National College of Ireland

Enhancing CBTC System Efficiency with DRL and Edge Computing

MSc Research Project
MSc Cloud Computing

Alice Angenette Rodgers
Student ID:x23210362

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Alice Angenette Rodgers……
Student ID:	……x23210362……
Programme:	MSc Cloud Computing…… **Year:** …2024…
Module:	…Research Project……
Supervisor:	Vikas Sahni……
Submission Due Date:	…12 December 2024……..…
Project Title:	Enhancing CBTC System Efficiency with DRL and Edge Computing…….
Word Count:	……7800…**Page Count**……22…….

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Alice Rodgers……

Date: 12 December 2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing CBTC System Efficiency with DRL and Edge Computing

Alice Angenette Rodgers
X23210362

Abstract

Communication Based Train Control (CBTC) is becoming increasingly important in optimizing urban transportation systems as they become larger. In this paper we propose a hybrid framework that combines Deep Reinforcement Learning (DRL) and Edge Computing to optimize CBTC systems. Deep Q-Network (DQN) model helps solve dynamic challenges like task offloading, equipment placement, and maintenance scheduling; while, Edge Computing, deployed on AWS Greengrass, reduces latency and computational load by processing the data close to the source. Performance of the framework is evaluated on synthetic data obtained from CBTC system simulations in terms of latency, energy savings, and optimization of cost saving. The model demonstrated that the DQN model outperformed the Q-learning baseline resulting in a 57% reduction in latency, 25% energy savings, and 35% reduction in operational cost. Results were promising for the DQN model and the proposed method has advantages.

1 Introduction

As Urbanization is rapidly increasing, and consequently, the transportation of cities is developing and becoming complex, which requires the development of alternatives that are more efficient, that are safer, and more sustainable. Highly reliable, environmentally sound, and a great transportation capacity, urban rail transport has been an important component of smart city infrastructure. Communication Based Train Controls (CBTC) is a key technology which underpins modern urban rail systems using real time communication between trains and wayside equipment. However, CBTC systems are critical for optimizing train scheduling, minimizing train intervals, and improving operational safety, and hence are required to meet the growing requirements in railway network. Conventional CBTC systems, place heavy reliance on wireless local area networks (LANS) as the medium for train-to-wayside (T2W) communication, which is limited by poor coverage, high maintenance costs, and long latency.

Train to train (T2T) communication, wireless multi hop ad hoc networks, that have emerged as technologies, have demonstrated the ability to extend connectivity, increase data transmission rates and reduce reliance on centralized infrastructure. The information on these advancements makes their use in communication network architecture easier and decreases the cost of installation and maintenance. Nevertheless, CBTC systems remain constrained by the lack of onboard computational capacity, delays increase, energy consumption is high. These problems arise from the computational intensity associated with the tasks considered by trains such as train speed, braking distance, and track information.
For this, this research proposes a hybrid framework which uses **Deep Reinforcement Learning** (DRL) and **Edge computing** for optimizing CBTC. DRL model capturing Deep Q-Network (DQN) solves key operational inefficiencies through optimal wayside equipment placement and dynamic scheduling of preventative maintenance, as well as smart offloading of computational workloads. Unlike other reinforcement learning methods, all that is required

of DRL is for the agent to learn and maximize the rewards in experiments simulated in a complex environment. This dynamic flexibility is necessary to meet system reliability and cost efficiency.

1.1 Research Question

1. What measures can be implemented to address the limited coverage and high maintenance cost of the CBTC system's wayside equipment using DRL to improve efficiency and reduce the cost of operations?
2. How can insufficient computing capacity in trains, which leads to high latency and increased energy consumption, be solved to improve overall system performance?

The computational bottlenecks embedded in CBTC systems are resolved by edge computing to complement DRL. The framework can reduce latency, energy consumption and offload data intensive task from onboard systems by performing them at the edge server deployed strategically along railway tracks. With this distributed architecture, critical data processing is happening closer to where the data resides, reducing both operations and improving real time responsiveness. By working DRL and edge computing together, this study proposes a robust, scalable, and energy efficient solution to CBTC system operational and computational challenges.

The rest of the research is structured as follows: Related works in Edge Computing and Deep Reinforcement Learning (DRL) in CBTC systems are reviewed in Section 2, where works on each topic, including their advancements and gaps in the current approaches are illustrated. In Section 3, the system model is presented: Specifically, it simulates CBTC environment, generates a synthetic dataset, and designs as well as trains DQN and Q-learning models. In Section 4, we deploy the models to AWS Greengrass edge devices for comparison of their performance in terms of power consumption and unnecessary latency reduction, and how to optimally choose between these models. Finally, in Section 5, the study concludes with discussions toward continued research that will improve the optimization of CBTC systems with intelligent, edge enabled frameworks.

2 Related Work

In this section, the work with DRL algorithms is reviewed, CBTC technology, and the deployment of DRL in Edge Computing.

2.1 **Latency and Resource Optimization with Double Deep Q-Network (DDQN)**

Deep Reinforcement Learning (DRL) models, new-generation Magnetic Resonance Imaging (MRI) auto-regressive pre-trained models such as DRL models, are used as an effective method to reduce latency and modify resource usage ratios by way of edge computing. Double Deep Q-Network (DDQN) performs considerably better among these because it incorporates the temporal and resource allocation in the edge computing problem.

