INFORMATICA, 2024, Vol. 35, No. 2, 227–253

© 2024 *Vilnius University*
DOI: https://doi.org/10.15388/24-INFOR553

Advancing Research Reproducibility in Machine Learning through Blockchain Technology

Ernestas FILATOVAS*, Linas STRIPINIS, Francisco ORTS, Remigijus PAULAVIČIUS

[1] *Institute of Data Science and Digital Technologies, Vilnius University, Akademijos str. 4, LT-08412 Vilnius, Lithuania*
e-mail: ernestas.filatovas@mif.vu.lt, linas.stripinis@mif.vu.lt, francisco.orts@mif.vu.lt, remigijus.paulavicius@mif.vu.lt

Received: February 2024; accepted: April 2024

Abstract. Like other disciplines, machine learning is currently facing a reproducibility crisis that hinders the advancement of scientific research. Researchers face difficulties reproducing key results due to the lack of critical details, including the disconnection between publications and associated models, data, parameter settings, and experimental results. To promote transparency and trust in research, solutions that improve the accessibility of models and data, facilitate experiment tracking, and allow audit of experimental results are needed. Blockchain technology, characterized by its decentralization, data immutability, cryptographic hash functions, consensus algorithms, robust security measures, access control mechanisms, and innovative smart contracts, offers a compelling pathway for the development of such solutions. To address the reproducibility challenges in machine learning, we present a novel concept of a blockchain-based platform that operates on a peer-to-peer network. This network comprises organizations and researchers actively engaged in machine learning research, seamlessly integrating various machine learning research and development frameworks. To validate the viability of our proposed concept, we implemented a blockchain network using the Hyperledger Fabric infrastructure and conducted experimental simulations in several scenarios to thoroughly evaluate its effectiveness. By fostering transparency and facilitating collaboration, our proposed platform has the potential to significantly improve reproducible research in machine learning and can be adapted to other domains within artificial intelligence.

Key words: machine learning, reproducibility, reproducible research, blockchain, distributed ledger technology, interoperability, blockchain-based platform, hyperledger fabric.

1. Introduction

In recent years, the field of machine learning (ML) has seen remarkable advances, with algorithms achieving unprecedented levels of performance in various tasks such as image analysis (Gudžius *et al.*, 2021; Gudzius *et al.*, 2022), voice (Mehrish *et al.*, 2023), and face recognition (Liu *et al.*, 2023), planning (Usuga Cadavid *et al.*, 2020), scheduling (Bertolini *et al.*, 2021), etc. The advent of ML, particularly deep learning (DL), has

*Corresponding author.

emerged as the cornerstone of artificial intelligence (AI) due to advances in computational capabilities (Zhang and Lu, 2021). However, the widespread dependence on vast datasets and intricate models presents a formidable challenge to reproducibility in ML research. Regrettably, as ML and other AI domains continue to gain prominence, researchers face a reproducibility crisis (RC) (Hutson, 2018). This crisis is characterized by the daunting challenge of replicating critical findings, worsened by the gap between published results and the underlying models, data, parameter configurations and other essential experimental components, frequently missing crucial details.

To address issues related to RC, both the research community and industry have actively explored a wide variety of solutions. Traditionally, researchers have often resorted to depositing their code and data into repositories such as GitLab or GitHub. However, this practice often proves insufficient, as it lacks crucial information, including runtime environments, contextual details, and system information (Rowhani-Farid and Barnett, 2018). Furthermore, reliance on data stored in repository databases for journals may be unreliable due to broken links (Pimentel *et al.*, 2019). In response to these limitations, recent advances in research technology have introduced open-access collaborative cloud-based platforms such as Code Ocean (https://codeocean.com), Whole Tale (https://wholetale.org), and Binder (https://mybinder.org). These platforms enable comprehensive capture of research environments, thus facilitating reuse, sharing, and reproducibility of research processes. Furthermore, online platforms such as OpenML (Vanschoren *et al.*, 2014) and ModelDB (Vartak *et al.*, 2016) have gained popularity within the ML community by providing storage and sharing capabilities for datasets and experimental results, thus fostering collaboration in open science. Furthermore, a myriad of tools and frameworks specifically designed to facilitate open science collaboration in ML have emerged, such as MLflow (https://mlflow.org) and Neptune.AI (https://neptune.ai), have emerged. Although the functionalities of these tools vary, most of them allow researchers and ML practitioners to execute, monitor, compare, and visualize their experimental results, encompassing the entire research process.

However, recent studies indicate that, while some of the reproducibility aspects required are adequately addressed by existing tools and platforms (Mora-Cantallops *et al.*, 2021), further development is necessary (Gundersen *et al.*, 2022). In addition, these tools and platforms often exhibit centralization, which can lead to issues such as limited operational transparency, traceability, and auditability of community-driven experiments. Additionally, the centralized nature of these systems poses risks, such as single-point failure, which undermines their overall reliability (Cao *et al.*, 2022).

Decentralized alternatives offer a fresh perspective on the management of RC issues, with the aim of avoiding the traditional pitfalls associated with centralized systems. These alternatives take advantage of advancements such as transparency, traceability, tokenization, consensus mechanisms, incentivization, codification of trust, and decentralized infrastructure (Juodis *et al.*, 2024; Marcozzi *et al.*, 2024). Using blockchain technology, pioneered by Nakamoto (Nakamoto, 2008) and built on prior research (Bayer *et al.*, 1993; Haber and Stornetta, 1991; Lamport *et al.*, 1982), effective decentralized solutions can be implemented (Knez *et al.*, 2022; Matulevičius *et al.*, 2022; Sakalauskas *et al.*, 2023).

For more detailed information on blockchain technology aspects, we refer the reader to (Filatovas *et al.*, 2022; Paulavičius *et al.*, 2019, 2021). The research community recognizes the vast potential of blockchain technology in addressing RC challenges and related issues across various research fields. Consequently, there is a growing development of blockchain-enhanced research workflow solutions aimed at improving provenance, reliability, and collaboration (Bag *et al.*, 2022; Coelho *et al.*, 2020; Hoopes *et al.*, 2022; Meng and Sun, 2021).

Several notable business-oriented projects are currently under development, aiming to combine blockchain technologies and AI, with a primary focus on ML, to enhance the research process, improve reproducibility, and foster business collaboration. SingularityNet (https://singularitynet.io) seeks to establish a decentralized marketplace for AI algorithms and federated learning (FL). Meanwhile, PlatON (https://www.platon.network) is dedicated to building a decentralized and collaborative AI network and global brain to promote the democratization of AI for safe general artificial intelligence. Additionally, FETCH.AI (https://fetch.ai) is developing a decentralized collaborative ML platform customized for various business applications. These solutions, geared towards business-oriented goals, are currently in the active development stage. Furthermore, the research community is increasingly intrigued by the intersection of ML and blockchain technologies as a means of addressing reproducibility issues. A review and comparative analysis of state-of-the-art works in this direction are provided in Secrion 2. It reveals that current solutions are limited, and only specific aspects of ML reproducibility are considered.

Drawing from an analysis of existing proposals and recognizing the advantages offered by blockchain technology in improving reproducibility alongside the capabilities of current ML research tools, this paper introduces a novel concept: a community-driven blockchain-based platform aimed at improving reproducibility in ML research. The platform operates on a decentralized blockchain network, fostering collaboration among organizations and researchers actively involved in ML. It seamlessly integrates various existing ML research frameworks and tools, leveraging blockchain features such as decentralization, immutability, a community-driven consensus mechanism, and interoperability. Furthermore, through the automation provided by smart contracts, the platform enhances the auditability of experimental results, promotes transparency, and fosters trust within the ML research community. In addition, it effectively stores experiment artifacts in a decentralized and immutable manner, further enhancing the reliability and reproducibility of ML research outcomes.

To sum up, the main contributions of this work are:

- It reviews the current state-of-the-art blockchain-based proposals for enhancing reproducibility in machine learning research.
- It introduces a novel concept of a community-driven blockchain-based decentralized platform designed for reproducible and auditable research in various machine learning domains.
- It proposes a metadata schema designed to extract experiment artifacts, securely storing them on a blockchain and enabling retrieval on demand, ensuring the complete reproducibility of experiments.

- It experimentally evaluates the platform's blockchain performance across diverse scenarios using Hyperledger Fabric infrastructure, confirming its suitability for real-world development.

The remainder of the paper is organized as follows. A review of the literature on existing blockchain-based proposals to enhance reproducibility in ML research is presented in Section 2. The concept of the blockchain-based platform designed to facilitate collaborative and reproducible research in ML is introduced in Section 3. Section 4 presents the concepts of the Hyperledger Fabric framework and describes the setup of the experimental environment. Subsequently, Section 5 evaluates the platform's blockchain performance. Finally, Section 6 discusses the results obtained, concludes the work, and offers perspectives on future platform development.

2. Related Work

This section reviews and compares the current state-of-the-art literature on blockchain-based proposals aimed at enhancing reproducibility in ML research. We used backward and forward snowballing (Wohlin, 2014) as the primary approach to identify relevant literature, with searches carried out in September 2023. Additionally, we provide concise descriptions of the publications that were obtained, focusing on their key contributions.

