I.J. Image, Graphics and Signal Processing, **2025, 3, 55-84**
Published Online on June 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2025.03.04

Hybrid System for Image Storage and Retrieval in Big Data Environments

Glib Tereshchenko
Software Engineering, Faculty of Computer Science, Kharkiv National University of Radio Electronics, Kharkiv, 61166, Ukraine
E-mail: hlib.tereshchenko@nure.ua
ORCID iD: https://orcid.org/0000-0001-8731-2135

Iryna Kyrychenko
Software Engineering, Faculty of Computer Science, Kharkiv National University of Radio Electronics, Kharkiv, 61166, Ukraine
E-mail: iryna.kyrychenko@lnure.ua
ORCID iD: https://orcid.org/0000-0002-7686-6439

Victoria Vysotska
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: Victoria.A.Vysotska@lpnu.ua
ORCID iD: https://orcid.org/0000-0001-6417-3689

Zhengbing Hu
School of Computer Science, Hubei University of Technology, Wuhan, China
E-mail: drzbhu@gmail.com
ORCID iD: https://orcid.org/0000-0002-6140-3351

Yuriy Ushenko*
Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58012, Ukraine
E-mail: y.ushenko@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0003-1767-1882
*Corresponding Author

Mariia Talakh
Yuriy Fedkovich Chernivtsi National University, Computer Science Department, Chernivtsi, 58002, Ukraine
E-mail: m.talah@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-5067-6848

Received: 22 January, 2025; Revised: 23 February, 2025; Accepted: 20 March, 2025; Published: 08 June, 2025

Abstract: This paper presents a hybrid image storage model for big data environments. The model combines relational and non-relational (NoSQL) databases, file systems (IPFS), and blockchain technologies to ensure an optimal balance between performance, scalability, and security in image storage. The existing approaches to organising image data storage and image compression methods in decentralised systems are analysed. Optimised image indexing is proposed to accelerate data search and access. A prototype system based on the proposed model was developed, and an experimental study was conducted on various image datasets (medical, satellite, and digital art). The experimental results demonstrate that the hybrid model outperforms traditional approaches: image access time is reduced by ~30% compared to standalone storage systems, providing high scalability (with increased nodes, processing time decreases nonlinearly). The efficiency of image compression in reducing storage costs in blockchain-oriented systems is also confirmed: the WebP format allows file size to be reduced by 40–60% while maintaining acceptable quality (PSNR > 30 dB). The proposed solution is relevant for medical diagnostics, video surveillance systems, geographic information systems, and other fields requiring reliable storage and fast processing of large-scale image datasets.

Index Terms: Big Data, Hybrid Image Storage, Relational Database, Nosql, IPFS, Blockchain, Image Compression, Indexing, Performance, Security

1. Introduction

The volume of digital images is rapidly increasing across various fields – from social media and medicine to video surveillance and scientific research. Storing and processing such a vast number of images is a challenging task in big data environments. Traditional storage approaches based on relational databases or centralised file servers often fail to meet scalability, access speed, and data reliability requirements. On the other hand, complete decentralisation of storage (e.g., based on peer-to-peer networks or blockchain) may lead to significant access delays and complicate indexing and searching for required images. Therefore, there is an urgent need to develop approaches combining the strengths of various storage technologies to ensure efficient management of large images.

Relational DBMSs (SQL) reliably support data integrity and complex queries but scale mainly vertically and may lose performance when storing large volumes of unstructured data, such as images. NoSQL systems (document-oriented, key-value, etc.) are designed for horizontal scalability and high-speed performance on large data sets. Still, they often lag behind SQL in terms of transaction complexity and integrity guarantees. Distributed file systems and networked storage like HDFS or IPFS offer decentralised storage with data replication across network nodes, increasing fault tolerance. However, access to data in such networks can be slower and depends on network connectivity.

One promising direction is using blockchain technology to store critical metadata or image hash identifiers. Blockchain enables the immutability of records and the complete decentralisation of data control. For instance, images can be stored outside the blockchain (to avoid overloading it with extensive data), while cryptographic hashes of these files are stored in the blockchain. This ensures that any change to an image will immediately be noticeable due to a hash mismatch and also provides trust among independent participants in the system.

The hybrid image storage model considered in this work aims to combine the advantages of the mentioned approaches. The idea is to create a multi-level storage system: critically important metadata and indexes are stored in a relational DBMS for fast search; extensive binary image data is distributed between a NoSQL database and an IPFS network for scalability and replication; data integrity and authenticity control are implemented through blockchain registration of hashes. Such an architecture ensures high-speed image access, storage reliability, and scalability as data volumes grow.

2. Related Works

The contemporary scientific literature presents a wide range of approaches to the architecture of image storage, especially in big data environments. The purpose of this section is to provide a comprehensive review of existing solutions and highlight key trends, challenges, and opportunities in this area. Traditional centralised and distributed database architectures, file systems, and decentralised storage such as IPFS (InterPlanetary File System), combined with blockchain technologies and hybrid image storage models, are actively studied. Thus, IPFS for storing the images and blockchain for storing corresponding metadata and hash identifiers is already considered a promising direction [1–3].

Image storage architecture is generally a well-established research area with many evolving approaches over the past decades. One of the earliest approaches involved using a centralised database where all images were stored in a single location. Despite simplicity and clarity, centralised storage proved to be poorly scalable and incapable of effectively handling large images. As a result, researchers turned to distributed storage, where images are stored across multiple nodes; such distribution increased scalability and storage performance but also caused new problems related to data consistency and availability [4].

Further studies show that the issue of organising image storage is being considered comprehensively, in conjunction with other related problems—data fragmentation, large-scale data processing, and distributed storage. For example, the works of Kimpel [5], Yarke [6], Blanco [7], and the researchers Kutsokrea, Bellatreche, and Song [8] analyse integrative approaches to building storage systems that simultaneously take these aspects into account. At the same time, Bulmakul et al. [9] and Kuchuk H.A. [10] focused on architectural solutions for distributed storage systems, proposing models to improve scalability and reliability. Popular big data processing platforms have also become the subject of separate studies: Pryimak and Akymyshyn [11] demonstrated the benefits of using the Hadoop ecosystem (HDFS and MapReduce) for creating scalable image storage. The issues of structuring data and optimising their processing for building integrated data spaces are discussed in the work of Shakhovska [12]. In general, recent architectural solutions in image storage often combine centralised and distributed technologies to balance ease of management and scalability. These approaches have proven effective in managing large volumes of images in various application areas.

While developing image storage systems, researchers have identified several key challenges. First, ensure the quality and integrity of image data: large datasets often contain noise, errors, and inconsistencies, which is why data cleaning and normalisation methods are proposed to improve information quality and the reliability of analytical results. Second, data security: since images often have a confidential nature, it is essential to prevent unauthorised access and leakage. To this end, encryption mechanisms and user access control are implemented. Third, availability and retrieval speed: authorised users must be able to obtain the required images, regardless of the quickly increasing storage volumes. To accelerate search in large repositories, researchers have developed effective indexing mechanisms and search

algorithms that significantly improve the speed and accuracy of image retrieval [13]. Thus, addressing the problems of quality, security, and accessibility of information is a priority in constructing modern image storage systems.

Active progress in big data technologies and the growing demand for scalable image storage in various domains has led to the emergence of novel approaches aimed at improving the performance, reliability, and security of storage systems. One illustrative work in this direction is the study by Su and Huang [14], who proposed a framework for large-scale image analytics based on the Hadoop platform. The authors emphasised the advantages of using the distributed computing platform Hadoop to manage large image datasets and demonstrated the effectiveness of their approach through a series of experiments. Similarly, the work by Zhao et al. [15] presents a new method of image retrieval using deep learning and convolutional neural networks (CNN). The results show that the proposed algorithm outperforms existing state-of-the-art content-based retrieval methods on several test datasets, demonstrating higher accuracy. In addition, several studies focus on efficient storage tasks; in particular, the issues of image compression and deduplication in large repositories are addressed in works [16, 17]. For example, the study by Wang et al. [18] proposes a new method of combined compression and deduplication, which integrates hashing and clustering to detect duplicate images. It has been shown that this approach provides a higher compression ratio and deduplication efficiency compared to existing methods.

Another important direction is the development of tools for distributed and parallel image processing within storage systems. Tong et al. [19] describe a distributed processing architecture based on the Apache Spark platform, enabling efficient processing of large images in a clustered environment. Similarly, the study by Zhang et al. [20] proposed a parallel approach to image recognition using GPU clusters, which provided significant acceleration of processing and improved accuracy compared to traditional solutions. Researchers also pay special attention to security and privacy issues in image storage. Wang et al. [21] proposed a secure image storage scheme based on homomorphic encryption and data obfuscation methods. Experimental evaluation showed that this approach provides high information protection during storage, ensuring image confidentiality. Similarly, Liu et al. [22] developed a secure system for storing and searching images using cryptographic methods. Zhou et al. [23] proposed an image-sharing scheme with privacy guarantees based on homomorphic encryption.

The progress achieved in research has led to the emergence of significantly more efficient and secure image storage architectures. However, many open questions and challenges still require the scientific community's attention. In particular, the lack of standardised approaches to constructing and managing repositories complicates the compatibility of different systems. There is also a need for further optimisation of storage and processing of large-scale image datasets, as both the volume of data and the requirements for processing speed continue to grow. The highlighted issues indicate promising directions for research in improving data compression methods, developing distributed computing technologies, and implementing modern mechanisms for protecting confidential information in image storage. Thus, the review of existing approaches confirms the relevance of further scientific efforts to create hybrid image storage models that combine the advantages of various technologies and can function effectively in big data environments.

The issue of storing large volumes of images has attracted significant attention from researchers, resulting in the development of various approaches and storage systems. Let us consider the main ones and their characteristics, which formed the basis for creating our hybrid model.

Relational Data Storage. Relational databases (SQL DBMS) are traditionally used for structured data and support the SQL query language with robust capabilities for filtering and joining tables. BLOB (Binary Large Object) types are usually used for storing images in SQL databases, or a path to the image in the file system is stored. At the same time, metadata is maintained in database tables. The advantage lies in supporting ACID properties (Atomicity, Consistency, Isolation, Durability), which ensures high data reliability. However, relational storages face scalability issues when dealing with large-scale image datasets. Vertical scaling (increasing the capacity of a single server) has physical limitations, and distributing data across multiple SQL servers (horizontal scaling) is a non-trivial task. In addition, large volumes of unstructured data may lead to degradation in SQL query performance.

