BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

A Machine Learning-Based Lightweight Consensus Framework for Blockchain-Enabled IoT Systems

Mubtasim Kamal Dihan
200041138
Abdullah
200041126
Amina
200041155

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

A Machine Learning-Based Lightweight Consensus Framework for Blockchain-Enabled IoT Systems

Mubtasim Kamal Dihan
200041138
Abdullah
200041126
Amina
200041155

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

Declaration of Candidate

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Mubtasim Kamal Dihan**, **Abdullah**, and **Amina** under the supervision of **Dr. Md. Sakhawat Hossen**, Professor, Department of Computer Science and Engineering and co-supervision of **Faisal Hussain**, Assistant Professor, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Dr. Md. Sakhawat Hossen
Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 29, 2025

Faisal Hussain
Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 29, 2025

Mubtasim Kamal Dihan
Student ID: 200041138
Date: September 29, 2025

Abdullah
Student ID: 200041126
Date: September 29, 2025

Amina
Student ID: 200041155
Date: September 29, 2025

Dedicated to those whose faith encouraged us, and to those whose doubt fueled our determination.

Contents

List of Figures

List of Tables

Acknowledgement

We would like to extend our sincere gratitude to our supervisor, Dr. Md. Sakhawat Hossen, for his continued guidance and thoughtful mentorship. Throughout the challenges we all encountered, he remained a constant source of support and understanding, offering us the time and insight that few in his position would so generously provide.

We are profoundly grateful to our co-supervisor, Faisal Hussain, who consistently made time for us whenever and wherever we sought guidance. His willingness to listen, including during weekends, and his thoughtful, in-depth feedback provided invaluable direction throughout our work. The practical advice and consistent encouragement he offered greatly shaped the way we approached and refined our ideas.

We sincerely thank Tasnim Ferdous Anan (CSE'18, IUT) for his guidance during the early stages of our work. His support in helping us understand the topics and even how to begin, despite it not being his formal responsibility, was invaluable in setting a strong foundation for our research.

We also express our gratitude to all faculty members at IUT, whose direct or indirect guidance, whether through small suggestions or substantial advice, has greatly contributed to our growth. Special recognition is extended to those who, through extraordinary support and encouragement, went above and beyond in inspiring and motivating us throughout this journey — those who know their own contribution need no further mention.

Finally, we thank our parents, whose love, guidance and unwavering support have shaped who we are and continue to inspire us in all aspects of life.

Abstract

The integration of blockchain with the Internet of Things (IoT) offers a promising approach to address the limitations of centralized IoT architectures. However, existing blockchain consensus mechanisms are often unsuitable for resource-constrained IoT devices and dynamic network conditions due to high computational demands or reliance on monetary-based participant selection.. In this work, we propose Dynamic & Periodic Proof of Evolutionary Model (DP-POEM), a lightweight, machine learning-based consensus mechanism tailored for blockchain-based IoT systems. DP-POEM selects a group of block producers for defined periods using supervised learning, reducing redundant computation while ensuring security through randomized block production within the group. It incorporates both static and dynamically changing device features, such as battery level and CPU usage, to select capable nodes efficiently, and implements fair participation mechanisms to balance network involvement over time. Theoretical analysis and experimental evaluation demonstrate that DP-POEM achieves high scalability, low latency, enhanced security, and improved applicability compared to traditional and state-of-the-art consensus protocols in dynamic IoT environments.

Chapter 1

Introduction

Technological innovation has consistently been driven by humanity's pursuit of a better quality of life. From the Industrial Revolution to the digital age, each leap in technology has focused on integrating advanced digital systems to make human activities more efficient, connected, and intelligent [1]. In this ongoing evolution, the Internet of Things (IoT) has emerged as a transformative force, seamlessly becoming an integral part of everyday life. By interconnecting devices, environments, and people, IoT has become a key catalyst in the reshaping of modern lifestyles and societal infrastructures [2]. However, most existing IoT architectures still depend on centralized cloud servers for data storage and processing. This centralized dependency introduces critical limitations, including a single point of failure, reduced scalability, and high operational costs as the number of connected devices increases [3]. Therefore, addressing these challenges is becoming essential for ensuring the long-term resilience and efficiency of IoT ecosystems.

Originally introduced as the foundational technology behind the Bitcoin cryptocurrency, blockchain has emerged as a powerful solution that can address several challenges faced by traditional IoT systems [4]. As a decentralized, distributed ledger technology, blockchain allows transactions to be recorded securely, transparently, and in a tamper proof manner across a network of nodes. Although blockchain technology was initially applied to cryptocurrencies, its core properties, including decentralization, immutability, and trustless consensus, have extended its applicability far beyond digital currencies to areas such as smart cities, healthcare, and agriculture [5]. When integrated with IoT, blockchain can effectively mitigate the issues of centralization, enhance data integrity, and improve trust and scalability within interconnected device networks [6]. The integration of blockchain with IoT can be simpli-

fied through Blockchain-as-a-Service (BaaS) platforms, which provide managed infrastructure, development tools, and cloud-based services. Such platforms reduce the complexity of implementation and accelerate the deployment of secure, scalable blockchain-enabled IoT systems [7].

Consensus mechanisms lie at the heart of blockchain technology. It enables blockchain to provide security and trust equivalent to centralized systems without relying on a single authority. By requiring the agreement of the majority of network participants on the validity and compliance of transactions or data, these mechanisms ensure a consistent and reliable ledger across all nodes [8]. However, most existing consensus mechanisms are not designed with the resource constraints of IoT devices in mind, and their applicability is limited in dynamically changing environments and device states. For example, PoW, PBFT, and PoA incur high energy consumption, while PoS, DPoS, and LPoS rely on stake-based systems that have little relevance for IoT devices [9]. More recently, PoEM [10] was proposed as a lightweight consensus mechanism based on machine learning for IoT devices. However, its computational demand due to repetitive usage of the model limits scalability in networks with large numbers of nodes. Additionally, it does not account for the dynamically changing conditions and status of IoT devices. Given these limitations, a consensus mechanism tailored for BaaS-enabled IoT systems must overcome poor scalability, low efficiency, limited applicability, and the challenges posed by the highly dynamic operating conditions of IoT devices.

To address these limitations, we propose a lightweight, machine learning-based consensus mechanism called Dynamic & Periodic Proof of Evolutionary Model (DP-POEM) for BaaS-enabled IoT systems. Unlike traditional approaches that select a single block producer at a time, DP-POEM selects a group of block producers, each generating one block sequentially over a defined period. This group selection leverages supervised machine learning algorithms, ensuring that no additional overhead is introduced, since all nodes are already ranked by the model. The group size adapts proportionally to the total number of nodes, enhancing scalability and reducing computational overhead for resource-constrained IoT devices. Security is preserved by introducing a randomized block production order within each group, which prevents targeted attacks without incurring additional overhead. The capability of IoT devices to participate in block production is determined not only by static device characteristics but also by dynamically changing features such as current battery level, CPU and storage usage, and recharge rate. These dynamic features are incorporated into the machine learning model, and the relevant information is efficiently shared between the nodes to ensure consistent execution of the model. Finally, DP-POEM

incorporates a fair participation mechanism that allows new nodes to join the consensus process while preventing repeated selection of the same nodes. By combining these strategies, DP-POEM achieves high scalability, broad applicability, and strong resilience in dynamic IoT ecosystems. Both theoretical analysis and experimental evaluation validate the effectiveness of the proposed approach.

To summarize, the main contributions of our work are threefold.

1. We minimize the redundant computational overhead of repeatedly selecting a single block producer by instead selecting a group of block producers, each responsible for generating one block during a defined period. Security is ensured through a randomized block production order within the group.
2. We incorporate the dynamically changing status of IoT devices into the block producer selection process, ensuring that only nodes with sufficient current capability participate, thereby reducing overall latency. Furthermore, we design a mechanism for efficient and reliable sharing of this dynamic information across the network.
3. We implement fair participation strategies to prevent recently selected nodes from being repeatedly chosen, while allowing new nodes to join the consensus process after a suitable interval.

The remainder of the work is organized as follows. In Chapter 2, we present an overview of blockchain technology, consensus mechanisms, and the integration of IoT, machine learning, and blockchain, along with a detailed review of the existing literature. In Chapter 3, we provide a formal and detailed description of the DP-POEM consensus mechanism, including the design choices and the motivations behind the selected approach. Chapter 4 presents both theoretical and experimental results, with comparisons against traditional and state-of-the-art consensus mechanisms to demonstrate the superiority of DP-POEM. Finally, in Chapter 5, we draw concluding remarks and discuss directions for future works.

Chapter 2

Background and Related Studies

In this chapter, we discuss the background of our research problem, covering its different parts and key ideas. We also review related works and explain how they connect to our study. We begin with the basics of blockchain, outlining its core concepts, how it works, and why it has emerged as a central focus of research in recent years. This is followed by a discussion on different consensus algorithms, which form one of the most important parts of the blockchain and are directly related to our proposed method. We then explore the use of blockchain in IoT in the next section. In this section, we justify the adoption of blockchain in the IoT environment and discuss different factors enabling that. We also examine various energy optimization methods in IoT from related research that can inspire the design of better consensus algorithms. Next, we discuss the integration of machine learning with blockchain, including how distributed learning can be applied in such systems. In this section, we also discuss the study that has served as the basis for our study. Finally, we review machine learning models relevant to our block producer selection approach, considering both their effectiveness and efficiency in the IoT environment.

2.1 Basics of Blockchain

Before the digital era, traditional paper-based ledgers were the standard method for recording all past transactions in chronological order. With the advent of the digital age, several research efforts explored the concept of distributed financial systems [11]. However, these early attempts often faced limitations, such as reliance on central authorities or vulnerability to issues such as double spending [12]. The introduction of Bitcoin by Satoshi Nakamoto in 2008 marked the first robust and practical implementation of a decentralized digital ledger [13]. It achieved trustless and tamper-resistant

transactions by combining cryptography, consensus mechanisms, and peer-to-peer networking, which is now commonly known as the blockchain [14]. In this section, we explore the basics of blockchain, beginning with its key components, followed by the different types of blockchain, and concluding with the factors that have recently made the technology a major point of attraction.

2.1.1 Key Components of Blockchain

In blockchain, the basic unit of data is called a transaction. Although a transaction most commonly represents the transfer of cryptocurrency, it can also record other types of information, such as contracts, identity data, or sensor readings. These transactions are then collected and grouped together into a block. Each block links to the immediately preceding block through a reference known as the parent block hash, which is the cryptographic hash of the previous block. The very first block in the chain is called the genesis block, and it does not have a parent. Figure 2.1 illustrates the general structure of a blockchain and the contents of a block, based on the Bitcoin blockchain. Although the overall architecture is similar across different blockchain systems, the specific metadata included in each block can vary depending on the particular blockchain protocol being used [15].

2.1.2 Merkle Tree Root

One of the important metadata fields, the Merkle tree root, is a key component of blockchain technology that supports scalability [16]. Without a Merkle tree, every node in the network would need to store a full copy of every transaction ever recorded on the blockchain, even if it does not require direct access to all of them [17]. A Merkle tree organizes transactions in a binary tree, where each leaf node contains the hash of a single transaction, and each parent node contains the hash of its two child nodes. This structure ensures that if even a single transaction is altered, the Merkle root, the top-most hash, changes, enabling efficient verification and integrity checking.

2.1.3 Digital Signature

Digital Signatures (DS) are used in blockchain technology to ensure the authenticity and integrity of transactions [18]. The sender creates a digital signature by first generating an irreversible hash of the transaction, known as the message digest, and then encrypting this digest with their private key. Given current computing capabilities, it is practically impossible to derive the private key from the message digest and the orig-

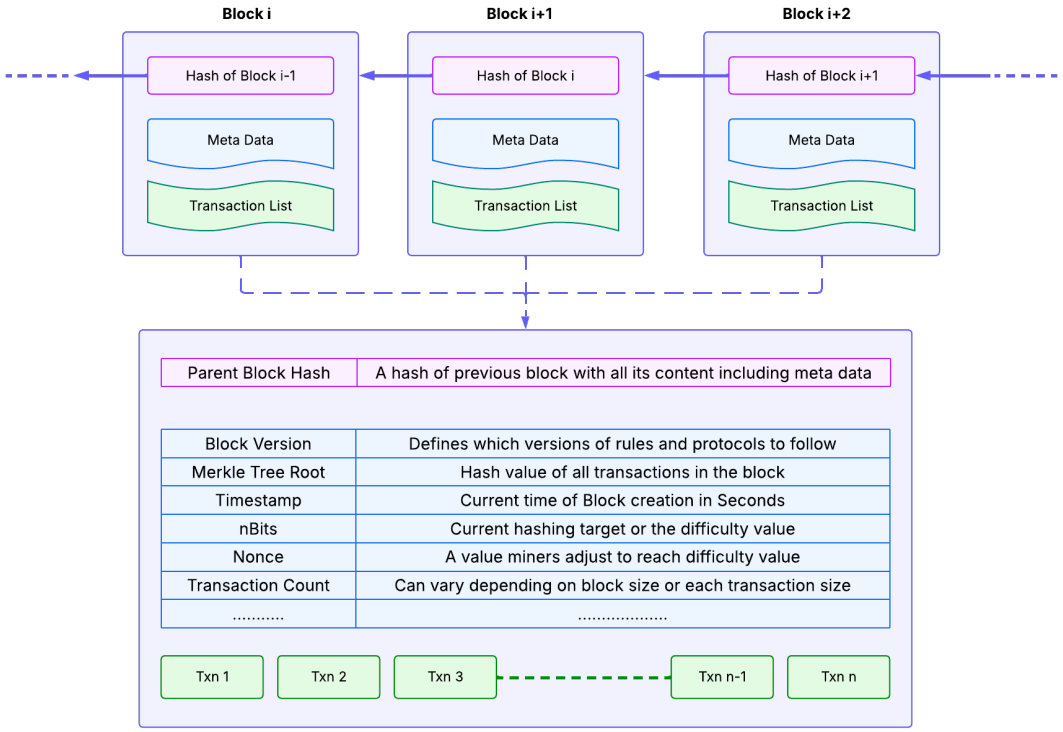


Figure 2.1: Illustration of key components of blockchain. Each block is linked to the previous block through a hash and contains a set of transactions. It also stores metadata that helps maintain the integrity and order of the blockchain.

inal transaction [19]. The receiver verifies the signature by decrypting the digest with the sender's public key and comparing it to a newly computed hash of the received transaction. Beyond the standard digital signature, there are several other signature schemes, including aggregate signatures [20], group signatures [21], blind signatures [22], and proxy signatures [23]. It is important to note that, in most blockchains, transactions themselves are not encrypted and remain visible to all participants; digital signatures serve solely to confirm authenticity and prevent tampering.

2.1.4 Smart Contract

Smart contracts are one of the most important components of blockchain technology. Although they were not introduced in the original Bitcoin paper, the concept was first proposed by Nick Szabo in the 1990s [24] and later popularized and implemented in the Ethereum Whitepaper [25]. Ethereum represents a new kind of blockchain that goes beyond simple peer-to-peer payments by enabling programmable transactions and decentralized applications through smart contracts [26]. Smart contracts are pieces of code that digitally enforce the terms of an agreement, eliminating the need

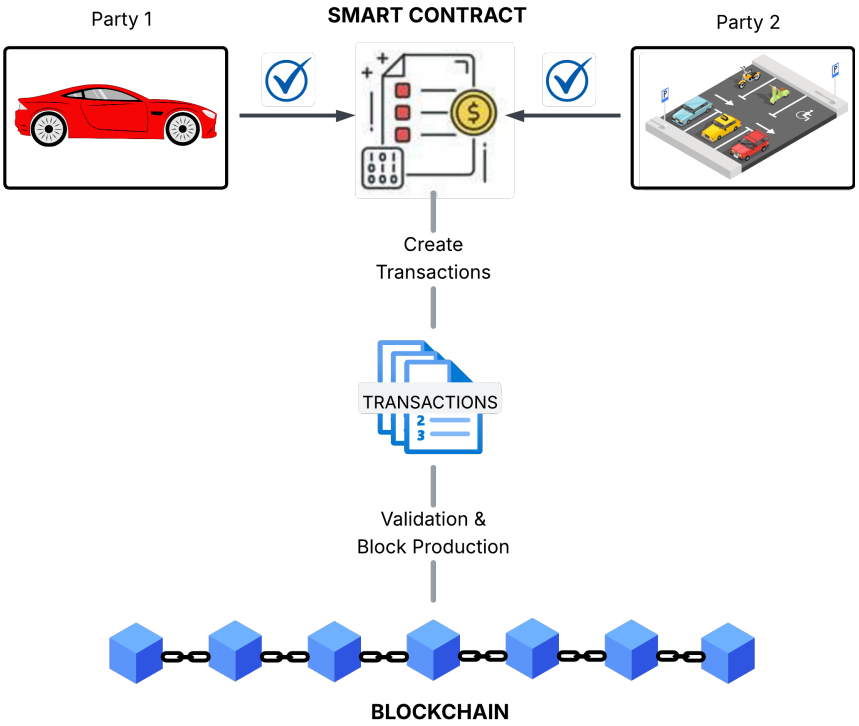


Figure 2.2: Illustration of the role of a smart contract in blockchain. The two parties define the terms through a smart contract, which is executable code that automatically enforces the agreement and records the resulting transaction on the blockchain.

for a trusted intermediary. By running automatically on a decentralized blockchain, they enable transactions between untrusted parties without direct interaction or extra commission costs [27]. However, running smart contracts requires resources, and hence, it is necessary to prevent misuse, such as endless or unnecessary computations. This is where gas becomes necessary. Gas is the unit that measures the cost of executing smart contract operations on Ethereum. Read-only functions (checking account balance) do not consume gas, but functions that change the blockchain do, since they must be recorded in a new block [28].

An example of a smart contract use case is a parking lot that uses the Ethereum blockchain for automated payments [29]. Without smart contracts, the system would require manual intervention to calculate and collect fees when a vehicle leaves. With smart contracts, sensors can detect when a vehicle enters and exits, and the information can optionally be stored on the blockchain. The contract can then automatically calculate the payment based on the duration of the stay and charge the vehicle owner accordingly. Different conditions can also be programmed, such as varying rates for cars, motorcycles, or minibuses. Figure 2.2 illustrates the role of smart contracts in a blockchain. A transaction is added to the blockchain only after it has been processed

by a smart contract agreed upon by both parties. In the parking lot example, the interacting parties are the vehicle and the parking system.

2.1.5 Types of Blockchain

The most common and widely accepted classification of blockchains divides them into public, private, hybrid, and sometimes consortium categories [30], [31], [32]. Our proposed consensus algorithm is mostly suitable for public blockchain, although it can be easily adopted in other categories too. In a public blockchain, anyone can join and participate; transactions are open to anyone to verify, and anyone can initiate transactions and engage in the consensus process. For this reason, public blockchains are often referred to as permissionless blockchains [33]. Examples include traditional blockchains such as Bitcoin and Ethereum. In contrast, a private or permissioned blockchain restricts participation to trusted parties authorized to verify and validate transactions. These blockchains are typically operated by a central authority and are invitation-only. Due to their controlled nature, private blockchains are often argued not to be true blockchains, as they contradict the trustless and open principles fundamental to blockchain technology [34]. Examples of private blockchains include Hyperledger Fabric, MultiChain and Corda.

Public blockchains are designed to tolerate both faulty and malicious nodes, maintaining security even in adversarial environments. Private blockchains generally assume trusted participants and focus on handling faults rather than malicious behavior, although some may include mechanisms to address limited malicious activity [35]. A consortium blockchain, often considered a separate category [36], is a type of permissioned blockchain that is semi-decentralized or semi-private, controlled by a group of entities rather than a single authority. Lastly, hybrid blockchains combine elements of public and private blockchains to provide maximum customization with the aim of achieving specific goals [37].

However, some researchers also divide blockchain into many other types based on different criteria. For example, Uddin et al. [4] mentioned a category of decentralized ledger technologies such as blockchain based on the data structure used. The typical chain-structured blockchains such as Bitcoin, EOS and Litecoin use a linked-list-like data structure to store transactions. On the other hand, graph-structured blockchains use a Directed Acyclic Graph (DAG) to store transactions, which allows transactions to be related to each other directly instead of being bundled in a block. The DAG-based blockchains are considered a significant advancement of blockchain technology due to their high scalability and efficiency, which makes them suitable for IoT applica-

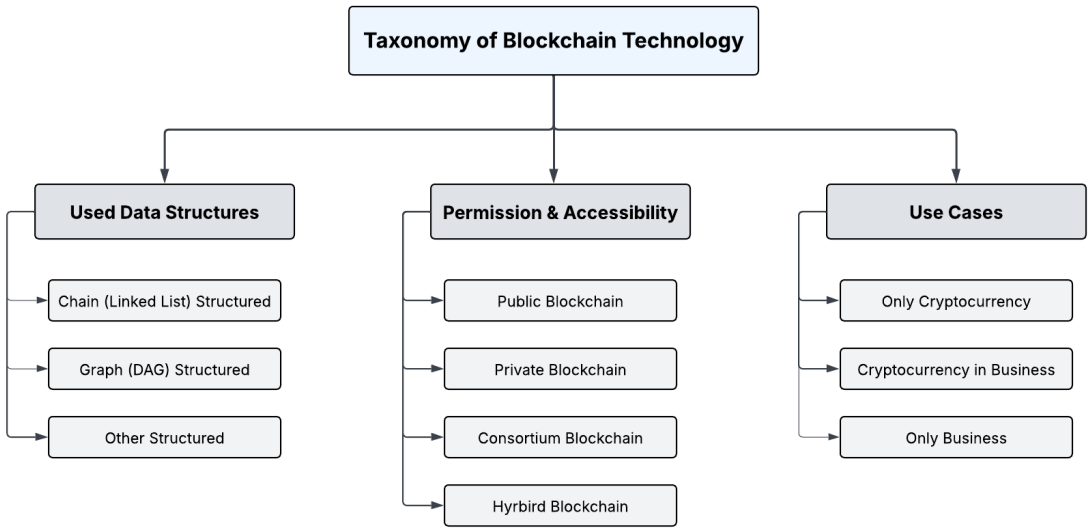


Figure 2.3: Different types of blockchain technology based on various criteria. Data structure refers to how transactions are recorded, permission and accessibility indicate who can participate, and use case describes the applications of blockchain.

tions [38]. In DAG-based blockchains like IOTA, each new transaction must validate at least two previous ones, creating a network of interlinked, mutually verified transactions. Unlike traditional blockchains, DAG does not require miners, enabling faster and more direct transaction confirmation [39]. It is to note that often such DAG based technologies are considered as part of distributed ledger technology (DTL) instead of directly considering them as blockchain.

In another study, blockchain was categorized into three types according to their applications [40]. The first type refers to blockchains used solely for cryptocurrencies, such as Bitcoin. The second type involves business applications where cryptocurrency is combined with smart contracts, as seen in Ethereum, which allows programmable execution. Finally, the third type is focused purely on business use without involving any cryptocurrency but still supports the execution of business logic through blockchain platforms, such as the Hyperledger project. Figure 2.3 summarizes all these types of blockchain technology based on different criteria.

2.1.6 Characteristics of Blockchain

The advent of blockchain technology has brought numerous benefits, many of which stem directly from its core characteristics [41]. Figure 2.4 summarizes these key characteristics and the advantages they provide.

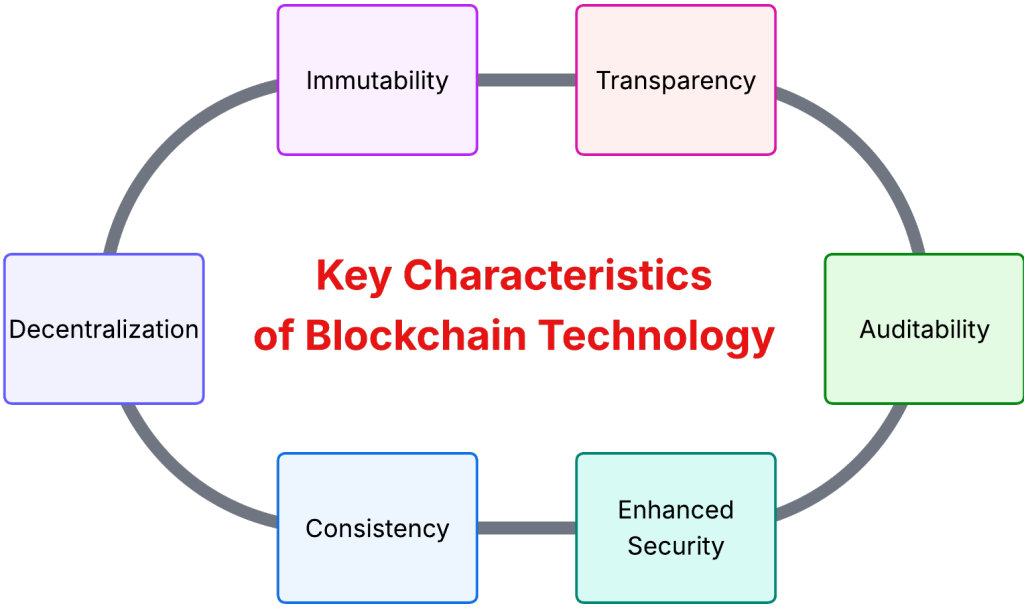


Figure 2.4: Key characteristics of blockchain technology. These features drive blockchain's strengths and support its applications across various domains.

With its decentralized nature, blockchain eliminates the need for a trusted third party, making it suitable for many use cases that require resolving bottlenecks and avoiding a single point of failure. One of the key strengths of blockchain is that the transaction data remain immutable over time. Since each new block contains the hash of its parent block, it becomes practically impossible to alter, update, or remove any recorded data once a block has been validated and added to the chain. This immutability ensures the integrity of the transaction history. In addition, blockchain transactions are transparent and accessible to all network participants, making auditing easier and increasing trust by reducing the possibility of data manipulation. Additionally, blockchain provides pseudo-anonymity, allowing users to interact with the system using their public keys without revealing their real identities.

Blockchain is more reliable and secure than other data storage systems in several aspects [42]. It is resistant to double spending attacks and offers greater resilience to Distributed Denial of Service (DDoS) attacks compared to centralized systems. In blockchain, transactions could potentially be tampered with by block producers, such as miners in Bitcoin, or by adversaries attempting to modify historical data. Blockchain prevents such tampering through the use of digital signatures and the chaining of blocks via parent hashes.

According to the CAP Theorem, any distributed data storage system, such as blockchain, can guarantee only two out of the three properties: Consistency (C), Avail-

ability (A), and Partition tolerance (P) [43]. In the context of blockchain, consistency means that all nodes maintain the same ledger state simultaneously, availability ensures that every request receives a response even if some nodes fail, and partition tolerance means that the system continues to operate despite a network partition that divides nodes into disjoint groups. All blockchain systems provide two of these properties. For example, Bitcoin prioritizes availability and partition tolerance, while consistency is achieved in a weaker form known as eventual consistency [44].

2.1.7 Beyond Cryptocurrency

The recent surge in blockchain research is largely driven by its potential to improve a wide range of traditional systems [45]. These works aim to adopt blockchain in domains beyond cryptocurrency [5]. Figure 2.5 shows some of these domain applications. In healthcare, blockchain is being explored to improve the interoperability of information exchange between organizations while maintaining privacy and security [46]. In supply chains, blockchain enables smart contracts that automatically execute transactions between parties without intermediaries, improving efficiency, reducing costs, and building trust through a shared ledger [47]. In smart cities, blockchain can securely manage and automate services such as energy distribution, traffic management, and public records, ensuring transparency and accountability between stakeholders [48]. In vehicular networks, it provides a decentralized, tamper-proof platform for applications including traffic management, route scheduling, data sharing, parking, and resource exchange [49]. Blockchain has also gained popularity in the realm of non-fungible tokens (NFTs), which are unique digital assets verified and traded on a blockchain [50]. Beyond these areas, blockchain has potential for adoption in fields such as insurance, education, finance, agriculture, e-voting, resource management, among others [51].

Many of these applications are dependent on the adoption of blockchain within IoT systems, as the devices involved are typically IoT devices [52]. These devices must be highly energy efficient, an area where blockchain still faces notable challenges. Balancing blockchain's computational requirements with the limited processing and power capacities of IoT devices remains a critical design concern. As a result, much of the existing research on blockchain focuses on improving its energy efficiency, either in general or for specific use cases. Our work also focuses on this aspect, with the aim of an improvement that could further facilitate the integration of the blockchain into the IoT. As IoT devices and blockchain technology continue to advance, blockchain can be more reliably deployed in a wider range of applications, helping to address

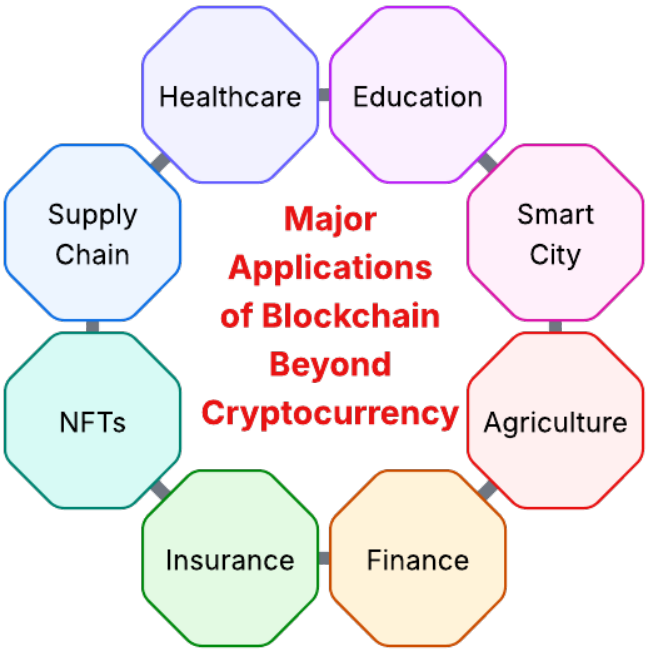


Figure 2.5: Different application domains where blockchain can be deployed to enhance existing systems.

the inherent limitations of conventional IoT networks [53]. Section 2.3 explores the challenges, motivations, and opportunities of blockchain in IoT in more detail.