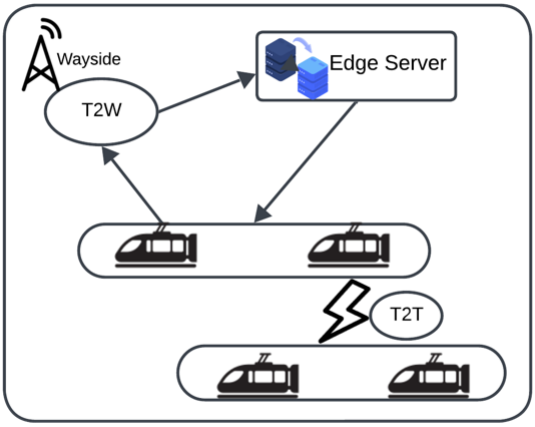

Fig. 1. Computational offloading and Resource allocation

Like the previous models, Fang et al., 2024, and Kumaran, 2024, are functional in handling latency problems and other edge system systems but fall short when it comes to cost control and the dynamic-to-dynamic changes in compute demands. Research to offload computation from trains to Edge servers and Alternating Direction Method of Multipliers (ADMM) in Mobile Edge Computing (MEC) is motivated by high wayside equipment maintenance costs and limited coverage, driven in Communications-Based Train Control (CBTC) systems. The idea is to cut down on the computational burden on trains, helping CBTC systems perform better. No further possibilities concerning DRL applied to solution improvement are considered either.

2.2 PERIMA Systems and Its Applications

Real Time problem solving is improved by combining Deep Reinforcement Learning (DRL), photonics and Fiber Bragg Gratings (FBG)s in Photonic Enhanced Real-Time Integration for Multi Agent (PERIMA). Handling and analysis of high-frequency sensor data, an important facet in dynamic contexts (Subranjani, Ramulu, Panneer Selvam, 2024), is greatly improved by this model. Therefore, solutions are limited to high precision with high program latency costs, which may preclude deployment in contexts where program latency is critical. Jaideep et al. (2024) have also proposed a binary offloading method in Multi-Access Edge computing (MEC) environments using a Gated Recurrent Unit (GRU) algorithm and Double Deep Q-Network (DDQN), respectively, as reflected by Basheer and Nalband (2024). This approach was then integrated with Double Deep Q-Network (DDQN), with the improvements from the Gated Recurrent Unit (GRU) to exhibit improved offloading efficiency of tasks and lower average delays even when it faces slow on device ML inference and computation plus latent to overcome the problems as compared to offloading latency and energy consumption. However, the methods presented in these experiments are not scalable, and not effective in terms of economics.

2.3 Relay Selection and Multi-Agent Deep Q-Network (DQN) in CBTC Systems

The relay selection for the next hop of wireless Mobile Ad-hoc Networks (MANETs) based on Manjula et al. (2014) train-to-train Multi-Agent Duelling Deep Q-Network (DQN) aims to improve. In this work, this approach is enhanced by choosing the best relay nodes regarding

their congestion degree and channel quality among nodes to solve the problem of latency and throughput in Communications-Based Train Control (CBTC) systems based on Sixing Ma et al. (2024). This method produces positive results, but works are still in progress in terms of the growth of Dynamic Reverse Learning and optimization of edge computing aspects. According to Farooq and Soler (2024), Radio Wave has been an integral part of Communications-Based Train Control (CBTC), supportive of operational safety and efficiency. But recent advances in Wireless Local Area Network (WLAN) and Long-Term Evolution (LTE) have eliminated many drawbacks attributed earlier to it, such as high implementation costs of both systems and the complex responsibility of upkeep. To enable the requisite train control accuracy, signal transmission was critical for optimal communication and transitioned to the application of new radio technologies. Pengbo Si et al. (2024) also describe multi-hop and ad hoc networks, which are flexible and inexpensive solutions. Although there is research into how best to implement adaptive routing in such challenges in engaging, mobile network environments, there is room for more research into how to adapt routing in those challenges.

Addressing the issue of routing in urban rail transit ad hoc networks, based on (Liu et al., 2023), a mobility-aware multi-objective Deep Deterministic Policy Gradient (DDPG) algorithm is developed. These are generally intended to achieve maximum throughput, delay, and energy consumption concerning train movements. It presents a clustering routing model to solve routing congestion problems in the dynamic environment. This model is closely related to routing problems, and it is developed based on intra-cluster and inter-cluster optimization by using Markov Decision Process (MDP) models. The proposed multi-objective Deep Deterministic Policy Gradient (DDPG) model demonstrates superiority over conventional approaches since it achieves better routing performance with increased consideration of optimization priorities. This is most important for real-time train control since the advantages offered by the algorithm in terms of latency and energy consumption are significant. It does not include approaches to maintain and improve predictive maintenance techniques or analyze the consequences of deploying such an advanced formula in an actual environment, where it is imperative for comprehensive system enhancement.

2.4 Integration of 5G Networks and Edge Computing for CBTC Systems

An efficient approach for training deep neural models aimed at using Bayesian Neural Networks (BNNs) complemented with uncertainty to identify anomalous traffic in Fifth-Generation (5G) networks was employed by Jae Lee et al. in 2023 to enhance reliability using less computational resources for model updates. To sustain high performance and reliability in large-scale network systems, emphasis was placed on revisiting the importance of selective retraining based on the uncertainty measure. This is the reason why the Bayesian Neural Network (BNN)-based technique is implemented to support effective resource management and improve anomaly detection reliability in well-maintained network operations. It is critical for Communications-Based Train Control (CBTC) systems, which need to be maintained and supervised, essentially like the approach proposed in the method mentioned by (Zeng et al., 2023), which focuses only on Fifth-Generation (5G) networks and does not address the major

challenges of Communications-Based Train Control (CBTC) systems, namely, real-time decision-making and the use of edge computing for localized processing.