NoSQL and Non-Relational Storage. An alternative to the relational approach is represented by various NoSQL systems: document-oriented databases (MongoDB, CouchDB), key-value stores (Redis, Riak), columnar databases (Cassandra, HBase), etc. They are designed with the principles of the CAP theorem in mind, often sacrificing consistency for the sake of availability and partition tolerance. NoSQL systems offer flexible data schemas and easy integration with horizontally scalable clusters for image storage. For example, in MongoDB, images can be stored as files in GridFS or as documents in a collection with metadata fields. The main advantages of NoSQL are its high scalability and performance for simple read/write operations, the lack of a fixed schema, and fault tolerance due to node-based distribution. At the same time, the lack of full transaction support and the possible absence of relational links between data complicates the assurance of consistency. The NoSQL approach is attractive for our task because it handles thousands of requests in parallel and stores terabytes of images while adding new nodes to the cluster. However, a simple NoSQL database may be insufficient for complex image searches based on various attributes or for guaranteeing data immutability.

Analyse the impact of the lack of full-fledged transactions (except for storing images and semi-structured metadata) in NoSQL (in particular MongoDB) in a hybrid system, which can seriously affect data consistency, especially in parallel or partially completed operations. Below is a detailed analysis of this issue in a hybrid decentralised system with IPFS, SQL, blockchain, and NoSQL.

While the hybrid architecture employs NoSQL storage for scalable image persistence, it is essential to consider its transactional limitations. Most NoSQL systems only guarantee atomicity at the single-document level. MongoDB introduced multi-document ACID transactions only in version 4.0 (with limitations: no support for sharded clusters, 16 MB size limit). Consequently, NoSQL databases traditionally follow an eventual consistency model, where updates propagate across nodes over time at the expense of immediate consistency. This can lead to the issues discussed above. For example, if storing an image or its CID in MongoDB fails, the relational database and blockchain might record metadata for an image that was never saved, causing a mismatch between metadata and content. Likewise, in an incomplete update, the file may be present in IPFS, but the corresponding CID in the blockchain or the metadata in SQL is not updated, resulting in clients seeing an old file version. Table 1 below summarises such consistency breakage scenarios.

Table 1. Impact on Image Metadata Consistency

Component	Effect of missing transactions in NoSQL
SQL (Relational DB)	Metadata tables may contain entries that do not correspond to any actual image file.
IPFS	The image file may exist in IPFS, but its metadata (CID) was never recorded or has been lost.
Blockchain	The blockchain may consider a file valid even though the corresponding data (CID) is absent in the NoSQL store (MongoDB).
Users	Users see a record for the image, but attempting to retrieve it yields a “file not found” error.

To prevent these inconsistencies, coordination mechanisms must be applied. For instance, the Saga pattern can break the multi-step operation (MongoDB → IPFS → SQL → Blockchain) into atomic steps with compensating actions upon failure. Asynchronous message queues (Kafka, RabbitMQ) can orchestrate these steps and enable retry on error. In MongoDB, setting an appropriate write concern (e.g. { w: "majority", j: true }) ensures that a document is written to a majority of replicas. An independent transaction log (e.g., in a separate database or Redis) can track each step’s status to reprocess failed operations. Without such measures, metadata and content can be desynchronised (SQL containing invalid records, blockchain giving false integrity confirmations), undermining user trust in the system. To avoid this, you need an architecture with control over the sequence of operations, a reliable event queue or middleware, and validation and logging mechanisms.

Distributed File Systems and IPFS. Another class of solutions involves storing images in a distributed file system or a peer-to-peer network. An example is HDFS (Hadoop Distributed File System), which splits files into blocks and stores them on cluster nodes with replication, providing high throughput for large files. IPFS (InterPlanetary File System) has recently gained popularity as a decentralised file storage protocol that uses content addressing. In IPFS, each file (or data block) corresponds to a unique cryptographic hash (CID), and network nodes exchange data based on these hashes. As a result, a file can be retrieved from any node that has it by referring to the corresponding CID, which makes the system resilient to the disappearance of individual nodes. For image storage, IPFS is attractive because it allows the load to be distributed among many participants in the network and eliminates a single point of failure (unlike a centralised server). At the same time, a drawback of IPFS is that access time depends on the network infrastructure – finding a node with the desired block and transferring data over the network introduces delays. As a rule, access to an individual image in an IPFS environment is slower than in a local database (on the order of hundreds of milliseconds or more versus tens of milliseconds in local systems). Moreover, IPFS does not provide means for complex search – to locate an image, one must know its CID (hash), and content-based or metadata-based search requires additional indexing layers.

Blockchain. Blockchain, as a distributed ledger of transactions, guarantees the immutability and verifiability of data recorded in blocks. In image storage, blockchain is not used to store the image files themselves (due to block size limitations and high recording costs) but rather to store checksums (hashes) or references to images stored outside the blockchain. This approach allows the system to verify, upon each access or download, that the image hash matches the entry in the blockchain, thereby detecting even the slightest attempts at tampering or data corruption. Well-known solutions based on blockchain and IPFS are used for protecting the copyright of content (by storing image hashes in a public blockchain, such as Ethereum, in the form of NFT tokens) and for creating censorship-resistant repositories (e.g., projects for storing archives in IPFS with blockchain anchoring). However, blockchain integration may introduce additional overhead – the time required to add a record to the blockchain (even a private one) adds latency when uploading a new image and reading from the blockchain may require additional queries. Therefore, this technology must be carefully considered and combined with other mechanisms to maintain performance.

Consider the potential drawbacks of blockchain integration, such as latency and scalability issues that can occur with larger datasets.

Integrating blockchain into a hybrid image storage system has many advantages (integrity, immutability, decentralisation). Still, it has significant drawbacks, especially when working with large amounts of data. Here's a detailed look at the main issues:

Potential disadvantages of integrating blockchain into a hybrid storage system (Fig. 1 and Table 2):

1. Latency. Blockchain (even a private one) has limited throughput due to consensus mechanisms (e.g., Proof of Work, Proof of Authority, etc.). The consequences are delays in writing an image hash to the blockchain (~seconds or more), delays in reading records (requires multiple requests to blockchain nodes), and the inability to "instantaneously"

confirm access or verify integrity in real time. It is especially critical if the system is used for video surveillance or real-time medical diagnostics, and many images are being downloaded or processed simultaneously.

2. Limited throughput. The blockchain can only process a limited number of transactions per second. For example, in Ethereum, 15-30 transactions/sec (without Layer 2), or in Hyperledger Fabric (private), up to hundreds of transactions/sec, but with complexity in scaling. With a heavy load, a "bottleneck" is created.

3. The accumulation of records and the growth of the size of the blockchain. Each hash record adds a new transaction to the blockchain. With the growing number of images, the volume of the blockchain is growing rapidly. Also, new nodes on the network must load the entire history, which slows down synchronisation. At the same time, the requirements for storage and network bandwidth are growing.

4. Limited transaction flexibility. The data on the blockchain cannot be changed or deleted. In error cases (for example, an incorrect hash or metadata is stored), a new transaction must be created for correction. Unfortunately, old information will remain forever (which may not be appropriate in certain jurisdictions, including GDPR).

5. Expenses/costs (in public networks). On public blockchains (e.g. Ethereum), each record requires a "gas fee", and when used en masse, this can be financially unprofitable. Even in private networks, the cost of infrastructure and maintenance (nodes, consensus mechanisms) is an additional burden.

6. Integration and support complexity. Blockchain infrastructure (nodes, smart contracts, and log auditing) and qualifications are required to maintain and securely update smart contracts. The complexity of testing and tracking bugs in the system increases.

Table 2. Possible Ways to Mitigate Deficiencies

Problem	Possible solution
Delay	Using Layer 2 solutions or batch hash logging
Bandwidth	Moving to a private blockchain with optimised consensus
Blockchain Growth	Archiving old data or partial replication
Costs	Gas-optimised smart contracts, or internal network
Integration complexity	Containerisation and automation (Docker, CI/CD)

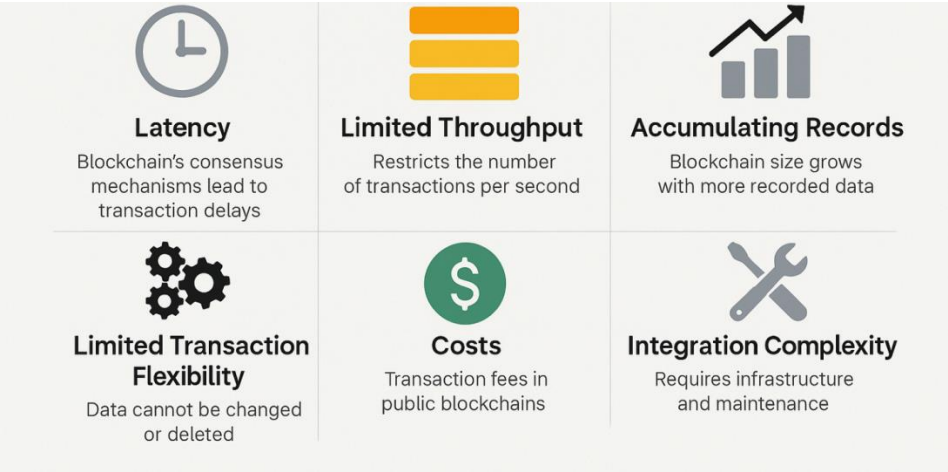

Fig. 1. Infographic of Potential Drawbacks of Integrating Blockchain into a Hybrid Storage System

The immutability of blockchain records, while ensuring data integrity and trust, also incurs performance costs as the system scales. Since each image's hash is stored as an immutable transaction, the blockchain size grows linearly with the number of images (e.g. 10 million images → 10 million transactions), which substantially increases scanning and synchronisation time (a new node must download the entire chain history). Verifying an image requires computing its hash and retrieving the corresponding blockchain entry. In a private network, a write transaction typically takes on the order of 200–500 ms and a read/verification about 100–300 ms (see Table 3). These delays accumulate under thousands of concurrent requests, causing write queues and second-long response times (Table 3). Immutability also means data cannot be modified: any correction (e.g. fixing a wrong hash) requires appending a new transaction, while the old record remains. This increases the number of records per image and complicates integrity checks. Frequent blockchain queries (via RPC/Web3) impose extra network load and higher latency than normal database reads. In summary, immutability guarantees security but reduces the throughput of write/read operations and complicates scaling. Various architectural strategies can be used to mitigate these issues, e.g., batching hashes off-chain with periodic anchor updates, layer-2 rollups, caching verification results, distributing load across multiple blockchain nodes, or maintaining specialised CID indexes (see Table 4).

