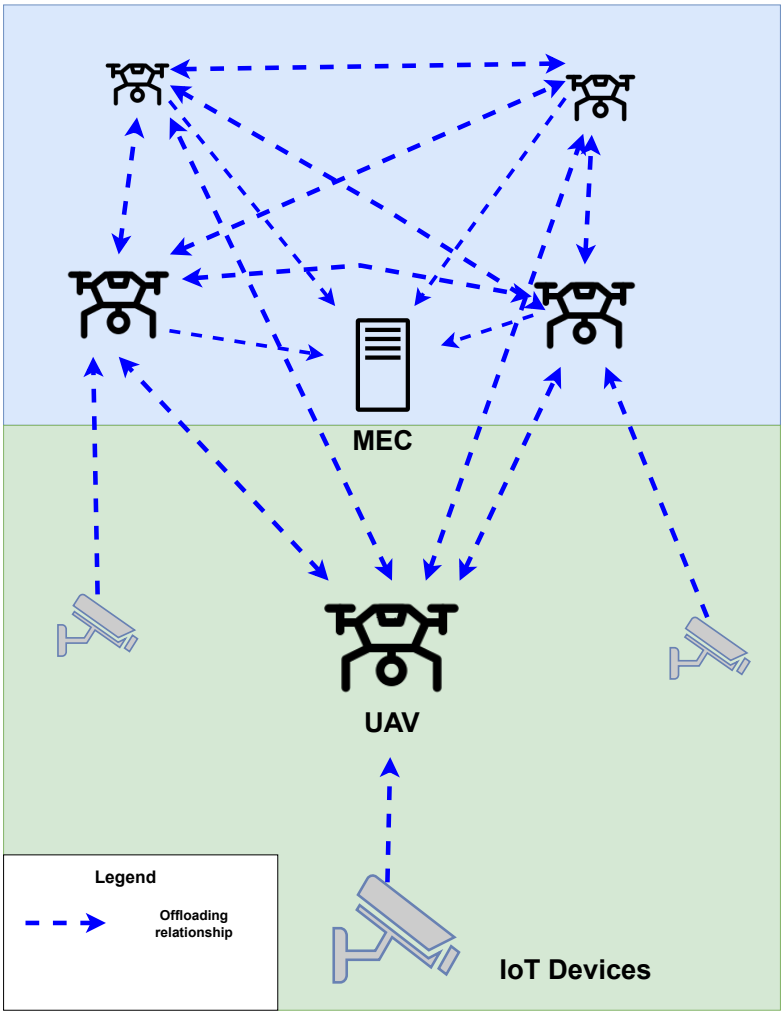


Figure 4.2: IoT-UAV-MEC communication model

4.2 Energy

We would like the MEC-assisted UAV network to last as long as possible. The decision-making algorithm needs to take into consideration the current energy levels of each processing unit, specifically those with a limited amount of energy. Allowing a computing resource to process too many tasks can lead

it to prematurely be out of commission. The UAVs are battery-operated, therefore they have finite energy. The current energy level is the percentage of the battery that the UAV has at time t.

Their current battery level at time $\mathcal{T}$ can be modeled using the following equation,

$$\Upsilon^R_{j'} = \Upsilon^B_{j'} - (\Upsilon^H_{j'} + \Upsilon^A_{j'} + \Upsilon^I_{j'}) * \mathcal{T} - \sum_{\substack{j \in \mathcal{J} \\ t \in \mathcal{T} \\ t' \in \mathcal{T}}} (\Upsilon^C_{j'} - \Upsilon^I_{j'}) * p_{\langle j,t \rangle j' t'}, \tag{4.9}$$

where $\Upsilon^B_{j'}$ is the initial battery level for UAV j', $\Upsilon^H_{j'}$ is the amount of energy required for the UAV to fly above the farm, $\Upsilon^A_{j'}$ is the amount of energy required by the antenna to send signals, $\mathcal{T}$ is the total simulation time, $\Upsilon^I_{j'}$ is the amount of energy required for the CPU to be idle, and $\Upsilon^C_{j'}$ is the amount of energy required for the CPU to run a task. To indicate whether or not UAV j' computed the task, there is a binary indicator $p_{\langle j,t \rangle j' t'}$, it will equal 1 when the task $\langle j, t \rangle$ was computed at UAV j' at time interval t'.

4.3 End-to-End Delay

The end-to-end delay $\Delta_{\langle j,t \rangle}$ is the total time it takes for a task to be completed by a computing resource, and is defined as

$$\Delta_{\langle j,t \rangle} = \Delta^{j_R}_I + \Delta^{j_P}_{j_R} + \Delta^Q_{\langle j,t \rangle} + \Delta^P_{\langle j,t \rangle}. \tag{4.10}$$

It is the sum of the delay to transmit the task from the IoT device I to the initial UAV j_R ($\Delta^{j_R}_I$), the delay to transmit the task from j_R to the computing resource that will perform the task j_P ($\Delta^{j_P}_{j_R}$), the delay of the task waiting in j_P's queue ($\Delta^Q_{\langle j,t \rangle}$), and the time it takes for the task to be computed by j_P ($\Delta^P_{\langle j,t \rangle}$).

4.4 Deadline Violation

Every task has its own predetermined deadline ($\gamma^D_{\langle j,t\rangle}$), this is the maximum time it takes for the task to be completed. As shown in Eq. (4.11) when the task's end-to-end delay exceeds the deadline, a deadline violation has occurred ($\eta = 1$).

$$v_{\langle j,t\rangle} = \begin{cases} 0, \text{ if } \Delta_{\langle j,t\rangle} \leq \gamma^D_{\langle j,t\rangle} \\ 1, \text{ otherwise} \end{cases} \tag{4.11}$$

Exceeding the task's deadline has varying levels of consequences depending on the task's type. For example, if the task is a fire task, then exceeding the deadline will lead to the fire going undetected. This can cause irreparable damages to the farmland such as loss of crops.

Chapter 5

Proposed Methodologies

5.1 Round Robin

Round Robin (RR) is a decision-making technique that only considers the order of the computing resource. All computing resources are placed in an ordered list. The tasks will be sent to the computing resource based on the computing resource's order. The destination where the task will be sent is based on where the previous task was sent. The task will be sent to the next computing resource on the list. Once the end of the list has been reached, the order will go back to the beginning and the task will be sent to the first computing resource on the list. This algorithm ensures that the MEC device and all of the UAVs receive an equal number of tasks.

5.2 Highest Energy First

The Highest Energy First (HEF) decision-making technique only considers the computing resources' remaining energy levels. All of the UAVs regularly update one another with their remaining battery levels. When a UAV receives a task from an IoT device, it first finds the computing resource with the highest remaining energy. Then it determines the difference between the

highest remaining energy level and the current UAV's energy level. If the difference is greater than the 1% threshold, then the task will be offloaded to the computing resource with the highest remaining energy, otherwise, the task will be computed locally. The 1% difference is to show that the highest remaining energy is significantly higher than the current UAV's energy level. If the current UAV's energy level is the same as the highest remaining energy level, then the task should be completed locally because otherwise, the network will spend communication resources to offload a task that can be done at a destination that has the same remaining battery level. There is an assumption that the MEC devices will have an unlimited amount of power, therefore to prevent a situation where all the tasks are sent to MEC, the number of times a task can be sent to MEC is limited to 20%.

Algorithm 1 Highest Energy First Offloading

```
1: destination = random(1, (J + 1))
2: if destination == MEC then
3:     destination = MEC's ID
4: end if
5: highestEnergy = current UAV's remaining energy
6: for i = 1 .. J do
7:     remainingEnergy = UAV_i's remaining energy level
8:     if remainingEnergy > highestEnergy then
9:         highestEnergy = remainingEnergy
10:        destination = UAV_i's ID
11:    end if
12:    if highestEnergy - currentEnergy < threshold then
13:        destination = current UAV's ID
14:    end if
15: end for
```

5.3 Lowest Queue Highest Energy First

The Lowest Queue Highest Energy First (QHEF) technique considers both the computing resources' queuing time as well as the remaining energy level in their offloading decision-making. Similar to the HEF technique, all of the UAVs regularly update one another with their current remaining energy levels and queue time. Queue time is considered to be the total time it will take for the computing resource to compute all of the tasks currently in its queue. When a UAV receives a task from an IoT device, it first finds the computing resource with the lowest queue time. If the lowest queue time is 0.5 seconds lower than the current UAV's queue time, then that computing device can be a potential offloading destination. Next, the algorithm will find the computing resource with the highest remaining energy level among the potential offloading destinations. If the current UAV can find a computing resource that meets all of the conditions above, then the task will be offloaded, otherwise, the task will be computed locally.

Algorithm 2 Lowest Queue Time and Highest Energy First Offloading

```
1: destination = Current UAV's ID
2: lowestQueue = Current UAV's queue delay
3: highestEnergy = Current UAV's remaining energy
4: // find the UAV/ MEC with the lowest queuing delay
5: for i = 1 .. (J + 1) do
6:     queue = UAV_i or MEC's queue delay
7:     if queue < lowestQueue then
8:         lowestQueue = queue
9:         destination = UAV_i's ID
10:    end if
11: end for
12: // Check to see if the difference between the queuing delays is significant
13: if (qDelay - lowestQueue) < delayThreshold) then
14:    destination = current UAV
15:    highestEnergy = current UAV's energy
16: end if
17: // find the UAV/ MEC with the highest remaining energy
18: for i = 1 .. (J + 1) do
19:    energy = UAV_i or MEC's energy
20:    if energy > highestEnergy then
21:        highestEnergy = energy
22:        destination = UAV_i's ID
23:    end if
24: end for
25: // Check to see if the difference between the energy levels is significant
26: if (highestEnergy - current UAV's energy) < energyThreshold) then
27:    destination = current UAV's ID
28:    lowestQueue = current UAV's queuing delay
29:    highestEnergy = current UAV's energy
30: end if
```

5.4 Q-Learning

5.4.1 Objective

The Q-Learning algorithm has a multi-objective function, it consists of two objectives: minimizing the minimum remaining energy, $\Upsilon^R_{j'}$, and the total number of deadline violations $v_{\langle j,t\rangle}$.

Maximize:

$$W * \min_{j' \in J} \Upsilon^R_{j'} - \frac{1-W}{\Theta} \sum_{\substack{j \in \mathcal{J} \\ t \in \mathcal{T}}} v_{\langle j,t\rangle} \tag{5.1}$$

5.4.2 State

The state consists of the task type (k), all of the computing resources' current CPU delays ($\Delta_{j' \in \mathcal{J}^+}$), and all of the UAVs' current battery levels ($\Upsilon^L_{j' \in \mathcal{J}}$). The state can be defined as

$$\mathbb{S}_{QL} = \{k, \Delta_{j' \in \mathcal{J}^+}, \Upsilon^L_{j' \in \mathcal{J}}\}. \tag{5.2}$$

5.4.3 Action

Each UAV is an independently trained agent that determines the destination where the task will be computed. Therefore the action space of each agent can be defined as

$$\mathbb{A} = \{x_{j' \in \mathcal{J}^+}\}, \tag{5.3}$$

where x is the computing resource that will execute the task. There are three possible options for x. The task can be computed locally, be offloaded to another UAV, or be offloaded to a nearby MEC server. The agent uses an epsilon greedy policy to select the action based on the Q-value of each

potential action calculated by Eq. (2.2) at the current state.

5.4.4 Reward

The reward function (R_{QL}) is defined as follows,

$$R_{QL} = (\Upsilon^L_{j_a} - 1) + (1 - \mathbb{E}(v_{j_a,t})) + \mathcal{V}^L_{j_a} * \mathbb{E}(v_{j_a,t})) \tag{5.4}$$

$$\Upsilon^L_{j_a} = \begin{cases} 2, & \text{if } \mathbb{E}(\Upsilon^R_{j_a}) - \max_{j' \in \mathcal{J}}(\mathbb{E}(\Upsilon^R_{j'})) \geq -\epsilon \\ 0, & \text{if } \mathbb{E}(\Upsilon^R_{j_a}) - \max_{j' \in \mathcal{J}}(\mathbb{E}(\Upsilon^R_{j'})) \leq -2 * \epsilon \\ 1, & \text{otherwise,} \end{cases} \tag{5.5}$$

$$\mathcal{V}^L_{j_a} = \begin{cases} -40, & \text{if } \mathbb{E}(v_{j_m,t})) = 0 \\ -20, & \text{if } \mathbb{E}(v_{j_r,t})) = 0 \\ -10, & \text{if } \exists j' \in (\mathcal{J}/(j_r \cup j_a))(\mathbb{E}(v_{j't})) = 0 \\ -1, & \text{otherwise.} \end{cases} \tag{5.6}$$

In the battery level term of the reward ($\Upsilon^L_{j_a} - 1$), if the difference between maximum battery capacity and the expected battery level is smaller than threshold e, then the battery level reward will be equal to 1. If the difference between the maximum battery level and the expected battery level is greater than or equal to $2e$, then the battery level reward will be equal to -1. When the difference between the maximum battery capacity and the expected battery level belongs to $[e, 2e]$, then the battery level reward will be 0.