Schelter *et al.* (2017) introduces a system architecture and presents a database schema for storing ML artifact metadata and experimentation data provenance. The authors discuss metadata extraction functionality for parameterized pipelines in SparkML and Scikit-learn to meet various production use cases at Amazon. Harris and Waggoner (2019) proposes a framework for training and sharing models and collecting data on a blockchain. The authors also discuss several incentive mechanisms to encourage contributors to submit data that improve the model's accuracy. Lu *et al.* (2019) introduces a data sharing architecture for privacy-preserving federated learning (FL). The paper proposes the Proof-of-Training-Quality consensus protocol, which combines the data model with the consensus process to better utilize computing resources. Sarpatwar *et al.* (2019) presents a vision to build a generic blockchain library that enables trust in distributed AI applications and processing. The authors discuss requirements for enabling trusted AI via blockchain and define key blockchain constructs and a provenance model to track training model provenance. Weng *et al.* (2019) presents a decentralized framework for training collaborative model ML and sharing local gradients in FL. The paper also introduces an incentive mechanism for participants and adapts a consensus protocol from Algorand to ensure consensus finality. Kannan *et al.* (2020) presents a decentralized trusted platform for collaborative AI. The authors propose a model for cross-organization data and resource sharing with various policy scenarios. Lüthi *et al.* (2020) proposes a graph-based provenance tracking model for managing AI assets and relationships in FL. They designed a smart contract for the Ethereum blockchain to implement the provenance model, adapted for a medical use case. Li *et al.* (2021) designs a decentralized FL framework and introduces the committee consensus mechanism to validate local gradients before appending them to the chain.

Mothukuri *et al.* (2021) proposes a decentralized framework for permissioned FL, with the aim of developing a secure application integrated to seal and sign-off asynchronous and synchronous collaborative tasks. Bathen and Jadav (2022) introduces a framework that combines AutoML techniques with blockchain to fully decentralize the design and training process, with the aim of introducing trust in AutoML pipelines without trusting the nodes performing various design, training, and testing steps. Coelho *et al.* (2022) proposes a blockchain platform architecture for collaborative research and reproducibility of experiment results, evaluating it using a scenario about genomic sequencing of the SARS-CoV-2 coronavirus. Khoi Tran *et al.* (2022) designs a decentralized platform based on blockchain and smart contracts for disseminating, storing, and updating ML provenance. They propose a novel architectural approach called Artefact-as-a-State-Machine to manage ML provenance on a blockchain. Lo *et al.* (2022) presents a blockchain-based trustworthy FL architecture, developing a data sampler algorithm to enhance fairness in training data and designing a smart contract-based data-model provenance registry. Stodt *et al.* (2022) aims to ensure the identity of ML models when solving security attacks, insufficient documentation, and traceability. The authors propose that the ML birth certificate and ML family tree be secured by blockchain technology. Ullah *et al.* (2023) proposes a blockchain-based FL approach for permissioned networks, integrating the Proof of Authority consensus protocol to improve scalability.

A summary of the main features of the described studies is provided in Table 1. Here, we highlight the analysed reproducibility aspects, application areas in ML, considered blockchain systems for implementation, integration or development of specific consensus protocols, foreseen integration with other ML tools/platforms, and implementation levels. Most solutions focus on specific reproducibility aspects, focusing primarily on the "model", "data", and "provenance", with less attention paid to the "parameters" and "results". Notably, environmental reproducibility remains largely overlooked across proposals. The first related publications emerged in 2017, most of them at the "idea" implementation level. However, recent proposals have progressed to higher levels, including "concept" or even "prototype". Public and private blockchain systems are considered for implementation, with solutions at the level of "idea" often lacking platform specification. Significant focus is placed on federated learning solutions, utilizing blockchain to improve fairness and protect data privacy. In this work, we emphasize studies that propose solutions focused on reproducibility through blockchain technology, while directing the reader to Beltrán *et al.* (2023) for a broader survey on decentralized FL. Permissioned systems, mostly Hyperledger Fabric, are frequently chosen for their suitability in FL processes that involve small groups of authorized actors. Efforts have been made to develop platforms for reproducible research in ML across various application areas, with both public Ethereum and private Hyperledger Fabric systems considered. Only a few solutions propose or integrate specific consensus protocols to improve the validation of developed models, data, or obtained results, indicating a need for further investigation in this area. Integration with other tools or platforms for model, data management, and sharing of experiment results is not considered extensively. Only Schelter *et al.* (2017) considers limited integration with SparkML and Scikit-learn for an Amazon use-case. Integrating with widely used ML tools

Table 1
Main features of the proposed state-of-the-art blockchain-based solutions to enhance reproducibility in ML research.

Source	Reproducibility aspect	Application area in ML	Blockchain platform	Specific consensus	Integration	Implementation level
Schelter *et al.* (2017)	Model, data, parameters, results	General	N/S*	✗	✓	Concept
Harris and Waggoner (2019)	Model, data, provenance	General	N/S*	✗	✗	Idea
Lu *et al.* (2019)	Model, data, parameters	Federated Learning	N/S	✓	✗	Idea
Sarpatwar *et al.* (2019)	Model, data, parameters, provenance	Federated Learning	N/S	✗	✗	Idea
Weng *et al.* (2019)	Model, parameters	Federated Learning	Corda	✓	✗	Concept
Kannan *et al.* (2020)	Model, data, provenance, results	General	Hyperledger Fabric	✗	✗	Concept
Lüthi *et al.* (2020)	Data, provenance	General	Ethereum	✗	✗	Concept
Li *et al.* (2021)	Model, parameters, provenance, results	Federated Learning	FISCO-BCOS	✓	✗	Concept
Mothukuri *et al.* (2021)	Model, parameters	Federated Learning	Hyperledger Fabric	✗	✗	Concept
Bathen and Jadav (2022)	Model, data, parameters, results	General	Hyperledger Fabric	✗	✗	Concept
Coelho *et al.* (2022)	Provenance, results	Federated Learning	Hyperledger Fabric	✗	✗	Prototype
Khoi Tran *et al.* (2022)	Model, data, parameters, provenance	General	Ethereum	✗	✗	Concept
Lo *et al.* (2022)	Model, data, provenance, results	Federated Learning	Parity	✗	✗	Concept
Stodt *et al.* (2022)	Model, provenance	General	N/S	✗	✗	Idea
Ullah *et al.* (2023)	Model, parameters	Federated Learning	N/S	✓	✗	Idea
This study	Model, data, provenance, parameters, results	General	Hyperledger Fabric	✓	✓	Concept

*N/S – Not Specified.

could significantly enhance platform efficiency, attract more participants, and support the sharing of experiment results and reproducibility.

The review conducted underscores a recent surge in attention from the research community towards developing and applying blockchain-based solutions to enhance reproducibility and data provenance in ML research. However, existing solutions are notably limited in scope, often focusing solely on specific aspects such as model and data sharing or provenance tracking. Some only address local gradients in FL, while others offer a general vision of how blockchain-based solutions could enhance trust and transparency in ML research. Furthermore, there is a glaring absence of consideration for interoperability with existing or emerging ML solutions. This highlights a pressing need for the research community to delve deeper into resolving research reproducibility challenges in the ML field and advancing the research cycle. To tackle the identified issues, our proposed solution

(see Section 3) involves the development of an ML community-driven blockchain-based decentralized platform that is scalable, interoperable, integrates with existing ML research tools, and adaptable across various AI research domains.

3. Proposed Concept of the Platform

This section is dedicated to presenting the concept of a community-driven blockchain-based platform designed to enhance reproducible research in ML. Firstly, we introduce a high-level architecture for developing such a platform, outlining and describing its main components and their interaction. Then, we dive into the metadata schema, a core component for extracting and storing experiment artifacts on the blockchain to ensure full reproducibility.

3.1. *High-Level Architecture*

In Fig. 1, we present a high-level architecture of the ML community-driven blockchain platform, focusing on key modules. It has been based on a realistic design principle, in which a consortium of stakeholders, including researchers from various organizations with expertise in ML, collaboratively conduct research and exchange experimental findings. The architecture consists of two primary modules: the AI Blockchain Network and AI Research, seamlessly integrated for enhanced functionality. The AI blockchain Network module serves as the infrastructure through which ML researchers can audit, validate experiment results, and store essential experimental artifacts (metadata) in a decentralized manner. All the metadata collected during the experiments are stored in blocks and distributed among peers. It can also be retrieved upon authorized request. The AI Research module of the architecture pertains to the integration of external ML tools and related data or ML model sources with the platform. In practice, this integration can be achieved via REST (representational state transfer) web services. To ensure high-level reproducibility of experiments carried out within the AI Research module, essential experimental metadata are extracted using the metadata schema via the transition script component of the

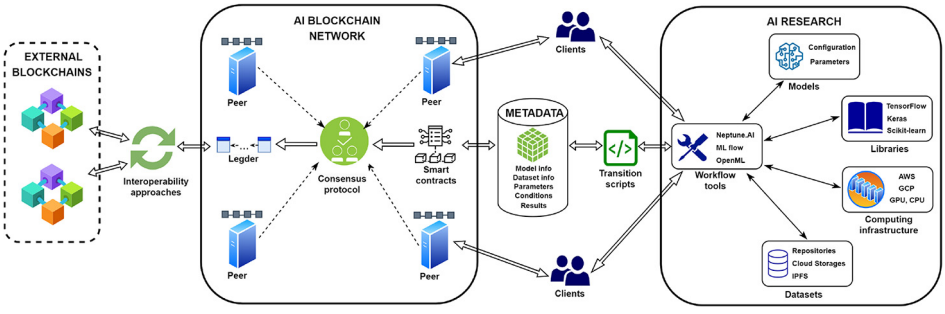


Fig. 1. High-level architecture of a blockchain-based platform designed to enhance research reproducibility in ML.

workflow tools (see Section 3.2), and subsequently stored on the blockchain in the form of a transaction. Integration with other blockchain platforms for AI research (for example, SingularityNet) is also planned through the external blockchain module, which could be facilitated using a variety of blockchain interoperability approaches. A detailed description and comparative analysis of these approaches are provided in Appendix A.

Additionally, we offer a comprehensive description of all main components and their interactions, placing particular emphasis on the platform's blockchain aspect.

