A Scalable Bitcoin Blockchain Network by Dynamic Block Size Adjustment using Reinforcement Learning

by

Sumaiya
20101124
Umme Jannat Taposhi
20101135
Redowanul Akbar
20101253
Md. Zunayedul Islam
22241165

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
November 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Sumaiya
20101124

Umme Jannat Taposhi
20101135

Redowanul Akbar
20101253

Md. Zunayedul Islam
22241165

Approval

The thesis/project titled "A Scalable Bitcoin Blockchain Network by Dynamic Block Size Adjustment Using Reinforcement Learning" submitted by

1. Sumaiya(20101124)
2. Umme Jannat Taposhi(20101135)
3. Redowanul Akbar(20101253)
4. Md. Zunayedul Islam(22241165)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on November 30, 2023.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Shayekh Bin Islam
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The revolutionary technology known as blockchain has revolutionized the way we handle and keep confidential information. In a blockchain network, transactions are organized into blocks that are connected to one another in a sequential and immutable chain. So, the size of the block is a crucial element in the blockchain technology. It influences the number of transactions handled per second, the confirmation time and the general performance of the network. However, the size of each block is fixed to 1 MB in the current technology and we cannot simply alter the block size of the blockchain which causes network congestion and transaction delay. So, we have proposed a model to enhance network performance by dynamically estimating the optimal block size using the Soft Actor-Critic (SAC) algorithm of Reinforcement Learning (RL). This model can predict appropriate block size in real-time by taking account into 4 different features of the network that effects the network performance greatly and resolve the 1 MB fixed block size issues. Soft Actor-Critic model enhances network performance by speeding up transaction confirmation times and making the most use of each block. By using the Soft Actor-Critic model, a scalable improvised network will not only reduce network congestion but it will guarantee consistent profits for the miners. Initially, we constructed a baseline block size prediction model using Long Short-Term Memory (LSTM) networks. However, due to LSTM's limitations in reflecting the dynamic nature of the blockchain, we implemented a better solution using Reinforcement learning. We have applied both online and offline reinforcement learning techniques using a simulator and a real-world dataset. Offline reinforcement Learning was not impressive as well due to the static nature of the data. But using online reinforcement learning gave us the desired results. In online RL, Gym library-based blockchain simulation environment have been utilized for real-time training and assessment. Later we have applied Inverse reinforcement learning (IRL) for reward shaping using Generative Adversarial Imitation Learning (GAIL). Nevertheless, our reward function and Soft Actor-Critic (SAC) model were most efficient to build a scalable improvised network that guaranteed optimal block utilization, minimum confirmation time and miners consistent profits.

Keywords: Reinforcement Learning; Blockchain Network; Bitcoin; LSTM; Inverse Learning; Deep Learning; Block Size, Miner's Revenue, Confirmation Time

Acknowledgement

First and foremost, we give thanks to the Almighty Allah, without whose intervention our thesis could not have been finished.

Second, thanks for your kind support and guidance in our work, Dr. Md. Golam Rabiul Alam, sir. Whenever we needed assistance and advice, he provided it.

We are grateful for your kind support and guidance in our work, Mr. Shayekh Bin Islam, sir. He gave us the guidance and encouragement we needed to finish this thesis.

And lastly, without our parents' unwavering support, it might not be possible. We are about to graduate thanks to their kind support and prayers.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

α Lagrage Multiplier(BEAR Q-learning)

α Temperature parameter(Soft Actor-Critic)

$\hat{E}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))]$ Expectation of the gradient of the logit with respect to discriminator parameters under expert trajectories

$\hat{E}_{\tau_i}[\nabla_w \log(D_w(s,a))]$ Expectation of the gradient of the logit with respect to discriminator parameters under the current policy

λ Controlling parameter(BEAR Q-learning)

λ Hyperparameter balancing policy improvement and entropy maximization(Generative Adversarial Imitation Learning)

λ Learning rate(Soft Actor-Critic)

$\nabla_\theta H(\pi_\theta)$ Gradient of the policy entropy

ϕ Policy parameter

π_ϕ Policy Function

π'_ϕ Target Policy

τ Soft update parameter for the target networks(Soft Actor-Critic)

τ Target network update rate(BEAR Q-learning)

$\tau_E \sim \pi_E$ Trajectories from an expert policy

$\tau_i \sim \pi_{\theta_i}$ Trajectoriees smapled from the current policy

θ_1 Q-funtion parameter

θ_2 Q-funtion parameter

θ_i Policy parameters at iteration i(Generative Adversarial Imitation Learning)

θ_i Q-function parameters for the i-th ensemble member

$\{Q_{\theta'_i}\}_{i=1}^K$ Target Q-functions

a Action

a_t Action Sampled from the policy at time t

BU Block Utilization

C_B Average confirmation time of all transactions

C_b Average confirmation time of all transactions in the block

C_f Average confirmation time of fixed block method

C_m Average confirmation time of different models

D Replay pool

$D_w(s, a)$ Discriminator output for state s and a

m Number of sampled actions for policy update

M_B Average miners' revenue

M_b Miners' revenue in the block

p Number of action samples

$Q(\bar{s}, \bar{a})$ Cost function deriveed from the discriminator's logit

r Reward

$r(s_t, a_t)$ Reward for the transition

R_c Ratio of confirmation time of different models with respect to fixed blocksize method

S State

s' Next state

s_{t+1} Next state sampled from the environment

$TRPO$ Trust Region Policy Optimization

W_c Weight Coefficient of average confirmation time, C

w_i Discriminator parameters at iteration i

W_m Weight coefficient of miners' revenue, M

$y(s, a)$ Temporal difference target for Q-function update

Chapter 1

Introduction

Three game-changing technologies have emerged as leaders in reshaping the technological, financial, and other landscapes in a world driven by innovation and digitization. Blockchain technology, Bitcoin, and Reinforcement Learning (RL) are driving this digital revolution with their distinct and disruptive features. In this introduction paragraph, we looked at the relative importance and combined potential of these technologies to understand how they can change our perceptions of money, trust, and making wise decisions. We will closely examine each of the three concepts and technologies—Bitcoin, blockchain, and RL—to better understand their underlying ideas, uses, and problems. We'll also look at how these technologies work together to create a future where people are more self-reliant, reliable, and able to make quick decisions when using digital devices. Blockchain, reinforcement learning, and bitcoin are a few of the best instances of how innovation can reshape and transform data security, transaction processing, and intelligent system development.

1.1 Research Problem

The most prominent and well-recognized cryptocurrency, Bitcoin, owes its worth and popularity to the ground-breaking blockchain technology that powers it. This technology confers on Bitcoin a number of unique characteristics, like immutability, consensus, trust, and transparency, all of which have combined to drive its market value to historically high levels. The price of a single Bitcoin reached an incredible peak of $59,984.98, demonstrating the enormous demand for this decentralized digital money [24]. Its status as a commonly used medium of exchange in the quickly changing financial landscape is demonstrated by the explosion in its acceptance. Bitcoin's limited and unique supply is a key component supporting its recent price rise. The number of Bitcoins that might ever exist was strictly limited to 21 million by the enigmatic creator of the cryptocurrency, Satoshi Nakamoto. This self-imposed restriction has a significant impact on the dynamics of the price of cryptocurrencies. When the target of 21 million Bitcoins is attained, no more will ever be created, creating a degree of scarcity and uniqueness similar to valuable metals like gold. About every four years, the Bitcoin network undergoes a crucial event known as "halving" which controls this restricted supply. A halving event lowers the rate at which new Bitcoins are created and distributed by half the reward given to miners who successfully add a new block to the blockchain. This is demonstrated by the fact that, as Bitcoin gets closer to its 21 million coin limit, the compensation that

miners received in 2019 for their work was 12.5 Bitcoins (BTC), but this payout has subsequently decreased to 6.25 BTC [25].

This controlled supply mechanism's main goal is to regulate and control Bitcoin's rate of inflation in order to preserve its monetary stability and protect it from the depreciation that unrestricted money printing may cause to traditional currency. But inside the Bitcoin ecosystem, this special feature has raised questions among researchers and miners. There is increasing concern that miners may eventually see decreasing compensation for the significant computational efforts needed to validate and create blocks in the blockchain, as the number of new Bitcoins drops with each halving occurrence.

Miners will depend more and more on the transaction fees they receive for processing and adding transactions to the blockchain when this scenario comes to pass. The main source of revenue for miners is now transaction fees rather than block rewards, which raises complicated concerns about the long-term stability and health of the Bitcoin network. It also starts conversations about how the network's economic model needs to change and adapt in order to keep miners sufficiently motivated and the network running properly.

Therefore, Bitcoin's unique characteristics, such as a restricted supply and the ground-breaking blockchain technology that powers it, are inextricably linked to both its astonishing rise to notoriety and its impressive market value. An essential component of preventing inflation and maintaining Bitcoin's scarcity is the idea of halving, which takes place every four years. This strategy guarantees the cryptocurrency's long-term financial stability, but it also brings up issues and doubts about miner payouts, which can call for constant modifications to the network's economic architecture. With the aim of preserving a fine balance between economic sustainability and the initial concept of a decentralized, trustless digital currency, the world of Bitcoin is always evolving.

The foundation of the Bitcoin design is a basic unit called a "block," each of which has a set size of one megabyte (MB). This design decision has been crucial to the network's functionality, but it has also created a number of issues that must be resolved to guarantee the currency's long-term sustainability.

The problem of slower communication times and increased transaction costs is one of the main issues with the 1 MB block size. By design, it takes 10 minutes on average for a new block to be created on the Bitcoin network. Compared to contemporary banking systems, this duration is comparatively slow, even though it is in line with the network's architecture. Additionally, there's no assurance that a particular transaction will appear in the next block that becomes accessible, even after this significant time commitment. This unpredictability in confirmation timeframes can be especially troublesome for users who want to transact quickly and effectively. In addition, there are usually significant costs involved in accelerating this process through the payment of extra fees, making it a costly option for individuals seeking to expedite their transactions.

The Bitcoin community has realized over time that in order to ensure the cryptocurrency's survival, answers to these enduring problems are required. Numerous strategies have been put up and investigated in an effort to lessen the restrictions imposed by the 1 MB block size. One popular theory proposes that the best way to improve the network's performance is to change the underlying algorithm. A different strategy suggests changing the block size to support additional transactions.
A variety of methods have been proposed by researchers in an effort to address these issues. For example, the researchers investigated the possibility of raising the block size from 1 MB to 8 MB in a noteworthy article in [5]. At first, there was some support for this change, as seen by the 75% of miners who expressed support. This method created a new set of difficulties despite its goal of improving the sluggish transaction confirmation times linked to 1 MB blocks. Block size increases unintentionally caused the propagation delay for new blocks to increase. Thus, the projected gains were offset by lengthier transaction confirmation times as a result of this delay.

Therefore, the 1 MB block size limit on the Bitcoin network has resulted in a number of issues, such as expensive transaction fees and sluggish confirmation times. Although a number of potential fixes, including changing the block size or the algorithm, have been investigated, these fixes frequently come with additional difficulties. As researchers and the larger community try to find a balance between allowing more transactions and preserving the security and integrity of the network, they are always looking for ways to improve the efficiency and scalability of Bitcoin. The long-term viability and utility of Bitcoin as a digital currency and financial system depend on resolving these problems.

While searching for innovative methods to tackle the problems caused by the size of the Bitcoin block, we came upon an interesting strategy called Segregated Witness, or SegWit for short. This methodology, which is described in detail in [27], entails the deliberate removal of signature data from transactions, which decongests the blockchain and increases block size as a result. Notably, digital signatures accounted for almost 65% of the data in a single transaction prior to the adoption of SegWit. SegWit cleverly addressed this problem by separating the transaction from the signature data, allowing more transactions to fit within each block. While this strategy appeared to be a good way to reduce congestion on the Bitcoin network in theory, the results in practice showed otherwise. In comparison to its theoretical promise, SegWit's actual impact turned out to be a complicated issue. Although it was successful in freeing up space within blocks, there were a number of constraints and practical factors that affected how much it improved network performance and transaction throughput.

Another line of inquiry examined the crucial area of Bitcoin security. A study that attempted to strengthen the security of the Bitcoin network by utilizing Machine Learning (ML) techniques was detailed in [26]. This strategy was especially interesting because it gave the network's defenses a fresh perspective. The authors of the research sought to increase the network's resilience against different security threats and weaknesses by incorporating machine learning. The study investigated the effects of applying machine learning to the Bitcoin ecosystem, paying special attention to the tools that miners have at their disposal. Using real-world data in a

prototype testing environment, the researchers evaluated the efficacy of supervised and semi-supervised machine learning models. The supervised machine learning method, logistic regression, fared better than the other approaches examined, according to the data. Interestingly, this improved speed directly affected the mining rate, a critical parameter in the Bitcoin network. Hence, the incorporation of machine learning (ML) into Bitcoin's security architecture is a creative and progressive method of strengthening the network's resistance against possible attacks. It shows how cutting-edge technology has the potential to be extremely important in bolstering the stability and security of decentralized financial systems. Furthermore, this study's results highlight how important it is to select the best machine learning algorithms and techniques in order to get the best outcomes; logistic regression stands out as a particularly effective option.

Researchers discovered the Segregated Witness (SegWit) protocol during their investigation of block size adjustment techniques. The protocol's goal was to expand block size by segregating transactional signature data from the block content. Nevertheless, the real-world effects of SegWit on transaction speed and network congestion turned out to be more complex than expected. In the meantime, research was also conducted on the integration of machine learning techniques to improve the security of Bitcoin. The research in [26], showed that certain machine learning techniques, leading the way in logistic regression, significantly increased the pace of mining and, consequently, the overall security of the Bitcoin network. The aforementioned research initiatives highlight the continuous growth and innovation present in the cryptocurrency space, as professionals look for new approaches to tackle its technological difficulties and maintain its integrity.

Block size on the Bitcoin blockchain has been a topic of a lot of research and investigation. The problem of resolving this specific bottleneck in the Bitcoin network has been the subject of numerous research, and a unifying theme among them has been the investigation of Machine Learning (ML) techniques in search of a fix. Even though these studies have provided insightful information, there is still room for creativity to fully address this issue by giving Bitcoin transactions a dynamically adjustable block size. The idea of a dynamically adjustable block size presents an interesting way to address the problem of maximizing Bitcoin's transaction processing capacity while taking into account a number of other network performance-affecting factors. It is possible to investigate this option by creating a new strategy that makes use of the amount of information gained from earlier research initiatives.

Particularly, this innovative method may use Reinforcement Learning (RL) to address the block size problem in the Bitcoin network. The original goal of reinforcement learning (RL), a machine learning paradigm, was to enable agents to learn, adapt, and generalize their behavior in a variety of settings and activities. In classical reinforcement learning, an agent has the ability to modify its behavior in response to rewards it receives from interacting with a particular environment. By using this fundamental idea, the Bitcoin network could be able to dynamically adjust the block size in response to shifting circumstances and demands.

By integrating reinforcement learning (RL) into the Bitcoin blockchain, the net-

work may be able to make data-driven judgments in real time regarding the ideal block size for a particular set of conditions. This flexibility may help to improve the user experience and overall network performance by reducing problems with congestion, transaction delays, and fee variations. In terms of conclusion, a number of studies have pointed to the use of ML algorithms as a potential solution to the block size issue in the Bitcoin blockchain. That being said, there is room for creativity if a dynamically changeable block size is developed while keeping in mind the insights from earlier studies. A viable solution to enable the Bitcoin network to dynamically adjust and optimize block size in response to shifting needs and conditions in real time is reinforcement learning. This strategy might be a big step toward improving the Bitcoin blockchain's efficiency and scalability.

An RL agent must be trained to make judgments in order to create a scalable bitcoin blockchain network and use RL to change the block size. Many factors need to be considered, such as fee levels, transaction backlogs, network congestion, and mining incentives. Using all of these historical data together with network feedback, the RL agent would be trained to understand how to dynamically adjust the block size.

Our work at hand is well-positioned to provide a thorough answer to the query of how to handle the difficulties presented by this particular block size issue. We want to use a multidisciplinary approach that encompasses a range of tactics and areas of expertise in our search for a solution. This comprehensive strategy ensures that we investigate the whole range of viable options and offer a well-rounded answer by addressing the issue from several perspectives. Long Short-Term Memory (LSTM), a neural network architecture that will act as our foundational model, is where we will begin this adventure. Long Short-Term Memory (LSTM) is a powerful and popular deep learning technology that is well-known for its ability to analyze sequential data. We seek to create a fundamental knowledge of the dynamics and complexities of the situation by using LSTM as our baseline model.

Moving beyond, we will discuss offline reinforcement learning, which is a promising technique to solving complicated issues, in more detail as we get beyond the baseline model. Our goal in this domain is to use multiple reinforcement learning (RL) algorithms, including Behavior Regularized Actor-Critic (BEAR), Batch-Constrained Q-Learning (BCQ), and Conservative Q-Learning (CQL). Applying these several RL algorithms demonstrates our dedication to investigating a variety of approaches, each with its own advantages and potential to aid in solving the challenge. To enhance its overall performance and maximize its decision-making abilities, we will also be able to train the RL agent in an offline environment.

As we go along, we'll use the OpenAI Gym environment and simulation to further our investigation into online reinforcement learning. This method adds a dynamic, real-time element to our approach to problem-solving. We can test our solution's capacity to adapt to changing situations by experimenting with various scenarios and integrating it with the OpenAI Gym environment. This project component is essential to guaranteeing the practicality of our methodology since it emulates the changing character of real-world problems. The development and application of

well-designed reward functions are essential to our method's efficacy. The behavior of our reinforcement learning agent is significantly shaped by these reward functions, which also direct the agent toward making decisions that result in desired outcomes. In order to construct reward functions that accurately reflect the subtleties of the given challenge, domain-specific knowledge and expertise will be consulted. We can adjust the RL agent's learning process to better match the problem's objectives through meticulous design and calibration.

As a result, our work is not just a one-dimensional attempt to solve a complex problem; rather, it is a multidimensional exploration that combines the precision of well-constructed reward functions, the power of LSTM, the adaptability of online reinforcement learning, and the versatility of various RL algorithms. This all-encompassing strategy demonstrates our dedication to fully comprehending and tackling the issue from a variety of angles. We think that this multi pronged approach will provide a strong and impactful response that might potentially have a significant effect in the area it addresses.

1.2 Research Question

The primary focus of this research is to provide an answer to the following question:

Can we use Reinforcement Learning (RL) to dynamically adjust the block size of Bitcoin blockchain and will this adjustment be effective, efficient and sustainable compared to the already established scaling solutions and conventional fixed block size mechanisms?

This study will provide a response to the aforementioned question by employing various strategies of different domains starting with LSTM as baseline model, offline reinforcement learning using various RL algorithms (CQL, BCQ, BEAR) and training the RL agent, online reinforcement learning using Open AI Gym environment and simulation, using inverse RL properties and utilizing appropriately constructed reward functions.

1.3 Research Gaps

In this section we are going to discuss about our findings which are directly related to our area of interest. We have searched numerous times for research in this field, yet we were unable to find a novel solution which covers all the grounds in terms of solving the issue created due to fixed block size. We found papers where the issues are resolved but none of those show absolute results and have gaps where they can be improved.

In [35], they used a machine learning model to predict the optimal block size for each block creation cycle which significantly improves the throughput by reducing the average transaction confirmation time without compromising security. But it falls short since it needs to be trained on historical data which affects the network's ability to adapt to sudden changes, such as an increase in transactions. On the other

hand, in [34], they have utilized a reinforcement learning approach to establish an optimal policy to decide whether to create a new block upon given scenario or not. But the RPPO algorithm that they used depends on historical data as well and is computationally very complex. It also lacks empirical evaluation on real-world data which is essential in practice.

1.4 Research Objective

This research intends to create a scalable Bitcoin blockchain network through dynamic block size adjustments. Reinforcement Learning (RL) is the proposed primary factor that will drive this research. LSTM is the baseline and the offline, online RL models will take information from the existing and simulated environment respectively and train their agents accordingly. Throughout the process the agents gained and trained knowledge, whenever a new block creation begins the model will generate an optimal block size.
The objectives of this research are:

1. To develop a RL model using different RL algorithms and predict optimal block size.

2. To compare the effectiveness of the suggested dynamic block size optimization strategy with established scaling solutions and conventional fixed block size Mechanisms.

3. To ensure adaptability in the block size to achieve higher transaction throughput, maximize the block utilization and increase the miners' profit as much as possible.

1.5 Research Contributions

This work represents a significant advancement in the field of blockchain scalability in many ways, particularly when considering the Bitcoin network. The study focuses more on practical implications than theoretical advancements in order to guarantee adaptability during peak transaction hours and ultimately improve overall network performance. By achieving these goals, the research has a greater overall impact on the Bitcoin ecosystem and offers a novel framework for building a blockchain that is more durable, scalable, and efficient for the cryptocurrency.
We have proposed a model to enhance network performance by dynamically estimating the optimal block size using the Soft Actor-Critic (SAC) algorithm of Online Reinforcement Learning (RL). This model can predict appropriate block size for each transaction and resolve the 1 MB fixed block size issue. Soft Actor-Critic model enhances network performance by speeding up transaction confirmation times, predicts optimal block size for each transaction and makes the most use of each block. By using the Soft Actor-Critic model, a scalable improvised network will not only reduce network congestion but also it will guarantee consistent profits for the miners.

The contributions of this research are:

1. This research's primary contribution is scaling bitcoin blockchain network and enhancing network's performance by dynamically estimating the optimal block size which is the Soft Actor-Critic (SAC) algorithm of Online Reinforcement Learning (RL).

2. The proposed model in this research minimizes the average confirmation time of transactions by estimating optimal block size.

3. It reduces space wastage of the block and maximizes the utilization of the produced block through its optimal block size prediction.

4. Our designed reward function stands alone and proves to be more efficient than inverse reinforcement learn (IRL) imposed rewards.

5. Our suggested solution addresses Bitcoin's scalability issue during peak hours and has the capacity to produce blocks quickly when needed.

6. We additionally proposed a blockchain simulator where we may train any kind of reinforcement learning algorithm in order to carry out other research.

1.6 Thesis Orientation

The arrangement of the paper is as follows.

- In Chapter 1 the introduction , the topics of motivation, problem statement, objectives and contribution are covered.

- In Chapter 2, relevant papers pertaining to blockchain optimisation are discussed

- In Chapter 3,We've discussed the system's background research. We also talked about several terms and the blockchain's existing design.

- In Chapter 4, we have discussed our research approach along with a vivid diagram depicting how the whole system works. Also we have discussed all the methodologies such as LSTM, offline RL, online RL, the simulator and simulation process, Gym environment in detail.

- In Chapter 5, we have done our result analysis and have done a comparative analysis of all the results at our hand. Where we have found our best Soft Actor-Critic model, which is more efficient than any other model.

- In chapter 6, we composed a brief conclusion and discussed our future plan of action for further optimization of blockchain.

Chapter 2

Related Work

This chapter focuses on gathering more knowledge on finding a better or a novel way for creating a scalable bitcoin blockchain network through dynamic block size adjustment by reviewing multiple journal articles and papers. Literature of related works have been discussed broadly in the following sections.

Initially, we learned about the fundamentals of blockchain technology, how it works at present and its applications in the real world. The paper [18] compared various consensus mechanisms, explained the taxonomy and architecture of blockchain, and discussed challenges like scalability, privacy, interoperability, energy consumption, and regulatory issues. It also presented a comparative study of the tradeoffs of blockchain and discusses the potential applications of blockchain technology in the future. Furthermore, a thorough analysis of several blockchain technology applications in various fields and businesses have been provided. According to [18], blockchain is being used in industries including banking, supply chain management, healthcare, energy, and more. The writers offered insightful explanations of the use of blockchain technology and its possible advantages for various fields. Moreover, it dived into the detailed transaction process of Bitcoin using blockchain technology. By reviewing this process we get the idea of how a block is created by the miners and also about the block structure. In [18], the small capacity of the block size and a major scalability issue in the current technology was highlighted.

The biggest and most successful implementation of this technology is Bitcoin [18]. In [3], the fundamentals of bitcoin have been discussed. It offered a thorough and instructive introduction of the core ideas and features of bitcoin. The author provided insightful explanations of how Bitcoin functions, its underlying technology, and its economic ramifications. The article provides a straightforward and succinct description of Bitcoin, for readers looking for a basic comprehension of this digital money. The author of [3] mentioned the decentralized nature of Bitcoin, cryptographic concepts, blockchain technology, and the function of miners in preserving the integrity of the network. The explanation of Bitcoin's economic ramifications and monetary properties benefitted from the author's experience as an economist.

Even though Bitcoin has its positive side, it is not without some restrictions. Bitcoin implements a fixed block size within its blockchain. Although it makes transactions very safe, scalability becomes the primary issue with the Bitcoin blockchain's set

block size which restricts its throughput. The greatest amount of data that may be included in a single block of transactions is referred to as the block size. A 1 MB block size cap was first imposed on Bitcoin transactions in order to maintain network security and stability [3]. The block size restriction, however, created a barrier for transaction processing as Bitcoin's acceptance and use grew, creating a number of problems such as slow times, limited capacity at transaction etc.

Focusing on the concern of fixed blocksize mentioned previously, further research has been done to find if there have been researches proposing dynamic block size to solve the scalability issue. The study [7] discussed the block size of the blockchain network and introduced a method to dynamically adjust that. A novel permissioned blockchain framework has been introduced by [7] which is adjusted to ensure security, higher throughput, lower latency. They first highlighted the limitations of the current applied methods of transaction through the blockchain. The current applied methods are secure but as the block size remains fixed, it is very slow, which increases latency and decreases throughput. As large blocks take longer to propagate, using the PBF(permissioned blockchain framework) of [7], the block sizes are decreased to a few KB which can increase the number of transactions by a lot. The paper also described Peer Inner Blockchain protocol(PIBP), which is used to validate these micro-blocks. Permissioned Trusted Trading Network Consensus(PTTNC) has been developed as a part of the framework [7]. The dynamic block size mechanism introduced by the authors offered a viable way to expedite transaction processing without compromising security of the algorithm. The authors then implemented their proposed algorithm in an experimental setup, where the results are compared with bitcoin on the basis of throughput of transactions, block frequency and block size where it is shown that the framework qualitatively outperforms Bitcoin. Finally, [7] gave us a brief overview of how to work in the domain of blockchain, how to define the states, what values to evaluate when we need to compare with the existing systems etc.

Up until now, there has not been much work done in the domain of blockchain with re-adjustable and autoscaling block sizes. One of the major issues for this is network stability. A consensus mechanism such as in the Bitcoin transactions process, is the foundation upon which blockchain networks function. The block size restriction is established to guarantee that blocks may be propagated and validated by network members in a timely manner. Because a dynamic block size allows for variations in the block size, it makes it more difficult to forecast the behavior of the network and its resource needs. Also, dynamically changing block sizes pose the risk of blockchain splits. If different nodes in the network have different block size constraints, the blockchain may split as a result of disagreements over the veracity of blocks and transactions. Users could get perplexed as a result, endangering the security and consensus of the network. Moreover, A dynamic block size approach needs to be properly planned and take a variety of factors into consideration, such as network congestion, transaction fees, and resource availability. It includes creating a consensus mechanism that might dynamically change the block size cap depending on the state of the network. In order to ensure that a system is stable, secure, and fair, it must go through a rigorous testing and analysis process.