The second portion of the reward ($(1 - \mathbb{E}(v_{j_a,t})) + \mathcal{V}^L_{j_a} * \mathbb{E}(v_{j_a,t}))$) depends on whether the action leads to a deadline violation. $\mathbb{E}(v_{j_a,t}))$ indicates to us if a deadline violation has occurred. $\mathbb{E}(v_{j_a,t}))$ equals 1 if a deadline violation has occurred and equals 0 otherwise. $\mathcal{V}^L_{j_a}$ determines the severity of the negative reward for a deadline violation. The level of severity depends on whether

a deadline violation could have been avoided with a different action. If a deadline violation is inevitable, then the severity is low. If the MEC device was idle, the agent should be encouraged to send the tasks to be completed at the MEC device. Therefore if a deadline violation could have been prevented if the task was sent to the MEC device, the most severe penalty is assigned. If the deadline violation could have been prevented by not offloading the task in the first place, the agent will receive the second-highest penalty. There were communication and computational resources that did not have to be used on a task that could have been performed locally. Finally, if a deadline violation could have been prevented by sending the task to any other UAV, the agent would receive the third-highest penalty.

5.5 Deep Q-Learning

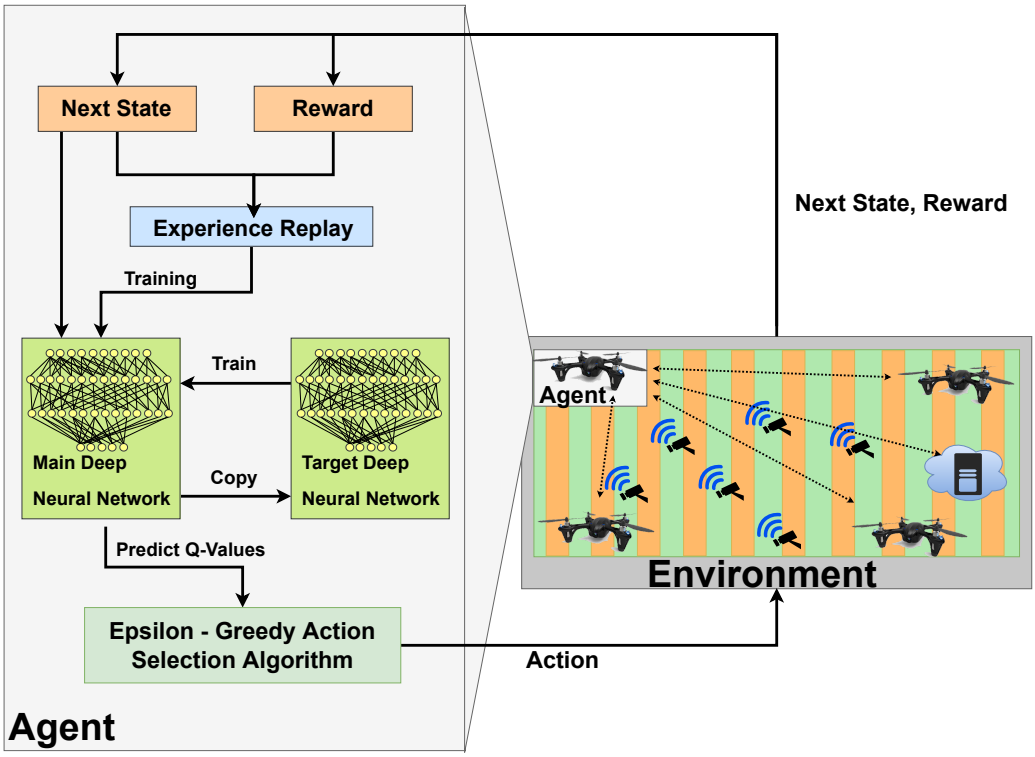


Figure 5.1: Deep Q-Learning smart farm system

5.5.1 Objective

The Q-Learning algorithm was extended by replacing the Q-tables with deep neural networks (DNN). Because it is an extension, the objective, action, and reward function remain the same.

5.5.2 State

Because DQL uses DNN instead of a Q-table to determine the Q-values of each action, the state space seen in Eq. (5.2) was expanded. The state of

the DQL algorithm contains task's type k, all computing resources' current queuing delay $\Delta_{j' \in \mathcal{J}^+}$, all of the UAVs' current battery level $\Upsilon^L_{j' \in \mathcal{J}}$, and all transmission delays between all computing resources $\Delta^{j_P \in \mathcal{J}^+}_{j'_R \in \mathcal{J}^+}$. The state is defined as,

$$\mathbb{S}_{DQL} = \{k, \Delta_{j' \in \mathcal{J}^+}, \Upsilon^L_{j' \in \mathcal{J}}, \Delta^{j_P \in \mathcal{J}^+}_{j'_R \in \mathcal{J}^+}\}. \tag{5.7}$$

5.5.3 Neural Network Architecture

The architecture of the DNN used in the DQL algorithm consists of an input layer with 10 neurons, two hidden layers with 32 neurons each, and an output layer with 5 neurons. The hidden layers both used the ReLU activation function. The DNN used the Mean Squared Error function for the loss function. The optimization function was Adam Optimizer with a learning rate of 0.001.

5.5.4 Reward

The reward function for the DQL algorithm is shown in Eq. (5.4).

5.6 Deep Risk Sensitive Q-Learning

DRS Q-Learning is an extension of the work of Pamuklu et al. in [98]. The objective function consists of three objectives: maximize the hovering time of the entire UAV network, minimize the uplink delay for processing a task, and guarantee that the total number of deadline violations will not exceed a specified threshold. To achieve those objectives, the problem definition is formulated as

Maximize: (5.8)

$$W * \min_{j' \in J} \Upsilon^R_{j'} - \frac{1 - W}{\Theta^M} \delta$$

Subject to: (5.9)

$$\sum_{\langle j,t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle} v_{\langle j,t \rangle} \leq \mathbb{V},$$

where $\Upsilon^R_{j'}$ is the current remaining battery level for UAV j', δ is the mean uplink delay, $v_{\langle j,t \rangle}$ is a binary indicator of whether a task has violated its deadline, and $\mathbb{V}$ is the maximum number of total deadline violations allowed. The weighting factors for consideration of the hovering time, and mean delay are W and Θ^M respectively.

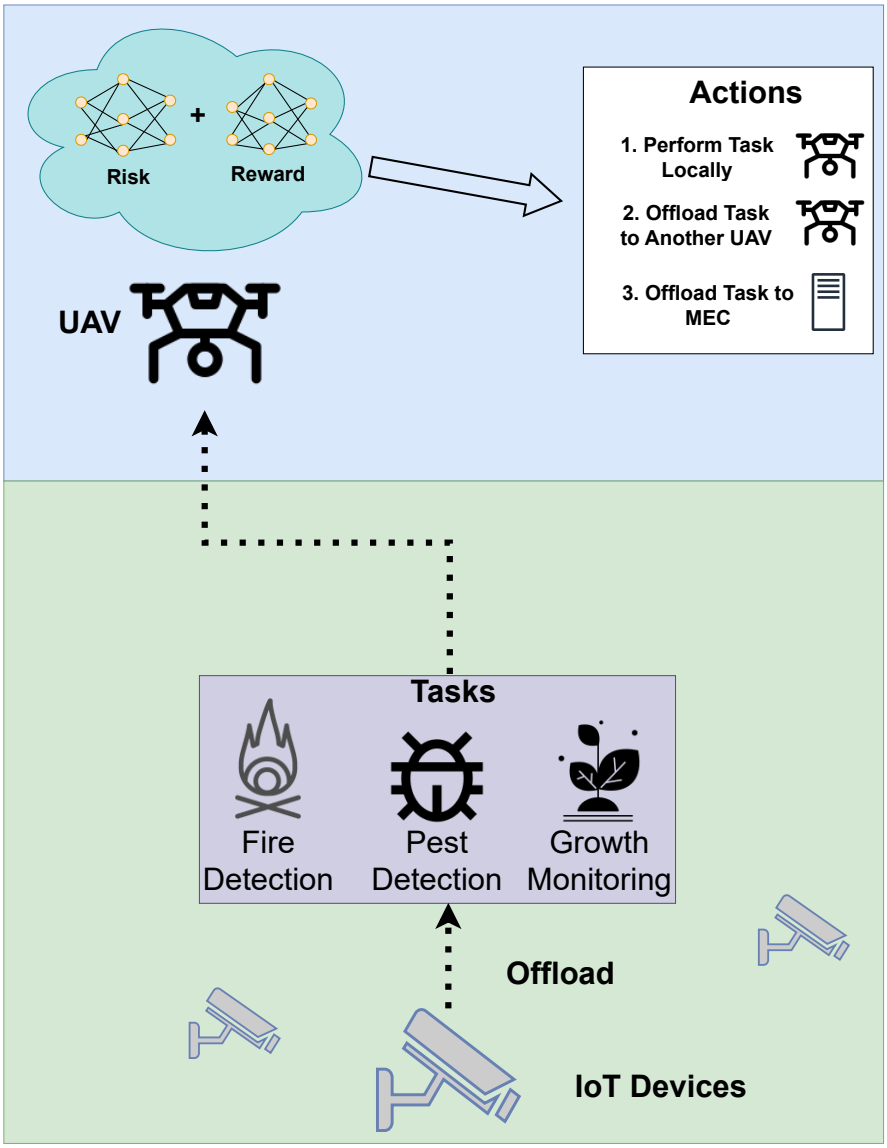


Figure 5.2: Deep Risk Sensitive Q-Learning system

5.6.1 State

Because a DNN is used to predict the Q-values instead of Q-tables, the state set from [98] can be expanded to include the transmission delays between all

of the computing resources in the network. Therefore the state for the DRS method can be defined as

$$\mathbb{S}_{DRS} = \{k, \Delta_{j' \in \mathcal{J}^+}, \Upsilon^L_{j' \in \mathcal{J}}, \Delta^{j_P \in \mathcal{J}^+}_{j'_R \in \mathcal{J}^+}, j_i\}, \tag{5.10}$$

where k is the task type, $\Delta_{j' \in \mathcal{J}^+}$ is the CPU delays for each UAV and MEC device, $\Upsilon^L_{j' \in \mathcal{J}}$ are the current battery levels of all UAVs, $\Delta^{j_P \in \mathcal{J}^+}_{j'_R \in \mathcal{J}^+}$ are the transmission delays between all UAVs and MEC devices, and j_i is the UAV that first received the task from the IoT device.

5.6.2 Risk State

An action can put the agent in a risk state, the action leads to a task exceeding its predetermined deadline. A risk state can be defined as a state in $\mathbb{S}_{DRS}$ where a deadline violation will occur. Therefore the set of risk states Ω can be defined as

$$\Omega = \{\omega | \omega \in \mathbb{S}_{DRS} \text{ and } \mathbb{E}(v_{j_a,t})) = 1\}. \tag{5.11}$$

5.6.3 Reward

The reward function Eq. (5.12) corresponds to how well the agent is adhering to the objectives defined in Eq. (5.8) [98]. The $(\Upsilon^L_{j_a} - 1)$ portion is proportional to how the action impacted the battery level of the computing resource that computed the task. Δ_{j_a} is the mean uplink delay of the computing resource a after the task was added to its queue.

$$R_{DRS} = -\left[W(\Upsilon^L_{j_a} - 1) - \frac{1 - W}{\Theta^M} * \Delta_{j_a}\right] \tag{5.12}$$

5.6.4 Risk

The risk function can be defined as

$$\mathbb{C} = \begin{cases} -\mathcal{V}_{j_a}^{L}, & \text{if } \mathbb{S}_1 \in \Omega \\ -1, & \text{otherwise} \end{cases} \tag{5.13}$$

It addresses the risk constraint defined in the objective function Eq. (5.8). If the agent reaches a risk state, then the agent will be penalized. The magnitude of that penalization is determined by $\mathcal{V}_{j_a}^{L}$. If a deadline could have been avoided, then the magnitude of $\mathcal{V}_{j_a}^{L}$ is high, otherwise it is low.

5.6.5 Policy

The agent uses an epsilon greedy approach to select an action from Eq. (5.3). Each agent has a Deep neural network (DNN) to predict Q-values for the risk (Q_{risk}) and reward (Q_{reward}) for each action. The two predicted Q-values are inputs in a function to determine the final Q-value, $\mathcal{Q}_a$, for each action. The agent then selects the action with the lowest $\mathcal{Q}_a$ value. The function to determine $\mathcal{Q}_a$ is defined as follows

$$\mathcal{Q}_a = \zeta * Q_{risk} + (1 - \zeta) * Q_{reward}. \tag{5.14}$$

ζ determines the focus of the agent, a high ζ value means that more focus is put on avoiding the risk state, whereas a low ζ means that the focus is more on optimizing the hover time and mean uplink delay. The value of ζ is updated dynamically after an episode in accordance with Algorithm 1 described in [98].