- **Peer:** In the platform's blockchain network, a peer represents an entity within the platform's ecosystem, such as an ML research institution or a group of researchers or developers (referred to as an organization). Each peer operates a dedicated node on the network, storing a full local version of the blockchain ledger. This setup enables data processing, auditing, and transparent querying by various ML workflow tools. Organizations can utilize their own computing resources or cloud-based services to participate in the blockchain network. Peers also play a crucial role in validating and executing transactions.
- **Consensus protocol:** The consensus protocol facilitates agreement among all peers in the blockchain network to accept a transaction containing experimental artifacts in the form of metadata on the blockchain, ensuring reliability and trust among participants. In the context of the platform, a protocol like Proof-of-Authority (PoA), which relies on a group of approved validators, could be employed. Additionally, to enhance the quality and validity of data stored on the blockchain, a specifically designed community-driven consensus protocol, such as Proof-of-Validity (PoV), could be utilized. The PoV protocol would ensure the validity of metadata related to experimental results stored on the blockchain, relying on proofs regarding data quality, characteristics of learning models and parameters, computational resources utilized, and achieved results. The consensus process for the PoA or PoV protocols is executed by a selected committee comprising the chosen researchers through dedicated nodes, which form a subset of all participants.
- **Incentive mechanism:** The consensus protocol takes into account the user's reputation and includes an incentive mechanism. This integrated mechanism serves as a motivating factor for participants to participate actively and honestly in the validation of the results of the collaborative experiment. The primary objective of the incentive mechanism is to generate and distribute value, ensuring that participants receive rewards or penalties based on their contributions to the validation of experimental results.
- **Ledger:** In a distributed and decentralized blockchain network, blocks containing transactions of experimental metadata are recorded using an immutable and traceable mechanism, ensuring privacy, security, and transparency. The blockchain ledger serves as a decentralized database that records all transactions across the network, guaranteeing transparency and security by cryptographically linking each block to previous ones, forming an immutable chain. Each transaction is considered completely immutable and verified using the consensus protocol. In the context of the platform, transactions store the results of experiment executions and all experiment artifacts in the form of a metadata schema (see Fig. 2).
- **Smart contract:** Smart contracts automate processes and improve participant trust by managing user relationships, overseeing data access, and verifying data validity before

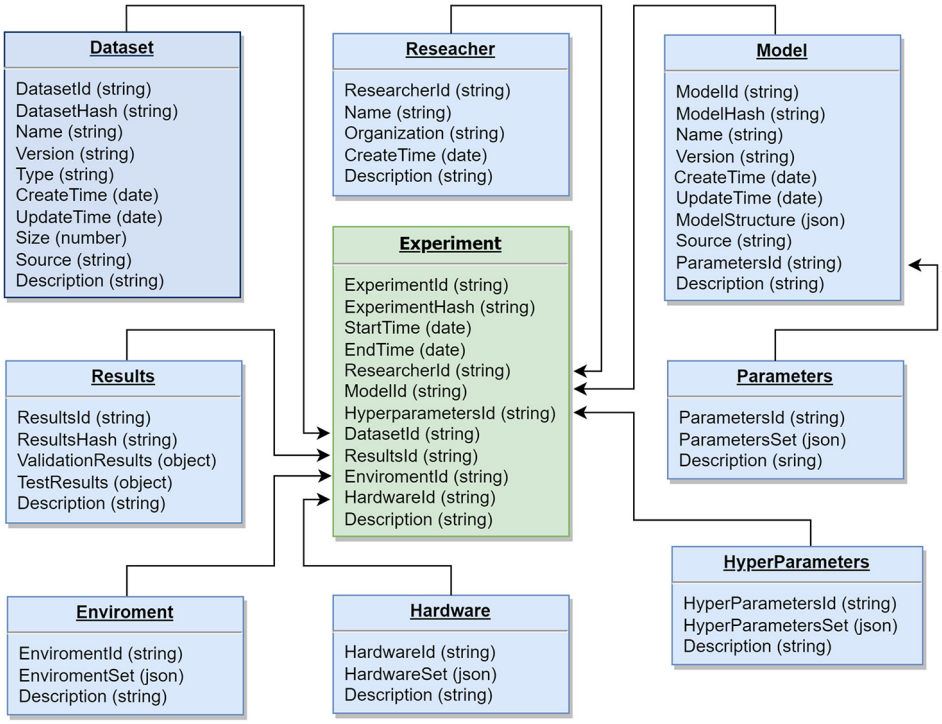


Fig. 2. Metadata schema for storing experiments artifacts in blockchain for ensuring full reproducibility.

storing them on the blockchain. Essentially, a smart contract is a program that automatically enforces consistent data management logic within the blockchain network. Additionally, smart contracts are transparent to all users based on the blockchain's permission settings, thereby enhancing trust and fairness. Once deployed, a smart contract is immutably recorded on the blockchain, with every peer in the network maintaining a copy in its local ledger and executing the smart contract based on transaction input. Smart contracts operate on the basis of data management functions that can be invoked by submitting a transaction to them. When a transaction is entered into the smart contract to fulfill certain conditions, the contract's implementation is executed immediately, fostering trust and transparency among participants in the blockchain network. In the context of our platform, smart contract functionality encompasses metadata management functions (such as experiment record, update, query, etc.), which receive requests from workflow tools upon the completion of a specific experiment run or when requesting experimental metadata from previously performed experiments.

- **Transition scripts:** These scripts serve as an intermediate transformation component to aggregate all essential experimental investigation artifacts generated by a workflow tool, including other relevant information about models, datasets, and computing infrastructure, into metadata form. Transition scripts are also responsible for extracting all the required information from the metadata saved on the blockchain as a transaction to reproduce a specific experiment within a workflow tool. As different workflow tools are

foreseen to be used on the platform, transition scripts for each tool must be developed to aggregate and retrieve all necessary experimental data, ensuring full reproducibility.

- **Clients:** Clients refer to registered and authenticated users, primarily researchers, who interact with peers within their respective organizations through a client application. The platform users extend beyond researchers, including developers and companies wanting to enhance collaboration and exchange research knowledge. The platform operates in such a way that clients' applications submit transaction proposals to store or request experiment metadata through peers and subsequently receive transaction updates once the process is completed. Clients conducting ML experiments utilize the ML research module based on externally developed and linked products.
- **Workflow tools:** Workflow tools are the primary interface between clients and the AI Research module. These tools facilitate the experimental investigation of ML by providing a platform with which clients can interact. Numerous tools, such as OpenML, MLflow, and Neptune.AI, have been developed to facilitate open science collaboration in ML. Although the functionality of these tools varies, most of them allow researchers and ML practitioners to execute, monitor, compare, and visualize their experiment results, covering the entire research process. Additionally, these tools often come with built-in integration with popular ML libraries or frameworks (such as TensorFlow, Keras, PyTorch, and Scikit-learn) and provide storage for datasets and ML models, or can integrate them from external sources, whether centralized or decentralized. Furthermore, they offer integration with external computing facilities such as Amazon Web Services (AWS) or Google Cloud Platform (GCP), allowing researchers to leverage the most suitable infrastructure for their experiments.

3.2. *Metadata Schema*

To efficiently store and retrieve reproducible experimental information on the decentralized blockchain network, we have developed a metadata schema (refer to Fig. 2) depicted in a database relation style. This schema is based on conducted experiments and covers various aspects of experimentation, including details about the researcher, trained model, parameters and hyperparameters settings, utilized dataset, environment and computing infrastructure, obtained experimental results, and other relevant investigation conditions. Such a structure allows platform participants to store experiment artifacts within transactions efficiently, as well as to verify conducted experiments and ensure full reproducibility. One of the main challenges in designing an experimentation metadata schema is finding the right balance between its generality and specificity, especially considering the anticipated integration of the platform with various ML workflow tools, each with differences in how they describe and export experimentation data. Therefore, we propose a middleground schema that addresses this concern. It is essential to note that the blockchain is not intended for storing large volumes of data, such as datasets, but rather for recording and regularizing sensitive information in the form of metadata. Therefore, we store metadata for the experimental artifacts, but not codes or datasets.

As the smart contract that implements data and transaction management was developed using JavaScript, the metadata schema also employs appropriate data types. Next,

we detail the components of the model, explaining their significance and relationships, and emphasizing the most crucial aspects while omitting descriptions of some fields that can be easily understood from their names.

- **Experiment:** The cornerstone of the metadata schema, this component aggregates data from all linked components. Each experiment is associated with a hash value (*ExperimentHash*), providing an immutable record of the experiment conducted and its related artifacts. Timestamps (*StartTime* and *EndTime*) are stored to denote the evaluation period of each experiment. The *Description* field, which is also serialized in other components of the metadata schema, serves to provide additional information about the experiment, including important details or the investigator's insights not covered by other fields.
- **Model:** This component describes the ML model used for training, including details such as its architecture, parameters, and configurations. Only authorized users may access the model when it is not publicly available. Therefore, only pointers (field *Source*) to the actual models are stored, along with hashes to verify the versions. The model's structure, particularly in deep learning, can vary widely, so its data are stored in JSON format extracted from the Workflow tool used for experiments.
- **Dataset:** This component provides details about the dataset used for training, validation, and testing. Similarly to the model, access to datasets may be restricted to certain users or groups. Therefore, only pointers (field *Source*) to the actual datasets are stored, along with hashes to verify the versions. Additionally, the field *Type* describes the type of data stored, such as text or images.
- **Parameters:** This component stores variables specific to the selected model, which are estimated by fitting the given data to the model. Examples include weights and biases in neural networks. These parameters are stored in JSON format and extracted from the workflow tool used for experiments.
- **Hyperparameters:** This component includes configurations for the model's experiments that are set before the experimental runs and remain unchanged during training. Examples include the data train-validation-test split ratio, learning rate, activation function, dropout rate, number of epochs, and batch size. Like parameters, hyperparameters are also stored in JSON format and extracted from the workflow tool.
- **Environment:** This component describes the programming languages (e.g., Python), libraries, and ML frameworks (such as Keras, NumPy, TensorFlow, Scikit-learn, and PyTorch) utilized in the experiment, also indicating the corresponding versions. To ensure flexibility, this information is stored in JSON format.
- **Hardware:** This component provides a description of the computing infrastructure utilized for ML experiments, including details such as the operating system, CPU, GPU, and RAM. This information is stored in JSON format for easy access and flexibility.
- **Results:** This component captures the results of the experiments carried out. It includes sets of validation and test results, such as accuracy and loss function values, stored in JSON format for efficient storage and retrieval.
- **Researcher:** This component corresponds to the description of the researcher or developer who conducted the experiment and initiated the transaction with the metadata stored on the blockchain.