This research will primarily examine how to dynamically increase or decrease block size on the Bitcoin network. Using predictive algorithms to do this would not be fruitful as the network parameters are ever changing. So by using meta reinforcement learning, we will try to train the model to adapt to the real time network parameters and predict the optimized block size more accurately. By exploiting the capabilities of meta reinforcement learning, we seek to create a flexible and intelligent method for dynamically altering the block size in response to current network conditions and evaluate the effects of such modifications on scalability, transaction throughput, and network performance. Finally, compare the proposed method with established scaling mechanisms and typical fixed block size methods.

Now that we obtained an objective, we searched for any research which provided a clear understanding of meta reinforcement learning to predict auto scaling in any domain field. [12] offered a substantial advancement in the area of meta-reinforcement learning. This work's main contribution is an effective meta reinforcement learning strategy for online adaptability in changing contexts. The algorithm employed here effectively develops a global model capable of swiftly adapting in dynamic contexts by drawing on previous experiences [12]. In this method, a dynamics model prior is trained via meta-learning so that it can quickly adjust to the local environment when used with current data. The writers then demonstrated their experimental work. These tests show that continuous control tasks may be performed online on both simulated and actual agents. The authors of [12] talked about two versions of their framework, recurrence-based adaptive learner(ReBAL) and gradient-based adaptive learner(GrBAL). While describing their models, they claimed although the employed model signifies the model predictive path integral control (MPPI), their model adaptation strategy is, in theory, independent of the model predictive control (MPC). By avoiding mistakes from building up over time, the usage of MPC corrects for model imperfections. This is because of replan at each time step using the most recent state data. The authors then showed their experimental results through which they showed that, when trained with comparable quantities of data, their model-based meta RL technique significantly outperforms previous methods such as normal model-based methods, online model-adaptive methods, model-free methods, and prior meta-reinforcement learning methods. As our selected domain field is also dynamic, this particular paper contributes to our knowledge by demonstrating a meta- reinforcement learning algorithm in a dynamic domain field [12].

To implement meta reinforcement learning in the domain field of blockchain technology, we have to build a framework that works with the existing blockchain system and as the acting algorithm can implement meta reinforcement learning. So after getting the prior knowledge about Meta Reinforcement learning and Blockchain technology, we need to identify potential integration points, design an architecture, create an incentive system that rewards and penalizes the agent and then uses simulated scenarios to evaluate its effectiveness.

In [32], the authors suggest a meta reinforcement learning strategy that makes use of predictive analytics and historical data to enhance autoscaling judgments. The difficulty of autoscaling in cloud computing settings is discussed in this research article. Reinforcement Learning (RL) has been presented as a potential method in

recent works for learning resource management rules to direct scaling activities in a dynamic and unpredictable cloud environment. However, RL approaches face the following difficulties when directing predictive autoscaling: poor decision-making precision, ineffective sampling, and a wide range of workload patterns that might lead to policies failing at test time. In order to achieve this, the authors proposes an end-to-end predictive meta model-based RL algorithm, incorporating a specially designed deep periodic workload prediction model as the input and embedding the Neural Process to direct the learning of the optimal scaling actions over numerous application services in the Cloud, with the goal of optimally allocating resource to maintain a stable CPU utilization level [32]. Their approach significantly outperformed the previous algorithms, and it has been implemented online at Alipay to facilitate the autoscaling of apps for the top payment platform in the world. For the multi-dimensional, multi-horizon workload prediction, they created a deep attentive periodic model, which offers accurate and trustworthy workload data for scaling. Then, a meta-learning model was used to train a dynamic prior of the workload to CPU usage map, with quick environment adaption incorporated to direct the learning of the best scaling actions across hundreds of online apps. The meta model-based RL method makes learning secure and data-efficient. Their solution has been used in the real world to support the scalability in the Cloud of a well-known mobile payment platform, with resource savings of almost 50% as compared to the rule-based method used in production.[32]. It successfully stabilizes the way that programs use the CPU.

These are some of the literature we reviewed to gain the knowledge and get the initial idea of what our domain is, what our topic is and how we merge and get the best of both worlds. Throughout the next few months, we will have to implement these ideas and find results to answer the question whether we can create a framework using meta reinforcement learning which will predict the dynamic block size and if it will be effective and sustainable or not. After finding out the initial idea of the thesis we move on to learning about a few of the models which are being currently used in research involving these two fields. A few meta reinforcement learning methods have been reviewed next to get an idea of which models will work more in favor of our purpose.

The proposed system in [29] is DMRO(Deep Meta Reinforcement learning-based Offloading) which shows that it is feasible to swiftly and flexibly get the ideal offloading strategy from a dynamic environment by combining the perceptive capacity of meta learning, the capability of quick environment learning of deep learning, the decision-making capacity of reinforcement learning. The authors build an edge-cloud offloading architecture that allows IoT devices to select whether to move their computational activities to edge servers or cloud servers. In order to reduce latency and energy consumption, edge servers decide which tasks to offload for each device based on task information [29]. By making this problem a multiobjective optimization problem, it minimizes delay. Making these judgments on offloading requires combining several parallel Deep Neural Networks (DNNs) with deep Q-learning algorithms [29].

In another research [13], an exciting method for policy search in reinforcement learn-

ing (RL) has been proposed that blends deep neuroevolutionary algorithms (EA) with gradient-based techniques of deep Reinforcement learning. The authors first described the already existing methods such as the Evolutionary algorithms, Estimation of distribution algorithm(EDA) etc. Then they went on discussing the cross-entropy methods, the deep deterministic Policy Gradient(DDPG) and twin delayed deep deterministic Policy gradient(TD3) which are the parts of the deep reinforcement learning models. Next the methods on which the CEMRL will operate have been discussed. The study investigates the interaction between RL and evolutionary algorithms. The authors demonstrate how combining evolutionary techniques with RL algorithms may enhance convergence and enhance exploration in policy search problems. The objective of the authors are to find out how the performance of CEMRL will be when compared to CEM only, or TD3. Also, it is compared with ERL as well.

MOEAD algorithm, a multiobjective evolutionary algorithm that uses decomposition techniques to address issues involving several objectives in optimization has been proposed in [1]. The main ideas of the algorithm are presented, along with certain advantages and a thorough evaluation of how well it performs in comparison to other cutting-edge methods [1]. The MOEAD method, which uses decomposition techniques to handle the difficulties of multiobjective optimization, is introduced in this study. The multiobjective problem is divided into a number of scalar subproblems by MOEAD, which solves them all simultaneously [1]. This decomposition-based strategy enables the objectives to be optimized while keeping convergence and diversity under check, resulting in more practical and useful solutions. With the help of their work in the article, decomposition strategies—which have long been researched in the world of mathematical programming—can be easily included into EAs for multiobjective optimization. In the MOEAD framework, new advancements in decomposition methods and other scalar optimization approaches may be easily combined with EAs [1].

Due to its simplicity, efficiency, and capacity for handling complicated issues, Proximal Policy Optimization (PPO) algorithms have become a wellliked strategy in reinforcement learning (RL). By utilizing proximal optimization approaches, PPO algorithms seek to overcome the difficulties associated with policy optimization in RL. [10] offers a thorough examination of PPO algorithms, highlighting the salient characteristics, benefits, and applications of each. It describes that the surrogate goal function used by PPO algorithms promotes minor policy modifications. PPO improves stability during optimization and avoids disastrous policy updates by reducing the divergence between the new and old policies. A trust area is established by PPO algorithms to regulate the size of policy changes. During each optimization phase, this area limits the maximum change in the policy, preventing policy collapse or overly large updates [10]. The PPO uses a proxy objective function to approximate the performance of the policy and direct policy modifications. By calculating the advantage of activities based on their value functions, the surrogate goal strikes a balance between exploration and exploitation. Furthermore, PPO modifies policies repeatedly utilizing small chunks of accumulated knowledge. With this method, parallel computing resources may be used effectively while also improving the efficiency of the samples [10].

The challenge of discovering control rules for continuous action spaces as opposed to discrete action spaces is referred to as continuous control. Concerns of continuous control have been effectively addressed with Deep Reinforcement Learning (DRL), which has been used to a variety of difficult challenges. In order to help agents learn control strategies in continuous action spaces, the authors of [4] propose a system that combines deep neural networks and reinforcement learning. The authors present the DDPG method, which combines policy gradients with Q-learning. Both the actor (policy) and critic (action-value function) components are represented by deep neural networks. The agent may learn directly from unprocessed observations because the actor network learns a deterministic policy while the critic network estimates the action-value function [4]. The efficiency of the DDPG algorithm on a variety of continuous control tasks from the OpenAI Gym environment is evaluated by the authors through numerous trials. They demonstrate the advantages of DDPG for learning high-performance control rules by contrasting it with a number of benchmark methodologies. The findings demonstrate that DDPG outperforms rival techniques and provides cutting-edge performance across a variety of workloads.

The authors of [21] first discuss how it is a challenge to leverage large pre-collected datasets in reinforcement learning. Although offline RL algorithms have the potential to learn excellent rules without relying on previously collected static datasets, they present a formidable challenge in practise, as traditional off-policy RL methods can falter due to overestimation of values caused by distributional differences between the dataset and the learned policy, especially when training on intricate and diverse data distributions. To overcome these limitations, Conservative Q-Learning (CQL) was introduced that trains on a conservative Q-function [21]. The Q-function is meant to guarantee that it does not overestimate the genuine value and hence provide a lower constraint on the predicted value of a policy. In real-world applications, Conservative Q-Learning improves the standard Bellman error objective with a simple Q-value regularizer in addition to the actor-critic and deep Q-learning implementations. In previous studies, CQL consistently outperforms earlier offline RL methods in the discrete and continuous control domains. Especially when dealing with complex and multimodal data distributions, since CQL frequently permits the learning of policies that eventually yield returns that are 2–5 times greater [21], it caught an august regard for our study.

According to the authors of [17] Off-policy reinforcement learning seeks to use the information learned from previous policies in an efficient manner. Actor-critic methods and Q-learning, two often used off-policy techniques, are extremely sensitive to the distribution of the data in practice. They typically struggle to advance much without gathering more on-policy data. To deal with such scenarios where off-policy experience is fixed and further interaction with the environment is restricted, Bootstrapping Error Accumulation Reduction (BEAR) was introduced [17] which effectively learns from a range of off-policy distributions, including random and suboptimal examples across an array of continuous control tasks. BEAR demonstrated success in diverse continuous-control tasks using a variety of off-policy datasets. These datasets contain scenarios generated by random, suboptimal, or excellent policies. BEAR consistently outperforms state-of-the-art techniques across several

training datasets, whereas earlier algorithms flourish only under certain dataset conditions [17].

The authors in [15] discuss conventional off-policy deep reinforcement learning algorithms like Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG) find it difficult to train efficiently in the fixed batch scenario because of their extrapolation errors. These errors make these operations pointless when the given data does not match the distribution of the current policies. Batch-Constrained deep Q-learning (BCQ) [15] may effectively learn from any fixed batch of data when performing continuous control tasks. The method lowers extrapolation errors in off-policy learning by using a state-conditioned generative model to construct actions that have been seen in the batch data. Furthermore, using a Q-network, the highest valued action that matches the data in the batch is selected from among these newly formed actions. The key finding is that, when working with incomplete datasets in finite deterministic Markov Decision Processes (MDPs), this batch-constrained technique is crucial for effective value estimation[15].

The goal of the [35] research's optimization problem is to increase the number of transactions throughout each cycle. A thorough learning framework is presented in recognition of the problem's intrinsic complexity and unsolvability. The approach includes training models with better generalization capabilities, predicting the ideal block sizes for each block generation cycle, and developing a data-driven infrastructure based on machine learning (ML). Extreme Gradient Boosting (XGB) is the algorithmic cornerstone that forms the basis of this suggested architecture.To help the blockchain create an adjustable block size, the learning framework examines nine characteristics related to the Bitcoin network. Using real-world data from the past four years as its training dataset, XGB shows a 61.21% predictive accuracy in identifying the ideal block sizes. When the approach is implemented, the Bitcoin ecosystem benefits greatly, as seen by a 9.3% increase in miners' earnings and a noteworthy 66.75% increase in transaction throughput. This game-changing result tackles the lengthy wait times and promotes a wider acceptance of Bitcoin in the digital banking space.

The author of [34] explores a novel scheme called ROBB (Reinforcement learning for Optimizing Blockchain Blocks), which addresses the major problems associated with block creation in blockchain networks by utilizing the capabilities of Reinforcement Learning and, in particular, the Recurrent Proximal Policy Optimization (RPPO) algorithm. In order to comprehend the significance of streamlining the block creation process, the paper first emphasizes the importance of blockchain technology. It presents ROBB as a novel solution and highlights how it can improve the security and efficiency of blockchain networks. The policy network and the value network are the two neural networks that make up ROBB. The policy network generates a probability distribution over actions based on the state of the environment as input and is implemented as a deep recurrent neural network. Finding a policy that improves the projected cumulative reward over time is its main objective. The training procedure is described in detail in the text, with a focus on the use of a surrogate objective function and the iterative nature of policy optimization. Another important component is the value network, which calculates the expected cumulative

benefit or advantage of being in a certain state and adhering to the current policy. This deep recurrent neural network offers feedback on the merits of various states and actions, which helps with the policy optimization process. A solid foundation for understanding ROBB's functionality is provided by the explanation of how these networks cooperate. In order to achieve a balance between policy improvement and divergence, the paper continues discussing policy updating and introduces the modified surrogate objective function used in RPPO. In order to measure the change from previous to current policies and regulate the size of policy updates, the clip and ratio functions are essential. The updating process's complexities are made clearer by the inclusion of equations and explanations.The value network is discussed in detail, explaining how it estimates the value or advantage function and directs the policy optimization procedure. The detailed computation of the loss function, which involves mean squared error, illustrates how the value network is trained to increase the accuracy of its estimations over time. One notable example of a novel reinforcement learning model intended to maximize blockchain block creation is ROBB. Its potential to address the issues related to block size and transaction processing in blockchain networks is enhanced by its integration of RPPO, use of two neural networks, and rigorous training procedure. With blockchain technology still developing, ROBB offers a promising first step toward more secure and productive blockchain networks.

Chapter 3

Background Study

In this section of the paper, we will discuss about the necessary information needed beforehand to complete this work. We will talk about the core concepts of the entire blockchain network, Bitcoin and how the transaction of bitcoin works and finally a generalized overview of reinforcement learning with a in-depth discussion about the different variations of reinforcement learning.

3.1 Blockchain Network

The operational diagram illustrating the procedure of a bitcoin transaction through a blockchain is shown in Figure 3.1.

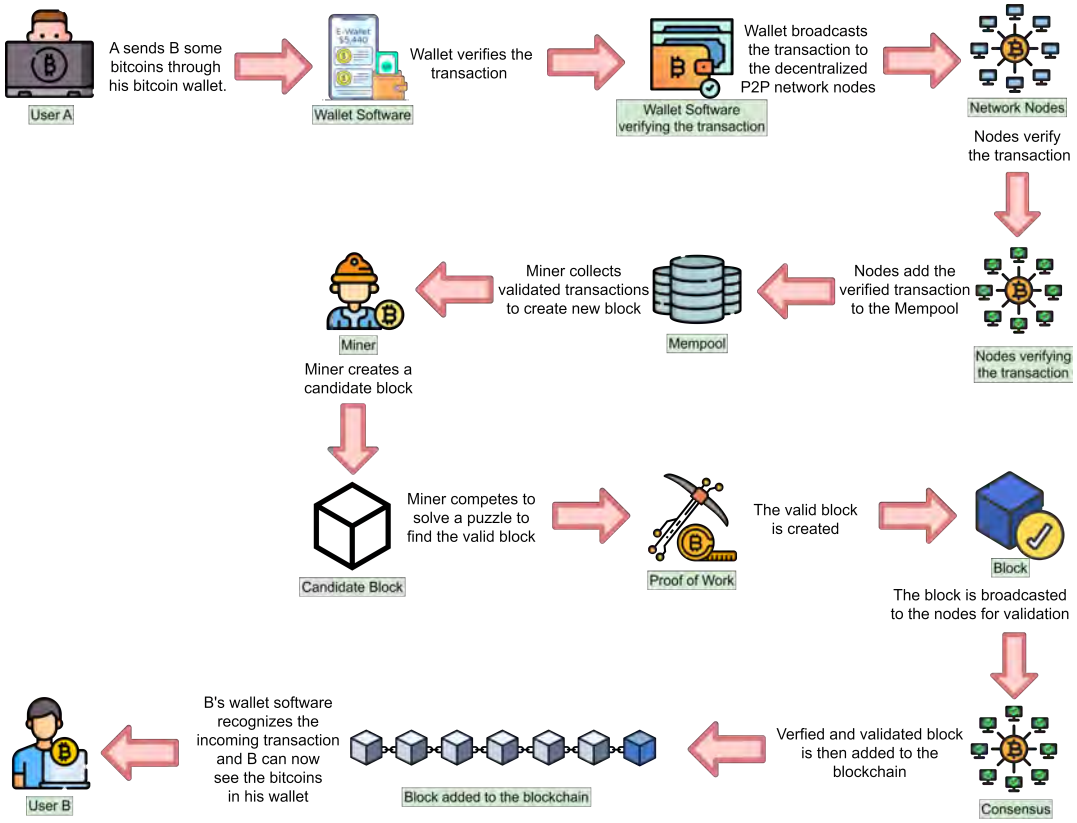

Figure 3.1: Operational diagram of Bitcoin transaction through blockchain network

Though the majority of its current applications are limited to tracking and storing transactions for cryptocurrencies like Bitcoin, Blockchain for processing payments and transferring money, enthusiasts for blockchain technology are creating and testing new applications for it. Blockchain-enabled transactions have the potential to settle in a matter of seconds and minimize, if not completely eliminate, bank transfer fees. Businesses could use blockchain technology to track goods in real time, identify inefficiencies in their supply chains, and monitor product quality control as goods are transferred from manufacturers to retailers. Microsoft is testing blockchain technology to give users control over who can access their data and create digital personas. Blockchain can act as a go-between for the secure exchange and archiving of corporate data across industries. A decentralized database that guarantees musicians retain their rights to their music and gives them transparent, real-time royalties could be made possible with blockchain technology. Blockchain and open source developers may be similar in this regard. Blockchain can function as an IoT network regulator by identifying and authenticating linked devices, keeping an eye on their activities, and evaluating their reliability. It can also be used to automatically evaluate the reliability of newly connected cars and smartphones. There are numerous possible uses for blockchain in the medical sector. Blockchain is being used by healthcare payers and providers to manage clinical trial and electronic medical record data while adhering to regulatory requirements.

The way transaction data is kept in blocks that are connected to form a chain gives blockchain its name. The blockchain expands proportionately with the volume of transactions. Within a distinct network with rules selected by its members, blocks record and validate the order and timing of transactions. After that, the blockchain is updated with these blocks.

The following are the four main ideas of blockchain:
Shared ledger: A distributed "append-only" system of record that is shared throughout the network of an organization is called a shared ledger. A transaction is only recorded once in a shared ledger, which eliminates effort duplication that is common in traditional business networks.

Permissions: Transaction security, authenticity, and verifiability are guaranteed by permissions. Organizations can more easily comply with data protection regulations, such as those outlined in the EU General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA), by having the ability to restrict network participation.

Smart Contracts: A smart contract is a set of rules or a contract that controls a business transaction. It is executed automatically as part of a transaction, is stored on the blockchain, and doesn't require human intervention.

Consensus: All parties agree to approve the network-verified transaction by consensus. Consensus techniques used by blockchains include multisignature, proof of stake, and PBFT (practical Byzantine fault tolerance).

These are just a few of the roles that different participants in each blockchain net-

work play:

Blockchain Users: individuals who are permitted to sign up for the blockchain network and conduct transactions with other users, typically businesses.

Regulators: Blockchain users who possess specific authorization to monitor network transactions.

Blockchain Network Operators: People with the specific power and authorization to design, develop, oversee, and maintain the blockchain network.

Authorized Certificators: People in charge of issuing and overseeing the various certificate kinds needed to operate a permissioned blockchain.

The private and public blockchains, which are the two main varieties, provide varying degrees of security. Public blockchains validate transactions and group them into blocks to be added to the ledger using PCs with public internet access. Nonetheless, only reliable companies are typically permitted to join private blockchains. Because any organization can join a public blockchain, it might not be the best choice for businesses that are concerned about the privacy of the data moving through the network. Both public and private blockchains have different participant identities. Anonymity is a fundamental component of public blockchain architecture. Verified users validate transactions on a private blockchain within a permissioned network. To come to an agreement, a procedure called "selective endorsement" is employed. Because only those with the proper access and authorization can maintain the transaction ledger, businesses stand to gain from this. While there are still some issues with this strategy, like insider threats, a highly secure infrastructure can solve a lot of them.

Blockchains are usually operated as public distributed ledgers on peer-to-peer (P2P) computer networks, where nodes follow an algorithm for consensus protocol to add as well as confirm new blocks of transactions. Because they permit forks and record modifications, blockchains can be regarded as secure by design and serve as an illustration of a system of distributed computers that has Byzantine fault tolerance. There have been suggested private blockchain applications for businesses. On the other hand, some have countered that because permissioned blockchains are more decentralized when properly designed, they might actually be safer in use than permissionless ones. Computerworld referred to the commercialized blockchains that were being offered as "snake oil" due to their lack of a suitable security model. A form of data storage called blockchain makes it difficult or impossible for others to access, alter, or take over the system. A distributed ledger, also known as a blockchain, is a system of interconnected computers that duplicates and distributes transactions among themselves.

In essence, a blockchain is a distributed, immutable, decentralized ledger made up of multiple blocks, each of which holds a different set of data. Cryptographic techniques are used to join the blocks into a chronological data chain. The consensus mechanism of a blockchain, which consists of a network of nodes that concur on

the validity of a transaction before adding it to the blockchain, incorporates data security.

Blocks: In a blockchain, a block consists of three primary parts:

1. The header includes information used in the mining process, like the previous block's hash value and a timestamp with a random number.

2. The real and primary data that is kept in the block, including transactions and smart contracts, is contained in the data section.

3. Finally, for verification purposes, the hash is a distinct cryptographic value that serves as a representation of the complete block.

Block Time: In a blockchain, block time is the amount of time it takes to create a new block. The block times of various blockchains can differ, ranging from a few seconds to minutes or even hours. Longer block times may extend the time it takes for transaction confirmations while decreasing the likelihood of conflicts. While shorter block times may result in faster transaction confirmations, the likelihood of conflicts is higher.

Hard Forks: A permanent split in a blockchain's history that creates two distinct chains is referred to as a "hard fork." It may occur when there is a fundamental modification to a blockchain's protocol and not all nodes concur on the update. Hard forks can split up already-existing cryptocurrencies or create new ones, and they must be resolved by network users coming to an agreement.

Decentralization: Decentralization is the core feature of blockchain technology. A single central authority cannot control the network thanks to a decentralized blockchain. Decentralization is the process of distributing decision-making power among a network of nodes that collectively approve and verify transactions before they are added to the blockchain. The advancement of security, trust, and transparency is facilitated by the decentralized structure of blockchain technology. It also reduces the possibility of relying on a single point of failure and data manipulation.

Finality: The irreversible confirmation of a transaction in a blockchain is known as finality. When a transaction is part of a block and the block is approved by the network, it becomes final and irreversible. This feature ensures the integrity of the data, discourages double spending, and provides a high level of security. Blockchain Types and Sustainability

Openness: The blockchain is accessible to anyone who wants to join the network because blockchain technology is open. This implies that anyone may join the system, validate operations, and add additional blocks to the blockchain as long as they're aware about the consensus rules. Openness encourages diversity, transparency, and innovation because it allows for stakeholder participation.

Blockchain integrates three innovative technologies:

1. Cryptography keys

2. A peer-to-peer network in a distributed ledger

3. A computing technique that keeps track of the documents and activities on the network.

There are two kinds of keys in cryptography: **public and private** [19]. These keys enable two-party transactions to be completed successfully. The purpose of these two unique keys is to establish a secure point of reference for digital identities. This secure identity is perhaps the most important aspect of Blockchain technology. In the context of cryptocurrencies, this identity is referred to as a "digital signature" and is used to approve and track transactions.

3.2 Bitcoin

Bitcoin is a virtual currency, also known as a cryptocurrency, that can be used to pay for goods and services from vendors who accept Bitcoin as payment. Bitcoin owners can trade, purchase, and sell goods and services directly to other users without the use of a bank or other intermediary. Since its launch in 2009, the value of Bitcoin, one of the most well-known virtual currencies available today, has increased significantly. The man who invented Bitcoin, only known by the pseudonym Satoshi Nakamoto, asserted that the platform's objective is to function as an electronic payment system based on cryptographic evidence rather than trust. Some bitcoin owners buy it as an investment with the expectation that its value will increase, while others use it as currency or accept payments in bitcoin. The digital exchange of highly encrypted, anonymous hash codes between bitcoins is accomplished through peer-to-peer (P2P) networks. Users' bitcoin transfers are tracked and verified by the peer-to-peer (P2P) network. The software that stores each user's bitcoin is called a digital wallet, and it also has a private key that is only known to the user and all of the addresses that the user uses to send and receive bitcoin. The distributed digital record known as the blockchain served as the model for the development of Bitcoin. A public ledger is a digital system that records transactions and associated information in multiple locations at the same time. Blockchain is one type of such system. Each transaction's details, including the buyer, seller, date, time, value, and exchange code, are stored in a blockchain's blocks. Because blockchain technology makes it extremely difficult to hack a system or fabricate data stored on it, data stored on it is safe and immutable. Every machine connected to a blockchain network has a copy of the ledger in order to prevent single points of failure. All of the distributed ledger's blocks need to be updated whenever one is modified. Because blockchain technology is decentralized, it is not governed by a single entity. Making blocks also gets harder when you have to figure out codes. On a PC or smartphone, bitcoin is kept in a digital wallet app. Wallets for cryptocurrencies are one of the best ways to protect bitcoin. Moreover, there are numerous varieties of wallets. With software wallets, users can keep a small amount of their bitcoin for daily use on a computer or smartphone and store the rest in an offline wallet. Wallet software installed on a USB drive or live CD, as opposed to the internet, allows for the physical security of offline wallets. Another type of offline wallet is a hardware wallet, which is a physical device like a flash drive that holds the user's private keys. Since signed transactions are completed on the device, the private keys are never exposed, even when connected to another device. Transaction authorization in multi-signature

wallets requires the use of two or more private keys. This eliminates a wallet's potential for being accessed in the event that it is lost or stolen. Three keys are kept safe: one on the user's mobile device, one in a secure location for backup, and one that can be kept with a multisignature provider.

Bitcoin transactions have an average size of 250 bytes and the block size is set at 1 MB, which means it can contain upto 4000 transactions on average every 10 minutes. But now the average has become 600 bytes due to complexity of the transactions[9]. Mining is the process of adding new bitcoin transactions into the system. Bitcoin miners use software that takes advantage of their processing power to solve algorithms related to transactions. In return, they get a certain quantity of bitcoin every block. This encourages miners to continue cracking the algorithms pertaining to transactions, thereby bolstering the system as a whole. Proof of work is the term for the procedure.