2.2 Consensus Algorithm in Blockchain

Consensus algorithms are the heart and soul of blockchain. They are one of the core mechanisms that enable the blockchain to achieve the same level of security and trust that centralized systems provide without relying on a central authority [54]. Unlike traditional systems, blockchain is not governed or controlled by a single entity. No central server dictates what state participants must maintain in the ledger. For example, in a centralized financial system, a bank server verifies whether a user has sufficient funds before updating account balances [55]. In blockchain, whether it involves a financial transaction like cryptocurrency or other applications beyond currency, the transactions or data must be recorded in the distributed ledger in a way that the majority of nodes (participants) agree that the information is valid and consistent with the system's rules. This agreement is precisely the responsibility of the consensus algorithm [56]. Consensus algorithms define the process that ensures all network participants agree collectively and consistently on the current state of the blockchain. The

state refers to the ledger that contains all transactions and data. Achieving consensus means agreeing not only on which transactions are valid, but also on their order within blocks and the sequence of blocks themselves. Without this shared agreement, the integrity and reliability of the blockchain would collapse, which is why consensus is one of the most studied components of blockchain [57]. In this section, we begin by discussing the role of consensus algorithms in blockchain and why they are considered the backbone of decentralized systems. We then introduce the main categories of consensus algorithms and examine the key algorithms in each, along with their design, operation, and trade-offs.

2.2.1 The Role of Consensus Algorithm

Figure 2.6 illustrates the role and position of the consensus algorithm within the workflow of a blockchain. The figure depicts the full lifecycle of a transaction, from its origin to its inclusion in the distributed ledger. A total of eight steps are shown, but the consensus mechanism is used in step 5, where it selects a block producer. This selection is critical because the block producer is responsible for validating transactions and creating the next block in the chain. If the chosen block producer is unreliable, malicious, or inefficient, it can lead to delays, rejected transactions, or even compromise the integrity of the ledger. Therefore, the effectiveness and security of the entire blockchain depend heavily on the proper functioning of this step, making the consensus algorithm a central component of blockchain operations [58].

Everything in a blockchain begins with a data exchange between two entities. We refer to them as the originator (sender) and the entity (receiver) to generalize across different blockchain applications. In a typical cryptocurrency system, the originator sends some form of cryptocurrency to the receiver. In more general applications, this can be any type of data, such as sensor readings from IoT devices, where the other entity may be an external user or some kind of devices. Regardless of the type, these data are packaged into a proper transaction format which can be signed by the originator (and optionally by the receiver for certain applications). Once prepared, the transaction is transmitted to the entire network.

Each participant in the blockchain network receives a copy of the transaction and stores it in a local transaction pool after validating its basic correctness [59]. Transactions are not immediately added to a block because the consensus algorithm must first select a new block producer or apply other rules, which may introduce delays. Once the consensus mechanism selects the next block producer, this producer bundles the transactions in an orderly manner into a new block and broadcasts this block

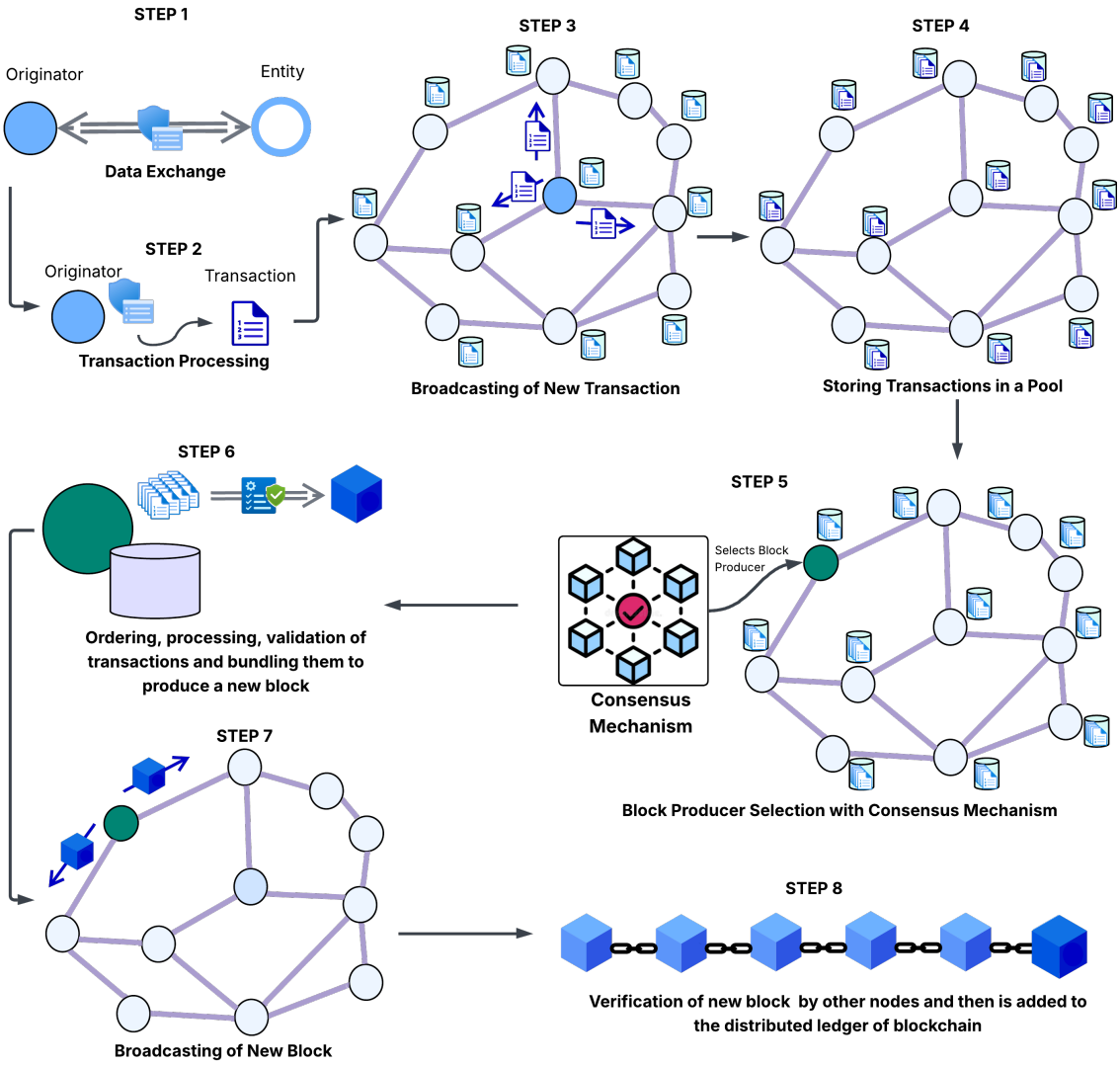


Figure 2.6: Illustration of the role of a consensus algorithm in blockchain with the transaction lifecycle. After a data exchange between parties, the originator packages it as a transaction and broadcasts it to the network. Nodes store the transaction in their local pools. The consensus algorithm selects a block producer, who validates and groups transactions into a block with the necessary metadata. This block is then broadcast to the network, verified by other nodes, and added to their ledgers.

to the network. Network participants then verify that the block has been properly constructed and, upon successful validation, add it to their own ledgers. This cycle continues for each new set of transactions, ensuring a continuously updated distributed ledger.

In most proof-based consensus algorithms, which will be discussed in Sections 2.2.3 and 2.2.4, the overall workflow follows the same pattern shown in the figure, with the key difference being how the consensus mechanism in step 5 selects the block

producer. Our work also focuses on modifying this part of the blockchain process while keeping the rest of the workflow unchanged.

A fork in blockchain occurs when the chain diverges into two potential paths, usually because two miners (the term used for block producers in Bitcoin) produce blocks at the same time or due to intentional protocol changes [60]. Although forking is not a concern for our proposed mechanism, it is worth mentioning because of its relevance to consensus mechanisms. Forks can be temporary (soft forks), occurring when multiple blocks are added to the chain simultaneously and resolved when the network converges on a single chain, or permanent (hard forks), resulting from intentional changes to the protocol rules that create a lasting divergence. A hard fork introduces non-backward-compatible changes that require all nodes to upgrade, often resulting in a split into separate blockchains (e.g., Bitcoin and Bitcoin Cash). In contrast, a soft fork is backward-compatible, allowing upgraded and non-upgraded nodes to coexist, though temporary chain splits can still occur.

2.2.2 The Types of Consensus Algorithms

Since the introduction of the Proof of Work (PoW) consensus mechanism by Satoshi Nakamoto in the original Bitcoin paper [13], a wide variety of consensus algorithms have been developed. However, no single consensus mechanism can be considered a strict improvement over others in all scenarios. There is no consensus algorithm that is optimal for every possible application but only the most suitable for a given application scenario [9]. Each consensus mechanism comes with its own strengths and weaknesses, making it suitable for certain use cases, while less appropriate for others. This diversity reflects the fact that different blockchain systems vary greatly in their requirements, such as performance, security, scalability, and governance. As a result, researchers and developers have created consensus algorithms that target specific goals or environments. Some are designed to address broad distinctions, such as public versus private blockchains, while others are tailored to highly specific domains, such as healthcare [61].

There is no globally accepted standard for categorizing consensus algorithms. Different researchers have proposed various classifications depending on their focus and requirements [62]. Nguyen and Kim [63] divided blockchain consensus into two main types: proof-based and voting-based. In proof-based consensus, nodes demonstrate their eligibility by performing tasks related to block production, while in voting-based consensus, nodes vote on which block or transaction should be accepted, and the majority decision determines the accepted state of the blockchain. Fu et al. [64]

Table 2.1: List of Different Types of Consensus Algorithms

Category	List of Some Consensus Algorithms
Proof Based Traditional	PoW[13], PoS[66], DPoS[67], PoUW[68], PoQ[69], PoA[70], PoAu[71], PoAh[72], PoET[73], PoCapacity[74], PoSpace[75], PoPF[76], PoB[77], PoI[78], PoH[79], PoLuck [80], PoR[81], PoV[82]
Proof Based Modern	DT-DPoS[83], CloudPoS[84], LPoS[85], PPoS[86], CPoA[87], GPoA [88], CPoAu[89], HDPoAu[90], DC-PoET[91], B-LPoET[92], PoPT[93], PoWeight[94], DPoI[95], PoP [96], PoEM [10]
Byzantine Fault Tolerance	PBFT[97], Adapting-PBFT[98], G-PBFT[99], SG-PBFT[100], T-PBFT[101], Dynamic-PBFT[102], SDMA-PBFT[103]
Miscellaneous	RAFT[104], Groupchain[105], Hyperledger Fabric[106], Ripple[107], Hashgraph[108], Tendermint[109]

proposed a four-category classification: ledger-based, voting-based, committee-and-voting-based, and fair-accounting-based consensus. Meanwhile, Zhou et al. [65] suggested another fourfold classification: crash fault tolerance (CFT), Byzantine fault tolerance (BFT), proof-of-something (PoX), and hybrid consensus.

To provide a clearer understanding of the mechanisms relevant to our consensus mechanism, we organize consensus algorithms into four main categories, as discussed in Sections 2.2.3–2.2.6. Table 2.1 presents a list of the consensus algorithms discussed, organized by different categories. Since our proposed mechanism is proof-based, we focus primarily on this category and further divide it into two subcategories: traditional and modern. We define traditional proof-based algorithms as those that introduce a distinct mechanism for selecting the block producer, whereas modern algorithms represent modifications or enhancements of these traditional approaches. For completeness, we also include two additional categories: Byzantine Fault Tolerance (BFT) mechanisms and miscellaneous approaches that do not fit neatly into any of the other three categories.

2.2.3 Proof Based Traditional Consensus Algorithms

The Proof of Work (PoW) mechanism [13] selects the block producer through a computational effort called mining. Miners compete to solve a cryptographic puzzle,

which typically involves finding a nonce (Number used ONCE) such that the hash of the block header meets a network defined difficulty target, such a certain number of leading zeros in the hash. This requires repeated hashing and trial and error, making block production probabilistic and resource intensive. A miner cannot begin calculating the nonce for future blocks in advance, because it first requires the exact hash of the immediately preceding block. The first miner to solve the puzzle gains the right to add the next block and collect the associated block reward and transaction fees. The requirement for substantial computational effort makes rewriting the blockchain extremely costly, and feasible only if an entity succeeds in controlling at least 51% of the total computing power, which is practically impossible.

Proof of Stake (PoS) [66] selects the block producer based on the amount of cryptocurrency a validator locks, or stakes, in the system. Validators are chosen pseudo randomly, with higher stakes increasing the probability of selection. Misbehaving validators, for instance those who double sign or attempt invalid transactions, can lose part of their staked coins through slashing, aligning economic incentives with network security. Delegated Proof of Stake (DPoS) [67] refines PoS by introducing a governance layer in which stakeholders vote for a small set of delegates who then take turns producing blocks. This approach reduces the number of active block producers, increasing throughput and reducing latency. DPoS protocols often include mechanisms to rotate delegates periodically and penalize inactivity or misconduct, further ensuring security and incentivizing consistent participation.

Other variants focus on useful contributions beyond raw computation or stake. Proof of Useful Work (PoUW) [68] ensures that the computational effort performed has practical applications, while Proof of Quality of service (PoQ) [69] selects block producers based on the quality of resources or service they provide. Proof of Activity (PoA) [70] combines PoW and PoS: miners first solve a PoW puzzle to propose a block, then the network uses PoS to choose which miners actually finalize it. Proof of Authority (PoAu) [71] and Proof of Authentication (PoAh) [72] rely on the identity or reputation of the validators rather than computational power or stake, often suitable for permissioned networks [110].

Several mechanisms leverage timing or resource contribution. These include Proof of Elapsed Time (PoET) [73], where a trusted execution environment generates a randomized wait time and the node with the shortest time produces the block. Proof of Capacity (PoCapacity) [74] and Proof of Space (PoSpace) [75] use allocated storage space as the resource for selection, rewarding nodes that commit larger amounts of disk capacity. Similarly, Proof of Bandwidth (PoB) [77] rewards nodes based on net-

work contribution or fairness metrics. Proof of Participation and Fees (PoPF) is the initial consensus algorithm used in JcLedger (Joint Cloud ledger) [111].

Other interesting variations include Proof of Importance (PoI) [78], which combines stake, transaction activity, and network reputation to determine block producers. Proof of History (PoH) [79] encodes the passage of time cryptographically, allowing nodes to efficiently verify the ordering of events and the selection of leaders. Proof of Luck (PoLuck) [80] uses a trusted hardware random number generator to select the block producer, whereas Proof of Retrievability (PoR) [81] requires the nodes to prove that they can retrieve stored data correctly before being allowed to create blocks. Proof of Vote (PoV) [82] integrates voting mechanisms into the selection, choosing validators based on the accumulated votes in the system.

Collectively, these traditional proof-based mechanisms illustrate the diversity of approaches to consensus. Although PoW and PoS remain the most widely known, the range of mechanisms demonstrates how networks can balance security, fairness, efficiency, and resource utilization. Many modern variants are built based on these traditional ideas by improving security, reducing energy consumption, or integrating additional metrics for block selection.

2.2.4 Proof Based Modern Consensus Algorithms

Several innovative approaches that are based on traditional proof-based concepts have been developed to meet the evolving demands of decentralized systems. DPoS has been enhanced into Dynamic Trust DPoS (DT-DPoS) [83], which incorporates a dynamic trust model to improve security and flexibility. CloudPoS [84] is a proof-of-stake–based consensus protocol that leverages cloud users' infrastructure to securely record data operations occurring in the cloud environment. Leased PoS (LPoS) [85] enables token holders to lease their coins to validators, allowing for passive participation in the network's consensus process. Periodic PoS (PPoS) [86] is a consensus algorithm for lightweight applications that introduces the concept of delaying transaction broadcast. Transactions are first multicast to a group of supervisor nodes. Then these nodes periodically forward the transactions to validators to produce blocks.

Credit-based Proof of Activity (CPoA) [87] enhances traditional PoA by using a credit reward system, multi-level node selection, and workload differentiation to incentivize honest nodes, penalize malicious behavior, and reduce the likelihood of fraudulent block creation. Similarly, Game Theory-based PoA (GPoA) [88] is another efficient version of PoA that relies on game theory to deal with selfish mining and majority

attacks. Concurrent Proof of Authority (CPoAu) [89] selects authority nodes based on reputation, security score, online age, and performance metrics, ensuring reliable validation of identities, blocks, and transactions while reducing misbehavior and improving the security and confidentiality of the blockchain. Honesty-based Distributed PoAu (HDPoAu) [90] focuses on honesty-based validation and also integrates PoW to take advantage of the collective computational resources of IoT.

B-LPoET [92] is a lightweight PoET-based consensus for private permissioned blockchains that reduces multi-node overhead, improves scalability, and allows participants to verify identities before joining, outperforming traditional PoET in transaction validation and block creation. Proof of Previous Transaction (PoPT) [93] is an improvement of PoPF used for the JCLedger blockchain. Proof-of-Weight (PoWeight) [94] is a consensus mechanism that assigns influence based on user weight, typically determined by coin holdings, allowing high-speed transactions and fork resistance. It is better suited for creative projects than commercial networks due to limited user incentives. Dynamic PoI (DPoI) [93] dynamically delegates block creation based on the calculated importance of each node, considering factors such as activity, trading, longevity and credit.

Proof of Play (PoP) [96] and Proof of Evolutionary Model (PoEM) [10] are two unique consensus algorithms that, although not direct improvements on existing approaches, are considered modern due to their innovative contributions to the evolving landscape. In multiplayer P2P games, players typically share real-time game data through a central server, causing high latency (often termed as ping) for distant players [112]. PoP addresses this by providing a blockchain-based consensus that does not require a central server and minimizes interference while enabling a secure, fully decentralized, and community-sustainable multiplayer environment. PoEM is one of the first attempts to integrate machine learning with blockchain consensus algorithms. We discuss their mechanisms in detail in Section 2.4.4 since it is closely related to our proposed approach.

2.2.5 Byzantine Fault Tolerance Consensus Algorithms

The concept of Byzantine Fault Tolerance (BFT) originates from the Byzantine Generals Problem, introduced by Leslie Lamport in 1983 [113]. In a distributed system, some components may malfunction or act maliciously, sending inconsistent or misleading information to others. This challenge is illustrated by the Byzantine Generals Problem: a group of generals of the Byzantine army are camped around an enemy city and must agree on a common battle plan, communicating only via messengers. Some

generals may be traitors, attempting to disrupt consensus. The goal is to design a strategy that ensures all loyal generals can agree on the same plan despite the presence of traitors. It has been shown that, with oral messages alone, consensus is possible only if more than two-thirds of the generals are loyal. [114]. BFT algorithms manage this by requiring multiple rounds of communication and cross-verification among nodes, ensuring that the influence of malicious actors is limited and consensus is reached only when a sufficient majority of honest nodes agree.

This concept has been applied in blockchain to develop consensus algorithms that defend against faulty or malicious nodes in highly secure systems. However, mechanisms such as Practical BFT (PBFT) [97] incur high overhead due to computational complexity as the number of nodes increases. G-PBFT [99] leverages the geographical location of fixed IoT devices to reach consensus, but its average latency is too high for real-time IoT applications. Similar limitations affect other PBFT variants, including Score Grouping PBFT (SG-PBFT) [100], Adapting-PBFT [98], EigenTrust-based PBFT (T-PBFT) [101], Dynamic-PBFT [102], and Scalable-Dynamic-Multi-Agent PBFT (SDMA-PBFT) [103]. Overall, these algorithms are most suitable in highly adversarial environments where safety and fault tolerance are prioritized over efficiency. Another similar consensus approach is Crash Fault Tolerance (CFT), which focuses only on handling nodes that fail by crashing or stopping, making it more efficient but unable to deal with malicious or inconsistent behavior [115].

2.2.6 Miscellaneous Consensus Algorithms

Several blockchain platforms implement consensus mechanisms designed for efficiency and scalability in permissioned or specialized networks. RAFT [104] uses the RAFT protocol, a leader-based consensus suitable for fast and reliable ordering of transactions in small networks. Groupchain [105] organizes nodes into groups to achieve consensus efficiently, improving throughput while maintaining fault tolerance. Hyperledger Fabric [106] separates transaction endorsement from ordering and uses a pluggable consensus model, allowing RAFT or Kafka-based ordering services to validate and commit transactions deterministically.

Other platforms focus on achieving fast, low-latency consensus in more decentralized environments. Ripple [107] relies on a consensus protocol among a trusted set of validators to confirm payments quickly and prevent double-spending. Hashgraph [108] uses a gossip-about-gossip protocol combined with virtual voting to achieve asynchronous Byzantine Fault Tolerant (aBFT) consensus without a blockchain. Tendermint [109] implements a Byzantine Fault Tolerant consensus that deterministically

orders transactions with high security and low latency, making it suitable for both public and private blockchains.

2.3 Integration of Blockchain in IoT

The Internet of Things (IoT) has become a transformative force in the ongoing quest to improve human life [116]. In this context, the term "things" means a wide range of connected entities, from small sensors to large-scale infrastructure, all working to connect electronic and electrical components into intelligent networks [117]. IoT devices rely on sensors to gather data and create a seamless link between the physical and digital worlds. In recent years, researchers and entrepreneurs have invested heavily in IoT, developing innovative services for a variety of applications [118]. However, the enormous volumes of data generated by these devices often overwhelm traditional centralized systems, causing performance bottlenecks and exposing security vulnerabilities [119]. The integration of blockchain technology into IoT presents a promising approach to these challenges, addressing issues such as single points of failure and insecure data management [120]. The decentralized peer-to-peer architecture of blockchain helps to reduce bottlenecks, strengthen security, and improve Quality of Service (QoS) in IoT networks [121]. In this section, we discuss the challenges of adopting blockchain in IoT and the motivations for overcoming them. We also highlight the current state-of-the-art studies on blockchain integration in IoT and consider future directions. Finally, we conclude with two relevant topics related to this integration: Blockchain as a Service (BaaS) and energy optimization methods.

2.3.1 Challenges of Adopting Blockchain in IoT

Several studies have identified various technical limitations of blockchain that hinder its adoption in IoT systems, which are often constrained by limited storage capacity and strict power consumption requirements. Any work similar to ours with the aim of integrating blockchain IoT needs to carefully assess these issues. Figure 2.7 highlights some of these challenges reported across various studies.

In their survey, Uddin et al. [4] highlighted a range of these challenges. One key issue is the unavoidable trade-off between power consumption and security when processing IoT data. For example, the Proof-of-work consensus for Bitcoin requires miners to compete in solving complex computations, resulting in extremely high electricity usage. This consumption has even exceeded the total electricity usage of some countries such as the Czech Republic, Austria, and Colombia [122]. Although this energy

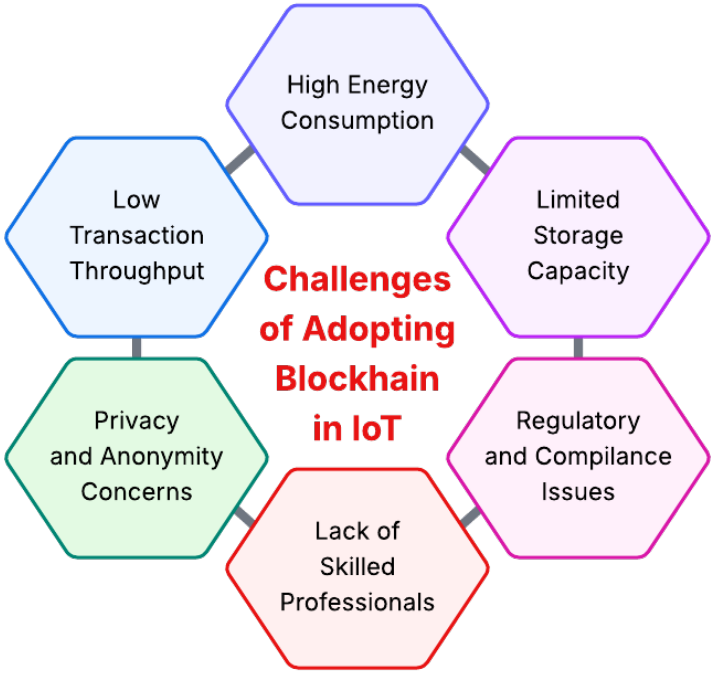


Figure 2.7: Challenges of Adopting Blockchain in IoT

cost is significant, it is also the reason PoW provides strong protection against Sybil attacks and ensures that blocks remain resistant to tampering. This energy challenge is closely related to another major concern, which is the performance of the system. A reliable IoT system must maintain high throughput to avoid delays in transaction processing. However, slower consensus algorithms can increase the time required for transactions to be confirmed, leading to reduced throughput.

In another study, Reyna et al. [123] pointed out that the limited storage capacity of IoT devices, combined with the high storage requirements of blockchain, makes it difficult to handle large-scale big data effectively. Privacy concerns are also another significant challenge in this regard. The transparency of blockchain can increase trust, but it can also compromise confidentiality because it offers only pseudo-anonymity. Achieving true anonymity requires the integration of additional privacy-preserving techniques, which can be inefficient or impractical in blockchain-based IoT systems. The technical challenges are further compounded by human and regulatory factors. As noted by Atlam et al. [124], the lack of skilled professionals in the relatively new blockchain domain creates a barrier to its adoption in IoT. All of these studies have warned of the uncertainty surrounding regulations and compliance. Although the absence of a central authority can be seen as an advantage, it can also allow fraudulent

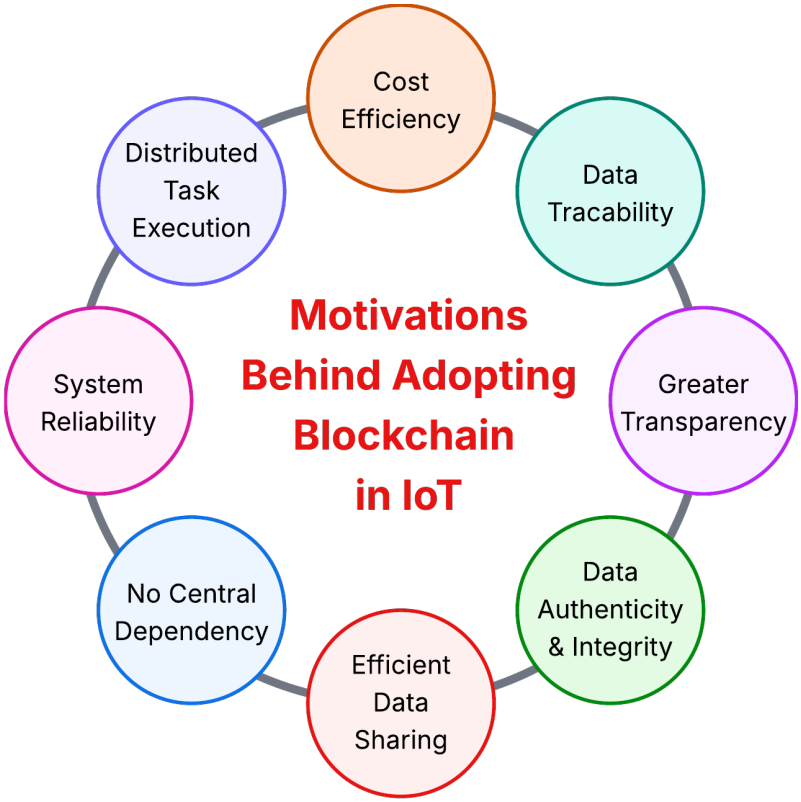


Figure 2.8: Motivations Behind Adopting Blockchain in IoT

transactions, scams, and other illegal activities to occur more easily than in systems governed by strict regulations.

2.3.2 Motivations Behind Adopting Blockchain in IoT

Blockchain has emerged as a technology that eliminates the need to trust conventional centralized authorities and, more broadly, removes the need for a central authority. However, this does not make the blockchain completely trustless. Rather, it can be viewed as a confidence machine that strengthens trust in the operations of a system between independent entities, indirectly reducing the need for traditional trust [125]. With these characteristics, blockchain can address the single point of failure problem, serve as an effective means of storing and processing IoT data, and mitigate various security issues present in centralized IoT systems. These benefits are presented in Figure 2.8.

In traditional IoT, the central server typically handles all computing tasks, which can lead to a system-wide failure if it stops working. In addition, such systems are vulnerable to malicious attacks like DDoS, Sybil attacks [126]. Blockchain enables the

replacement of centralized servers with edge server clusters, allowing distributed task execution in collaboration with IoT devices [127]. Cost reduction is a key objective for organizations seeking to scale up their operations. The expenses associated with infrastructure, maintenance, networking equipment, and intermediaries make the deployment of current IoT applications considerably costly. Blockchain has the potential to save money in these cases [128]. Certain applications, such as tracking goods in a supply chain or managing patient medical records, require the ability to trace data back to its origin. Blockchain facilitates the rapid verification of historical data while ensuring its authenticity and integrity [4]. The transparency of blockchain further encourages adoption in IoT systems, enabling connected networks to share information efficiently while maintaining privacy, which can be important in systems such as hospital management. In addition, blockchain provides benefits such as immutability, tamper resistance, and a degree of anonymity.