Each independent agent has two DNNs to predict the Q-values for the risk and reward values for every action. The DNN consists of two hidden layers with 32 neurons with ReLu activation each. The DNNs were trained using a batch size of 500 and an experience replay size of 100k entries. The DNN

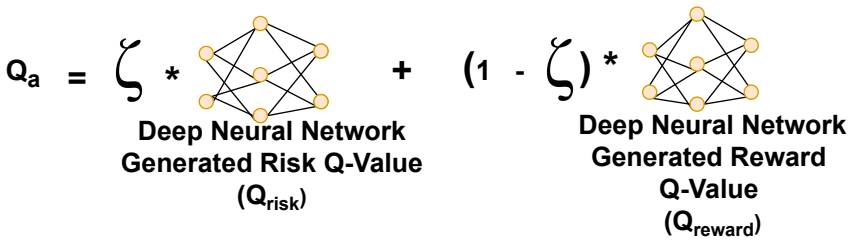


Figure 5.3: Deep Risk-sensitive Q-value generation process

uses an Adam optimizer with a learning rate of 0.001 and a Mean Squared Error loss function.

5.7 Deep Risk Quantification Q-Learning

5.7.1 Objective

There are three objectives in the RQ approach. First, maximize the hovering time of the UAV network by extending the UAV with the lowest battery level ($\min_{j' \in J} \Upsilon^R_{j'}$). Second, minimize the mean uplink delay (δ). Unlike the DRS approach, there is no hard constraint on the total number of deadline violations, instead, the third objective is to minimize the number of severe deadline violations $E(v_{\langle j,t \rangle}) = 1$.

Maximize:

$$W * \min_{j' \in J} \Upsilon^R_{j'} - \frac{1-W}{2 * \Theta^M}\delta - \frac{1-W}{2 * \Theta^D} \sum_{\langle j,t \rangle \in \langle \mathcal{J}, \mathcal{T} \rangle} \mathbb{E}(v_{\langle j,t \rangle}) \tag{5.15}$$

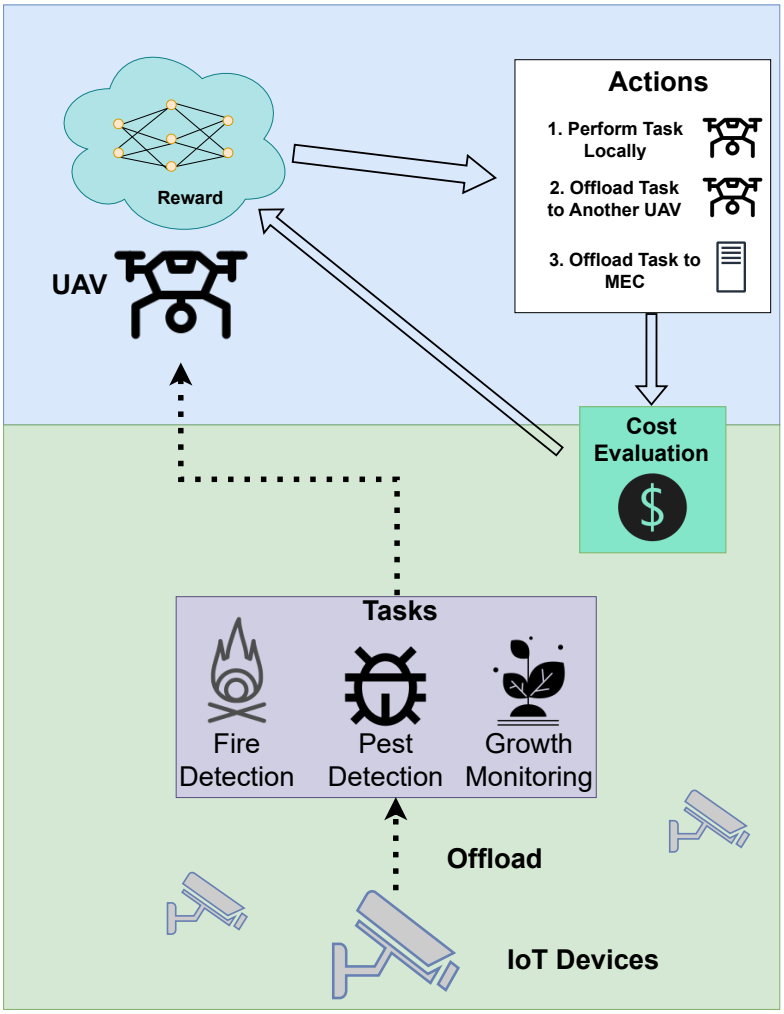


Figure 5.4: Deep Risk Quantification Q-Learning system

The state used in this methodology is defined in Eq. (5.10). The agent uses the epsilon greedy approach to select an action from the set of actions defined in Eq. (5.3).

5.7.2 Risk Cost Functions

This approach has three metrics to evaluate the potential cost of each action in terms of risk, the cost of end-to-end delay, the cost of energy, and the total cost.

The end-to-end delay risk cost function takes the task's current end-to-end delay ($\Delta_{\langle j,t\rangle}$) as an input and determines the risk cost of the task based on the distance between the $\Delta_{\langle j,t\rangle}$, and the task's deadline.

$$\mathcal{C}_d(\Delta_{\langle j,t\rangle}) = \begin{cases} -\mathbb{E}(\Delta_{\langle j,t\rangle}) \text{ if } \Delta_{\langle j,t\rangle} < \gamma^D_{\langle j,t\rangle} \\ 0 \text{ if } \Delta_{\langle j,t\rangle} = \gamma^D_{\langle j,t\rangle} \\ \mathbb{E}(\Delta_{\langle j,t\rangle}) \text{ if } \Delta_{\langle j,t\rangle} > \gamma^D_{\langle j,t\rangle} \end{cases} \tag{5.16}$$

$$\textbf{Where: } \mathbb{E}(\Delta_{\langle j,t\rangle}) = h_{\langle j,t\rangle}|(\gamma^D_{\langle j,t\rangle} - \Delta_{\langle j,t\rangle})|^s + (1 - h_{\langle j,t\rangle})|(\gamma^D_{\langle j,t\rangle} - \Delta_{\langle j,t\rangle})|^w \tag{5.17}$$

The energy risk cost ($\mathcal{C}_e$) is the cost corresponding to the UAV's current battery level. The battery level will inevitably decrease as the UAVs are in operation in the smart farm network, however, certain actions will lead to a higher increase in energy consumption than others. In addition, as the UAV's battery level gets closer to complete depletion, the situation becomes more risky because the UAV is getting closer to becoming inoperable. The significant increase in risk is shown in the exponential growth (g) in the energy risk cost equation,

$$\mathcal{C}_e(\Upsilon^R_{j_a}) = (1 - \Upsilon^R_{j_a})^g, \tag{5.18}$$

where the input is the UAV's current battery level, and the output is determined by the distance between the current battery level and the full battery capacity.

The total cost determines the current overall risk level of the UAV after

the action. It consists of the energy risk cost ($\mathcal{C}_e$) and the end-to-end delay cost ($\mathcal{C}_d$). Γ is the energy risk cost's scaling factor.

$$\mathcal{C}_T = \Gamma * \mathcal{C}_e + \mathcal{C}_d \tag{5.19}$$

5.7.3 Risk Measurement

The total cost of an action, ($\mathcal{C}_{T_a}$), is calculated after the action is performed. ($\mathcal{C}_{T_a}$) is added to an ordered list of the total costs ($\mathbb{Z}_{j_a}$) for all of the previous $n-1$ actions that agent j_a performed. Including ($\mathcal{C}_{T_a}$), there is a total of n elements in ($\mathbb{Z}_{j_a}$) at time t. The lowest $\frac{\alpha}{2}\%$ of n elements in $\mathbb{Z}_{j_a}$ is stored in a list called $\mathbb{Z}_{L_{j_a}}$. The highest $\frac{\alpha}{2}\%$ of n elements in $\mathbb{Z}_{j_a}$ is stored in a list called $\mathbb{Z}_{H_{j_a}}$. The risk caused by action a is measured by taking the mean of the highest $\frac{\alpha}{2}\%$ costs and the lowest $\frac{\alpha}{2}\%$ costs. The equation for the risk measurement after action a is defined as

$$\eta = \frac{\sum \mathbb{Z}_{L_{j_a}} + \sum \mathbb{Z}_{L_{j_a}}}{\frac{\alpha}{100} * n}. \tag{5.20}$$

5.7.4 Reward

The reward function considers the effect of the action on the current UAV and the UAV with the highest costs. Therefore, in the reward functions, there are two reward values: the value for the current UAV (μ_c) and the value for the UAV with the highest costs (μ_m). The reward function R_{RQ} is defined as

$$R_{RQ} = (1-\beta)\mu_c + \beta\mu_m, \tag{5.21}$$

where β is the magnitude consideration given for the current UAV's costs.

The reward values μ_c and μ_m can both be defined by Eq. (5.22). The magnitude of the reward value depends on the UAV's CVaR value before (η_p)

and after (η_a) the action was executed. The agent should be encouraged to select the action that decreases the UAV's average risk cost (i.e. $\eta_a < \eta_p$). Therefore the agent will be rewarded with positive values for reward. If the UAV's average risk cost stayed the same, (i.e. $\eta_a < \eta_p$), this is acceptable behavior but not ideal, which is why the agent's reward is still positive but the magnitude is not high. If the UAV's average risk costs are increased, then the agent chose an action that significantly increased battery consumption, uplink mean delay, or exceeded the deadline violation of a dangerous task. The agent is discouraged from this behaviour by being awarded negative rewards. The formula to determine the value of the reward value for a UAV (μ) is defined as

$$\mu = \begin{cases} 20, \text{ if } (\eta_a < \eta_p) \wedge (\eta_a < 0) \\ 10, \text{ if } (\eta_a < \eta_p) \wedge (\eta_a = 0) \\ 1, \text{ if } (\eta_a < \eta_p) \wedge (\eta_a > 0) \\ 5, \text{ if } (\eta_a = \eta_p) \wedge (\eta_a < 0) \\ 2, \text{ if } (\eta_a = \eta_p) \wedge (\eta_a = 0) \\ -5, \text{ if } (\eta_a = \eta_p) \wedge (\eta_a > 0) \\ 1, \text{ if } (\eta_a > \eta_p) \wedge (\eta_a < 0) \\ -1, \text{ if } (\eta_a > \eta_p) \wedge (\eta_a = 0) \\ -10, \text{ if } (\eta_a > \eta_p) \wedge (\eta_a > 0). \end{cases} \tag{5.22}$$

5.7.5 Risk Quantification Parameters

In terms of the cost function parameters for the RQ method, the energy risk growth is equal to 2 ($g = 2$), the fire task end-to-end delay risk growth value of 8 ($s = 8$), and the other task risk growth value of 1 ($w = 1$). For the risk measurement, an α value of 2 was used. In RQ's reward function, $\beta = 0.75$.

Chapter 6

Simulation

All of the proposed task-offloading decision-making algorithms used the same simulation tool. To compare the performance of the different algorithms, key performance indicators (KPI) such as remaining energy levels, percentage of deadline violations, and distribution of percentage of deadline violations were used. These KPIs evaluated the efficacy of the proposed algorithms in achieving the objectives.

6.1 Simulation Tool: Simu5G

The UAV-MEC smart farm was simulated using the Simu5G [99] library on the Omnet++ platform [100]. Simu5G provided the modules to simulate a wireless network that complies with 3GPP standards for 5G and Long-term evolution (LTE).

6.2 Key Performance Indicator (KPI)

The following key performance indicators were used to evaluate the performance of the proposed decision-making algorithms.

6.2.1 Remaining Battery Levels

As shown in Eq. (6.1), a UAV's remaining battery level (Υ_U), is the amount of power the UAV has left at the end of the simulation (Υ_j^R), divided by the total power capacity (Υ_j^B).

$$\Upsilon_U = \frac{\Upsilon_j^R}{\Upsilon_j^B} \tag{6.1}$$

The minimum remaining battery levels were used to compare the algorithms' performance in terms of energy. The minimum remaining battery level indicates how long the entire UAV network can last hovering above the farmland.

6.2.2 Percentage of Deadline Violations

According to Eq. (4.11), a deadline violation occurs when the uplink delay exceeds the task's predetermined deadline.

Deadline Violation Percentage Per Node

Out of all deadline violations that occurred in the network, the percentage of those deadline violations that occurred at computing resource t. In Figures 6.7, 6.9, and 6.12, they compare the deadline violation percentage.

Deadline violation Per Task Type

The task type of each deadline violation was monitored. In Figures 6.10 and 6.13, they compare the proportion of the deadline violation per task type.

6.3 Farm Environment

The farm environment had 16 IoT devices, 4 UAVs, and 1 MEC device. In the simulation, the IoT generated fire detection, pest detection, and growth monitoring tasks. An exponential distribution was used to model the tasks' interarrival time ($1/\lambda$).