In summary, a community-driven platform based on the concept described offers mechanisms that ensure trust, traceability, auditability, and full reproducibility in scientific experiments. This is achieved by leveraging blockchain technology to enhance the provenance of ML models and datasets, as well as computing infrastructure, and securely store experimentation metadata. The use of such a platform could improve the sharing of knowledge of experiments, resulting in a faster innovation cycle by facilitating collaboration and knowledge exchange between researchers, developers, and business entities. Next, we experimentally assess the blockchain implementation of the platform's concept within the Hyperledger Fabric infrastructure.

4. Experimental Infrastructure and Setup

In the upcoming section, we provide a brief overview of the fundamental concepts of the Hyperledger Fabric framework, which is utilized to implement the blockchain network. In addition, we will describe the experimental environment and its setup.

4.1. *Hyperledger Fabric*

In the current landscape of blockchain technology, along with the widely known Ethereum public blockchain system – credited as the first to support smart contracts – a variety of alternative systems have emerged, including Cosmos, Solana, Cardano, Polkadot, etc. (Ray, 2023). These blockchain systems boast scalability and robust development tools, catering to a diverse range of needs. The proposed platform is planned to operate on the principles of Proof-of-Authority (PoA) or Proof-of-Validity (PoV) consensus algorithms, facilitating a permissioned blockchain environment where only specific entities are authorized to participate. Hyperledger Fabric (HF) (Androulaki *et al.*, 2018) emerged as the preferred choice developed by IBM as a distributed ledger solution under the Linux Foundation. HF stands out for its robust support of strong identities and smart contracts, enabling organizations to share data across a distributed database without the need for individual users to place trust in others. It should be noted that HF's user-friendly nature and its declared satisfactory performance, thanks to its modular architecture tailored to business needs, ensure rapid transaction processing. Key features include permissioned access, high scalability, confidentiality, and flexibility. A Hyperledger Fabric blockchain network, composed of three organizations, is illustrated in Fig. 3, highlighting its key components.

In addition, we provide a brief description of the main Hyperledger Fabric components used in the paper, outlining their purposes and interactions. HF comprises several essential components, including peer nodes, ordering nodes, ledger, chaincode, Membership Service Provider (MSP), channels, Fabric Software Development Kit (SDK), and Fabric Certificate Authority (CA). Within HF, the ledger is divided into two elements: the world state and the blockchain. The world state contains the current state or latest values, with each node/peer maintaining an individual copy. Changes to the world state are synchronized across all peer nodes. The blockchain records transaction logs, capturing every change made to the world state. Access to the ledger is restricted to authorized

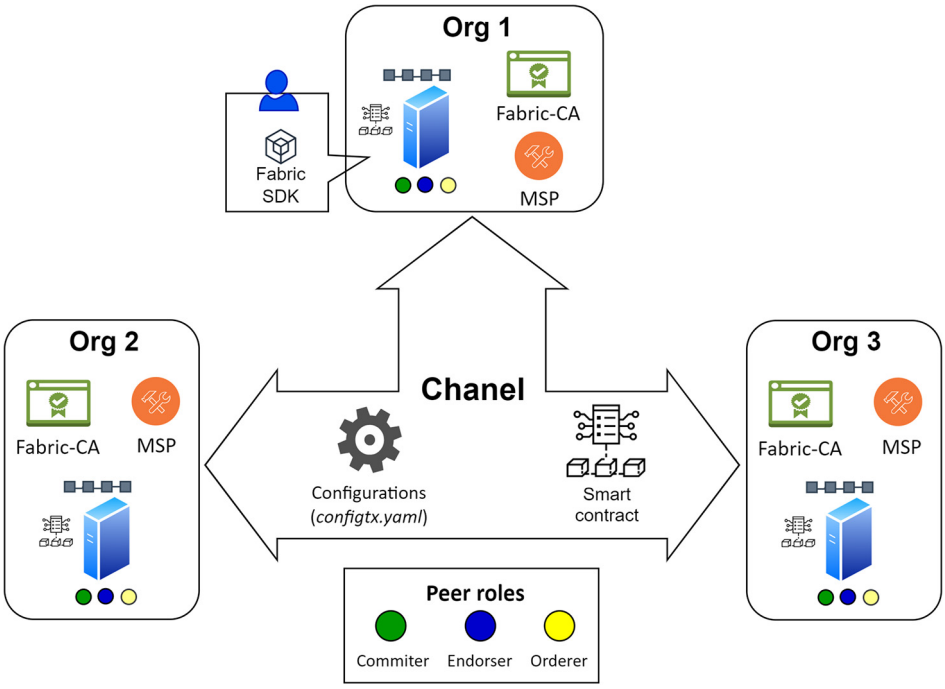


Fig. 3. A Hyperledger Fabric blockchain network, composed of three organizations.

smart contracts. Channels serve as private subnetworks facilitating secure communication among a subset of network participants, governed by pre-agreed rules outlined in the channel configuration. The peer nodes are responsible for maintaining the ledger copies associated with the network channels, executing chaincode, and validating transactions. The peer nodes can perform various roles, such as committing, endorsing, or ordering. Endorsing peers provide a seal of approval to transactions that are then submitted to the blockchain along with the endorsement. Committing peers store the transaction in their ledger upon submission. Additionally, they order nodes sequence and package validated transactions into blocks before transmitting them to peer nodes for validation and ledger commitment. Chaincode, also known as smart contracts, handles transactions among network participants and can be written in various programming languages. The MSP manages the identities of the network, registers new identities, and authenticates users. HF SDK, available in languages such as Node.js, Java, and Python, offers developers APIs to interact with the HF network. The CA issues and revokes digital certificates for network participants used for authentication and authorization. Transactions in HF undergo multiple stages, including execution, consensus-based ordering, and validation, before being stored in the ledger. The *configtx.yaml* file in HF serves as a configuration file used to define network parameters, including organizations, orderers, channels, and policies, essential for network initialization and operation. For a more detailed Hyperledger Fabric description, we refer to the official documentation (Fabric, 2024).

4.2. *Environment and Setup*

The concept of the platform revolves around two integrated modules: the AI Research module, which utilizes existing ML tools for experimentation, and the AI Blockchain Network, proposed in this study to enhance reproducibility in the ML research lifecycle. Therefore, in our experimental investigation, we focus on exploring the AI Blockchain Network components and evaluating the influence of Hyperledger Fabric parameters on system performance to identify optimal settings and potential bottlenecks within the blockchain infrastructure.

To experimentally evaluate the platform's blockchain performance, we constructed a blockchain network using the Hyperledger Fabric environment (v2.4.9), deployed as Docker (v24.0.6) containers on a local computer with an Intel(R) Core(TM) i5-10300H CPU running at 2.50 GHz and 8 GB of RAM. This network consists of three organizations (as depicted in Fig. 3), each with a peer and an ordering service represented by ordering nodes utilizing the RAFT consensus protocol. Configured to allow all organizations to validate transactions, the network's distributed ledger relies on the CouchDB database. We employed Hyperledger Caliper (v0.4.2) to monitor the network's performance, with transactions generated by clients, where users in Hyperledger Fabric correspond to these clients (ML researchers) interacting with the system via a client application. A smart contract, or chaincode, written in JavaScript, manages the developed metadata model and transactions, with performance evaluation focusing on three main transaction types: record, update, and query. Each transaction includes simulated metadata extracted from an ML workflow tool, specifically Neptune.AI, representing experiments carried out on the MNIST dataset (Deng, 2012) using various ML models. The record function creates new ledger entries, the update function modifies existing information, and the query function retrieves required data, where record and update transactions are submitted by application clients through peer nodes to the world-state database, while query transactions target the world state without involving the Orderer. Upon transaction verification, new blocks are added to the blockchain, with a send rate set at 100 TPS (transactions processed per second) and up to one hundred thousand transactions simulated in each round. We monitored the network's performance in terms of latency (measured in seconds), representing the time from transaction submission to response, and throughput, corresponding to the average number of transactions processed per second (TPS), while also assessing the workload on the main system components like Organizations, Orderers, and the Ledger.

5. Experimental Evaluation

This section presents the results and their analysis of the performance evaluation of the constructed and blockchain network. Three distinct scenarios were examined to assess the system's performance across various aspects, including the impact of block configuration, scalability, and workload fluctuations due to variations in main system parameters. Within each scenario, multiple test cases were explored, each adjusting key system parameters,

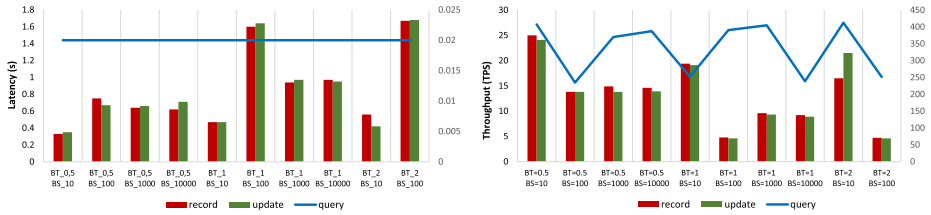
with each test case executed over 10 rounds, capped at a maximum duration of 1 hour per experiment. Subsequently, average values were calculated for each test case to gauge overall performance.