2.3.3 Current State of Blockchain in IoT

Blockchain technology has the potential to address a range of challenges related to security, privacy, and scalability. However, these benefits come with the difficulty of managing the vast amounts of data generated by IoT sensors when processed on-chain. In addition, the confidentiality of transactions is not fully guaranteed in public blockchains, as transaction patterns can often be analyzed to reveal sensitive information [129]. To mitigate these issues, numerous solutions have been proposed across different domains, including healthcare, smart cities, vehicular networks, supply chain and agriculture. Table 2.2 summarizes some of these recent studies in various fields.

A number of innovative studies have been carried out to enhance the applicability of blockchain in the healthcare sector. These efforts primarily focus on medicine and drug traceability, remote patient monitoring, and the management of electronic medical records [130]. Ismail et al. [131] proposed a cluster-based blockchain architecture to reduce the computational and communication overhead in healthcare data management. They introduced the concept of canal to facilitate the sharing of confidential transactions with defined groups. In another study, Drugledger [132] was introduced to improve drug traceability and regulation by expanding UTXO workflow of blockchain, thus enhancing performance in lightweight systems. Similarly, Srivastava et al. [133] developed a scalable blockchain framework for remote patient monitoring by employing the GHOSTDAG protocol, a transaction confirmation mechanism that structures each transaction as a node within a directed acyclic graph rather than rely-

Table 2.2: Recent Studies Focused on Adopting Blockchain in IoT

Ref	Domain	Description
[131]	Healthcare	A clustering-based lightweight network architecture for healthcare management using blockchain.
[132]	Healthcare	An efficient framework to improve drug traceability and regulation using an expanded UTXO workflow.
[133]	Healthcare	A scalable blockchain framework for remote patient monitoring using the GHOSTDAG protocol.
[134]	Smart City	A lightweight P2P network to securely store IoT data on cloud storage.
[135]	Smart City	A blockchain-based framework to improve cybersecurity in smart cities.
[136]	VANET	A blockchain- and IoT-based framework to connect waste collection vehicles.
[137]	Agriculture	A BC-IoT food supply chain management system to monitor food production and processing.

ing on a single linear chain of blocks.

Blockchain has significant potential to improve the efficiency and reliability of services in smart cities. A smart city is an urban area where residents are connected through an integrated network of computerized systems that collect, process, and analyze various types of data to improve quality of life and provide a wide range of efficient services. Existing smart city systems often rely on centralized servers controlled by third parties, giving residents little or no control over their personal data [138]. To address this issue, several frameworks have been proposed. Paul et al. [134] developed a lightweight peer-to-peer network with low computational cost, in which the communication of each IoT device is treated as a transaction and securely stored in cloud storage. In another study [135], a blockchain-enabled approach was introduced to improve the cybersecurity of smart cities while minimizing the overhead associated with blockchain integration.

Vehicular Ad-hoc Network (VANET) is another domain that is currently exploring the integration of blockchain with IoT devices. Saad et al. [136] proposed a VANET framework using IoT devices to connect waste collection vehicles in urban areas. Blockchain was used to secure communication between vehicles and make the system more trustworthy. They also presented practical results by deploying actual vehicles in Karachi, Pakistan. In agriculture, a food supply chain management architecture

was proposed that integrates IoT devices with blockchain to address storage, scalability, and security challenges in existing mechanisms [137]. Using this framework in agriculture facilitates remote monitoring of food production at every stage along the supply chain, ultimately improving food quality, reducing food waste, and providing consumers with better traceability to ensure safe, high-quality and sustainable food.

2.3.4 Future of Blockchain in IoT

Although hundreds of studies have explored blockchain adoption in IoT, real-world implementations remain relatively limited [139]. This low adoption at the organizational level is influenced by several factors. Dehghani et al. [140] conducted a hands-on study, including interviews with several North American organizations, and identified both negative factors that hinder adoption and positive factors that could encourage future uptake. Figure 2.9 illustrates the seven key factors identified in this study.

One major challenge is that blockchain is still a relatively new and evolving technology. Its continuous development and frequent changes make it appear volatile and immature. Moreover, most organizations lack the necessary technical expertise, and hiring new personnel to bridge this gap increases costs, discouraging adoption. Uncertainty in the legal and regulatory status of blockchain also plays a major role. Many organizations fear that a single lawsuit could threaten their entire business. In addition, the absence of established standards further hampers adoption. Unlike sectors with well-defined models, such as the OSI model for network communication, blockchain lacks universal guidelines. As a result, organizations often create their own standards, leading to fragmentation, limited interoperability, and difficulties in coordinating future efforts.

Blockchain technology can be attractive because it improves the quality of data. In sectors such as supply chain management, products can be tracked at every stage of the process, ensuring better auditability and reliability of data. When organizations believe that their systems can easily exchange and integrate information with blockchain, they gain confidence in adopting it. Higher perceived interoperability lowers integration complexity and further strengthens adoption intentions. In addition, blockchain offers properties such as transparency, immutability, and decentralization that enhance trust and collaboration within business networks. By ensuring that all participants share a single consistent repository, blockchain fosters stronger cooperation and accountability. Together, these factors positively influence the intention of organizations to adopt blockchain.

Figure 2.9: List of factors influencing an organization's intention to adopt blockchain. Positive factors encourage adoption, while negative factors create challenges or barriers for organizations.

In addition, traditional blockchain, with its high computational costs and limited scalability, is not well suited for the IoT environment. However, the future of blockchain in IoT could be transformed by the technology introduced by IOTA (Internet of Things Application) [141]. IOTA is a next-generation distributed ledger that overcomes the limitations of traditional blockchain, offering virtually unlimited scalability for IoT applications. IOTA uses a Tangle, a directed acyclic graph (DAG), instead of a blockchain to store transactions. Each transaction acts as a vertex in the graph, and when a new transaction is issued, it must validate two previous transactions. Network participants perform lightweight proof-of-work (PoW) to confirm transactions and prevent spam or attacks. Unapproved transactions, called tips, are selected using algorithms like Uniform Random Tip Selection or Weighted Random Walk to ensure fair validation. Transactions gain cumulative weight as they are approved directly or indirectly, which helps prioritize newer and more important transactions. This structure allows the Tangle to scale efficiently for IoT devices, eliminates the need for mining, and requires much less PoW compared to traditional blockchains [142]. Another important benefit of IOTA is that its signature scheme is quantum-resistant. Unlike the digital signatures used in Bitcoin and many traditional blockchains, which could potentially be broken by sufficiently powerful quantum computers, IOTA uses the Winternitz one-time signature scheme [143], which is designed to remain secure against quantum attacks [144].

Whether it is IOTA or another form of blockchain technology, it is inevitable that blockchain will eventually harness its potential in real-world IoT environments, given the immense benefits it offers [145]. Many transformative technologies in the past experienced slow adoption due to a limited understanding of their benefits or arriving slightly ahead of their time. Deep neural networks in machine learning provide a

clear example of this phenomenon [146]. The future of blockchain and its integration with IoT depends on the efforts of innovative researchers. Although their initial contributions may appear modest, over time these incremental advances will collectively unlock the full potential of the technology, ultimately making it a cornerstone for enhancing quality of life in the interconnected world of IoT.

2.3.5 Blockchain as a Service for IoT

With no established standard, the barrier to developing blockchain-based applications remains relatively high [147]. One of the earliest public-sector adopters was the United Arab Emirates government for the city of Dubai and the U.S. state of Illinois, but their approaches differed. Dubai aimed to build a unified system, while Illinois focused on creating a platform for individual pilot projects [148]. Integrating a full blockchain framework into IoT systems on one's own can be cumbersome, as it requires a dedicated infrastructure. Blockchain as a Service (BaaS) addresses this challenge by providing ready-made solutions through cloud, IoT and edge platforms. It offers capabilities such as network deployment, system monitoring, smart contract testing, and consensus management, enabling developers to focus on business logic instead of infrastructure. Similar to cloud services, BaaS handles resources, bandwidth, and connectivity while outsourcing the technical overhead to providers. This approach accelerates adoption by making blockchain application development, testing and deployment more efficient and accessible [149].

As with most breakthrough technologies, the major tech giants have quickly moved to offer BaaS solutions. Alibaba's BaaS [150], built on Kubernetes, supports multiple frameworks such as Hyperledger Fabric, Enterprise Ethereum, Quorum, and its own Ant Blockchain. IBM's platform [151], based on Bluemix and Hyperledger Fabric, prioritizes enterprise-grade security and scalability while simplifying smart contract development through Hyperledger Composer. Microsoft's Azure BaaS [152] accommodates various protocols and introduces Cryptlets for secure interoperability between cloud, middleware, and client applications. Oracle's blockchain cloud service [153] , powered by Hyperledger, focuses on performance and security, offering APIs and automated management tools. Amazon's Managed Blockchain [154] uses a modular architecture that enables members to build, manage, and scale decentralized networks with Hyperledger Fabric. Besides these, there have been many independent efforts by researchers to develop BaaS platforms to allow businesses to deploy blockchain technology with minimal effort [155].

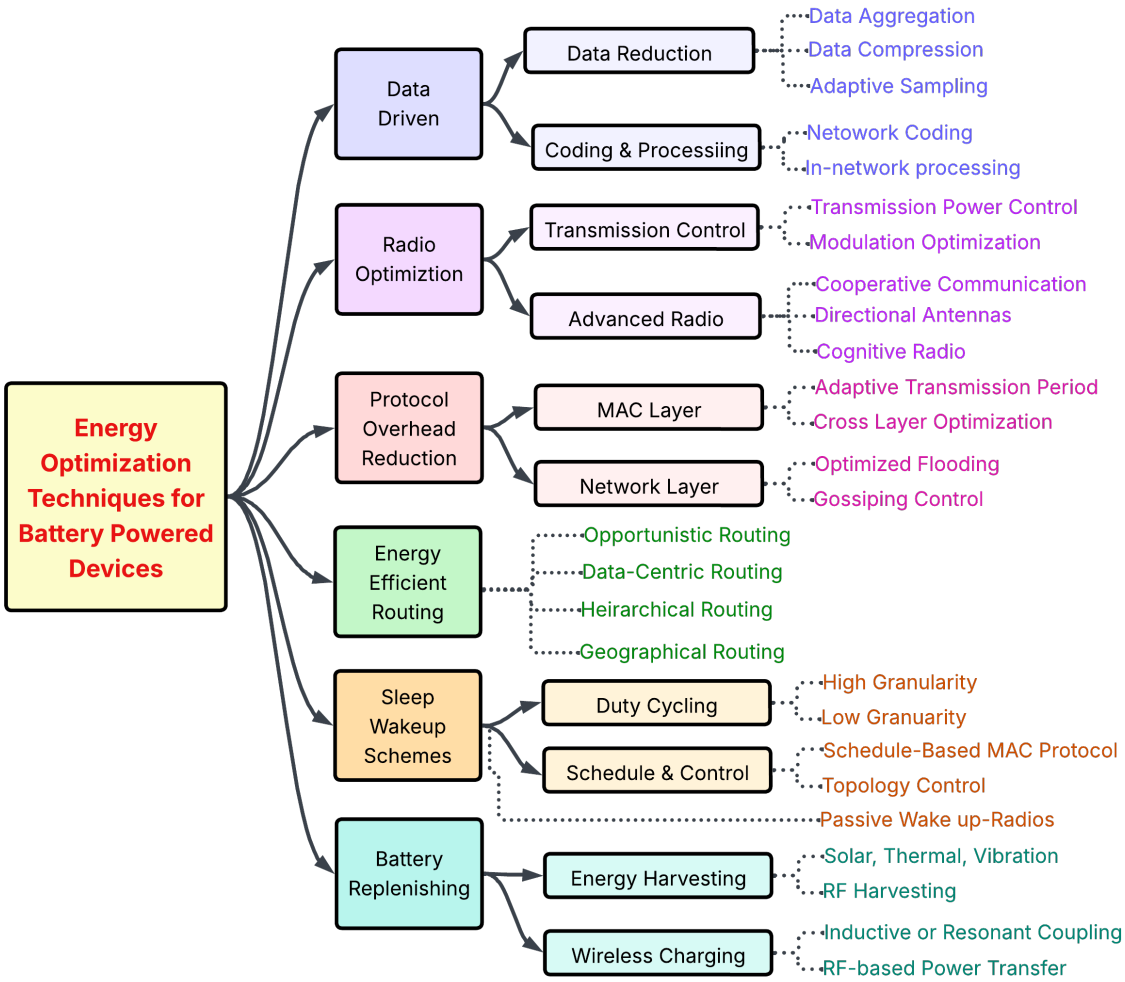


Figure 2.10: Broad hierarchical categories of energy optimization methods for networks of battery-powered devices. Many of these methods are applicable to other fields that aim to optimize energy efficiency.

2.3.6 Energy Optimization Methods

Since the inception of battery-powered devices, researchers have explored various approaches to reduce energy consumption, improve efficiency, extend battery life, and optimize overall performance [156]. One of the first forms of such networks is the Wireless Sensor Network (WSN), composed of distributed sensor nodes that communicate wirelessly and collect environmental data such as temperature, motion and pollution levels [157]. More recently, these battery-powered devices have expanded into IoT networks, along with other specialized forms such as Vehicular Ad Hoc Networks (VANETs), Mobile Ad Hoc Networks (MANETs), Delay Tolerant Networks (DTNs), and Low Power Wide Area Networks (LPWANs). The energy optimization methods applied in these networks are broadly categorized in Figure 2.10. Since our work

focuses on battery-powered devices such as IoT, it is important to briefly highlight these methods, as a consensus algorithm designed for IoT could potentially incorporate them.

The energy optimization methods illustrated in the figure have been collected and systematically organized into a hierarchical framework based on several studies [158], [159], [160], [161]. Among the various techniques used in battery-powered networks, widely adopted approaches include data aggregation and adaptive sampling, which reduce communication overhead by combining or selectively sampling sensor data prior to transmission. Duty cycling alternates nodes between active and sleep states to reduce idle power consumption. Hierarchical routing, such as cluster-based schemes, mitigates redundant transmissions by organizing nodes into groups. Energy harvesting techniques, including solar or vibration-based methods, capture ambient energy to replenish batteries and extend the device lifetime. Collectively, these methods improve energy efficiency, minimize waste, and prolong the operational lifespan of battery-powered networks. Further details can be found in the referenced articles.

2.4 Integration of ML and AI in Blockchain

In the 21st century, machine learning (ML) and artificial intelligence (AI) have become increasingly integrated into nearly every aspect of human life. Their applications range from highly sensitive domains such as medical procedures and disease detection to areas such as e-commerce, healthcare, smart cities, and cybersecurity [162]. What began in the 1960s with relatively simple pattern recognition tasks such as letter identification [163] has now advanced to solving highly complex problems that would take humans days to complete, or may even be impossible. This progress has been made possible through improvements in ML algorithms, the availability of large datasets, and the growth of computing power [164]. The Internet of Things (IoT) already applies ML solutions to strengthen security, optimize storage, and address processing challenges [165]. Similarly, blockchain technology can benefit greatly from its integration with AI, particularly in IoT environments [166]. Although IoT devices generally lack the expensive graphical processing units (GPUs) required for deep learning mechanisms, lightweight supervised and unsupervised learning methods, when properly optimized, can serve as practical alternatives to many of the computationally demanding tasks in blockchain systems without compromising security [167]. In this section, we discuss how existing security techniques are being enhanced by ML methods and how these techniques can be generalized to blockchain. We then examine how consensus algorithms can benefit from ML techniques, followed by a

discussion of a specific consensus algorithm that has attempted to integrate ML into blockchain.

2.4.1 Security Enhancement using AI techniques

Machine learning and deep learning techniques have become essential tools for enhancing security in a wide range of fields [168]. Gaining an understanding of how these techniques operate and the benefits they deliver provides important insights into how these approaches can be adapted to strengthen security within blockchain systems.

Network Intrusion Detection Systems (NIDS) are among the most researched and widely adopted applications of machine learning in security enhancement. In a network environment, intruders can attempt to launch various new attacks, which are becoming increasingly difficult to detect due to the massive growth of network traffic [169]. Machine learning approaches can significantly reduce the number of false alarms typically observed in traditional solutions and improve the overall reliability of detection [170]. In general, NIDS can be implemented as either hardware or software systems that automatically monitor incoming and outgoing traffic. They analyze different parameters of network flow and compare them with a predefined profile to determine whether an attack is occurring [171]. These profiles, or traffic datasets, can be used to train machine learning models, thus replacing conventional rule-based systems. Depending on the size of the dataset and the complexity of the network, the algorithms used may range from relatively simple methods such as decision trees and k-means clustering to advanced deep learning techniques such as convolutional and recurrent neural networks. [172].

Anomaly detection is a core aspect of intrusion detection, focusing on identifying deviations from normal system or network behavior that may indicate unauthorized or malicious activity. Although intrusion detection includes both signature-based and anomaly-based approaches, anomaly detection specifically enables the identification of previously unknown attacks by modeling normal behavior [173]. An anomaly is any data point that deviates from the expected pattern, which may or may not be malicious. Unlike traditional intrusion detection systems, anomaly detection systems (ADS) can use semi-supervised or unsupervised learning to identify unusual data without assigned labels [174]. Deep learning techniques have further advanced ADS, substantially improving detection accuracy [175]. Malware, or malicious software, is another critical security concern addressed with machine learning. Malware refers to software designed to disrupt, damage, or gain unauthorized access to a system with-

out the user's knowledge [176]. As malware becomes increasingly sophisticated and evasive, traditional detection methods often fail, making machine learning essential, even for Android and other mobile platforms [177].

Software-Defined Networking (SDN) is a modern networking paradigm that enhances scalability, flexibility, and manageability. By separating the control plane from the data plane (responsible for forwarding packets), SDN enables centralized control, improved network performance, and easier programmability of network operations [178]. ML has been increasingly applied in SDN to optimize various aspects, including traffic flow forecasting [179] and route optimization [180]. Beyond general performance improvements, ML-based solutions have also been developed to enhance SDN security, such as predicting network attack patterns [181], mitigating DDoS attacks [182], and supporting intrusion detection [183]. In general, implementing various network functions, such as firewalls, Network Address Translation (NAT), and others, requires separate networking devices. This approach makes developing new network functions increasingly difficult and time-consuming, as it involves creating new devices, training professionals on the latest technologies, and incurring high costs. Network Function Virtualization (NFV) addresses these challenges by using mainstream devices and virtualization to facilitate network functions. This allows the deployment of new functions simply by provisioning a virtualized environment within the existing infrastructure [184]. NFV also uses ML to enhance security for functions such as anomaly detection [185]. Moreover, approaches that integrate SDN with NFV often utilize different ML algorithms to further improve the security of the combined implementation [186].

2.4.2 Usage of ML and AI Algorithms in Blockchain

Several studies have highlighted how blockchain can be used in conjunction with ML and AI algorithms to improve different parts of existing approaches [187]. Some have proposed complete blockchain-based architectures integrated with ML for specific applications, while others have focused on strengthening security and introducing mitigation techniques. In addition, certain works have explored modifications to the internal components of the blockchain itself.

Although blockchain and ML may seem like two distant technologies, the reality is far from this perception [199]. Over time, several researchers have proposed numerous approaches that improve existing systems using ML. Previously, a number of domains that employ blockchain and IoT were summarized in Table 2.2. Many of these application areas have also attempted to integrate ML with blockchain to provide various

Table 2.3: Recent Studies Incorporating ML with Blockchain

Ref	Specialty	Description
[188]	Supply Chain	Framework for tracking perishable food distribution using blockchain, fuzzy reasoning, and learning techniques.
[189]	Supply Chain	Supply chain management and recommendation system for drugs combining blockchain and ML for smart pharmaceutical industry.
[190]	Supply Chain	Intelligent vaccine distribution system leveraging blockchain, IoT sensors, and ML models.
[191]	Healthcare	Blockchain- and IoT-based management of vaccination workflows with integrated learning support.
[192]	Healthcare	Service platform providing healthcare analytics with deep learning over blockchain infrastructure.
[192]	Agriculture	System to ensure transparency in agricultural logistics through blockchain and IoT with ML.
[193]	Agriculture	Smart agricultural framework for detecting crop pests using IoT monitoring, blockchain, and ML.
[194]	Education	System for detecting forgery and ensuring reliable academic results via blockchain and ML.
[195]	Sports	Sports outcome prediction model incorporating blockchain with hidden Markov approaches.
[196]	IoT	Enhancing IoT quality of service through integration of blockchain control and ML algorithms.
[197]	Security	Detection of abnormal network traffic patterns for blockchain using ML-based monitoring.
[198]	Block Size	Bitcoin sustainability framework by dynamically adjusting block size with predictive analysis.
[10]	Consensus	Lightweight evolutionary consensus method for IoT-focused blockchain-as-a-service platforms.

refinements, and some have additionally incorporated IoT. As shown in Table 2.3, fields such as supply chain [188], [189], [190], healthcare [191], [192], agriculture [192], [193], education [194], and even sports [195] have used these technologies in innovative ways to bring multiple improvements. Some of the security aspects discussed in Section 2.4.1, such as anomaly detection, can also play a critical role in blockchain networks [200]. Blockchain is prone to various anomalous activities that can compromise its security. Machine learning approaches used in traditional systems can therefore be adapted to enhance the security of blockchain technology [197]. Additionally, researchers have attempted to bring IoT, ML, and blockchain together under a unified framework [196].

Beyond traditional domains, machine learning can be applied to improve many core

components of blockchain technology. CheSuh et al. [196] address the scalability limitations of the Bitcoin network, such as long transaction verification times and high fees, which hinder greater adoption. They formulate an optimization problem to increase the number of transactions per block and propose a machine learning framework to dynamically predict optimal block sizes. The framework uses Extreme Gradient Boosting (XGB), trained on four years of historical network data with nine network-related attributes, to determine block sizes with high accuracy. On the other hand, Zhao et al. [10] focus on another core component: the consensus algorithm. They model block producer selection as a learning-to-rank problem that ranks network participants based on various attributes and chooses the highest-ranked node as the block producer. Section 2.4.4 provides a more detailed discussion of this consensus algorithm.

2.4.3 Opportunities of ML Based Consensus Algorithms

A consensus algorithm that uses machine learning is increasingly seen as an important direction for the future of blockchain [201]. By adding intelligence to the consensus, ML can improve network security by detecting nodes that behave suspiciously or maliciously. Instead of relying on traditional rule-based systems to spot problems, ML can continuously monitor node behavior, resource usage, and message exchanges to isolate risky nodes proactively. ML models can also dynamically adjust consensus parameters, such as latency limits, voting weights, or fault thresholds. These models learn from historical data on node behavior and transactions, creating a more detailed reputation system. Unlike traditional reputation systems that look at only a few factors, ML can analyze many security, reliability, and performance indicators, giving a more accurate and flexible measure of node trustworthiness.

In IoT environments, where devices have limited resources and networks often change, ML adds flexibility to consensus protocols. Instead of using a single fixed algorithm, ML can adjust parameters such as block size, timeouts, and communication frequency to maintain efficiency under different conditions. This flexibility can also extend to switching between different consensus algorithms. For example, ML could use real-time network data and past patterns to select the best algorithm at any moment. During low network load, a resource-heavy method, such as Proof of Work, could be used to maximize security, while under high traffic, a lighter approach, such as Proof of Elapsed Time, may be better for performance. If unusual activity is detected, the system could switch to a Byzantine fault–tolerant method to maintain security. ML can also help validate transactions and blocks by comparing them with

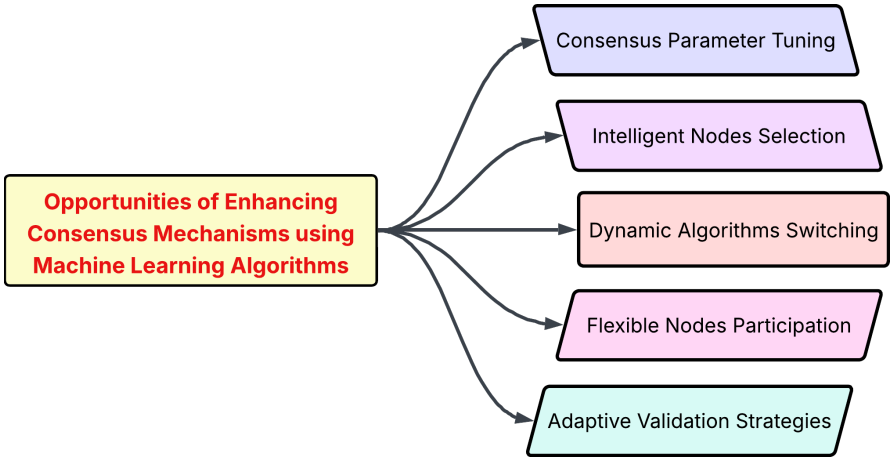


Figure 2.11: Opportunities of Enhancing Consensus Algorithms Using ML

patterns from previously verified data. Figure 2.11 will provide an overview of the potential improvements discussed for consensus algorithms enabled by ML techniques.

2.4.4 Proof of Evolutionary Model

Proof of Evolutionary Model, or PoEM [10], proposed in 2023, is one of the earliest attempts to develop a consensus algorithm based on machine learning. In brief, PoEM replaces the traditional block producer selection methods with a model based approach. In PoW, the block producer is the node that solves a computational puzzle; in PoS, it is the node with the highest personal stake; and in DPoS, it is the node with the most delegated stake. Similarly, in PoEM, the node ranked highest by a supervised machine learning model based on specific characteristics of each node and past performance is selected as the block producer.

Mechanism

In PoEM, the authors framed the consensus problem of selecting a block producer as a learning task, which aims to rank nodes according to specific constraints. They refer to this as Constrained Learning to Rank (CLTR). In this context, constraints include factors such as node trustworthiness, minimum stake, and similar requirements. The model ranks all nodes according to various features and constraints, and the top-ranked node is selected as the block producer. Their learning model consists of three components: punishment, node, and content. The punishment model penalizes a block producer that fails to produce a block successfully. The node model captures

individual characteristics of each node, including hardware features to assess its capability for resource-intensive tasks such as block production, as well as behavioral features based on its past performance in the blockchain. The content model considers the transactions to be included in the block, which forms part of the training data. This ensures that node selection is less predictable and can vary slightly with the content of each block, providing some level of uniqueness for individual transactions.

If no pre-trained model exists, they defined the initial phase of the BaaS-based IoT system as a cold startup. During this phase, the system can run existing consensus protocols such as PoW, PoS, or DPoS. Once sufficient data is collected from system logs during this phase, it is used to train the model, which is then deployed. The initially trained model from the cold startup phase is incrementally updated and evolved to improve its generalization capability. According to their approach, any supervised machine learning ranking algorithm can be used, as long as no data are missing for a particular node. The four algorithms they selected are simple linear regression (LR), LR with gradient boosting (LR_GBDT), factorization machines (FM) and field-aware FM (FFM), while other methods are left for future exploration. We discuss these algorithms in detail in Section 2.5.1 alongside other relevant supervised learning methods. Existing decentralized technologies, including federated learning [202], already provide secure model training in distributed environments. Additionally, they proposed a smart contract that stores hundreds of test sets from system logs to verify the correctness and security of the model.

In their system, all nodes maintain a copy of the current iteration of the trained model. Each node also receives a copy of the transactions through broadcasting. It allows any node to verify which node was rightfully chosen as the block producer. The transactions are required for inference with the model as it is part of the feature due to the content component of the model. A block producer may bundle multiple transactions in a block instead of one to improve consensus efficiency. When a new node joins the system, the next model iteration includes it in the ranking. However, it is highly unlikely to be selected as a block producer initially due to the lack of positive behavioral data. In real-world IoT systems, devices frequently join or leave the network. When a node exits, an entry is recorded in an exit table, and only nodes not listed in this table are eligible as validators. If a node fails to generate a block on time for any reason, the next highest-ranking node is selected to produce the block.

Limitations

PoEM has been an influential study for our work, and so we highlight some of its limitations in this section. PoEM trains its model and selects the block producer based mainly on static hardware data, such as battery power, CPU speed, and RAM. However, in an IoT environment, these fixed factors alone do not fully determine the capacity of a node. For instance, a node may have high battery power but be unable to recharge due to external factors, leaving it with very low energy. Such a node should not be chosen as a block producer.

When a new node joins the system, it is included in the ranking, but becoming the top-ranked node is nearly impossible if the system has been running for a while. Existing nodes may already have accumulated significant positive behavior scores, making it extremely difficult for newcomers to reach the top, even if their hardware is strong. This creates a centralization problem, where the participation of newly joined nodes is severely limited by established high-ranking nodes. Nevertheless, if a new node shows strong hardware potential and behaves reliably, selecting it as a block producer could be beneficial, as long as it has not done any misbehavior yet.

PoEM's algorithm also does not handle missing hardware features. Some IoT devices may rely on energy harvesting, in which case their harvesting rate should be considered, and rare devices may have additional hardware such as GPUs, for example, in road-monitoring systems. Furthermore, although the model is incrementally trained, the top-ranked nodes often remain similar across iterations. For example, one of the top ten nodes in one iteration may be selected in the next as a block producer, making repeated inferences potentially wasteful in resource-constrained IoT devices. It may be more efficient to use a single inference to select multiple block producers over a short period if the results are expected to remain stable. Finally, PoEM does not explicitly include a mechanism to prevent the same node from being repeatedly selected as a block producer in consecutive rounds. Nodes with a consistently high reputation can dominate selection, which can exacerbate centralization and reduce fairness.