There is an assumption that the MEC device has more computing resources and therefore the MEC's processing times ($\gamma^{P}_{\langle j,t\rangle}$ (MEC)) for all tasks are two times faster than the UAVs'($\gamma^{P}_{\langle j,t\rangle}$ (UAV)) processing times.

6.4 Energy Consumption Parameters

The parameters in Table 6.1 and Eq. (4.9) were used to model the remaining UAV energy in our simulations. There is an assumption that this simulation used a battery that is similar to the one found in [101]. To calculate the power consumption of hovering, Equation 2 from [102] was used, as well as their assumptions such that that the UAV will have: 4 rotors, fluid density of $1.204kg/m^3$, rotor disc area of $0.2m^2$, frame's mass $M = 1.5$kg, and battery and payload $b = 3$ kg. The equation from [102] is shown in Eq. (6.2), where W is the mass of UAV in kg, b is the mass of the battery and other payloads in kg, g is gravity in Newtons, ρ is the fluid density in kg/b^3, ς is the rotor disc area in m^2, and n is the number of rotors. To derive the hovering power consumption, the implementation of Equation 2 is as follows,

$$P = (M+b)^{\frac{3}{2}}\sqrt{\frac{g^3}{2\rho\varsigma n}} \tag{6.2}$$

$$P = (1.5+3)^{\frac{3}{2}}\sqrt{\frac{9.8^3}{2(1.204)(0.2)(4)}}$$
$$\approx 211Watt - hour.$$

Table 6.1: UAV Energy Consumption Parameters

Power Consumption Parameters	Value (Watt-hour)
Fully Charged Battery	570, 627
Hovering	211
Antenna	17
Idle CPU [1]	4320
Busy CPU	12960

In all simulations, the following energy consumption values were used with Eq. (4.9). A UAV hovering (Υ_j^H) consumed 211 watt-hours. The antenna (Υ_j^A) needed 17 watt-hours. The CPU needed 4320 watt-hours to operate if it is idle (Υ_j^I), and 12960 watt-hours if the CPU is computing a task (Υ_j^C). The power consumption parameters are summarized in Table 6.1.

6.5 Results

6.5.1 Performance Evaluation of Proposed Reinforcement Learning Techniques

All proposed reinforcement learning techniques used the following RL parameters, a learning rate of 0.05, and a discount factor of 0.85. All models were trained using a dataset of 100 recorded simulations.

[1] Idle and running CPU energy consumptions are selected based on limited simulation time. Thus, UAVs that will operate for ten hours can be simulated, and the difference in energy performance between the methods can be tested.

Q-Learning

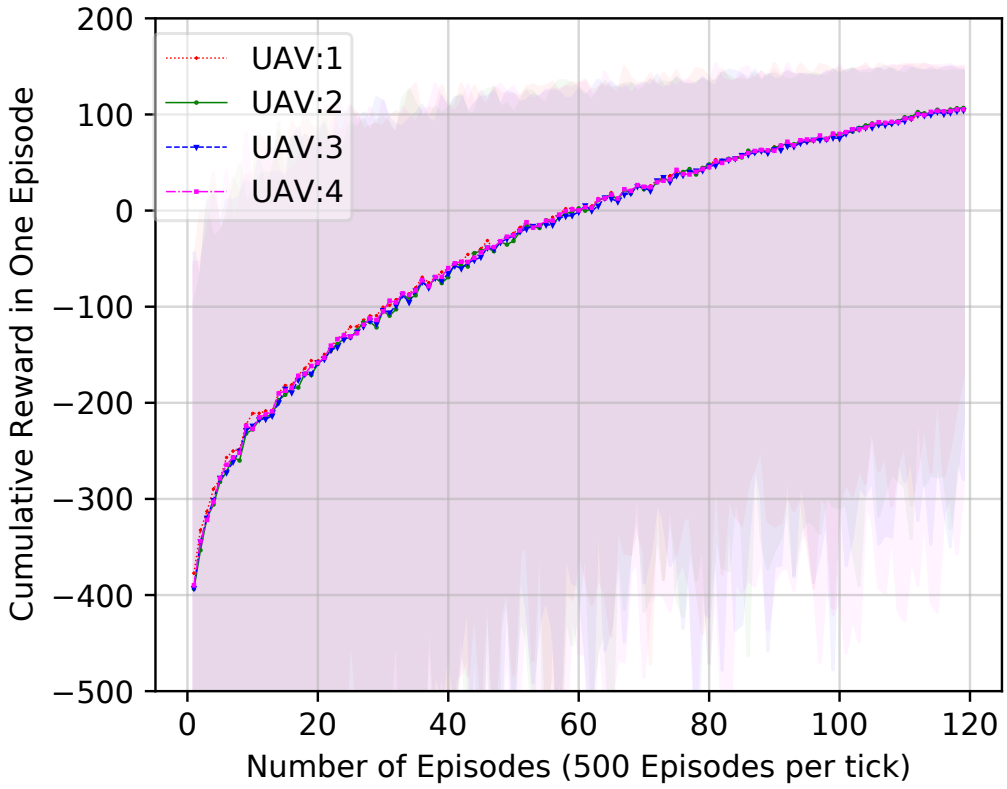


Figure 6.1: Q-Learning reward convergence

Figure 6.1 shows Q-Learning's cumulative reward per episode after a total of 60K episodes. The solid lines show the agent's average cumulative reward for every 500 episodes. The shaded area shows the maximum and minimum cumulative reward for every 500 episodes. Figure 6.1 demonstrates that the cumulative reward converges after approximately 55k episodes. The agents all converged to an average cumulative reward of 100.

Deep Q-Learning

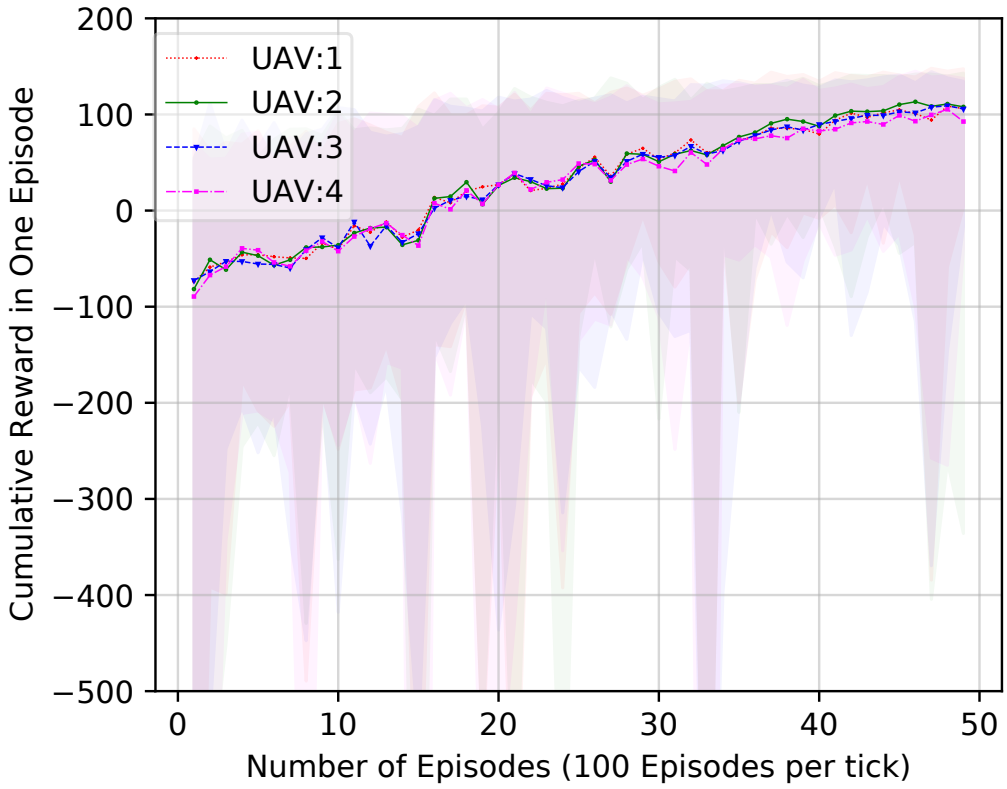
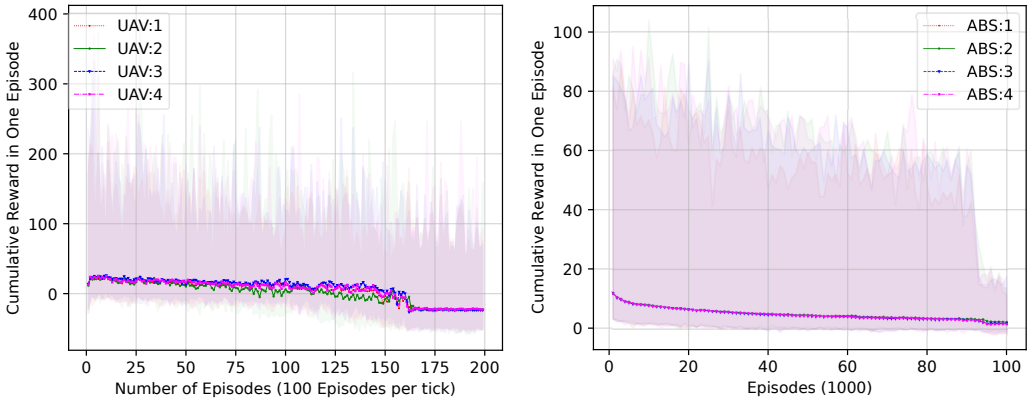


Figure 6.2: Deep Q-Learning reward convergence

Figure 6.2 shows the average cumulative reward of the DQL method after a total of 5k episodes. The figure demonstrates that the average rewards began converging after 4200 episodes. Adding a DNN improved the speed and variation of achieving convergence. Firstly, the DQL method was able to reach the optimal reward 13 times faster than the Q-Learning. In both Q-Learning and DQL methods, the variation of the reward gets smaller as the number of episodes increases. This indicates that the agents get better at learning the optimal solution. The variation in the DQL method's cumulative reward decreases quicker than in the tabular Q-Learning method. This is

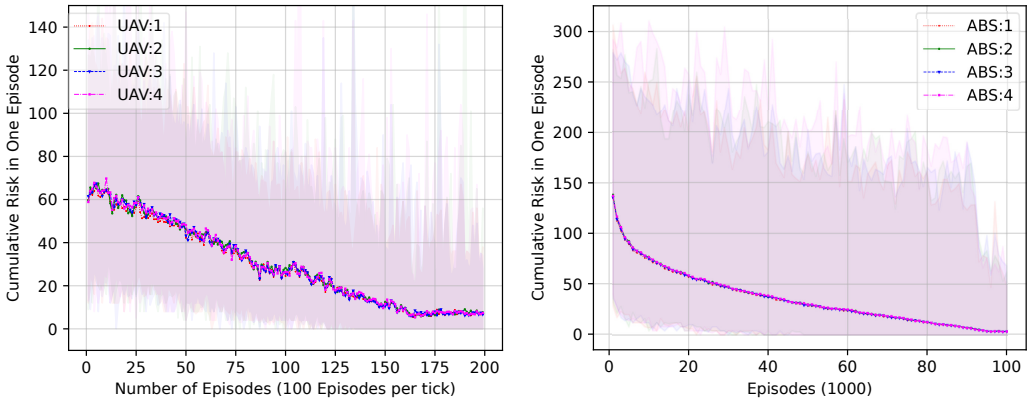
because as the number of episodes increases, the DNN in the DQL method gets better at predicting the Q-value associated with each action.

Deep Risk-Sensitive Q-Learning



(a) Deep Risk-Sensitive (b) Risk-Sensitive [98]

Figure 6.3: Reward convergences of Deep Risk-Sensitive and Risk-Sensitive methods



(a) Deep Risk-Sensitive (b) Risk-Sensitive [98]

Figure 6.4: Risk convergences of Deep Risk-Sensitive and Risk-Sensitive methods

Figures 6.3 and 6.4 show that deep learning enabled the DRS agents to find the optimal solution 5 times faster than their tabular risk-sensitive counterparts. The risk and reward converged after around 16.25k episodes, while the tabular risk-sensitive method converged after around 95k episodes. The variances in the cumulative reward and risk values were also reduced in DRS because DNNs were more accurate at predicting the risk and reward Q-values. In addition, the DRS agents were able to find an optimal solution with a lower reward compared to the risk-sensitive agents.

Deep Risk Quantification Q-Learning

In Figure 6.5, the cumulative reward of an episode for the Risk Quantification (RQ) method after a total of 10k episodes. The RQ method's reward was able to achieve convergence after 8750 episodes.