5.1. *Scenario 1: Impact of Block Size and Time on System Performance*

Block size (BS) and batch timeout (BT) (block creation time) are critical parameters with significant implications for system performance. Initially, their impact on system performance was assessed. BT denotes the waiting time after the first transaction arrives before creating a block, while BS represents the maximum number of transactions that can fill a block. Scenario 1 examined three BT cases – 0.5, 1, and 2 seconds – based on previous experiments and four BS cases – 10, 100, 1000, and 10000 transactions per block. Such a wide range of transaction sizes to be included in a block was investigated to fully explore the impact of their number and pinpoint potential bottlenecks. Throughout this scenario, the number of system clients remained fixed at 5. Refer to Fig. 4 for the results in 10 different test cases.

Figure 4a illustrates the average latency of the record, update, and query functions. Notably, the latency for the query function consistently remains low, with a stable value of 0.02 seconds across all test cases, attributed to its read-only nature. The latency of the record and update functions exhibits slight variations among test cases, but maintains consistency overall. The lowest latency values were observed with a batch timeout of 0.5 seconds and a block size of 10 transactions, producing latencies of 0.33 seconds for the record and 0.35 seconds for update functions. However, as BT increases to 1 and 2 seconds with BS fixed at 10 transactions, latency increases slightly to 0.47 seconds for both functions. In contrast, increasing BS to 100 transactions results in higher latency, peaks at 1.6 seconds for the record and 1.64 seconds for update functions with BT set to 1 second. Notably, latency is highest with BS $= 100$ and BT $= 0.5$ seconds, reaching 0.75 seconds for record and 0.71 seconds for update functions. However, increasing BS to 1000 and 10000 transactions leads to a drop in latency values for record and update functions, maintaining consistency within the same BT groups.

In Fig. 4b, the average values of the measured throughput are presented. Notably, the throughput for the query function fluctuates from approximately 250 TPS to 410 TPS, significantly higher than the throughput for the record and update functions, which varies from around 5 TPS to 25 TPS. The highest throughput values for the record and update functions were obtained in the test cases with BS $= 10$, resulting in 25 TPS for the record and 24.1 TPS for the update function with BT $= 0.5$, 19.4, and 19.1 TPS with BT $= 1$, and 16.5 and 21.5 TPS with BT $= 2$. In the case of BS $= 10$, the query function exhibited optimal performance only in cases with BT $= 0.5$ and BT $= 2$, resulting in 407.4 and 412.5 TPS, respectively. Conversely, similar to latency, the lowest throughput results for the record and update functions within each BT group were observed when the block size was set to 100 transactions. For this BS $= 100$, in the case of BT $= 0.5$, the update and record functions resulted in 13.8 TPS, while in the case of BT $= 1$, the record function achieved 4.8 TPS and the update function achieved 4.6 TPS.



(a) Latency (b) Throughput

Fig. 4. Scenario 1. Latency and throughput values of record, query, and update chaincode functions for different investigated test cases with varying block size and time.

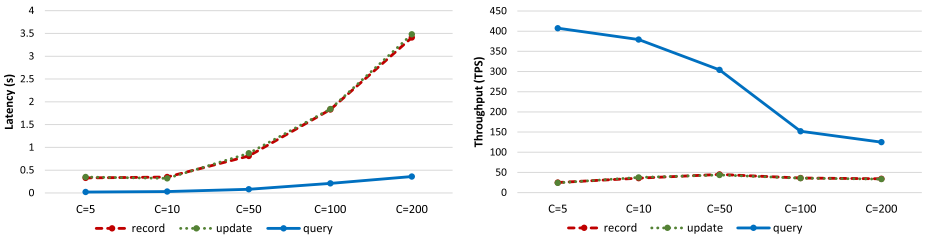
Overall, we notice an inverse correlation between network throughput and latency. Smaller block sizes, despite exhibiting lower throughput, require less time to fill the block with data, resulting in reduced latency. The choice between prioritizing latency reduction or increasing network throughput depends on the system developers' preferences. In conclusion, while the system performance in most analysed cases was satisfactory, this scenario helps identify more efficient block size and batch timeout values.

5.2. *Scenario 2: Impact of Number of Clients on System Performance*

When developing a platform with the potential for user growth, it is essential to assess the system's scalability capabilities. In this scenario, we examine how the system's performance is affected by the number of clients interacting with the system and initiating transactions. Here, the block size and batch timeout values were fixed at BS = 10 and BT = 0.5, respectively, as they demonstrated optimal performance in terms of latency and throughput in Scenario 1.

Figure 5 presents the system performance results for five different test cases varying the number of clients from 5 to 200. Similarly to Scenario 1, we observe that the latency values for the record and update functions are similar and significantly differ from those of the query function. As shown in Fig. 5a, for a relatively small number of clients (C = 5 and C = 10), latency is minimal. For instance, with C = 5, record, update, and query functions yielded latency values of 0.33, 0.35, and 0.02 seconds, respectively. With C = 10, these values were 0.35, 0.32, and 0.03 seconds, respectively. However, as the number of clients increases significantly, the latency also increases, reaching its highest values for the test case with 200 clients, resulting in 3.41, 3.48, and 0.36 seconds for the record, update and query functions, respectively. This represents a 10.33-fold increase for record functions, a 9.94-fold increase for the update function, and an 18-fold increase for the query function.

We observe a consistent decrease in throughput for the query function (see Fig. 5b) with the increase in the number of clients, which decreased from 407.4 to 125 TPS, resulting in a 3.26-fold decrease. Interestingly, there is no consistent decrease in throughput for the record and update functions. The lowest throughput values were obtained in the C = 10 case, with 25 and 24.1 TPS for record and update functions, respectively, while the highest throughput values were recorded in the C = 50 case, with the record and update functions reaching peaks at 45 and 43.7 TPS, respectively.



(a) Latency (b) Throughput

Fig. 5. Scenario 2. Latency and throughput values for record, query, and update chaincode functions for different tested cases investigated with varying numbers of clients.

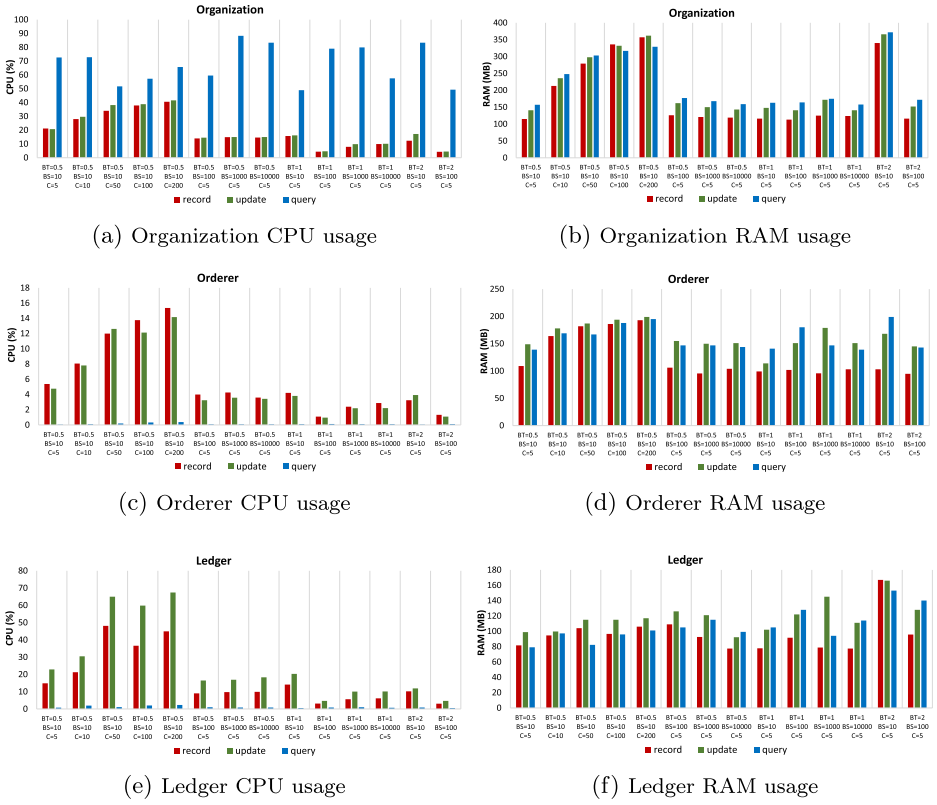
We can conclude that the increase in the number of clients in the blockchain network has a much greater impact on latency than on throughput. As the number of clients increases, the network is more likely to become overloaded, although the tests that were conducted have demonstrated respectable system performance even with a relatively high number of clients. In practice, if necessary, even greater network growth and overload can be overcome by expanding network resources, such as network channels and peer nodes.

5.3. *Scenario 3: CPU and Memory Usage Analysis*

The blockchain operations executed by the peer nodes require noticeable computing power and memory resources. Therefore, it is crucial to assess system performance from a resource usage perspective. In this scenario, we conduct an analysis of CPU and RAM utilization to pinpoint bottlenecks, identify the most computationally intensive components, and highlight any memory leakage issues. Our thorough examination reveals that peer nodes from different organizations consume comparable amounts of resources. Therefore, in Fig. 6, we present resource utilization metrics for key node roles, including Organization, Orderer, and Ledger, in all previously investigated test cases.

Organization In the Organization (refer to Fig. 6a), CPU usage is predominantly allocated to the query function, ranging from 48.95% to 88.31%. Both the record and update functions exhibit similar CPU usage values in all test cases, ranging from 4.33% to 41.55%. Lower block timeout values (BT = 0.5) result in increased CPU usage for record and update compared to higher BT values (BT=1 and BT=2) with the same number of clients (C = 5). An increase in the number of clients from 5 to 200 (observed in test cases with BT = 0.5 and BS = 10) reveals a consistent increase in CPU usage, from 21. 21% to 40. 54% for the record function and from 20.74% to 41% for the update function. Furthermore, smaller block sizes (BS = 10) generally require more CPU usage for record and update functions compared to higher BS values, with the lowest CPU values observed when BS = 100. However, such dependencies are not observed for the query function.