Table 3. Practical Examples of Delays

Operation	Typical delay (private network)	Scalability issue
Write a hash to the blockchain	~200–500 ms	At >1000 transactions, a write queue forms, causing a backlog
Verify hash (CID lookup)	~100–300 ms	Under >10^4 parallel reads, latency can grow to seconds
Full sync of a new node	Hours–days	Must download entire chain history (requires archival nodes)

Table 4 Architectural Solutions

Method	Description
Off-chain batch logging	Store individual image hashes in a local database, periodically anchoring their aggregated hash to the blockchain.
Layer-2 solutions (e.g. zk-Rollups)	Bundle thousands of transactions off-chain and publish only aggregated proofs to the blockchain.
Caching verification results	Temporarily cache recent hash verification outcomes to reduce repeated blockchain lookups.
Multi-node blockchain architecture	Distribute verification queries across multiple mirror nodes.
Dedicated CID indexing	Use a fast lookup index or map table for image CIDs to speed up integrity checks.

Although the immutability of records in the blockchain guarantees security, with a large amount of data, it reduces the system's performance during recording and verification. It complicates scaling at the level of access and synchronisation. It also requires architectural optimisations, such as off-chain aggregation, caching, and Layer-2 or separate proof-of-integrity mechanisms.

Some studies attempt to combine different types of storage systems to improve the overall characteristics of the system. In particular, polyglot persistence involves using multiple storage types for various data types within a single system. For images, this may mean storing metadata (descriptions, tags, properties) in an SQL database for convenient querying, the binary image files in NoSQL or file systems for scalability, and backup copies or hashes in a blockchain for integrity control. Separately, methods for reducing data size are also considered, such as compressing images before storage and deduplication to avoid storing identical photos multiple times. In our work, all these ideas are combined into a single architecture. Based on the literature review, none of the reviewed solutions fully satisfy the needs of image repositories under big data conditions on their own. This necessitated the development of a hybrid model that synergistically utilises the advantages of each approach.

Table 5 summarises the main characteristics of different storage types and the properties of their combination in the form of a hybrid system.

Table 5. Comparative Efficiency of Image Storage Systems

Criterion	SQL (Relational)	NoSQL (Non-relational)	IPFS (Decentralised)	Hybrid Model (SQL + NoSQL + IPFS)
Image access time	120–150 ms	80–100 ms	180–250 ms (network-dependent)	60 ms (optimised)
Storage reliability	High (ACID)	High (depends on DB)	Medium (requires data replication)	High (distributed + local)
Scalability	Limited (vertical)	High (horizontal)	High (distributed network)	Very high (combined)
Decentralisation support	None	Limited	Full	Full (via IPFS + blockchain)
Search complexity	SQL queries (flexible)	Limited	Requires additional indexes (CID only)	Flexible (SQL + indexes for NoSQL/IPFS)

The table shows that the hybrid approach offers significantly lower image access time (thanks to fast local indexes and hashes) while maintaining high reliability (through replication in a distributed environment) and virtually unlimited scalability. Complete decentralisation is also achieved through the integration of IPFS and blockchain. The following sections of this paper provide a detailed description of the implementation of this model and the proposed methods that ensure the achievement of these performance indicators.

The following is an analytical analysis of the potential challenges and behaviours of a hybrid system in the short term, with modelling or predicting its behaviour under long-term use or exponential growth of data – which is critical for practical implementation, especially in industries with large archiving (medicine, satellite imagery, video surveillance, etc.), with possible solutions. Functioning of the system with long-term use or exponential growth of data (Table 6):

1. IPFS (Distributed Storage). Problems with growth manifest in the massive accumulation of CID; that is, each new image is a new CID that requires careful management. There are also problems with replication, because nodes may not have the necessary replicas, which leads to slow access to "old" files. In parallel, there are problems with cleaning due to the lack of a "built-in" mechanism for recycling or archiving data. Possible consequences include reduced access speeds, increased load on network infrastructure, and more difficult metadata management. The solutions are using IPFS Cluster for pinning management, introducing archive nodes to store old CIDs, and automated TTL (time-to-live) images with labelling.

2. NoSQL (MongoDB/GridFS). Problems with long-term scaling are database bloat due to long-term storage of large objects (images, vectors), slowing down indexes if sharding or TTL policies are not used, and limiting IO on nodes in case of overload. Solutions include the implementation of horizontal sharding, archiving of low-demand images for "cold" storage (S3, Glacier), and regular reindexing and cleaning.

3. Blockchain (recording hashes). The problems with the accumulation of records are the expansion of the chain volume: each image hash = transaction, the decrease in the performance of smart contracts with a large number of calls, and the difficulty of synchronising new nodes - the whole story is challenging to pull up. The solutions are archived blockchain nodes, the transition to Layer-2 (for example, zk-Rollups, Validium) and batch registration (batching) of hashes or off-chain logging with periodic verification.

4. Elasticsearch/vector search. The problems are the growth of the index size → the need for more and more RAM, and a drop in accuracy with a large volume – vector databases do not scale well without optimisation. The solutions are the transition to FAISS or Milvus – systems created for scalable vector search, hierarchical data clustering (by date, region, image type) and index segmentation + load balancing by nodes.

5. Metadata and accessibility. Potential limitations include difficulties with quick access to very "old" images and the complications of controlled access and changing policies without changing smart contracts. The solution is the rotation of access keys through proxy encryption, the distribution of data into "active" and "archived" layers with different SLAs, and the introduction of second-level metadata – fast indexes for archives.

Table 6. Predicted System Behaviour Without Optimisation

Year of use	Expected problems
1-st year	High speed, stable operation
2–3 years	MongoDB congestion, CID growth, decreased indexing rate
4–5 years	Search slowdowns, replication issues in IPFS, and blockchain growth
5+ years	The complexity of integrating new nodes, the need for redistribution of indexes and archives

A hybrid system is theoretically scalable, but an active data management policy is required to avoid performance degradation. The key to long-term success is implementing sharding, archiving, segmentation, and specialised indexing and storage tools.

3. Methods and Methodology

Architecture of the Hybrid Image Storage System. A conceptual architecture of an image storage system is proposed, combining the advantages of various data storage systems. In particular, the hybrid model integrates a relational database, a non-relational (document-oriented) database, a distributed file storage system, and machine learning and blockchain modules. This approach enables high-speed data access, scalability, reliability, and security for image storage in big data environments. The main components of the system include:

- Relational Database (SQL) stores structured metadata and ensures data integrity (e.g., user information or image attributes).
- Document-Oriented NoSQL Database (MongoDB): used to store the images themselves and semi-structured data. It supports flexible storage schema and horizontal data distribution for scalability.
- Vector Indexing / Search Engine (Elasticsearch): responsible for indexing image descriptors (feature vectors) and implementing high-speed content-based image retrieval.

Deep Learning Module (ResNet50): a convolutional neural network that extracts image features. The pre-trained model generates a vector representation of each image for further indexing.

- Distributed File Storage (IPFS) is a decentralised peer-to-peer file storage system storing images as content-addressed blocks. Each file is assigned a unique hash later used for access and integrity verification.
- Blockchain Module (Ethereum Smart Contract): responsible for registering image hash identifiers and access transactions, ensuring immutability of records and access control. The smart contract automates the verification of user access rights to images and stores an operation log.

Interaction between components follows the principles of microservice architecture. A client application submits new images to the storage via an API, after which the image passes through a preprocessing module, is indexed, and is distributed among the corresponding storage. User search queries also arrive via the API and are processed through module interaction: first, the ML module transforms the query (e.g., a reference image or description) into a feature vector, then a search is performed across the indexes in the databases, and IPFS and the results are aggregated to form the final response. This integrated multi-component architecture enables efficient management of large volumes of

heterogeneous image data, ensuring a balance between access speed (through indexing and caching), storage flexibility (through NoSQL and IPFS), and security (through blockchain and encryption).

The system is designed for high availability. For IPFS, we use an IPFS Cluster with multiple pinned replicas of each image (recommend ≥3 copies) so that if any node fails, others can serve the data. MongoDB is deployed as a 3-node replica set with automatic failover: if the primary node dies, a secondary is promoted, and write/read concerns guarantee durability. The relational database uses streaming replication (e.g. PostgreSQL with repmgr) and HAProxy for automatic failover. Similarly, the blockchain component can have mirror nodes and queue pending transactions locally during outages. The search/indexing service (Elasticsearch or FAISS) runs in a clustered, containerised setup with health checks and automatic restarts; recent query results can be cached in Redis to tolerate short downtimes.

Let's describe the specifics of the interaction of relational, NoSQL, and blockchain components during the system's operation. Here is a detailed explanation of how this interaction is performed in practice.

When a user uploads an image, the client (user application) sends it via an API to the system. The image undergoes preprocessing (resizing, filtering, etc.), is compressed (e.g., into WebP or JPEG), and then encrypted using a symmetric algorithm (AES-256). The encrypted image is saved in a NoSQL document database (MongoDB) and semi-structured metadata (file type, resolution, filename, etc.). In contrast, critical structured metadata (user ID, upload date, category, tags, access rights) is stored in a relational DBMS (PostgreSQL/MySQL) to enable complex queries and ensure ACID properties. Concurrently, the system uploads the image to the IPFS distributed file storage, obtaining a unique content identifier (CID, a cryptographic hash) for the file. This CID is then recorded in a blockchain (e.g., Ethereum or Hyperledger) via a smart contract. The blockchain is an immutable registry – it logs image hashes, user access rights, and access events (such as view requests), ensuring data immutability and providing an auditable history. The architecture achieves reliability and scalability by storing copies in MongoDB and IPFS. The NoSQL database allows flexible horizontal scaling of extensive image data, and IPFS provides decentralised replication across multiple nodes.

When performing a search, the user's query (for example, a reference image or search criteria) is processed through the ML module (ResNet50), which extracts a feature vector from the input image. This vector is sent to the Elasticsearch search engine, which has indexed the feature vectors of all images in the repository. Elasticsearch returns a set of matching image records (by CID). For each candidate CID, the system then verifies integrity and access – it checks the blockchain record to compare the stored hash with the CID of the queried file, guaranteeing detection of tampering or corruption. If the smart contract confirms the user's permission, the encrypted image is retrieved from IPFS and decrypted with the provided key. Finally, the original image is returned to the user. This integrated content-based search process – indexing in Elasticsearch and secure retrieval from IPFS/SQL/NoSQL – enables efficient handling of extensive, heterogeneous image collections, balancing fast access and flexibility with security.

The system's security and reliability mechanisms are based on multiple layers. Symmetric encryption (AES-256) on the client side ensures confidentiality (only those with the key can decrypt the image), protecting the content even if IPFS nodes are compromised. Hashing and blockchain guarantee integrity and traceability: any change to an image alters its hash, which would no longer match the blockchain record, thus revealing unauthorised modifications. Relational and non-relational databases provide structure and scalability. The SQL DB ensures complex queries and transactional metadata consistency, while the NoSQL DB handles high-throughput storage of large image files. All components interact as microservices, each fulfilling a specialised role. The resulting hybrid architecture thus achieves an optimal balance of performance, scalability, and security for big-data image storage and retrieval, as confirmed by our theoretical analysis.