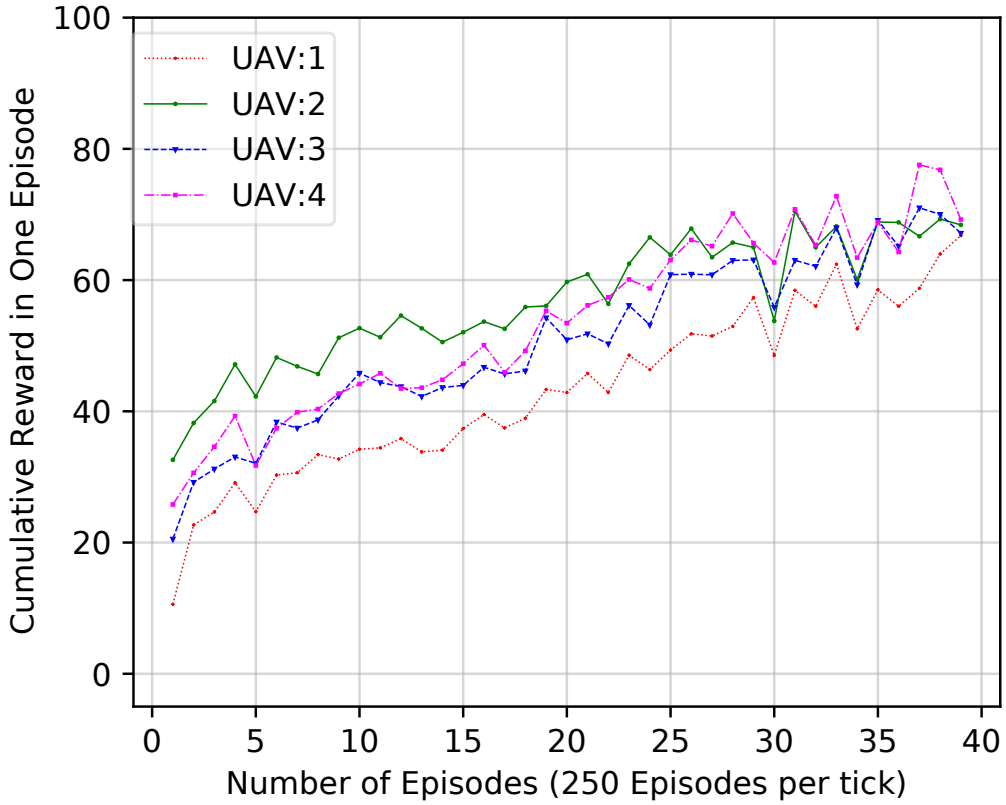


Figure 6.5: Risk Quantification Reward convergence

6.5.2 Comparing Rule-Based Heuristic Techniques with Q-Learning Techniques

Simulation Parameters

The simulation time was 50s (T = 50). The battery capacity for all UAVs was 570 watt-hour. The task simulation parameters from Table 6.2 were used.

Table 6.2: Task Parameters for Q-Learning Simulation

Task Type	$(1/\lambda)$	$\gamma^{D}_{\langle j,t \rangle}$	$\gamma^{P}_{\langle j,t \rangle}$ (UAV)	$\gamma^{P}_{\langle j,t \rangle}$ (MEC)
Fire detection	0.25s	0.3s	0.1s	0.05s
Pest detection	0.25s	0.8s	0.5s	0.25s
Growth monitoring	0.5s	5s	0.1s	0.05s

All simulation results are generated from the mean results from 10 runs with different seeds.

Energy Performance

Out of all the algorithms displayed in Figure 6.6, RR had the lowest remaining battery level in every UAV. This is because unlike all of the other algorithms, RR has no consideration for energy in its decision-making algorithm. Because of the even task distribution, RR did not take advantage of the MEC's unlimited power supply and more powerful computing resources. This is demonstrated in the low remaining energy levels in all the UAVs in Figure 6.6, and the low percentage of deadline violations in the MEC in Figure 6.7. Out of all the non-machine learning algorithms, QHEF had the best performance. It had the highest minimum remaining battery level, as well as the highest remaining battery level in all nodes. Compared to the machine learning algorithms, it is lower by a difference of 2 to 4. The difference between QHEF's and DQL's highest minimum remaining battery level is approximately 2%.

In Figure 6.6, tabular Q-Learning had the highest remaining energy level across all of the UAVs. It significantly performed better than the non-machine learning algorithms. There was a difference of approximately 4 between Q-Learning's remaining energy level and QHEF's remaining energy level in each UAV. DQL performed better than the non-machine learning algorithms, but was worse than tabular Q-Learning by a difference of 1-2, as evidenced in Figure 6.6.

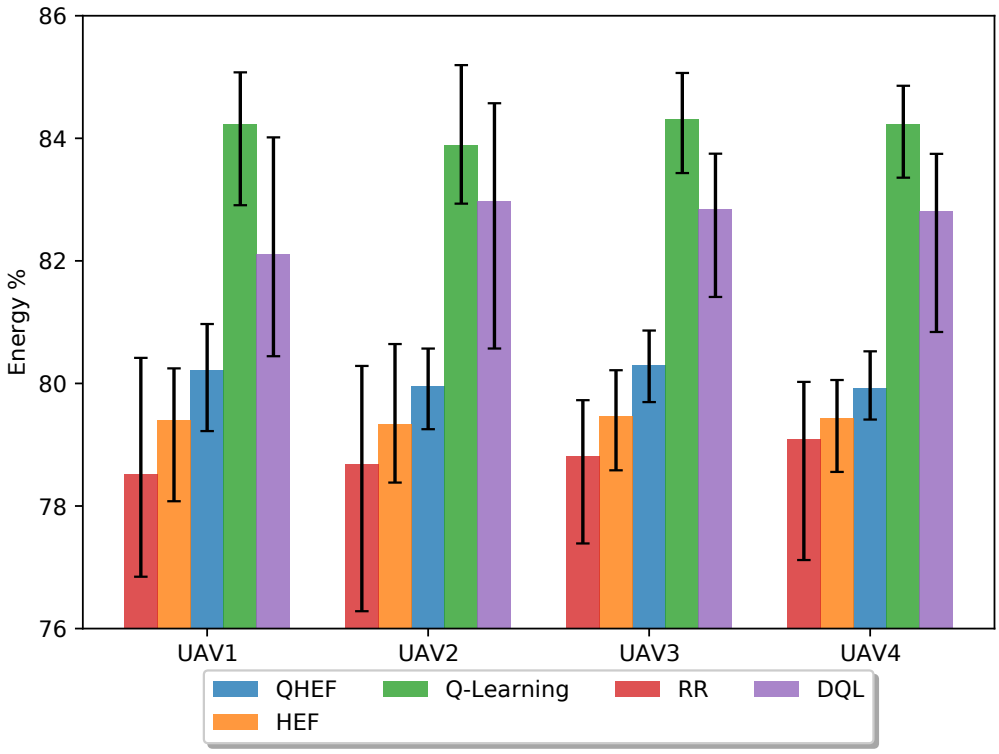


Figure 6.6: Remaining energy comparison between heuristic rule-based techniques and Q-Learning techniques

Deadline Violation Performance

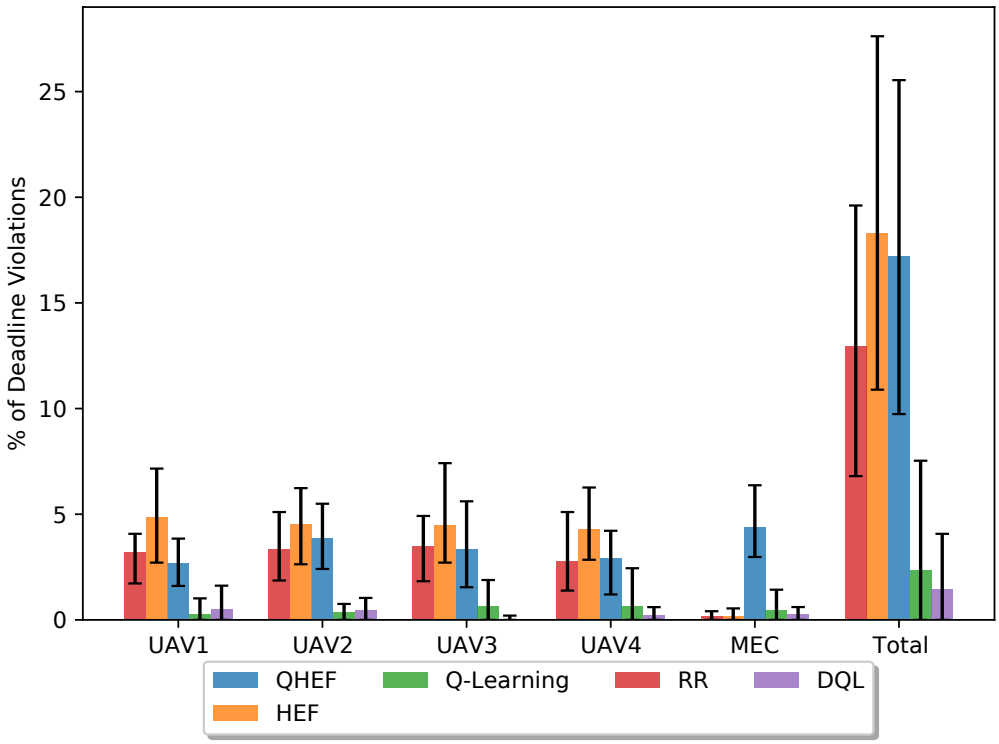


Figure 6.7: Deadline violation percentage comparison between heuristic rule-based techniques and Q-Learning techniques

Figure 6.7 shows that out of all the heuristic rule-based algorithms, RR had the lowest total percentage of deadline violations. In this decision-making algorithm, the tasks are evenly distributed among all computing resources. This is why the percentage of deadline violations is similar across all UAVs. HEF had the highest total percentage of deadline violations. Unlike the other algorithms, HEF only considered the computing resources' remaining energy levels. Most of HEF's deadline violations occurred at the UAVs and not the MEC. In terms of the percentage of deadline violations, QHEF performed better than HEF but worse than RR. QHEF considered both queuing time

and energy, however, it sent all of the tasks to the MEC device which increased the MEC device's queue time and therefore increased the number of deadline violations.

Q-Learning significantly outperformed the non-machine learning algorithms in terms of deadline violations. Figure 6.7 illustrates that there is a difference of 10 between the RR and Q-Learning's percentage of deadline violations. Deadline violations are where DQL began significantly outperforming the non-machine learning algorithms. DQL had approximately 12 times fewer deadline violations than the leading heuristic algorithm RR. QHEF and HEF had approximately 15-16 times more deadline violations than DQL. It also performed better than Q-Learning by a difference of 1.

Figures 6.6 and 6.7 show that the non-machine learning algorithms' performances fall behind the machine learning algorithms in terms of energy and deadline violations. In terms of deadline violations, the machine learning algorithms had significantly lower deadline violations. This can be attributed to the fixed nature of the heuristic algorithms. The heuristic algorithms make decisions based on fixed rules that do not adapt to the changing situation.

6.5.3 Comparing Risk-Sensitive Techniques With Traditional Q-Learning Techniques

Simulation Parameters

The simulation time was 50s (T = 50). The battery capacity for UAVs 1 and 2 UAVs was 570 watt-hour each. The battery capacity for UAVs 3 and 4 was 670 watt-hour each. The task simulation parameters presented in Table 6.3 were used.

Table 6.3: Task Parameters for Risk-Sensitive Simulation

Task Type	$(1/\lambda)$	$\gamma^{D}_{\langle j,t \rangle}$	$\gamma^{P}_{\langle j,t \rangle}$ (UAV)	$\gamma^{P}_{\langle j,t \rangle}$ (MEC)
Fire detection	0.25s	1s	0.1s	0.05s
Pest detection	0.25s	2s	0.2s	0.1s
Growth monitoring	0.5s	155s	1.5s	0.75s

All simulation results are generated from the mean results from 10 runs with different seeds.

Energy Performance

In comparison to other machine learning algorithms, Q-Learning outperformed the other algorithms in terms of remaining energy. In Figure 6.8, Q-Learning's remaining energy level was 1-2% higher than DQL across all UAVs. Unlike the Risk Sensitive method, there isn't a big range in remaining energy levels amongst all the UAVs. The UAVs' remaining energy levels are within less than 1% of one another. DRS' performance was comparable to DQL's. All of their remaining battery levels were within less than 1% of one another in Figures 6.8 and 6.12. DRS also did not experience a big range in energy levels among the UAVs like the Risk-Sensitive method. In addition, its lowest remaining energy level was higher than Risk-Sensitive's lowest remaining level. DRS' lowest remaining energy level was 81%, and Risk-Sensitive was 78%, which shows a difference of 3. This could be attributed to the usage of deep neural networks.