Regarding RAM usage in the Organization (refer to Fig. 6b), it ranges from approximately 113 MB to 357 MB for the record function, from approximately 141 MB to 366 MB for the update function, and from approximately 157 MB to 372 MB for the query



(a) Organization CPU usage (b) Organization RAM usage

(c) Orderer CPU usage (d) Orderer RAM usage

(e) Ledger CPU usage (f) Ledger RAM usage

Fig. 6. Scenario 3. CPU and RAM utilization by Organization, Orderer, and Ledger for record, query, and update chaincode functions across all investigated test cases.

function. Unlike CPU usage, all functions (record, update, and query) exhibit similar RAM usage values across all test cases. In most cases, the update function utilizes slightly more RAM, with an average difference of 8.23%. Additionally, the query function tends to use the most RAM, averaging 12.57% more compared to the record function and 4% more compared to the update function. However, this trend for the query function diminishes as the number of clients increases. Furthermore, an increase in the number of clients from 5 to 200 (observed in test cases with BT = 0.5 and BS = 10) results in a consistent increase in RAM usage, from 115 MB to 357 MB for the record function, from 141 MB to 362 MB for the update function, and from 157 MB to 329 MB for the query function. Overall, increasing the number of clients from 5 to 200 resulted in a 2.58-fold increase in RAM utilization on average. In particular, there is no significant influence of BT and BS on RAM usage, except in the case BT = 2, BS = 10, C = 5.

Orderer In contrast to the Organization, the Orderer primarily utilizes CPU computing power for the record and update functions (refer to Fig. 6c). The workload of the query in the Orderer is negligible, reaching a CPU usage value of only 0.37% with a large number

of clients (C = 200). The record and query functions exhibit comparable CPU usage values across all test cases. For a fixed number of clients (C = 5), the highest CPU usage values are observed for the record and update functions in test cases with BS = 10, resulting in 5.37% for the record and 4.46% for the update function in the case BT = 0.5, 4.21% and 3.81% in the case BT=1, and 3.25% and 3.93% in the case BT = 2. An increase in the number of clients (observed in test cases with BT = 0.5 and BS = 10) results in a constant CPU usage increase for all functions – from 5.37% to 15.37% for the record function, from 4.76% to 14.16% for the update function, and from 0.05% to 0.37% for the query function.

Regarding RAM usage, the Orderer exhibits a narrower range (refer to Fig. 6d) compared to the Organization. RAM usage varies from approximately 95 MB to 124 MB for the record function, from approximately 114 MB to 162 MB for the update function, and from approximately 139 MB to 160 MB for the query function. For a fixed small number of clients (C = 5), compared to the record function, the update function requires 49.47% more RAM, and the query function requires 50.76% more RAM. There is no significant impact of block size and block timeout on RAM usage. When increasing the number of clients from 5 to 200 (test cases with BT = 0.5 and BS = 10), some RAM usage increase is observed, but it is not as significant compared to the Organization and results in a 50% increase in RAM utilization on average.

Ledger Similarly to the Orderer, the Ledger primarily utilizes CPU computing power for the record and update functions (refer to Fig. 6e). The workload of the query function is small, averaging 1.08%, and reaches a CPU usage value of 2.34% only with a large number of clients (C = 200). However, the update function consistently exhibits the highest CPU load in all test cases, averaging 51.75% more compared to the record function. Similarly to the Organization case, for lower block timeout values (BT = 0.5), the CPU usage for the record and update is higher compared to test cases with higher BT values (BT = 1 and BT = 2) when considering appropriate BS values (BS = 10; BS = 100; BS = 1000; BS = 10000) and the same number of clients (C = 5). The increase in the number of clients requires significantly more CPU resources (observed in test cases with BT = 0.5 and BS = 10) – when comparing cases with 5 and 50 clients, there is a 3.24 times increase for the record function and a 2.84 times increase for the update function. However, a stable growth trend in CPU usage is not observed – starting from C = 50, CPU usage begins to stabilize.

In terms of RAM usage, the Ledger requires the least amount compared to the Organization and the Orderer (refer to Fig. 6f). On average, the record function requires 96 MB, the update function requires 118 MB, and the query function requires 107 MB. There is no significant impact of the number of clients on RAM usage – the increase from 5 to 200 clients resulted in only a 25.33% increase on average. Similarly, block size and block timeout have no significant impact on RAM usage.

Overall, within the Organization, CPU computing power is predominantly utilized for the query function, reaching the highest workload values among all nodes. On the contrary, the workload of the query function at the Orderer and Ledger is minimal. However, the scenario is reversed for the record and update functions, as they require much more CPU

computational power on the Orderer and Ledger. Interestingly, these functions exhibit a similar level of CPU workload on a node. It is worth noting that both the Organization and the Ledger require more CPU resources than the Orderer. Additionally, across all node types, CPU workload appears to be less influenced by block size and block timeout, but more affected by the number of clients. When considering RAM utilization, there are no significant differences between functions and neither block size nor block timeout have a substantial impact on RAM usage. The number of clients has a greater impact on the Organization, although its influence diminishes in the case of the Orderer and Ledger. In summary, RAM usage across all cases remains relatively low and feasible for the majority of modern computers.

In conclusion, the CPU and RAM resource analysis conducted underscores the suitability of the Hyperledger Fabric infrastructure for the developed blockchain-based reproducibility platform at the real-world application level.

6. Conclusions and Challenges

This paper introduces a novel concept: a blockchain-based platform designed to facilitate reproducible research in machine learning, adaptable to various domains within artificial intelligence. The platform operates on a decentralized blockchain network, comprising organizations and researchers actively engaged in ML, and seamlessly integrates various ML research frameworks and tools. Using blockchain features such as decentralization, immutability, and a community-driven experiment validation consensus mechanism, along with automation through smart contracts, the platform enhances transparency and trust in ML research. It ensures the immutable storage of experiment artifacts within transactions and facilitates their retrieval by other researchers. Experimental results examining the performance of the blockchain network demonstrate that most block configurations provide sufficient latency and throughput for transactions. Moreover, experiments involving varying numbers of clients showcase a commendable level of scalability for the platform. Additionally, resource usage analysis reveals that running a node on the platform is affordable for a standard modern computer. In conclusion, the Hyperledger Fabric infrastructure proves to be well-suited for developing the platform at a real-world application level.

Although the platform built on the architecture presented holds significant potential, several challenges must be addressed to ensure its effectiveness. Integration with existing ML tools emerges as a critical priority, facilitating experimental metadata extraction, its storage on the blockchain network, and subsequent retrieval. This integration is essential for streamlining research processes and ensuring seamless adoption of the platform. Establishing a community-driven consensus protocol, complemented by robust reputation management mechanisms, becomes imperative to guarantee the accurate validation of results. This framework will maintain the integrity of the research results and foster trust within the ML community. Moreover, implementing appropriate incentive mechanisms is essential to incentivize ML investigators to actively participate in model and experiment validation. Finally, integration with other AI research-oriented blockchains will increase the platform's attractiveness and usability. These incentives will encourage engagement and collaboration, driving the platform's growth and impact.

A. Appendix. Review and Comparative Analysis of Blockchain Interoperability Approaches

In this section, we provide a description and comparative analysis of the main blockchain interoperability approaches, as well as indicate potential approaches that could be used to implement the platform's interoperability with other AI research-oriented blockchains.

In many practical applications, there arises a need to interact with data and assets across different blockchains (Li *et al.*, 2023). However, various application scenarios and use cases entail different requirements for blockchain access control, throughput, scalability, etc. To address this, numerous interoperability approaches and solutions have been proposed to enable communication, interaction and information sharing among different blockchain networks (as seen in the survey papers Belchior *et al.*, 2021; Ren *et al.*, 2023; Wang *et al.*, 2023). Based on the literature, blockchain interoperability approaches can be classified into seven categories: (1) Side-chains, (2) Notary schemes, (3) Hash locking, (4) Trusted relays, (5) Blockchain engines, (6) Blockchain of blockchains, and (7) Agnostic protocols. We further provide a description of each approach, highlighting its specificity.

- **Side-chains.** A side-chain is a separate blockchain linked to the main blockchain, that typically enhances its functionality by offering a mechanism to conduct transactions off the main blockchain and enabling scalability and enhanced functionality. These side-chains operate by pegging assets from the main blockchain, facilitating independent transaction processing with tailored consensus mechanisms. Additionally, assets from the main chain can be seamlessly transferred to the side-chain for processing and vice versa, fostering interoperability. Main blockchains can host multiple side-chains, each dedicated to specific use cases or applications, thus accommodating diverse needs.
- **Notary schemes.** The primary objective of a notary scheme is to establish a trusted mechanism to validate events within a blockchain and attest their validity to another blockchain. Within the notary mechanism, a trusted third party or a consortium of parties can oversee multiple chains, facilitating transactions on one chain in response to valid events or specific requests, such as those initiated through smart contracts. Notary schemes verify the integrity of cross-chain transactions by employing techniques such as Merkle trees, digital signatures, and timestamping. Although notary schemes may face challenges related to decentralization, their implementation remains straightforward.
- **Hash locking.** Hash locking, also known as hash time locking contracts (HTLCs), utilizes hash functions and delayed transaction execution on the blockchain. These smart contracts facilitate asset exchanges between different blockchains by locking assets and establishing the corresponding time and unlocking conditions to ensure atomicity. Transactions remain locked until both parties agree on their obligations, ensuring secure and trustless exchanges. Through hash-locking, atomic swaps enable asset exchanges between different blockchains without intermediary intervention, enhancing efficiency and decentralization in cross-chain transactions.
- **Trusted relays.** A trusted relay serves as a bridge, enabling the seamless provision of smart contract services across various blockchains. Through this approach, essential

block information, such as block headers, is replicated from the source blockchain to the target blockchain via verifiable smart contracts. Contracts in one chain effectively assume the role of clients in another chain, enabling recipient chains to verify activities that occur in other chains. This approach allows the target blockchain to independently validate the existence of data on the source blockchain. Notably, trusted relays operate at a chain-to-chain level, bypassing the need for trust in distributed nodes, thereby distinguishing them from notary schemes.