2.5 Machine Learning Methods

Over the past decades, ML algorithms have been developed and refined with incremental as well as breakthrough improvements for a wide variety of purposes. Most of the first machine learning applications in the earlier decades were limited to relatively simple supervised and unsupervised tasks. However, the true potential of deep neural networks was unlocked by NVIDIA in 2007 [203] with the introduction of the Com-

pute Unified Device Architecture (CUDA), based on a prior work by Buck et al. [204]. CUDA enabled general-purpose computing on GPUs, making it possible to exploit the massive parallelism originally designed to render video game graphics. Unlike CPUs, which are optimized for sequential tasks with a small number of powerful cores, GPUs contain thousands of lightweight cores designed for parallel workloads, making them ideally suited for large-scale linear algebra operations at the heart of deep learning. By allowing these cores to process matrix multiplications and tensor operations simultaneously, CUDA provided the computational efficiency required to train large-scale neural networks [205]. This transformation not only accelerated research, but also made deep learning practical in real-world applications ranging from computer vision to natural language processing. For our work, supervised learning approaches are the most suitable, particularly because they can be efficiently deployed on lightweight IoT devices. Therefore, in this section, we provide a detailed discussion of different supervised methods. We also briefly touch on other approaches, such as unsupervised learning, deep learning and reinforcement learning, since with future advancements they may become increasingly practical for integration into our system.

2.5.1 Supervised Learning Algorithms

Supervised learning is a machine learning approach in which models are trained on labeled datasets, meaning that each data instance is paired with its correct output. The process typically involves collecting and preprocessing data, splitting it into training and testing sets, and then using the training data to learn the mapping between input features and target labels. Once trained, the model predicts the results for unseen data and is evaluated against the expected labels [206]. The two primary types of supervised learning are classification, where the goal is to predict discrete or categorical outputs (e.g., binary or multi-class labels), and regression, where the task is to estimate continuous values (e.g., predicting prices, age, or temperature).

Regression

Linear models form the foundation for many regression-based supervised learning approaches. Linear Regression (LR) [207] is the most basic technique, modeling the relationship between predictors and a continuous response variable as a weighted linear combination of features. Although effective for problems with linear dependencies, it may overfit in the presence of multicollinearity, a situation where two or more input variables are highly correlated and provide redundant information to the model. To address this, Ridge Regression (Ridge) [208] introduces an $L2$ regularization penalty

that shrinks the magnitudes of the coefficients to improve stability and generalization. Conversely, Least Absolute Shrinkage and Selection Operator (LASSO) [209] applies an $L1$ penalty, which not only reduces overfitting but also encourages sparsity in the learned coefficients, thus performing implicit feature selection. Another scalable alternative is the Stochastic Gradient Descent (SGD) [210] Regressor, which employs stochastic gradient descent to optimize linear models efficiently. This method is particularly suited for high-dimensional or large-scale datasets where traditional solvers become computationally expensive.

Beyond linear approaches, ensemble methods have proven highly effective in supervised learning. Random Forest (RF) [211] constructs multiple decision trees on bootstrapped subsets of the data and aggregates their outputs, thereby reducing variance and improving robustness compared to single-tree models. Similarly, gradient boosting techniques iteratively train weak learners to correct the residuals of previous models, producing strong predictive performance. Extreme Gradient Boosting (XGBoost) [212] is a widely adopted implementation that incorporates regularization and efficient handling of sparse features, making it highly accurate and scalable. Light Gradient Boosting Machine (LightGBM) [213], on the other hand, is optimized for speed and memory efficiency, employing histogram-based splitting and leaf-wise growth strategies to handle very large datasets. A hybrid variant, Linear Regression with Gradient Boosting (LR-GB) [214], integrates linear regression with gradient boosting, combining the interpretability of linear models with the flexibility of ensemble learning.

Another important family of supervised learning models is Factorization Machines (FM) [215], which are designed to effectively capture pairwise feature interactions in high-dimensional and sparse datasets. Unlike traditional linear models that treat features independently, FM leverages factorized parameterization to learn latent representations of features, enabling the model to generalize well even with limited observations. This makes FM particularly effective in tasks such as recommendation systems, click-through-rate prediction, and ranking problems where interaction effects play a crucial role. An extension of this approach is the Field-Aware Factorization Machine (FFM) [216], which improves FM by considering not only feature interactions, but also the specific field or group to which a feature belongs. In FFM, each feature has multiple latent vectors corresponding to its interactions with features from different fields, allowing the model to learn more fine-grained interaction patterns. This added flexibility often leads to significant performance improvements in domains with structured categorical data, such as computational advertising and personalized recommendation systems, although it comes at the cost of higher computa-

Table 2.4: Description of Common Regression-based Supervised Learning Approaches

Name	Description
LR	A linear model that predicts continuous outputs based on weighted input features.
Ridge	A linear model with L_2 regularization to reduce overfitting in correlated features.
LASSO	A linear model with L_1 regularization promoting sparse coefficients for feature selection.
SGD	A scalable linear model optimized with stochastic gradient descent.
RF	An ensemble of decision trees that aggregates predictions to improve accuracy and reduce variance.
XGBoost	A gradient boosting method with regularization and efficient sparse data handling.
LightGBM	A fast and memory-efficient gradient boosting method using leaf-wise tree growth.
LR-GB	Combines linear regression with gradient boosting for interpretability and flexibility.
FM	Captures pairwise feature interactions using factorized latent vectors for sparse data.
FFM	Extends FM by using field-specific latent vectors to model interactions more precisely.

tional complexity. Figure 2.4 provides a summary of each of the discussed regression-based supervised learning approaches.

Classification

In addition to regression-focused methods, several supervised learning techniques are specifically designed for classification tasks, where the goal is to predict discrete labels. Some classification algorithms are adaptations of the regression methods discussed earlier and modified to handle discrete label prediction. Common classification algorithms include Logistic Regression for binary or multi-class problems, Support Vector Machines (SVM) [217] that find optimal hyperplanes to separate classes, Decision Trees that partition feature space based on label purity, Random Forest and Gradient Boosting methods (e.g., XGBoost, LightGBM), which aggregate multiple weak classifiers to improve accuracy and robustness. While methods such as Linear Regression, Ridge, or Lasso are traditionally used for predicting continuous outputs, they can also

be adapted for classification by applying thresholds to the predicted values or using logistic transformations, allowing them to approximate class probabilities.

2.5.2 Unsupervised Learning Algorithms

Unsupervised learning involves identifying patterns or structures in data without relying on labeled outputs [218]. Common algorithms include K-Means Clustering [219], which partitions data into distinct clusters based on feature similarity, Hierarchical Clustering [220], which builds a tree-like structure to represent nested groupings of data points, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [221], which identifies clusters of varying shapes based on data density. Probabilistic approaches, such as Gaussian Mixture Models (GMM) [222], takes advantage of the Bayes' rule to estimate the probability of each data point belonging to a latent cluster. Principal Component Analysis (PCA) [223] and t-Distributed Stochastic Neighbor Embedding (t-SNE) [224] are widely used for dimensionality reduction and visualization, capturing the most informative features while reducing noise.

These unsupervised learning techniques cannot be directly applied to our consensus mechanism, as the model requires labeled data for training. However, future work could explore adaptations that use unsupervised methods in novel ways. For instance, dimensionality reduction techniques could be employed to decrease model complexity, making them more suitable for resource-constrained IoT devices. Instead of assigning ranks to new nodes without a history of successful block production, their fixed configurations or other intrinsic features could be used to cluster them with existing reliable nodes. Only nodes within such clusters would then be considered for block production, effectively combining structural similarity with reliability. Additionally, semi-supervised learning can be explored in scenarios where only a portion of the data is labeled, allowing the model to leverage both labeled and unlabeled information to improve node ranking or reliability estimation.

2.5.3 Deep Learning Algorithms

Conventional machine-learning techniques often required extensive feature engineering and domain expertise to transform raw data into suitable representations for classification or prediction. Deep learning overcomes this limitation through representation learning, automatically extracting hierarchical features from raw data via multiple layers of non-linear transformations [225]. Each layer captures increasingly abstract patterns, enabling complex functions to be learned: for example, in image recognition, lower layers may detect edges, intermediate layers assemble motifs, and higher

layers identify objects. These feature hierarchies are learned directly from the data, without manual design, allowing deep learning models to uncover intricate structures in high-dimensional datasets [226]. This capability has enabled remarkable advances across domains such as computer vision, speech recognition, drug discovery, genomics, and natural language processing, often surpassing traditional machine-learning methods in both accuracy and scalability. These deep learning techniques are already being applied in various security-related applications, as discussed in Section 2.4.1, and with the growing capabilities of IoT devices and the improved efficiency of deep learning frameworks, they have the potential to be incorporated into machine-learning-based consensus algorithms.

For image classification, several models have been developed, including ResNet [227], EfficientNet [228], and InceptionNet [229]. For object detection, You Only Look Once (YOLO) [230] models are among the most popular, enabling rapid detection of objects in images or videos. Recurrent Neural Networks (RNNs) are widely used to model dynamic relationships and capture patterns in time series data [231]. To address the vanishing or exploding gradient problems encountered by RNN in complex sequences such as videos or text, two prominent variants of RNN have been developed: Long Short-Term Memory (LSTM) [232] and Gated Recurrent Unit (GRU) [233]. LSTM networks introduce memory cells regulated by input, forget, and output gates to retain information over long periods, while GRUs simplify this architecture using only update and reset gates, reducing the number of parameters and improving efficiency on smaller datasets [234]. Bidirectional LSTM (BiLSTM) is a variant of LSTM that processes sequential data in both forward and backward directions, which is particularly useful for video or text data where a frame or token may have significant dependencies on both past and future elements [235].

2.5.4 Reinforcement Learning Algorithms

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make sequential decisions by interacting with an environment and receiving rewards or penalties based on its actions [236]. The agent's decision-making process is governed by a policy, which defines how actions are chosen in response to environmental states. The central goal of RL is to discover an optimal policy that maximizes the expected cumulative reward over time. Key theoretical elements of RL include the value function, which estimates the long-term reward expected from a given state, and the action-value function (Q-function), which predicts the expected reward of performing a specific action in a particular state [237]. RL algorithms often rely on

frameworks such as the Markov Decision Process (MDP) to formalize these interactions. Techniques such as Q-learning, policy gradients, and actor-critic models are frequently employed to address different types of decision-making problems.

Reinforcement Learning has demonstrated remarkable success across diverse domains, including robotics, autonomous navigation, finance, and game AI, due to its ability to handle dynamic, uncertain environments. In recent years, Deep Reinforcement Learning (DRL) has further enhanced RL's potential by combining it with deep neural networks to approximate complex value and policy functions, enabling applications such as AlphaGo and advanced robotic control [237]. In the context of blockchain and distributed systems, RL can be applied to ML-based consensus mechanisms, where individual nodes act as agents that learn optimal participation strategies. By receiving feedback in the form of rewards for successful block validation or penalties for faulty behavior, these agents can adapt their policies to improve overall network performance, reliability, and energy efficiency. This adaptive learning capability allows RL-based consensus models to dynamically respond to network changes, optimizing system behavior in a way that static rule-based mechanisms cannot.

Chapter 3

Proposed Methodology

In this section, we examine our proposed consensus mechanism, DP-POEM, in meticulous detail. At the outset of this chapter, we provide a concise synopsis of the overall procedure of our consensus mechanism and briefly outline the rationale behind its principal component. Following this, we elaborate on the individual components along with the underlying motivations in depth. Our methodology for resolving different problems is also articulated, accompanied by a discussion of feasible alternatives. For the sake of thoroughness, we consider virtually every possible alternative, some of which may never be pragmatically implemented, while others could be pursued through more sophisticated techniques in future research. Next, we delve into the machine learning component of our consensus algorithm, examining its internal structure, core purpose, potential feature set, and the process by which it is continuously trained. We then outline several approaches for generating the initial version of the ML model, which must be deployed at the inception of the IoT use case. Finally, we conclude with several miscellaneous considerations that, while noteworthy, may remain beyond the immediate scope of blockchain consensus-oriented studies.

3.1 Proposed Consensus Mechanism

The role of a consensus algorithm is to establish a reliable process for selecting a new block producer. Our proposed mechanism approaches this challenge from a fresh perspective. It is designed for networks such as IoT or other energy-conserving battery-powered systems, where participants voluntarily engage in the blockchain and contribute to block creation in order to preserve system integrity. Block producers are chosen through a machine learning model that ranks nodes based on their historical performance, current capacity to generate blocks, and the condition that they have

not served as a producer recently. In this section, we provide the description of the workflow of our ML-based consensus mechanism and conclude with an outline of its three main components in separate subsections.

3.1.1 Workflow of Proposed Consensus Mechanism

The foundation of DP-POEM, our consensus mechanism, is a lightweight machine learning model that predicts and ranks the suitability of each node aiming to become a block producer in the next cycle. The nodes seeking this role transmit their current hardware and network status to a small rotating set of nodes called supervisors. These data include metrics such as battery level, network bandwidth, and other operational parameters that assess each node's capability and likelihood of successfully producing a block. Specific applications can define these metrics as needed, offering considerable flexibility. Because these parameters are highly dynamic and change from time to time, it is important to aggregate the current status across all nodes. The supervisor nodes perform this task efficiently while keeping the communication overhead low.

Once supervisors receive the status data from all candidate nodes, they apply the latest machine learning model to predict the suitability score of each node. The resulting scores are then broadcast to the network and the *top-n* ranked nodes are selected as block producers for the upcoming cycle. Supervisors also package the inputs used in the machine learning inference as a transaction, allowing network participants to verify that the model was executed on authentic and accurate data. Multiple supervisors perform this task redundantly to prevent a single compromised supervisor from impairing network performance through manipulated inputs or operational failure. The total number of supervisors remains small, ensuring that overall system efficiency is not significantly affected. During this inference cycle, the model also identifies the next set of m supervisors, distinct from the block producers. These candidates are nodes with the highest suitability scores that have successfully served as block producers at least once. Specific applications can further impose stricter selection criteria if needed.

After the block producer list is finalized, block production begins. However, the highest-ranked node does not automatically assume the first position, as revealing the exact order of block producers could introduce security risks. Although all selected block producers participate in the cycle, their production order is determined through a secure randomization process.To avoid additional complexity in generating a globally consistent random value, this number is derived from the hash of the previous block, which is inherently accessible to all nodes through the blockchain

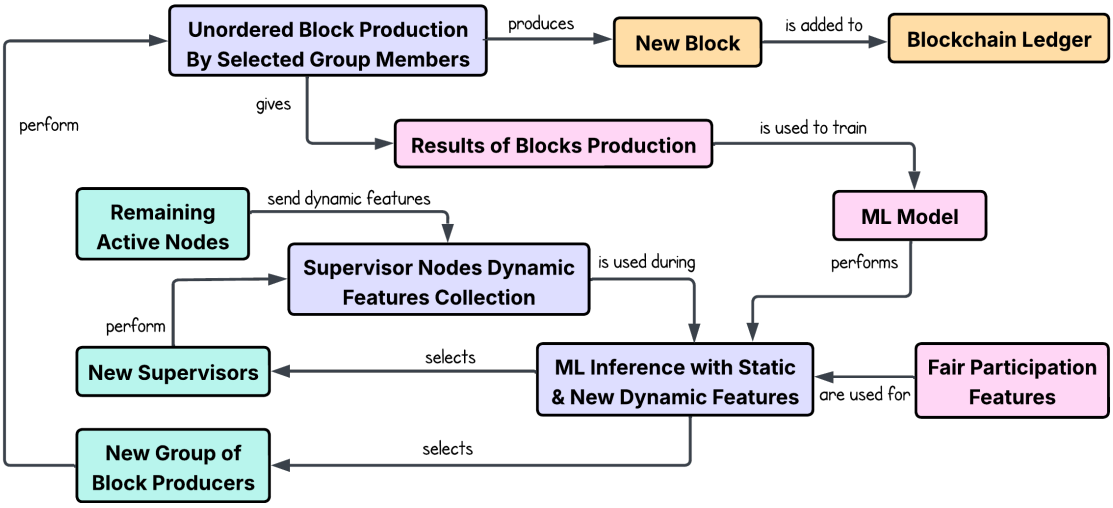


Figure 3.1: High-level illustration of the workflow of our proposed mechanism. Supervisors apply ML models to select the next supervisors and a group of block producers, using static and dynamic features of active nodes, as well as fair participation features. The selected group carries out block production in a randomized order, and each block is added to the blockchain after validation. The result of block production, whether success or failure, is then used as a label to train and improve the ML model for future decisions.

protocol. The hash is converted into an integer, r, ranging from 1 to n, where n represents the size of the block producer group. The r^{th} node that has not yet produced a block is then selected as the next block producer. Since all nodes possess a copy of the preceding block hash, they independently generate the same random value, ensuring a consistent selection of the next block producer. Each subsequent block producer only learns its identity after the previous producer completes their block, as the random value cannot be computed beforehand without the hash of a completed block, thereby reinforcing the robustness of the system.

By the time the current group of block producers completes block generation, the newly selected supervisors begin collecting requests from active nodes that wish to participate as block producers in the next cycle. Although, in principle, all nodes in a BaaS-based IoT system should ideally participate in the consensus process, it is assumed that some nodes may remain inactive due to the challenging nature of IoT environments, such as energy-saving requirements or other operational constraints. The machine learning model is incrementally updated based on the performance of the previous block producers, adjusting feature weights to decrease the likelihood of failure and increase the likelihood of success. Supervisors use the latest model to select the next group of block producers, taking into account the active nodes and their current feature profiles. This cycle repeats continuously, enabling the model to adapt

incrementally and become progressively more robust, while remaining responsive to the dynamic requirements of specific or changing applications.

The overall process is illustrated with the flow chart in Figure 3.1. Our consensus algorithm is built around three key components: (i) selecting a group of block producers, (ii) employing supervisor nodes to perform ML inference with dynamic features, and (iii) ensuring fair participation across the network. A high-level overview of these components is presented in subsections 3.1.2, 3.1.3 and 3.1.4. The detailed discussion of each component is then provided in dedicated sections: 3.2, 3.3 and 3.4.

3.1.2 Selecting a Group of Block Producers

In most conventional consensus mechanisms, only a single block producer is chosen at a time. Once that producer completes its task, the selection process is repeated for the next round. These mechanisms rarely provide a way to select a group of validators, and when they do, ensuring both security and energy efficiency is often unreliable.

Our approach introduces a more robust solution by leveraging machine learning models, specifically supervised models such as linear regression. These ML models generate a score for every node in the network, ranking them according to performance and capability. Instead of selecting just one producer, our mechanism proposes choosing a group of top-ranked nodes to serve as block producers across multiple consecutive rounds. This strategy addresses a critical limitation of ML-based ranking methods: computational cost. Typically, scoring requires a separate inference for each node during every selection cycle. As the number of nodes grows into the tens of thousands, this approach becomes increasingly inefficient and unsustainable [238]. By selecting multiple top-ranked producers in advance, with the exact number of nodes to select varying depending on total nodes in the system, our mechanism eliminates the need for repeated inference in every round, thereby conserving energy and improving efficiency. It improves long-term block production efficiency by reducing the time required to produce the next block since no selection is required, thereby enhancing scalability.

A potential concern might be whether security is compromised by not using the updated model before each individual block producer selection. In our consensus mechanism, after every block production, the consensus ML model is incrementally trained on new data. This data includes the outcome of the most recent production cycle, labeled based on whether each node successfully produced a block, together with the features used in its selection. The process is comparable to mini-batch stochastic gra-

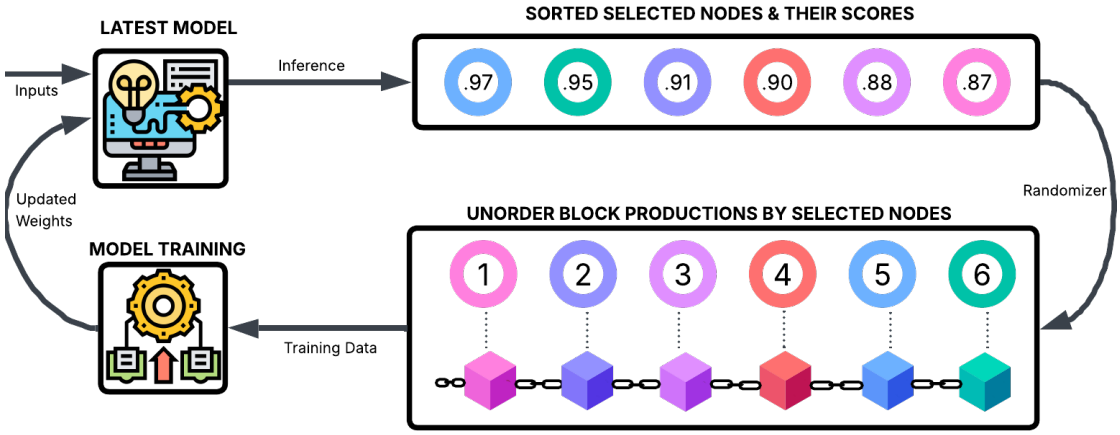


Figure 3.2: High-Level illustration of the process of selecting a group of nodes as block producer in our proposed mechanism. The ML model in our consensus mechanism selects the top-n nodes by predicted score. From this group, a novel randomized method selects the block producer, with the result revealed only after the new block is produced, preventing advance knowledge and attacks. Data from each block cycle updates the model weights for inference in the next cycle.

dient descent, where small updates gradually refine the model. Although these incremental updates improve generalization in the long run, the weight changes between individual iterations are minimal [239]. As a result, the nodes identified as the top candidates in earlier inferences typically remain among the highest-ranked options for next inferences too. Therefore, the quality and reliability of the selected block producers are not meaningfully compromised, ensuring that security and performance remain intact while also reducing unnecessary computational overhead.

Another security concern arises if the exact identity of the next block producer is known in advance, as that node could become a target of attacks such as DDoS, potentially disrupting its operation. To address this, our mechanism introduces a secure randomization step in determining the next producer. Rather than assigning the i^{th} ranked node to the i^{th} block, the next producer is chosen based on a random value generated from the hash of the preceding block, which is accessible to all nodes via the blockchain protocol. This hash is converted into an integer, r, between 1 and n, where n is the size of the block producers group, and the r^{th} node that has not yet produced a block is selected. Each node independently derives the same random value, ensuring consistency while keeping the identity of the immediate producer unknown until the previous block is completed. This approach preserves the advantages of ranking while significantly reducing the risk of targeted attacks. By keeping the immediate producer uncertain until the very last moment, the system strengthens security without sacrificing efficiency.

A top-level illustration of this process is shown in Figure 3.2. Concerns related to this avoidance of repeated inference are discussed in more detail in Section 3.2. We also provide a more in-depth discussion on the suitable group size and the method for determining the random order of nodes using the previous block hash.

3.1.3 Selecting Nodes Based on Dynamic Features

In an IoT environment, the capabilities of a device are strongly constrained by its hardware limitations. Selecting a node as a block producer, which is a relatively resource intensive task, requires careful consideration of its hardware specifications. These may include factors such as battery capacity, CPU power, memory, and other relevant attributes. Traditional consensus mechanisms face difficulties here, since manually incorporating each of these parameters into the selection process can be tedious and error prone. In contrast, a machine learning based mechanism provides a more streamlined solution. New hardware characteristics can be integrated simply by adding them as features in the dataset. This makes the system naturally extensible, since adding a new feature is as straightforward as adding a column, and including additional devices is as simple as adding rows.

However, static hardware specifications are not the only limiting factors for IoT devices. A device may have a large battery capacity and a high recharge rate, making it almost immune to battery related failures under normal conditions. Yet, there can be periods when the device is unable to recharge, causing its battery to drop to critically low levels. For example, consider a smart city IoT device that normally recharges its battery from the power grid. During a storm, a prolonged power outage can prevent it from recharging, leaving it nearly depleted. Similarly, an energy harvesting IoT device that depends on solar energy can fail to recharge during extended cloudy conditions. These scenarios highlight the importance of considering dynamic features in addition to static ones. Relevant dynamic features include the current battery level, real time recharge rate, and other operational conditions that can affect the reliability of a device. Our consensus mechanism incorporates these dynamic values as features within the machine learning model, ensuring that block producer selection remains adaptive to changing environments and resilient against sudden failures.

A major challenge in handling these features is that every node in the network would need to know the dynamic information of every other node. Exchanging this information directly introduces quadratic time complexity in the election process, which is as inefficient as traditional BFT mechanisms and is therefore highly unsustainable in IoT environments. To address this, we introduce a group of nodes called supervi-

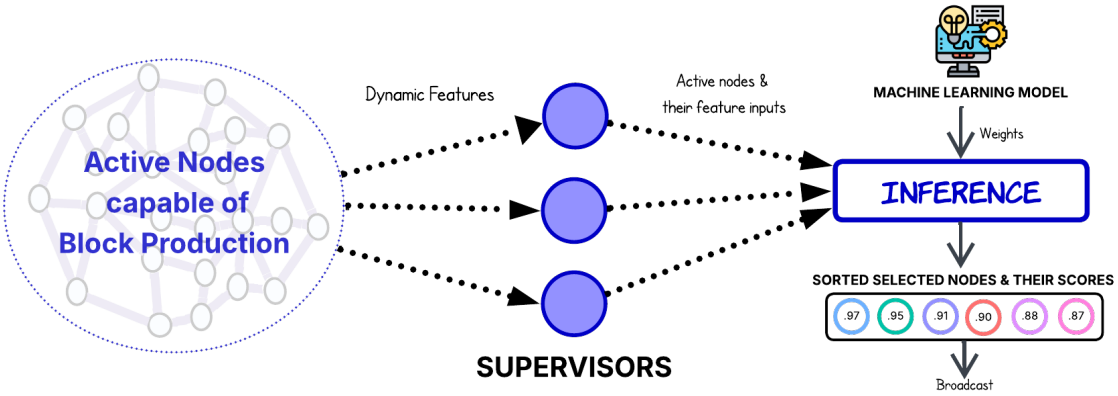


Figure 3.3: High-Level illustration of the process of using supervisors node to enable the use of dynamic features in our proposed mechanism. Nodes send the dynamic features to supervisor nodes who uses latest ML model to find highest ranked nodes as block producer. The result is broadcast to the network.

sors, whose role is analogous to that of election commissioners in traditional voting systems. Before each machine learning inference, supervisor nodes collect data from the nodes that are currently active and therefore, capable of becoming block producers. Using this information, the supervisors select the next group of block producers and also determine the supervisors for the subsequent round. Multiple supervisors are employed whose job is exactly the same to ensure that no single malicious node can manipulate the selection process. Figure 3.3 shows a brief overview of the use of supervisor nodes.

To guarantee integrity, the dynamic information shared by the competing nodes is recorded on the blockchain, allowing each node to verify whether the supervisors used authentic data or not. The supervisors themselves are not fixed; they rotate in every ML inference. Their selection follows the same ML-based process as that of block producers, with stricter requirements if necessary. This approach reduces the communication complexity of the election process to linear order, as it depends only on the number of competing nodes when the number of supervisors remains fixed or does not scale with increasing nodes. By selecting a group of block producers on a larger frequency, the system further reduces communication overhead. Unlike conventional methods, our approach not only preserves efficiency but amplifies it, delivering a consensus mechanism that is simultaneously scalable, resilient, and performance optimized. We discuss this part of the consensus in more details in Section 3.3.

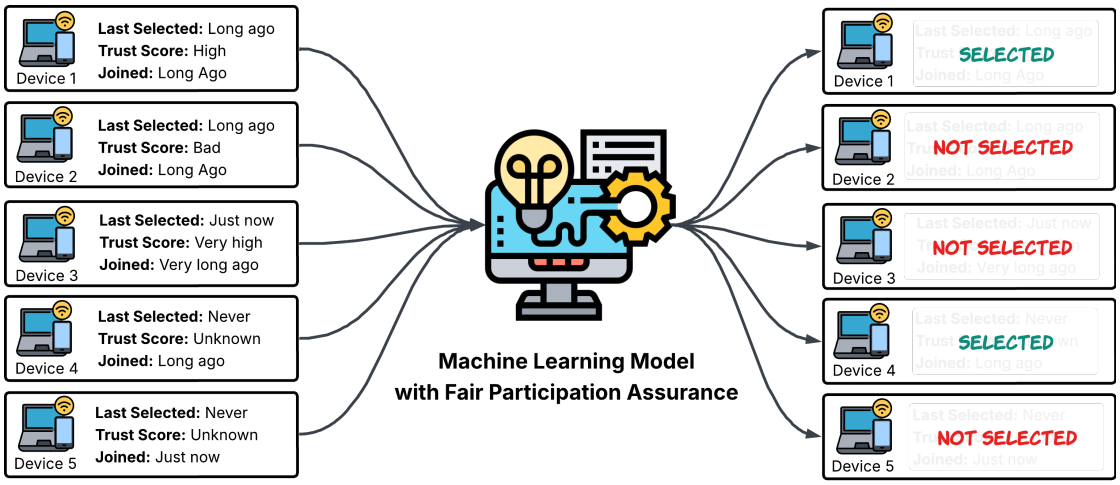


Figure 3.4: High-Level illustration of the process of ensuring fair participation of all nodes in our proposed mechanism. Alongside reliability and performance scores, features are included in the machine learning model to avoid repetitive selection of the same nodes and to provision for new nodes to participate.

3.1.4 Selecting Nodes While Ensuring Fair Participation

In a truly decentralized system, it is essential to ensure fair participation among all honest nodes [240]. However, a node should not be prioritized solely because it has been part of the blockchain network for a long time or has a history of honesty. Newer nodes, even with lower initial trust, may also be valuable block producers and should not be overlooked if they have never misbehaved. To maintain fairness, repeated selection of the same nodes within a short block production cycle should be avoided. A node that has already produced a block should ideally not be reselected until sufficient time has passed or until other capable nodes have had their opportunity.