Initially, bitcoin mining was done on individual computer processors, or CPUs; the more cores and faster a CPU had, the more profit it could make. Following this, the majority of bitcoin miners started utilizing application-specific integrated circuits and field-programmable gate arrays before switching to multi-graphics card systems. These actions were taken in an effort to use less electricity and discover more hash codes below a specified target. Because people are willing to trade bitcoins for goods, services, and other currencies, it has a value comparable to other currencies. However, since its launch in 2009, the price of bitcoin has repeatedly increased, decreased, and then increased exponentially again. The swings are viewed as volatile by many. The stock market's prices have fluctuated because of a variety of reasons, such as businesses endorsing or rejecting the currency and remarks made by well-known figures.

But there are additional factors that contribute to the value of bitcoin. For instance, a currency must possess some of the following qualities in order to be accepted: it must be durable, transportable, scarce, and difficult to counterfeit.

Bitcoin has transformed digital transactions and the finance industry since its launch in 2009. It also introduced the underlying blockchain technology and a revolutionary idea: cryptocurrency. Bitcoin has completely changed our understanding of money, trust, and decentralized networks. We will examine the intricacies of Bitcoin's operation in this essay, covering everything from transactions to mining, in order to give readers a thorough grasp of the underlying technology and operational procedures. Fundamentally, Bitcoin is a digital currency created to facilitate peer-to-peer trades without the use of middlemen like banks. It is based on a distributed ledger known as the blockchain and runs on a decentralized network. All Bitcoin transactions are permanently recorded in chronological order on the blockchain, which guarantees security and openness.

In order to manage and store Bitcoin, the user must first create a digital wallet, which is essentially a program. To approve transactions, the wallet creates a private key; to receive Bitcoin, it creates a distinct public address, which is a cryptographic key.

Starting a Transaction: The wallet owner starts a transaction in order to transfer Bitcoin to a recipient. To complete this action, the recipient's public address and the desired amount must be supplied.

Transaction Signing: To verify ownership and approve the transfer, the wallet signs the transaction using the private key. The transaction's integrity and security are guaranteed by this signature.

Transaction Propagation: After the transaction is signed, it is sent out into the Bitcoin network, where it is intercepted by different nodes, or computers, that look after the network infrastructure. These nodes verify that the sender has the necessary balance and that the transaction complies with the network's regulations.

Mining and Inclusion: The mempool is a pool where legitimate transactions are gathered. Special network users known as miners choose transactions from the mempool and add them to a fresh block. Miners compete to solve proof-of-work problems, which are challenging mathematical puzzles.

Updates on Consensus and Blockchain: The new block is appended to the blockchain by a miner upon solving the puzzle. All nodes in the network must concur on the legitimacy of the block and the transactions within it in order for the consensus mechanism to function.

Transaction Confirmation: When a transaction is appended to the blockchain and incorporated into a block, it is deemed confirmed. The number of confirmations is determined by the security requirements of the network; higher levels of confidence are associated with more confirmations.

Mining is the process of producing new Bitcoins as well as verifying and adding transactions to the blockchain. Mining is a crucial part of the Bitcoin network and serves a variety of purposes.

Transaction Initiation: To begin the process, users open transactions in their Bitcoin wallets. The recipient's Bitcoin address and the amount to be sent are included in these transactions.

Transaction Broadcasting: Newly created transactions are received by the Bitcoin network. The security and validity of every transaction are ensured by a digital signature that is generated using the user's private key.

Mempool: Once a transaction is approved, it is placed in a "mempool" to await verification and subsequent addition to the blockchain. Miners select transactions from this pool to include in a new block.

Proof-of-Work: Miners take on to solve a computationally demanding mathematical puzzle, and the first person to finish gets to propose the next block. Players compete to solve a difficult mathematical puzzle known as the "proof-of-work." To

solve this puzzle, you need to find a specific value known as the "nonce," which, when combined with the transaction data and the previous block's hash, produces a hash value that has a fixed amount of leading zeros. To maintain an average block creation time of approximately ten minutes, the proof-of-work puzzle's difficulty varies over time. The more miners that participate, the harder the puzzle becomes, and vice versa.

Block Creation: The miner adds the validated transactions for a new block once the puzzle has been solved.

Reward and Fees: In exchange for their efforts in cracking the code and appending a fresh block to the blockchain, miners receive fresh Bitcoins as well as transaction fees.

Blockchain Consensus: Nodes validate the proposed block after it is disseminated to the network. Consensus mechanisms guarantee that the state of the blockchain is agreed upon by all nodes. The right to propose the following block belongs to the miner who solves the proof-of-work puzzle first. Together with the unique "coinbase transaction," which pays the miner with freshly created Bitcoins and transaction fees, this block also contains the mempool's validated transactions. All transactions contained in the recently added block are regarded as verified. Depending on the security requirements of the network, a transaction may receive fewer or more confirmations; the more confirmations received, the higher the confidence level in the transaction's validity.

The decentralized technology employed by Bitcoin is truly amazing; mining and transactions coexist peacefully to produce secure and efficient financial transactions. Its underlying blockchain technology provides transparency, security, and immutability, making it a promising invention that has the potential to completely transform financial systems as well as other industries. Understanding the intricacies of Bitcoin's operations is crucial to understanding the digital currency's inner workings as well as its broader impact on our evolving financial landscape. From the standpoints of the user, who starts the transaction, and the miner, who verifies and adds the transaction to a new block on the Bitcoin blockchain, a Bitcoin transaction is a complicated process that takes several steps. [2]

User's viewpoint

- **Start of Transaction**: A user chooses to transfer Bitcoin to a different user or organization. To initiate a new transaction, they use a Bitcoin wallet, which can be a hardware or software program.
- **Transaction Information**: The recipient's Bitcoin address—a cryptographic representation of their wallet—is specified by the user. Additionally, they input the desired Bitcoin transfer amount.
- **Selection of Input**: To finance the transaction, the particular unspent transaction outputs (UTXOs) are chosen by the user's wallet. Unused Transaction Exchange Orders (UTXOs) are Bitcoin that the user has received in the past

but hasn't yet been spent. There must be enough UTXOs in the user's wallet to cover the transaction amount.

- **Creation of Transactions**: The transaction is assembled by the user's wallet, which also contains references to the recipient's address, the amount that is specified as an output, and the chosen UTXOs as inputs.
- **Electronic Signature**: The user's wallet creates a digital signature with their private key in order to approve the transaction. This signature guarantees that only the legitimate owner may use the chosen UTXOs and verifies who owns them.
- **Sending Out the Transaction**: The Bitcoin network receives notice of the signed transaction. It is transmitted to different network nodes, or the computers that take part in upkeep of the Bitcoin network.

And a more complicated and elaborate structure can be seen in the miners' end of the work.

Miner's Perspective

- **Taking in Transactions :** Network nodes, including miners, receive the broadcasted transaction and add it to their mempool, which is a temporary storage area for pending transactions.
- **Transaction validation :** Miners can verify the legitimacy of a transaction by checking its digital signature, ensuring that it conforms with network rules, and ensuring that the UTXOs used as inputs are real and unused.
- **Mempool Involvement :** Once confirmed, the transaction is added to the mempool. Among the many factors that miners consider when selecting transactions from the mempool are transaction fees and transaction age.
- **Mining and Proof-of-Work :** On the Bitcoin blockchain, miners engage in a competitive process to produce a new block. In order to complete this process, a computationally complex mathematical puzzle called the "proof-of-work" must be solved. The right to suggest the following block belongs to the first miner to solve this riddle.
- **Assembling Blocks :** The mempool's verified transactions are included in a new block that is put together by the victorious miner. Additionally, they contain a unique transaction known as the "coinbase transaction."
- **Block Transmission :** The suggested block is disseminated throughout the Bitcoin network, where additional nodes confirm its authenticity.
- **Updates on Consensus and Blockchain :** Network nodes come to an agreement regarding the proposed block's validity. The new block is appended to the blockchain and the transactions contained within are deemed confirmed if consensus is reached.

- **Confirmation of Transaction :** It is confirmed when the user's transaction is included in a block. The security requirements of the network determine how many confirmations a transaction receives. Increased confirmations boost trust in the legitimacy of the transaction.

3.3 Reinforcement Learning (RL)

Reinforcement learning (RL), a branch of machine learning, examines how intelligent agents ought to act in particular scenarios in order to maximize the idea of cumulative reward. Reinforcement learning is one of the three basic paradigms of machine learning, along with supervised and unsupervised learning. Reinforcement learning differs from supervised learning in that it does not rely on labeled input/output pairs or the explicit correction of suboptimal actions. The main objective instead is to strike a balance between exploitation (of previously discovered knowledge) and exploration (of uncharted territory).

Since dynamic programming techniques are used in numerous algorithms for reinforcement learning within this context, the environment typically appears as a Markov decision process (MDP). Reinforcement learning algorithms differ from conventional dynamic programming techniques primarily in that they target large Markov decision processes in which exact techniques become impractical, rather than assuming knowledge of a precise mathematical representation of the Markov decision process [8]. The mathematical study of decision-making is known as reinforcement learning (RL). It comprises figuring out the best course of action to pursue in order to optimize advantages. As with children exploring their environment and learning skills that help them achieve goals, this ideal behavior is learned through interacting with the environment and observations of the way it responds. When there is no supervisor present, the student has to figure out on their own what steps to take in order to maximize the reward. This is like doing a trial-and-error search with this discovery process. The potential for a later reward is just as important as the immediate one for determining the quality of an action. Because it can learn the behaviors that eventually result in success in an unfamiliar environment without the help of a supervisor, reinforcement learning is an incredibly potent algorithm.

Two broad categories of reinforcement learning algorithms are model-free and model-based algorithms.[31] Specifically, model-free algorithms do not build an explicit model of the environment or the MDP. They are more like trial-and-error algorithms, where the best course of action is determined directly by using actions to conduct environmental experiments. Algorithms without models can be either policy-based or value-based. Value-based algorithms believe that an accurate estimation of each state's value function leads directly to the optimal policy. The agent communicates with the environment to examine the motions of states and rewards through the use of a recursive relation that is characterized by the Bellman equation.It is possible to estimate the MDP's value function given enough trajectories. Finding the best course of action after the value function is established only requires acting avariciously in relation to the value function at each stage of the procedure. Popular value-based algorithms include Q-learning and SARSA. Conversely, policy-based algorithms do not need to model the value function; instead, they estimate the op-

timal policy directly. They use learnable weights to directly parametrize the policy, turning the learning problem into an obvious optimization problem. Like value-based algorithms, the agent samples state and reward trajectories, but it makes explicit use of this information to optimize the strategy by increasing the average value function over all states. The Monte Carlo policy gradient (REINFORCE) and deterministic policy gradient (DPG) are two well-liked policy-based reinforcement learning algorithms. Throughout the training process, instability is brought about by the wide variety of policy-based approaches. Value-based approaches are not suitable for modeling constantly changing spaces, despite being more stable. Combining value-based and policy-based techniques, the actor-critic algorithm is one of the best algorithms for reinforcement learning. In order to enable the efficient use of training data with stable convergence, this algorithm parametrizes both the policy (actor) and the value function (critic). Reinforcement learning is one area of machine learning. It entails taking the right action to optimize reward in a particular situation. Numerous software programs and gadgets use it to figure out what the best course of action or behavior is in a certain situation. In contrast to supervised learning, which involves training the model with the correct answer pre-existing in the training set, reinforcement learning involves training the model in the absence of an answer and using the reinforcement agent's judgment to determine how to accomplish the given task. Even in the absence of a training dataset, it will eventually learn from its experiences. Algorithms that use reinforcement learning interpret results and determine the best course of action are used. The algorithm gets feedback after every action to help it decide if the decision it made was neutral, wrong, or correct. It's a useful method for automated systems that need to make numerous small decisions on their own without human assistance. A self-governing, self-teaching system, reinforcement learning basically learns by making mistakes. It learns by doing in order to attain the best results, or, to put it another way, it takes actions with the intention of maximizing rewards.

Reinforcement learning (RL) and deep learning are combined in the machine learning subfield known as deep reinforcement learning (deep RL). Computational agent learning to make decisions through trial and error is a problem that RL takes into consideration. By integrating deep learning into the system, deep reinforcement learning (deep RL) enables agents to make decisions based on unstructured input data without requiring manual state space engineering. Deep reinforcement learning algorithms have the capacity to process very large inputs such as every pixel that is displayed on a video game and determine the best course of action for achieving a given goal such as raising the score in the game. Applications of deep reinforcement learning are numerous and include robotics, computer vision, natural language processing, video games, education, transportation, finance, and healthcare. Finding a balance within exploration and exploitation is crucial for an RL agent. the dilemma of deciding which activities to pursue in order to find higher rewards those that are known to yield substantial rewards or which should be explored first. RL agents usually use a stochastic policy (e.g., a Boltzmann distribution in discrete action spaces, or a Gaussian distribution in continuous action spaces) to gather data in order to induce basic exploration behavior. Curiosity-driven or novelty-based exploration seeks to motivate the agent to explore unknown ground in search of the best answers. This is achieved by adding terms that promote exploration and by

changing the loss function or even the network's architecture.

The distinction between off-policy algorithms, which may derive a policy from data created by a random policy, and on-policy algorithms, which must assess or enhance the policy that gathers data, is crucial in reinforcement learning. Value-function based approaches, like Q-learning, generally have better sample-efficiency and are more suitable for off-policy learning; this is because they reuse data, which lowers the amount of data needed to learn a task. The furthest point of offline (or "batch") reinforcement learning is when a policy is learned from an existing dataset without further environment interaction.

Determining an agent's reward function based on its behavior is known as inverse reinforcement learning. By estimating the demonstrator's reward and using RL to optimize a policy to maximize returns, inverse reinforcement learning (also known as apprenticeship learning) can be utilized for learning from demonstrations. For inverse reinforcement learning and imitation learning in different forms, deep learning techniques have been applied.

A group of agents that collaborate and co-adapt rather than a single agent is used in many reinforcement learning applications. These agents can operate cooperatively, as in many multi-agent systems in the real world, or competitively, as in many games. The problems presented in this setting are studied through multi-agent reinforcement learning.

Online reinforcement learning involves the agent directly gathering data through interaction with the environment to build up a batch of experience. It then immediately applies what it has learned from this experience (or replay buffer) to modify its policy.

Over the past few years, reinforcement learning has advanced quickly. From tabular approaches that are limited to solving simple toy problems, reinforcement learning has become capable of handling extremely complex problems like playing Go, developing robotic manipulation skills, or operating autonomous vehicles. Sadly, RL has not been widely adopted for practical purposes. Although existing RL techniques have demonstrated their capacity to identify high-performing policies for complex problems involving high-dimensional raw observations (like images), their use is frequently problematic or unfeasible. This stands in direct opposition to supervised learning approaches, which are widely used and very successful in many industries and research domains. Reinforcement learning mainly depends on:

Agent: The software you teach to accomplish a task you designate.

Environment: The setting in which an agent acts.

Action: An action taken by the agent that modifies the surroundings.

Reward: An action's assessment, akin to feedback.

States: The agent notices this. In an interactive setting, agents learn by making mistakes and applying feedback (Reward) from their own experiences and actions. In essence, the agent explores various actions on the environment and gains knowledge from the responses it receives. The objective is to identify an appropriate course of action that would maximize the agent's cumulative total reward.

Policy: In traditional reinforcement learning theory, the term "functional approximation" refers to the use of neural networks to approximate the policy, which is essentially a function that maps states to actions. In a reinforcement learning problem, the ultimate objective is to either learn the parameters of this functional approximation or learn a policy, which determines a distribution over steps conditioned on states, π(a|s). Reinforcement learning is an iterative process that gathers data through interaction with the environment. In RL theory, this information is also called experiences. If we comprehend how an agent interacts with its surroundings, it becomes simple to understand why data is referred to as experience.[8]

On-Policy RL

Usually, the most recent learned policy is used to gather experiences, which are then used to enhance the policy. This is a virtual conversation. In order to gather the samples, the agent engages with the surroundings.

In on-policy reinforcement learning, πk gathers its own data and updates the policy. To decide which spaces and actions to explore and sample next, we optimize the existing policy πk. This implies that we will work to enhance the action selection policy that the agent currently employs. The policy used to generate data is referred to as the behavior policy.

The agent in on-policy reinforcement learning (RL) keeps track of both a value function (Q or V) and a policy (often represented as π). The value function calculates the probable cumulative reward an agent can receive from a specific state or state-action pair, whereas the policy is a visualization from states to actions. Trial-and-error interaction is how on-policy RL agents engage with their surroundings. They choose actions based on their existing policy, carry them out in the environment, and track the ensuing states and rewards. Exploration and exploitation must be balanced in on-policy reinforcement learning. In addition to exploring the environment in search of better policies, the agent makes use of its current policy in order to maximize rewards right away. Finding this balance is frequently difficult.

The agent gathers a series of experiences and uses this information to improve its policy through updates. Using methods such as policy gradients, which modify the policy to raise the likelihood of actions that result in higher returns, is a popular strategy in on-policy reinforcement learning. In addition to improving its policy, the agent estimates what is anticipated from every state or state-action pair in order to assess the value of its policy. The agent can make well-informed decisions about what to do in different situations thanks to these value estimates.

When learning an optimal policy, on-policy reinforcement learning algorithms fre-

quently need more data samples than off-policy approaches. This is due to the fact that they depend on the experiences gathered under the present policy, which might not be the best for further investigation.

When compared to their off-policy counterparts, on-policy algorithms are typically more robust and less susceptible to divergence or overfitting. But in some circumstances, this stability might mean a slower rate of learning.

Numerous fields, such as robotics, gaming, natural language processing, autonomous vehicles, and recommendation systems, use on-policy RL. It works especially well in situations where there must be continuous contact with the environment and where learning must gradually adjust to environmental changes.

Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Asynchronous Advantage Actor-Critic (A3C), REINFORCE, and Proximal Policy Optimization (PPO) are common on-policy RL algorithms. Certain elements of policy optimization such as sample stability and efficiency are addressed by these algorithms[8].

Off-policy RL

In the traditional off-policy setting, the agent's experience is added to a data buffer (also called a replay buffer) D. With each new policy πk, more data is collected, so samples from π0, π1,..., πk make up D. All of this data is utilized to train an improved new policy, πk+1. The agent interacts with the environment to collect the samples.The computation can utilize older samples collected with the older policies thanks to off-policy learning. The policy is updated by sampling a buffer that holds interactions and experiences from its own prior policies. Sample efficiency rises because we are unable to collect samples once more whenever a policy is changed.

Off-policy reinforcement learning, like on-policy RL, entails sustaining a policy (typically represented as π) and a value function (typically represented as Q or V). The value function calculates the expected cumulative rewards, while the policy specifies the agent's approach to choosing actions in various states. The agent can acquire knowledge from a predetermined dataset that was gathered using any random policy in off-policy reinforcement learning. This implies that the agent can engage with the environment in ways other than direct interaction. Typically, state-action-reward transitions make up the dataset. Off-policy RL makes it possible to distinguish clearly among exploration and exploitation. With one policy—the behavior policy—the agent can investigate its surroundings, and with another—the target policy—it can take advantage of its knowledge. One of the main benefits of off-policy approaches is this division.

Off-policy RL algorithms utilize important sampling techniques for adjusting for policy changes resulting from differences between the target policy (the policy that is being learned) and the behavior policy (used for information collection). When estimating expected returns, this difference is taken into account using importance sampling weights [23].

Off-policy reinforcement learning algorithms seek to use the gathered dataset to learn and enhance the target policy. Typical techniques for maintaining the value function and policy independently are Q-learning and SARSA. Because off-policy RL techniques can make use of previously gathered data, they may be further sample-efficient than on-policy techniques. This makes them useful in scenarios where gathering real-time data is risky or expensive.

Certain off-policy reinforcement learning algorithms may display slow convergence or stability problems, particularly in highly dimensional or constantly changing space scenarios. To lessen these problems, strategies like experience replay and targeted networks are used.

Applications of off-policy reinforcement learning can be found in many fields, such as recommendation systems, autonomous systems, finance (portfolio management and trading, for example), and gaming (Deep Q-Networks for Atari games, for example). It is especially helpful in situations where real-time data collection is hazardous or impractical.

Actor-Critic techniques like Advantage Actor-Critic (A2C), Q-learning, and Deep Q-Networks (DQN) are popular off-policy reinforcement learning algorithms. Several facets of off-policy learning, such as sample efficiency and policy stability, are the focus of these algorithms.

Online RL

A versatile and adaptable machine learning framework that has gained popularity recently is online reinforcement learning, or RL. Online reinforcement learning (RL) uses an agent that learns by trial and error how to make choices in real-time environments, as opposed to offline RL, which uses fixed datasets. This paper examines the fundamental ideas, difficulties, and uses of online reinforcement learning, highlighting its importance in the rapidly developing field of artificial intelligence.The foundational ideas of reinforcement learning underpin online reinforcement learning. It consists of an agent, an environment, and a reward signal as its three primary parts. By interacting with the surroundings, the agent acts in a way that maximizes a cumulative reward over time. Online RL, on the other hand, presents different difficulties because it requires constant interaction with the environment, unlike offline RL. Online role-playing is characterized by a trade-off between exploration and exploitation. Exploiting identified actions to maximize instant rewards must be balanced with exploring novel behaviors to learn regarding the environment. Because of this trade-off, adaptive algorithms must make wise decisions because ill-informed ones could have unfavorable effects. With online reinforcement learning, agents actively interact with real-time environments to learn through a constantly changing and adaptable approach to machine learning. To fully realize the potential of online RL, certain challenges posed by its dynamic nature—such as non-stationarity, safety concerns, and exploration—must be addressed. However, this methodology has shown to be extremely beneficial in numerous applications where ongoing learning and instantaneous decision-making are crucial.

Offline RL

A branch of machine learning called offline reinforcement learning (RL) focuses on teaching agents to make decisions using static datasets as opposed to real-time environment interaction. Offline reinforcement learning (RL) uses past data to improve policies and enhance decision-making, in contrast to online RL, which works in dynamic and changing environments. We explore the fundamentals, difficulties, and uses of offline reinforcement learning in this essay, highlighting the value of offline RL in situations where real-time data collection is difficult or expensive. The fundamental ideas of conventional reinforcement learning, such as the use of an agent, an environment, and a reward signal, are also shared by offline reinforcement learning. But it's different in the way it gathers information. The agent does not actively engage with the environment; instead, it depends on a fixed dataset that is typically gathered through simulations or observations. Finding a policy that works well in similar situations is the main objective. Distinctive Strengthening In situations where immediate data collection is difficult or impossible, learning is a useful strategy. It has applications in many different domains, despite having its own set of challenges, such as data quality, distribution imbalance, and exploration limitations. In circumstances in which communication with the environment is impractical, offline reinforcement learning (RL) is an essential tool for decision-making because it allows agents to learn from past data.

The Bitcoin network functions under a distinct set of regulations and incentives that preserve its safety and efficiency. It is praised as an innovation in technology and an innovator in the field of cryptocurrencies. The dynamic block size is one important component that maintains the Bitcoin network running. The total number of transactions that can be handled in a specific amount of time is determined by the block size, which also controls the capacity of every block in the blockchain. Sustaining the network's overall functionality, scalability, and efficiency requires accurate forecasting and control of the dynamic block size.

Determining the ideal block size is a difficult undertaking that necessitates evaluating a number of variables, such as the amount of transactions, congestion in the network, and fee dynamics in the market. The main difficulty lies in finding a balance between minimizing network utility by analyzing as many requests as possible and avoiding congestion, which can result in bottlenecks in the network.

An essential component of this effort is online reinforcement learning, which allows the Bitcoin network to make decisions about block size modifications in real time. accustomed agents that continuously monitor network conditions, learn from past actions, and modify the size of blocks policy to maintain an ideal balance between processing transactions and network efficiency can be created using algorithms like DDPG, SAC, and TD3. As the network runs, these online algorithms for reinforcement learning can be very helpful in optimizing block size choices.

The online method is enhanced by offline reinforcement learning, which lets us examine past data and refine block size prediction models. We can draw conclusions

from historical Bitcoin network data, spot trends, and improve block size regulations thanks to methods like ARS. We can create more precise predictive models and train agents that can make wise decisions when operating online by using a wealth of historical transaction data, fee market habits, and network congestion.

We have created a unique environment that mimics the dynamics of the Bitcoin network, including transaction arrivals, miner behavior, fee market swings, and block creation, in order to effectively apply these reinforcement learning techniques. Our reinforcement learning agents can adapt and learn in a controlled environment before being deployed in the real Bitcoin network thanks to this realistic testing ground.

Ensuring the continuous success and scalability of the Bitcoin network heavily relies on the prediction and management of the dynamic block size. Through the integration of online and offline reinforcement learning methodologies and the utilization of algorithms like DDPG, SAC, ARS, and TD3, our objective is to create intelligent agents that possess the ability to dynamically modify the block size in reaction to network circumstances. This novel strategy could keep the Bitcoin ecosystem at the forefront of the digital economy by improving scalability, maintaining efficiency, and optimizing network performance.

Chapter 4

Methodology

The approach we are adapting to solve the issues discussed till now is illustrated in the following diagram.

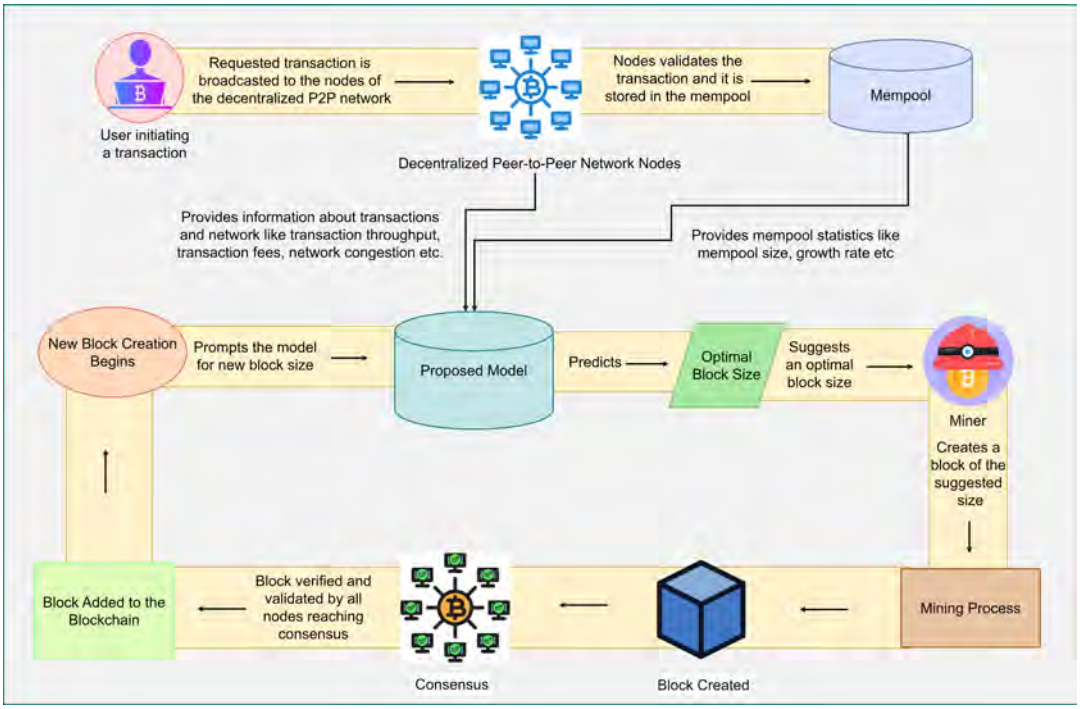

Figure 4.1: Operational diagram of blockchain network with the proposed model

Figure 4.2 depicts how the proposed model works. Following is a brief description of the individual modules.