- **Blockchain engines.** A blockchain engine is a sophisticated framework designed to offer reusable components across various layers—data, network, consensus, incentive, and contracts—to enable the creation of tailored blockchains. These blockchains, in turn, power decentralized applications that seamlessly interact with one another. Typically, an engine requires a shared infrastructure that supports services across multiple layers, including network and consensus mechanisms. Unlike mere relays among blockchains, this infrastructure integrates all necessary components, fostering enhanced availability and compatibility. However, despite their potential, many blockchain engine-based solutions remain in the early stages of development due to the complexities involved in their implementation.
- **Blockchain of blockchains.** The blockchain of blockchains can be defined as a framework for building decentralized applications that seamlessly interact across multiple blockchains, enabling each blockchain to function autonomously. These frameworks offer core components such as the mainchain, network infrastructure, consensus mechanisms, incentives, and smart contracts to facilitate the development of application-specific custom blockchains, also known as subchains, that can interconnect with each other. In contrast to side-chain solutions, where all operations are centrally coordinated by the mainchain, the blockchain of blockchains employs a notary scheme. Here, the mainchain acts as a notary, documenting activities occurring on subchains. This approach ensures interoperability while maintaining decentralization and scalability across the network.
- **Agnostic protocols.** These protocols address the challenges of interoperability that arise from the diversity of blockchain platforms by establishing a standardized set of rules and interfaces. A blockchain agnostic protocol usually denotes a unified platform that facilitates the simultaneous operation of multiple blockchains, providing a layer of abstraction to enable cross-chain communication among distributed ledgers. Such a platform empowers end-users to conduct transactions across different blockchains and facilitates the transfer of assets and data between these networks. However, due to their nature, agnostic protocols may not ensure backward compatibility, meaning that existing blockchains need to modify their source code to integrate with these protocols.

All described interoperability approaches are rooted in a trust model that can be broadly classified into two categories: trusted third party (TTP) and synchrony (Zamyatin *et al.*, 2021). In a TTP model, a trusted third party can serve as the coordinator to ensure accurate execution of cross-chain communications. On the other hand, synchrony relies on the assumption of synchronous communication among participants and utilizes locking mechanisms, typically in the form of smart contracts, to facilitate interoperability.

Table 2
Comparative analysis of interoperability approaches.

Approach	Type	Decentralization	Scalability	Security	Connected chains	Communication	Trust model	Use case
Side-chains	Chain-based	Medium	High	Medium	Two	Bidirectional	Synchrony	Asset transfer
Notary schemes	Chain-based	Low	High	High	Two	Unidirectional, Bidirectional	TTP	Atomic Swap, Asset transfer, Cross-chain oracles
Hash locking	Chain-based	High	Low	High	Two	Bidirectional	Synchrony	Atomic Swap, Asset transfer
Trusted relays	Bridge-based	High	Medium	Medium	Two	Unidirectional, Bidirectional	TTP	Atomic Swap, Asset transfer, Chain relay, Cross-chain Oracles
Blockchain engines	Bridge-based	Medium	Medium	High	Multiple	Bidirectional	TTP	Atomic swap, Asset transfer, Chain relay, Cross-chain oracles
Blockchain of blockchains	DApp-based	Medium	High	Medium	Multiple	Bidirectional	TTP	Asset transfer, Cross-chain oracles
Agnostic protocols	DApp-based	Medium	Medium	Medium	Multiple	Bidirectional	TTP	Asset transfer, Cross-chain oracles

Blockchain interoperability facilitates a wide range of use cases. Atomic swaps enable the direct exchange of cryptocurrencies or assets between two blockchain systems without the need for intermediaries. Asset transfer involves relocating assets from one blockchain to another, typically involving locking or burning the asset in the original blockchain before releasing a corresponding representation on the destination blockchain. Cross-chain oracle facilitates seamless data transfer or provision between different blockchains. Unlike asset transfers, data transfers via cross-chain oracles can occur across multiple blockchains without locking or burning the original blockchain. While in chain relay, the communication between blockchain networks is based on intermediary nodes.

Based on the state-of-the-art literature, we present a summary of the main compared features of the described interoperability approaches in Table 2. In addition to categorizing the type of approach, we analyse three prominent and conflicting aspects of the blockchain trilemma: decentralization, scalability, and security (Buterin, 2017). Moreover, we ascertain the potential number of connected chains, specify communication direction, examine the trust model, and present used cases.

We can observe three distinct types of approaches utilized for interoperability: chain-based, bridge-based, and decentralized application-based (DApp-based). Chain-based blockchain interoperability approaches are primarily dedicated to homogeneous blockchain systems where assets are of the same type. On the contrary, bridge-based approaches aim to interconnect heterogeneous blockchain systems, while DApp-based approaches focus on exchanging information between applications built on top of blockchains. Considering the blockchain trilemma aspect, each solution proposes a trade-off between decentralization, scalability, and security, suggesting specific approach considerations depending on the developed system's aims and purposes. More recent interop-

erability approaches, such as blockchain engines, blockchain of blockchains, and agnostic protocols, offer the possibility of connecting and exchanging information between more than two blockchains. In particular, all approaches support bidirectional communications. Moreover, with the exception of side-chains, each offers different use cases.

Various interoperability solutions have been developed for different types of approaches across various blockchains, including the widely popular public blockchain Ethereum (for more details, refer to (Belchior *et al.*, 2021; Ren *et al.*, 2023)). Considerable attention has been paid to interoperability solutions for permissioned blockchains such as Hyperledger Fabric, which is utilized in this study to build a blockchain network. For example, Hyperledger Cacti (Cacti, 2024) serves as a multi-faceted pluggable interoperability framework designed to connect networks built on different blockchain technologies and facilitate transactions that span multiple networks. This project emerged from the merger of the Weaver Lab project with Hyperledger Cactus. Another example is Hyperledger Firefly (FireFly, 2023), a pluggable framework that aims to transact on multiple blockchains. It is an open-source project that aims to provide a decentralized, adaptable, and secure integration between blockchains and various platforms. Additionally, Hyperledger YUI (YUI, 2022) is an open-source project of Hyperledger Labs that targets blockchain interoperability using the inter-blockchain communication (IBC) protocol by Cosmos (Kwon and Buchman, 2015). Lastly, Wecross (WeCross, 2019) is an open-source blockchain interoperability platform that supports popular permissioned blockchains, including Hyperledger Fabric. These solutions can be considered to implement interoperability between the platform and other blockchains oriented to AI research.

Acknowledgements

This research received funding from the Research Council of Lithuania (LMTLT), agreement No. S-MIP-21-53.

References

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y. Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, Ch., Vukolić, M., Cocco, S.W., Yellick, J. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In: *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15.

Bag, R., Spilak, B., Winkel, J., Härdle, W.K. (2022). Quantinar: a blockchain p2p ecosystem for honest scientific research. arXiv preprint. arXiv:2211.11525.

Bathen, L.A.D., Jadav, D. (2022). Trustless AutoML for the Age of Internet of Things. In: *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, pp. 1–3.

Bayer, D., Haber, S., Stornetta, W.S. (1993). Improving the efficiency and reliability of digital time-stamping. In: *Sequences II: Methods in Communication, Security, and Computer Science*. Springer, pp. 329–334.

Belchior, R., Vasconcelos, A., Guerreiro, S., Correia, M. (2021). A survey on blockchain interoperability: past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8), 1–41.

Beltrán, E.T.M., Pérez, M.Q., Sánchez, P.M.S., Bernal, S.L., Bovet, G., Pérez, M.G., Pérez, G.M., Celdrán, A.H. (2023). Decentralized federated learning: fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 25(4), 2983–3013.

Bertolini, M., Mezzogori, D., Neroni, M., Zammori, F. (2021). Machine learning for industrial applications: a comprehensive literature review. *Expert Systems with Applications*, 175, 114820.

Buterin, V. (2017). *The Meaning of Decentralization*. https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

Cacti (2024). *Hyperledger Cacti*. https://www.hyperledger.org/projects/cacti.

Cao, B., Wang, Z., Zhang, L., Feng, D., Peng, M., Zhang, L., Han, Z. (2022). Blockchain systems, technologies, and applications: a methodology perspective. *IEEE Communications Surveys & Tutorials*, 25(1), 353–385.

Coelho, R., Braga, R., David, J.M.N., Dantas, M., Ströele, V., Campos, F. (2020). Blockchain for reliability in collaborative scientific workflows on cloud platforms. In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, pp. 1–7.

Coelho, R., Braga, R., David, J.M.N., Stroele, V., Campos, F., Dantas, M. (2022). A blockchain-based architecture for trust in collaborative scientific experimentation. *Journal of Grid Computing*, 20(4), 35.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6), 141–142.

Fabric, H. (2024). *Hyperledger Fabric Docs*. https://hyperledger-fabric.readthedocs.io/en/latest/index.html.

Filatovas, E., Marcozzi, M., Mostarda, L., Paulavičius, R. (2022). A MCDM-based framework for blockchain consensus protocol selection. *Expert Systems with Applications*, 204, 117609.

FireFly, H. (2023). *Hyperledger FireFly Docs*. https://hyperledger.github.io/firefly/.

Gudžius, P., Kurasova, O., Darulis, V., Filatovas, E. (2021). Deep learning-based object recognition in multispectral satellite imagery for real-time applications. *Machine Vision and Applications*, 32(4), 98.

Gudzius, P., Kurasova, O., Darulis, V., Filatovas, E. (2022). AutoML-based neural architecture search for object recognition in satellite imagery. *Remote Sensing*, 15(1), 91.

Gundersen, O.E., Shamsaliei, S., Isdahl, R.J. (2022). Do machine learning platforms provide out-of-the-box reproducibility? *Future Generation Computer Systems*, 126, 34–47.