The same holds for newly joined nodes in the system. In IoT environments, nodes frequently join and exit dynamically. Although a node that has just entered may not be immediately suitable as a block producer, once it has remained in the blockchain for a reasonable period, a minimum guaranty of participation should be ensured. New nodes should not be dismissed simply because they arrived later, as lateness alone is not a valid basis for judgment. A capable new node can strengthen the blockchain, and in the long run, including them enhances system reliability by increasing the pool of honest participants, outweighing the risks of a small minority of dishonest ones.

The existing consensus mechanisms do not adequately prevent repeated selection of the same nodes as block producers, nor do they provide a robust way to integrate

new nodes into the system. This limitation arises from the complexity of designing the appropriate variables and incorporating them into general rules. Using machine learning in our approach simplifies the process, as it allows the introduction of additional features to guide selection. In our proposed mechanism, we design these features carefully to ensure that nodes are not starved of block production opportunities for too long or selected excessively. Newly joined nodes are initially treated as if they have just completed a block production cycle. Hence, their time to first eligibility roughly aligns with that of existing nodes being able to become a block producer again after producing a block. The lack of history of new nodes can be compensated for by assigning higher importance to other static features until they are selected for the first time. The risk posed by dishonest new nodes is mitigated by the adaptability of the machine learning approach, which can downweight repetition-avoidance parameters if many new nodes behave dishonestly. This novel approach is discussed in greater detail in Section 3.4. Figure 3.4 shows a high level overview of this approach.

3.2 Avoiding Repeated Inference

Our consensus mechanism is designed to minimize the number of inferences required in an ML-based consensus algorithm while maintaining strong security guaranties. To achieve this, we first select the top block producers based on the total number of active nodes. Then, using a secure mechanism, we randomize the order of block production among the selected group. In this section, we provide a quantitative justification and a detailed description of this component of our consensus mechanism, along with a careful consideration of its different design choices.

3.2.1 The Cost of Inference

In the context of Machine Learning, inference refers to the process of using a trained model to generate predictions or decisions on previously unseen data. The lifecycle of an ML model begins with the training phase, where the model learns patterns from a training dataset, often validated on a separate validation set to avoid overfitting. Once the model achieves sufficiently high performance, it is deployed in real-world scenarios to produce outputs [241]. During inference, the model is typically kept fixed, but in some applications it can be updated or fine-tuned incrementally with small batches of new data to improve accuracy and adaptability over time.

Our consensus mechanism uses supervised learning methods whose job is to rank all nodes in the system based on their predicted score. This requires an identical com-

Table 3.1: Inference time (seconds) for different supervised learning models across varying number of nodes

Model	Number of Nodes				
	50,000	**100,000**	**250,000**	**500,000**	**1,000,000**
LR-GB	0.0416	0.0821	0.4179	0.7250	1.4905
LASSO	0.0093	0.0106	0.0197	0.0396	0.0773
LightGBM	0.1243	0.2713	0.6399	1.5892	2.4951
LinearRegression	0.0058	0.0088	0.0270	0.0481	0.0957
RandomForest	0.2085	0.6523	0.9891	3.0918	5.0777
Ridge	0.0032	0.0107	0.0301	0.0520	0.0777
SGDRegressor	0.0035	0.0080	0.0227	0.0394	0.0801
XGBoost	0.0479	0.0964	0.2420	0.4810	0.9698

putation for each node, which can be as simple as a multiplication or a split evaluation in a decision tree depending on the type of model used. Hence, as the number of nodes increases, the total time required to generate scores for all nodes tends to grow linearly. To confirm this, we measured the inference times of several supervised learning models, which were discussed in Section 2.5.1, for varying numbers of nodes. The results are presented in Table 3.1. All models were tested on an Intel Xeon(R) CPU running at 2.20 GHz with a single core enabled, a configuration comparable to resource-constrained IoT devices that do not have GPUs. For each case, 50 features were used.

For all models, the inference time exhibits a steady linear growth as the number of nodes increases. This inference time accounts only for generating scores for all nodes and does not include the time required to rank them. Ranking requires the use of sorting functions, and as the number of nodes increases, the sorting time can increase significantly. However, sorting only a subset of top nodes instead of the full set can slightly reduce the time complexity. It should be noted that the actual inference time may vary slightly depending on the weights of the model and the training data, but the linear trend remains consistent. Similarly, although to a lesser extent than inference, the sorting time can also vary slightly between models.

Nevertheless, it is evident that performing repeated inference as the number of nodes grows can become costly. Even when using relatively inexpensive models, such as Linear Regression, the computational cost increases with the number of nodes. IoT devices are expected to maintain low latency and high throughput even as the system scales. Performing a full inference each time to select a single block producer is inefficient and can quickly outweigh the benefits of using machine learning models,

making the system unscalable. In this situation, having scores for all nodes in each inference provides an opportunity to enhance the system by selecting multiple block producers from a single run.

3.2.2 The Opportunity of Selecting Multiple Nodes

To find the best-ranking nodes, it is necessary to compute the suitability of each node. There is no alternative to this computation when using supervised learning methods. Although this introduces some overhead, it also presents a unique opportunity to optimize the consensus algorithm. The top-ranked node is the absolute best but the nodes immediately following it are not insignificant. Some of these nodes would have been selected as the top producers in future inferences anyway.

If we use one inference per selecting each block producer, we would only have one training data to update the model. Regardless of this training data, the overall weight update would be minimal. As a result, for subsequent inferences, the overall top nodes would remain unchanged. This implies that performing multiple inferences would produce nearly identical results, wasting both time and resources on computationally constrained IoT devices. Additionally, setting up training also consumes resources but it does not significantly improve the model for one training data. Although small batches of data can be used, a batch size of one is almost never recommended. A proper balance in batch size is necessary to achieve meaningful model updates [242]. Allowing multiple block producers to generate blocks simultaneously provides more data and creates the right balance for the machine learning model to perform meaningful updates, ultimately delivering better performance.

The proposed consensus mechanism aims to ensure fair participation of all nodes, preventing repetitive selection of existing nodes while facilitating the inclusion of new nodes. Its design incorporates unique features that allow new nodes to reach the top rankings if they have been part of the system for a sufficient period of time. This is further supported by selecting a group of block producers. Even with the best features, it may be difficult for new nodes to reach the very top if only a single block producer is selected. By selecting a group of block producers and combining this with multiple fair-participation assurance features, new nodes are more likely to be included in the set of selected block producers, even if they are not the highest-ranked nodes but are close to them. In this way, the selection of multiple block producers not only conserves resources but also actively contributes to improving the overall system.

Table 3.2: Number of selected block producers under different scaling relations as the number of total active nodes increase

Active Nodes (N)	Number of Block Producers				
	$\sqrt{N}$	$\log_2 N$	$N^{1/3}$	$N/100$	$N/500$
1,000	32	10	10	10	2
10,000	100	13	22	100	20
50,000	224	16	37	500	100
100,000	317	17	47	1000	200
250,000	500	18	63	2500	500
500,000	708	19	82	5000	1000
1,000,000	1000	20	100	10000	2000

3.2.3 The Number of Block Producers to Select

Determining the optimal number of top-ranked nodes to select can be challenging. Selecting too many risks including unreliable nodes in the system, while selecting too few diminishes the benefits of having multiple block producers. Ideally, this number should be determined based on the total number of nodes aspiring to become block producers in the next cycle, following a non-linear relationship. By doing so, the overall asymptotic time complexity of the block producer selection process can be significantly reduced.

To strike a proper balance, we propose that the number of block producers be chosen as the square root of the number of active nodes, as follows:

$$BP_{\text{size}} = \sqrt{N_{\text{active}}} = \sqrt{N_{\text{total}} - N_{\text{inactive}}} \tag{3.1}$$

Here, BP_{size} denotes the number of block producers selected, N_{active} is the number of active nodes who are currently capable of being block producers, N_{total} is the total number of nodes in the system, and N_{inactive} refers to the number of nodes not active.

Table 3.2 illustrates how the number of block producers changes as the total number of participating nodes grows. Selecting more block producers as the number of nodes increases does not, in itself, reduce the security of the system, since the proportion of malicious nodes typically remains constant. A smaller block producer group is more susceptible to attacks. The logarithmic relationship increases very slowly and therefore, does not scale well with larger networks, making it less practical in terms of efficiency and security. On the other hand, purely fractional methods (e.g., $N/100$ or

$N/500$) scale linearly with network size, which can become excessive and resource-intensive for very large networks. Power-based relationships such as $\sqrt{N}$ or $N^{1/3}$ provide a more balanced growth. They increase the number of block producers as the network expands but at a slower rate than linear methods. Among these, the square root relationship offers the most effective balance between reliability, efficiency, and scalability. The number is also not too low when the active node size is small, which is why we recommend it. Nonetheless, our framework is flexible, and each system can adjust the number of block producers to select according to its specific security and performance requirements.

3.2.4 Randomization to Find Next Block Producer

Machine learning models generally behave like a black box. Although certain patterns may receive higher priority, the exact outcomes are inherently unpredictable. Similarly, it is not possible to deterministically control which node will become the next block producer. This unpredictability persists even when selecting a group of block producers and it is a desirable trait for a secure consensus algorithm. For example, in PoW, miners attempt to find a nonce that satisfies the difficulty target. This process is essentially random as the ideal nonce changes for every block and cannot be guessed in advance. In our approach, selecting a group of block producers introduces a controlled degree of randomness. However, if the block production of selected nodes was strictly serial, the earliest chosen block producers could become more vulnerable to targeted attacks. Hence, it is necessary to ensure that the block production within the group does not follow a defined order.

Even if the order is random, it is necessary that this random order is known to all nodes in the system, but not in advance. In this way, the block producer in the group would immediately know that they are supposed to produce the block, while the others would know that they are not. Introducing additional communication to convey this information would diminish the benefits gained from avoiding extra inferences. Therefore, every node must locally select the same node. However, if the random value can be generated far in advance, the benefit of randomness is essentially lost. Hence, the requirement is that every node in the system selects the same random node from the group of block producers without expensive computation, at a time just before block production begins. To achieve this, we propose using the hash of the last block to generate the random value.

In every blockchain, the previous block in the system must be confirmed before producing the next block. This is because a new block includes the hash of its parent

block as part of its data, which is required to generate the current block hash. After a block producer finishes producing a block, it broadcasts the block to the rest of the network. This entire process is illustrated in Figure 2.6. Our system follows the same principle. Hence, after a particular node in the group finishes producing a block, all nodes in the system, including the selected block producers who are yet to produce their blocks, receive a copy of the new block. Using this block hash to select a member from the group also ensures that the new block producer, along with every other node in the system, learns their identity only after the previous block producer has completed their block, leaving little to no time to launch targeted attacks.

The block hash is converted into an integer ranging from 1 to the number of remaining block producers who have yet to produce a block. First, each character in the previous block hash is converted to an integer using its ASCII value. These integers are then summed to form a large number. Finally, this number is mapped to the desired range using a modulus operation. The calculation of the random value can be formulated as follows:

$$r = \left(\sum_{i=1}^{k} \text{ASCII}(h_i) \right) \bmod |BP_{rem}| + 1 \tag{3.2}$$

In this equation, h_i denotes the i-th character of the previous block hash, k is the total number of characters in the hash, and $|BP_{remaining}|$ represents the number of block producers who have not yet produced a block. The resulting value r corresponds to the selected random node from the remaining block producers.

In this context, we can provide two definitions. The machine learning model returns a sorted set of m block producers (N), $BP_{size} = \{N_1, N_2, N_3, \dots, N_m\}$. At any given time, if x random nodes (RN) have already produced blocks, then $BP_{done} = \{RN_1, RN_2, RN_3, \dots, RN_x\}$. Consequently, the remaining nodes form the ordered set $BP_{rem} = BP_{size} \setminus BP_{done}$. Each random value r selects the r-th element or node from the ordered set BP_{rem}. Then, this node is added to the unordered set BP_{done} and BP_{rem} is updated. After each ML inference, the elements of these sets are reset.

3.2.5 Algorithm for Group Selection

To demonstrate this block production operations with a set of block producers, we design Algorithm 1. The full description is as follows:

ML inference (line 1): First, the supervisor nodes perform the ML model inference

Algorithm 1 Full ML Inference and Block Production Cycle

Require: Latest Model: M_{latest}, Nodes Data: $data_i$ $(0 < i < n)$

1: BP_{size} = supervisors . $M_{latest}(data\ , n)$
2: m = len(BP_{size})
3: $BP_{done} = \{\ \}$
4: **for** $i = 1$ to m **do**
5: $\quad BP_{rem} = BP_{size} \setminus BP_{done}$
6: $\quad$ r = generate_node_index(BP_{rem} , prev_block_hash)
7: $\quad$ cur_block_producer = BP_{rem} . at(r)
8: $\quad BP_{done}$. add(cur_block_producer)
9: $\quad$ cur_block_producer . produce_block(txn_list)
10: **end for**
11: BP_{size} . clear()
12: BP_{done} . clear()

based on the latest data collected from all active nodes. The top-m nodes are returned, where m depends on the total number of active nodes, n. For next steps, these results are passed to all other nodes by supervisors, which is not shown here.

Initializations (lines 2-3): The steps onward are run by all nodes in the system. The values required to run the for loop are initialized at this stage. Here, m is the number of selected block producers, and BP_{done} is used to keep track of nodes that have already produced blocks.

Blocks Production (lines 4-10): Initially, the set of nodes that are yet to produce blocks is determined. Then, a random number r is chosen and the corresponding node from the remaining set is selected as the block producer. This node is also added to the BP_{done} set, and it produces the block. Note that this algorithm is executed on all nodes and since each node knows the same random value, only the selected node actually produces the block.

Ending (lines 11-12): The sets are cleared as they are no longer needed. The function then returns. For the next block production cycle, the algorithm is run again after updating the ML model with the latest data between cycles.

3.3 Handling Dynamic Features

Our consensus mechanism is designed to adapt to the dynamic features of each node. Unlike fixed configurations, where node properties can be predetermined, these features change over time. To ensure consistent inference and identical results across all nodes, it is necessary for them to operate on uniform data. To achieve this, we intro-

duce a rotating group of nodes, called supervisors, who are chosen based on trust and reliability. Each supervisor serves for a limited duration that covers a single ML inference and extends only until the next group of block producers is selected. During their role, supervisors collect the features of active nodes and, at the end of the process, perform the inference. Then they broadcast the results to the rest of the network. They also share the ML data used as transactions, which are recorded on the blockchain. It ensures that all nodes can verify the authenticity of the data. In this section, we first discuss the importance of incorporating dynamic features and explain why their use necessitates the introduction of supervisor nodes. We then describe the specific responsibilities of these supervisors and outline their selection process.

3.3.1 Importance of Dynamic Features

Battery life is one of the most critical concerns in IoT devices. These devices are often deployed in environments where recharging or replacing the battery is not a practical or reliable process. For example, IoT nodes used in environmental monitoring can be placed in remote locations with no direct access to energy sources, making frequent battery replacement unfeasible [243]. In such dynamic and demanding conditions, devices can experience premature battery failure, leading to unexpected shutdowns [244]. Beyond just the current battery level, other features also affect the availability of a node. These may include residual energy capacity, charging rate (if energy harvesting is possible), duty-cycle behavior, or even thermal conditions that affect power efficiency. Monitoring such energy-related characteristics provides a more accurate picture of the reliability of the node.

In addition to energy constraints, network-related features also play a decisive role. For example, in a wireless environment, reduced bandwidth or intermittent connectivity can make certain nodes temporarily unreachable, regardless of their energy state. Moreover, not all nodes are equally motivated to participate in block production. A node might prioritize other essential tasks, such as continuous sensor readings or data aggregation, over the resource-intensive process of block generation for a particular period. To conserve energy, nodes can also switch to low-power states such as sleep, hibernation, or deep sleep. Therefore, it is crucial not only to understand the energy status of IoT devices but also to account for their willingness and ability to participate in block production at a given time.

Hence, beyond the general fixed configuration features, it becomes essential to take into account these dynamic attributes of the IoT nodes. Since such characteristics vary over time, they must be assessed immediately before the selection of block producers.

Relying on a full network-wide broadcast, where every node shares its state with all others, would introduce significant communication overhead and waste energy. To address this, we introduce a rotating group of supervisors. While the current set of block producers is active, these supervisors gather the latest features and interests of other active nodes. For simplicity, we refer to this message sent by active nodes, which includes their current state data, as an interest message. Using these collected data, they then apply the most recent ML model to perform inference, ensuring that the selection of the next block producers reflects the real-time conditions of the network. This result is shared with the rest of the network.

3.3.2 Inference By the Supervisors

At any given time in our consensus mechanism, a group of block producers operates in a randomized order to generate blocks. Alongside them, a set of supervisors, who are selected together with the current block producers, collect the interests of other active nodes that are capable to participate in the next round of block production. Supervisors for the next round are also chosen from this pool of active nodes. However, the system can be configured to allow nodes to express interest specifically in becoming supervisors if required. In either case, the current supervisors continuously gather interest messages throughout the entire duration of block production. Once the final block producer in the group has completed its task, the supervisors execute inference using all the accumulated interest data. They can reliably determine this moment because each block producer, after finishing its block, broadcasts it to the network. By counting these broadcasts, supervisors know when the last producer has finished.

After the supervisors perform the inference at the end of the block production cycle, they obtain a ranked list of nodes based on their scores. This result is then shared with the rest of the network. For efficiency, the supervisors may share only the list of selected nodes if the system permits. All nodes in the system receive a copy of the inference result from every supervisor. Ideally, there should be no variance in these results. If discrepancies occur, nodes accept the result that is most consistent across supervisors. To strengthen security, an additional safeguard can be applied where nodes accept a result only if it matches the majority of supervisors, rather than simply the closest match. If the majority condition is not met, a backup group of supervisors or some other methods is activated to provide the inference result. The exchange of information between active nodes and supervisors takes place in the form of transactions, which ensures that authenticity can be verified. Furthermore, the data used by supervisors to perform inference is also stored as transactions on the

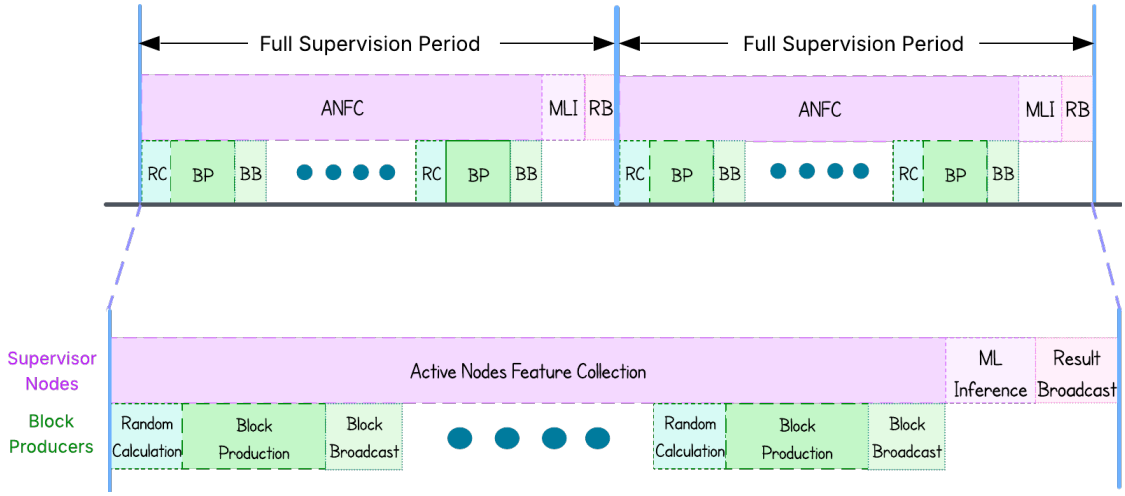


Figure 3.5: Illustration of the duties of supervisor nodes during their term. Selected supervisors collect the feature values of active nodes while the current block producers generate their blocks. Once the last block producer completes its task, the supervisors perform inference using the collected data, broadcast the results to the network, and hand over responsibilities to the next set of supervisors. Note that the actual duration is not to scale.

blockchain. This provides an immutable record, enabling any node to verify that the supervisors relied on authentic and validated data when carrying out the inference. Figure 3.5 reveals the responsibilities of supervisors in form of duty cycles.

The introduction of multiple supervisors is intended to add robustness to the system. Although the supervisor nodes are carefully selected based on high trust, reliability, and current availability, there is always the possibility that one or two can fail or behave maliciously. However, the likelihood that a majority of supervisors do so at the same time is extremely low. For this reason, choosing a small group of highly reliable supervisors provides a practical balance between security and efficiency. We recommend keeping the number of supervisors mostly constant as the total number of nodes in the system increases. This design ensures that the supervisor count remains a small fixed factor and prevents the communication complexity from approaching quadratic growth, which would occur if the number of supervisors scaled directly with the number of nodes. Additionally, fixed supervisor count means the likelihood of risky nodes being selected is also low. A suitable way to achieve this can be through the use of a logarithmic function. As shown in Table 3.2, applying a logarithmic growth pattern with realistic sizes of an IoT system keeps the number of selected supervisors within a manageable range.

3.3.3 Ensuring Use of Authentic Data

Two key concerns may arise with respect to the authenticity of the data used. First, the data sent by the active nodes can be falsified. For example, a node with low battery charge may deliberately send fake data claiming to have a high charge. Without system logs, there is no straightforward way to verify this. Second, supervisors themselves may misbehave by manipulating the data provided by active nodes and using forged values to select block producers of their own choice. In any trust-based decentralized system like blockchain, the primary incentive for honest behavior comes from the promise of meaningful rewards or from the threat of punishment and exclusion from future participation [245]. However, this requires a verifiable way for the rest of the network to detect misbehavior. Our consensus mechanism addresses these concerns by leveraging existing blockchain features, which ensure that additional overhead is kept minimal.

In our design, data exchanged between supervisors and active nodes is treated as standard blockchain transactions. Each submission is signed digitally, ensuring authenticity and giving active nodes a verifiable copy of what they sent. Supervisors, when submitting their inference results, also record the ML data they used as one or more blockchain transactions. This allows any node in the network to confirm that its original signed data was indeed used in inference. If a supervisor is challenged and found to have tampered with the data, it loses trust and is permanently excluded from future selection. This creates a strong incentive for supervisors to not misbehave. For general nodes, exaggerating their data also carries consequences. If such nodes fail to produce blocks due to low energy or other inconsistencies, their credibility within the blockchain is diminished. Over time, this loss of trust reduces their influence and role in the system, ensuring that honest behavior is consistently rewarded.

Allowing nodes to send their data throughout the entire block production cycle does not create a significant risk of outdated information. This is because the number of block producers is determined using a square root function, which keeps the group size moderate and avoids long delays. Moreover, since data exchange is spread across the whole production cycle, it does not impose a heavy short-term load on the network. A similar concern may arise during the initial selection of block producers, since nodes express interest before block production begins. However, the reasoning remains the same. The time window is narrow, and any chance of short-term changes in node states is handled by the ML model. If the condition of a node appears to be unreliable, the model ensures that such nodes are not selected. The model achieves this through continuous fine-tuning with the data of system it is deployed, which en-

ables it to adapt to changing conditions and remain aligned with the length of each block production cycle.

3.3.4 Next Supervisors Selection

For proper decentralization, no node in the system should be allowed to hold a role for an extended period of time. Therefore, supervisors are also required to rotate. The duration of a supervisor's role lasts only until the current group of block productions is completed. At that point, the supervisors run ML inference to select both the next block producers and the next set of supervisors. This design allows for alternative approaches. The expression of interest to become a supervisor by active nodes may be kept separate from the interest message for block production, or both may be combined. If they are different, two separate inferences can be run with the same model, or even with two distinct models. On the other hand, if the same model is used for a single inference, stricter conditions can be applied for supervisor selection, such as requiring a certain degree of trust or prior experience in block production. This ensures that newly joined nodes cannot immediately assume the critical role of supervisor. Unlike block producers, supervisors do not require randomization since their tasks are identical. As a result, the system can simply select the top-ranked nodes that satisfy the stricter conditions for supervisors.

For our proposed mechanism, we suggest using a single interest message and a single inference, with the additional requirement that a node must have served as a block producer at least once. Active nodes send their interest message to the current set of supervisors to become either a block producer or a supervisor. The supervisors then run the inference and generate a sorted list of all active nodes. From this list, the top r nodes that satisfy the conditions are selected as supervisors, while the next top m nodes are chosen as block producers for the upcoming cycle. The number of supervisors follows two-base logarithmic relationship with number of active nodes which canbe considered just as constant relationship, as seen in Table 3.2. This process can be formally expressed as:

$$\begin{aligned} res_{full} &= ML_{latest}\,(data\,,\,n) \\ sup_{list} &= res\,.\,\text{get\_top}\,(r\,,\,sup\_rules) \\ BP_{size} &= res\,.\,\text{get\_top}\,(m\,,\,not\_in\_sup) \end{aligned} \tag{3.3}$$

An issue related to supervisors is the potential deadlock in the system if a majority of them fail. In such a case, all block producers may have already completed their

blocks, which means that there is no ongoing block production, and the supervisors are unable to select new block producers, causing the system to halt. This situation can be handled in a manner similar to how block producer failures are addressed in systems where duties are assigned to specific nodes. A recovery mechanism is triggered if supervisors do not provide results within a predefined time. We propose two alternative approaches to handle this, depending on the system design. The first approach is to select one or more groups of backup supervisors during each cycle. These backup supervisors remain inactive unless the primary supervisors fail, in which case the backup group briefly collects interest messages and runs the inference. The second approach is to allow all nodes to run an emergency inference without dynamic data, using fixed features such as reliability or trust that do not change. This eliminates the need for supervisors in such situations. The block producers and supervisors selected through this emergency process can then manage the system, after which the normal election process resumes in the next cycle. Although these concerns may raise questions about the reliability of the system, the probability of such failures occurring is extremely low. Nevertheless, these mechanisms ensure robustness and provide a way to maintain continuity even in extreme situations.

3.3.5 Algorithm for Supervisors

The job of supervisors and how they perform it is shown in Algorithm 2. If a node finds itself to be supervisors, then it collects the data from active nodes and at the end of block production cycle, it performs inference with latest model which is then broadcast to the rest of the network. The detailed description is as follows.

Assume Role (lines 1-5): If a node identifies itself as a supervisor, it assumes the role and proceeds with the subsequent operations. Otherwise, it exits the function and performs other tasks.

Interest Collection (lines 6-12): Supervisors collect interest messages sent by other active nodes in the system and add them to the dataset used for ML inference. Each message is digitally signed to ensure authenticity. This process continues until the last block producer completes block production.

ML Inference (lines 13-18): After data collection, the number of supervisors and block producers to be selected is determined. ML inference is then performed on the collected data to generate a sorted list of block producers. The supervisors are also selected as a disjoint set from the block producers.

Result Broadcast (lines 18-20): Supervisors convert the ML data used into

Algorithm 2 Role Performance by Supervisor

Require: Supervisors List: sup_{list} , Latest Model: ML_{latest}

if this *in* sup_{list} **then**
    assume_supervisor_role ()
else
    return
end if
$data = \{\ \}$
while last_block_production *is not* done **do**
    listen_for_interest()
    if interest_received **then**
        $data$ += signed_interest()
    end if
end while
$n = len\,(data)$
$r = log\,(2\,, n)$
$m = root\,(2\,, n)$
$res_{full} = ML_{\,latest}\,(data\,,\ n)$
$new\_sup_{list} = res\,.\,\text{get\_top}\,(r\,,\ sup\_rules)$
$BP_{size} = res\,.\,\text{get\_top}\,(m\,,\ not\_new\_sup)$
$data_{txn} = \text{convert\_into\_txn (data)}$
$\text{broadcast}\,(data_{txn}\,,\ new\_sup_{list}\,,\ BP_{size})$

blockchain-compatible transactions and broadcast the results along with this data to the rest of the network.

3.4 Ensuring Fair Participation

In a properly fair system, all honest nodes should have equal opportunities to participate. New nodes should also have a reasonable chance to demonstrate their capabilities once they have been part of the system for a sufficient period. The high levels of trust of older nodes can disproportionately inflate their scores, reducing the opportunities for newer nodes. To address this, we design a set of unique features specifically aimed at preventing repetitive selection, some of which are integrated with the ML model, making the approach suitable for ML-based consensus. Specific systems can add or remove features as needed. Our handling of missing performance data further ensures that new nodes are given a fair chance to participate. In this section, we discuss these novel features and how they can be calculated.

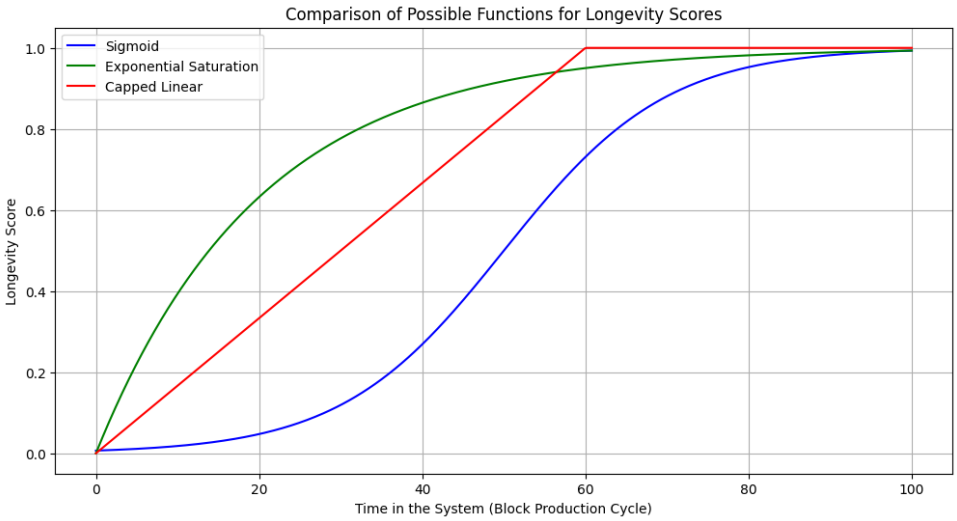


Figure 3.6: The behavior of different functions with increasing node duration in the system.