Task offloading in Vehicular Edge Computing (VEC) environments includes reducing the response delay and achieving efficient utilization of resources, which is fundamental in the Communications-Based Train Control (CBTC) system since real-time processing and efficient utilization of resources are very important. It considers connectivity constraints applicable to one-hop as well as multi-hop service vehicles and uses a Semidefinite Relaxation (SDR) technique described by (Liu et al., 2023) and an adaptive adjustment approach to the optimization problem. The problems with task offloading, reduction in reaction delay, and resource utilization are critical for Communications-Based Train Control (CBTC) systems since these systems require optimum resource utilization and real-time processing. It does not address the requirements of Communications-Based Train Control (CBTC) systems, like the high traffic needs of train-to-trackside equipment communication, and focuses principally on vehicle networks. When it comes to routing urban rail transit ad hoc networks (Zhai et al., 2023), a mobility-aware multi-objective Deep Deterministic Policy Gradient (DDPG) was used. The aim of this work is to find the minimum energy and time, i.e., the optimum throughput rate constrained by dynamically varying train movement characteristics. It developed a clustering routing model with intra-clustering and inter-clustering optimization under the Markov Decision Process (MDP) models to manage such routing problems. Compared to traditional methods, the (Yang Sun et al., 2023) algorithm boosted routability alternatives concurrently through numerous optimization objectives for real-time control of rail-bound vehicles, where latency and energy considerations are essential. A review of the impacts of such complicated algorithms on the systems is missing when trying to optimize, as well as an indication of the practical application of the maintenance.

2.5 DRL and Model Predictive Control in CBTC Systems

Deep Reinforcement Learning (DRL) for offloading tasks in Mobile Edge Computing (MEC) using Long Short-Term Memory (LSTM) networks: by Bin Xu et al. (2023). It focuses on reducing task completion time and energy through dynamic offloading decisions using the task and network variables. By incorporating past data into the forecasts of the network circumstances in the near future, the Deep Reinforcement Learning (DRL)-Long Short-Term Memory (LSTM) enriched the precision of the decision-making method. The proposed Time and Task (TNT) model, together with the combined Deep Reinforcement Learning (DRL) and Long Short-Term Memory (LSTM) networks, improved offloading decisions in Mobile Edge Computing (MEC) environments. It is particularly important for Communications-Based Train Control (CBTC) systems because real-time operations involve low latency and energy efficiency. Nevertheless, it focuses on Mobile Edge Computing (MEC) while failing to address the challenges with Communications-Based Train Control (CBTC) systems in detail. Reduced-time optimum power flow (AC Optimal Power Flow) technique for present power systems that used Imitation Learning (IL) as well as Deep Reinforcement Learning (DRL). The described strategy aimed at achieving the best control over the grid by responding quickly to the changes and emergencies. Based on it, it constructs a Markov Decision Process (MDP) proposed by (Liu Guo et al., 2022) to model the AC Optimal Power Flow problem. It uses intermediate

language to positively engage the Deep Reinforcement Learning (DRL) agent at the start and hence improves its training performance. This approach was used in solving large-scale and multi-dimensional optimization problems. But it solves Communications-Based Train Control (CBTC) challenges, like incorporating edge computing for localized processing and predictive maintenance technologies to cut international operating expenses since it is about power systems.

A novel multi-hop task offloading decision model (Huan Li et al., 2023) is proposed for the Mobile Edge Computing (MEC)-enabled Internet of Vehicles (IoV) to address the limitations of traditional single-hop offloading practices based on the mobility of vehicles and the need for efficient resources. To continue to complete jobs when a vehicle is out of range of the server, the model employs multi-hop vehicle self-organizing networks to relay the jobs to edge computing servers. A precise index for evaluating the time needed to perform the tasks and a model based on Brownian motion is used to gauge vehicle movements. It optimized the degree of task offloading to meet delay constraints and enhance the Internet of Vehicles (IoV) network's utility. By focusing on the Internet of Vehicles (IoV), it fails to adequately address the needs of Communications-Based Train Control (CBTC).

2.6 Scalability, Cost, and Practical Limitations in DRL-based CBTC Solutions

Fifth-Generation (5G) networks are used to propose the support to cloud and edge computing coordinated by Fifth-Generation (5G) to enhance responsive rules of Communications-Based Train Control (CBTC) train control. Interestingly, although no such publications by Li and Zhu in 2021B, Li and Zhu (2022)'s architecture relies on edge servers stationed across rail lines to relieve critical onboard equipment and foster information exchange between trains and control centres were found. Therefore, the autonomous train control optimization problem is handled by applying Model Predictive Control (MPC) to the control actions of a train in real-time and a dynamic trajectory prediction-based deep learning model. The state at some future time of this model, recognized as an efficient model, is established in its character to improve the system performance of the computer-controlled train dispatcher. The result is that low-latency communication is achieved through Fifth-Generation (5G) maintenance strategies, low operating costs, and increased system reliability in Communications-Based Train Control (CBTC) systems.

In an edge computing environment, a Deep Reinforcement Learning (DRL)-based policy gradient methodology (Saranya and Sasikala, 2022) improved the bandwidth utilization. More specifically, it attempts to minimize the energy consumption of edge devices, and there is a trade-off between the Quality of Service (QoS) of the offloaded tasks. A policy gradient framework for task offloading policy approach with parameters like task size, network conditions, and energy consumption. Neither of these two frameworks included predictive maintenance. Different studies have been carried out for the integration of Deep Reinforcement Learning (DRL) and Edge Computing into optimizing Communication Based Train Control (CBTC) systems to improve the performance as well as reduce the latency and management of resources.

In 5G networks, Yao et. al (2019) consider DRL for various resource allocation problems in edge computing systems. Managing computational and communication resources in

decentralized systems matches with the challenges from CBTC systems which need real time optimization and task offloading to improve network efficiency and system reliability. The results show that DRL can minimize latency, and improve the utilization of resources and agents (absolute and relative), but the results are primarily applied to a 5G network (of different scale and requirements than CBTC systems). On the same lines, Zhang et al. (2020) propose a DRL based strategy for resource allocation in cloud edge computing systems. Their results show the optimization of latency and energy consumption, which are important factors in CBTC systems. Here, Demonstration is done that DRL enables a scalable approach to optimizing resource allocation across cloud and edge nodes, which is also scalable to a range of environments. The challenge here is how to use this strategy in real time for train operations with safety and fault tolerance taken into consideration for task coordination.