Haber, S., Stornetta, W.S. (1991). How to time-stamp a digital document. In: Menezes, A.J., Vanstone, S.A. (Eds.), *Advances in Cryptology-CRYPTO' 90*. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 437–455.

Harris, J.D., Waggoner, B. (2019). Decentralized and collaborative AI on blockchain. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, pp. 368–375.

Hoopes, R., Hardy, H., Long, M., Dagher, G.G. (2022). SciLedger: a blockchain-based scientific workflow provenance and data sharing platform. In: *2022 IEEE 8th International Conference on Collaboration and Internet Computing (CIC)*. IEEE, pp. 125–134.

Hutson, M. (2018). Artificial intelligence faces reproducibility crisis. *Science*, 359(6377), 725–726.

Juodis, M., Filatovas, E., Paulavičius, R. (2024). Overview and empirical analysis of wealth decentralization in blockchain networks. *ICT Express*.

Kannan, K., Singh, A., Verma, M., Jayachandran, P., Mehta, S. (2020). Blockchain-based platform for trusted collaborations on data and AI models. In: *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, pp. 82–89.

Khoi Tran, N., Sabir, B., Babar, M.A., Cui, N., Abolhasan, M., Lipman, J. (2022). ProML: a decentralised platform for provenance management of machine learning software systems. In: *Software Architecture: 16th European Conference, 2022, Proceedings*, ECSA 2022, Prague, Czech Republic, September 19–23. Springer, pp. 49–65.

Knez, T., Gašperlin, D., Bajec, M., Žitnik, S. (2022). Blockchain-based transaction manager for ontology databases. *Informatica*, 33(2), 343–364.

Kwon, J., Buchman, E. (2015). *Comsos: A Network of Distributed Ledgers*. https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md.

Lamport, L., Shostak, R., Pease, M. (1982). The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3), 382–401.

Li, L., Wu, J., Cui, W. (2023). A review of blockchain cross-chain technology. *IET Blockchain*, 3(3), 149–158.

Li, Y., Chen, C., Liu, N., Huang, H., Zheng, Z., Yan, Q. (2021). A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network*, 35(1), 234–241.

Liu, F., Chen, D., Wang, F., Li, Z., Xu, F. (2023). Deep learning based single sample face recognition: a survey. *Artificial Intelligence Review*, 56(3), 2723–2748.

Lo, S.K., Liu, Y., Lu, Q., Wang, C., Xu, X., Paik, H.Y., Zhu, L. (2022). Towards trustworthy AI: blockchain-based architecture design for accountability and fairness of federated learning systems. *IEEE Internet of Things Journal*, 10(4), 3276–3284.

Lu, Y., Huang, X., Dai, Y., Maharjan, S., Zhang, Y. (2019). Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Transactions on Industrial Informatics*, 16(6), 4177–4186.

Lüthi, P., Gagnaux, T., Gygli, M. (2020). Distributed ledger for provenance tracking of artificial intelligence assets. In: Friedewald, M., Önen, M., Lievens, E., Krenn, S., Fricker, S. (Eds.), *Privacy and Identity Management. Data for Better Living: AI and Privacy. Privacy and Identity 2019*. *IFIP Advances in Information and Communication Technology*, Vol. 576. Springer, Cham, pp. 411–426. https://doi.org/10.1007/978-3-030-42504-3_26.

Marcozzi, M., Filatovas, E., Stripinis, L., Paulavičius, R. (2024). Data-driven consensus protocol classification using machine learning. *Mathematics*, 12(2), 221.

Matulevičius, R., Iqbal, M., Elhadjamor, E.A., Ghannouchi, S.A., Bakhtina, M., Ghannouchi, S. (2022). Ontological representation of healthcare application security using blockchain technology. *Informatica*, 33(2), 365–397.

Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., Poria, S. (2023). A review of deep learning techniques for speech processing. *Information Fusion*, 99, 101869.

Meng, Q., Sun, R. (2021). Towards secure and efficient scientific research project management using consortium blockchain. *Journal of Signal Processing Systems*, 93, 323–332.

Mora-Cantallops, M., Sánchez-Alonso, S., García-Barriocanal, E., Sicilia, M.A. (2021). Traceability for trustworthy AI: a review of models and tools. *Big Data and Cognitive Computing*, 5(2), 20.

Mothukuri, V., Parizi, R.M., Pouriyeh, S., Dehghantanha, A., Choo, K.K.R. (2021). FabricFL: blockchain-in-the-loop federated learning for trusted decentralized systems. *IEEE Systems Journal*, 16(3), 3711–3722.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf.

Paulavičius, R., Grigaitis, S., Igumenov, A., Filatovas, E. (2019). A decade of blockchain: review of the current status, challenges, and future directions. *Informatica*, 30(4), 729–748.

Paulavičius, R., Grigaitis, S., Filatovas, E. (2021). A systematic review and empirical analysis of blockchain simulators. *IEEE Access*, 9, 38010–38028.

Pimentel, J.F., Murta, L., Braganholo, V., Freire, J. (2019). A large-scale study about quality and reproducibility of jupyter notebooks. In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, pp. 507–517.

Ray, P.P. (2023). Web3: A comprehensive review on background, technologies, applications, zero-trust architectures, challenges and future directions. *Internet of Things and Cyber-Physical Systems*, 3, 213–248.

Ren, K., Ho, N.M., Loghin, D., Nguyen, T.T., Ooi, B.C., Ta, Q.T., Zhu, F. (2023). Interoperability in blockchain: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(12), 12750–12769.

Rowhani-Farid, A., Barnett, A.G. (2018). Badges for sharing data and code at Biostatistics: an observational study. *F1000Research*, 7, 90. https://doi.org/10.12688/f1000research.13477.2.

Sakalauskas, E., Bendoraitis, A., Lukšaitė, D., Butkus, G., Vitkutė-Adžgauskienė, D. (2023). Tax declaration scheme using blockchain confidential transactions. *Informatica*, 34(3), 603–616.

Sarpatwar, K., Vaculin, R., Min, H., Su, G., Heath, T., Ganapavarapu, G., Dillenberger, D. (2019). Towards enabling trusted artificial intelligence via blockchain. In: Calo, S., Bertino, E., Verma, D. (Eds.), *Policy-Based Autonomic Data Governance*, *Lecture Notes in Computer Science*, Vol. 11550. Springer, Cham, pp. 137–153.

Schelter, S., Boese, J.H., Kirschnick, J., Klein, T., Seufert, S. (2017). Automatically tracking metadata and provenance of machine learning experiments. In: *Machine Learning Systems Workshop at NIPS 2017*, Long Beach, CA, USA.

Stodt, J., Stodt, F., Reich, C., Clarke, N. (2022). Verifiable machine learning models in industrial IoT via blockchain. In: *Proceedings of the 12th International Advanced Computing Conference*, Hyderabad, Telangana, pp. 16–17.

Ullah, I., Deng, X., Pei, X., Jiang, P., Mushtaq, H. (2023). A verifiable and privacy-preserving blockchain-based federated learning approach. *Peer-to-Peer Networking and Applications*, 16(5), 2256–2270.

Usuga Cadavid, J.P., Lamouri, S., Grabot, B., Pellerin, R., Fortin, A. (2020). Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing*, 31, 1531–1558.

Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L. (2014). OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2), 49–60.

Vartak, M., Subramanyam, H., Lee, W.E., Viswanathan, S., Husnoo, S., Madden, S., Zaharia, M. (2016). ModelDB: a system for machine learning model management. In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pp. 1–3.

Wang, G., Wang, Q., Chen, S. (2023). Exploring blockchains interoperability: a systematic survey. *ACM Computing Surveys*, 55(13s), 1–38.

WeCross (2019). *WeCross*. https://github.com/WeBankBlockchain/WeCross.

Weng, J., Weng, J., Zhang, J., Li, M., Zhang, Y., Luo, W. (2019). Deepchain: auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, 18(5), 2438–2455.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering – EASE '14*. ACM Press, pp. 1–10.

YUI (2022). *Hyperledger YUI*. https://labs.hyperledger.org/labs/yui.html.

Zamyatin, A., Al-Bassam, M., Zindros, D., Kokoris-Kogias, E., Moreno-Sanchez, P., Kiayias, A., Knottenbelt, W.J. (2021). Sok: Communication across distributed ledgers. In: Borisov, N., Diaz, C. (Eds.), *Financial Cryptography and Data Security, FC 2021*, *Lecture Notes in Computer Science*, Vol. 12675. Springer, Berlin, Heidelberg, pp. 3–36.

Zhang, C., Lu, Y. (2021). Study on artificial intelligence: the state of the art and future prospects. *Journal of Industrial Information Integration*, 23, 100224.

E. Filatovas received a PhD degree in informatics engineering from Vilnius University, Lithuania, in 2012. He is currently a principal researcher and co-founder of the Blockchain and Quantum Technologies Group at the Institute of Data Science and Digital Technologies, Vilnius University. His main research interests include blockchain and distributed ledger technologies, parallel and quantum computing, global and multiobjective optimization, and machine learning.

L. Stripinis received a PhD degree in informatics from Vilnius University, Lithuania, in 2021. He is currently a researcher at Vilnius University. His research interests include global optimization, optimization software, parallel and distributed computing, and machine learning.

F. Orts is a senior researcher at Vilnius University, Lithuania. Additionally, he is a PhD professor at the International University of La Rioja, and a collaborating professor at the Universitat Oberta de Catalunya, both universities in Spain. He also collaborates actively with the Supercomputing-Algorithms group at the University of Almeria, Spain. He has worked as a computer engineer in construction, stock market and IT service companies, with more than 15 years of experience in the sector.

R. Paulavičius received a PhD degree in computer science from Vytautas Magnus University, Kaunas, Lithuania, in 2010. He was a postdoctoral researcher at Vilnius University, Vilnius, Lithuania, and a research associate at Imperial College London, London, UK. He is currently chief researcher and professor at Vilnius University. His research interests include global optimization, optimization software, parallel and quantum computing, and distributed ledger technologies.