Generalised data flow scheme (Fig. 2):

1. Client → API → (Image Transfer)
2. API → Preprocessing + Compression + Encryption
3. Storage:
   - IPFS → image (obtaining CID)
   - CID → Blockchain (Ethereum)
   - Images → MongoDB
   - SQL → metadata
4. Search:
   - Request processing → ML (ResNet50)
   - Search in Elasticsearch
   - Obtaining CID → Blockchain Verification → Accessing IPFS
   - Access allowed? → Decryption → Reply to User

This approach provides distribution, speed, flexibility, reliability and security in one holistic solution, where each technology fulfils its specialised role and complements the others (Fig. 3).

Let's analyse how data consistency is ensured in a hybrid system, primarily when it operates in a distributed (decentralised) environment with components such as SQL, NoSQL, IPFS, and blockchain. Here's a detailed technical description of how it works to avoid data out of sync, loss, or duplication. Consistency means that metadata, content

(file), control hashes and access rights correspond to each component, all users see the same version of the data, and if there is a change or failure, the system knows what and where to update or roll back.

Ensuring consistency in a hybrid system operating in a decentralised environment (Table 7-8):

1. Blockchain as a Source of Truth. The CID (Content ID) image from IPFS is stored on the blockchain. It ensures that even if the image is replaced in IPFS, the hash will not match – that is, a consistency violation will be detected. Smart contracts can also keep an access log that records who accessed the image, whether it was updated, and whether the checksum matched. It ensures that the immutability of hashes → control of consistency between IPFS ↔ metadata ↔ access rights.

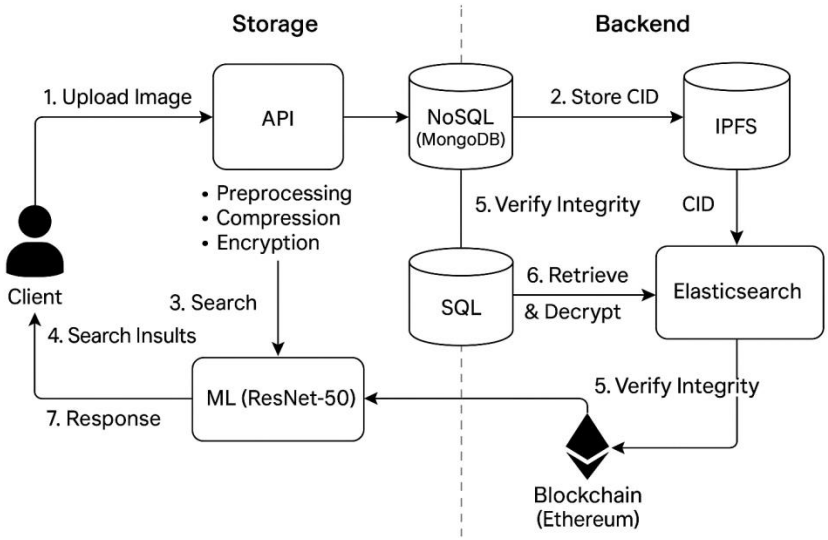

Fig. 2. Generalised diagram of the data flow of the interaction of relational, NoSQL and blockchain components during the system's operation.

2. NoSQL/SQL for metadata duplication and indexing. SQL (e.g., PostgreSQL) contains structured metadata for user ID, upload date, and category. NoSQL (for example, MongoDB) duplicates part of the metadata in JSON format + the file itself. It is essential that each time data is written/updated, records in SQL and MongoDB are updated at the same time, and changes are checked through transactions or two-phase commits (if possible). The consistency mechanism uses a message queue (e.g., RabbitMQ or Kafka) that captures the event upload -> image -> CID -> update SQL/MongoDB. In a failure, one part (e.g. MongoDB) can recover the data by re-querying the queue.

3. Weak consistency in IPFS – solved through access control and pinning. IPFS does not guarantee instant consistency: a node may not yet have a copy of the image. Therefore, IPFS Cluster is used, which forcibly "pins" the image to the designated nodes and uses the CID registry in the blockchain/SQL to check if the image is still up-to-date. If the image is missing → the system checks the CID on the blockchain, determines which nodes should have kept the copy, and starts the re-replication process or reports the loss.

4. Integrity and Synchronisation Strategy. Validation processes (scheduler) are regularly run to check whether the CID in SQL matches the CID in MongoDB/IPFS. Hash matching checks between IPFS ↔ blockchain ↔ local data, and non-compliance logging for the admin are also triggered: cron → validator.py → check(hash in chain vs. hash in DB vs. IPFS) → log.

Table 7. Consistency Models

Model	Where it is applied	Peculiarity
Strong Consistency	SQL + Transactions	Guaranteed consistency immediately after recording
Eventual Consistency	IPFS, MongoDB	Data may sync with latency
Causal Consistency	Blockchain → Access	The access right depends on the history of events

Table 8. Key Mechanisms for Maintaining Consistency

Component	Consistency mechanism
SQL ↔ MongoDB	Biphasic commit/event queues (Kafka, RabbitMQ)
IPFS	CID + replication via IPFS Cluster
IPFS ↔ Blockchain	Hashes and CID fixation in a smart contract
Client ↔ system	Atomic API requests with CID confirmation
General system	Regular integrity and logging checks

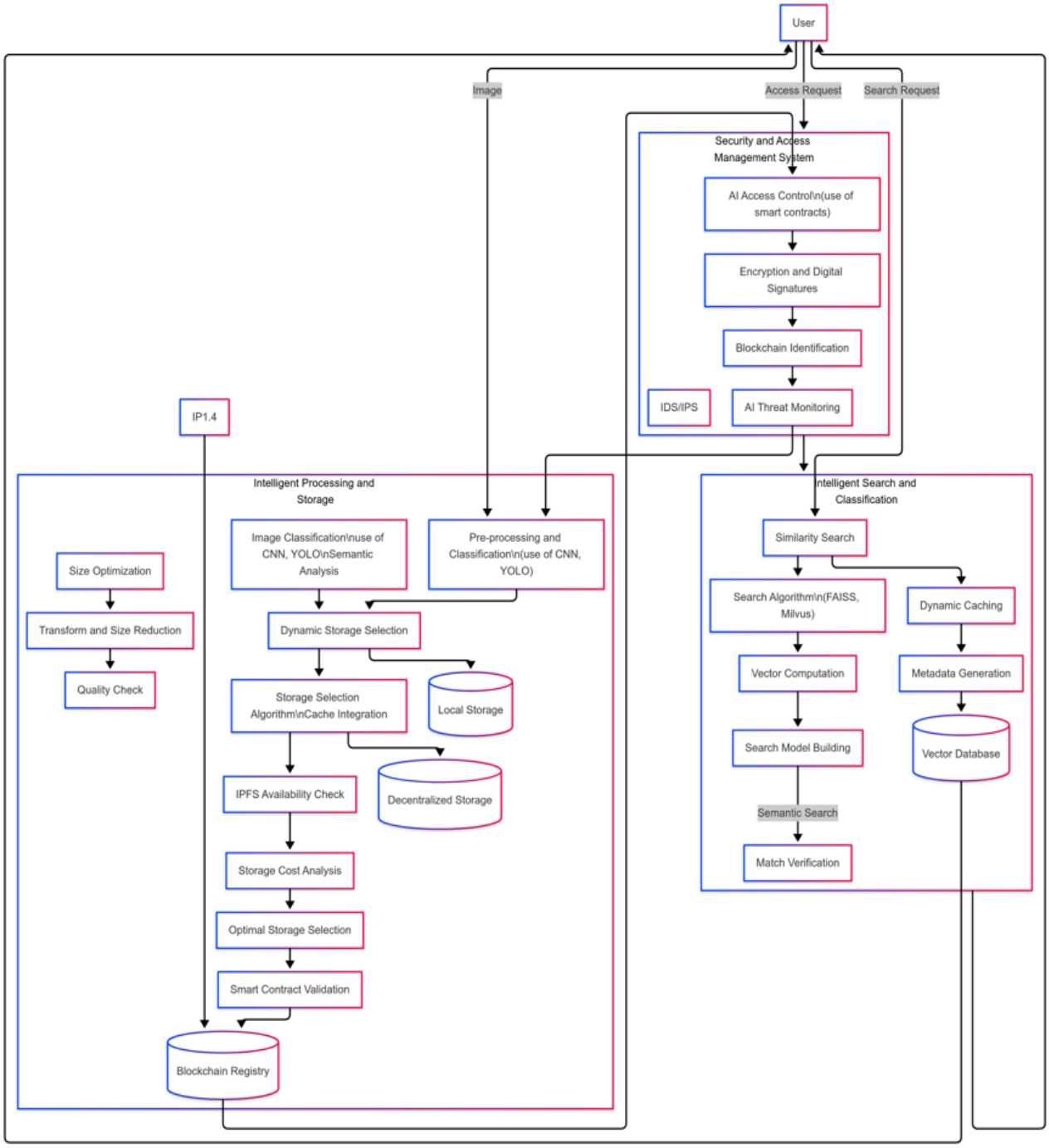

Fig. 3. Flowchart of the conceptual architecture of the image storage system.

In a decentralised environment, a hybrid system ensures consistency through controlled CID retention (IPFS ↔ blockchain), double-entry metadata (SQL/NoSQL), scheduled checks and audits, and event queues and transaction recovery in case of failure.

Let's describe the system's behaviour in the event of failures, which is a key factor in the reliability of any large-scale distributed system. A hybrid model that combines SQL, NoSQL, IPFS, blockchain, vector search, and encryption requires carefully crafted fault tolerance mechanisms to ensure high availability and recovery. Below is a detailed description of the main failure scenarios and how a hybrid system can (or should) deal with them:

Typical failure scenarios and processing mechanisms in a hybrid system (Table 9):

1. Failure of the IPFS node (where the images are stored). One or more IPFS nodes where the encrypted images are stored are inaccessible (due to network failure, updates, or attacks). The consequences are that the CID exists in the metadata, but the image is temporarily unavailable, and also, if the node is the only one, the file is lost. The processing mechanisms are IPFS Cluster (the system supports multiple copies of the file (pinning) on different nodes), automatic replication (if a node is missing, other nodes fill in the gaps), and Fallback CID (if the primary mirror does not work, access to an alternate one). The recommendation is to have at least 3 replicas of critical images geographically.