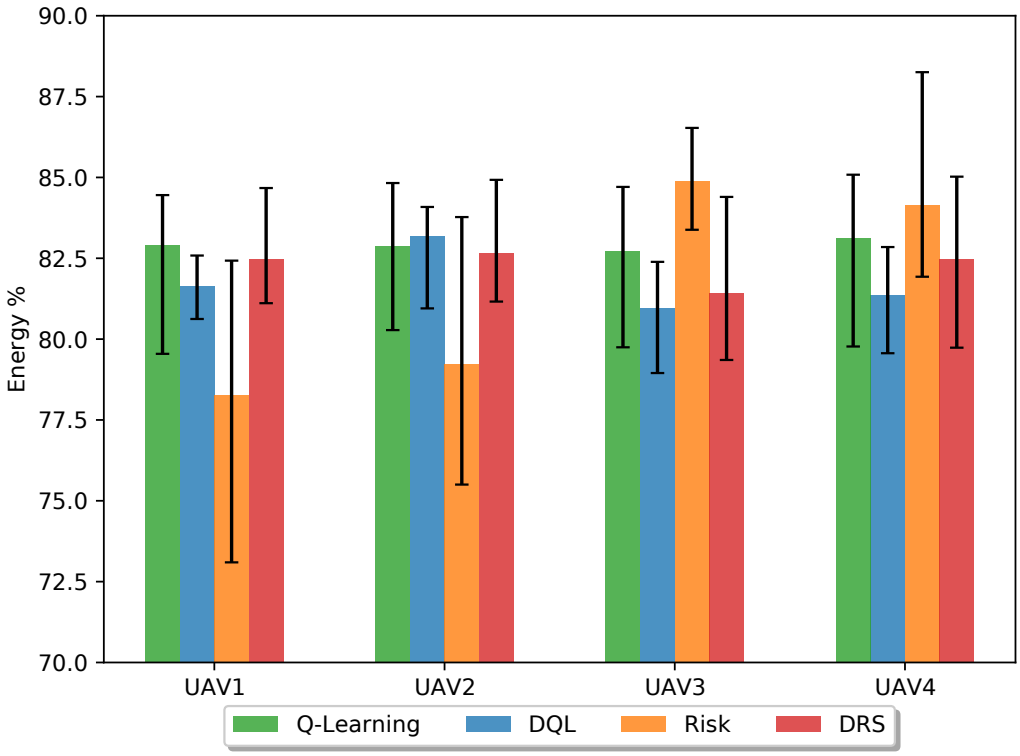


Figure 6.8: Remaining energies of machine learning algorithms

Deadline Violation Performance

Q-Learning was outperformed by the other machine learning algorithms in terms of the percentage of deadline violations. This was shown in Figure 6.9 where Q-Learning had the highest percentage of deadline violations amongst all machine learning algorithms. One explanation is that Q-Learning has limited state space and does not consider the transmission times between all of the nodes in its decision-making algorithms. Figures 6.6 and 6.8 indicate that Q-Learning prioritized preserving the UAVs' energy levels rather than the deadline violations. This is also confirmed in Figure 6.9, where most of Q-Learning's deadline violations occur in UAV3 and UAV4. In this simulation, UAV3 and UAV4 have higher battery capacities than the other UAVs and

Q-Learning chose to send more tasks to those UAVs to preserve the energy levels of the other UAVs. DQL performed better than Q-Learning, but it had approximately 1% more deadline violations than the methods than the risk-sensitive methods.

The DRS method had the second lowest percentage of deadline violations. DRS' total percentage of deadline violations was 0.93%. It performed better than DQL by 1. DRS' better performance is attributed to its risk state. The Risk-Sensitive and DRS methods both had a state space dedicated to the anticipation of a deadline violation. This proved to be a better method of avoiding deadline violations than the general state seen in Q-Learning and DQL. The Risk-Sensitive and DRS methods' total percentages of deadline violations were 0.32% and 0.93% respectively.

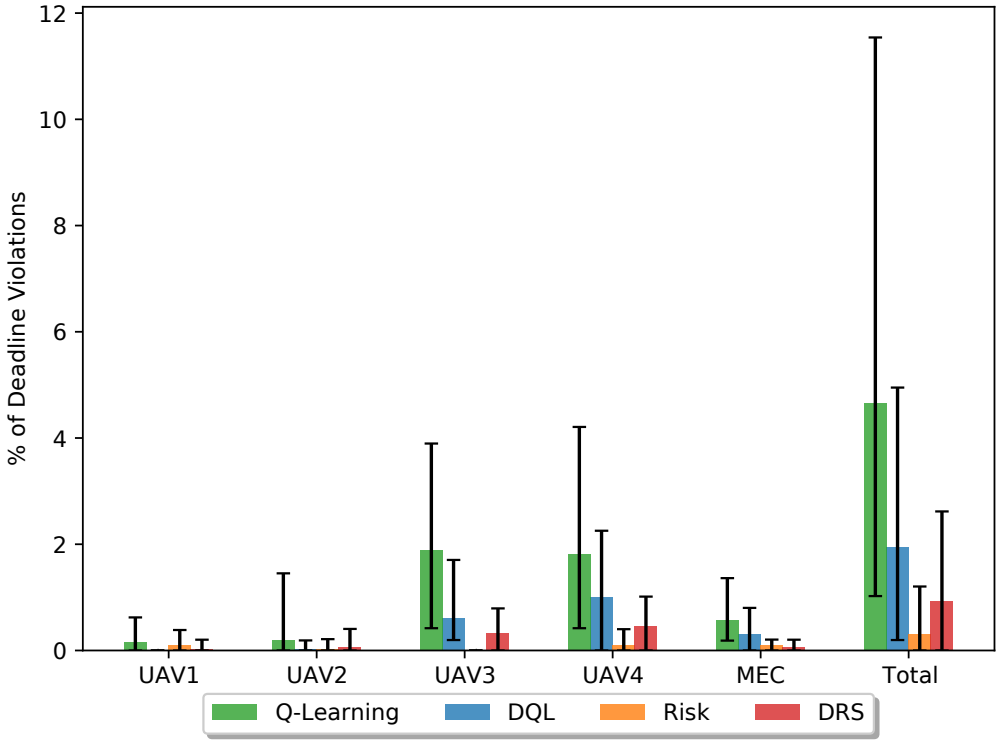


Figure 6.9: Deadline violation percentage among machine learning algorithms

Figure 6.10 highlights that across all techniques, most of the deadline violations are fire tasks. The deadline violations are almost evenly split between fire detection tasks and pest detection tasks in the Q-Learning technique. The majority of DQL, risk-sensitive, and DRS' deadline violations are fire detection tasks.

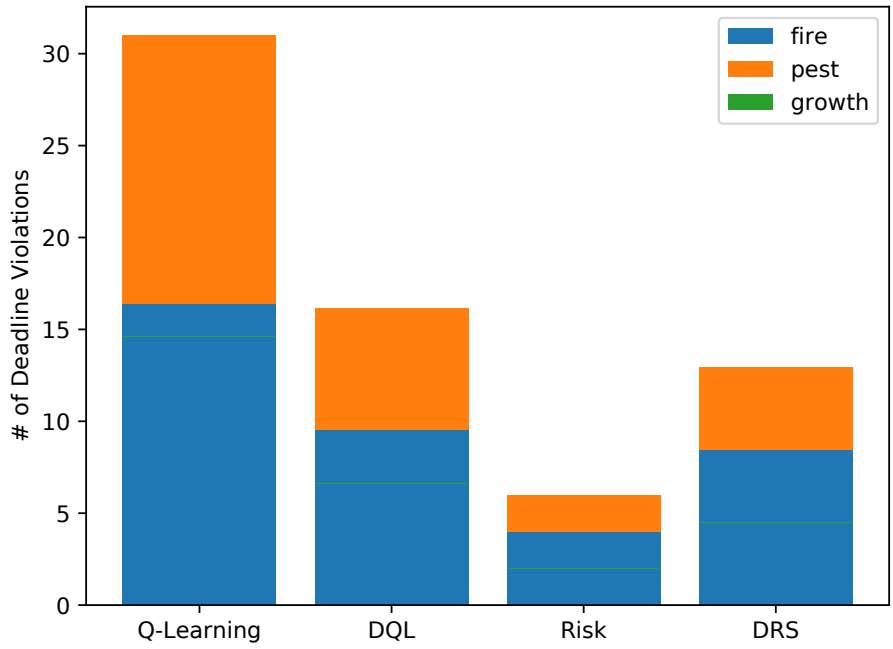


Figure 6.10: Deadline violation distribution by task for machine learning algorithms

6.5.4 Comparing Deep Risk Quantification With Other Deep Learning Techniques

Simulation Parameters

The simulation time was 5s (T = 5). The battery capacity for UAVs 1 and 2 UAVs was 570 watt-hour each. The battery capacity for UAVs 3 and 4 was 670 watt-hour each. The task simulation parameters from Table 6.3 were used. All simulation results are generated from the mean results from 10 runs with different seeds.

Energy Performance

According to Figure 6.11, DRS performed slightly better than RQ. DRS' lowest remaining energy level was 96.9%, meanwhile, RQ's minimum remaining energy level was 96.6%. RQ had the lowest minimum remaining energy levels with the lowest remaining energy level at 96.6%. Its low remaining energy level can be attributed to its focus on reducing deadline violations for risky tasks like fire detection.

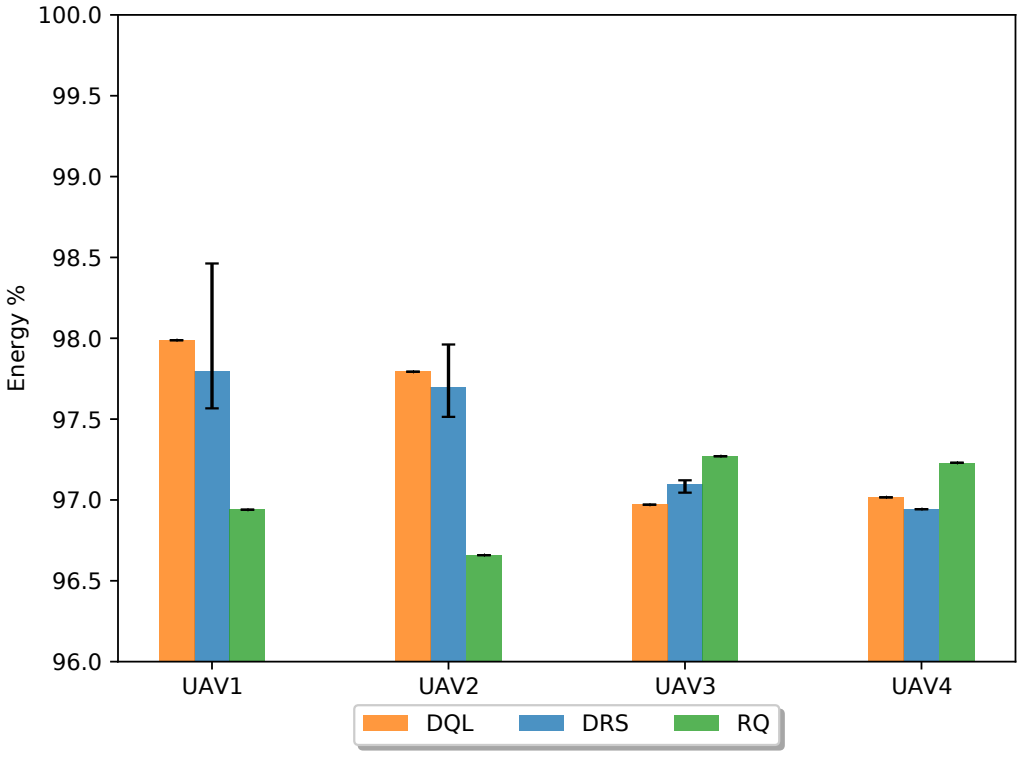


Figure 6.11: Remaining energy levels of deep learning algorithms

Deadline Violation Performance

When DQL is compared to the other deep learning methods in 6.12, it has a significantly higher percentage of deadline violations. 11% of its tasks

were deadline violations compared to the 4-5% seen in the DRS and Risk Quantification (RQ) methods. Unlike DQL, DRS and RQ both highlight deadline violations as risky behaviour and train their agents to avoid the risk if possible. RQ's quantification functions enabled it to significantly reduce the percentage of deadline violations. In Figure 6.12, it had the lowest total percentage of deadline violations at 4%. Most of its deadline violations occurred at UAVs 1 and 2, unlike DQL and DRS whose deadline violations mostly occurred at the MEC device.