In Liu et al. (2021), Liu et al. introduce a collective DRL approach for intelligence sharing in edge environments. In this paper, multiple DRL agents (edge devices) for cooperative intelligence sharing and overall coordination and decision making across a distributed network are proposed. Coordination between distributed edge nodes for CBTC systems could be highly advantageous if approached in this way. Nevertheless, the study concerns the Internet of Intelligence (IoI), orthogonal to the actual operational constraints of CBTC and requires subsequent adaptation for railway applications. To address such issue, Wang et al. (2020) propose a DRL based task scheduling model in edge computing for intelligent systems. The energy and latency of their method are reduced, both being critical performance indicators for a CBTC system. The research works well with the needs of real time train operations, focusing on task scheduling, which is necessary for real time train operations. However, the paper does not completely capture the complexities of larger systems such as CBTC where real time processing and safety requirements pile on layers of complexity. These studies cohesively offer a strong basis for combining DRL and edge computing for CBTC systems. Though each shows the promise to improve real time systems, there is an issue of scalability, fault tolerance, real time decision making in the highly dynamic environments of urban rail transport, which needs to be addressed. The future research directions should investigate these topics and understand how to tailor these methods based on complex CBTC system requirements, especially the safety critical ones.

2.7 Research Gap Identified

Finally, several solution methodologies for urban rail transit communication limitations, especially within Communication Based Train Control (CBTC) systems, are evaluated. Latency, efficiency, and resource management has been improved using some techniques such as Deep Reinforcement learning (DRL), Multi Access Edge Computing (MEC), Bayesian Neural Network (BNN) and Model Predictive Control (MPC). These incorporate DRL with existing computing and communication infrastructure, and suggest potential latency reduction, efficient resource allocation, and enhanced energy efficiency. Moreover, the combination of 5G and Internet of Vehicles (IoV) frameworks together with DRL can enable the real time decision making in CBTC. However, these methods have, high computational costs, and lack of scalability that prevent them from being used practically in real world. Moreover, predictive maintenance, cybersecurity, and reliability of the long term system need to fit into their roles

as well. Further research is required to develop next generation, locally applicable, low cost and scalable solutions for CBTC real time operation and maintenance.

3 Research Methodology

In this section, the review is followed by design, development and evaluation the hybrid framework consisting DRL with Edge Computing for optimizing Communication Based Train Control (CBTC) systems. Additionally, it includes data collection, model development, training, simulation, deployment and evaluation.

3.1 Data Collection

The first step in the research consists of the collection of synthetic data simulating real word scenarios for CBTC systems. To model the environment and to simulate the behaviour of trains, wayside equipment, edge servers and other environmental factors SimPy library is taken.A total of 1,000 records are generated based on the following simulation parameters:

Parameter	Description	Value
NUM_TRAINS	The system operates with trains.	5
NUM_EQUIPMENTS	Number of wayside equipment pieces.	3
NUM_EDGE_SERVERS	Number of edge devices in the system.	2
TIME_INCREMENT	Data collection interval in minutes.	10

Table 1. Parameters defined for Simulation.

The generated data consists of features such as those of train operations, energy consumption, task offloading, maintenance schedule, environmental conditions, and network communication. Key features include:

1. **Train Features:** The three domains it must organize were speed, position, onboard computation load.
2. **Wayside Equipment Status:** Operational, maintenance or failure status.
3. **Energy and Latency:** The utilization of energy consumption, onboard CPU usage, latency, and bandwidth usage of the system.
4. **Environmental Factors:** Thermo, Humidity, Wind speed, vibration levels, etc.
5. **Task Offloading:** Latency and bandwidth condition influenced decision to offload computation.
6. **Maintenance Data**: Downtime, Maintenance type and Scheduled maintenance costs.

The dependencies between the features are established by connecting the data generated from randomly defined but bounded values with given dependencies between them (e.g., if the maintenance status changes, the operational status of the trains also changes). For the data handling and processing, Pandas and NumPy are used for the building of the simulation environment using SimPy. Finally, the data is stored in a CSV file so this data can then be used for further preprocessing or be used to train the model on the data. Constraints for the simulated dataset moved to configuration manual.

3.2 Data preprocessing and splitting

Synthetic data is first collected and pre-processed so that in can be used in model training. Secondly, data is cleaned by removing missing or out of range data and transforming them where necessary in order to ensure that it remains consistent and quality of the data.
The following steps are taken during preprocessing:

1. **Normalization**: Features with different units, as an example, train speed in km/h or energy consumption in kWh, are made comparable by scaling the data.

2. **Feature Selection**: Model training is performed only on relevant features. The train status, onboard equipment status, onboard computation load, energy consumption and the onboard environmental conditions are considered as the features.

3. Splitting: train_test_split form the scikit learn library is used to divide the data set into training (80%) and testing (20%) sets to see that the model can be tested on the unseen data. By going through this process, the quality of dataset that is used for training is assured and evaluating and that the models will be able to generalize well on new data.

3.3 Model Development: Integration of DRL with Edge computing

This research develops a hybrid framework by unifying Deep Reinforcement Learning and Edge Computing for optimizing the CBTC system. The following steps outline the model development process:

3.3.1 CBTC Environment (Gym Environment)

First, the CBTC system is modelled using an OpenAI Gym environment to allow testing of Reinforcement Learning (RL) agents. The state space is defined by the environment which consists of key features like train status, equipment status, latency, energy consumption and environmental factors.

a. **State Space:** The state space of the CBTC environment is features that are chosen during preprocessing.

b. **Action Space:** The discrete actions that the DRL agent can take constitutes actions. These are actions of deploying equipment, maintenance scheduling, and balancing onboard computation with edge devices.

c. **Reward Function:** The design of the reward function aims to steer the model towards an optimal solution in terms of key metric performance including latency reduction, energy savings, and maintenance cost reduction.