Memory Buffer

Memory buffer works as an efficient sampling method for training updates. It is an experience-storing data structure. It allows for off-policy learning by storing previous experiences (state, action, reward, and next state). During training, samples from the memory buffer are utilized. The experiences that are gathered during the agent's interaction with the environment are stored in the memory buffer. Each experience is composed of a tuple (state, action, reward, and future state). When

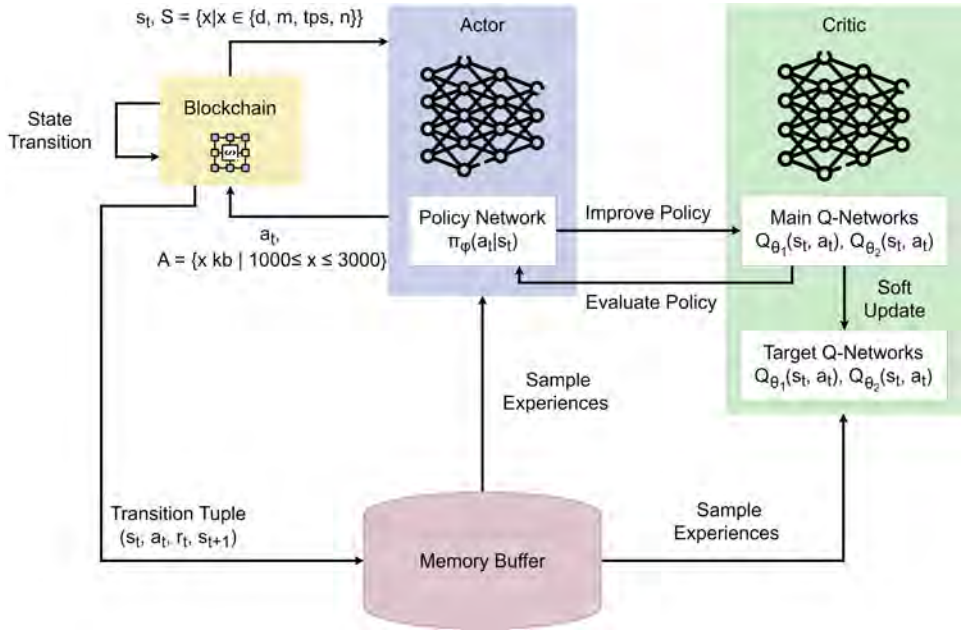

Figure 4.2: The top level overview of the proposed model for predicting optimal block size

the buffer fills up, fresh information replaces the old information that was previously saved for each of these encounters. Because SAC is an off-policy method, it separates the policy being optimized from the data collecting policy. The memory buffer allows the algorithm to collect samples from previous interactions that might not be indicative of the current policy, allowing it to learn from a wider range of situations. The memory buffer improves sample efficiency through the reuse of previous experiences. SAC can benefit from a varied range of experiences rather than depending just on the most recent ones, which could hasten the learning process. One way to disrupt the temporal correlation between consecutive samples is to store experiences in the buffer. This is especially crucial for training stabilization since temporally uncorrelated data might shield the learning algorithm from being unduly impacted by the data's sequential patterns.

Actor module

The agent's policy is contained in the actor module. It outputs a probability distribution over the continuous action space and accepts the current state as input. A Gaussian distribution with a mean and a standard deviation is frequently used to parameterize this distribution in SAC. Usually, the actor is represented by a neural network. This network's design, which is taught during training, is in charge of translating states into action distributions. SAC works with continuous action spaces, in contrast to algorithms created for discrete action spaces. The parameters of a probability distribution, such as the mean and standard deviation for a Gaussian, are output by the actor and are used to sample actions. This makes a large variety of continuous and differentiable action spaces possible for SAC to handle. The trade-off between exploration and exploitation revolves around the actor module. By including an entropy regularization term in its objective function, SAC promotes policies with higher entropy, which in turn encourages exploration. This

maintains equilibrium between investigating novel approaches and making use of the existing policy. The probability distribution that the actor produces is sampled to determine the actions that it chooses. The environment is then used to carry out these actions. Both effective policy optimization and efficient exploration depend on the parameterization of the action distribution.

Critic module

The critic module's main responsibility is to estimate the value function, or in the case of SAC, the Q-function. The expected cumulative reward of performing a specific action in a given state and then adhering to the current policy is represented by the Q-function. Usually, the critic is implemented as a neural network, where training allows the architecture to be learned. It receives a pair of states and actions as input, and produces the estimated Q-value, which is the expected return. To estimate the action-value function, or Q(s, a), where "s" is the state and "a" is the action, SAC uses a critic. The expected return of acting "a" in state "s" and adhering to the existing policy is shown by the Q-value. The critic module indirectly aids in the estimation of the state-value function, or V(s), in addition to the Q-function. The expected return of being in state "s" and adhering to the current policy is represented by the state-value function. The temporal difference (TD) error, which quantifies the difference between the target Q-value and the predicted Q-value, is used to update the critic during training. The critic's parameters are adjusted based on this error to enhance its value estimations. Target networks are frequently used by SAC for the critic in order to stabilize training. The parameters of the main critic network are periodically updated to these target networks, which have slowly moving parameters. This method aids in reducing the problems associated with reinforcement learning's moving target problem.

4.1 Data Collection

Collecting the dataset was rather easier due to bitcoin being an integral part of cryptocurrencies, which means complete transparency. This means that anybody online can view the blocks and all the transactions which are related to the block itself. But this does not, however, provide any background information about the sender or receiver. This was important in the case of analyzing the bitcoin network. While we can easily get our hands on the block information, discovering the network status at any moment especially when the block was created is difficult. To start the research for this paper, we needed information such as average block size, transactions per second etc. And we found that information from a website. This website contained only the average values of the day,but we needed data for the hour or even better, data for the minutes. But we had to make do with what we had taken from that website. We used data from 29th, July 2020 to 28th July 2023. From this time period we took 7 tables which are described below:

- **Avg-confirmation-time:** The Average confirmation time for transactions on the blockchain network, measured in seconds. This measure offers information about the network's efficiency and transaction processing speed.

- **Difficulty:** The degree of difficulty of the consensus algorithm used by the blockchain network. A desired block production rate is maintained throughout time by adjusting the difficulty. It displays the computational work miners must do to verify transactions and build new blocks.
- **Mempool-count:** The amount of transactions in the network's Mempool that are presently unconfirmed. A crucial sign of network congestion and transaction backlog is the Mempool count.
- **Miners-revenue:** The total amount of money the blockchain network miners made. Both block rewards and transaction costs are included in this. The amount of money that miners make is a gauge of the financial incentives for mining.
- **N-transactions-per-block:** The average amount of transactions each block contains. This measure gives information on the throughput and capacity of the network's transactions.
- **Transactions-per-second:** Measured in transactions per second (TPS), the speed at which transactions are completed on the blockchain network. This number shows how well the network can manage a large volume of transactions.
- **Avg-block-size:** The amount of data that makes up a block on the blockchain network on average. The size of each block has a significant role in establishing the network's capacity and resource needs.

Other than meempool count and transactions per second, we had 1 null value in all of our tables, so it was quite simple to handle these values. But to make sure the dataset is adaptable with our algorithms we have to do some pre-processing to it.

4.2 Data Synthesis using Gaussian Distribution

The distributions of real-world data are frequently remarkably similar to those of gaussian distributions. The distribution of many natural events, including individual heights, measurement mistakes, and stock prices, closely matches a Gaussian distribution. By using Gaussian distribution we can generate synthetic data that resembles these real-world patterns using Gaussian data creation. Datasets for data exploration and visualization exercises can be produced with the aid of gaussian data creation. Gaussian data generation can be utilized in simulation and modeling situations to produce synthetic datasets that represent particular statistical qualities or characteristics of the data we want to model.

To generate the same contextual data, we convert the daily data into data per minute. We have data of 1095 days, meaning our dataset contains 1095 rows worth of data. If we convert the data per day into data per minute we generate a total amount of 1,577,200 rows worth of data. Then for each feature we set a minimum and maximum value which is based upon the previous datasets values. If there are any values larger than or smaller than the values of the initial dataset, we clip that data from the generated dataset to make sure the dataset remains as correct as possible. Finally we use this generated dataset to make it compatible for the models.

For that we need MDP formulation.

4.3 Baseline Model: Long Short-Term Memory (LSTM)

4.3.1 LSTM Architecture

In the field of deep learning, recurrent neural network (RNN) architectures such as Long Short-Term Memory (LSTM) have become increasingly popular, especially for applications that need sequential data, like time series analysis, speech recognition, and natural language processing. The vanishing gradient issue is a basic issue with conventional RNNs that is resolved by LSTM networks.

In addition to handling gradient vanishing and explosion problems that can arise during training in standard RNNs, LSTMs are made to capture long-range dependencies. Because of this, they are highly suitable for problems involving the learning and prediction of patterns over long data sets. The memory cell, which is the main invention of LSTM, has the ability to learn and retain information across long time steps.

There are three interacting gates in an LSTM cell:

- **Forget gate (ft)** : The Forget Gate is responsible for deciding which data from the previous cell state should be stored or erased. It generates a "forget" vector that scales the cell state's elements based on the input, which includes the previous cell state and the current one.
- **Input gate (it)** : The input gate determines which newly acquired data should be kept in the cell state. Additionally, it creates an "input" vector that can be added to the cell state by taking the current input and the prior cell state.
- **Output gate (Ot)** : This component determines which data from the cell state should be utilized to generate the output. It generates an "output" vector, which is a filtered representation of the cell state, after taking into account the input and current cell state.

4.3.2 LSTM Training

Initially we chose Long Short-Term Memory (LSTM) since its architecture made learning and remembering information over extended sequences feasible, which regulate the flow of information within the cell. In our work, the primary dataset that we employed to train LSTM offers proof of past Bitcoin network transactions. This dataset, which includes a wide range of transactions, blocks, and user behavior, was carefully selected from a vast amount of data that can be found on blockchain websites. This dataset is composed of numerous online trades and transactions that were extracted from the decentralized Bitcoin blockchain. The present study employs a

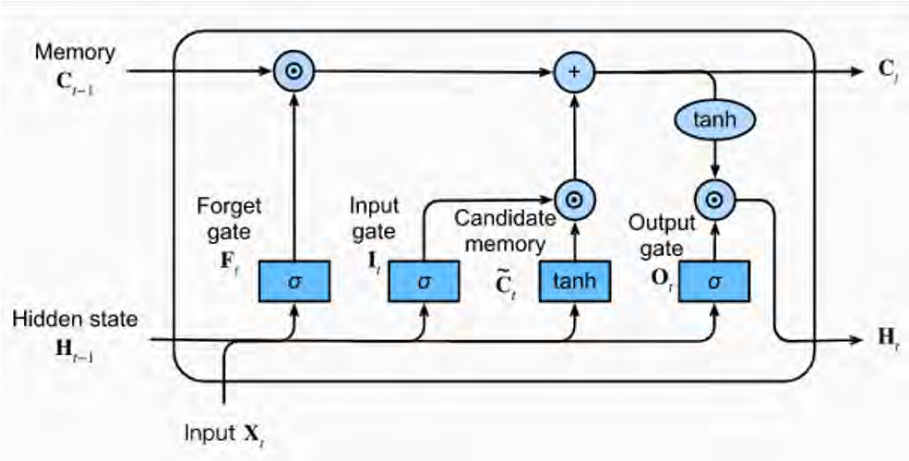

Figure 4.3: LSTM Architecture

meticulously chosen dataset that was taken from blockchain websites to investigate the realm of virtual currencies, decentralized ledger technology, and their intricate relationship altogether.

In this work, we implemented our Long Short-Term Memory (LSTM) model using multiple Python packages. These libraries included Scikit-Learn for data splitting, NumPy (imported as np), Pandas (imported as pd), MinMaxScaler for preprocessing data, and Keras for constructing the LSTM model. Matplotlib was also used for the data visualization.

Data Preprocessing

Starting with feature scaling which is a popular preprocessing method in machine learning. It modifies data to fit into a given range, typically between 0 and 1, without changing the relative relationships between the data. This can be especially helpful for machine learning algorithms that are sensitive to the magnitude of input characteristics, such as gradient descent-based techniques. In the work, we have used the MinMaxScaler from the scikit-learn library in Python to perform feature scaling on two arrays, X and y. X and y here are input features and target variables respectively. For our case input features are: average confirmation time, difficulty, mempool count, miners' revenue, n-transactions per block and transactions per second. Our target variable is average block size.

Following scaling, we separated our input and target data; 80% of the data will be used to train the machine learning model, while the remaining 20% will be used to assess the model's effectiveness. This division enables the model to be tested on a different set of data and trained on a different chunk to determine how well it generalizes to new, untested data. Testing for overfitting and evaluating the model's predictive power on fresh data is an essential phase in the model-building process.

LSTM Model

Using the Keras library, we built a Long Short-Term Memory (LSTM) neural network. The model's architecture is described in full below:

1. A Sequential model was created using Keras.
2. LSTM layers were added to the model for sequential data analysis.
3. Dense layers were used for the final output.

It generates sequences from your training and testing datasets using a function named "create_sequences". Time series or sequence-based machine learning tasks frequently use this type of sequence construction. Given that the model variable is set to 5, it generates sequences with a length of 5. To put it another way, it will utilize the five consecutive data points from the input features to forecast the value of the target variable for the following data point. This function requires three inputs. The input characteristics consist of a data sequence (X). The goal variable, which determines the length of the sequences it wants to produce, is linked to the input characteristics (y) and input size. After generating the sequence it returns two numpy arrays.

Initially, a sequential model has been initialized. Using Python's Keras deep learning toolkit, the model defines a neural network model. Regression tasks, where the objective is to predict continuous numerical values, seem to be the intended use case for this approach. One layer at a time can be added to a sequential model, which is a linear stack of layers. This enables the construction of a feedforward neural network.

The model is then expanded with LSTM (Long Short-Term Memory) layers. Recurrent neural networks (RNNs) of the LSTM type operate well with sequence data and time series forecasting. This indicates that there are fifty units, or neurons, in the LSTM layer. Information from the input sequences is processed by these neurons. The LSTM layer uses Rectified Linear Unit ('relu') activation function. It gives the model non-linearity and aids in its ability to pick up intricate patterns. A dense layer containing a single neuron is put after the LSTM layer. The output layer is in charge of generating a single numerical value that represents the model's forecast. Regression modeling is being used in this instance to forecast a continuous value.

Besides, throughout training, the weights of the model are updated by the Adam optimizer. This well-liked optimisation approach adjusts the training rate to converge more quickly.

Finally, it trains a neural network model in Python using the Keras deep learning library. The model is trained using the fit method with the given training data and parameters. The model will be trained for 50 epochs meaning it will go through the training data 50 times.Additionally, validation data is available to track the model's performance throughout training. It makes use of the goal values that match the testing sequences. At the conclusion of each epoch, the model's performance on the validation data is assessed. This makes it easier to keep an eye on whether the model is overfitting or functioning well on training data but badly on testing data.

4.3.3 LSTM Evaluation

After prediction we plotted training loss and valediction loss which is converging But after a time it is observed that the convergence rate is constant. Later on, the model's predictions on the training and testing datasets are stored in two arrays, respectively to evaluate it further. By using the trained model (model) on the matching input data sequences, these predictions are produced. The algorithm applies an inverse transformation to the expected values to return them to their original scale after it has made its predictions. After scaling your target variable for model training, usually as part of data preprocessing, you must perform this crucial step. It guarantees that the original, unnormalized values are used for the RMSE (Root Mean Squared Error) computations.

Once predictions have been converted back to their original scale. The RMSE for the training and testing datasets is computed in the following two lines. Regression model performance is often assessed using the root mean square error (RMSE) metric. It calculates the average difference between the goal values and the projected values. The formula for RMSE is:

$$RMSE = \sqrt{\sum (actual - predicted)^2 / N} \tag{4.1}$$

The elements of the formula are broken down as, actual: The values that are actually (observed) or the ground truth, predicted: The figures that a regression model projects, Σ: The sum symbol denotes that the squared differences for every data point should be added up, N: The total number of observations or data points utilized in the computation.

The RMSE values for the training and testing datasets are finally printed by the code, giving information about the accuracy and generalization capacity of the model to previously unseen data. A better fit between the model and the data is shown by lower RMSE values. The RMSE values are used to evaluate the accuracy with which the model's predictions match the actual target values. In our case the RMSE values for training and testing are very close which indicates underfitting.

4.4 Offline Reinforcement Learning (RL) Model

We carried out thorough data pre-processing in this section by using Gaussian distribution techniques to synthesize data. After that we created a Markov Decision Process (MDP) and carefully built the action space, observation space, and reward function. We then divided the dataset into training and testing sets, 75% for model training and the remaining 25% for evaluation in order to enable robust model training and assessment. Three different Reinforcement Learning (RL) models—Batch Constrained Q-Learning, Conservative Q-Learning, and Bootstrapping Error Accumulation Reduction (BEAR)—were applied in our methodology. Every model was created to tackle certain issues in the provided setting, resulting in a varied range of solutions.We used Fitted Q-Evaluation (FQE), a cutting-edge approach from Offline Policy Evaluation (OPE), to evaluate the effectiveness of our RL models. FQE offered a strict framework for assessing our models' performance, guaranteeing an

objective and trustworthy evaluation of their efficacy. This integrated methodology provides a thorough and in-depth exploration of the problem domain, providing insightful information about the performance and generalization capabilities of the implemented models. It combines data synthesis, MDP formulation, dataset partitioning, and utilization of diverse RL models evaluated through sophisticated OPE algorithms.

4.4.1 Data Pre-Processing

From the synthetic dataset, we then followed usual pre-processing steps such as handling missing values, data visualization, splitting data into train and test sets etc. After that, we have to prepare the dataset in a way that a reinforcement learning agent can use to learn which is Markov Decision Process (MDP). We will discuss MDP formulation in the next subsection in details.

4.4.2 Markov Decision Process (MDP) Formulation

The Markov Decision Process (MDP) formulation is a key paradigm in the study of decision-making and reinforcement learning. Decision-making difficulties can be formally and mathematically represented using MDPs. They provide a well-structured framework for simulating scenarios in which an agent interacts with its environment, takes actions, and is rewarded as a result of those activities. MDPs simplify difficult decision-making issues into an approachable form. This abstraction aids in high-level problem comprehension, isolates important elements, and facilitates analysis and solution. Making a series of decisions over time in order to solve many real-world problems has an impact on the outcomes and state of the world in the future. Such sequential decision-making scenarios are handled by MDPs, enabling the agent to make long-term plans.

The generated dataset we got from the gaussian distribution will be needed to be MDP formulated in order to use this for the training models. For this we need to decide the states, what features will consist of the states, what will the actions be, how do we determine the reward and based on these, what will the next state be.

- **Observations :** For our research, we will select 4 features to define the states, namely: Transactions-per-second, n-transactions-per-block, Mempool-count and difficulty. These 4 features will determine the states or observations.
- **Action :** The action for our research is predicting the dynamic blocksize.
- **Reward :** We define the reward function by minimizing the average confirmation time and maximizing the miners revenue.
  We formulate the reward like this:

$$R = W_m \times \frac{M_b}{M_B} + W_c \times \frac{C_B}{C_b} - \begin{cases} \frac{C_b}{W_c} & \text{if } C_B < C_b \\ \frac{W_m}{M_b} & \text{if } M_B > M_b \end{cases} \tag{4.2}$$

where M_B denotes the average miner's revenue of the blockchain and M_b denotes the miner's revenue of the block, W_m is the weight coefficient of miner's revenue, M. C_B denotes the average confirmation time of all transactions added in the blockchain and C_b denotes the average confirmation time of all transactions added in the block, W_c is the weight coefficient of average confirmation time, C.

- **Episode :** To round out the MDP, we also insert a done flag. This done value is by default 0. After we get all the data of a single day, we call that an episode and we make the done value 1.

To complete the formulation, we create a dictionary consisting of the current state, actions that can be taken in that state, rewards for whichever action is taken, and based on the action taken, what next state will the algorithm pursue. It will consist of the same 4 features of the next state. This way we can generate an MDP of the dataset and feed it into the reinforcement learning model.

4.4.3 Offline Reinforcement Learning Model Training

An array of tools and algorithms for knowledge extraction from static datasets are encapsulated in the Python library D3RLPY, which is used for offline reinforcement learning. It is a useful tool for academics and professionals who want to use historical data efficiently. The goal of offline reinforcement learning (RL), a well-known branch of reinforcement learning, is to train intelligent agents using past data instead of in-the-moment interactions [28]. The D3RLPY library, a flexible collection of algorithms made to handle the challenges of offline learning, is one of the mainstays of this field. In this portion, we explore the D3RLPY library and highlight three of its well-known algorithms: Bootstrapping Error Accumulation Reduction (BEAR), Conservative Q Learning (CQL), and Batch Constraint Q Learning (BCQ). We will examine the fundamental ideas underlying these algorithms, as well as their special qualities and offline reinforcement learning applications.

Batch Constraint Q Learning (BCQ)

Among the D3RLPY library, Batch Constraint Q Learning (BCQ) stands out as an innovative method for resolving offline reinforcement learning problems. BCQ uses a distributional approach to Q-value estimation, emphasizing the modeling of Q-value prediction uncertainty. This method makes it possible to handle the unknown dynamics better and have a deeper understanding of the environment. BCQ uses an action masking technique to limit the range of actions that can be chosen within the dataset. By using this constraining mechanism, the possibility of investigating high-variance or possibly dangerous actions is decreased. BCQ adds noise to the process of choosing an action. In order to maintain a balance between exploration and exploitation, this stochasticity pushes the agent to explore outside of the limitations. Algorithm 1 shows the detail working mechanism of BCQ. [14].

This algorithm uses constraint-based optimization to train the Q-function which is different from CQL as CQL used regularization and loss function to train the Q-function. BCQ abides by various action restrictions. This makes this algorithm

very effective while solving offline real-world interactions. The process begins when the algorithm gets the MDP data as input and all the transitions from the MDP. It also takes the amount of iterations as input as well. Target network update rate is also given as input which controls the rate at which the Q-network will be updated. The next input is mini-batch size N, which lets the algorithm know how many transitions will be sampled from Batch B(collection of the data) in each iteration. Max perturbation ϕ refers to the maximum perturbation which can be applied to the actions when the algorithm is sampling the actions. number of sampled actions, n is also given as input which indicates the amount of actions sampled for every state-action pair. And Finally a weight factor λ is given for regularizing the Q Function.

Conservative Q Learning (CQL)

With a risk-aware approach to offline reinforcement learning, Conservative Q Learning (CQL) tackles the exploration challenge without gathering new data. CQL emphasizes making decisions that are sensitive to risk. It seeks to teach a prudent approach that steers clear of acts carrying undue risk or uncertainty. Value at Risk (VaR) Minimization: CQL specifically reduces the Value at Risk, which is a gauge of possible worst-case performance. This strategy promotes safer behavior [14]. CQL, like BCQ, uses a distributional critic to provide additional details regarding the Q-value distribution. Making risk-aware decisions and comprehending uncertainty are aided by this. Algorithm 2 shows the detail working mechanism of CQL [14].
The usual off policies such as Q-learning often fail due to overestimation of the value of an action, which is caused by the differences between the dataset and the policy it learnt. On the other hand, Conservative Q-learning(CQL) reduces these overestimation issues and improves the stability of an offline RL algorithm. Due to enforcing conservatism, it teaches the Q-function that some action values might be over optimistic, which could cloud the judgment of the model. By doing this, Conservative Q-learning creates a mechanism which can participate in controlling the learning policies from various distinctive data distributions.

Bootstrapping Error Accumulation Reduction (BEAR)

Another sophisticated algorithm in the D3RLPY library is called Bootstrapping Error Accumulation Reduction, or BEAR. The goal of BEAR is to minimize the mistakes made in the process of bootstrapping. It seeks to reduce overestimation bias, a frequent problem in Q-learning. By discouraging risky behavior, policy regularization—which BEAR incorporates—enforces a more conservative policy.To identify the best policy with the least amount of risk, BEAR integrates value estimation and policy optimization.
The algorithm begins with a Q-ensemble which consists of random parameters. It also initializes actor network $\pi\phi$, Lagrange Multiplier α, initializes Q-networks. First it samples a mini batch of transitions from dataset D, from that mini batch, it samples P action samples which are used to compute the parameter λ. Then a target value y(s, a) is calculated for every state-action pain which is based on the Q values of the Q networks with the mixing parameter λ. It updates the Q-function for each Q-network in the ensemble by updating the parameters and minimising the mean squared bellman error,sample m action samples from the current policy and n ac-

tions from the dataset, D is used to compute the Maximum Mean Discrepancy loss. After that the actor network parameter and Lagrange multiplier gets updated while minimizing the values using dual gradient descent. Finally using these values the target Q-network gets updated. These steps are repeated for N iterations.

Algorithm 1: BEAR Q-Learning (BEAR-QL)

Input: Dataset D, target network update rate τ, mini-batch size N, sampled actions for MMD n, minimum λ

Initialize Q-ensemble $\{Q_{\theta_i}\}_{i=1}^{K}$, actor π_ϕ, Lagrange multiplier α, target networks $\{Q_{\theta'_i}\}_{i=1}^{K}$, and a target actor $\pi_{\phi'}$, with $\phi' \leftarrow \phi$, $\theta'_i \leftarrow \theta_i$

Reward:

$$R = W_m \times \frac{M_b}{M_B} + W_c \times \frac{C_B}{C_b} - \begin{cases} \frac{C_b}{W_c} & \text{if } C_B < C_b \\ \frac{W_m}{M_b} & \text{if } M_B > M_b \end{cases} \tag{4.3}$$

for $t \leftarrow 1$ **to** N **do**

  Sample mini-batch of transitions $(s, a, r, s') \sim D$

  Q-update:

  Sample p action samples: $\{a_i \sim \pi_{\phi'}(\cdot|s')\}_{i=1}^{p}$

  Define $y(s,a) := \max_{a_i}[\lambda \min_{j=1,\dots,K} Q_{\theta'_j}(s', a_i) + (1-\lambda)\max_{j=1,\dots,K} Q_{\theta'_j}(s', a_i)]$

  $\forall_i$,$\theta_i \leftarrow \arg\min_{\theta_i}(Q_{\theta_i}(s,a) - (r + \gamma y(s,a)))^2$

  Policy-update:

  Sample actions $\{\hat{a}_i \sim \pi_\phi(\cdot|s)\}_{i=1}^{m}$ and $\{a_j \sim D(s)\}_{j=1}^{n}$, n preferebly an intermediate integer(1-10)

  Update ϕ, α by minimizing Equation 1 using dual gradient descent with Lagrange multiplier α

  Update Target Networks:

  $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$; $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$

end for

Above is the algorithm explaining how Bootstrapping Error Accumulation Reduction works.