3.4.1 Longevity Probability

A node should be rewarded for remaining in the system for a sufficient duration. Unless the node engages in any misbehavior, which is detected through performance related features, its probability of selection should increase with the time it has been in the system. However, if this probability continues to grow without limit, new nodes would have little to no chance of participating. Therefore, the probability must reach a maximum after a certain duration. This ensures that new nodes are not unfairly deprioritized once they have been in the system long enough. Figure 3.6 illustrates how several of these functions behave as the input duration increases and shows when and how they reach their maximum.

The capped linear function increases linearly up to a certain duration, after which it reaches a maximum value and remains constant. This ensures that nodes are rewarded for participating in the system while preventing excessive advantage for long-term nodes, giving all nodes equivalent recognition in each round. The exponential saturation function rises quickly at first and then gradually flattens as time increases. This behavior allows nodes to gain substantial credit early on, while preventing their probability from growing indefinitely. The sigmoid function increases slowly at the beginning, accelerates near a midpoint, and then gradually approaches its maximum value. This provides a smooth balance, giving new nodes a fair chance but not after just joining. The equations for these functions are given below, where the parameter k controls the rate of growth and the point at which saturation occurs:

$$
\begin{aligned}
\text{Sigmoid:} \quad & f_{\text{sigmoid}}(t) = \frac{1}{1 + e^{-k(t-t_0)}} \\
\text{Exp. Saturation:} \quad & f_{\text{exp}}(t) = 1 - e^{-kt} \\
\text{Capped Linear:} \quad & f_{\text{capped}}(t) = \begin{cases} k \cdot t & \text{if } t < t_{\text{cap}} \\ 1 & \text{if } t \geq t_{\text{cap}} \end{cases}
\end{aligned}
\tag{3.4}
$$

In our case, we suggest the use of the sigmoid function. Very early on, nodes in the system should not be rewarded too much, unlike in the capped linear or exponential functions. The sigmoid provides a relatively low score for nodes at the beginning. At the midpoint, when they have demonstrated sufficient longevity, they should be rewarded at a higher rate for remaining in the system, while still maintaining a relative score compared to nodes that have been in the system for a long time. Using the sigmoid ensures all of these properties. The value k can be selected based on the system requirements. Additionally, using ML provides a unique opportunity in this case. The value of k can even be treated as a feature of the ML model, allowing the model to learn and adjust it according to the evolving needs of the system.

3.4.2 Starvation Duration

Simply being in the system for a long time does not mean that a node has produced a block or has not produced a block. The starvation duration feature ensures that nodes that have not had the opportunity to produce a block see their chance increase steadily, while nodes that have produced recently are not prioritized. Starvation duration is an integer value that increases by one each time a node expresses interest in block production but is not selected. If a node expresses interest and is selected, this value is reset to 0. Therefore, the node that most recently produced a block will have a starvation duration of 0. On the other hand, this duration decreases by one, down to a minimum of 0, each time a node does not show interest. This ensures that if a node is inactive for an extended period, possibly due to energy or network issues, its starvation duration is reduced, reflecting a lower probability of selection due to their potential unreliability. The calculation for each node is as follows:

$$
SD\left(node_i\right) = \begin{cases} SD\left(node_i\right) + 1 & \text{if } \textit{active} \text{ and } \textit{not selected} \\ 0 & \text{if } \textit{active} \text{ and } \textit{is selected} \\ \max\left(SD\left(node_i\right) - 1\,, 0\right) & \text{if } \textit{inactive} \end{cases}
\tag{3.5}
$$

The starvation duration increases linearly without any saturation, unlike the longevity score. This is because the longer a node waits, the higher its probability of being selected should be. A low value for a recent block producer, combined with an increasingly high value for a node that has waited longer, ensures fair avoidance of repetitive selection. For new nodes, the starvation duration starts at 0 upon joining and gradually increases as they express interest in block production. This treats them in a manner similar to that of older block producers, increasing their probability of selection linearly over time. Alternatively, an exponentially increasing function could be used instead of a linear increase. This would facilitate an even higher probability of selection for nodes that have waited longer, further emphasizing fairness. Note that this feature is not naturally in the range 0 to 1, which is often preferred for supervised models that are sensitive to feature scales. Unlike categorical features that can be converted using one-hot encoding, integer features such as starvation duration are usually normalized or standardized before being used in such models. This is discussed in Section 3.5.4.

3.4.3 Mean ML Score

Each time an active node expresses an interest in participating in the block production process, the ML model predicts a score for it. This score, generally in the range of 0 to 1, indicates the suitability of the node. For each active node, this predicted score can be used as a feature, providing a unique signal without introducing additional overhead to design a new feature. Over multiple rounds, a node may receive predicted scores but may not be selected. Instead of discarding these values, they are accumulated and the mean is computed to generate the feature value. To avoid storing every single score, we maintain the current mean ($ml\text{-}\mu$) and a counter of the number of predictions (c) for the node. To update the mean with the latest predicted score s_{latest}, we multiply the current mean by c, add the new score, and divide the sum by $c+1$. The counter c is then incremented by 1. The equation is as follows:

$$\begin{aligned} ml\text{-}\mu_{\text{new}} &= \frac{ml\text{-}\mu_{\text{current}} \cdot c + s_{\text{latest}}}{c+1} \\ c_{\text{new}} &= c+1 \end{aligned} \tag{3.6}$$

If a node is not active, its mean ML score value remains unchanged. This slightly compensates for the negative effect in the starvation score of not participating . Treating the score as 0 and incrementing c would reduce and saturate the mean too much if the node remains inactive for an extended period. Therefore, resetting both the mean

value and c to 0 is preferable to simply increasing c by 1. Specific use cases can choose the behavior according to their requirements. When a node is selected as a block producer, its mean ML score value is reset to 0. In this way, the feature rewards nodes for not producing a block for a long time while maintaining a high score. The sum of the scores can also be used as a feature, either separately or in place of the mean. This sum can function as a starvation score that does not reset when the node does not participate.

3.4.4 Missing Performance Data

Nodes that do not yet have any experience in block production naturally start with performance-related features at zero, which reduces their chance of being selected. In our system, the participation of new nodes is supported through two mechanisms. The first is the use of the fair participation features described earlier. The second is the selection of a group of nodes as block producers. Even if it is very difficult for a new node to achieve the top-most rank, the repetition-avoidance feature allows them to still be included within the selected group, since selection is not limited to only the top node. Additionally, another possible approach that can be used is to artificially inflate the missing performance data in some controlled manner.

Some values cannot and should not be adjusted, such as the number of blocks produced or the average block production time. A node that has produced at least one block should not receive less credit than a node that has never contributed. However, such features can be balanced with complementary measurements, such as the count of failed block production attempts, where machine learning can also determine the extent of punishment. Other features, on the other hand, can be adjusted. For example, a trust score may be calculated in a non-linear manner, incorporating both positive and negative feedback, similar to EigenTrust [246]. In any case, for malicious or misbehaving nodes, these feature values naturally remain low. Therefore, instead of initializing the values of new nodes at very low levels, a system may choose to start them at a mean or default value that is higher than malicious nodes but lower than fully trusted ones.

3.5 Formulation of the Consensus ML Model

Up to this point, the machine learning model has been presented as a supervised learning framework that integrates multiple features to generate a final suitability score. In this section, we examine the internal architecture of the model in greater detail, out-

lining the functional roles of its components and their contributions to the overall process. The potential features are identified and systematically categorized, followed by a discussion on the procedures required to preprocess and prepare them for effective training.

3.5.1 Supervised Learning Problem

Supervised learning is a fundamental paradigm in machine learning in which a model is trained using labeled data [247]. Formally, the training dataset is defined as follows:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{3.7}$$

where $x_i \in \mathbb{R}^d$ represents the input feature vector of dimension d, $y_i \in \mathcal{Y}$ denotes the corresponding output label, and N is the number of training samples. The learning task involves approximating an unknown target function $f : \mathbb{R}^d \to \mathcal{Y}$ by learning a hypothesis function $h(x;\theta)$ parameterized by θ. The optimization objective is typically expressed as:

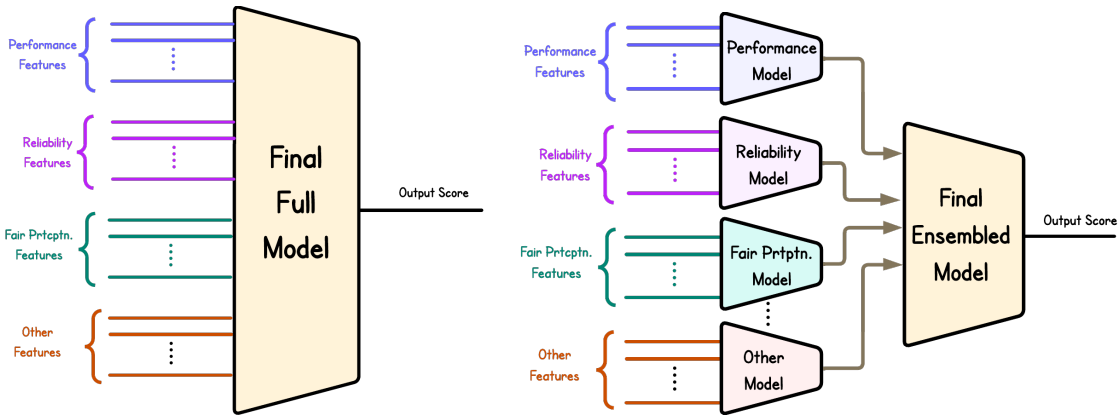
$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}\big(y_i, h(x_i;\theta)\big) + \Omega(\theta), \tag{3.8}$$

where $\mathcal{L}$ is a loss function that measures prediction error and $\Omega(\theta)$ is a regularization term to control model complexity and punish excessive weights. Depending on the choice of model, $\mathcal{L}$ and $\Omega(\theta)$ may take different forms, such as the ℓ_1-norm in Lasso, the ℓ_2-norm in Ridge regression, or more advanced penalty terms in ensemble methods like Gradient Boosting (e.g., XGBoost).

In the context of our proposed consensus mechanism, this supervised learning formulation is applied as a regression-based ranking problem. Each node $j \in \{1, 2, \ldots, N\}$ is associated with a feature vector $x_j \in \mathbb{R}^d$ that captures all types of attributes. The model produces a real-valued prediction $\hat{y}_j = h(x_j;\theta)$, where a higher score reflects greater suitability for block production. Once trained, nodes are ranked by their predicted scores, and a subset of top-ranked nodes is selected as block producers.

3.5.2 Model Structure

The general structure of supervised learning involves training a single model using all available features, where the model learns weights for each feature in order to achieve



(a) Full Combined Model **(b)** Stacked Ensembled Model

Figure 3.7: Comparison of two variations of supervised model structures for ranking nodes. The left model uses all features in a single model, while the right model combines multiple sub-models trained on distinct feature groups. Other features can be grouped differently depending on the use case. We recommend the right variation for best performance.

the best possible predictions. However, when a wide variety of heterogeneous features are included, such as features representing performance, reliability, and fairness, the model's effectiveness may be reduced due to conflicting objectives or complex interactions among features [248]. In such cases, it can be advantageous to divide the feature space into separate groups, train dedicated models for each group, and then combine their outputs through a higher-level model. This approach effectively decomposes the learning task and can be seen as a simplified ensemble method [249], resembling the layered structure of a neural network but in a more modular form.

Figure 3.7 illustrates two variations of the supervised regression model for ranking nodes. The first model, shown in Figure 3.7(a), takes all features as input into a single supervised regression model, where each feature contributes directly to the predicted suitability score. This approach is simple and straightforward but may struggle when the feature set is heterogeneous, as conflicting objectives or interactions among performance, reliability, and fairness-related features can reduce predictive effectiveness. The second variation, shown in Figure 3.7(b), uses a stacked ensemble approach, where features are grouped based on their type or purpose, and separate sub-models are trained for each group. The outputs of these sub-models are then combined through a higher-level model to produce the final node ranking. This decomposition allows the model to capture different aspects of node behavior more effectively, improves robustness to feature interactions, and provides a modular framework that can adapt to changes in feature design or importance. Overall, the ensemble approach can offer better generalization and fairness in node selection, as it combines multiple

models to reduce bias and improve robustness against individual model errors. In contrast, the single-model approach provides simplicity, lower computational overhead, and easier implementation.

3.5.3 Features to Learn

With our proposed approach, the number and types of features are modular and can be adjusted according to the specific use case, highlighting the flexibility and applicability of the consensus mechanism. This allows deployment of unique features tailored to a particular scenario. For example, in an environmental monitoring system, features could include temperature, humidity, air quality, or soil moisture, which can help estimate the probability of node disconnection or failure due to environmental hazards.

The general features we propose include performance-related ones, which track past performance and apply punishment for misbehavior; reliability and availability-related features, which assess current and overall hardware and network conditions that may reduce node availability and block production probability; and fair participation features, which prevent repetitive selection of the same nodes and allow new nodes to join. Although we recommend including these features in all systems, they can be excluded if not required. If no reliability features are needed, the use of supervisor nodes is not necessary, as each node can independently run the inference with the same inputs and models to produce identical results, requiring no additional communication. Then, this approach would be similar to PoEM [10], with the main differences being the selection of a node group and mechanisms to facilitate participation of new nodes.

The extended list of potential features across different categories is presented in Table 3.3. The purpose of the table is to be comprehensive by including virtually all relevant types of features for completeness. Additional features can be designed according to specific requirements, and some can be omitted if not needed. One noteworthy category not previously discussed is the Block Content Features. These features capture information related to the content of previous blocks or the blocks yet to be produced. Although they do not directly measure the inherent capability of a node as a block producer, including them can introduce a small degree of variability among the top-ranked nodes without significantly compromising fairness. Moreover, Block Content Features can help personalize block producer selection for specific blocks. Since the content features can differ for each block, using the same model with the same set of other inputs can still produce slightly different groups of block producers for different

Table 3.3: Extended List of Potential Model Features

Category	List of Possible Features
Past Performance Features	Successfully Produced Block Count, Average Block Time, Failed Block Generation Count, Reputation Score, General Trust Score, Supervisor Role Count, Model Training Data Contribution
Hardware Availability Features	Remaining Battery Life, Energy Harvesting Rate, Power Charging, Current CPU Usage, Remaining Memory Storage, Free CPU Cores, Power Consumption
Network Availability Features	Bandwidth, Throughput, Latency, Packet Loss Rate, Connection Stability, Signal Strength, Link Quality, Network Availability Ratio
Configuration Reliability Features	CPU Power, CPU Frequency, Memory Size, Core Count, Cache Size, Battery Power, Charging Capability
Fair Participation Features	Starvation Length, Longevity Probability, Mean ML Score, Cumulative ML Score, Selection Frequency, Join Date
Block Content Features	Timestamp, Transactions Size, Block Size, Content Hash, Previous Block Hash, Transactions Hash

blocks. This variability makes predicting ML outputs more challenging, which can be advantageous in certain applications.

3.5.4 Data Preparation

In supervised learning, proper preprocessing of input features is crucial to ensure that the model can learn effectively and that all features contribute appropriately to the output [250]. Categorical features, which take values from a finite set of discrete categories, are commonly transformed using one-hot encoding. Given a categorical feature x_i with K possible categories, one-hot encoding represents it as a K-dimensional binary vector $\mathbf{v}_i$ such that:

$$v_{ij} = \begin{cases} 1 & \text{if } x_i \text{ belongs to category } j, \\ 0 & \text{otherwise,} \end{cases} \qquad j = 1, 2, \ldots, K. \tag{3.9}$$

This transformation allows the model to treat each category independently without imposing an ordinal relationship that does not exist.

For numerical features, standardization or normalization is often required to ensure that all features are on a comparable scale. Standardization transforms a feature x_i to have zero mean and unit variance:

$$x_i^{\text{std}} = \frac{x_i - \mu_i}{\sigma_i}, \tag{3.10}$$

where μ_i and σ_i are the mean and standard deviation of feature x_i across all training samples. Alternatively, min-max normalization scales the feature to a fixed range, typically $[0, 1]$:

$$x_i^{\text{norm}} = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}}, \tag{3.11}$$

where $x_i^{\min}$ and $x_i^{\max}$ denote the minimum and maximum values of x_i in the dataset. Both standardization and normalization help prevent features with larger numeric ranges from dominating the model training and improve convergence for gradient-based optimization methods.

For the model in our consensus algorithm, applying these preprocessing steps is necessary for several features. For example, signal strength may fall into discrete categories such as low, medium, or high. Such features should be transformed using one-hot encoding. Other numerical features that are not naturally within the range $[0, 1]$ should be standardized or normalized. Hence, depending on the dataset, all features need to be appropriately converted before training the model and performing the inference.

3.5.5 Continuous Training

The continuous learning concept of our consensus algorithm comes from training the model with live system data. Throughout the system lifecycle, the ML model adapts over time to provide a better selection of block producers that reflects the changing needs of the network. At each block production cycle, some block producers successfully generate a block, while others fail or produce an invalid block. In either case, a node with certain input features either succeeds or fails in block production. This outcome serves as the label, while the features used during inference are treated as inputs to retrain the model. Because this information is recorded on the blockchain, it can be reused for training. Training may occur after each block producer completes its task or after all producers in a cycle have finished. For efficiency and performance,

we recommend training after a group of block producers has completed, since initiating training consumes resources [251] but the incremental improvement when using just one individual block is minimal.

The training dataset can be represented mathematically by a matrix $\mathcal{D}$ as follows:

$$\mathcal{D} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2d} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} & y_n \end{bmatrix}.$$

Here, each row corresponds to a single block producer during a block production cycle. The first d columns of each row, $x_{i1}, x_{i2}, \ldots, x_{id}$, form the feature vector describing the state of node i at the time when it was selected as block producer. The last column, y_i, serves as the label indicating the outcome of the node's block production attempt. A label of 1 is assigned if the node successfully produces a valid block, and a label of 0 is assigned if the node fails or produces an invalid block.

Our consensus algorithm is most suitable for business-oriented use cases, particularly those applied in BaaS IoT systems. Unlike cryptocurrency-focused mechanisms, the main incentive here is to build trust in the system to maintain critical data in areas such as healthcare, agriculture, smart cities, and environmental monitoring. These use cases may operate on either private or public blockchains, and our consensus algorithm is designed to support both. In private BaaS IoT systems, the training of our consensus algorithm can be managed by an upper-layer server connected to the IoT devices. This server does not store or process the blockchain data itself, which preserves the benefits of a decentralized ledger, but it can still coordinate and control the training process. In public BaaS IoT systems, where no upper server exists, training can instead be carried out through federated learning [252]. Several studies have proposed methods to improve the security of federated learning [253], and our mechanism is flexible enough to adopt any of these approaches as required.

3.6 Initial Consensus Model Generation

At the very beginning of an IoT deployment, there is no reliable model available due to the lack of sufficient training data, which makes accurate decision-making difficult. Therefore, it is necessary to design methods that allow the system to initialize a model capable of making reasonable predictions from the start, even with minimal

data or prior knowledge. In this section, we propose several solutions and initialization strategies that a system can use to address this challenge and effectively create an initial model that can later be refined as more data becomes available.

3.6.1 Pre-trained Model

Investing resources to build a full machine learning model from scratch is rare these days. Rather, a large number of real-world machine learning applications use transfer learning approaches [254]. The idea of transfer learning is to take advantage of an already trained machine learning model from related domains to achieve better performance by slightly fine-tuning the model weights [255]. The usefulness of transfer learning is especially evident in applications such as Natural Language Processing (NLP), Speech Recognition, or Image Processing, where it is either practically impossible or prohibitively expensive to collect sufficient amounts of training data [256].

The concept of transfer learning is inspired by the way people can apply previously acquired knowledge to solve new problems more quickly or even with improved solutions [257]. For example, a student who is proficient in C can learn languages like C++ or Java, or even a quite different language like Python, much faster compared to the time required to learn their first programming language. This is why, in formal education at institutions such as universities, one language is typically taught thoroughly, while other languages are approached in an application-oriented way rather than being taught in detail. This phenomenon is closely related to human psychology. The first language learned by a child does not have any prior reference, but when people learn a second or foreign language, they often try to connect it to the grammar, words, and alphabets already acquired from their first language [258].

The cost of training supervised learning methods is relatively low, but the challenge of data scarcity is still a problem that transfer learning can solve. This suggests that a pre-trained model could be beneficial in our consensus algorithm, allowing it to efficiently learn how to select a suitable block producer based on past experience in different domains. For instance, an application using our mechanism in healthcare could quickly fine-tune the model for agriculture-related tasks. Many performance-related features may still have similar effects, with the main differences arising from device types or domain-specific features. The most difficult part, however, is actually finding a suitable pre-trained model. Real-world blockchain applications that incorporate machine learning are still limited, which means there are currently no readily available models to reuse. Still, keeping this option in mind makes the consensus mechanism more adaptable and future-proof.

3.6.2 Custom Model

An alternative to using a model pre-trained on real-world applications is to artificially generate a custom model, either in a simulation environment, by manually annotating randomly generated data, or by assigning suitable weights to each feature. While such models may not perform as well as those pre-trained on real-world data, they can still serve as a strong starting point if proper care is taken to replicate real-world conditions. Over time, the custom model can adapt to the specific use case. A key advantage of this approach is that the model can be tailored to the logic and requirements of the target application from the very beginnin, something that may not always be possible with a pre-trained model developed for different domains.

A proper simulation environment for an IoT-based blockchain platform would ideally account for a wide range of variables. However, replicating every factor of the real-world environment is practically infeasible. In most cases, simulations approximate reality by assuming reasonable or closely related values for many of these variables. Although not identical to a model pre-trained on the exact environment, such simulated approaches can still be more effective than starting without a model at all, as will be discussed in the following sections. The same applies to randomly generated datasets without labels. With sufficient manual effort and proper quality control, these datasets can be annotated to include meaningful labels [259]. Typically, this involves assigning labels to each data row, for instance, a binary label (0 or 1) to indicate whether the annotator believes a node should act as a block producer. Additional labels may be used to filter out rows with unrealistic feature relationships, which can easily arise when data is generated randomly. For example, it would be unrealistic for a node to have a poor trust score if it had never produced a block. Another possible approach is to directly manipulate the weights or decision-making process of the machine learning model. However, this requires a deeper understanding of the inner workings of the model, which can be particularly challenging for more complex architectures, such as XGBoost.

3.6.3 Cold Startup

To address the challenge of achieving consensus before a machine learning model has been trained with sufficient data, the concept of cold startup in the initial stage of BaaS-based IoT systems was introduced in PoEM [10]. Their approach proposes temporarily relying on existing consensus protocols such as PoW, PoS, or DPoS until the system gathers enough data to train an effective model. Depending on the requirements of a given use case, an appropriate mechanism can be selected. The authors

further argue that IoT systems can learn the operational characteristics of these traditional consensus mechanisms and subsequently improve upon them. In principle, any rule based consensus mechanism can serve as a temporary solution during the cold startup phase. This cold startup approach can also be seamlessly integrated into our proposed consensus algorithm. However, a general drawback of this approach is that the selected initial mechanism may not always be well suited to the resource constraints of IoT devices. Nevertheless, in scenarios where developing a reliable custom model is infeasible, a pre-trained model is unavailable, or the use case demands the proven security guaranties of established consensus mechanisms, this strategy provides a practical and robust alternative.

3.6.4 Rule Based Mechanism

Most of the existing mechanisms are rule based and cold startup phase requires to use these rule based consensus mechanism in the initial stage. One interesting alternative of this can be to use some rule based mechanism without including any consensus algorithms itself. These rule based mechanism still requires some small level of randomness for security purposes. One such mechanism was proposed in the consensus mechanism Period Proof of Stake (PPoS) by Anan et al. [86]. In PPoS, the selection of the block producer follows a rule-based mechanism in which each node is assigned a score that combines randomization, trust, and repetition values. During the selection phase, each node i receives a temporary random factor $r \in [0, 1]$. The selection score is then calculated as:

$$\text{Score}_i = r + \beta \cdot \frac{T_i}{T_{\max}} - \gamma \cdot \frac{R_i}{2^{32}} \tag{3.12}$$

where T_i is the trust value of node i, $T_{\max}$ is the maximum trust value among all nodes, R_i represents the repetition penalty for repeated selections, and β and γ are scaling factors that control the relative importance of trust and repetition. The node with the maximum Score_i is then selected as the block producer. After each round, the trust values are updated based on performance: successful block producers receive an increment in trust, while misbehaving nodes are penalized through a left-shift operation on their trust value, thereby reducing their likelihood of being selected in future rounds.

A rule-based mechanism of this type ensures that block producer selection balances trustworthiness and fairness while maintaining the ability to adapt to the dynamic behavior of a specific use case. It also provides greater flexibility to capture and reflect

the behavior patterns of a particular application. For instance, rules can incorporate current hardware states, which aligns with the design of our consensus mechanism but is not commonly implemented in other existing protocols. Nevertheless, the absence of formal guaranties from established consensus mechanisms may introduce potential security risks, which must be carefully considered.

Chapter 4

Results and Analysis

In this chapter, we present both the theoretical and numerical analyses of our consensus algorithm. We begin by examining its security aspects, analyzing various characteristics that contribute to the overall security of the blockchain. Next, we study its complexity in an asymptotic sense and compare the results with other consensus mechanisms. We also evaluate the algorithm in the context of the CAP theorem, discussing how it maintains the fundamental properties of a distributed system. In the following section, we describe the setup used for the numerical analysis and briefly outline the other consensus algorithms included for comparison, along with the evaluation metrics applied. Finally, we present the results using line charts and bar graphs.

4.1 Characteristics Analysis

Our consensus algorithm preserves the consistency and key advantages of existing blockchain ledgers while enhancing the performance and security aspects of IoT networks. To understand this, it is first important to understand its characteristics with theoretical results before diving into proper simulations. In this section, we first discuss how the proposed mechanism ensures security and the incentives that motivate nodes to behave honestly. We then present a theoretical analysis, focusing on the asymptotic behavior of block production, communication, and computational complexity, along with relevant definitions. Finally, we examine how the algorithm achieves decentralization and scalability compared to other consensus mechanisms, and analyze its properties in the context of the CAP theorem.

4.1.1 Security Analysis

Our consensus algorithm enhances the unpredictability of block producer selection by introducing a "black-box" mechanism that prevents participants from forecasting or manipulating who will generate the next block. This design greatly increases randomness and fairness in the selection process. As a result, both internal and external adversaries face significant uncertainty, making it computationally infeasible to predict or impersonate the upcoming producer within the allowable time frame.

The protocol also reduces the motivation to launch 51% attacks or other malicious behaviors by aligning system incentives with honest participation. In our consensus algorithm, incentives and rewards are tied to the performance and trust scores that nodes earn through their contributions to the machine learning model's performance component.Through a game-theoretic perspective, the interaction between honest nodes and potential attackers can be represented as a mixed-strategy game. In equilibrium, both parties achieve optimal expected returns, ensuring that no participant gains additional benefits from deviating unilaterally. This equilibrium condition discourages attackers from attempting to dominate the network, as the cost and risk outweigh the potential gain [260].

The utility analysis further demonstrates that rational nodes are incentivized to act honestly under equilibrium conditions. The probability of malicious behavior decreases when honest nodes gain higher rewards for correctly identifying or rejecting invalid blocks, and when the penalties for failing to do so increase. Conversely, reducing the rewards for accepting suspicious blocks or the penalties for rejecting valid ones further stabilizes the system. In general, the model ensures that cooperative behavior remains the most rational and profitable strategy for all participants, strengthening both the security and reliability of the consensus mechanism.

4.1.2 Asymptotic analysis

Asymptotic analysis is the study of an algorithm's behavior and resource requirements as the input size approaches infinity, providing a theoretical measure of its scalability and efficiency. In the context of our consensus mechanism, we analyze the time required to perform different aspects with an increasing number of inputs in the long run. For this, we analyze three different definitions: block production, communication, and computing complexity. Block production complexity refers to how many operations are required to produce blocks, which in other words means selecting a block producer, as one block is produced by one block producer. Computing time complex-

Table 4.1: Asymptotic Complexity Comparison of Consensus Mechanisms

Consensus Method	Consensus Process	Block Producer Selection Complexity	Computing Time Complexity	Communication Time Complexity
PoW	Computation	$O(k)$	$O(b)$	$O(n)$
PoS	Computation	$O(k)$	$O(b)$	$O(n)$
DPoS	Computation	$O(k/c)$	$O(b)$	$O(n)$
PBFT	Communication	$O(k)$	$O(b)$	$O(n^2)$
PoA	Communication	$O(k)$	$O(b)$	$O(n)$
PoET	Computation	$O(k)$	$O(b)$	$O(n)$
PoEM	Model	$O(k)$	$O(1)$	$O(n)$
DP-PoEM (ours)	Model	$\mathcal{O}(\sqrt{k})$	$\mathcal{O}(1)$	$\mathcal{O}(n)$

ity represents the computational consumption for each node to reach an agreement. Finally, communication complexity indicates the amount of message passing required in general to reach consensus. The complexity of our consensus algorithm compared to others is presented in Table 4.1.