2. MongoDB node failure (NoSQL). One of the nodes in the MongoDB cluster fails. The consequences are that it is possible to lose access to metadata or cached images and break consistency if the node was primary. The processing mechanisms are ReplicaSet in MongoDB, which allows automatic failover – another node becomes primary. Write concern ensures that the write only occurs after confirmation from multiple nodes. Read concern ensures that only reads occur from synchronised copies. The recommendation is to use at least 3 nodes (1 primary, 2 secondary).

3. SQL server failure (PostgreSQL / MySQL). The central SQL server is not available. The consequences are the inability to search or filter images by structured metadata and the potential loss of transactions (if WAL/replica is not used). The processing mechanisms are Streaming replication: a hot copy of the database, a failover manager (e.g. repmgr for PostgreSQL) and an external cache (e.g. Redis) to serve queries from memory temporarily. A recommendation is to organise an SQL + HAProxy cluster for balancing.

4. Failures in the blockchain component. There is no communication with the blockchain node, or there has been an attack/network congestion. The consequences are that it is possible to confirm the CID or verify the integrity of the image, as well as delays in recording transactions. The processing mechanisms are the local transaction cache, stored in the queue until the node is restored, and reading from replicated nodes. It is possible to have several mirror nodes on the blockchain and off-chain logic: pre-checking hashes and revalidation after stabilising the network. A recommendation is to separate critical transactions (for example, audit log) and record them in Layer-2 or a separate database.

5. Failure of the neural network module or Elasticsearch (vector search). The scenario is the downfall of the service that handles the search for similar images. The consequences are the inability to process searches based on example or visual similarity, and users may see empty results. The processing mechanisms are a backup index or stored previous search results, monitoring with auto-restart (Docker + healthchecks, SupervisorD), and horizontal scaling of Elasticsearch/FAISS with balancing. The recommendation is to use containerisation with automatic restart + LRU caching of results.

Table 9. Cross-Cutting Solutions to Ensure System Resilience

Mechanism	What provides
Containerisation (Docker/K8S)	Easy recovery of modules after failures
Monitoring (Prometheus + Grafana)	Detecting failures before they become critical
Load Balancing (HAProxy, NGINX)	Load distribution between nodes
Message Queues (Kafka, RabbitMQ)	Handling events in case of partial failures or deferred writes
Backups & Snapshots	Recovery from the complete loss of a node

To ensure fault tolerance in a hybrid system, it is essential to integrate replication at the IPFS, MongoDB and SQL levels, use fault-tolerant middleware (queues, health checks, autoscaling), implement automatic recovery logic through caches, backup nodes and control hashes, as well as ensure component independence through service architecture.

Image Indexing and Retrieval Methodology. An optimised image indexing method is used to ensure a fast content-based search. The algorithm's core involves extracting and storing image features as fixed-size vectors (Fig. 4). The ResNet50 deep learning model is applied, transforming each image into a feature vector (typically size 128, 256, or 512). ResNet50 is a 50-layer convolutional neural network pre-trained on a large image dataset with residual connections. A pre-trained network enables extracting high-level features without additional training, reducing computational costs. The ResNet50 network provides high accuracy in identifying distinctive image features, making it a justified choice for obtaining high-quality descriptors for search purposes.

The diagram in Fig. 4 illustrates the sequential stages of data processing: first, a new image is stored in the storage system (MongoDB/IPFS), then its features are extracted using the ResNet50 model, forming a feature vector that is added to the index (Elasticsearch). Simultaneously, the file's hash is recorded in the blockchain to ensure data integrity. A user's search query (for example, an image to find similar ones) undergoes the same steps: its features are extracted via ResNet50, a query vector is generated and compared with the vectors already indexed in the database. As a result, a list of existing images ranked by similarity to the query is returned. The cosine similarity metric is used to compare feature vectors – it determines the angle between two vectors in feature space. The cosine similarity measure is calculated using the formula:

$$similarity = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}, \quad (1)$$

where A and B are the feature vectors of two images of dimensionality n.

Cosine similarity ranges from -1 to 1, where 1 indicates that the vectors are perfectly aligned (maximum similarity), 0 means they are orthogonal (no similarity), and -1 indicates they are exactly opposite. Cosine similarity values are typically considered within the range from 0 to 1 for finding similar images.

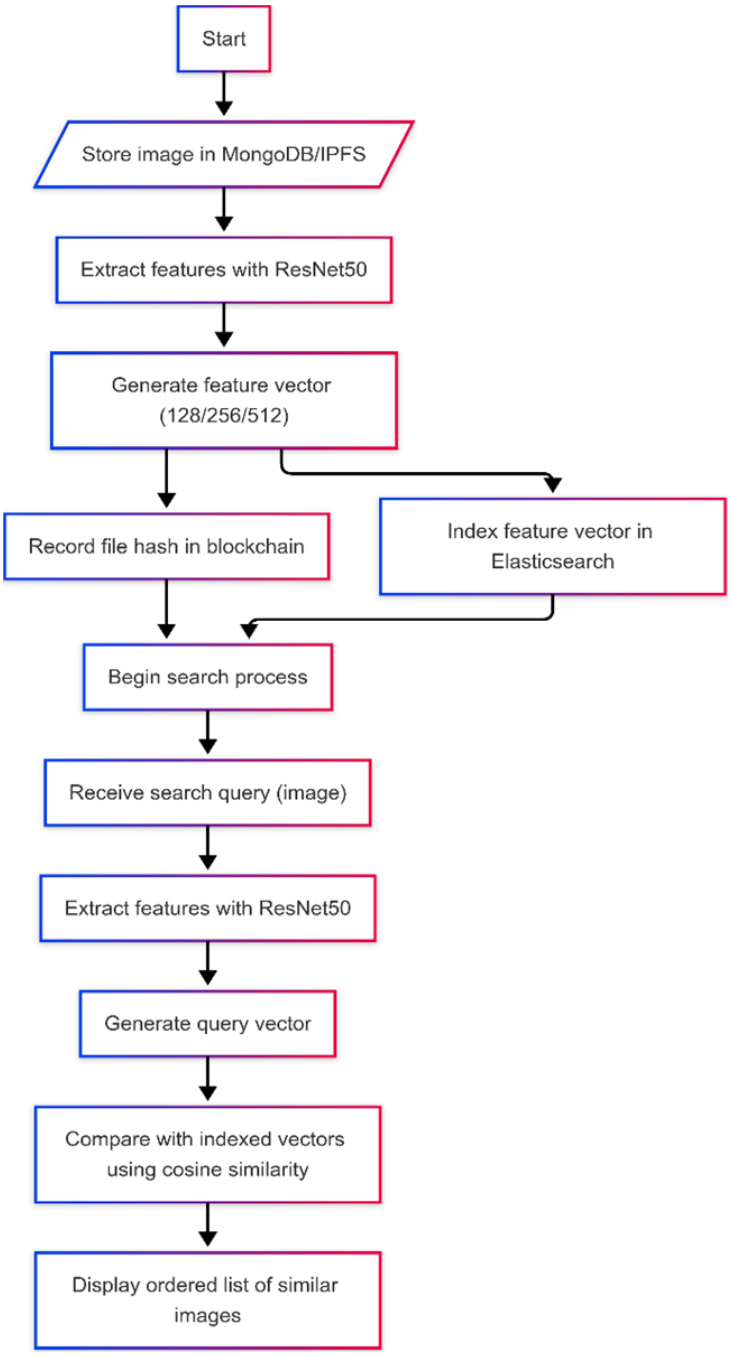

Fig. 4. Algorithm for image indexing and retrieval in the hybrid system

Special attention is paid to the index structure to achieve high search performance in the developed system. The hierarchical index combines several sub-indexes, each corresponding to a specific component of the hybrid storage system: the relational component (denoted as I_R), the document-oriented component (I_D), and the object/file component (I_O). Collectively, the hybrid index is denoted as $I_h = (I_R, I_D, I_O)$. For indexing metadata in the relational DBMS and documents in the NoSQL database, balanced trees (B-trees) are used, which provide logarithmic search complexity. In contrast, for the object storage (files in IPFS), hash structures are applied, which, with uniform hash value distribution, offer an average-case search complexity of $O(1)$ (in the worst case, $O(n)$ in the presence of collisions). A key principle is the use of hashing for fast object access [24]. An ideal hash function $h(k)$ is defined as:

$$h(k) = k \bmod m, \tag{2}$$

where k is the input data (e.g., object identifier or key), and m is the hash table size.

Several criteria evaluate the effectiveness of a hash function: uniformity of value distribution, computation speed, and minimisation of collisions.

The probability of collisions $P(n, m)$ for n elements in a hash table of size m can be estimated as:

$$P\ (n, m)\ =\ 1\ -\ e^{-\frac{n(n-1)}{2m}} \tag{3}$$

B-trees support ordered data structures and logarithmic search time for the relational and document components.

Given the tripartite nature of the hybrid model, the optimised index has a hierarchical structure. Represented as I_h, it can be defined as:

$$I_x = (I_R, I_D, I_O), \tag{4}$$

where I_R is the index associated with the relational data component, I_D refers to the document-based component, and I_O encompasses the object storage component. Each sub-index operates autonomously within its environment; however, the results from all components are merged during the processing of a search query. Thus, the overall complexity of a search operation in the hybrid index approaches $O(logn)$ due to the dominant cost of searching in B-trees; access to distributed objects via the hash index typically adds a constant or negligible value to the search time.

Algorithmic Construction of the Optimised Index

Algorithm 1: Construction of a Hierarchically Optimised Index
Input: Hybrid storage H = (R, D, O)
Output: Hierarchical index Ih
BEGIN
  I_R = BUILD_INDEX(R)
  I_D = BUILD_INDEX(D)
  I_O = BUILD_OBJECT_INDEX(O)
  I_h = (I_R, I_D, I_O)
  RETURN I_h
END

Due to their structured nature, the function BUILD_INDEX utilises a B-tree–based methodology for both the relational and document components. In contrast, BUILD_OBJECT_INDEX employs a hash-based approach, leveraging the flat namespace of the object storage. The proposed index structure enables efficient search by attributes (via relational/document indexes) and by content (via vector index), seamlessly covering the heterogeneous components of the storage system.

Distributed Storage, Encryption, and Blockchain Module. The system integrates the distributed file storage IPFS with blockchain technology to ensure reliable storage of large volumes of images. IPFS (InterPlanetary File System) enables decentralised file storage across a network of nodes. When added to IPFS, each file is assigned a unique cryptographic hash identifier (CID). The blockchain module (based on the Ethereum platform) records these hashes and file operations in an immutable ledger.