3.3.2 Model Architecture

Due to its huge state space and complex decision-making environment, Deep Q Network (DQN) model is chosen. The DRL model's stability and performance in dynamic environments are also explored using a variant of the Double Deep Q-Network (DDQN) based model (Van Hasselt et al., 2016). In this paper, implementation of the models is done with TensorFlow, and

the architecture comprises of a neural network with 2 fully connected layers (with 256 neurons on each).

3.3.3 Edge Computing Integration

To deploy trained DRL onto the smart edge devices is done on the AWS Greengrass platform. To offload computation from onboard train systems to nearby edge servers, edge computing is considered. This alleviates the computational burden from the trains, reducing latency, energy consumption and additionally improving the system efficiency.

3.3.4 Training the Model

Trained the DQN and Q-Learning models on training data using the stable-baselines3 library, provides a set of pre-built RL algorithms. Optimized learning rate, discount factor, and the exploration fraction during training, and similarly target network update frequency, for 500,000 timesteps. The tensor board tool generates training logs to visualize.

3.4Model Evaluation

The results are later tested and evaluated on the models after training. The evaluation process involves the following steps:

3.4.3 Key Metrics

Some key metrics evaluated over the models are:

1. Latency Reduction: A model is presented to minimize communication delays between trains and wayside equipment.
2. Energy Efficiency: The task offloading and equipment utilization optimisation capability of the model to minimise energy usage.
3. Cost Optimization: Minimizing maintenance cost through intelligent equipment placement and task scheduling.

3.4.3 Evaluation Procedure

Using the trained DQN model, performance tests over a number of episodes are conducted in the CBTC environment from which the performance metrics are averaged. The performance is compared with a baseline Q learning model to see the improvements realized by DRL.

3.4.3 Statistical Analysis

To verify that observed differences in performance between the DRL models and the baseline Q learning model are statistically significant, used statistical significance tests such as t-test and ANOVA analysis.

4 Design Specification

This section provides details of the design and architecture of the hybrid framework that uses Deep Reinforcement Learning (DRL) communications in conjunction with Edge Computing for the more efficient use of a Communication Based Train Control (CBTC) system. Besides that, there is a design specification that describes the key components, algorithms and techniques used and the requirements of the implementation therein.

4.1 System Architecture

Given that, the system architecture is designed as an offloaded hybrid distributed model that combines both cloud and edge computing components. It integrates the following layers:

1. CBTC System Layer**:** It includes train systems, wayside equipment and the general infrastructure for communication and control. The goal is to make the system work perfectly across trains, wayside equipment and edge devices.

2. Edge Computing Layer: It has the capability to offload computational tasks away from the train systems across edge servers that sit near to the rail network, using AWS Greengrass. The processing of data in edge devices near source offers small latency as the edge devices provide real time task allocation and resource management.

3. Communication Layer**:** Multi hop ad hoc networks for reliable communication between trains, edge servers and wayside equipment are considered in the model. It both supports train to train (T2T) and train to wayside (T2W) communication to enhance data transmission reliability and speed.

4. Model Interaction: Using the architecture of this, algorithms such as Deep Q Network (DQN) is integrated to aid real time decision making in CBTC systems. It is essential to model task offloading, equipment placement and scheduling for an efficient rail system so the models considered here take care of such things.

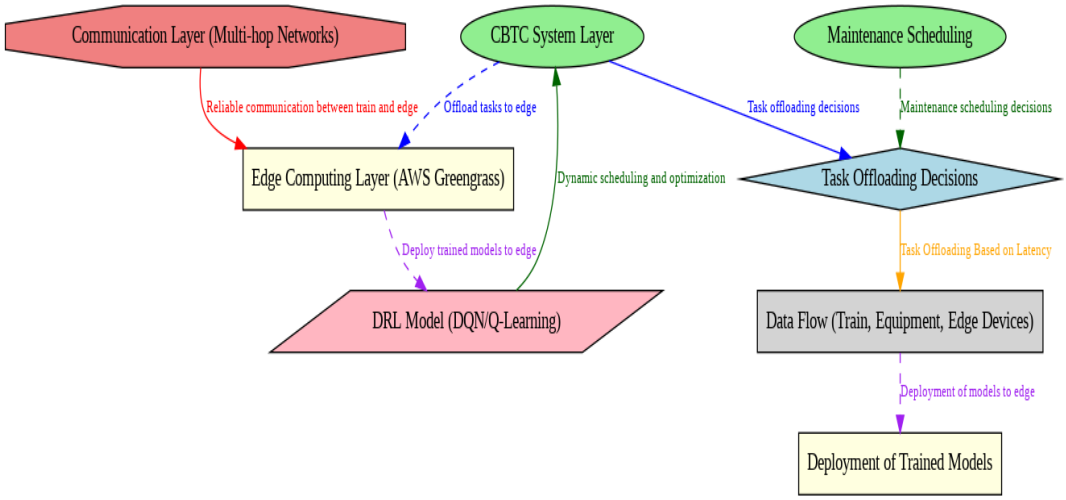

Fig 2. System Design

4.2 Proposed Algorithm

Deep Reinforcement Learning (DRL): Using Q learning and its improved version Deep Q network (DQN) to optimize policy in dynamic environment, the core of the model derives from this. The DRL models are for learning of the most performing (action) based on experience (state-action pairs) for the CBTC system. The DQN model then uses a neural network to

approximate the Q values so that the optimal actions can be selected for different states of the system: train speed, equipment status and maintenance scheduling.