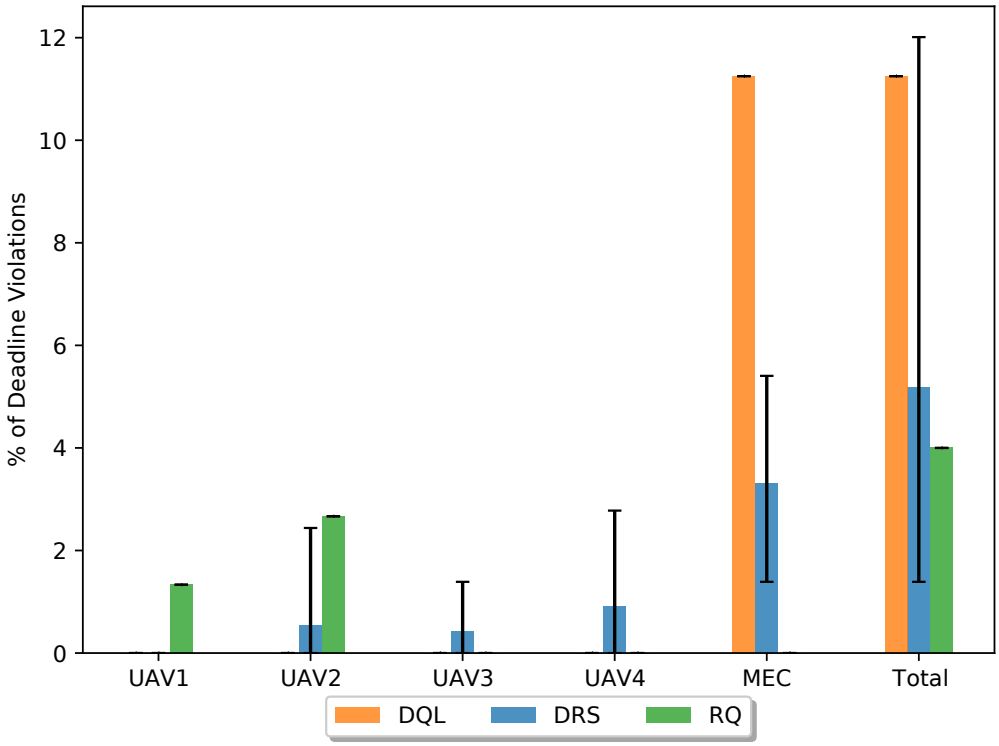


Figure 6.12: Deadline violation percentages of deep learning algorithms

Figure 6.13 demonstrates that unlike all of the methods that came before it, RQ was able to eliminate deadline violations for fire detection tasks. In Figures 6.10 and 6.13, the fire detection tasks make up at least half of the

deadline violations in all of the machine learning techniques aside from RQ. The fire detection task has the shortest deadline and is one of the most frequently received tasks. This increases the probability of a deadline violation occurring. By quantifying the risks of violating a deadline violation for each task type, RQ was able to eliminate fire task deadline violations, which in turn reduced the total percentage of deadline violations.

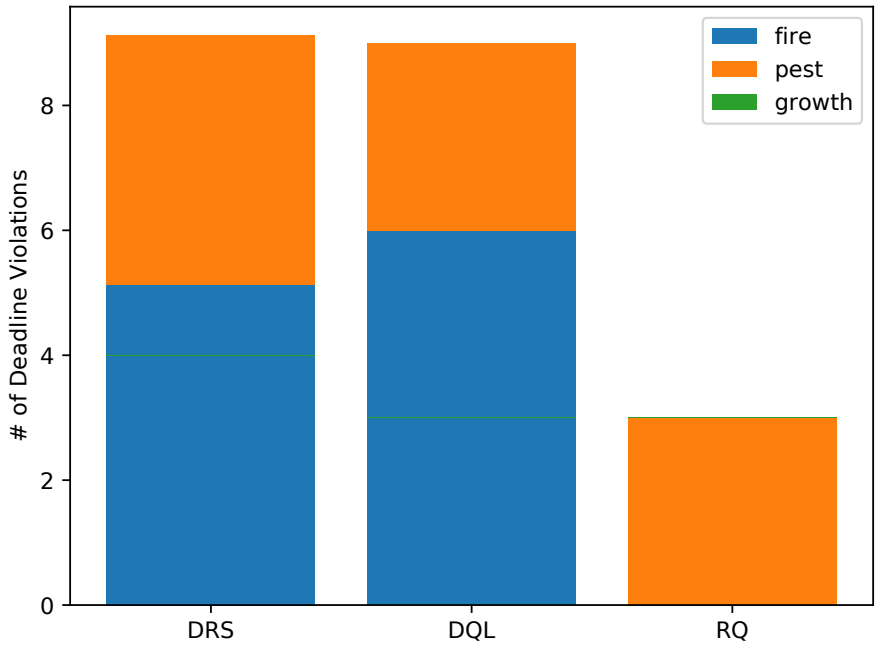


Figure 6.13: Deadline violation task distribution for deep learning algorithms

Chapter 7

Conclusion and Future Works

7.1 Conclusion

Smart farming and precision agriculture are the answers for sustainably feeding a population. They promote minimal waste of resources such as water and pesticide, by constantly monitoring the state of the farm using sensors. IoT-enabled cameras are an example of sensors that can be deployed across the farm. They capture images from various points of view and the current state of the farm can be determined through image classification. Information is relayed from the sensors to a central server and decisions are made based on the latest data. Many of the sensors and devices are battery-operated, and to extend their operational time, UAVs and MEC devices can be deployed as computational resources where the IoT devices can offload. Image classification tasks such as fire detection can be both computationally intensive and time-sensitive. Long delays can result in damages to the farm that are beyond repair. In the proposed smart farm environment three things are monitored using image classification tasks, fires, pests, and crop growth. Due to their limited energy and processing power, the IoT devices offload these tasks to nearby hovering UAVs. The UAVs are also limited in energy capacity and all the tasks are time-sensitive, therefore the UAVs must decide

whether to compute the task locally or offload the task. Each UAV has an independent decision-making algorithm to help them decide where the task will be computed. The objectives of the decision-making algorithms are to conserve the UAVs' battery while also minimizing the number of deadline violations.

Three rule-based decision-making algorithms were introduced, RR, HEF, and QHEF. These algorithms used rules based on heuristics to determine the location to send the task. Four machine-learning techniques were also introduced, Q-Learning, DQL, DRS, and RQ. These algorithms used RL to evaluate the impact of each action on the long-term gain from achieving the objectives. From the results, it can be concluded that the machine learning algorithms, Q-Learning, DQL, DRS, and RQ were better than the non-machine learning algorithms. They performed better in terms of remaining battery levels, and total number of deadline violations. The machine learning algorithms were able to adapt to the changing situation, and some were able to evaluate precisely the risk of each action. Meanwhile, the non-machine learning algorithms like RR, HEF, and QHEF had rigid rules that were based on heuristics. Some of the heuristics applied to certain KPIs, some were not.

Identifying deadline violations as a risk, reduced the percentage of deadline violations. Among the machine learning algorithms, DRS and RQ had the lowest percentage of deadline violations. Furthermore, quantifying the risk in terms of cost of damages, allows the agents to eliminate critical risks. The two most common tasks that caused deadline violations were fire detection tasks and pest monitoring tasks. Because of its short deadline and high occurrence, the fire detection task was responsible for at least half of the deadline violations in all machine learning algorithms except RQ. Because it was able to quantify the risk involved with violating a task deadline for each task type, the RQ method was able to eliminate all deadline violations for fire detection tasks. This led it to further reduce the number of total deadline violations, and it had the lowest percentage of deadline violations

among all machine learning algorithms.

7.2 Future Works

There are a few ways the work presented in this thesis can be expanded. The current network topology remains static throughout the simulation, however, UAVs are not always static and can be easily deployed. Further work can be done to investigate how the proposed techniques fare with a changing network topology. For example, an increase and decrease in the number of UAVs and how the other UAVs can adapt to the changes. In addition, how does the addition or deletion of new agents impact the ability to reach the optimal solution? Different machine learning techniques such as federated learning and graph neural networks can be explored to find the optimal solution in a dynamic network topology.

Another avenue where the current problem can be explored is security. Networks are prone to cyberattacks such as denial of service, and spoofing. Nodes in a wireless network communicate through the air which makes them particularly more susceptible to these types of attacks. The cyberattacks can affect the network's productivity and resources. A denial of service attack can make the UAV unable to process tasks. A spoofing attack will artificially increase the load of tasks which will quickly the UAVs' battery levels as well as increase the number of deadline violations because they are processing tasks that are not legitimate. Not only do cyberattacks affect the network, but they can also impact the agents' training and their ability to reach an optimal solution. A spoofing attack impacts the agents' training data because they are creating fake data which can mislead the agent into a false optimal solution. Federated learning is a possible good candidate for this problem because each agent trains their independent model, and then a global model is formed based on an aggregation of a subset of models, the global model is then distributed to all agents as the new baseline for their training. The

distribution of the global model to all agents may help prevent an agent from being skewed by spoofed data. There are also iterative methods and adversarially robust techniques to investigate to improve the deep neural network against adversarial attacks.

References

[1] Nokia, "Smart agriculture - The fight to feed 10 billion." https://www.nokia.com/thought-leadership/articles/fight-to-feed-10-billion/?did=D00000006824&gad_source=1&gclid=EAIaIQobChMI9OGE35HxggMVcCmtBh3ccw1uEAAYASAAEgIwz_D_BwE. Accessed: 2023-12-01.

[2] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming," *IEEE Access*, vol. 7, pp. 156237–156271, 2019.

[3] S. Qazi, B. A. Khawaja, and Q. U. Farooq, "IoT-Equipped and AI-Enabled Next Generation Smart Agriculture: A Critical Review, Current Challenges and Future Trends," *IEEE Access*, vol. 10, pp. 21219–21235, 2022.

[4] S. Moradi, A. Bokani, and J. Hassan, "UAV-based Smart Agriculture: a Review of UAV Sensing and Applications," in *2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 181–184, 2022.

[5] M. A. Al-Shareeda, S. Manickam, and M. A. Saare, "Intelligent Drone-based IoT Technology for Smart Agriculture System," in *2022 Interna-*

tional Conference on Data Science and Intelligent Computing (ICDSIC), pp. 41–45, 2022.

[6] A. V. Savkin and H. Huang, "Deployment of Unmanned Aerial Vehicle Base Stations for Optimal Quality of Coverage," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 321–324, 2019.

[7] G. Amponis, T. Lagkas, M. Zevgara, G. Katsikas, T. Xirofotos, I. Moscholios, and P. Sarigiannidis, "Drones in B5G/6G Networks as Flying Base Stations," *Drones*, vol. 6, no. 2, 2022.

[8] P. Juyal and S. Sharma, "Strawberry Plant's Health Detection for Organic Farming Using Unmanned Aerial Vehicle," in *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, pp. 1–6, 2021.

[9] K. Fitchard, "The flying horticulturist: How nokia bell labs AI and drones are helping aerofarms revolutionize vertical farming," Aug 2021.

[10] Y. Lina and Y. Xiuming, "Design of Intelligent Pest Monitoring System Based on Image Classification Algorithm," in *3rd International Conference on Control and Robots (ICCR)*, pp. 21–24, IEEE, Dec 2020.

[11] Y. Zhao, L. Chen, Y. Li, and W. Tian, "Efficient task scheduling for Many Task Computing with resource attribute selection," *China Communications*, vol. 11, no. 12, pp. 125–140, 2014.

[12] A. Ali, M. M. Iqbal, H. Jamil, F. Qayyum, S. Jabbar, O. Cheikhrouhou, M. Baz, and F. Jamil, "An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing,"

[13] J. Xu, C. Liu, X. Zhao, S. Yongchareon, and Z. Ding, "Resource Management for Business Process Scheduling in the Presence of Availability Constraints," *ACM Trans. Manage. Inf. Syst.*, vol. 7, oct 2016.

[14] G. Idoje, T. Dagiuklas, and M. Iqbal, "Survey for smart farming technologies: Challenges and issues," *Computers & Electrical Engineering*, vol. 92, p. 107104, 2021.

[15] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications," *IEEE Access*, vol. 7, pp. 105100–105115, 2019.

[16] P. K. Reddy Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, and Q.-V. Pham, "Unmanned Aerial Vehicles in Smart Agriculture: Applications, Requirements, and Challenges," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17608–17619, 2021.

[17] L. Goel and A. Mishra, "A Survey Of Recent Deep Learning Algorithms Used In Smart Farming," in *2022 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–6, 2022.

[18] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, and X. Wang, "Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies," *IEEE/CAA Journal of Automatica Sinica*.

[19] E. Dahlman, S. Parkvall, J. Skold, and P. Beming, *3G Evolution: HSPA and LTE for Mobile Broadband*. USA: Academic Press, Inc., 2007.

[20] ETSI, "3rd Generation (UMTS)." https://www.etsi.org/technologies/mobile/3g. Accessed: 2023-11-01.

[21] Dahlman, Erik and Parkvall, Stefan and Skold, Johan, *4G: LTE/LTE-Advanced for Mobile Broadband*. USA: Academic Press, Inc., 2nd ed., 2013.

[22] ETSI, "4th Generation (LTE)." https://www.etsi.org/technologies/mobile/4g. Accessed: 2023-11-01.

[23] ITU-R, "Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications-2020 (IMT-2020)," tech. rep., Interatuional Telecommunication Union, 2022.

[24] ETSI, "5G." https://www.etsi.org/technologies/mobile/5g. Accessed: 2023-11-01.

[25] Y. Yifei and Z. Longming, "Application scenarios and enabling technologies of 5G," *China Communications*, vol. 11, no. 11, pp. 69–79, 2014.