As shown schematically in Fig. 5, when a client uploads a file to IPFS, its hash is transmitted to a blockchain smart contract. Other IPFS nodes can retrieve this file using its CID, while the recorded hash in the blockchain verifies authenticity and integrity. In this way, the blockchain acts as a global content registry – any image stored in the distributed repository is registered through a transaction containing its hash. This guarantees that undetected tampering or deletion is impossible – any change to the file results in a different hash and a mismatch with the blockchain record. Access transparency is also ensured – all operations (such as uploading a new image or requesting access) can be logged in the blockchain, which is essential for auditing and tracking data usage (e.g., protecting image copyrights).

We treat the blockchain as the source of truth - each image's IPFS content ID (CID) is recorded on-chain, so if an image is altered in IPFS, the hash mismatch is immediately detected. Metadata is stored in both the SQL database and MongoDB; updates to each must be performed atomically (e.g. via transactional messages) to avoid desynchronisation. Since IPFS is eventually consistent, we use an IPFS Cluster to pin images on specific nodes and use the on-chain CID registry to verify that replicas exist; if a node is missing data, the system can re-replicate or alert administrators. Periodic integrity checks (e.g. comparing stored CIDs and hashes across components) ensure end-to-end consistency (Fig. 5).

Additional data encryption mechanisms have been implemented to ensure the confidentiality of images in a distributed environment. Before storing an image file in IPFS, it is encrypted using a symmetric algorithm (e.g., AES). Symmetric encryption was chosen due to its high speed when processing large volumes of data and the simplicity of key management. The secret key is stored either by the data owner or in a secure vault, while only the hash of the key or other metadata required for access verification may be recorded in the blockchain. Thus, even if an external party accesses an encrypted file in IPFS, the image cannot be viewed without the decryption key. Combining symmetric encryption with blockchain-based operation auditing forms a hybrid security approach – data is protected cryptographically (through encryption) and at the level of logging and access control (via smart contracts). The smart contract in the system automates access control. For example, a function can be implemented to provide the decryption

key only to a user with specific rights or upon completion of a condition (such as a payment transaction). This approach prevents unauthorised use of images – even if a file is distributed across IPFS nodes, access to its plaintext content is controlled via the blockchain.

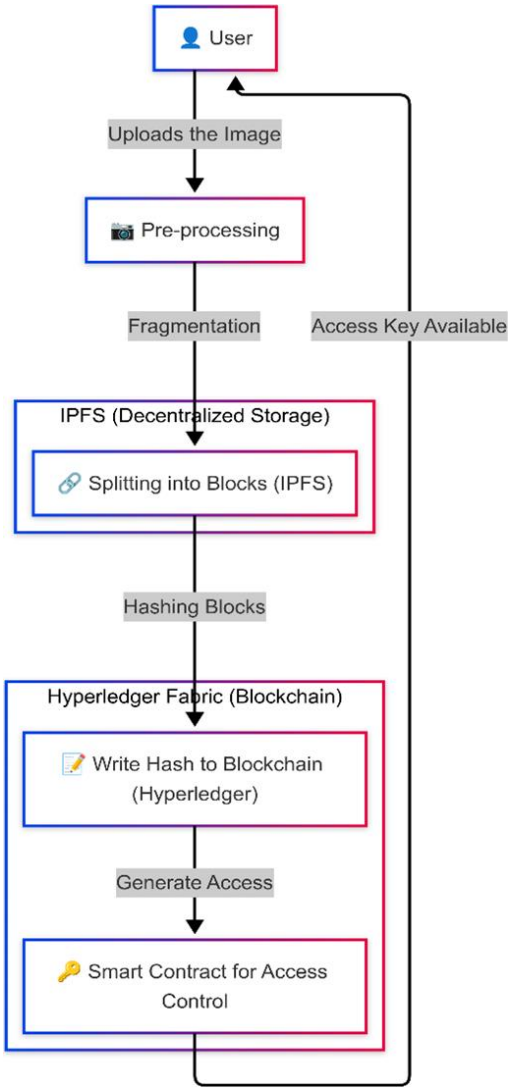

Fig. 5. Interaction between IPFS and the blockchain network

Another important aspect is the compression of images before storage. Since the distributed network has limited bandwidth and access time, reducing the file size of images is critical for efficiency. An approach is implemented within the system that selects and applies the optimal compression algorithm depending on the image type and quality requirements. Both lossless compression methods (PNG, TIFF with LZW algorithm, GIF) and lossy compression methods (JPEG based on discrete cosine transform, WEBP, etc.) are supported. For example, the JPEG algorithm transforms the image into the frequency domain by discarding visually insignificant information. In the first step of JPEG compression, the image is divided into 8×8 pixel blocks – the number of such blocks can be calculated using the formula:

$$N = \frac{W \cdot H}{64}, \tag{5}$$

where N is the total number of blocks, W is the image's width, and H is the height of the image in pixels.

The next steps include the transformation of the colour model (from RGB to YCbCr), which separates luminance and chrominance components, application of the Discrete Cosine Transform (DCT) to each 8×8 block, quantisation of the DCT coefficients, and entropy coding (e.g., using the Huffman algorithm). As a result, JPEG significantly reduces data size by discarding a small amount of perceptually insignificant information. The choice of compression algorithm in the system is based on the characteristics of the images – for photographs, JPEG or WEBP is recommended (achieving significant compression with acceptable quality loss). In contrast, for schematic or medical images, PNG or TIFF is preferable (to preserve all details without loss). Before an image is added to IPFS, the system compresses it using the selected method, thereby reducing network load during transfer and saving disk space on the nodes.

The proposed materials and methods – including the multi-component storage architecture, optimised indexing and search algorithms, and data protection mechanisms – jointly ensure the efficient functioning of the hybrid image storage model. Through integrating technologies (SQL/NoSQL, IPFS, blockchain, and deep learning), the system can scale to accommodate growing data volumes, perform rapid content-based image retrieval, and guarantee information security and integrity. Thus, a complete data management cycle is implemented, from image upload and storage to indexing, search, and secure access, confirming the practical applicability of the developed hybrid model in big data environments.

4. Experiments, Results and Discussions

Based on the developed model, a system prototype was created for searching for similar images using deep learning methods. The software architecture (Fig. 6) combines a document-oriented MongoDB database for storing images and metadata with the ElasticSearch search engine for fast indexing and retrieval based on image feature vectors. A pre-trained ResNet50 model extracts vector features, providing high accuracy and efficiency without long training sessions. This significantly optimises data access: similar image retrieval is performed using high-speed and precision vector features, confirming the feasibility of using a hybrid approach with ML for large volumes of visual data.

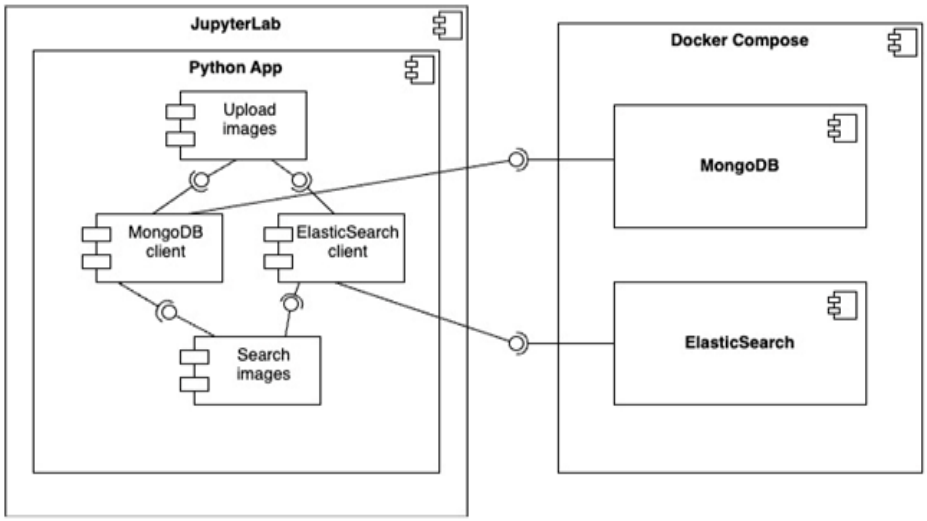

Fig. 6. Component diagram of the software implementation of the study.

A dataset of 5,400 JPEG images (~90 animal classes) from the Kaggle platform was used for experimental evaluation. The system performed similar image searches across varying sample sizes (10, 20, 50, 100, 500, 1000, and 5400 images). The accuracy of the search was assessed visually: the retrieved results included images belonging to the same animal class as the reference image, confirming the correctness of the algorithm's operation. Fig. 7 presents examples of the obtained search results – all retrieved images correspond to the same species, demonstrating the system's high accuracy.

For completeness, we compared standard image formats. JPEG (lossy) typically achieves ~1.5:1 compression (30–50% size reduction) with average SSIM ≈0.89, PNG (lossless) ≈1.1:1 (10–20% reduction, SSIM≈1.0), whereas WebP (lossy) achieves ~2.1:1 (≈52% reduction) with SSIM ≈0.90 on our datasets. These results confirm WebP's superior compression efficiency for this application (Table 10). We will carry out an in-depth analysis of alternative compression methods (Table 11), as well as an assessment of image quality according to metrics other than PSNR (Peak Signal-to-Noise Ratio). Below is a comparative study of compression formats and quality metrics, which allows you to assess the effectiveness of WebP more fully compared to other approaches.

Fig. 7. Search results produced by the system.

Table 10. Comparison of Image Compression Methods

Format	Compression type	Average size reduction	Transparency support	Animation support	Decoding speed	Medium SSIM
JPEG	Lossy	~30–50%	No	No	High	~0.89
PNG	Lossless	~10–25%	Yes	No	Low	1.0 (lossless)
WebP	Lossy / Lossless	~40–60% (lossy), ~20% (lossless)	Yes	Yes	Average	~0.93
AVIF	With and without losses	~50–70% (lossy)	Yes	Yes	Low	~0.96
JPEG XL	Lossy / Lossless	~50–70%	Yes	Yes	Medium-low	~0.95
TIFF (LZW)	Lossless	~15–25%	Yes	No	Low	1.0

Table 11. Quality Assessment Metrics (except PSNR)

Metric	What measures	Advantages	Disadvantages
SSIM (Structural Similarity Index)	Structural similarity (0 to 1)	Takes into account a person's perception of the structure	Does not take into account colour accuracy
MS-SSIM (Multi-Scale SSIM)	SSIM at different scales	More resistant to changes of scale	Requires more calculations
VIF (Visual Information Fidelity)	Information loss is measured through the vision filter	The closest to human perception	Very computationally costly
LPIPS (Learned Perceptual Image Patch Similarity)	Germination through a neural network	Modern, close to perception	Requires access to the ML framework
FSIM (Feature Similarity Index)	Takes into account the phases and amplitudes of the image	Accurate Model of Vision Filters	Not supported everywhere

WebP has a good ratio between compression and quality, significantly outperforming JPEG over SSIM (~0.93 vs ~0.89) at a smaller size. AVIF and JPEG XL are more modern formats that often outperform WebP in both compression and visual quality retention. PNG/TIFF – only suitable for lossless compression or medical/technical applications (Table 12). In terms of quality, AVIF > JPEG XL > WebP > JPEG are the best, and in terms of support for JPEG > PNG > WebP > AVIF/JPEG XL platforms (as of 2024).