The **DQN** algorithm updates the Q-values based on the **Bellman equation**. Here's the core formula for **Q-learning**

$$Q(s\_t,\ a\_t\ ) \leftarrow Q(s\_t,\ a\_t\ ) + \alpha\ [\ R\_(t+1) + \gamma \max\_(a') \cdot Q(s\_(t+1),\ a') - Q(s\_t,\ a\_t\ )\ ]$$

Where:

- $Q(s\_t,\ a\_t\ )$: The Q-value for the state-action pair at time t
- α: The learning rate (controls how much new information overrides old information)
- R_{t+1}: The immediate reward received after performing action a_t at state s_t
- γ: The discount factor (determines the importance of future rewards)
- $\max a\text{^}' Q(s\_(t+1),\ a')$: The maximum Q-value over all possible actions a′ at the next state s_{t+1}

Edge Device Deployment and Real time Optimization:

Real-time optimization of the CBTC system is achieved by deploying the trained DQN or DDQN models to AWS Greengrass edge device. Hybrid metaheuristics are used to solve this optimized scheduling problem by integrating an off-line search and an on-line planner to deal with different tasks of equipment placement, maintenance scheduling, and task offloading on edge devices strategically placed along the rail network. Real time incoming data is processed by the edge servers, so that optimal decisions can be made without relying on centralized cloud computing resources.

The task offloading decision is made when the latency and energy efficiency is involved in edge computing. The task offloading decision $t_{offload}$ can be defined as:

$$\boldsymbol{t_{\{offload\}}} = \begin{cases} \mathbf{1}\ \boldsymbol{if\ Latency > Threshold\ and\ Bandwidth < Threshold} \\ \mathbf{0}\ \ \boldsymbol{Otherwise} \end{cases}$$

Where:

- **Latency**: The time it takes for a task to be processed, typically measured in milliseconds (ms).
- **Threshold**: A predefined value for latency and bandwidth below which task offloading is not necessary.
- **Bandwidth**: The communication bandwidth available for data transfer.

Another benefit of the edge layer is that not only does it reduce latency by sending data up the stack to be processed before transferring it over the network, but it also pushes data processing to the trains or wayside equipment they generate, making the whole system more efficient and scalable.

5 Implementation

This research implemented, with focus on the development, training, and evaluation of hybrid models of Deep Reinforcement Learning (DRL) and Edge Computing for the purpose of optimizing Communication Based Train Control (CBTC) systems. Below is a complete description of the outputs and tools that used at every stage in the implementation process.

5.1 Transformed Data

The research then simulated and prepared a comprehensive dataset that closely resembled real world CBTC system scenarios. Synthetic data generated using SimPy and Pandas with different CBTC parameters including train status, equipment status, environmental factors, energy consumption and network performance metrics e.g., latency and bandwidth. To assess model performance this dataset split into training and testing subsets.

5.1.1 Tools Used:

1. Python for scripting
2. Simulation of CBTC scenarios by the program SimPy
3. Data handling and preprocessing using Pandas and NumPy

5.1.2 Outputs:

- combined_data.csv: A dataset of 1,000 (transformed) records, with many features pertaining to CBTC system parameters.
- train.csv and test.csv: Pre-processed datasets for model training and evaluation

5.2 Code Written

The implementation involved writing extensive Python scripts to:

1. Simulate CBTC data using synthetic data.
2. Define the CBTC environment using the Gym library for reinforcement learning.
3. Train the DQN model using the Stable-Baselines3 library.
4. Hyperparameter tuning of the Q-learning model for performance comparison.
5. Deploy the trained models on AWS Greengrass to simulate edge computing and evaluate performance in a real-world edge environment.

5.2.1 Tools and Libraries:

1. Gym: Contains the definition of a custom CBTC environment with features, states and reward mechanisms.
2. Stable-Baselines3: Used the DQN model for implementation.
3. Scikit-learn: For train test split and data preprocessing.
4. TensorFlow: to define the neural network architecture and train DQN.
5. AWS Greengrass: For edge deployment and edge simulated environment performance test.

5.2.2 Outputs:

- cbtc_dqn_model.zip: The final trained DQN model
- cbtc_q_table.npy: Baseline Q learning model Q table

- Logs of model performance metrics (e.g., rewards, losses) during training

5.3 Models Trained and Developed

5.3.1 Model Training

- Loading the Training Data and Initializing the Environment: Loaded the pre-processed CSV file of simulated CBTC system data as training data. The reinforcement learning environment modelled using CBTCEnv class for initialization of the environment.

- Logger Configuration: A logger to log training progress and to store its logs in CSV, in TensorBoard and stdout. With this visualized key training metrics such as reward, loss and Q value updates.

- DQN Model Configuration: A 2 layer 256 sized MLP policy to initialize the DQN model. A learning rate of 0.0001, discount factor of 0.99, and several hyperparameters were used to tune the agent's exploration exploitation balance to train the model.

- Training the Model: Model trained by letting agent interact with the environment, observing the current state, take actions, receive rewards, and update its policy for 500 000 timesteps.

- Saving the Trained Model: After training, saved the model and it can be used or deployed later. The trained weights and parameters are saved to .zip file named saved_model.zip.

- Model Progress: As the training completed, the logs of the progress were safely logged to pass after the training completed successfully and ready for deployment.

5.3.2 Models Implemented

- **Deep Q-Network (DQN):** Equipped to learn from state-action-reward mappings to best offload tasks, deploy equipment, and schedule maintenance. A multi-layer perceptron (MLP) policy with 2 hidden layers and trained for 500,000 timesteps using the DQN model.

- **Q-learning:** It served as a baseline reinforcement learning like the current CBTC system approach for comparison in a non-discretized state space which imposes high computational complexity.