4.4.4 Evaluation using Offline Policy Evaluation (OPE)

Evaluating learned policies' efficacy and quality is a critical first step toward optimizing intelligent agents in the field of reinforcement learning. A collection of methods known as "offline policy evaluation" (OPE) is intended to assess and gauge a policy's effectiveness without the need for in-person interactions. Fitted Q-Evaluation (FQE) is a particularly potent and cutting-edge method among these strategies. This essay will examine the idea of OPE, its importance, and the inner workings of FQE, illuminating how it uses past data to assess and improve learned policies.

In the field of reinforcement learning, offline policy evaluation is essential, especially when getting fresh data from real-world interactions is expensive, time-consuming, or not possible at all. OPE methods aid in evaluating a policy's quality by revealing how well it functions in a particular setting. Through policy comparison made possible by OPE, researchers and practitioners can choose the best policy for a given task. OPE aids in assessing the possible dangers and unknowns related to a learned policy, an important factor in applications where safety is paramount. It is an essential part of offline reinforcement learning because it helps evaluate policies learned from static datasets.

Fitted Q-Evaluation (FQE), one of the many OPE techniques, is a creative and promising method for estimating a policy's performance off-policy. Counterfactual reasoning and causal inference serve as sources of inspiration for FQE. FQE can reason counterfactually because it is based on a causal inference framework. It calculates the expected value of a policy's performance while accounting for the possible results of making different decisions in the past. FQE simulates potential courses of action under the policy under review in order to carry out counterfactual rollouts. It calculates the expected return for every action by taking these "what-if" scenarios into account. FQE uses a model-free methodology to estimate value functions, frequently utilizing machine learning techniques like function approximation or deep Q-networks (DQNs). FQE strives for consistency in its estimations, guaranteeing the stability and dependability of policy evaluation. This is essential for credible policy analysis.

Offline Policy Evaluation (OPE) has advanced significantly with the introduction of Fitted Q-Evaluation (FQE). Through the application of counterfactual reasoning and causal inference principles, FQE allows for a thorough understanding of how learned policies would have functioned in the past. Its capacity to estimate expected returns and simulate alternate courses of action offers important insights into risk analysis, comparison, and policy assessment. FQE and OPE techniques together have the potential to produce more intelligent and resilient agents that can learn from and adapt to past data as reinforcement learning research progresses.

4.5 Online Reinforcement Learning Model

Following figure provides the overview of the proposed model training process.

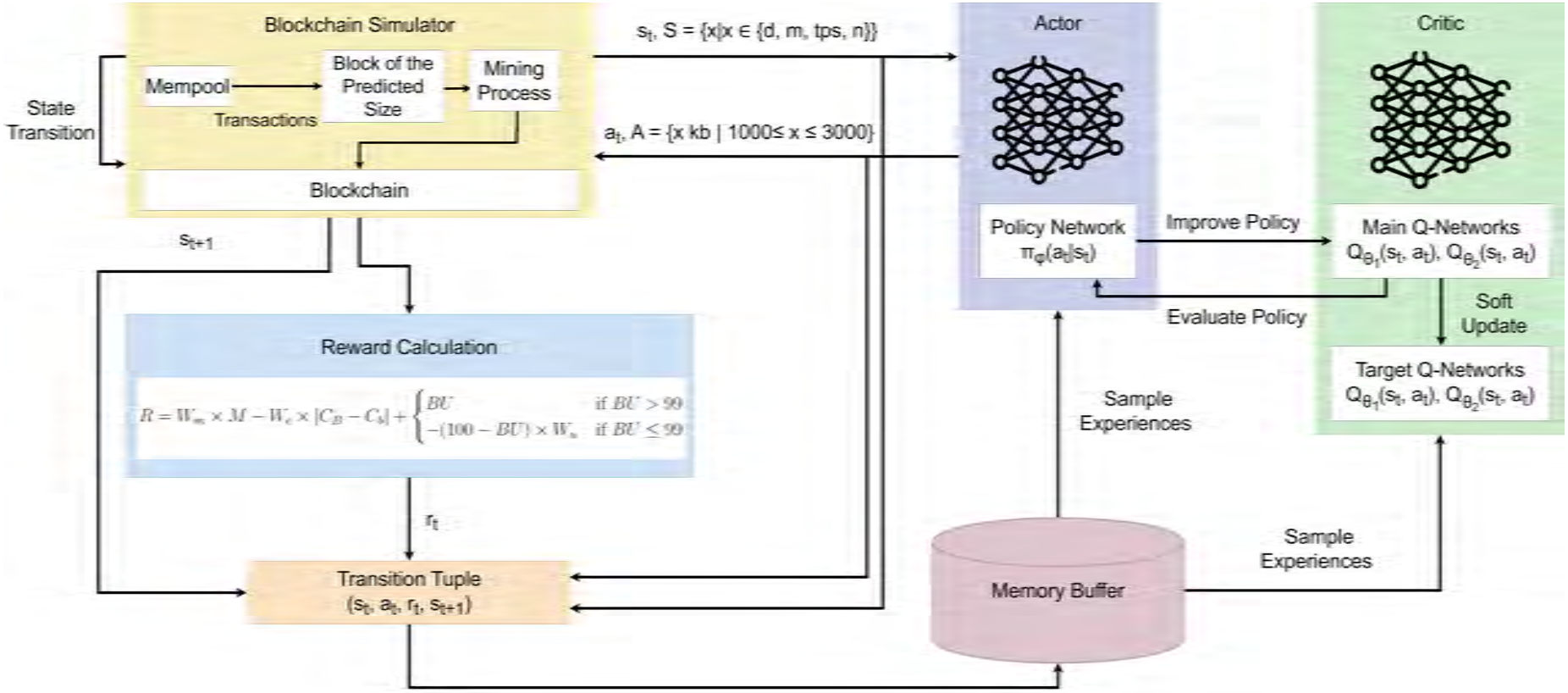

Figure 4.4: Overview of Online Reinforcement Learning Training using SAC Model

In the field of reinforcement learning, online training is a paradigm-shifting technique that allows intelligent agents to interact with dynamic environments and learn and adapt in real-time. In this part of the paper, we investigate the creative use of virtual instruction in the intricate and dynamic field of blockchain technology. To train online reinforcement learning models, such as Soft Actor-Critic (SAC), Twin Delayed Deep Deterministic Policy Gradients (TD3), Truncated Quantile Critic (TQC), we created a simulated environment that mimics the dynamic behavior of a blockchain network using OpenAI Gym library. Additionally, we used the Imitation library and Inverse Reinforcement Learning (IRL) techniques to reward shape and direct agents toward human-like behavior in the blockchain ecosystem simulation.

4.5.1 Blockchain Simulator Creation

The world of finance and decentralized systems has changed as a result of blockchain technology, which is best illustrated by Bitcoin. Because For our study, we built a simulated setting that depicts the complexities of creating a block in the blockchain networks. We replicated the mining, transaction processing, and mempool within this environment. Our agents used this simulated environment as an experimental playground to investigate and choose the best possible action for every situation.

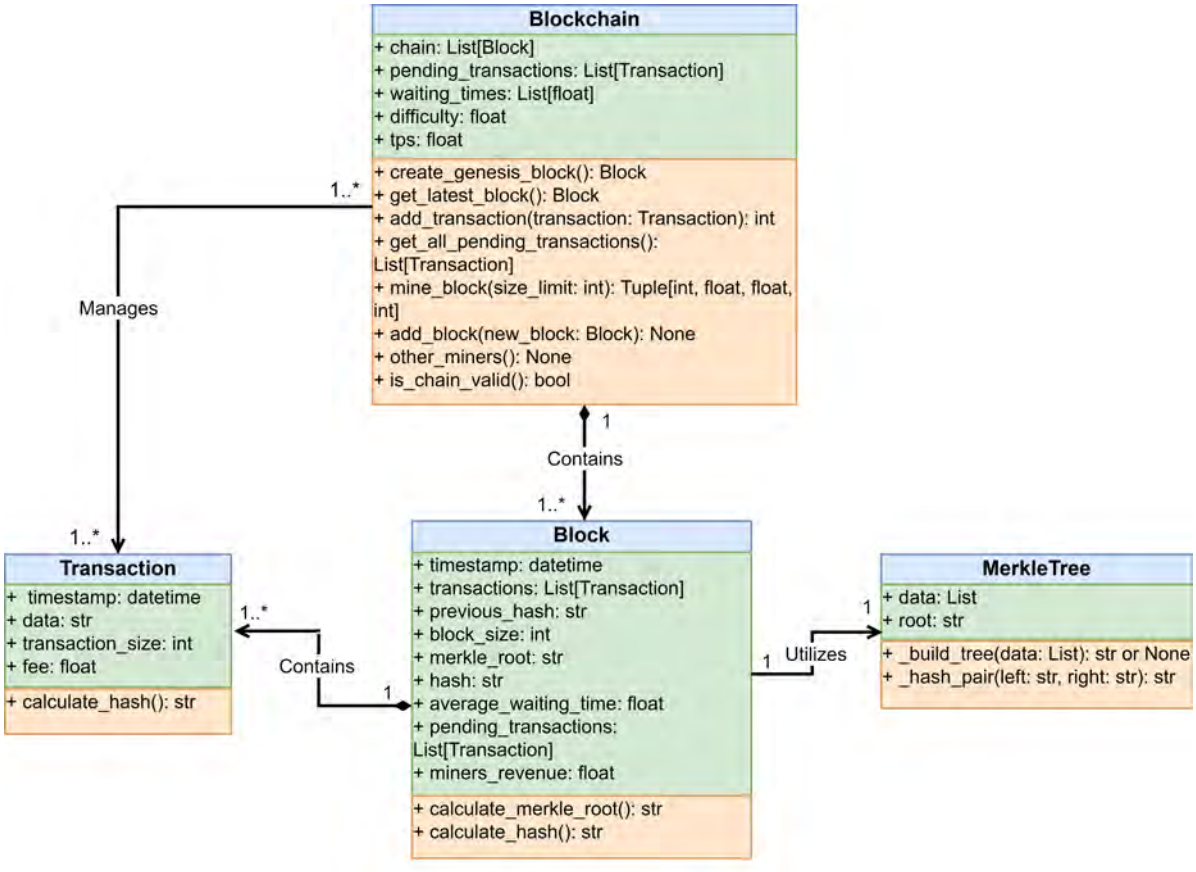

Figure 4.5: Class Diagram of Bitcoin Blockchain Simulator

Description of the classes of the simulator are as follows:

MerkleTree

Merkle Tree class is essential to simulating the fundamental architecture of blockchain technology. A key element of the blockchain, Merkle Trees offer a safe and effective means of storing and confirming the accuracy of transactions within a block. The Merkle Tree class is the foundation of the ledger in our simulator, enabling us to build a safe and tamper-evident data structure. This class combines individual transaction hashes to create a single root hash that summarizes the contents of the block, allowing for the efficient construction of Merkle Trees for every block. The Merkle Tree class ensures the security and reliability of the blockchain by expediting the verification of individual transactions and making it easier to arrange data within blocks.

Transaction

One essential part of our simulator that captures and controls the essential characteristics of a blockchain transaction is the Transaction class. The timestamp, data, and transaction size are its three main properties. Each is essential to defining and capturing the specifics of a transaction.

- **Timestamp**: A crucial feature that records the exact moment a transaction is added to the blockchain is the timestamp property. It is stored in Unix time format, which uses the seconds that have passed since January 1, 1970, to represent time. The blockchain's chronological ordering and consistency of transactions are guaranteed by the timestamp. Tracking the order of transactions and creating a comprehensive historical record of all blockchain activity depend on this temporal information.

- **Data**: The core of the transaction is located in the data property. It contains the main payload, or the data related to the transaction. Depending on the particular blockchain use case, the type of data can change dramatically. A cryptocurrency blockchain, for example, might have a data field that contains information about the amount of money being transferred, any additional notes or messages that go along with the transaction, or even complex instructions for smart contracts. The Transaction class is adaptable to a broad range of blockchain applications because of its flexibility in handling data.

- **Transaction Size**: The transaction size property indicates how much space the transaction takes up in the blockchain's storage. It measures the amount of data in the transaction and quantifies it in bytes. In order to maximize resource management and storage efficiency within the blockchain network, it is essential to comprehend transaction size. By effectively controlling transaction sizes, the blockchain can minimize processing and storage costs while scaling to accommodate an increasing number of transactions.

Block Class

A key element in our simulator that imitates the composition and operation of a blockchain block is the Block class. This class includes a number of attributes that capture important details and features of a block in the blockchain.

- **Block Creation Timestamp**: The precise moment a block is created and added to the blockchain is recorded by the timestamp property in the Block class. Blocks are arranged consistently and chronologically thanks to the usage of the Unix time format.

- **Transactions Added to the Block**: A list of the transactions that are a part of the block is stored in this property. The fundamental units of a blockchain are transactions, and this property keeps track of every transaction that has been approved for inclusion in the block.

- **Hash of Previous Block**: Every block contains the hash of the previous block in order to preserve the chain's integrity and continuity. Blocks are linked in a tamper-resistant and sequential manner thanks to this property.

- **Block Size (Dynamic)**: The block size property indicates the block's dynamic size, which varies based on the quantity and kind of transactions it contains. Block size, which is expressed in bytes, is a crucial component in maximizing the scalability and efficiency of the blockchain.

- **Merkle Root**: A cryptographic hash created from the block's transaction data is known as the Merkle root. It is used to confirm the integrity of the included transactions and acts as a summary of every transaction in the block.

- **Hash**: The hash property holds the block-wide cryptographic hash, which includes the timestamp, Merkle root, hash of the previous block, and other block-specific data. This hash is essential to blockchain security as it provides a unique identification for the block.

- **Average wait time for transactions in the block**: The average waiting time of the block's transactions is determined by this property. It gauges how long it took for a transaction to be confirmed in a block after it was first started, offering information about how efficiently transactions are processed.

- **Block Miner's Revenue**: The total amount of money earned by the miner who effectively mined the block is represented by the block miner's revenue property. It comprises the fixed block reward, which is normally 6.25 BTC (Bitcoin) for every block in the Bitcoin network, as well as transaction fees

paid by users to include their transactions in the block. One of the main incentives for miners to protect the blockchain network is their income.

Blockchain Class

Our simulator's Blockchain class is the central component that mimics the structure and operation of a block inside the blockchain network. This class includes a number of properties and functions intended to control the block creation functionality.

- **Chain**: A list containing every block in the blockchain is the chain attribute. The blockchain is sequential and unchangeable because each block is connected to the previous one via its hash value.

- **Pending Transactions**: The blockchain's mempool is represented by the pending transactions attribute. The fee-to-size ratio of each transaction determines its priority in the min-heap, or min priority queue, that is used to implement it. The transactions with the highest fee per byte can be efficiently retrieved thanks to the min-heap.

- **Waiting Times**: Each transaction's confirmation time is preserved by the blockchain's waiting times attribute. By measuring the amount of time it takes for a transaction to be included in a block, it gives information about how quickly transactions are confirmed on the network.

- **Difficulty**: The blockchain network's current computational difficulty is indicated by the difficulty attribute. It is a crucial setting in the proof-of-work consensus process that regulates block creation speed and upholds network security.

- **Transactions Per Second (TPS)**: The TPS attribute shows how well the blockchain network can process transactions. It provides a performance metric for assessing the efficiency of blockchain technology by counting the number of transactions the network can process in a single second.

- **Block Genesis (Method)**: The blockchain's first block was made using the genesis block technique. The first block, also known as the "genesis block," is created using this method and forms the basis of the blockchain. The genesis block, which is normally one kilobyte in size, has all the information needed to get the blockchain started.

- **Get_Latest _Block (Method)**: The most recent block to be added to the blockchain can be obtained using the get latest block method. It gives users access to the blockchain's most recent state, complete with transactions, hashes, and other features.

- **Add Transaction (Method)**: Transactions (pending transactions) can be added to the mempool using the add transaction method. After being added to the min-heap and ranked according to their fee-to-size ratio, transactions are prepared for inclusion in upcoming blocks.

- **Get All Pending Transactions (Method)**: This method retrieves all pending transactions from the mempool. This gives the blockchain network's users visibility into every transaction that is pending confirmation.

- **Mine Block (Method)**: Block creation is handled by the mine block method. To create a new block, it adds transactions from the mempool and accepts a size limit as input. The procedure starts the mining process to find the correct nonce and makes sure that the block size limit is not exceeded.

- **Add Block (Method)**: Newly mined blocks are added to the blockchain using the add block method. By adding the recently formed block to the chain, this technique updates the blockchain's state and keeps it continuous.

- **Other Miners (Method)**: The shared mempool can be accessed and used by other miners in the blockchain network through the use of the other miners method. The decentralization and competition that are intrinsic to blockchain networks are encouraged by miners' ability to retrieve pending transactions from the mempool and incorporate them into their own blocks.

The Blockchain class allows our simulator to precisely model and analyze blockchain's behaviour, such as transaction processing, mining, and network management. It does this by encapsulating the fundamental parts of a blockchain system which is block creation and size determination.

4.5.2 Simulation

The way the simulator works is by simulating the basic procedures and exchanges of a block creation in blockchain network. It makes use of the Blockchain class, which contains all of the essential blockchain features, such as block and transaction management, mempool management, mining, and confirmation timeframes.

The Blockchain class instance is first created. This instance is the basis for all subsequent interactions and represents the blockchain network. The Transaction class is to create transaction instances, which are then added to the network. Every transaction has three distinct characteristics: size, fee, and data. The fundamental characteristics of actual blockchain transactions are reflected in these properties.

Once the transactions are created, they can be added to the blockchain's mempool. To add newly created transactions to the mempool, the add_transaction method is used that the Blockchain class provides. For pending transactions, the mempool

serves as a waiting area. The network's miners can view the available transactions in the shared mempool and choose which ones to include in their blocks. Within the mempool, transactions are ranked according to their fee-to-size ratio. Block creation in the blockchain network is largely dependent on miners. To get the transactions they want to include in their blocks, they can access the mempool. The Blockchain class's mine block method is used by miners to start the block creation process. To guarantee that blocks don't go over the network's capacity, this technique places a cap on the expected block size. Until the designated size limit is reached, miners keep adding transactions from the mempool to their block. In order to maximize miners' revenue, miners must carefully choose which transactions to accept in this process, which is similar to real-world mining activities. To mimic this scenario, we created the mempool which is maintained with the help of a Heapq which follows a min heap which calculates the transaction fee divided by the transaction size and sorts it from the smaller to the largest. The miner's block is generated and added to the blockchain when the block size limit is reached. By adding the new block to the chain and updating it within the Blockchain class, the simulator replicates this procedure. This guarantees the blockchain's ongoing existence. The simulator determines the waiting time or confirmation time for each transaction as blocks of transactions are added. This measure shows how long it took for a transaction to be verified inside the blockchain network and added to a block. Transactions are eliminated from the mempool once they are successfully added to a block. This action mimics how transactions actually work in the real world, where they are confirmed and then removed from mempool. The creation of blocks, mempool management, transaction selection, mining operations, and confirmation time computations are all precisely modeled by the simulator by following these procedures. This method makes it possible to thoroughly examine and test blockchain operations, offering insightful knowledge about transaction prioritization, network performance, and other crucial blockchain aspects.

4.5.3 Simulator Environment Creation in OpenAI Gym

The successful application of Reinforcement Learning (RL) in the case of dynamic block size modification for the Bitcoin blockchain requires the creation of an efficient and representative Gym environment. Because it includes the essential elements and behaviors needed for reinforcement learning, this environment effectively simulates the operation of the block creation process in blockchain. The following describes the general simulation structure, the step function, the observation space, the action space, and the design and operation of the gym environment.

- **Observation Space :** Four essential characteristics that capture the current state of the Bitcoin blockchain system at any given time define the observation space. Among these attributes are:

    1. **Difficulty (d):** How hard the cryptographic puzzle is for miners to solve in order to add a new block to the blockchain.
    2. **Mempool Count(m)**: The quantity of pending transactions in the memory pool that are awaiting inclusion in the subsequent block is known as the mempool count.

3. **Transaction Per Second (tps):** The speed at which updates are sent out to the network for transactions.
4. **N Transactions Per Block (n):** The mean quantity of transactions found in every block that is mined.

- **Action Space :** We have a continuous action space ranging from 1MB to 3MB. From this range our agent will be predicting the block size which is the single, crucial action the agent has to take given an observation. In the reinforcement learning setting, agent's decision on predicting the ideal block size is a crucial aspect of how well the blockchain network functions.

- **Reward Calculation :** After an action is taken, the reward is calculated using a particular formula designed to reduce confirmation time and increase miners' earnings while maintaining a desired percentage of the block utilization. By doing this, it is ensured that the RL agent will learn to optimize block sizes with the dual objectives of maximizing miner profitability and decreasing transaction confirmation times. The reward is calculated according to this equation:

$$R = W_m \times M - W_c \times |C_B - C_b| + \begin{cases} BU & \text{if } BU > 99 \\ -(100 - BU) \times W_u & \text{if } BU \leq 99 \end{cases} \quad (4.4)$$

  where M denotes the miner's revenue of the block, W_m is the weight coefficient of miner's revenue, M. C_B denotes the average confirmation time of all transactions added in the blockchain and C_b denotes the average confirmation time of all transactions added in the block, W_c is the weight coefficient of average confirmation time, C. Block utilization percentage of the block is denoted by BU and W_u is the weight coefficient of BU.

- **Next Observation**: Following a block's mining, the environment uses the "next observation" method to determine the next observation, updating the features according to the blockchain's current state.

- **Episode**: Each training episode has 200 timesteps, which correspond to the RL agent's sequential decision-making process. The environment resets after the 200 timesteps are finished, giving the agent several episodes to learn and adjust.

Step Function: The environment's dynamics are determined by the step function. The RL agent acts in each step, giving an estimated block size. This block size is then used by the blockchain simulator to mine a block. This procedure mimics how miners choose block sizes in the real world depending on demand and network conditions.

By building this Gym environment, the basic framework for teaching RL agents to dynamically modify block sizes in the Bitcoin blockchain is established. This environment offers a realistic simulation for efficient reinforcement learning (RL) training by incorporating important blockchain features into the observation space, defining a single but powerful action, and structuring the step function and reward calculation to mimic real-world blockchain dynamics.

4.5.4 Online Reinforcement Learning Model Training

Online Reinforcement Learning gains a dynamic and adaptable dimension with learning. As agents engage with the environment in online RL, they constantly pick up new information and adjust their policies in real time. In contrast, learning takes place offline on pre-collected data in batch reinforcement learning. Online reinforcement learning systems work well in situations where decisions need to be made quickly and data is continuously flowing. For our thesis, the main three models we used are Soft Actor-Critic(SAC), Truncated Quantile Critics (TQC) and Twin Delayed Deep Deterministic(TD3) algorithms. The parameters have been tuned of each model as well as the reward function cofficients for better results before training using Optuna Library.

Soft Actor Critic (SAC)

Based on the maximum entropy reinforcement learning framework, Soft Actor Critic, or SAC, is an off-policy actor-critic deep reinforcement learning algorithm [11]. The actor's goals in this framework are to maximize entropy and expected reward simultaneously. In other words, to complete the task successfully and as haphazardly as possible. Previous Q-learning deep reinforcement learning techniques have been developed using this framework. SAC is a stable stochastic actor-critic formulation combined with off-policy updates.

There are several benefits to the SAC goal. Initially, the policy is designed to encourage broader exploration while abandoning paths that are obviously unpromising. Secondly, various modes of near-optimal behavior can be captured by the policy. When there are several actions that appear equally appealing in a problem setting, the policy will assign the same probability mass to each action.

Soft Actor Critic (SAC) algorithm bridges the gap between stochastic policy optimization and DDPG-style approaches by optimizing a stochastic policy in an off-policy manner. Although it was released almost simultaneously with TD3, it is not a direct replacement. However, it does use the clipped double-Q trick, and because SAC policies are inherently stochastic, it also ends up benefiting from target policy smoothing.

A key component of SAC is regularization of entropy. The goal of policy training is to optimize the trade-off between expected return and policy entropy, which is a measure of randomness [16]. This is closely related to the exploration-exploitation trade-off, whereby higher entropy leads to greater exploration, which can expedite subsequent learning.

Algorithm 2: Soft Actor-Critic

Input: θ_1, θ_2, ϕ. ▷Initial parameters

$\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$. ▷Initialize target network weights

$D \leftarrow \emptyset$. ▷Initialize an empty replay pool

Initialize environment E

State : $S_t = x | x \in d, m, tps, n$

Action : $a_t = (1, 3)$

Reward :

$$R = W_m \times M - W_c \times |C_B - C_b| + \begin{cases} BU & \text{if } BU > 99 \\ -(100 - BU) \times W_u & \text{if } BU \leq 99 \end{cases} \quad (4.5)$$

for *each iteration* **do**

  for *each environment step* **do**

    $a_t \sim \pi_\phi(a_t|s_t)$. ▷Sample action from the policy

    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. ▷Sample transition from the environment

    $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$. ▷Store the transition in the replay pool

  for *each gradient step* **do**

    $\theta_i \leftarrow \theta_i - \lambda Q \nabla_{\theta_i} \hat{J}_Q(\theta_i)$ for $i \in \{1, 2\}$. ▷Update the Q-function parameters

    $\phi \leftarrow \phi - \lambda \pi \nabla_\phi \hat{J}_\pi(\phi)$. ▷Update policy weights

    $\alpha \leftarrow \alpha - \lambda \nabla_\alpha \hat{J}(\alpha)$. ▷Adjust temperature

    $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$. ▷Update target network weights

Output: θ_1, θ_2, ϕ. ▷Optimized parameters

Twin Delayed Deep Deterministic (TD3)

TD3 extends the reinforcement learning algorithm DDPG[11], adding a few tweaks to address overestimation bias in the value function. Specifically, it makes use of target policy smoothing, delayed update of target and policy networks, and clipped double Q-learning. Although DDPG can occasionally achieve very high performance, it is often fragile when it comes to hyperparameters and other types of tuning. A typical DDPG failure mode is when the learned Q-function starts to significantly overestimate Q-values. This leads to policy breaking because it takes advantage of the Q-function's errors [20]. In order to solve this problem, the Twin Delayed DDPG (TD3) algorithm introduces three crucial tricks:

- **First Trick:** Double-Q Learning with Clips. The term "twin" refers to the fact that TD3 learns two Q-functions rather than one, and the Bellman error loss functions use the lesser of the two Q-values as the targets.
- **Second Trick:** Policy updates that are "delayed." Compared to the Q-function, TD3 changes the policy (and target networks) less frequently. One policy update is suggested by the paper for each of the two Q-function updates.

- **Third Trick:** Smoothing of Target Policies. By adding noise to the target action, TD3 smoothes out Q along changes in action, making it more difficult for the policy to exploit Q-function errors.