The block producer selection complexity of all other consensus mechanisms is in the order of k because to select one block producer, the consensus is required to run once. For example, in PoW, for one block producer selection, one nonce is required to be found; in PoS, the staking lottery is considered once for each block producer; and in PoEM, the ML model is run once to find the highest-ranked node to be selected as block producer. For ours, as we select a group of block producers, this value is lower than that of other consensus mechanisms. As we choose the number of block producers as the square root of the total nodes, we present it as $\mathcal{O}(\sqrt{k})$, but it can differ slightly if a specific implementation chooses a different relation. Only for DPoS, it is k/c, as it chooses a group of delegates where the variable c determines the specific logic of group selection.

For computing complexity, the term b depends on the specific consensus and is not a direct representation of the same inputs, such as the number of nodes. For PoW, b depends on the length of the hash; for PoET, it depends on the random wait time; and for PoA, it depends on trust calculation within the authority. For PoEM and our consensus mechanism, DP-PoEM, it is constant because the computation only involves running the ML model. The communication complexity of ours also remains the same as that of other consensus mechanisms with linear complexity because the number of supervisors is considered mostly constant. For PBFT, this is quadratic, as every node is required to send messages to every other node.

Table 4.2: Performance Comparison of Consensus Mechanisms

Consensus Method	Decentralization	Scalability	CAP	Adversary Tolerance
PoW	High	Low	AP	< 50% computing
PoS	Medium	Medium	AP	< 50% stakes
DPoS	Low	Very High	CP	< 50% stakes
PBFT	Low	Low	CP	< 33% malicious nodes
PoA	Low	Low	AP	< 50% faulty nodes
PoET	Medium	High	AP	< 50% faulty nodes
PoEM	High	High	CP	< 50% faulty nodes
DP-PoEM (Ours)	High	Very High	CP	< 50% faulty nodes

4.1.3 Performance analysis

The overall theoretical improvement of our consensus mechanism in terms of performance is presented in Table 4.2. This table demonstrates the applicability of our consensus mechanism within a BaaS-based IoT system. Overall, the scalability of our mechanism is considered very high, similar to DPoS, as we select a group of block producers instead of a single one, enabling better scalability with an increasing number of nodes. However, in DPoS, the use of a fixed set of nodes to select delegates significantly reduces decentralization. In contrast, our approach maintains decentralization comparable to more open mechanisms such as PoEM or PoW, where all nodes can participate and be selected as block producers. Moreover, due to our fair participation features, it can even be considered more decentralized.

We discussed the CAP theorem in the context of blockchain in Section 2.1.6, which states that only two of Consistency (C), Availability (A), and Partition Tolerance (P) can be fully maintained simultaneously. Table 4.2 presents how different consensus mechanisms, including ours, align with these characteristics in blockchain systems. All blockchain-based consensus mechanisms inherently maintain Partition Tolerance due to the distributed nature of the network. Mechanisms such as PoW, PoS, and PoA, which do not rely on a fixed set of entities to maintain consensus, generally prioritize Availability over Consistency. In contrast, mechanisms like PoEM and DP-PoEM rely on a model-driven decision process that can delay availability to preserve consistency. Similarly, DPoS depends on a fixed set of delegate nodes, leading it to favor Consistency over Availability. Therefore, these mechanisms are better characterized as CP systems rather than AP systems.

Similar to PoA, PoET, and PoEM, our consensus mechanism can tolerate up to 50% faulty nodes. In most consensus mechanisms, the term “faulty nodes” represents

the measure of adversary tolerance; however, this interpretation varies slightly across different methods. For instance, in PoW, nonce calculations are performed in a distributed manner, so the security depends not directly on the number of faulty nodes but on the proportion of total computing power controlled by adversaries—requiring it to remain below 50%. Likewise, in PoS and DPoS, the system remains secure as long as adversarial entities control less than 50% of the total stake in the network.

4.2 Experimental Settings

The full-scale simulation of blockchain systems remains an active area of research, with several studies proposing methods to realistically simulate the performance of blockchain consensus mechanisms within real-world network environments. Numerous researchers have developed simulators aimed at modeling either the entire blockchain or specifically its consensus component. BlockSim [261] is one such blockchain simulator that enables discrete-event simulation of blockchain processes. To focus more on the consensus layer, several dedicated consensus simulators have also been proposed, including but not limited to SimBlock (2019) [262] and JABS (2023) [263]. However, these simulators often suffer from limited customizability for integrating new or unconventional consensus mechanisms and tend to face compatibility issues with evolving software environments due to poor maintainability. Therefore, we developed our own numerical simulation framework to analyze and compare the performance of our proposed consensus mechanism. In this section, we discuss our simulation setup, the performance metrics used for evaluation, and briefly describe how the compared consensus mechanisms were modeled.

4.2.1 Simulation Environment

We developed a numerical simulation environment in Python 3.10 to analyze the performance of our proposed consensus mechanism. An alternative would be to use packet-level or event-driven simulation. The environment simulates a BaaS-based IoT network, where each IoT node acts as a blockchain participant responsible for block generation, message propagation, and model-based verification. To maintain uniformity, all nodes are simulated as identical hardware entities, modeled after a dummy IoT device configured with an Intel Core i7 processor (2.8 GHz) and 16 GB RAM. The metrics are derived using closed-form probabilistic models, such as binomial success probability for block producer selection and epidemic broadcast delay for message propagation. To ensure fairness, the number of simulated nodes n is varied from 50

to 1000 (i.e., 50, 100, 250, 500, 750, 1000), and each experiment is repeated 100 times, with average values reported. Network parameters such as node online probability and block producer failure rate are systematically swept to assess robustness.

4.2.2 Performance Metrics

To evaluate our consensus mechanism comprehensively, we use the following three metrics.

Average Latency (s): It is measured as the expected time required to successfully produce one block. This is determined by considering the time that a particular consensus protocol takes to select a block producer, followed by the time the block producer spends to successfully generate the block.

Computation Overhead: It is measured as the total expected computational effort per node to participate in the consensus process. This effort generally includes block producer selection and verification of the correct block. Depending on the specific consensus mechanism, it may include hashing, staking, model inference, or other computational tasks.

Active Participation Time (s): It is measured as the expected time a node actively contributes to the consensus process. If a node is not selected as a block producer, or knows it cannot be selected, it may remain inactive for a period, which is particularly important for IoT devices seeking to preserve energy.

4.2.3 Compared Consensus Mechanisms

To evaluate our consensus mechanism, DP-PoEM, we compare its performance with three historical consensus mechanisms and four state-of-the-art mechanisms:

PoW: Each node repeatedly calculates hash values by iterating through a nonce until a difficulty target is met, which we set to 5 (number of leading zeros). The first node to find a valid nonce produces the block.

PoS: Similar to PoW, but selection is based on stake rather than computation. It works like a lottery system, where each node holds "tickets" proportional to its stake, giving nodes with higher stakes a higher chance of being selected as the block producer.

PBFT: Typically used in permissioned blockchains, PBFT reaches consensus through a three-phase message exchange process among all participating nodes.

DPoS: Nodes with the highest stakes elect a group of delegates, and only the selected delegates are eligible to produce blocks.

PoA: A simplified variant of PBFT that requires fewer message exchanges, making it more efficient for permissioned settings.

PoET: Each node in a trusted execution environment (TEE) is assigned a random wait time, and the node with the shortest wait time is selected to produce the block.

PoEM: A machine learning model, maintained in a distributed manner, selects the node with the highest suitability score as the block producer. The model is incrementally updated and run once for each block producer selection.

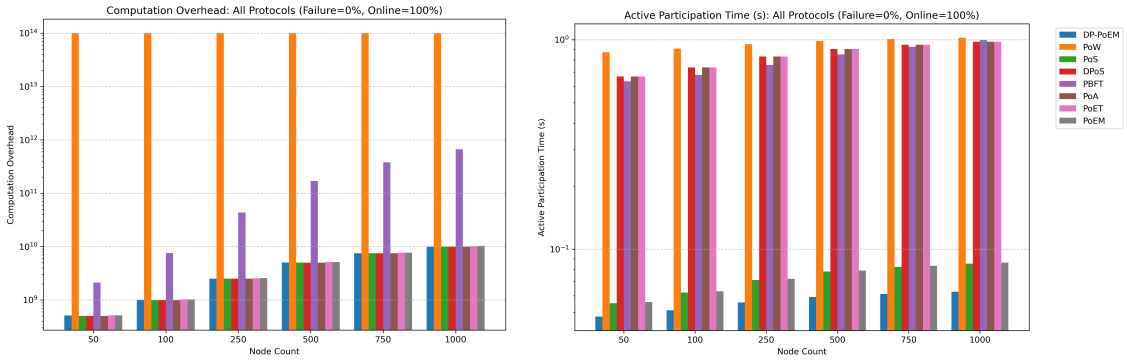
4.3 Results Analysis

In this section, we present the results of all performance metrics across the selected consensus mechanisms. We first provide an overall comparison with all other consensus mechanisms and then discuss PoEM in more detail, as it is another machine learning-based consensus mechanism similar to ours. Finally, we briefly examine how the consensus mechanisms perform in terms of average block production latency under extreme conditions, such as when fewer nodes are online or when a larger number of nodes fail.

4.3.1 Performance Comparison with All Mechanisms

The performance in terms of computational overhead and active participation time is illustrated in Figure 4.1. As shown in Figure 4.1(a), the computational overhead of PoW is significantly higher than that of other consensus mechanisms due to the intensive hash computations required. PBFT shows a moderate level of overhead, while the remaining mechanisms exhibit relatively similar performance. Our consensus mechanism achieves slightly better results than others, primarily due to the efficiency of the machine learning model and the reduced number of required computations. As discussed in the following section, this leads to a marginal improvement over PoEM. In terms of active participation time (from Figure 4.1(b)), the performance advantage of our consensus mechanism becomes more evident. Because a group of block producers is selected simultaneously, the overall participation time is reduced. Although DPoS also employs a group-based approach, its round-robin operation makes it comparatively less efficient.

The performance in terms of average block production latency (in seconds) is shown



(a) Computational Overhead **(b)** Active Participation Time

Figure 4.1: Computational Overhead and Active Participation Time Comparison with all consensus mechanisms with increasing number of nodes.

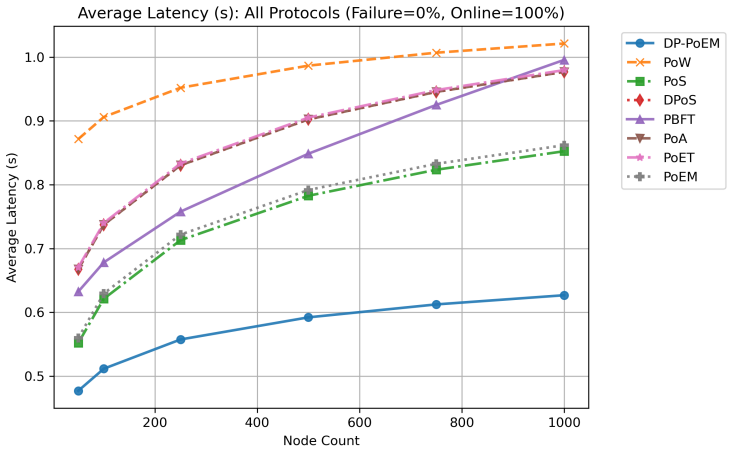
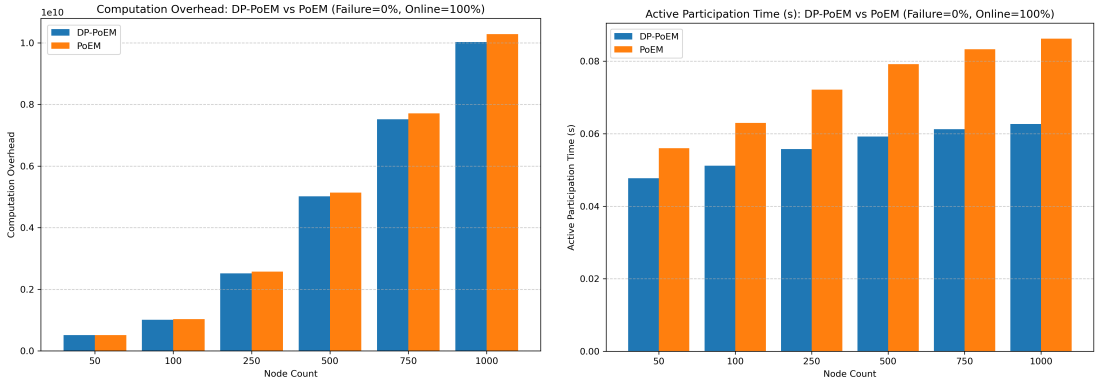


Figure 4.2: Average Latency Comparison with all consensus mechanisms with increasing number of nodes.

in Figure 4.2. Compared to other metrics, the latency values remain within a comparable range across all consensus mechanisms. This is why a line chart is chosen to present the results instead of a bar graph, as it more effectively illustrates performance trends. However, it is evident from the curve that our proposed mechanism exhibits a slower rate of latency increase compared to others. This improvement is primarily attributed to the selection of multiple block-producing nodes, a feature not directly present in most other mechanisms. Additionally, increasing the size of the producer groups further contributes to the reduction in latency. Since our consensus mechanism maintains both security and decentralization, this analysis highlights its superior balance of efficiency and robustness compared to other approaches.



(a) Computational Overhead **(b)** Active Participation Time

Figure 4.3: Computational Overhead and Active Participation Time Comparison with PoEM with increasing number of nodes.

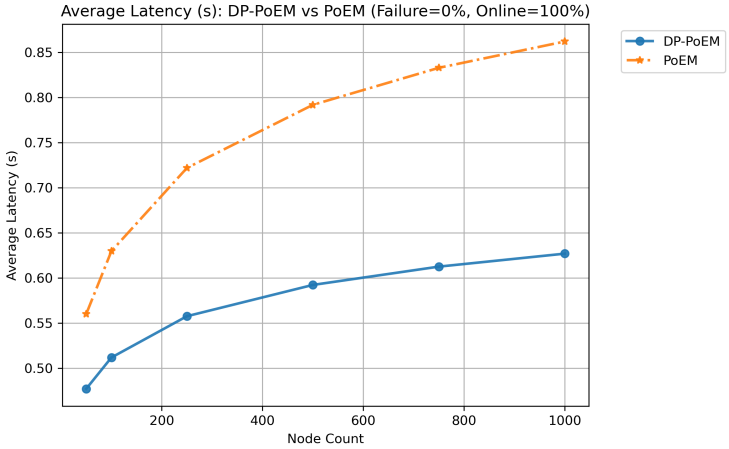


Figure 4.4: Average Latency Comparison with PoEM with increasing number of nodes.

4.3.2 Performance Comparison with POEM

PoEM is another consensus mechanism that employs a machine learning model similar to ours but follows a different design approach. Since our proposed mechanism also integrates machine learning, we provide dedicated figures—Figure 4.3 and Figure 4.4—to enable a focused comparison between the two.

As shown in Figure 4.3(a), the computational overhead of both mechanisms is almost identical, which aligns with the general definition of this metric. Both require machine learning inference, block verification, and related computations. The number of inference operations performed does not directly influence this metric. However, the active participation time for our proposed mechanism increases much more slowly compared to PoEM, as illustrated in Figure 4.3(b). This is because, once the

group of block producers is determined, non-selected nodes can remain inactive for a longer period, conserving resources while maintaining system readiness.

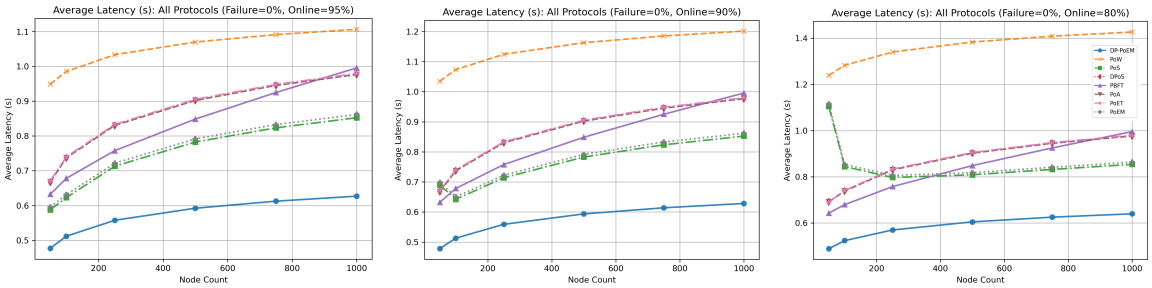
In contrast, the average latency of PoEM increases almost linearly beyond approximately 250 nodes, as the time spent on ML inference becomes the dominant contributor to overall latency. This corresponds to the fact that PoEM computes a suitability score for each new node individually. In our mechanism, however, the latency grows much more slowly and non-linearly, as the group size scales with the square root of the total number of nodes, thereby reducing the impact of network expansion, as shown in Figure 4.4.

4.3.3 Performance Comparison in Exceptional Environments

A robust consensus mechanism should be capable of maintaining stable and reliable performance even under difficult or exceptional network conditions. Although such situations occur infrequently, they represent critical stress points that test the resilience and adaptability of any distributed system. If a consensus mechanism collapses or exhibits excessive delay under these conditions, it may lead to network stalls, data inconsistency, or even temporary system unavailability. Hence, ensuring operational continuity in the presence of failures or partial outages is a vital indicator of real-world applicability, especially for IoT-based blockchain networks where connectivity and resource availability can fluctuate frequently.

Exceptional conditions can take various forms—from a few devices going offline temporarily to widespread failures caused by environmental or infrastructural disruptions. For instance, adverse weather events such as storms or heavy rainfall may result in power loss across multiple IoT devices in a region, while network congestion or gateway malfunction may disconnect several nodes simultaneously. To capture such realistic failure patterns, we simulate a scenario where a certain percentage of nodes become offline and analyze its impact on the average block production latency. It is also important to note that a node may be offline not only because of permanent failure but also due to deliberate energy-saving behavior, temporary maintenance, or bandwidth optimization—common practices in IoT networks that prioritize energy efficiency.

Figure 4.5 illustrates the average latency of different consensus mechanisms as the proportion of online devices decreases. As the number of active nodes decreases, the performance of PoEM and PoS deteriorates more rapidly, resulting in a significant increase in latency. This is because PoEM relies primarily on static hardware-based



(a) 95% Online **(b)** 90% Online **(c)** 80% Online

Figure 4.5: Average Latency Comparison with all consensus mechanism with when number of online nodes decreases.

features of devices, which are not indicative of their current online status. Consequently, offline devices can still be selected as block producers, further delaying block generation—especially when the total number of nodes is low. In contrast, our proposed consensus mechanism, DP-POEM, dynamically selects block producers based on their real-time activity status. This adaptive selection process substantially reduces the likelihood of choosing offline nodes, allowing the system to maintain efficient block production even under stress. Therefore, our consensus mechanism demonstrates strong resilience and reliability in exceptional operating environments.

Chapter 5

Conclusion

In this work, we propose DP-POEM, a lightweight, machine learning-based consensus mechanism that introduces a periodic consensus process through group-based block producer selection and leverages dynamically changing features of IoT devices. This design is particularly suitable for blockchain-enabled IoT platforms, especially those utilizing BaaS frameworks. To efficiently manage dynamic features, we employ a rotating set of supervisor nodes responsible for collecting and sharing device status information. Furthermore, to ensure fair participation across the network, DP-POEM incorporates mechanisms that prevent repetitive selection of the same nodes while allowing new nodes to gradually join the consensus process. The proposed mechanism is theoretically analyzed in terms of security, consistency, and scalability, and its effectiveness is further validated through extensive numerical simulations. The evaluation results demonstrate that DP-POEM significantly outperforms existing consensus approaches in scalability, efficiency, and applicability to dynamic IoT environments.

Future work may explore alternative machine learning paradigms, such as unsupervised or reinforcement learning, while maintaining a lightweight design to further enhance the block producer selection process in dynamic IoT environments. Another promising direction is the development of efficient methods for distributed training and model generation across decentralized networks such as blockchain, reducing the reliance on centralized computation. Additionally, future research could investigate adaptive consensus mechanisms capable of dynamically adjusting parameters or switching strategies based on network conditions, node performance, and device energy levels. Finally, new IoT-oriented consensus designs can be developed that incorporate energy optimization techniques directly into the consensus process, thereby achieving higher energy efficiency without compromising security or scalability.

Bibliography

[1] R. D. Atkinson and D. Castro, "Digital quality of life: Understanding the personal and social benefits of the information technology revolution," *Available at SSRN 1278185*, 2008.

[2] N. Kaushik and T. Bagga, "Internet of things (iot): Implications in society," in *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*, 2020.

[3] B. Cao et al., "When internet of things meets blockchain: Challenges in distributed consensus," *Ieee Network*, vol. 33, no. 6, pp. 133–139, 2019.

[4] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, "A survey on the adoption of blockchain in iot: Challenges and solutions," *Blockchain: Research and Applications*, vol. 2, no. 2, p. 100 006, 2021.

[5] M. H. Miraz and M. Ali, "Applications of blockchain technology beyond cryptocurrency," *arXiv preprint arXiv:1801.03528*, 2018.

[6] X. Wang et al., "Survey on blockchain for internet of things," *Computer Communications*, vol. 136, pp. 10–29, 2019.

[7] D. Li, L. Deng, Z. Cai, and A. Souri, "Blockchain as a service models in the internet of things management: Systematic review," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 4, e4139, 2022.

[8] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE access*, vol. 9, pp. 43 620–43 652, 2021.

[9] N. Chaudhry and M. M. Yousaf, "Consensus algorithms in blockchain: Comparative analysis, challenges and opportunities," in *2018 12th international*

conference on open source systems and technologies (ICOSST), IEEE, 2018, pp. 54–63.

[10] Y. Zhao, Y. Qu, Y. Xiang, Y. Zhang, and L. Gao, "A lightweight model-based evolutionary consensus protocol in blockchain as a service for iot," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2343–2358, 2023.

[11] L. P. Nian and D. L. K. Chuen, "Introduction to bitcoin," in *Handbook of digital currency*, Elsevier, 2015, pp. 5–30.

[12] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies*, 2020.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[14] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain: research and applications*, vol. 3, no. 2, p. 100 067, 2022.

[15] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[16] H. Liu, X. Luo, H. Liu, and X. Xia, "Merkle tree: A fundamental component of blockchains," in *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, IEEE, 2021, pp. 556–561.

[17] Y. Yu, Y. Li, J. Tian, and J. Liu, "Blockchain-based solutions to security and privacy issues in the internet of things," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 12–18, 2019.

[18] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, "Cryptographic primitives in blockchains," *Journal of Network and Computer Applications*, vol. 127, pp. 43–58, 2019.

[19] M. Parmar and H. J. Kaur, "Comparative analysis of secured hash algorithms for blockchain technology and internet of things," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 3, 2021.

[20] D. Boneh, "Aggregate signatures," in *Encyclopedia of Cryptography and Security*, Springer, 2011, pp. 27–27.

[21] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Annual international cryptology conference*, Springer, 1997, pp. 410–424.

[22] C. Li, Y. Tian, X. Chen, and J. Li, "An efficient anti-quantum lattice-based blind signature for blockchain-enabled systems," *Information Sciences*, vol. 546, pp. 253–264, 2021.

[23] A. Manzoor, A. Braeken, S. S. Kanhere, M. Ylianttila, and M. Liyanage, "Proxy re-encryption enabled secure and anonymous iot data sharing platform based on blockchain," *Journal of Network and Computer Applications*, vol. 176, p. 102 917, 2021.

[24] N. Szabo, "Smart contracts: Building blocks for digital markets," *EXTROPY: The Journal of Transhumanist Thought,(16)*, vol. 18, no. 2, p. 28, 1996.

[25] V. Buterin et al., "Ethereum white paper," *GitHub repository*, vol. 1, no. 22-23, pp. 5–7, 2013.

[26] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Ethereum smart contract analysis tools: A systematic review," *IEEE Access*, vol. 10, pp. 57 037–57 062, 2022.

[27] H. Taherdoost, "Smart contracts in blockchain technology: A critical review," *Information*, vol. 14, no. 2, p. 117, 2023.

[28] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-peer Networking and Applications*, vol. 14, no. 5, pp. 2901–2925, 2021.

[29] C. Profentzas, M. Almgren, and O. Landsiedel, "Tinyevm: Off-chain smart contracts on low-power iot devices," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2020, pp. 507–518.

[30] P. Paul, P. Aithal, R. Saavedra, and S. Ghosh, "Blockchain technology and its types—a short review," *International Journal of Applied Science and Engineering (IJASE)*, vol. 9, no. 2, pp. 189–200, 2021.

[31] C. Komalavalli, D. Saxena, and C. Laroiya, "Overview of blockchain technology concepts," in *Handbook of research on blockchain technology*, Elsevier, 2020, pp. 349–371.

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

[33] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, and S. Shimizu, "Privacy preservation in permissionless blockchain: A survey," *Digital Communications and Networks*, vol. 7, no. 3, pp. 295–307, 2021.

[34] V. J. Morkunas, J. Paschen, and E. Boon, "How blockchain technologies impact your business model," *Business Horizons*, vol. 62, no. 3, pp. 295–306, 2019.

[35] E. Strehle, "Public versus private blockchains," *BRL working paper, Blockchain Research Lab, Tech. Rep.*, 2020.

[36] A. R. Khettry, K. R. Patil, and A. C. Basavaraju, "A detailed review on blockchain and its applications," *SN Computer Science*, vol. 2, no. 1, p. 30, 2021.

[37] H. W. Marar and R. W. Marar, "Hybrid blockchain," *Jordanian Journal of Computers and Information Technology (JJCIT)*, vol. 6, no. 4, pp. 317–325, 2020.

[38] B. Cao et al., "Performance analysis and comparison of pow, pos and dag based blockchains," *Digital Communications and Networks*, vol. 6, no. 4, pp. 480–485, 2020.

[39] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, "A comparative analysis of dag-based blockchain architectures," in *2018 12th International conference on open source systems and technologies (ICOSST)*, IEEE, 2018, pp. 27–34.

[40] S. S. Sabry, N. M. Kaittan, and I. Majeed, "The road to the blockchain technology: Concept and types," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 7, no. 4, pp. 1821–1832, 2019.

[41] K. K. Vaigandla, R. Karne, M. Siluveru, and M. Kesoju, "Review on blockchain technology: Architecture, characteristics, benefits, algorithms, challenges and applications," *Mesopotamian journal of Cybersecurity*, vol. 2023, pp. 73–84, 2023.

[42] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–34, 2019.

[43] E. A. Lee, S. Bateni, S. Lin, M. Lohstroh, and C. Menard, "Quantifying and generalizing the cap theorem," *arXiv preprint arXiv:2109.07771*, 2021.

[44] R. Wattenhofer, *The science of the blockchain*. CreateSpace Independent Publishing Platform, 2016.

[45] J. Frizzo-Barker, P. A. Chow-White, P. R. Adams, J. Mentanko, D. Ha, and S. Green, "Blockchain as a disruptive technology for business: A systematic review," *International Journal of Information Management*, vol. 51, p. 102 029, 2020.

[46] I. Abu-Elezz, A. Hassan, A. Nazeemudeen, M. Househ, and A. Abd-Alrazaq, "The benefits and threats of blockchain technology in healthcare: A scoping review," *International Journal of Medical Informatics*, vol. 142, p. 104 246, 2020.

[47] M. M. Queiroz, R. Telles, and S. H. Bonilla, "Blockchain and supply chain management integration: A systematic review of the literature," *Supply chain management: An international journal*, vol. 25, no. 2, pp. 241–254, 2020.

[48] J. Xie et al., "A survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2794–2830, 2019.

[49] T. Alladi, V. Chamola, N. Sahu, V. Venkatesh, A. Goyal, and M. Guizani, "A comprehensive survey on the applications of blockchain for securing vehicular networks," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1212–1239, 2022.

[50] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[51] J. Abou Jaoude and R. G. Saade, "Blockchain applications–usage in different domains," *IEEE Access*, vol. 7, pp. 45 360–45 381, 2019.

[52] M. Hussain et al., "Blockchain-based iot devices in supply chain management: A systematic literature review," *Sustainability*, vol. 13, no. 24, p. 13 646, 2021.

[53] V. Maurya et al., "Blockchain-driven security for iot networks: State-of-the-art, challenges and future directions," *Peer-to-Peer Networking and Applications*, vol. 18, no. 1, p. 53, 2025.

[54] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, IEEE, 2017, pp. 2567–2572.

[55] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial innovation*, vol. 2, no. 1, p. 24, 2016.

[56] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–35, 2023.

[57] A. Baliga, "Understanding blockchain consensus models," *Persistent*, vol. 4, no. 1, p. 14, 2017.

[58] C. Zhang, C. Wu, and X. Wang, "Overview of blockchain consensus mechanism," in *Proceedings of the 2020 2nd International Conference on Big Data Engineering*, 2020, pp. 7–12.

[59] Y. Liu, Z. Fang, M. H. Cheung, W. Cai, and J. Huang, "Economics of blockchain storage," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020, pp. 1–6.

[60] N. C. Yiu, "An overview of forks and coordination in blockchain development," *arXiv preprint arXiv:2102.10006*, 2021.

[61] P. Arul and S. Renuka, "Blockchain technology using consensus mechanism for iot-based e-healthcare system," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1055, 2021, p. 012 106.