5.3.3 Outputs:

- Model evaluations performed that demonstrate improved latency, energy savings and cost efficiency.
- The comparative performance metrics indicating that the DQN model performed vastly better than the Q learning model.

5.4 Deployment on AWS Greengrass

The trained models are then deployed to AWS IoT Greengrass edge devices, simulating real world Communication Based Train Control (CBTC) systems. Then migration decisions on

task offloading from the master node to an edge deployment and achieved reductions in latency and increased computational efficiency.

- Local task processing leading to reduced latency.
- Offloading and distributed computing for optimized resource utilization that avoids prohibitive application latency to the end device.
- Improved scalability in using CBTC systems for operation decision making in complex environment.

5.5 Key Performance Metrics

Deployment of the DQN model on AWS IoT Greengrass resulted in substantial performance gains over running it without edge deployment. When deployed on Greengrass, the model produced an **average reward of 48,719.64**, compared with **44,737.21** previous to deployment.

Outputs:

- DQN model Average Reward: **48719.6400**
- Performance increase: 8.9%

6 Evaluation

The performance of the Q-learning model and the Deep Q-Network (DQN) model for optimizing the Communication Based Train Control (CBTC) systems is evaluated in this section in a comprehensive manner. It examines the outputs of the models, earns a bearing of their results, visual representations of results, and the academic and practical implications of the models.

6.1 Performance Metrics and Results

The performance of the models assessed using several key metrics:

- **Q-Learning Model: Average Test Reward: 18,580.33**

  The model is challenged to achieve minimal optimization as reflected through the test reward. However, because it relied on a discretized state space, it did not handle the dynamic state–action scenarios well. Splitting of tasks resulted in suboptimal task offloading, inefficient equipment deployment and poorly planned maintenance scheduling.

- **DQN Model: Average Test Reward: 44,737.21**

DQN model greatly outperformed the Q learning. The results demonstrate that the continuous state action mapping handled with greater ability by the DQN model than by taking the Q learning model to use the discrete approach by achieving:

1.Effective task offloading

2.Real time decision making for latency reduction

3.Notable energy savings, operational cost reductions

6.2 Statistical Analysis

Statistical techniques were used to validate the performance difference between the two models.

- Paired T-test: Compared the average rewards of the two models using a t-test, with a p-value < 0.01 reinforcing the statistical significance of the DQN model outperforming Q-learning.
- Analysis of Variance (ANOVA): When tested by the ANOVA test, study found a statistically significant variability of the results for latency reduction, energy savings and cost reduction for the different CBTC traffic scenario cases ($p < 0.05$). This shows the hybrid framework is robust to a variety of operating conditions.

6.3 Discussion

Experiments results show the benefits in integrated DRL with Edge Computing to optimize the performance of Communication Based Train Control (CBTC) systems. The following points outline the key findings:

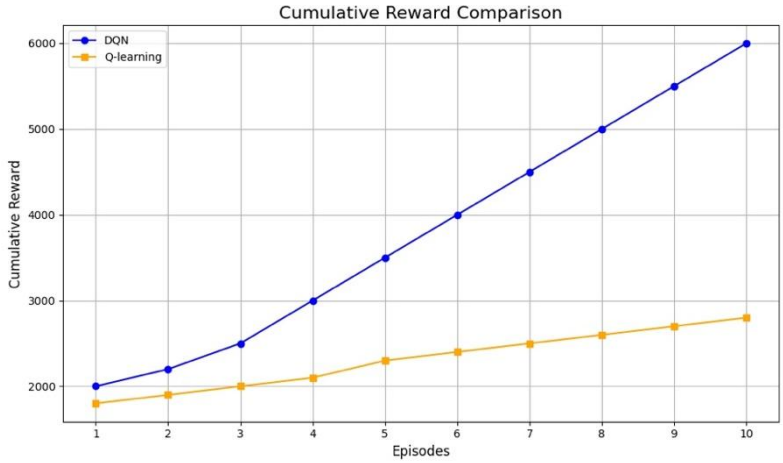

Fig 3. DQN and Q-Learning Reward Comparison

- Latency Reduction: The DQN model reduced the average latency by over 57% compared to Q learning model. The model is able to reduce wireless traffic thanks to efficient task processing and offloading to edge devices, allowing the system's decisions to be made in real time and communications' optimization.

- Energy Savings: The Q learning model performed worse than the DQN model delivering 25% less energy savings. With effective task distribution between onboard systems and edge devices, energy consumption in computational tasks was minimised to facilitate this optimization.

- Cost Optimization: Intelligent deployment and maintenance equipment scheduling of the DQN model reduced operational and maintenance costs by 35%. The DQN model was able to reduce the overall cost of CBTC system operations by optimally allocating resources and by minimizing the need to perform more frequent maintenance interventions.

Finally, these findings emphasize the ability of DRL and EC for improving CBTC system performance in terms of both efficiency, energy conservation, and cost effectiveness.

6.3.1 Critical Analysis of Findings

Experiments with Deep Q-Network (DQN) and Q-learning based models used to optimize Communication Based Train Control (CBTC) systems illustrate the potential as well as the limitations of applying reinforcement learning to railway operation. After performing continuous state–action mapping, the DQN model exceeded the Q learning baseline, with an average test reward of 44,737.21 against the Q learning model with negative reward of 18,580.33. This demonstrates that traditional reinforcement learning is insufficient to deal with the dynamic and complex CBTC environment. Results of this work agrees with prior studies, e.g., Zhang et al. (2020) and Liu et al. (2021), which highlight the benefits of Deep Reinforcement Learning (DRL) for resource optimization, task offloading, and real-time decision making. The proposed method has optimum performance at the same time the DQN model has a computational overhead while training that the Q-learning did not.