When combined, these three tips produce performance that is significantly better than baseline DDPG.

Truncated Quantile Critic (TQC)

The novel model known as Truncated Quantile Critic (TQC) increases the potential of conventional distributional reinforcement learning algorithms. TQC is an innovative method for dealing with variance in the estimation of value distributions and was developed to improve the effectiveness and resilience of RL systems. As such, it is well-suited for applications where accurate risk-aware decision making is critical[30].

TQC is an extension of distributional RL, which aims to estimate the whole return distribution rather than just the expected values. This offers a more comprehensive depiction of risk and uncertainty in the learning process. To estimate the quantiles of the return distribution, TQC uses quantile regression. This method offers a more nuanced view of risk by enabling the model to capture a more thorough understanding of the variability in possible outcomes. The mechanism to focus on a restricted range of quantiles introduces the "truncated" aspect of TQC. This enhances training stability and enables the model to focus on the most pertinent portions of the distribution, which is especially helpful in situations where extreme values could distort learning. Improved robustness against uncertainties is a result of TQC's emphasis on truncated quantiles. Focusing on particular segments of the distribution reduces the model's sensitivity to exceptions and extreme values, resulting in learning that is more steady and dependable. Because of its distributional structure, TQC is naturally suited for risk-aware making choices. Because TQC can estimate entire distributions, agents can take into account a wider range of potential outcomes, which makes it especially useful in applications where risk management and understanding are crucial [22]. TQC is adaptable in a variety of contexts due to its resilience and risk awareness. TQC is a potent tool for solving problems in the real world because of its capacity to handle uncertainty. Applications include finance, portfolio management, robotics, and autonomous systems.

4.6 Reward Shaping using Inverse Reinforcement Learning

For our purposes, we took two separate approaches when generating the reward function for the online RL models. The first approach is simple trial and error formulation of the reward function where we started with an arbitrary reward function consisting of the parameters which we would work with as our objectives. Then through several trials we managed to generate an optimal reward function which produced values which were highly appreciated by our judgment. The second method is to generate a reward using inverse RL, where we provide the steps taken and the algorithm picks out the reward for us, which is simply the opposite of what

the rl models do. And to do that we used Generative Adversarial Imitation Learning(GAIL).

Two distinct methods in the field of reinforcement learning—Inverse Reinforcement Learning (IRL) and Generative Adversarial Imitation Learning (GAIL)—address the crucial problem of determining the best policies based on expert demonstrations. IRL and GAIL, while having different formulations and approaches, are fundamentally similar in that they both use a specialist trajectories as a means of guiding learning.

Finding a latent reward function which most closely accounts for an expert agent's observed behavior is the fundamental goal of IRL. Conventional reinforcement learning involves teaching agents to maximize a predefined reward signal. Nevertheless, creating an explicit reward function which reflects the subtleties of desired behavior may be a difficult undertaking in many real-world situations. To address this problem, IRL approaches it from the opposite direction: given expert demonstrations, it aims to deduce the underlying reward structure which drives the observed behavior.Typically, the procedure entails modeling the trajectory distribution of the expert and bringing a reward function so that the agent's expected return under this new reward is consistent with the expert behavior that has been observed. Techniques such as Maximum Entropy IRL encourage the model to capture all of the ways in which the expert could have acted in the given environment by taking into account both the entropy of the agent's policy and the observed trajectories. In order to mimic expert behavior, GAIL, on the other hand, uses a generative adversarial framework and ideas from generative adversarial networks (GANs). In GAIL, a discriminator is simultaneously trained to distinguish between expert and generated trajectories, while a policy generator strives to generate behavior that is identical from that of an expert. The discriminator and generator engage in adversarial play to produce a fine-tuned policy that closely resembles the expert's behavior. The discriminator plays a critical role in influencing how people learn. Through continuous refinement of its expert and generated trajectory discrimination, it directs the generator to produce behavior consistent with its demonstrated expertise. Without explicitly depending on a custom reward function, this adversarial training framework reflects the fundamental basis of the expert's behavior quite well.

Although IRL and GAIL utilize distinct approaches to leverage expert demonstrations, they are united by the utilization of expert trajectories as an invaluable data source to direct the learning process. The goal of IRL is to determine the reward function that provides an implicit metric for the best course of action and rationalizes the observed behavior. GAIL, on the other hand, uses adversarial training with a direct focus on policy optimization, utilizing the expert trajectories to improve the behavior of the agent.

When one considers that both IRL and GAIL place a strong emphasis on learning from experience in complex and uncertain environments, they can be seen to work well together. Expert trajectories act as a link between the intended policy—a refined policy (GAIL) or a reconstructed reward function (IRL)—and the observed behavior. This intersection highlights how flexible expert demonstrations are as a

valuable information source, enabling agents to adjust and perform well in tasks where reward shaping is difficult or impractical.

4.6.1 Generative Adversarial Imitation Learning (GAIL)

Using the ideas of adversarial training, the robust framework known as Generative Adversarial Imitation Learning (GAIL) proposed by [6], enables agents to learn from expert demonstrations and produce behavior that closely resembles the demonstrated expertise. In order to enable policy development from expert trajectories, generative adversarial networks (GANs) and imitation learning components are combined in a technique called generative adversarial imitation learning. The main goal is to teach a policy to produce behavior that, when evaluated by a discriminator, is identical to that of an expert. GAIL facilitates effective skill transfer by presenting the process of learning as a game in which the policy and discriminator compete. This promotes the policy to imitate the expert's behavior.

Expert trajectories (τE) sampled from an expert policy (πE) are initialized before GAIL starts. The learning agent uses these trajectories as a model to emulate. GAIL runs in a loop that is iterative. The algorithm updates the discriminator parameters (wi) by combining real and generated trajectories, sampling agent trajectories (τi) from the current policy ($\pi\theta$i) in each iteration (indexed by i). Training the discriminator to discern between expert and agent-generated trajectories is the aim. A gradient ascent step is utilized to update the discriminator. While minimizing the log likelihood of the agent-generated trajectories, the goal is to maximize the log likelihood of the expert trajectories. In order to improve the discriminator's ability to discern between expert and agent behavior, an adversarial training step is implemented. After that, Trust Region Policy Optimization (TRPO) is used to update the policy. The log probability that the updated discriminator assigned to the trajectories created by the agent serves as the basis for this update's cost function. Stability in the learning process is maintained by the KL-constrained natural gradient step, which makes sure that the policy update does not stray too far from the current policy. GAIL's ability to learn a policy that mimics expert behavior without the need for explicit reward signals is its primary innovation. This is especially helpful in situations where it is difficult or impractical to design a reward function. GAIL efficiently modifies the reward landscape by coordinating the agent's actions with those of an expert, directing the agent toward desired results.

Algorithm 3: Generative Adversarial Imitation Learning

Input: Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
for $i = 0, 1, 2, \ldots$ **do**
  Sample trajectories $\tau_i \sim \pi_{\theta_i}$
  Update the discriminator parameters from w_i to w_{i+1} with the gradient
  $\hat{E}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{E}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))]$
  Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$.
  Specifically, take a KL-constrained natural gradient step with
  $\hat{E}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta)$,
  where $Q(\bar{s}, \bar{a}) = \hat{E}_{\tau_i}[\log(D_{w_{i+1}}(s,a))|s_0 = \bar{s}, a_0 = \bar{a}]$
end for

Chapter 5

Result Analysis

A comprehensive analysis of the study's results was performed to assess the findings corresponding with the research objectives and research question.The collected and presented data were meticulously scrutinized to identify patterns, trends, and correlations.The discussion delves into the consequences of the discoveries and their compatibility with existing theoretical frameworks and literature. The study's scope is refined by identifying its limits and possible sources of bias.The analysis of research findings is essential in evaluating the broader implications of the bitcoin blockchain network, advancing our comprehension and finding solution of the fixed block size problem.

Evaluation Matrices

Before diving to the result analysis, we will look into some evaluation or performance matrices based on which our result has been meticulously established. For our baseline model Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) values are used to evaluate the model. To evaluate offline RL we have used one of the off policy evaluation's Fitted Q Evaluation (FQE). Lastly to evaluate online RL results, mainly to evaluate our reward function, we have find the mean reward to evaluate the algorithm's performance.

Mean Absolute Error (MAE): The MAE is calculated by taking the average of the absolute differences between the projected values and the actual values. It quantifies the average magnitude of errors, regardless of their direction.

$$MAE = \frac{(\sum_{n=1}^{N} |\hat{r}_n - r_n|)}{N} \tag{5.1}$$

Mean Squared Error (MSE): The MSE is calculated by taking the average of the squared discrepancies between the projected values and the actual values. The squaring operation magnifies the effect of larger errors.

$$MSE = \frac{(\sum_{n=1}^{N} (\hat{r}_n - r_n)^2)}{N} \tag{5.2}$$

Root Mean Squared Error (RMSE): The RMSE The is calculated by taking the square root of the Mean Squared Error (MSE). The independent variable is

measured in the same unit as the dependent variable and quantifies the average size of mistakes on a scale that is comparable to the original data.

$$RMSE = \sqrt{\frac{(\sum_{n=1}^{N}(\hat{r}_n - r_n)^2)}{N}} \tag{5.3}$$

where $\hat{r}_n$ represents the predicted rating, r_n represents the true rating in the testing dataset, and N is the number of rating prediction pairs between the testing data and prediction result.

These metrics are frequently employed in regression issues to assess the degree of alignment between a model's predictions and the actual values. Lower values of MAE (Mean Absolute Error), MSE (Mean Squared Error), and RMSE (Root Mean Squared Error) signify superior performance of the model. However, if the error values are very close it indicates underfitting. On the other hand, if the error values are high it means overfitting. The selection of the appropriate metric may rely on the distinct attributes of the data and the current challenge.
Fitted Q Evaluation (FQE) : The FQE algorithm utilizes iterative regression to approximate Q functions and ultimately predicts the target policy value by integrating an estimator of the Q function [33].
Consider F as a set of function approximators. The FQE (Fitted Q Evaluation) framework can be concisely described as:
We let $\hat{Q}_{H+1} = 0$, and for $h = H, H-1, \ldots, 1$, we iteratively solve

$$\hat{Q}_h = \arg\min_{f \in F}\{\frac{1}{N}\sum_{n=1}^{N}[f(s_n, a_n) - y_n]^2 + \lambda\rho(f)\} \tag{5.4}$$

where $y_n = r(s_n, a_n) + \int_A \hat{Q}_{h+1}(s_{n+1}, a)\pi(a|s_{n+1})\,da$, $\lambda > 0$, and $\rho(\cdot)$ is a proper regularizer.

In offline reinforcement learning (RL), the FQE fit function observes and evaluates the agent's exploration and exploitation policies [33]. The observation can be expressed as values indicating loss. If the loss numbers drop, it indicates that the model is performing effectively. Conversely, an increase in loss values during evaluation using the FQE (Fitted Q Evaluation) fit function generally suggests a decline in the model's performance.

Mean of Rewards: Run online reinforcement learning algorithms for a specified number of episodes and return the mean and standard deviation to evaluate their performance.The agent's cumulative episode rewards are the returns. The mean return indicates the agent's average performance, while the standard deviation measures variability. By analyzing the mean reward graphs we have evaluated the online RL results.

$$\text{Mean of Rewards} = \overline{\text{Reward of 100 episodes}} \tag{5.5}$$

5.1 Baseline Model: Long Short Term Memory

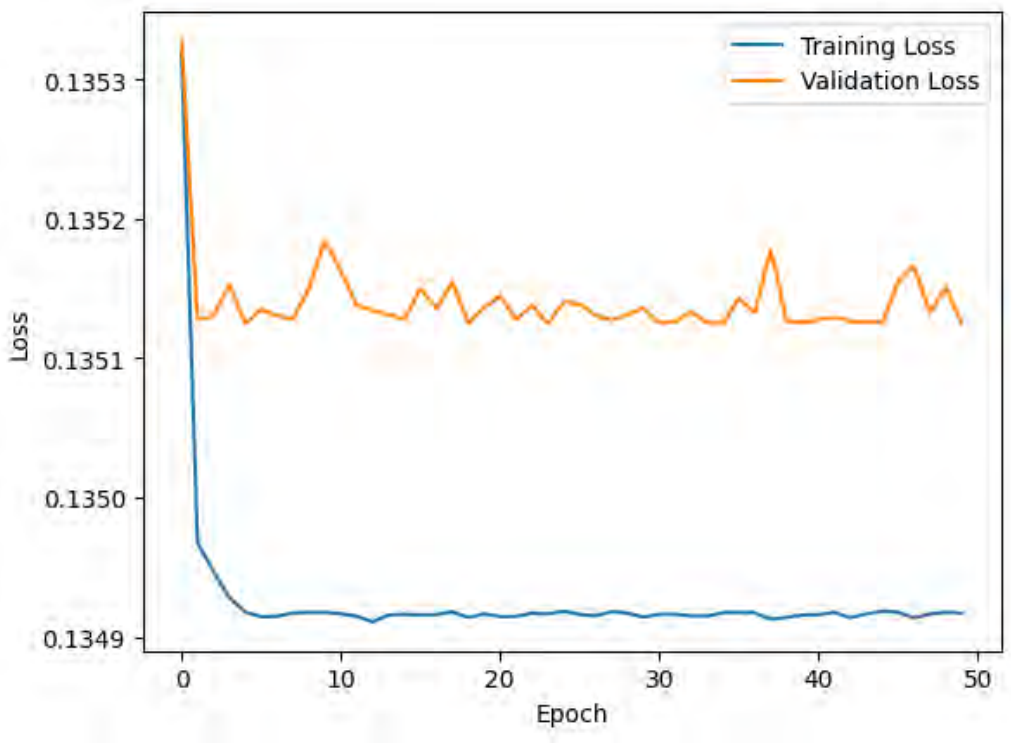

Figure 5.1: Training and Validation Loss of LSTM

In Figure 5.1, we can see throughout each epoch, validation loss is higher than training loss, which translates to the fact that the model is underfitting. This occurs when the model underfits the data and generates poor values with greater errors. This is why we can not rely on using LSTM as a solution to our problem. As shown

Table 5.1: Error Metrics for LSTM Model

Phase	**RMSE**	**MSE**	**MAE**
Training	0.671	0.450	0.593
Testing	0.672	0.451	0.594

in Table 5.1, the Root Mean Squared Error(RMSE), Mean Squared Error(MSE) and Mean Absolute Error(MAE) values are very similar which means that the model is unable to differentiate between the train data and the test data. So even if the model provides a prediction, it will not be an optimal prediction and after training while testing the model is unable to produce an approximate prediction. This is another major reason for which we are unable to use LSTM to solve our problem.

5.2 Offline Reinforcement Learning Model

5.2.1 Training Phase

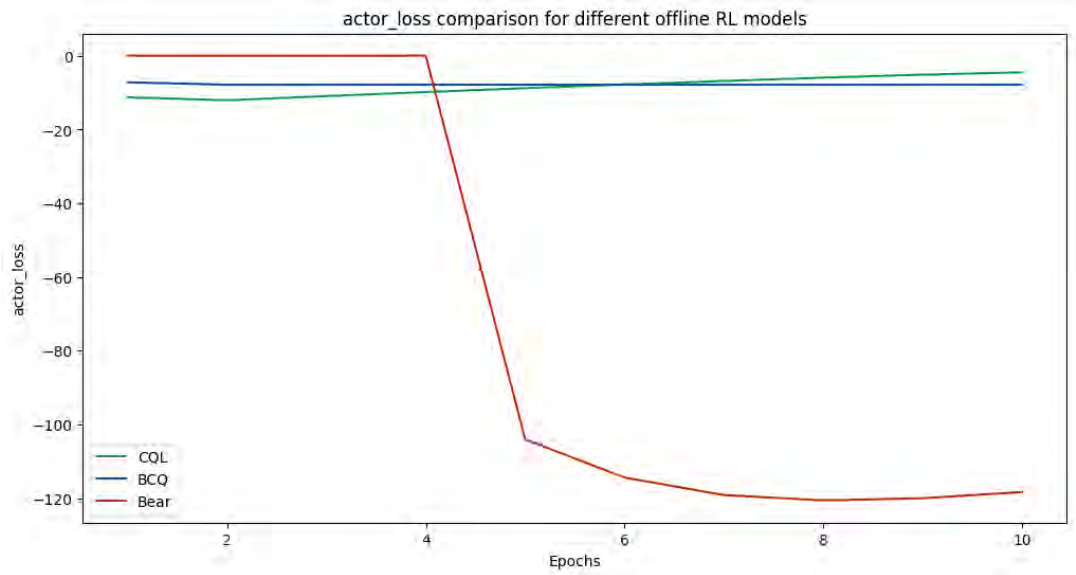

Figure 5.2: Actor Loss of CQL, BCQ and BEAR in Training Phase

In Figure 5.2, we have the actor loss comparison between the three offline models used BEAR, BCQ and CQL. Actor loss signifies the policy on which the agent is working based upon. The higher the actor loss, the worse prediction it provides resulting in significantly low rewards. So the optimal policy will provide an actor loss graph which will follow the downward trend. Among the algorithms we used, BEAR provides the best results in terms of actor loss.

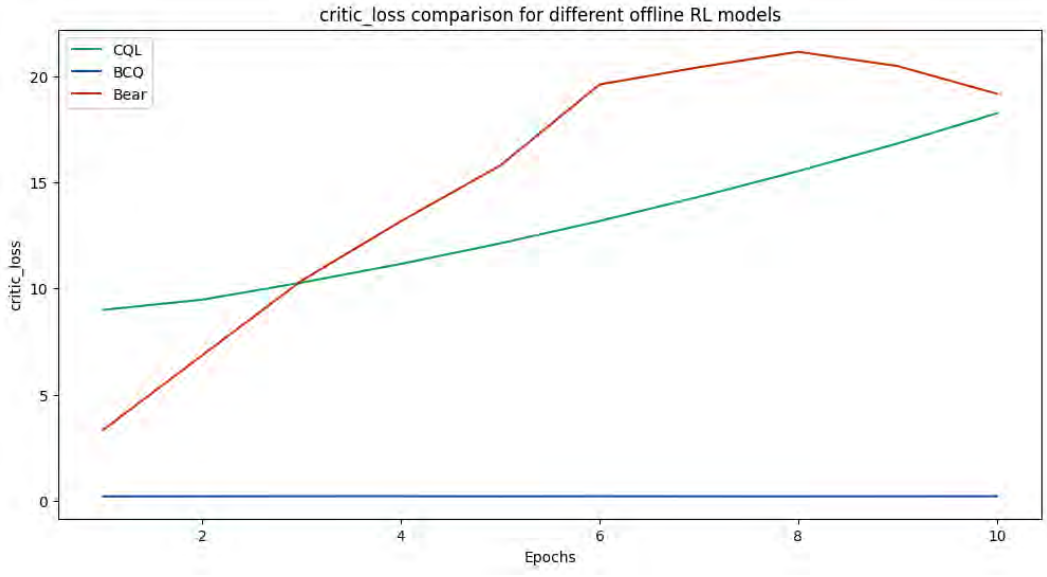

Figure 5.3: Critic Loss of CQL, BCQ and BEAR in Training Phase

We can see the disparity between the three offline models used in terms of Critic loss in Figure 5.3, which depends on the value of the reward. Increased critic loss refers to the fact that even after exploration the models are unable to complete exploitation properly which means the agents are not able to make optimal decisions, so instead of getting higher reward, they are being pulled down by penalties. Less critic loss indicates a much better exploration and exploitation by the model. In the graph we can see that BCQ is providing the best results in term of critic loss.

Figure 5.4 depicts the difference in terms of alpha loss between BEAR and CQL. The adjustment made to guarantee conservative estimates and avoid overestimating

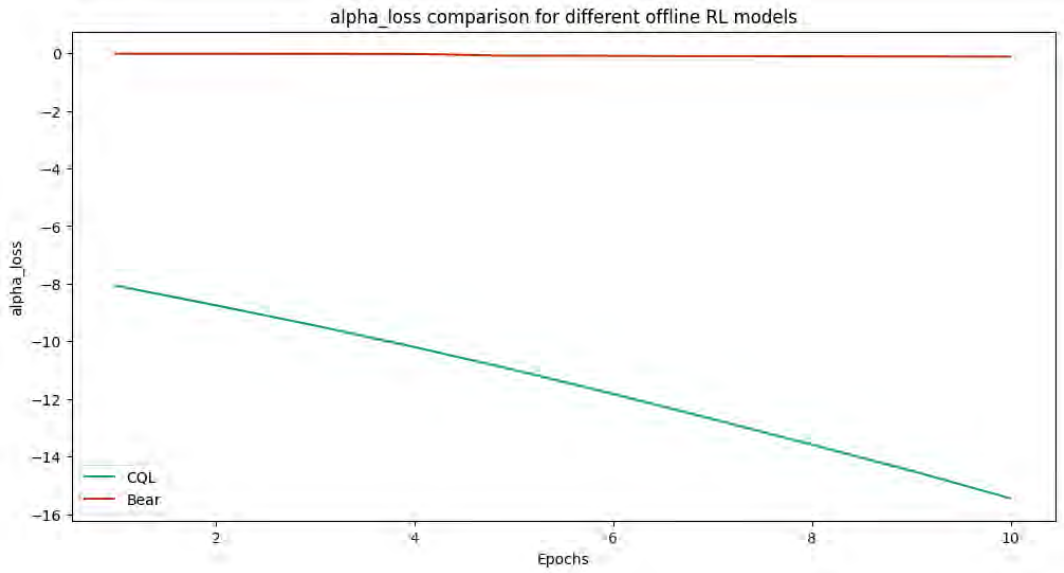

Figure 5.4: Alpha Loss of CQL and BEAR in Training Phase

Q-values is reflected in the alpha loss.It is also related to the modification done to keep the ensemble's level of uncertainty high, avoiding predictions that are too optimistic. The method of maintaining a higher level of conservatism or uncertainty across epochs is consistently demonstrated by a higher and straight-line trend in the alpha loss graph for BEAR. This may be done on purpose to make sure offline data is learned carefully. A declining trend however with lower CQL values points to a possible lessening or refining of the degree of conservatism across historical periods. It's possible that CQL is adjusting to the data and progressively growing more certain of its forecasts. Due to this CQL is providing somewhat better results than BEAR.

In Figure 5.5, we can see the imitator loss difference between BCQ and BEAR. In offline reinforcement learning models, imitator loss is the loss attributed to the imitator network. Using the provided offline dataset, the imitator network is in charge of producing actions that mimic the behavior of an expert policy. The imitator network in BEAR appears to be encountering more difficulties or uncertainty when attempting to imitate the expert policy, as indicated by the higher imitator loss in BEAR relative to BCQ. The offline dataset's nature or the environment's complexity could be to blame for this. With a lower imitator loss, BCQ may have been more successful in training its imitator network to mimic the expert policy.

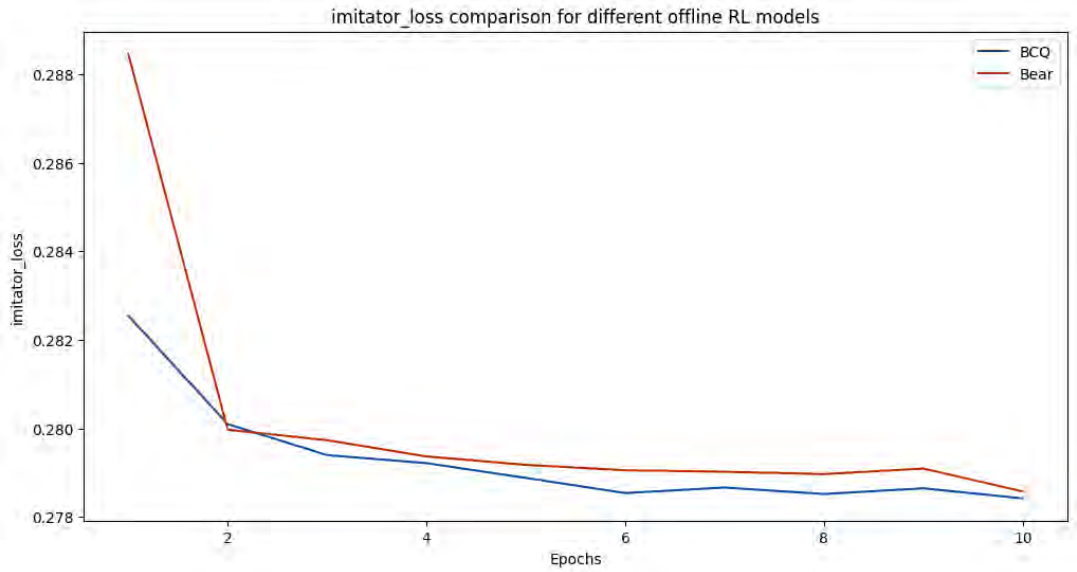

Figure 5.5: Imitator Loss of BCQ and BEAR in Training Phase

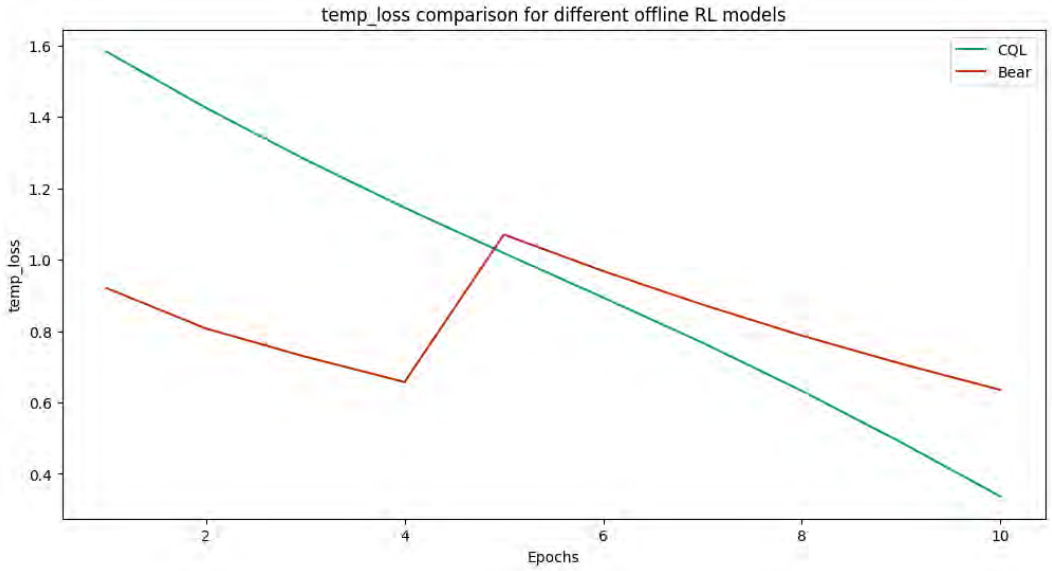

Figure 5.6: Temp Loss

A loss related to a temperature parameter is referred to as "temp loss" in the context of offline reinforcement learning models. This parameter regulates the amount of exploration and exploitation that occurs during training in algorithms that use entropy regularization. In Figure 5.6, the decreasing trend of CQL suggests that the temperature parameter has less of an effect over epochs. This could indicate that as the model gains confidence in its actions and learns more, there will be a gradual decrease in the need for exploration. The trajectory of BEAR shows a decrease at the beginning, an increase in the middle, and a final decrease at the end, indicating a subtle shift in the relative importance of exploration and exploitation. BEAR adjusts itself more than CQL in this way.