[62] H. Xiong, M. Chen, C. Wu, Y. Zhao, and W. Yi, "Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms," *Future Internet*, vol. 14, no. 2, p. 47, 2022.

[63] G.-T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain.," *Journal of Information processing systems*, vol. 14, no. 1, 2018.

[64] X. Fu, H. Wang, and P. Shi, "A survey of blockchain consensus algorithms: Mechanism, design and applications," *Science China Information Sciences*, vol. 64, no. 2, p. 121 101, 2021.

[65] S. Zhou, K. Li, L. Xiao, J. Cai, W. Liang, and A. Castiglione, "A systematic review of consensus mechanisms in blockchain," *Mathematics*, vol. 11, no. 10, p. 2248, 2023.

[66] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, no. 1, 2012.

[67] S. M. S. Saad and R. Z. R. M. Radzi, "Comparative review of the blockchain consensus algorithm between proof of stake (pos) and delegated proof of stake (dpos)," *International Journal of Innovative Computing*, vol. 10, no. 2, 2020.

[68] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work," *Cryptology ePrint Archive*, 2017.

[69] B. Yu, J. Liu, S. Nepal, J. Yu, and P. Rimba, "Proof-of-qos: Qos based blockchain consensus protocol," *Computers & Security*, vol. 87, p. 101 580, 2019.

[70] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[71] S. Joshi, "Feasibility of proof of authority as a consensus protocol model," *arXiv preprint arXiv:2109.02480*, 2021.

[72] D. Puthal and S. P. Mohanty, "Proof of authentication: Iot-friendly blockchains," *IEEE Potentials*, vol. 38, no. 1, pp. 26–29, 2018.

[73] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, Springer, 2017, pp. 282–297.

[74] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Annual Cryptology Conference*, Springer, 2015, pp. 585–605.

[75] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi, "Proofs of space: When space is of the essence," in *International conference on security and cryptography for networks*, Springer, 2014, pp. 538–557.

[76] X. Fu, H. Wang, P. Shi, and H. Mi, "Popf: A consensus algorithm for jcledger," in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, IEEE, 2018, pp. 204–209.

[77] M. Ghosh, M. Richardson, B. Ford, and R. Jansen, "A torpath to torcoin: Proof-of-bandwidth altcoins for compensating relays," 2014.

[78] R. Krishnamurthi and T. Shree, "A brief analysis of blockchain algorithms and its challenges," *Architectures and frameworks for developing and applying blockchain technology*, pp. 69–85, 2019.

[79] A. Yakovenko, *Solana: A new architecture for a high performance blockchain v0. 8.13*, 2018.

[80] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *proceedings of the 1st Workshop on System Software for Trusted Execution*, 2016, pp. 1–6.

[81] Y. Ren, H. Guan, Q. Zhao, and Z. Yi, "Blockchain-based proof of retrievability scheme," *Security and Communication Networks*, vol. 2022, no. 1, p. 3 186 112, 2022.

[82] K. Li, H. Li, H. Hou, K. Li, and Y. Chen, "Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2017, pp. 466–473.

[83] Y. Sun, B. Yan, Y. Yao, and J. Yu, "Dt-dpos: A delegated proof of stake consensus algorithm with dynamic trust," *Procedia Computer Science*, vol. 187, pp. 371–376, 2021.

[84] D. Tosh, S. Shetty, P. Foytik, C. Kamhoua, and L. Njilla, "Cloudpos: A proof-of-stake consensus design for blockchain integrated cloud," in *2018 IEEE 11Th*

international conference on cloud computing (CLOUD), IEEE, 2018, pp. 302–309.

[85] Y. Shifferaw and S. Lemma, "Limitations of proof of stake algorithm in blockchain: A review," *Zede Journal*, vol. 39, no. 1, pp. 81–95, 2021.

[86] T. F. Anan, A. I. M. Mahi, and T. K. Arnob, "Ppos: An optimized consensus protocol for iot devices," Islamic University of Technology, Bachelor Thesis, Department of Computer Science and Engineering (CSE), Islamic University of Technology, 2023.

[87] D. Wang, C. Jin, H. Li, and M. Perkowski, "Proof of activity consensus algorithm based on credit reward mechanism," in *International Conference on Web Information Systems and Applications*, Springer, 2020, pp. 618–628.

[88] Z. Boreiri and A. N. Azad, "A novel consensus protocol in blockchain network based on proof of activity protocol and game theory," in *2022 8th International Conference on Web Research (ICWR)*, IEEE, 2022, pp. 82–87.

[89] A. Nazir et al., "An optimized concurrent proof of authority consensus protocol," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2023, pp. 874–877.

[90] S. Alrubei, E. Ball, and J. Rigelsford, "Hdpoa: Honesty-based distributed proof of authority via scalable work consensus protocol for iot-blockchain applications," *Computer Networks*, vol. 217, p. 109 337, 2022.

[91] A. Pal and K. Kant, "Dc-poet: Proof-of-elapsed-time consensus with distributed coordination for blockchain networks," in *2021 IFIP Networking Conference (IFIP Networking)*, IEEE, 2021, pp. 1–9.

[92] A. A. Khan, S. Dhabi, J. Yang, W. Alhakami, S. Bourouis, and L. Yee, "B-lpoet: A middleware lightweight proof-of-elapsed time (poet) for efficient distributed transaction execution and security on blockchain using multithreading technology," *Computers and Electrical Engineering*, vol. 118, p. 109 343, 2024.

[93] F. Xiang, W. Huaimin, and S. Peichang, "Proof of previous transactions (popt): An efficient approach to consensus for jcledger," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2415–2424, 2019.

[94] A. Andrey and C. Petr, “Review of existing consensus algorithms blockchain,” in *2019 International Conference" Quality Management, Transport and Information Security, Information Technologies"(IT&QM&IS)*, IEEE, 2019, pp. 124–127.

[95] B. Xiao, C. Jin, Z. Li, B. Zhu, X. Li, and D. Wang, “Proof of importance: A consensus algorithm for importance based on dynamic authorization,” in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2021, pp. 510–513.

[96] H. Y. Yuen, F. Wu, W. Cai, H. C. Chan, Q. Yan, and V. C. Leung, “Proof-of-play: A novel consensus model for blockchain-based peer-to-peer gaming system,” in *Proceedings of the 2019 ACM international symposium on blockchain and secure critical infrastructure*, 2019, pp. 19–28.

[97] N. Stifter, A. Judmayer, and E. Weippl, “Revisiting practical byzantine fault tolerance through blockchain technologies,” in *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*, Springer, 2019, pp. 471–495.

[98] J. Mišić, V. B. Mišić, X. Chang, and H. Qushtom, “Adapting pbft for use with blockchain-enabled iot systems,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 33–48, 2020.

[99] L. Lao, X. Dai, B. Xiao, and S. Guo, “G-pbft: A location-based and scalable consensus protocol for iot-blockchain applications,” in *2020 IEEE international parallel and distributed processing symposium (IPDPS)*, IEEE, 2020, pp. 664–673.

[100] G. Xu et al., “Sg-pbft: A secure and highly efficient distributed blockchain pbft consensus algorithm for intelligent internet of vehicles,” *Journal of Parallel and Distributed Computing*, vol. 164, pp. 1–11, 2022.

[101] S. Gao, T. Yu, J. Zhu, and W. Cai, “T-pbft: An eigentrust-based practical byzantine fault tolerance consensus algorithm,” *China Communications*, vol. 16, no. 12, pp. 111–123, 2019.

[102] X. Hao, L. Yu, L. Zhiqiang, L. Zhen, and G. Dawu, "Dynamic practical byzantine fault tolerance," in *2018 IEEE conference on communications and network security (CNS)*, IEEE, 2018, pp. 1–8.

[103] L. Feng, H. Zhang, Y. Chen, and L. Lou, "Scalable dynamic multi-agent practical byzantine fault-tolerant consensus in permissioned blockchain," *Applied Sciences*, vol. 8, no. 10, p. 1919, 2018.

[104] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 172–181, 2019.

[105] K. Lei, M. Du, J. Huang, and T. Jin, "Groupchain: Towards a scalable public blockchain in fog computing of iot services computing," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 252–262, 2020.

[106] E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[107] D. Schwartz, N. Youngs, A. Britto, et al., "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, p. 151, 2014.

[108] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, vol. 34, pp. 9–11, 2016.

[109] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, no. 11, pp. 1–11, 2014.

[110] D. Puthal, S. P. Mohanty, P. Nanda, E. Kougianos, and G. Das, "Proof-of-authentication for scalable blockchain in resource-constrained distributed systems," in *2019 IEEE international conference on consumer electronics (ICCE)*, IEEE, 2019, pp. 1–5.

[111] F. Xiang, W. Huaimin, S. Peichang, F. Yingwei, and W. Yijie, "Jcledger: A blockchain based distributed ledger for jointcloud computing," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, IEEE, 2017, pp. 289–293.

[112] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The server allocation problem for session-based multiplayer cloud gaming," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1233–1245, 2017.

[113] L. Lamport, "The weak byzantine generals problem," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 668–676, 1983.

[114] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.

[115] L. Tseng and N. H. Vaidya, "Fault-tolerant consensus in directed graphs," in *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, 2015, pp. 451–460.

[116] N. Sharma, M. Shamkuwar, and I. Singh, "The history, present and future with iot," in *Internet of things and big data analytics for smart generation*, Springer, 2018, pp. 27–51.

[117] P. Suresh, J. V. Daniel, V. Parthasarathy, and R. Aswathy, "A state of the art review on the internet of things (iot) history, technology and fields of deployment," in *2014 International conference on science engineering and management research (ICSEMR)*, IEEE, 2014, pp. 1–8.

[118] S. J. Ramson, S. Vishnu, and M. Shanmugam, "Applications of internet of things (iot)–an overview," in *2020 5th international conference on devices, circuits and systems (ICDCS)*, IEEE, 2020, pp. 92–95.

[119] Y. Yu, Y. Li, J. Tian, and J. Liu, "Blockchain-based solutions to security and privacy issues in the internet of things," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 12–18, 2018. DOI: 10.1109/MWC.2017.1800116

[120] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and iot integration: A systematic survey," *Sensors*, vol. 18, no. 8, p. 2575, 2018.

[121] W. Viriyasitavat, L. Da Xu, Z. Bi, D. Hoonsopon, and N. Charoenruk, "Managing qos of internet-of-things services using blockchain," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1357–1368, 2019.

[122] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.

[123] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future generation computer systems*, vol. 88, pp. 173–190, 2018.

[124] H. F. Atlam, A. Alenezi, M. O. Alassafi, and G. Wills, "Blockchain with internet of things: Benefits, challenges, and future directions," *International Journal of Intelligent Systems and Applications*, vol. 10, no. 6, pp. 40–48, 2018.

[125] P. De Filippi, M. Mannan, and W. Reijers, "Blockchain as a confidence machine: The problem of trust & challenges of governance," *Technology in Society*, vol. 62, p. 101 284, 2020.

[126] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, IEEE, 2008, pp. 3–17.

[127] R. Huo et al., "A comprehensive survey on blockchain in industrial internet of things: Motivations, research progresses, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 88–122, 2022.

[128] A. N. Mufiidah and R. D. Destiani, "The benefits, challenges, and future of blockchain and the internet of things," *Blockchain Frontier Technology*, vol. 3, no. 1, pp. 13–25, 2023.

[129] A. Al Sadawi, M. S. Hassan, and M. Ndiaye, "A survey on the integration of blockchain with iot to enhance performance and eliminate challenges," *IEEE Access*, vol. 9, pp. 54 478–54 497, 2021.

[130] P. Ratta, A. Kaur, S. Sharma, M. Shabaz, and G. Dhiman, "Application of blockchain and internet of things in healthcare and medical sector: Applications, challenges, and future perspectives," *Journal of Food Quality*, vol. 2021, no. 1, p. 7 608 296, 2021.

[131] L. Ismail, H. Materwala, and S. Zeadally, "Lightweight blockchain for healthcare," *IEEE Access*, vol. 7, pp. 149 935–149 951, 2019.

[132] Y. Huang, J. Wu, and C. Long, "Drugledger: A practical blockchain system for drug traceability and regulation," in *2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, IEEE, 2018, pp. 1137–1144.

[133] G. Srivastava, A. D. Dwivedi, and R. Singh, "Automated remote patient monitoring: Data sharing and privacy using blockchain," *arXiv preprint arXiv:1811.03417*, 2018.

[134] R. Paul, P. Baidya, S. Sau, K. Maity, S. Maity, and S. B. Mandal, "Iot based secure smart city architecture using blockchain," in *2018 2nd international conference on data science and business analytics (ICDSBA)*, IEEE, 2018, pp. 215–220.

[135] A. El Bekkali, M. Essaaidi, and M. Boulmalf, "A blockchain-based architecture and framework for cybersecure smart cities," *IEEE Access*, vol. 11, pp. 76 359–76 370, 2023.

[136] M. Saad, M. B. Ahmad, M. Asif, M. K. Khan, T. Mahmood, and M. T. Mahmood, "Blockchain-enabled vanet for smart solid waste management," *IEEE Access*, vol. 11, pp. 5679–5700, 2023.

[137] S. A. Bhat, N.-F. Huang, I. B. Sofi, and M. Sultan, "Agriculture-food supply chain management based on blockchain and iot: A narrative on enterprise blockchain interoperability," *Agriculture*, vol. 12, no. 1, p. 40, 2021.

[138] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017.

[139] M. AlShamsi, M. Al-Emran, and K. Shaalan, "A systematic review on blockchain adoption," *Applied Sciences*, vol. 12, no. 9, p. 4245, 2022.

[140] M. Dehghani, R. W. Kennedy, A. Mashatan, A. Rese, and D. Karavidas, "High interest, low adoption. a mixed-method investigation into the factors influencing organisational adoption of blockchain technology," *Journal of Business Research*, vol. 149, pp. 393–411, 2022.

[141] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, p. 30, 2018.

[142] M. Alshaikhli, T. Elfouly, O. Elharrouss, A. Mohamed, and N. Ottakath, "Evolution of internet of things from blockchain to iota: A survey," *IEEE Access*, vol. 10, pp. 844–866, 2021.

[143] J. Buchmann, E. Dahmen, S. Ereth, A. Hülsing, and M. Rückert, "On the security of the winternitz one-time signature scheme," *International Journal of Applied Cryptography*, vol. 3, no. 1, pp. 84–96, 2013.

[144] S. Popov and Q. Lu, "Iota: Feeless and free," *IEEE Blockchain Technical Briefs*, vol. 6, p. 964, 2019.

[145] K. W. Prewett, G. L. Prescott, and K. Phillips, "Blockchain adoption is inevitable—barriers and risks remain," *Journal of Corporate accounting & finance*, vol. 31, no. 2, pp. 21–28, 2020.

[146] T. J. Sejnowski, *The deep learning revolution*. MIT press, 2018.

[147] Z. Wan, M. Cai, J. Yang, and X. Lin, "A novel blockchain as a service paradigm," in *International Conference on Blockchain*, Springer, 2018, pp. 267–273.

[148] A. Nordrum, "Govern by blockchain dubai wants one platform to rule them all, while illinois will try anything," *IEEE Spectrum*, vol. 54, no. 10, pp. 54–55, 2017.

[149] J. Song, P. Zhang, M. Alkubati, Y. Bao, and G. Yu, "Research advances on blockchain-as-a-service: Architectures, applications and challenges," *Digital Communications and Networks*, vol. 8, no. 4, pp. 466–475, 2022.

[150] Alibaba, *Blockchain as a service:what is baas?* Accessed on: 2 September, 2025. [Online]. Available: `https://www.alibabacloud.com/help/en/blockchain-as-a-service/latest/what-is-baas`

[151] IBM, *Blockchain for supply chain solutions*, Accessed on: 2 September, 2025. [Online]. Available: `https://www.ibm.com/solutions/blockchain-supply-chain`

[152] Microsoft, *Digitizing trust: Azure blockchain service simplifies blockchain development*, Accessed on: 2 September, 2025. [Online]. Available: https://azure.microsoft.com/en-us/blog/digitizing-trust-azure-blockchain-service-simplifies-blockchain-development/

[153] Oracle, *Oracle blockchain*, Accessed on: 2 September, 2025. [Online]. Available: https://www.oracle.com/apac/blockchain/

[154] Amazon, *Amazon managed blockchain*, Accessed on: 2 September, 2025. [Online]. Available: https://aws.amazon.com/managed-blockchain/

[155] W. Zheng, Z. Zheng, X. Chen, K. Dai, P. Li, and R. Chen, "Nutbaas: A blockchain-as-a-service platform," *IEEE Access*, vol. 7, pp. 134 422–134 433, 2019.

[156] M. S. Whittingham, "History, evolution, and future status of energy storage," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1518–1534, 2012.

[157] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[158] R. Soua and P. Minet, "A survey on energy efficient techniques in wireless sensor networks," in *2011 4th joint IFIP wireless and mobile networking conference (WMNC 2011)*, IEEE, 2011, pp. 1–9.

[159] S. Buzzi, I. Chih-Lin, T. E. Klein, H. V. Poor, C. Yang, and A. Zappone, "A survey of energy-efficient techniques for 5g networks and challenges ahead," *IEEE Journal on selected areas in communications*, vol. 34, no. 4, pp. 697–709, 2016.

[160] S. Chaudhary, N. Singh, A. Pathak, and A. Vatsa, "Energy efficient techniques for data aggregation and collection in wsn," *International Journal of Computer Science, Engineering and Applications (IJCSEA) vol*, vol. 2, 2012.

[161] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer networks*, vol. 67, pp. 104–122, 2014.

[162] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN computer science*, vol. 2, no. 3, p. 160, 2021.

[163] A. L. Fradkov, "Early history of machine learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385–1390, 2020.

[164] G. Allen, "Understanding ai technology," 2020.

[165] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: Current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020.

[166] H. F. Atlam, M. A. Azad, A. G. Alzahrani, and G. Wills, "A review of blockchain in internet of things and ai," *Big Data and Cognitive Computing*, vol. 4, no. 4, p. 28, 2020.

[167] A. Shrivastava, K. M. Krishna, M. L. Rinawa, M. Soni, G. Ramkumar, and S. Jaiswal, "Inclusion of iot, ml, and blockchain technologies in next generation industry 4.0 environment," *Materials Today: Proceedings*, vol. 80, pp. 3471–3475, 2023.

[168] C. Merlano, "Enhancing cyber security through artificial intelligence and machine learning: A literature review," *Journal of Cybersecurity*, vol. 6, p. 89, 2024.

[169] U. S. Musa, M. Chhabra, A. Ali, and M. Kaur, "Intrusion detection system using machine learning techniques: A review," in *2020 international conference on smart electronics and communication (ICOSEC)*, IEEE, 2020, pp. 149–155.

[170] L. Haripriya and M. A. Jabbar, "Role of machine learning in intrusion detection system," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, IEEE, 2018, pp. 925–929.

[171] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Procedia Computer Science*, vol. 171, pp. 1251–1260, 2020.

[172] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning

and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, e4150, 2021.

[173] S. Omar, A. Ngadi, and H. H. Jebur, "Machine learning techniques for anomaly detection: An overview," *International Journal of Computer Applications*, vol. 79, no. 2, 2013.

[174] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[175] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[176] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102 526, 2020.

[177] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124 579–124 607, 2020.

[178] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): A survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[179] A. A. Shuvro, M. S. Khan, M. Rahman, F. Hussain, M. Moniruzzaman, and M. S. Hossen, "Transformer based traffic flow forecasting in sdn-vanet," *IEEE Access*, vol. 11, pp. 41 816–41 826, 2023.

[180] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, "A survey on machine learning techniques for routing optimization in sdn," *IEEE Access*, vol. 9, pp. 104 582–104 611, 2021.

[181] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in sdn using machine learning approach," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, 2016, pp. 167–172.

[182] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, "Machine learning algorithms to detect ddos attacks in sdn," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 16, e5402, 2020.

[183] G. Logeswari, S. Bose, T. Anitha, et al., "An intrusion detection system for sdn using machine learning," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 867–880, 2023.

[184] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: State of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE network*, vol. 28, no. 6, pp. 18–26, 2014.

[185] S. Zehra et al., "Machine learning-based anomaly detection in nfv: A comprehensive survey," *Sensors*, vol. 23, no. 11, p. 5340, 2023.

[186] S. Shahzadi et al., "Machine learning empowered security management and quality of service provision in sdn-nfv environment," *Computers, Materials and Continua*, vol. 66, no. 3, pp. 2723–2749, 2020.

[187] M. Imran, U. Zaman, Imran, J. Imtiaz, M. Fayaz, and J. Gwak, "Comprehensive survey of iot, machine learning, and blockchain for health care applications: A topical assessment for pandemic preparedness, challenges, and solutions," *Electronics*, vol. 10, no. 20, p. 2501, 2021.

[188] Z. Shahbazi and Y.-C. Byun, "A procedure for tracing supply chains for perishable food based on blockchain, machine learning and fuzzy logic," *Electronics*, vol. 10, no. 1, p. 41, 2020.

[189] K. Abbas, M. Afaq, T. Ahmed Khan, and W.-C. Song, "A blockchain and machine learning-based drug supply chain management and recommendation system for smart pharmaceutical industry," *Electronics*, vol. 9, no. 5, p. 852, 2020.

[190] H. Hu, J. Xu, M. Liu, and M. K. Lim, "Vaccine supply chain management: An intelligent system utilizing blockchain, iot and machine learning," *Journal of business research*, vol. 156, p. 113 480, 2023.

[191] A. A. Khan et al., "Healthcare ledger management: A blockchain and machine learning-enabled novel and secure architecture for medical industry," *Hum. Cent. Comput. Inf. Sci*, vol. 12, p. 55, 2022.

[192] P. Bhattacharya, S. Tanwar, U. Bodkhe, S. Tyagi, and N. Kumar, "Bindaas: Blockchain-based deep-learning as-a-service in healthcare 4.0 applications," *IEEE transactions on network science and engineering*, vol. 8, no. 2, pp. 1242–1255, 2019.

[193] W. Rahman et al., "Automated detection of harmful insects in agriculture: A smart framework leveraging iot, machine learning, and blockchain," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 9, pp. 4787–4798, 2024.

[194] D. Shah, D. Patel, J. Adesara, P. Hingu, and M. Shah, "Integrating machine learning and blockchain to develop a system to veto the forgeries and provide efficient results in education sector," *Visual computing for industry, biomedicine, and art*, vol. 4, no. 1, p. 18, 2021.

[195] P. Cao, G. Zhu, Q. Zhang, F. Wang, Y. Liu, and R. Mo, "Blockchain-enabled hmm model for sports performance prediction," *IEEE Access*, vol. 9, pp. 40 255–40 262, 2021.

[196] L. N. CheSuh, R. A. Fernandez-Diaz, J. M. Alija-Perez, C. Benavides-Cuellar, and H. Alaiz-Moreton, "Improve quality of service for the internet of things using blockchain & machine learning algorithms," *Internet of Things*, vol. 26, p. 101 123, 2024.

[197] J. Kim et al., "A machine learning approach to anomaly detection based on traffic monitoring for secure blockchain networking," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3619–3632, 2022.

[198] M. Monem 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.

[199] S. Kayikci and T. M. Khoshgoftaar, "Blockchain meets machine learning: A survey," *Journal of Big Data*, vol. 11, no. 1, p. 9, 2024.

[200] M. U. Hassan, M. H. Rehmani, and J. Chen, "Anomaly detection in blockchain networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 289–318, 2022.

[201] K. Venkatesan and S. B. Rahayu, "Blockchain security enhancement: An approach towards hybrid consensus algorithms and machine learning techniques," *Scientific Reports*, vol. 14, no. 1, p. 1149, 2024.

[202] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106 854, 2020.

[203] I. Buck, "Gpu computing with nvidia cuda," in *ACM SIGGRAPH 2007 courses*, 2007, 6–es.

[204] I. Buck et al., "Brook for gpus: Stream computing on graphics hardware," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 777–786, 2004.

[205] T. Kalaiselvi, P. Sriramakrishnan, and K. Somasundaram, "Survey of using gpu cuda programming model in medical image analysis," *Informatics in Medicine Unlocked*, vol. 9, pp. 133–144, 2017.

[206] S. Dridi, "Supervised learning-a systematic literature review," *preprint, Dec*, 2021.

[207] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, "Linear regression," in *An introduction to statistical learning: With applications in python*, Springer, 2023, pp. 69–134.

[208] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

[209] J. Ranstam and J. A. Cook, "Lasso regression," *Journal of British Surgery*, vol. 105, no. 10, pp. 1348–1348, 2018.

[210] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, "Accelerating stochastic gradient descent for least squares regression," in *Conference On Learning Theory*, PMLR, 2018, pp. 545–604.

[211] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[212] 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.

[213] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[214] S. Zhou, S. Wang, Q. Wu, R. Azim, and W. Li, "Predicting potential mirna-disease associations by combining gradient boosting decision tree with logistic regression," *Computational biology and chemistry*, vol. 85, p. 107 200, 2020.

[215] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining*, IEEE, 2010, pp. 995–1000.

[216] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 43–50.

[217] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[218] Z. Ghahramani, "Unsupervised learning," in *Summer school on machine learning*, Springer, 2003, pp. 72–112.

[219] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[220] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[221] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Density-based spatial clustering of applications with noise," in *Int. Conf. knowledge discovery and data mining*, vol. 240, 1996.

[222] C. Rasmussen, "The infinite gaussian mixture model," *Advances in neural information processing systems*, vol. 12, 1999.

[223] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[224] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione, "Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets," *Nature communications*, vol. 10, no. 1, p. 5415, 2019.

[225] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[226] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[227] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[228] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*, PMLR, 2021, pp. 10 096–10 106.

[229] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[230] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024.

[231] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[232] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[233] K. Cho et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[234] S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example," in *2020 Interna-*

tional workshop on electronic communication and artificial intelligence (IWECAI), IEEE, 2020, pp. 98–101.

[235] K. Yousaf and T. Nawaz, "A deep learning-based approach for inappropriate content detection and classification of youtube videos," *IEEE Access*, vol. 10, pp. 16 283–16 298, 2022.

[236] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[237] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[238] P. Jain et al., "Dynamic space-time scheduling for gpu inference," *arXiv preprint arXiv:1901.00041*, 2018.

[239] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2880–2887.

[240] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM symposium on principles of distributed computing*, 2017, pp. 315–324.

[241] Hazelcast, *What is machine learning inference?* Accessed on: 13 September, 2025. [Online]. Available: `https://hazelcast.com/foundations/ai-machine-learning/machine-learning-inference/`

[242] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[243] G. Callebaut, G. Leenders, J. Van Mulders, G. Ottoy, L. De Strycker, and L. Van der Perre, "The art of designing remote iot devices—technologies and strategies for a long battery life," *Sensors*, vol. 21, no. 3, p. 913, 2021.

[244] B. Jolly, "Iot device battery life: Go slow for fast insights into challenging conditions," in *2021 IEEE international midwest symposium on circuits and systems (MWSCAS)*, IEEE, 2021, pp. 680–683.

[245] R. Han, Z. Yan, X. Liang, and L. T. Yang, "How can incentive mechanisms and blockchain benefit with each other? a survey," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38, 2022.

[246] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 640–651.

[247] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia: case studies on organization and retrieval*, Springer, 2008, pp. 21–49.

[248] M. A. El Mrabet, K. El Makkaoui, and A. Faize, "Supervised machine learning: A survey," in *2021 4th International conference on advanced communication technologies and networking (CommNet)*, IEEE, 2021, pp. 1–10.

[249] H. E. Cagnini, S. C. D. Dôres, A. A. Freitas, and R. C. Barros, "A survey of evolutionary algorithms for supervised ensemble learning," *The Knowledge Engineering Review*, vol. 38, e1, 2023.

[250] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data preprocessing for supervised leaning," *International journal of computer science*, vol. 1, no. 2, pp. 111–117, 2006.

[251] E. Guerra, F. Wilhelmi, M. Miozzo, and P. Dini, "The cost of training machine learning models over distributed data sources," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 1111–1126, 2023.

[252] D. C. Nguyen et al., "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[253] C. Ma et al., "When federated learning meets blockchain: A new distributed learning paradigm," *IEEE Computational Intelligence Magazine*, vol. 17, no. 3, pp. 26–33, 2022.

[254] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: A friendly introduction," *Journal of Big Data*, vol. 9, no. 1, p. 102, 2022.

[255] S. J. Pan, “Transfer learning,” *Learning*, vol. 21, pp. 1–2, 2020.

[256] S. Niu, Y. Liu, J. Wang, and H. Song, “A decade survey of transfer learning (2010–2020),” *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 151–166, 2021.

[257] H. C. Ellis, “The transfer of learning.,” 1965.

[258] P. Nation, “The role of the first language in foreign language learning,” 2003.

[259] J. C. Chang, S. Amershi, and E. Kamar, “Revolt: Collaborative crowdsourcing for labeling machine learning datasets,” in *Proceedings of the 2017 CHI conference on human factors in computing systems*, 2017, pp. 2334–2346.

[260] J. Gao, B. Adjei-Arthur, E. B. Sifah, H. Xia, and Q. Xia, “Supply chain equilibrium on a game theory-incentivized blockchain network,” *Journal of Industrial Information Integration*, vol. 26, p. 100 288, 2022.

[261] C. Faria and M. Correia, “Blocksim: Blockchain simulator,” in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 439–446.

[262] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, “Simblock: A blockchain network simulator,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 325–329.

[263] H. Yajam, E. Ebadi, and M. A. Akhaee, “Jabs: A blockchain simulator for researching consensus algorithms,” *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 1, pp. 3–13, 2023.