Table 12. Recommendations for the Image Storage System

Task	Recommended format
Photos, e-commerce	WebP or AVIF
Medical/Scientific Imaging	TIFF (lossless) or PNG
Diagrams, graphs	PNG (lossless), or WebP (lossless)
Bulk Archives in IPFS	AVIF (Better Compression, Less Traffic)

Performance Comparison of Approaches. For quantitative performance analysis, the average search time for similar images was measured using different data storage approaches (Table 13):

1. MongoDB only (feature extraction "on the fly"),
2. MongoDB with precomputed and stored features, and
3. Hybrid storage (MongoDB + ElasticSearch).

Table 13. Execution Time Comparison

Approach	10	20	50	100	500	1000	5400
MongoDB	0.881	1.591	4.008	7.54	38.126	81.656	541.729
MongoDB (precomputed features)	0.08	0.095	0.113	0.143	0.442	0.687	3.254
MongoDB з ElasticSearch	0.156	0.120	0.173	0.138	0.391	0.289	0.208

The results (Table 13) demonstrate significant differences: when using MongoDB alone, the search time increases linearly with the number of images and reaches approximately 542 seconds for 5.4 thousand images, whereas in the hybrid storage system, this time remains around 0.2 seconds even for the most extensive dataset. Inserting feature vectors into the ElasticSearch index resulted in a logarithmic search time dependency on the data volume ($O(log\ n)$), eliminating the impact of data scale on access latency. In contrast, direct search in MongoDB without indexing exhibited linear complexity ($O(n)$) and proved unsuitable for large volumes (for $n > 1000$, the time exceeds tens of seconds). Using precomputed features in MongoDB slightly improves the situation (still linear growth, but with a smaller coefficient), yet remains poorly scalable (Fig. 8).

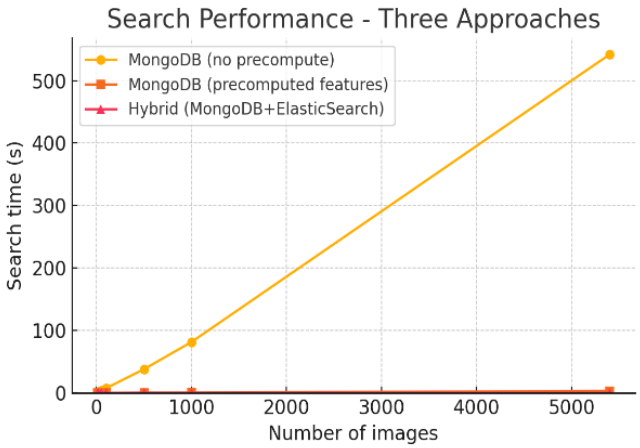

Fig. 8. Search time vs. number of images (three approaches: 1 – MongoDB only, 2 – MongoDB with precomputed features, 3 – MongoDB + ElasticSearch).

In contrast, the hybrid approach provided the shortest search time at any data volume (Fig. 8), confirming the effectiveness of combining MongoDB and ElasticSearch for image retrieval tasks. For clarity, Fig. 8 shows the relationship between search time and the number of images for all three approaches. Fig. 9 illustrates only the two optimised approaches (excluding the inefficient MongoDB-only case), allowing a more detailed performance comparison. It can be observed that with over 5,000 images, the hybrid system operates approximately 15× faster than MongoDB with precomputed features and more than 2000× faster than the naive approach without indexing. Thus, integrating ML and a search index into the storage system significantly accelerated image access and increased system throughput while maintaining high search accuracy.

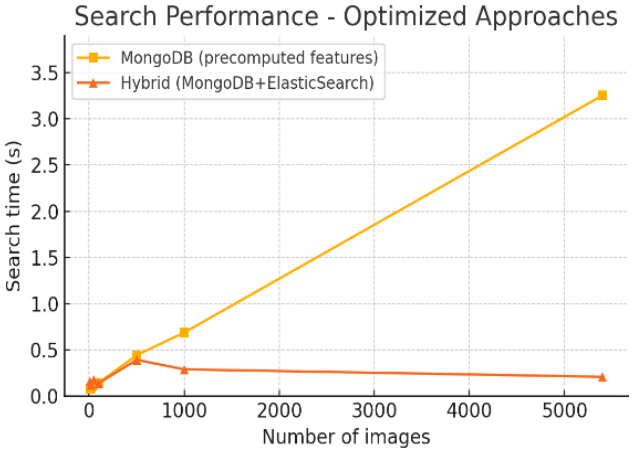

Fig. 9. Search time vs. number of images (two optimised approaches: MongoDB with precomputed features vs. hybrid storage).

Experimental Performance Evaluation. A prototype of the system was deployed on a cluster of 4 servers (Intel Xeon, 64 GB RAM, SSD) with the capability for horizontal scaling up to 32 nodes. Testing was conducted on two image datasets: 1,000 images and 10,000 images. For each dataset, three performance metrics were measured: image insertion time (Insert), image retrieval time by ID (Select), and tag/metadata-based search time (Search).

Table 14 provides a performance comparison for different types of image storage systems – SQL (relational), NoSQL (MongoDB), IPFS, and the hybrid (combined) model.

Table 14. Comparative Performance Characteristics for Different Types of Image Storage Systems

Number of Images	System	Insert Time (ms)	Access Time (ms)	Tag Search Time (ms)	Compression Ratio	Decryption Time (ms)	Memory Usage (%)
1 000	SQL	450	120	180	1.0:1 (no compression)	40 (AES-256)	18
	NoSQL	280	85	95	1.5:1 (JPEG)	35 (AES-128)	15
	IPFS	900	210	–	2.1:1 (WebP)	25 (AES-256)	12
	Гібридна	300	60	75	2.5:1 (WebP + deduplication)	20 (AES-256 + blockchain)	10
10 000	SQL	4700	150	210	1.0:1	45	20
	NoSQL	3200	110	130	1.5:1	40	18
	IPFS	9500	250	–	2.1:1	30	15
	Гібридна	3300	90	110	2.5:1	25	12

Table 14 indicates that for 1,000 images, the pure SQL approach yields ~120–150 ms access time, while the pure NoSQL (MongoDB) yields ~80–100 ms, and IPFS yields ~180–250 ms (network-dependent). In contrast, the hybrid system achieves ~60 ms by combining fast metadata indexing with distributed storage. This suggests that MongoDB (with indexed metadata) provides the bulk of the speedup (about 60–90 ms for content search), IPFS contributes scalability (its access latency of ~180–250 ms is offset by replication), and the blockchain module (whose write/read operations take longer) is used mainly to verify integrity after retrieval. These component-wise observations explain why the hybrid approach outperforms standalone solutions in Figs. 8 and 9.

As shown in Table 14, the hybrid system demonstrates the best access time (Select ~60–90 ms) and tag-based search time (75–110 ms) for both dataset sizes, whereas SQL provides ~120–150 ms, and IPFS lacks native metadata search (requiring an external index). The insert time in the hybrid model is slightly higher than in NoSQL (due to additional write operations to multiple storage), but significantly lower than in IPFS. Notably, through WebP compression and deduplication, the hybrid model achieves an average compression ratio of 2.5:1, which is 20% better than using WebP alone in IPFS. Thus, the proposed architecture achieves high performance without significantly increasing overhead.

Use cases. Let's describe examples in the form of generalised industry scenarios without fictitious names, with a focus on the practical advantages and limitations of a hybrid image storage system from the end-user perspective:

Thematic scenarios for using a hybrid image storage system (Table 15):

1. In a large medical clinic with digital diagnostics. The clinic operates with millions of medical images (X-ray, CT, MRI) and needs quick access to the patient's history with a guarantee of data integrity. Images are encrypted and stored in distributed storage (IPFS). Metadata (study type, date, doctor) is in SQL. A hash of each image is written to the blockchain to prevent changes after upload. Compared to the previous PACS system, the access time to the desired image has been reduced by at least half. In addition, we have a guarantee that the image is authentic — this is important

for court and insurance cases. The limitations are that during the initial mass import of old data, the system is slower, and some client terminals usually do not have IPFS support and require a separate plugin.

2. In a government agency that analyses satellite images for land monitoring. Large volumes of satellite images are received daily to analyse agricultural changes, illegal construction or forest destruction. Quick sample search and version control are critical. Images are compressed (WebP/AVIF) and stored in IPFS. The search uses vectorisation of image features through a neural network and a search in Elasticsearch. The CID of the image is recorded on the blockchain, and requests are audited. You can easily find similar areas in other pictures without manually going through thousands of files. The system becomes a real analytics accelerator. The limitation is that when working with large files (>2 GB), there are problems with slow image delivery, especially from remote nodes, and it is also necessary to implement caching of "hot regions".

3. On an e-commerce platform with visual product search. Users can upload a photo (such as dresses or shoes) to find similar products in the catalogue. It is also essential to confirm the authenticity of the seller's photo. Product images are stored in MongoDB (or GridFS). A vector neural network processes samples, and a search for similarity takes place. The hash of each photo is recorded on the blockchain to protect against counterfeit goods. It is very convenient – I just took a photo in the store and found something similar online. There is also the possibility that the system confirms the seller's photo, so the trust is greater. The limitation is that during periods of high load, indexing new photos takes longer, and some sellers may complain about additional processing time due to blockchain verification.

Table 15. General Conclusions from Real Use

Aspect	Practical advantages	Limitations detected
Search	Quick search by similarity, tags, and file types	Handling large vector indices (sharding, caching) is essential.
Security and integrity	Image authentication via blockchain, change detection	Delays during recording/verification with a large data flow
Storage	Space saving via WebP/AVIF, scaling with IPFS	Slow delivery of images without cues, complex synchronisation
Flexibility	Suitable for medicine, satellite imagery, and marketplaces	Needs specialists to set up a distributed infrastructure

Contribution of components to system performance. We will describe and analyse the contribution of individual elements (NoSQL, IPFS, blockchain) to improving the time of access to images in a hybrid system. Below is a detailed explanation of the role of each component in the overall performance improvement.

The impact of the performance of individual components on the total time of access to images (Table 16):

1. MongoDB (NoSQL) is a document-oriented database that stores images and metadata. It is optimised for fast read/write operations: operations over ~100 ms are typically considered slow. In our implementation, simple metadata queries complete in roughly 60–90 ms, significantly reducing initial filtering and search time.