5.2.2 Testing Phase

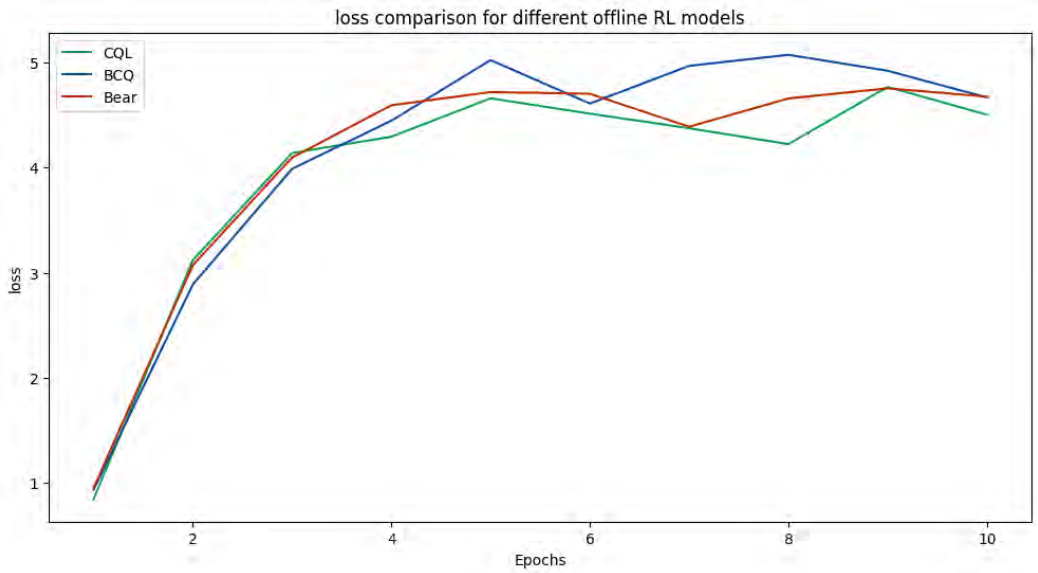

Figure 5.7: Loss of CQL, BCQ and BEAR after Evaluation with FQE

Finally in Figure 5.7 we can see the total value of the loss function for all three algorithms. An upward trending loss value graph is a sign that the model is having trouble fitting the training set or is diverging. This may indicate that the model is not learning effectively and that its performance is not being improved by the parameter updates made during training. It is possible that the training data is being overfitted, which means that noise or certain patterns in the training set are being captured but are not being applied to new examples. For this reason as the loss values of all three algorithms are following the same trend and going up rather than decreasing, these algorithms might not provide the best solution to our problem as well.
Similarly for both LSTM and Offline RL models, we used static data using an offline dataset we produced from online resources. But the bitcoin blockchain network is very rarely static. It is a dynamic environment and our solution must be able to cope with that as well. For all of these reasons we must move on to an environment which can emulate the dynamic tendencies of the bitcoin blockchain network and algorithms which will use Reinforcement learning to predict the optimal blocksize namely online RL models.

5.3 Online RL Model and Reward Shaping

5.3.1 Loss Functions

In Figure 5.8, we see the difference of actor loss between the three online algorithms in the training part. Just like the offline version, the online models also have actor loss and the lower the actor loss, the more better and optimal steps are taken by the agent resulting in better rewards. All three algorithms show promise here with a downward trend in each line. The lowest values are seen in the TQC line. But

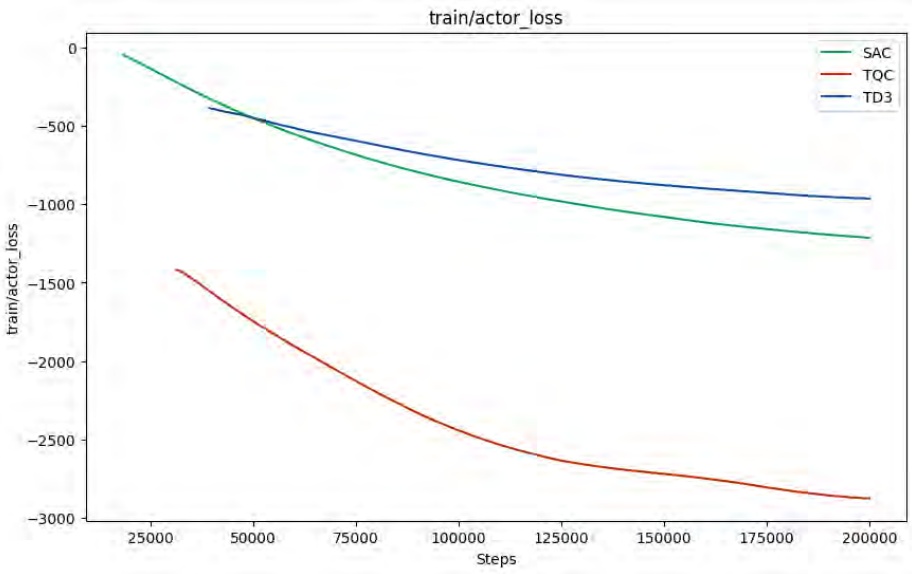

Figure 5.8: Actor Loss of SAC, TQC and TD3 in Training Phase

the downward trend of all of the lines refers to the fact that all three algorithms are converging.

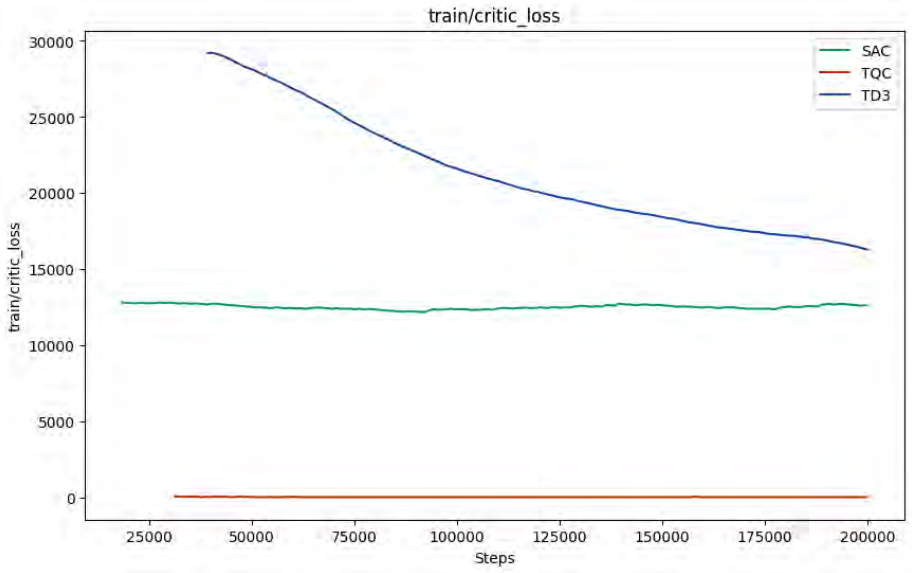

Figure 5.9: Critic Loss of SAC, TQC and TD3 in Training Phase

The critic loss of the three chosen algorithms during the training phase is depicted in Figure 5.9. Just like offline models, less critic loss indicates a much better exploration and exploitation by the model. Here in the graph we can see the lowest is again seen in TQC, with the downward trending graph from TD3 which means that as the training goes on this algorithm is improving its exploitation of the environment and making better decisions.

In Figure 5.10, a comparison in the entropy coefficient parameter between the SAC and TQC can be seen. One parameter that affects the trade-off between exploration and exploitation is the entropy coefficient. It is employed to control the degree of

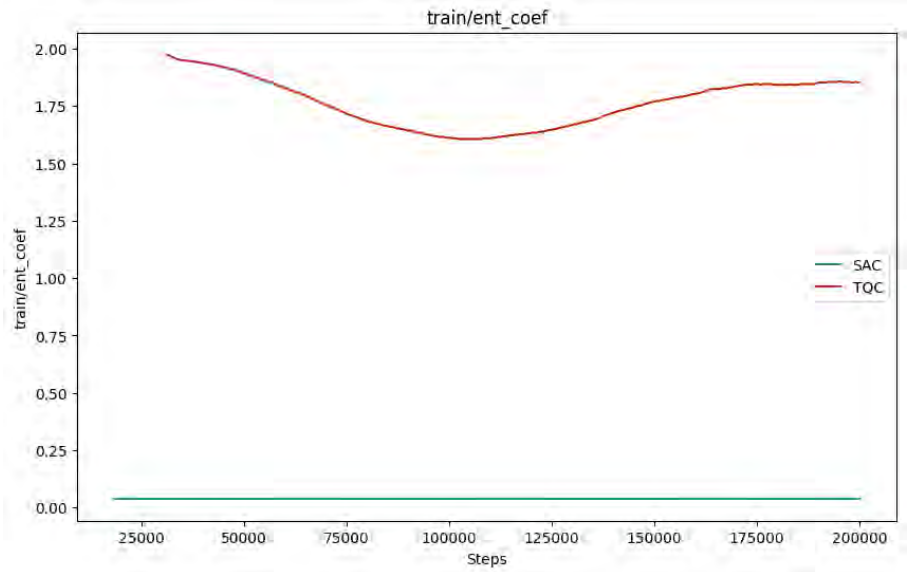

Figure 5.10: Entropy Coefficient of of SAC and TQC in Training Phase

entropy, or unpredictability, in the activities of the policy. The policy is encouraged to explore a wider range of actions the higher the entropy coefficient. This graph shows how SAC and TQC approach exploration differently, with TQC taking a more exploratory approach with fluctuations in its entropy coefficient and SAC favoring lower entropy and exploitation. Due to its constant low entropy coefficient, SAC might be prioritizing learning the policy over exploration, possibly at the expense of some degree of exploration. Given its fluctuations and initial higher entropy coefficient, TQC might be more flexible in its approach to exploration. The variations may indicate that TQC modifies its degree of investigation in response to changes in the environment or in learning. This parameter is not that important to our solution but it is still worth the mention.

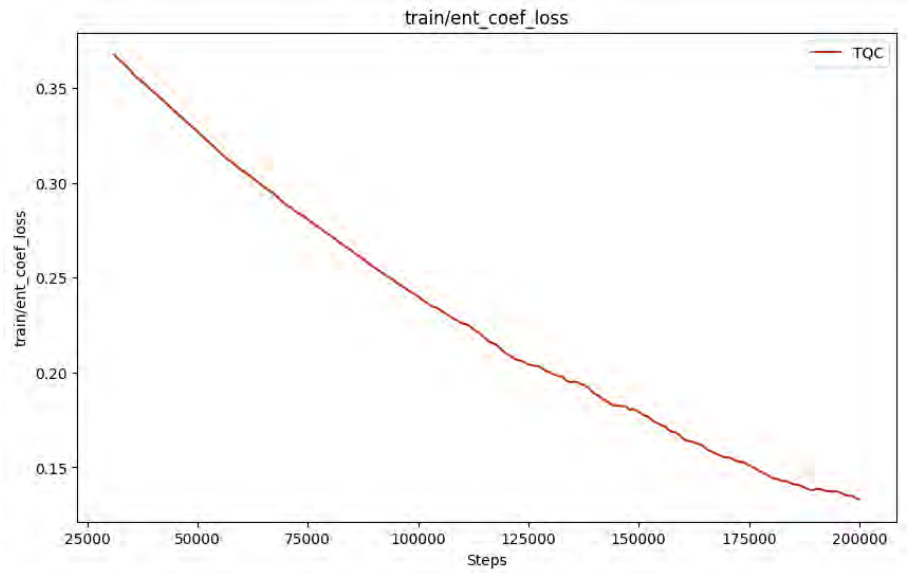

Figure 5.11: Entropy Coefficient Loss of TQC in Training Phase

The loss value of the entropy coefficient parameter of TQC is shown in Figure 5.11. It is a unique value unique to this model only. Typically, algorithms like TQC that employ entropy regularization are linked to the entropy coefficient loss. A component of the overall loss function that promotes or inhibits the policy's degree of stochasticity is the entropy coefficient loss. A decline in the loss correlated with the entropy coefficient over time steps is indicated by a downward trend in the mentioned figure. This implies that in order to better fit the environment's learning dynamics, the algorithm is progressively optimizing or modifying the entropy coefficient.

5.3.2 Reward

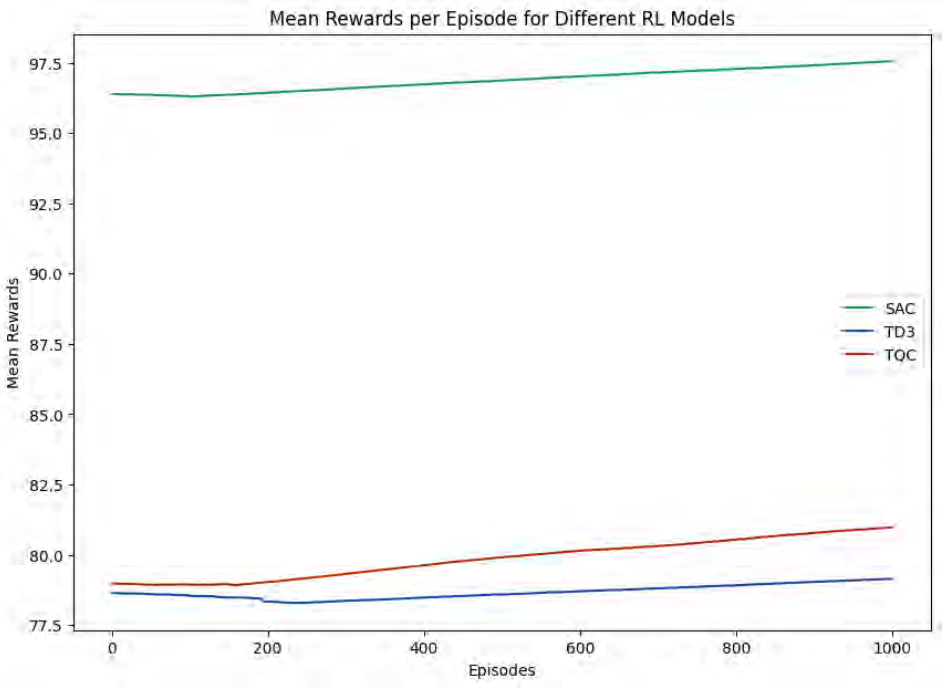

Figure 5.12: Mean Reward Per Episode of SAC, TD3 and TQC in Training Phase

In Figure 5.12, we are comparing SAC, TD3 and TQC models and their respective mean rewards per episode in the training phase. Each episode consists of 200 time steps. So the average of that is plotted in the graph. Firstly all three algorithms show an upward trend meaning that with training the rewards are getting higher, resulting in the agents making better decisions. But here also we can see that the SAC algorithm is much higher than the rest of the two.

In Table 5.2, we can see the reward values of different online models which we trained in our environment and also the inverse RL variant of each of these after evaluating. The highlight of the table is that the rewards are exceptionally better than any of the previous methods we used. Among the algorithms we used, the Soft Actor Critic model(SAC) gives us the best results in terms of rewards. The higher the rewards value means that in every step the model took the better decisions which when calculated the entire mean would provide the most. From this table we can clearly see the SAC model is generating the most rewards.

Table 5.2: Rewards of Different RL Models after Evaluation

Models	Mean Reward	STD Reward
SAC	23926.346	999.372
SAC IRL	18907.007	1268.051
TD3	18651.235	1260.117
TD3 IRL	19156.924	877.763
TQC	20366.622	2253.465
TQC IRL	20324.564	2482.507

5.3.3 Block Utilization

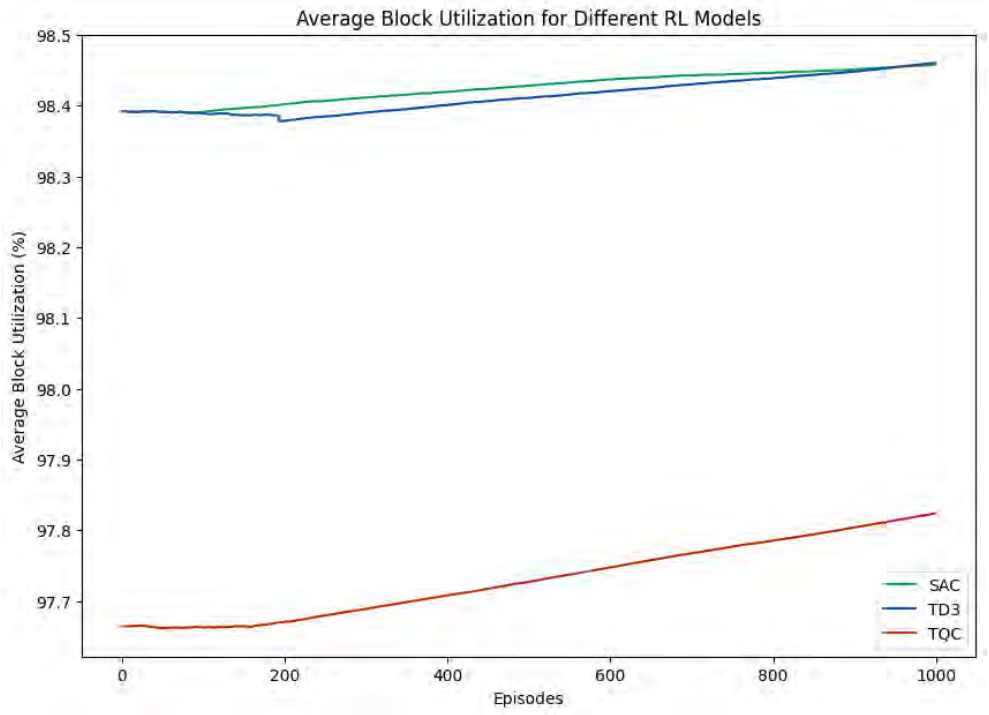

Figure 5.13: Average Block Utilization of SAC, TD3 and TQC in Training Phase

Figure 5.13 shows the average block utilization of our proposed models. Here we can see an upward trend among the algorithms translating that while training, the block utilization increases in all three algorithms, and SAC and TD3 here reigns supreme again over TQC, with TD3 producing the best value and SAC producing a value almost as good as TD3. Our proposed models have increased the block utilization to a greater extent while decreasing the average confirmation time.

Looking at Figure 5.14, we can observe the maximum amount of block utilization generated by the three suggested algorithms. These are very similar to each other,

showcasing that rather than having a fixed block size of 2MB or 3MB, the dynamic blocksize predicted by our algorithms generate blocks which are utilized almost completely rather than having very low utilization. This indicates the fact that the generated blocks by these algorithms are optimal.

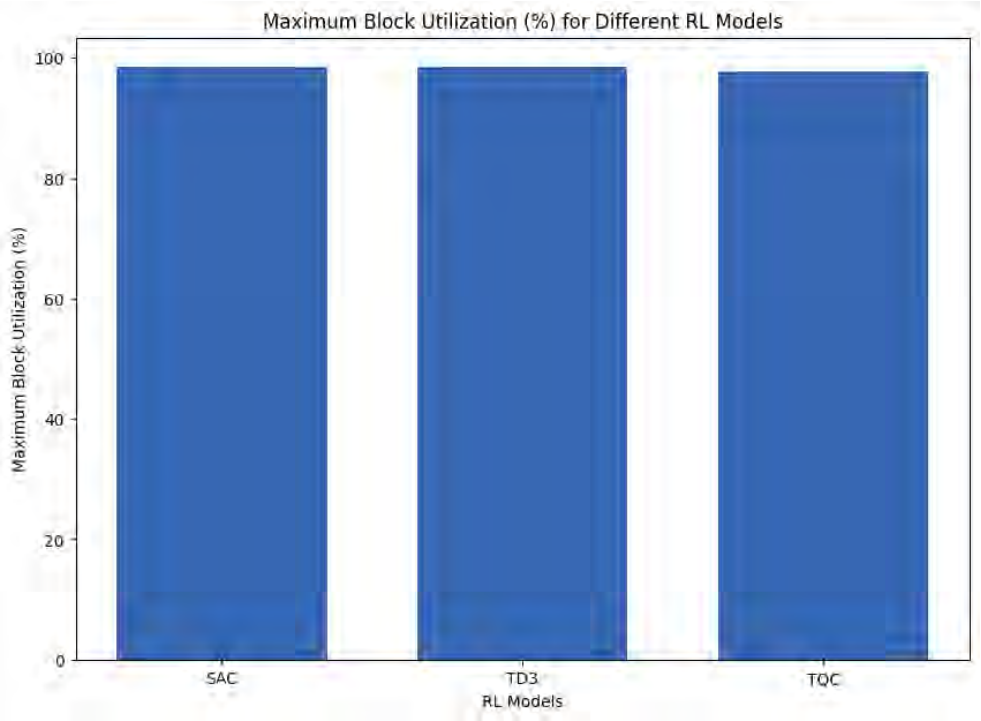

Figure 5.14: Comparison of Maximum Block Utilization of SAC, TQC and TD3 in Training Phase

To find out whether dynamic blocksize is more feasible than fixed blocksizes, we have done experiments with fixed blocksizes along with the RL models. Average block utilization of all the models are depicted in Figure 5.15. We experimented with three fixed block size models with each having a block size of 1MB, 2MB and 3MB. We would like to utilize as much of the block we generated as possible to make sure we can get the most from what our hardwares can offer. A block with the size of 3MB, might not be possible for the general miners, which means only a handful people can generate these. This means the bitcoin network will become centralized, which goes against one of the primary reasons for the invention of Bitcoin which is decentralization and transparency of all transactions. So for us to take into account, the average block utilization must be over 95% which is why we will not be taking 2mb and 3MB in account. Other than that we also have the comparison between the rest of the models used; the RL models with and without IRL and ROBB[34]. We can see that every model is delivering a decent block utilization other than the fixed 2MB and 3MB sizes. So, for evaluating our proposed models we will need to compare more parameter which are done in the following subsections.

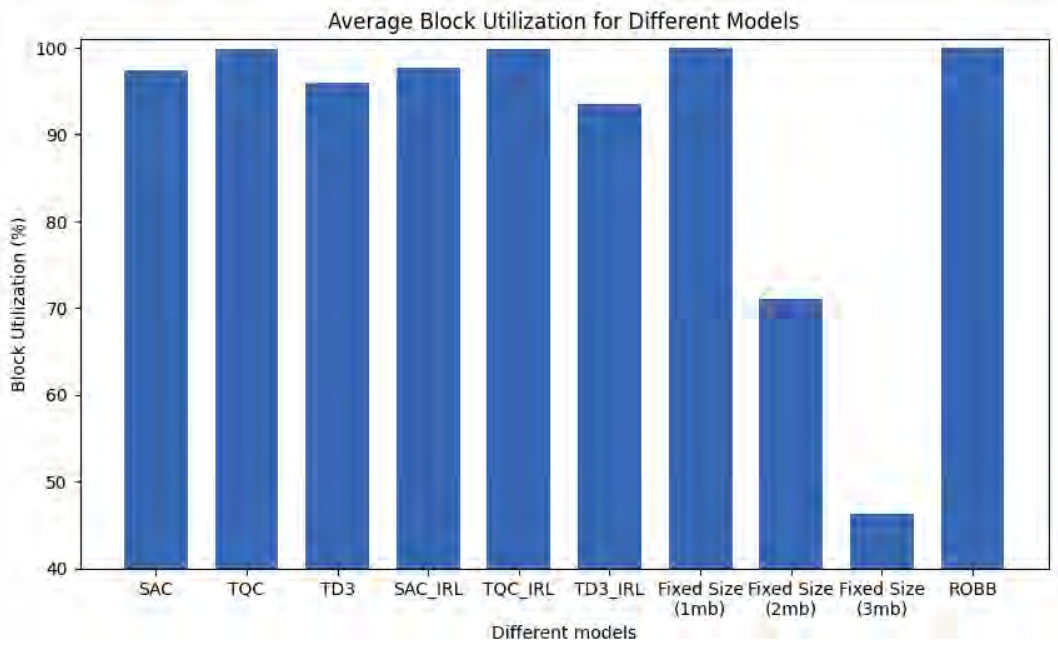

Figure 5.15: Comparison of Average Block Utilization of SAC, TQC, TD3 with IRL Results, Fixed Size (1MB) and ROBB

5.3.4 Miners' Revenue

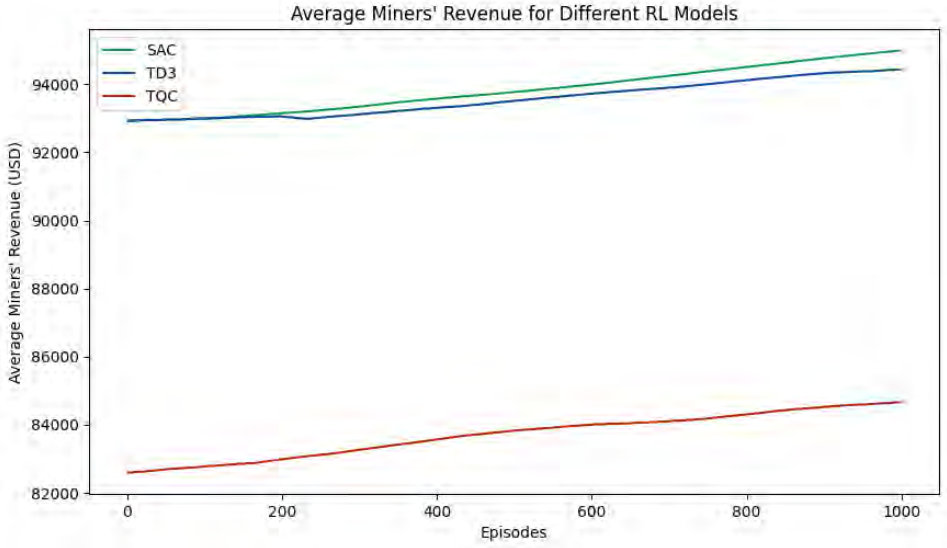

Figure 5.16: Average Miners' Revenue of SAC, TQC, TD3 in Training Phase

One of the most crucial objectives of our project is shown in Figure 5.16. Here we can see the direct comparison between the average Miners' revenue for SAC, TD3 and TQC. Up until now TQC was giving the best values. But for this parameter, SAC is giving the highest amount of revenue in USD. All three models do show upward tendencies which means while training, all three algorithms are able to increase and maximize the miners revenue which is one of the major goals of our project. So this graph here shows that SAC generates the highest amount of revenue on an

average after one thousand episodes.