[26] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From IoT to 5G I-IoT: The Next Generation IoT-Based Intelligent Algorithms and 5G Technologies," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 114–120, 2018.

[27] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[28] S. Sarkar and A. Debnath, "Machine Learning for 5G and Beyond: Applications and Future Directions," in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 1688–1693, 2021.

[29] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial Intelligence-Enabled Cellular Networks: A Critical Path to Beyond-5G and 6G," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 212–217, 2020.

[30] S. Kim, T. Lee, and K. Kim, "Research on the traffic type recognition technique for advanced network control using Floodlight," in *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pp. 1–6, 2020.

[31] W.-E. Chen, X.-Y. Fan, and L.-X. Chen, "A CNN-based Packet Classification of eMBB, mMTC and URLLC Applications for 5G," in *2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA)*, pp. 140–145, 2019.

[32] J. Li and X. Zhang, "Deep Reinforcement Learning-Based Joint Scheduling of eMBB and URLLC in 5G Networks," *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1543–1546, 2020.

[33] Y.-H. Hsu and W. Liao, "eMBB and URLLC Service Multiplexing Based on Deep Reinforcement Learning in 5G and Beyond," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1467–1472, 2022.

[34] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the Sky: A Tutorial on UAV Communications for 5G and beyond," *Proceedings of the IEEE*, vol. 107, pp. 2327–2375, 2019.

[35] L. Zhang, H. Zhao, S. Hou, Z. Zhao, H. Xu, X. Wu, Q. Wu, and R. Zhang, "A Survey on 5G Millimeter Wave Communications for UAV-Assisted Wireless Networks," *IEEE Access*, vol. 7, pp. 117460–117504, 2019.

[36] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[37] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[38] M. Mehrabi, H. Salah, and F. H. P. Fitzek, "A Survey on Mobility Management for MEC-enabled Systems," in *2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 259–263, 2019.

[39] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[40] S. Zhang, H. Zhang, and L. Song, "Beyond D2D: Full Dimension UAV-to-Everything Communications in 6G," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 6592–6602, 2020.

[41] F. Zhou, R. Q. Hu, Z. Li, and Y. Wang, "Mobile Edge Computing in Unmanned Aerial Vehicle Networks," *IEEE Wireless Communications*, vol. 27, pp. 140–146, Feb 2020.

[42] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," *IEEE Internet of Things Journal*, vol. 6, pp. 2241–2263, 2019.

[43] M. Abrar, U. Ajmal, Z. M. Almohaimeed, X. Gui, R. Akram, and R. Masroor, "Energy Efficient UAV-Enabled Mobile Edge Computing for IoT Devices: A Review," *IEEE Access*, vol. 9, pp. 127779–127798, 2021.

[44] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[45] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2392–2431, 2017.

[46] A. Mehta, J. K. Sandhu, and L. Sapra, "Machine learning in wireless sensor networks: A retrospective," in *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 328–331, 2020.

[47] M. R. Mahmood, M. A. Matin, P. Sarigiannidis, and S. K. Goudos, "A comprehensive review on artificial intelligence/machine learning algorithms for empowering the future iot toward 6g era," *IEEE Access*, vol. 10, pp. 87535–87562, 2022.

[48] W. Lee, M. Kim, and D.-H. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1276–1279, 2018.

[49] A. Zappone, M. Debbah, and Z. Altman, "Online energy-efficient power control in wireless networks by deep neural networks," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2018.

[50] A. Chriki, H. Touati, and H. Snoussi, "Svm-based indoor localization in wireless sensor networks," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1144–1149, 2017.

[51] N. Kuxdorf-Alkirata, G. Maus, and D. Brückmann, "A device-free indoor localization system based on supervised learning and bluetooth low energy," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 413–418, 2021.

[52] H. A. Abbas, N. W. Boskany, K. Z. Ghafoor, and D. B. Rawat, "Wi-fi based accurate indoor localization system using svm and lstm algorithms," in *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pp. 416–422, 2021.

[53] D. Castro-Hernandez and R. Paranjape, "Classification of user trajectories in lte hetnets using unsupervised shapelets and multiresolution wavelet decomposition," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 7934–7946, 2017.

[54] N. Sinclair, D. Harle, I. A. Glover, J. Irvine, and R. C. Atkinson, "An advanced som algorithm applied to handover management within lte," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 1883–1894, 2013.

[55] Y. Chang, X. Yuan, B. Li, D. Niyato, and N. Al-Dhahir, "A joint unsupervised learning and genetic algorithm approach for topology control in energy-efficient ultra-dense wireless sensor networks," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2370–2373, 2018.

[56] L. Zhang, W. Liu, Q. Liu, M. Jin, and S.-J. Yoo, "Unsupervised clustering for nonlinear equalization in indoor millimeter-wave communications," *IEEE Access*, vol. 7, pp. 714–727, 2019.

[57] J. Huang, M. L. Huang, P. H. Tan, Z. Chen, and S. Sun, "Semi-supervised deep learning based wireless interference identification for iiot networks," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pp. 1–5, 2020.

[58] A. Chakraborty, L. E. Ortiz, and S. R. Das, "Network-side positioning of cellular-band devices with minimal effort," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2767–2775, 2015.

[59] M. Camelo, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, and S. Latre, "A semi-supervised learning approach towards automatic wireless technology recognition," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 1–10, 2019.

[60] P. E. Iturria-Rivera, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, "Hierarchical deep q-learning based handover in wireless networks with dual connectivity," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 6553–6558, 2022.

[61] Y. Yao, H. Zhou, and M. Erol-Kantarci, "Deep reinforcement learning-based radio resource allocation and beam management under location uncertainty in 5g mm wave networks," in *2022 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2022.

[62] R. Sarikhani and F. Keynia, "Cooperative spectrum sensing meets machine learning: Deep reinforcement learning approach," *IEEE Communications Letters*, vol. 24, no. 7, pp. 1459–1462, 2020.

[63] S. Saeidian, S. Tayamon, and E. Ghadimi, "Downlink power control in dense 5g radio access networks through deep reinforcement learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[64] W. F. Sharpe, "The Sharpe ratio," *The Journal of Portfolio Management*, vol. 21, no. 1, p. 49–58, 1994.

[65] J. Fernando, "Sharpe ratio: Definition, formula, and examples." https://www.investopedia.com/terms/s/sharperatio.asp. Accessed: 2022-08-01.

[66] F. A. Sortino and L. N. Price, "Performance measurement in a downside risk framework," *The Journal of Investing*, vol. 3, no. 3, p. 59–64, 1994.

[67] R. T. Rockafellar, S. Uryasev, *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[68] Y. Li, "Deep Reinforcement Learning," *CoRR*, vol. abs/1810.06339, 2018.

[69] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy Efficient Resource Allocation in UAV-Enabled Mobile Edge Computing Networks," *IEEE Transactions on Wireless Communications*, vol. 18, pp. 4576–4589, Sep 2019.

[70] J. Zhao, Y. Wang, Z. Fei, and X. Wang, "UAV Deployment Design for Maximizing Effective Data with Delay Constraint in a Smart Farm," in *International Conference on Communications in China (ICCC)*, IEEE, Aug 2020.

[71] P. Cao, Y. Liu, C. Yang, S. Xie, and K. Xie, "MEC-Driven UAV-Enabled Routine Inspection Scheme in Wind Farm Under Wind Influence," *IEEE Access*, vol. 7, pp. 179252–179265, 2019.

[72] C. Zhang, B. Lin, L. Lyu, X. Hu, and S. Qi, "Energy-Efficient Transmission and Computation Resource Optimization for UAV-assisted NOMA-based Maritime Internet of Things," in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, 2023.

[73] H. Guo and J. Liu, "UAV-Enhanced Intelligent Offloading for Internet of Things at the Edge," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2737–2746, 2020.

[74] X. Liu, Z. Liu, B. Lai, B. Peng, and T. S. Durrani, "Fair Energy-Efficient Resource Optimization for Multi-UAV Enabled Internet of Things," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3962–3972, 2023.

[75] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147–3159, 2020.

[76] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak, and H. Yanikomeroglu, "Energy-Efficient Multi-UAV Data Collection for IoT

Networks with Time Deadlines," in *IEEE Global Communications Conference*, 2020.

[77] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Future Wireless Networks: Challenges, Opportunities, and Open Issues," *IEEE Vehicular Technology Magazine*, vol. 14, pp. 70–77, 9 2019.

[78] F. Khoramnejad and M. Erol-Kantarci, "On Joint Offloading and Resource Allocation: A Double Deep Q-Network Approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, pp. 1126–1141, 12 2021.

[79] X. Chen and G. Liu, "Energy-Efficient Task Offloading and Resource Allocation via Deep Reinforcement Learning for Augmented Reality in Mobile Edge Networks," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10843–10856, 2021.

[80] C. Xu, G. Zheng, and X. Zhao, "Energy-Minimization Task Offloading and Resource Allocation for Mobile Edge Computing in NOMA Heterogeneous Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16001–16016, 2020.

[81] K. Wang, J. Jin, Y. Yang, T. Zhang, A. Nallanathan, C. Tellambura, and B. Jabbari, "Task Offloading With Multi-Tier Computing Resources in Next Generation Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 306–319, 2023.

[82] M. A. Ebrahim, G. A. Ebrahim, H. K. Mohamed, and S. O. Abdellatif, "A Deep Learning Approach for Task Offloading in Multi-UAV Aided Mobile Edge Computing," *IEEE Access*, vol. 10, pp. 1–1, Sep 2022.

[83] A. Sacco, F. Esposito, G. Marchetto, and P. Montuschi, "A Self-Learning Strategy for Task Offloading in UAV Networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4301–4311, 2022.

[84] N. Zhao, Z. Ye, Y. Pei, Y. C. Liang, and D. Niyato, "Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-assisted Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[85] C. Yang, B. Liu, H. Li, B. Li, K. Xie, and S. Xie, "Learning Based Channel Allocation and Task Offloading in Temporary UAV-assisted Vehicular Edge Computing Networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9884–9895, 2022.

[86] N. Kiran, C. Pan, S. Wang, and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks," *Journal of Communications and Networks*, vol. 22, no. 1, pp. 1–11, 2020.

[87] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.

[88] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep Learning Empowered Task Offloading for Mobile Edge Computing in Urban Informatics," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7635–7647, 2019.

[89] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, "Deep Reinforcement Learning for Delay-Oriented IoT Task Scheduling in SAGIN," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 911–925, Feb 2021.

[90] M. Akbari, M. R. Abedi, R. Joda, M. Pourghasemian, N. Mokari, and M. Erol-Kantarci, "Age of Information Aware VNF Scheduling in Industrial IoT Using Deep Reinforcement Learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 2487–2500, 8 2021.

[91] N. I. A. Apandi, S. Tian, W. Hardjawana, P. L. Yeoh, and B. Vucetic, "Sharpe Ratio for joint user association and subcarrier allocation design in downlink heterogeneous cellular networks," in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, pp. 1–5, 2017.

[92] S. Wang, S. Xu, S. Guo, P. Yu, H. Meng, and J. Lu, "Risk-Aware Task Offloading and Resource Allocation in FiWi HetNets," in *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, 2022.

[93] Z. Ling, F. Hu, Y. Zhang, F. Gao, and Z. Han, "Distributionally Robust Chance-Constrained Optimization for Communication and Offloading in WBANs," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–6, 2020.

[94] B. Zhou, W. Saad, M. Bennis, and P. Popovski, "Risk-Aware Optimization of Age of Information in the Internet of Things," in *International Conference on Communications*, pp. 1–6, IEEE, 2020.

[95] W. Hoiles, S. M. S. Tanzil, and V. Krishnamurthy, "Risk-Averse Caching Policies for YouTube Content in Femtocell Networks using Density Forecasting," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 331–346, 2021.

[96] Papers With Code, "Image Classification." https://paperswithcode.com/task/image-classification, 2023. Accessed: 2023-12-01.

[97] A. Bouguettaya, H. Zarzour, A. Kechida, and A. M. Taberkit, "Deep learning techniques to classify agricultural crops through UAV imagery: a review," *Neural Computing and Applications*, vol. 34, p. 9511–9536, Mar. 2022.

[98] T. Pamuklu, A. C. Nguyen, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "IoT-Aerial Base Station Task Offloading with Risk-Sensitive Reinforcement Learning for Smart Agriculture," *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2022.

[99] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181176–181191, 2020.

[100] OpenSim Ltd., "What is OMNeT++?." https://omnetpp.org/intro/, 2019. Accessed: 2021-10-19.

[101] HSE - Unmanned Aerial Vehicles (UAV), "High Power Drone Battery 6S-HV (LiHV) 25000mah (22.8V)." https://hse-uav.com/product/high-power-drone-battery-6s-hv-lihv-25000mah-22-8v/, 2021. Accessed: 2021-05-01.

[102] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, pp. 70–85, 1 2017.