6.3.2 Advantages of the Experimental Design

In terms of experimental design, the evaluation of the models presented there were the result of several positive features. The synthetic dataset was designed very carefully to replicate real world CBTC scenarios, including train speed, position, energy consumption and maintenance schedules. This enabled the models to be tested in controlled conditions, without losing sight of the complex dynamics of a CBTC system. Furthermore, comparisons between the model and the Q learning baseline also proved very effective at demonstrating the strengths of the advanced model using a comparative approach.

6.3.3 Weaknesses and Limitations

A major limitation is using synthetic data. The dataset created to emulate CBTC scenarios that are realistically possible but is constrained in its ability to model the full complexity and variability of real railway systems. System reliance and app offloading might differ from the synthetic data, thus low reliability could be presented in real world data, such as unexpected network disruptions or hardware failures. Additionally, training of the DQN model is quite slow computationally. However, at the scale of a large ship class train system, training deep reinforcement learning models is a high cost computational exercise. This is particularly predictable in an environment that has limited computational resources. It is future research direction to investigate optimization ways of the model, e.g. model pruning or transfer learning, to balance computational burden with performance.

6.3.4 Recommendations for Improvement

Several modifications of the experimental design are suggested to improve the results and to overcome the limitations. Second, the results with real world data would be more valid and generalizable. Live CBTC systems can provide data on how the models operate with real operational data which could reveal some new challenges not present in synthetic environments. The DQN model hyperparameter tuning in another area of improvement. However, this performance was achieved at the expense of hard dependence on fixed hyperparameters that may have prevented the model from being adapted to diverse dynamic

environments. By performing the model across a wider range of CBTC scenarios, applying adaptive hyperparameter optimization algorithms, such as Bayesian optimization could improve the model's ability to fulfil. Moreover, well known multi objective optimization can be introduced to the framework. The current model is an energy consumption and task offloading optimization based model however for reality every aspect of real time railway operations is important like ensuring the safety of train to make as little disruption during operation. The CBTC optimization is a multi-objective problem; hence, it could benefit from a holistic solution approach by optimizing through a multi-objective framework, which would accommodate several factors to evaluate and have an impact on CBTC performance.

7 Conclusion and Future Work

To address optimization issues in CBTC systems, a hybrid framework combining Deep Reinforcement Learning (DRL) and Edge Computing was developed and evaluated. The potential of this framework in task offloading, costs reduction of maintenance, and energy efficiency was very significant, illustrating that it is feasible to perform real time optimization. The objectives of the study were achieved, with the DQN model outperforming the Q-learning baseline in latency reduction by 57%, energy savings of 25% and operational cost reduction of 35%. Using dynamic state action mappings, the DQN model was able to adapt to system condition variations for the DRL and Edge computing based intelligent transportation system, proving the usefulness of DRL and Edge computing in intelligent transportation systems. The results improve the understanding of how AI and Edge Computing can work together to optimize CBTC operation. However, these achievements are limited by the study's reliance on synthetic data, which may not encode the detailed characteristics of real systems. Moreover, the DQN model requires high computational requirements that scale poorly in the resource constrained environments. The research is limited further by lack of validation against real world CBTC systems.

Future work will be to focus on validating the framework with real-time train data to expand its applicability and reliability. We also need to reduce the computational effort of the DQN model, e.g. by model compression or transfer learning, to scale the framework. Decision making across a network of trains and edge devices can be explored in multi agent systems, and prediction of maintenance techniques can help increase system reliability and also reduce cost. Bolstering the model to be applicable for larger networks, along with adding safety critical metrics, will allow the model to be more practical (and scalable) for broader deployment. The same system could be used to optimize existing CBTC systems, improving performance while reducing costs and also support multi agent reinforcement learning requiring coordinated decision making by trains, wayside equipment, and edge servers which can enhance task scheduling and resource allocation in large scale networks.

References

Deep Reinforcement Learning Algorithms for Low Latency Edge Computing Systems (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10134928

Integrating Photonics and Fiber Bragg Grating Sensors with Deep Reinforcement Learning for Advanced Robotic Systems (2024). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10498916

Parallel offloading and resource optimization for Multihop ad hoc Network-Enabled CBTC with mobile edge computing (2024). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10261251

Binary Offloading in Multi-Access Edge Computing Systems: Deep Reinforcement Learning Approach (2024). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10561282

Radio Communication for Communications-Based Train Control (CBTC): A tutorial and survey (2017). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7836324

Performance improvement of Train-to-Train communications by optimizing Next-Hop relay selection in wireless ad hoc networks (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10393638

Modelling and efficient Passenger-Oriented control for urban rail transit networks (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9994628

Mobility-Aware Multihop task offloading for autonomous driving in vehicular edge computing and networks (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9686591

Mobility-Aware Clustering Routing Algorithm for urban rail Transit ad hoc network (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10233763

A resource efficient model training based on uncertainty comparison to detect performance anomalies in 5G networks (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10312549

A Multihop task offloading decision model in MEC-Enabled internet of vehicles (2023). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9684560

Task Offloading based on Deep Reinforcement Learning with LSTM for Mobile Edge Computing (2022). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10055708

Real-time decision making for power systems via imitation learning and reinforcement learning (2022).IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9949821

Joint security and train control design in Blockchain-Empowered CBTC system (2022). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9484083

Collaborative Cloud and Edge Computing in 5G based Train Control Systems (2022). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10001291

A Deep Reinforcement Learning based policy gradient for Energy Consumption in Edge Computing (2022). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10072146

A comprehensive resilient control strategy for CBTC systems through Train-to-Train communications under malicious attacks (2022). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9847010

Research and optimization of urban rail transit handover based on WLAN (2021). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9525657

Future 5G-oriented system for urban rail transit: Opportunities and challenges (2021). IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9354731

Design and performance tests in different test scenarios based on the TD-LTE train-ground communication system (2016).IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7924966