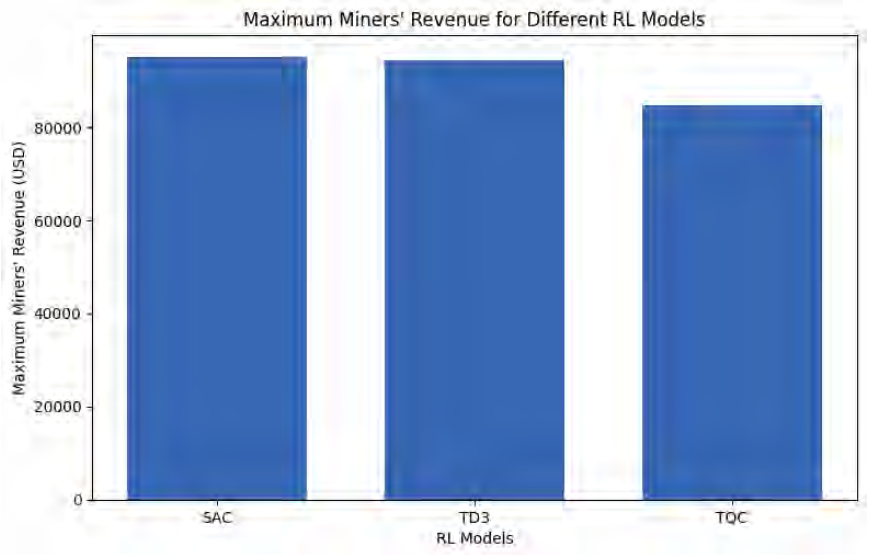

Figure 5.17: Comparison of Maximum Miners' Revenue of SAC, TQC and TD3 in Training Phase

In Figure 5.17 we can see the maximum miners' revenue for each algorithm as bar charts throughout the entire training phase. Within the one thousand episodes and each episode consisting two hundred time steps, the maximum value came from the SAC algorithm. This and the previous graph means that the miners can generate maximum revenue and maximum average revenue if the blocks are generated according to the specifications of the SAC model.

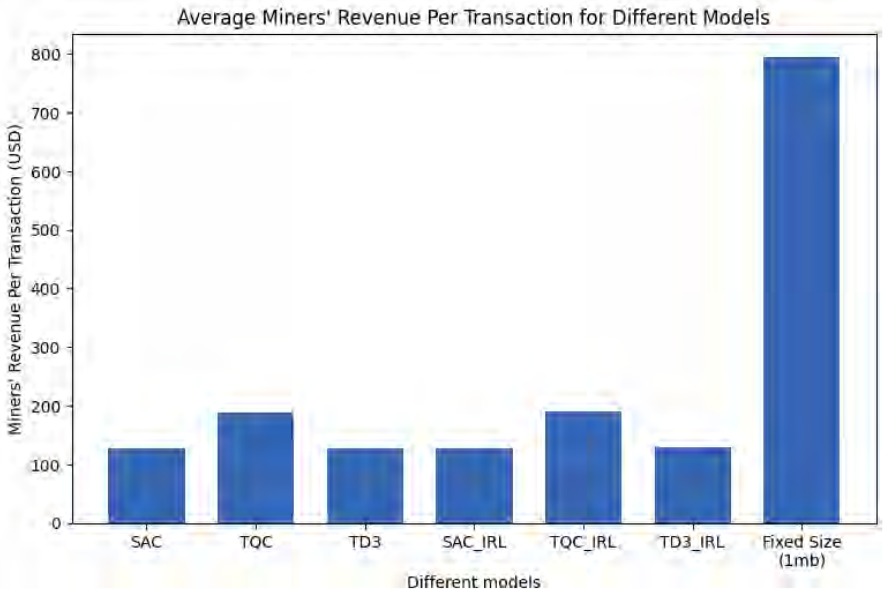

Figure 5.18: Comparison of Average Miners' Revenue of SAC, TQC, TD3 with IRL Results, Fixed Size (1MB) and ROBB

In Figure 5.18 we can see the comparison between the RL models during the testing

phase and the actual 1MB fixed block size methods' average miners' revenue. As we can see in the graph the already implemented method is generating much more in comparison to the RL models. This is because the block size is much bigger than the ones generated by the RL models. But this greater amount of block size is not always the best solution. We will see later on how even though the fixed size version is seeming to be generating the most revenue, in comparison of confirmation time, the RL models will outshine this method.

5.3.5 Confirmation Time

Another primary objective of our project is to lessen the average confirmation time as much as possible. In Figure 5.19, we can see the comparison between each models in nano second. Here we can see, even though TQC worked better as an RL agent, when it comes to our environment TD3 and SAC are out performing it. All three lines have a downward trend meaning that throughout training the average confirmation time is decreasing which corresponds with our goal. In this graph we can see that model TD3 is giving the least amount of time when it comes to average confirmation time.

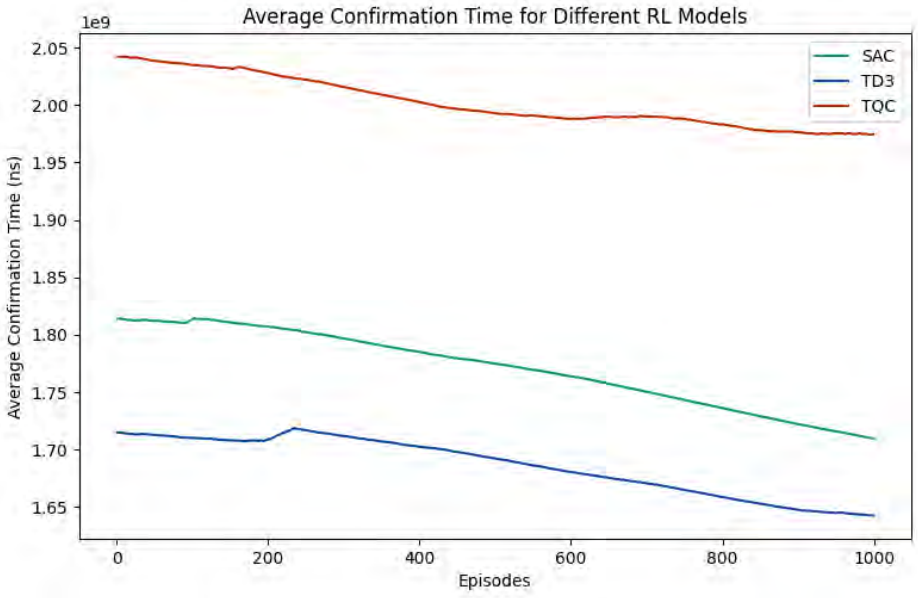

Figure 5.19: Average Confirmation Time of SAC, TQC and TD3 in Training Phase

In Figure 5.20 we can see during training which model gave the minimum confirmation time among the three RL models. Here we can see TD3 giving the minimum value during the entire training. However the minimum confirmation time produced by SAC is not that much larger than the one from TD3, which is why even though TD3 generated the least amount of confirmation time, on an average SAC is still taking less confirmation time for the transactions.

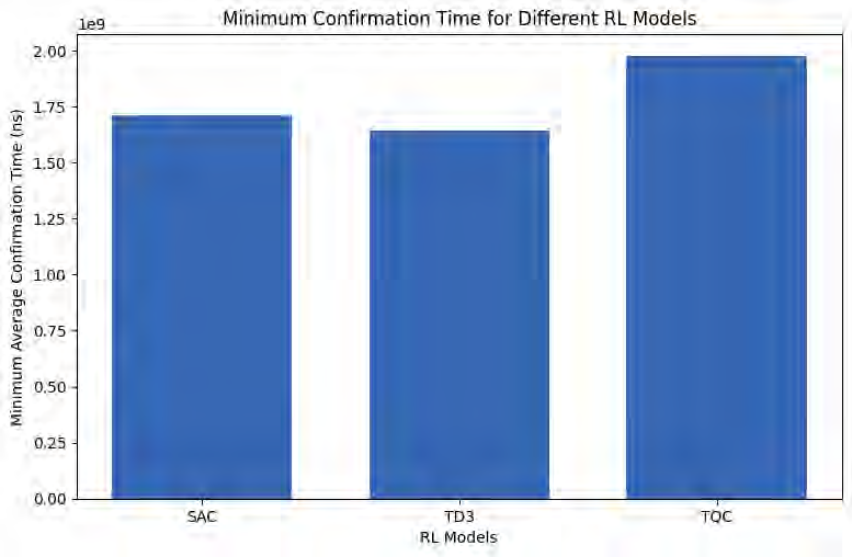

Figure 5.20: Minimum Confirmation Time of SAC, TQC and TD3 in Training Phase

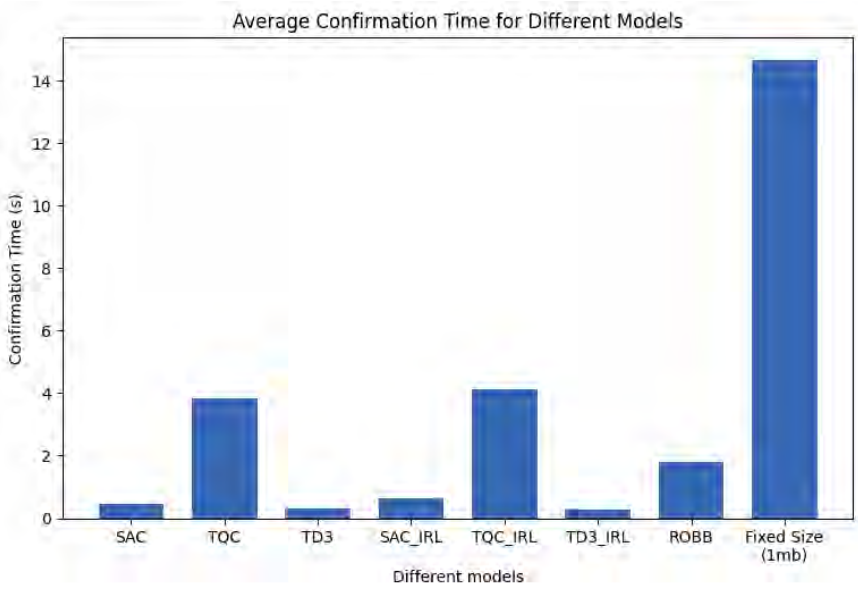

Figure 5.21: Comparison of Average Confirmation Time of SAC, TQC, TD3 with IRL Results, Fixed Size (1MB) and ROBB

In Figure 5.21, we can see the direct comparison between the fixed block size mechanism that is implemented now with our methods using RL models after evaluating. We also took into account another model ROBB by [34] which still showed more confirmation time on an average than the RL models. In this case the fixed block size method is taking more than fourteen seconds to confirm the transactions on an average whereas the SAC is taking less than a second. This is where our model outshines all existing methods and satisfies one of our major objectives to reduce the average confirmation time. Here we can see direct comparison to already existing methods and showcasing our ones.

5.3.6 Inter-Parametric Comparison

As the objective of this research was to minimize the average confirmation time and maximize the miners' revenue while maintaining a decent block utilization rate, we have to compare these 3 parameters together to fully understand how the models are performing. To do that we have taken a ratio of average miners' revenue and confirmation time. Figure 5.22 shows the comparison of this ratio between the models.

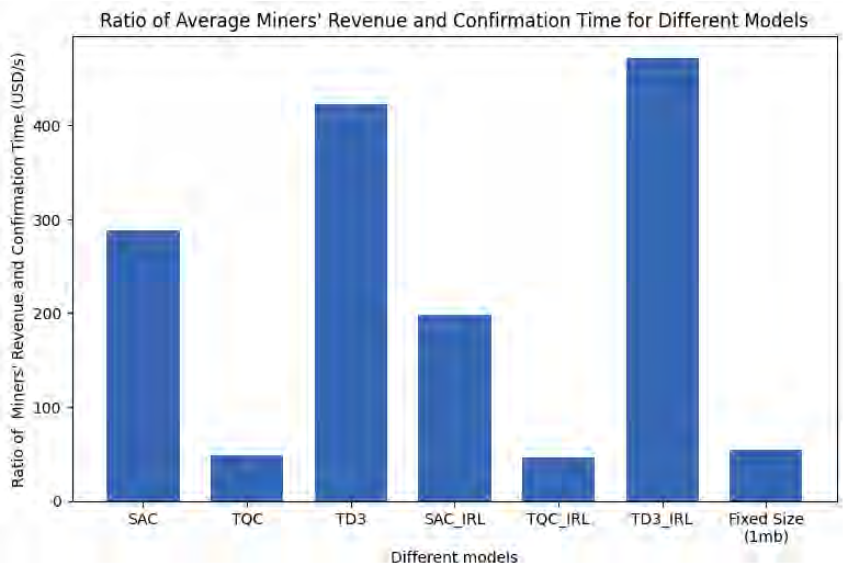

Figure 5.22: Ratio (Miners' Revenue/Confirmation Time) Comparison of SAC, TQC, TD3 with IRL Results, Fixed Size (1MB)

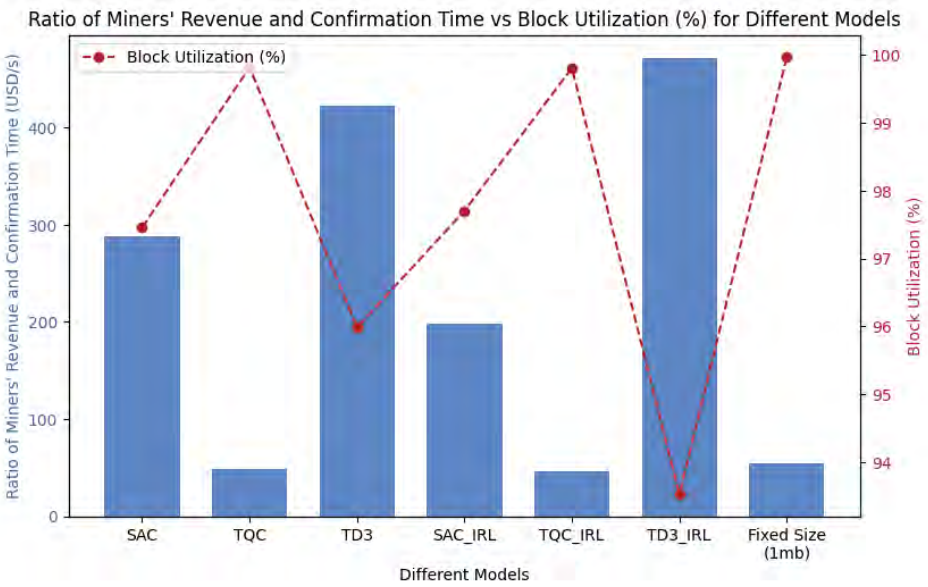

Figure 5.23: Ratio (Miners' Revenue/Confirmation Time) Comparison of SAC, TQC, TD3 with IRL Results, Fixed Size (1MB) with Block Utilization

By taking the block utilization in consideration, we get the Figure 5.23. From this, we can see that even though the ratio is the highest for TD3 IRL model but the block utilization is very low. In another case, even though the block utilization is very high but the ratio of revenue and confirmation time is very low for TQC model. We need something that balances between these two which is exactly what SAC model is giving us. Even though SAC model is not giving us the highest in ratio but it is better than most while maintaining a decent block utilization rate. So, SAC model is actually helping us fulfil the objectives while maintaining a block utilization rate over 97%.

Finally, as shown in Table 5.3, we can see the specific values of the Algorithms to do a more direct mathematical comparison between everything. Here we can see in the models that the models are generating better result than the ones trained using IRL. This once again goes on to prove the fact that the reward function generated by us using trial and error method is providing a more optimal result than the one generated by IRL models. The highest value of the ratio is given by TD3 IRL model, but it has a lower block utilization percentage. So among all the models, SAC is giving a very low confirmation time, a much higher block utilization, and a decent amount of revenue generated per confirmation time. In conclusion of this analysis we can vouch for SAC model to be the most optimal model with our generated reward function to predict the dynamic block size.

Table 5.3: Performance of Different Models

Models	**Average Confirmation Time (sec)**	**Average Block Utilization (%)**	**Miners' Revenue Per Transaction (USD)**	**Ratio = (Revenue/ Confirmation time) (USD/s)**
SAC	0.440	97.454	126.837	288.265
SAC IRL	0.646	97.701	127.687	197.657
TD3	0.302	95.986	127.747	423.003
TD3 IRL	0.275	93.530	129.642	471.425
TQC	3.822	99.812	187.437	49.041
TQC IRL	4.120	99.803	190.664	46.277
Fixed Size (1MB)	14.651	99.962	794.22	54.209
ROBB[34]	1.8	100		

We can view the block utilization parameter for each of the models that were dis-

cussed in table 5.3. Block utilization is one of the parameters that we prioritize the highest. The reason is that block creation is highly dependent on the hardware, and we want to make the most use of the blocks we create. For this reason, we want to maximize block utilization. Having said that, we can observe that the Twin Delayed Deep Deterministic Policy Gradient (TD3) model and the evaluated model of the same algorithm using inverse reinforcement learning produce the least amount of block utilization out of all eight methods listed in 5.3. For the algorithm to consider as a potential solution to our problem, we would like to set the block utilization parameter above 97%. With this bound being set, all reinforcement learning algorithms except TD3 and TD3 evaluated by IRL will be compared to another parameter which is confirmation time with respect to the fixed size method. Following is the equation of the ratio of confirmation time of different models, C_m with respect to fixed blocksize method, C_f 5.6.

$$R_c = \frac{C_f}{C_m} \cdot [BU > 97\%] \tag{5.6}$$

To visualize the improvement of confirmation time among the reinforcement learning models we would like to use this equation 5.6 to measure the equivalent amount of data which can be sent from one user to another. Here we are comparing the confirmation time of each model with the confirmation time of the currently existing model(Fixed block size(1MB)). In the current established method of bitcoin blockchain network, each transaction takes about 14.651 seconds to be completed. We would like to showcase how many transactions can be completed in the same exact amount of time but with reinforcement learning models. From figure 5.24 we

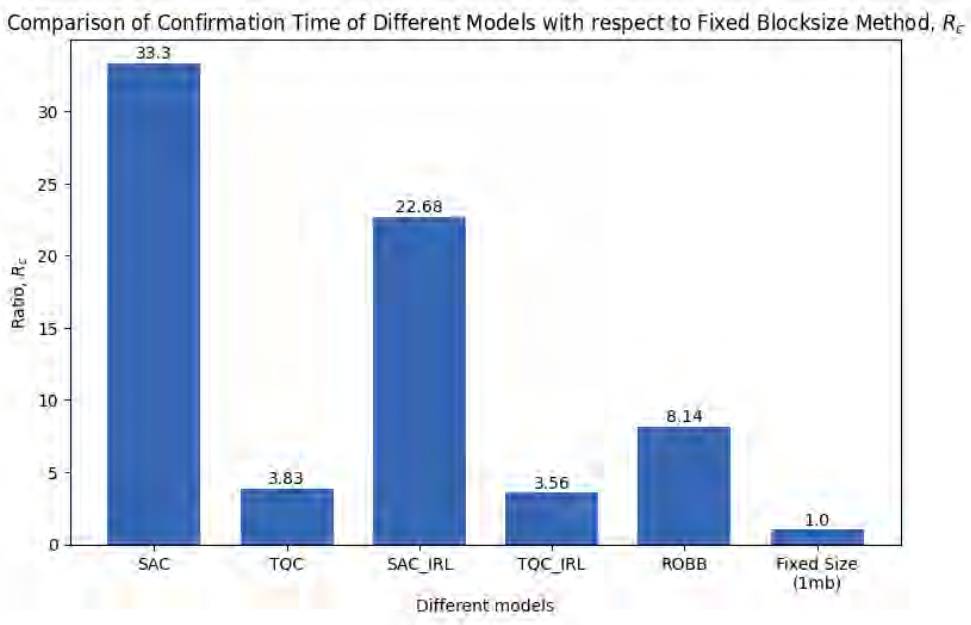

Figure 5.24: Comparison of Confirmation Time of Different Models with respect to Fixed Blocksize Method, R_c

can see the vast difference between the currently utilized method and the reinforcement learning models namely our proposed model Soft Actor-Critic (SAC) model.

Here we can see, if the same transaction could be sent using all methods, the current established method of fix block size would take 14.651 seconds to confirm one transaction whereas Soft Actor-Critic model will be able to confirm 33 equivalent transactions in that exact same time, which is higher than any other model in comparison. This provides proof to the fact that the Soft Actor-Critic model will work the best in terms of scalability of the blockchain network.

Chapter 6

Conclusion

To conclude, in this paper, we investigated the use of Reinforcement Learning (RL) methods to forecast the Bitcoin blockchain network's dynamic block size. Long Short-Term Memory (LSTM) networks, offline RL models (BEAR, CQL, BCQ), and online RL models (SAC, TD3, TQC) were all thoroughly compared in our study. Furthermore, two different reward functions were employed, one obtained by trial and error and the other through the use of Inverse RL, namely Generative Adversarial Imitation Learning (GAIL). Following extensive testing, we discovered that the Soft Actor-Critic (SAC) model performed better than other models when combined with the reward function created by trial and error. With an average confirmation time of 0.440 seconds, a block utilization rate of 97.454%, and a noteworthy generation of 288.265 USD per second, the SAC model produced impressive results. The SAC model's success highlights how well online reinforcement learning techniques adjust dynamically to the changing characteristics of the blockchain network. SAC showed better performance in predicting ideal block sizes, reducing confirmation times, and optimizing block utilization by learning from its interactions with the environment. Additionally, our research advances knowledge about RL's potential for modifying and enhancing blockchain networks. SAC's demonstration of the adaptive nature of RL models provides opportunities for the dynamic modification of crucial parameters, presenting a viable solution to blockchain technology's scalability issues.

Nonetheless, it is imperative to recognize specific limitations within our research. Firstly, the unique features of the Bitcoin network might limit the applicability of our findings in other blockchain contexts. Furthermore, there is some subjectivity introduced in the choice of reward functions and RL models, which affects the overall performance. Subsequent investigations could examine the resilience of these results in various blockchain applications and assess how sensitive they are to choices in reward functions.
Our findings point to the potential of reinforcement learning to transform blockchain networks through dynamic parameter optimization. Together with our created reward function, the SAC model's performance signifies a major advancement in resolving scalability issues with blockchain technology. RL provides a potent tool for adjusting to the dynamic requirements of these decentralized systems as blockchain networks develop, opening the door for more effective and scalable blockchain architectures.

6.1 Future works

In the last several years, there has been a lot of research done on the topic of long-term viability in Bitcoin. To achieve the best results, we used a variety of techniques in this research, including LSTM, offline reinforcement learning, and online reinforcement learning. Out of all of them, LSTM and offline RL was trained and tested in a static setting which shows unfavorable outcomes. Consequently, the online version's implementation of the RL agent in a dynamic environment demonstrates a notable improvement in terms of block size adjustment and overcoming the 1MB size constraint of blocks during Bitcoin transactions.

In the future, we would also like to introduce multi agent RL. Because compared to single-agent reinforcement learning, multi-agent reinforcement learning has different opportunities and challenges, such as managing non-stationarity, addressing the problem of credit assignment in a multi-agent scenario, and coping with the creation of sophisticated strategies. So that miners can produce blocks even faster without hindering security in situations with significant network traffic and complex circumstances.

Bibliography

[1] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[2] M. Moser, "Anonymity of bitcoin transactions," 2013.

[3] F. Velde *et al.*, "Bitcoin: A primer," 2013.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[6] J. Ho and S. Ermon, "Generative adversarial imitation learning," *CoRR*, vol. abs/1606.03476, 2016. arXiv: 1606.03476. [Online]. Available: http://arxiv.org/abs/1606.03476.

[7] X. Min, Q. Li, L. Liu, and L. Cui, "A permissioned blockchain framework for supporting instant transaction and dynamic block size," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 90–96. DOI: 10.1109/TrustCom.2016.0050.

[8] A. Nowé and T. Brys, "A gentle introduction to reinforcement learning," in *Scalable Uncertainty Management: 10th International Conference, SUM 2016, Nice, France, September 21-23, 2016, Proceedings 10*, Springer, 2016, pp. 18–32.

[9] J. Göbel and A. E. Krzesinski, "Increased block size and bitcoin blockchain dynamics," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, 2017, pp. 1–6.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[12] A. Nagabandi, I. Clavera, S. Liu, *et al.*, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[13] A. Pourchot and O. Sigaud, "Cem-rl: Combining evolutionary and gradient-based methods for policy search," *arXiv preprint arXiv:1810.01222*, 2018.

[14] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," *CoRR*, vol. abs/1910.01708, 2019. arXiv: 1910.01708. [Online]. Available: http://arxiv.org/abs/1910.01708.

[15] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*, PMLR, 2019, pp. 2052–2062.

[16] T. Haarnoja, A. Zhou, K. Hartikainen, *et al.*, *Soft actor-critic algorithms and applications*, 2019. arXiv: 1812.05905 [cs.LG].

[17] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[18] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019. DOI: 10.1109/ACCESS.2019.2936094.

[19] H. Sheth and J. Dattani, "Overview of blockchain technology," *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146*, 2019.

[20] S. Dankwa and W. Zheng, "Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," ser. ICVISP 2019, Vancouver, BC, Canada: Association for Computing Machinery, 2020, ISBN: 9781450376259. DOI: 10.1145/3387168.3387199. [Online]. Available: https://doi.org/10.1145/3387168.3387199.

[21] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[22] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 13–18 Jul 2020, pp. 5556–5566. [Online]. Available: https://proceedings.mlr.press/v119/kuznetsov20a.html.

[23] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *CoRR*, vol. abs/2005.01643, 2020. arXiv: 2005.01643. [Online]. Available: https://arxiv.org/abs/2005.01643.

[24] CoinGecko, *Cryptocurrency prices by market cap*, https://www.coingecko.com/en, Accessed: 2021-10-15? 2021.

[25] L. Conway, *What is bitcoin halving? definition, how it works, why it matters*, https://www.investopedia.com/bitcoin-halving-4843769, Accessed: 2021-10-15? 2021.

[26] W. Fan, J. Kim, I. Kim, X. Zhou, and S.-Y. Chang, "Performance analyses for applying machine learning on bitcoin miners," in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, IEEE, 2021, pp. 1–4.

[27] J. Frankenfield, *Segregated witness (segwit): Definition, purpose, how it works*, https://www.investopedia.com/terms/s/segwit-segregated-witness.asp, Accessed: 2021-10-15? 2021.

[28] A. Nair, A. Gupta, M. Dalal, and S. Levine, *Awac: Accelerating online reinforcement learning with offline datasets*, 2021. arXiv: 2006.09359 [cs.LG].

[29] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.

[30] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html.

[31] P. Januszewski, "Model-free and model-based reinforcement learning, the intersection of learning and planning," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1849–1851.

[32] S. Xue, C. Qu, X. Shi, *et al.*, "A meta reinforcement learning approach for predictive autoscaling in the cloud," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4290–4299.

[33] R. Zhang, X. Zhang, C. Ni, and M. Wang, "Off-policy fitted q-evaluation with differentiable function approximators: Z-estimation and inference theory," in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, 17–23 Jul 2022, pp. 26 713–26 749. [Online]. Available: https://proceedings.mlr.press/v162/zhang22al.html.

[34] A. Dutta, "Robb: Recurrent proximal policy optimization reinforcement learning for optimal block formation in bitcoin blockchain network," M.Sc. thesis, Brac University, 2023.

[35] M. Monem, M. T. Hossain, M. G. R. Alam, *et al.*, "A sustainable bitcoin blockchain network through introducing dynamic block size adjustment using predictive analytics," *Future Generation Computer Systems*, vol. 153, pp. 12–26, 2024, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2023.11.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23004065.