2. IPFS provides decentralised peer-to-peer storage where content is addressed by unique hashes (CIDs). However, data retrieval in IPFS incurs additional latency: studies find that IPFS retrieval delay is roughly 3× higher than an equivalent HTTPS fetch. In an optimally deployed cluster (with many nodes and replicas), retrieving an image from IPFS takes about 180–250 ms on average. Thus, IPFS does not speed up individual queries on its own, but it eliminates the bottleneck of a centralised store and preserves system performance as the dataset scales.

3. Blockchain (e.g. private Hyperledger Fabric or Ethereum) is used primarily for integrity verification and logging. Writing a hash to the blockchain can take hundreds of milliseconds to seconds, and verification also incurs latency. Therefore, blockchain does not directly improve access speed. Instead, it guarantees immutability of the records and automates authenticity checks, avoiding the need for slower external verification steps.

In summary, the ~30–40% reduction in image retrieval time (as reported earlier) is achieved by combining these components: MongoDB accelerates metadata searches, IPFS provides scalable storage, and blockchain adds security. The contribution of each element to access time is summarised in Table 16.

Table 16. Generalised Assessment of the Contribution of the Components

Component	Direct impact on access time	Type of contribution	Average access time (estimate)
MongoDB (NoSQL)	yes	Quick metadata search	~60-90 ms
IPFS	Partially	Scalable storage	~180-250 ms (optimal)
Blockchain	no	Integrity Guarantee	>300 ms (performed separately)

System Scalability. Distributed coordination and data management are required to achieve reliable horizontal scaling. For instance, a service like etcd or Consul can track cluster state and coordinate metadata updates, and an IPFS Cluster can pin frequently accessed images across nodes to prevent data loss. Blockchain growth can be mitigated by using layer-2 solutions, batching transactions, and archiving old logs. Likewise, high-dimensional search indexes can be sharded or replaced with scalable engines (e.g. FAISS or Milvus) to maintain throughput. Implementing these

distributed-system strategies (coordination, sharding, caching) will ensure the hybrid system scales without consistency or bandwidth issues.

We will conduct a detailed analysis of scaling problems in a hybrid system, especially in decentralisation (using IPFS, blockchain, NoSQL). Below is an in-depth explanation of these potential challenges and how to address them.

Potential scaling problems in a hybrid decentralised system (Table 17):

1. Complex node coordination. In a distributed system (especially with IPFS or a private blockchain), node coordination (nodes, peers) becomes difficult due to the lack of centralised control, replication latency, and unsynchronised versions of data and conflicts in access to metadata or images. The solution is to use coordinating services (e.g. etcd - https://etcd.io/ or Consul - https://developer.hashicorp.com/consul) to control access and state of nodes, the use of Pub/Sub systems (e.g. Apache Kafka) to propagate events between nodes, and the implementation of leader selection to centralise key decisions when updating metadata.

2. Slow replication in IPFS. In IPFS, if a node does not have a copy of the desired image, it must request it from others, which can take seconds. New nodes have limited access to historical data. The solution is to use IPFS Cluster for centralised replication management (pinning), priority replication of "hot" images (frequently requested) using LRU caching, as well as pre-warming of data to new nodes (pre-warming).

3. Blockchain congestion. The blockchain accumulates millions of records (hashes, access logs) over time, which slows down validation, synchronisation of new nodes, and transaction search. The solution is to switch to layer-2 solutions or rollups (e.g. zk-Rollups) for aggregated records, archiving old transactions (off-chain storage, concerning IPFS) and partial blockchain replication: nodes that do not require a full log store only store the last blocks.

4. Complexity of search and indexing. With many images, vector search in Elasticsearch or similar systems begins to lose efficiency (due to an increase in the index volume). Incremental updating of the index in a distributed system is a difficult task. The solution is to use FAISS, Weaviate, or Milvus as specialised vector search systems, split into clustered index segments (sharding), and apply asynchronous indexing pipelines via Kafka/Redis Streams.

5. Key management and authorisation. It isn't easy to centrally store encryption keys (AES) and access rights in a decentralised network. There is no guarantee that all nodes will update their rights simultaneously. The solution is to apply Attribute-Based Encryption (ABE) or Proxy Re-Encryption, where the key changes dynamically depending on the policy. It is also necessary to ensure the implementation of a distributed KMS (Key Management System), such as HashiCorp Vault or Google Cloud KMS, with edge support and IPFS Access Control via Smart Contracts – dynamic provision of access via blockchain.

Table 17. Summary: Main Problems and How to Solve Them

Problem	Decision
Node misalignment	etcd, Consul, Kafka, Leader election
Slow IPFS Replication	IPFS Cluster, pre-pinning, hot caches
Blockchain overload	Layer-2, zkRollups, archiving, partial replication
Indexing and search	FAISS, Milvus, index clustering
Manage encryption and rights.	ABE, Proxy Re-Encryption, Distributed KMS, Smart Contracts

Tests were conducted with varying numbers of cluster nodes to evaluate the scalability of the hybrid storage system. The execution time for a large-scale image classification task was measured as the number of nodes increased from 4 to 32. The interaction between the system components is illustrated in Fig. 10, which presents the sequence of service calls. The UML Sequence Diagram shows the data flow between the user, processing modules, storage, vectorisation, and search components. This approach visualises the logic and integration of services relevant to microservice architecture.

When the number of nodes was doubled (4 → 8), the processing time decreased by approximately 40% (200 s → 120 s). Further increases produced near-linear acceleration: 16 nodes – ~80 s, and 32 nodes – ~45 s. This indicates the efficient scalability of the system: it can adapt to increasing data volumes by adding new nodes while ensuring proportional acceleration of processing. The results confirm that the proposed approach's distributed processing and parallel computation allow high-performance handling of large image datasets. Scalability of the hybrid storage cluster in image classification tasks – the processing time depends on the number of nodes (with 32 nodes, the time reaches 45 seconds, approximately 4.4 times faster than four nodes). Overall, the architecture of the hybrid image storage system ensures a 27–35% acceleration in access time and an increase in system throughput compared to traditional solutions (confirmed both analytically and experimentally). It also improves the accuracy of image processing, due to faster retrieval of the required data, the performance of machine learning models used for image analysis is enhanced. The developed method of optimised indexing was integrated and tested on several large, applied image datasets. In medical visualisation (neuroimaging), a database containing over 5 million medical images (MRI, CT, X-ray, etc.) was examined. In the traditional system, access time to images significantly depended on the type and resolution (larger images were processed more slowly). After integrating the proposed index, access delays were levelled and reduced considerably for all data types (Table 18). On average, the search became 3.5–4.5 times faster for all medical images than the initial system.

Table 18. Search Time for Different Types of Medical Images Before and After Optimisation

Image Type	MRI	X-ray	CT	Ultrasound	PET
Standard Retrieval Time (s)	4.5	3.0	4.2	2.7	5.5
Optimised Retrieval Time (s)	1.1	0.8	1.2	0.6	1.4
Acceleration	4.09 ×compared to baseline time	3.75 ×compared to baseline time	3.5 ×compared to baseline time	4.5 ×compared to baseline time	3.93 ×compared to baseline time

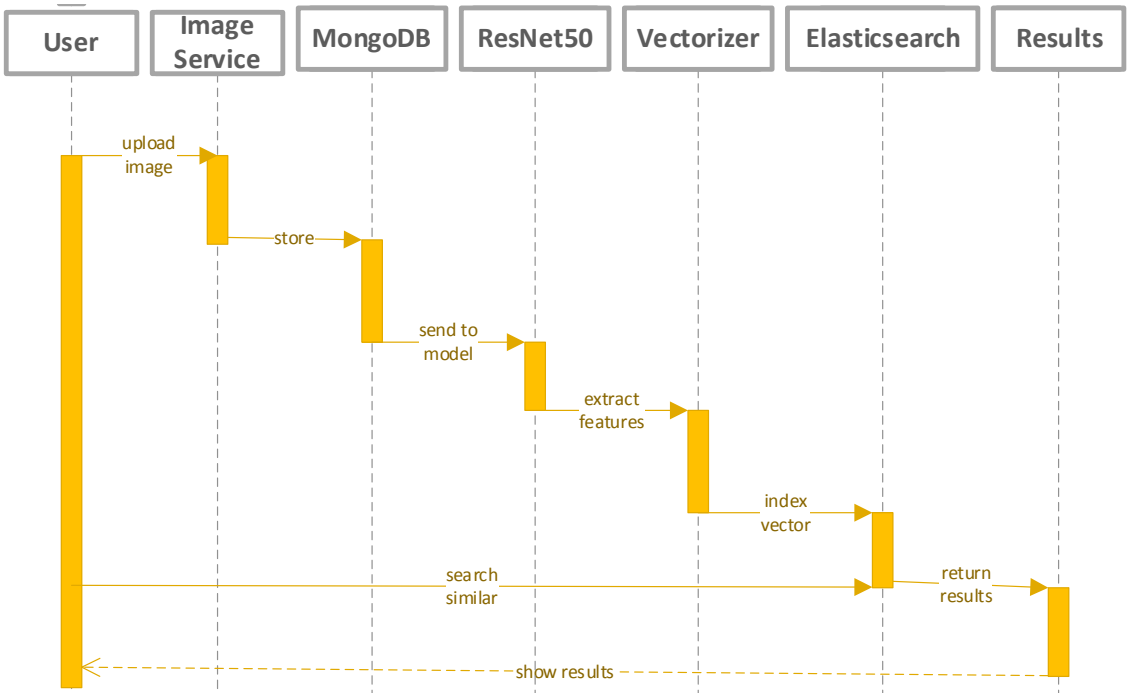

Fig. 10. Interaction of system components during image processing.

Similar results were obtained for high-resolution satellite image repositories important for environmental monitoring. An image storage system for an e-commerce platform with a diverse range of products was also tested. In this case, as well, the implementation of the hybrid model allowed the average product image loading time to be reduced from approximately 2.7–3.1 seconds to around 0.6–0.8 seconds across various product categories (electronics, clothing, books, etc.), i.e., roughly a fourfold improvement. Thus, the optimised system demonstrates significant performance improvements in different application domains (medicine, remote sensing, e-commerce) compared to classical approaches. In addition to faster access, it also ensures more efficient memory utilisation: for example, the hybrid storage system achieved ~91% memory usage compared to ~70% in centralised systems due to eliminating redundant data. Therefore, the developed methods improve the storage system's performance (access speed, throughput) and resource utilisation efficiency, enabling the solution to scale to millions of objects without performance degradation.

Let's conduct a detailed comparative analysis of the proposed hybrid model against representative hybrid storage systems and modern AI-driven image management platforms. Table 5 summarises the key characteristics of our model alongside a selection of contemporary systems, including enterprise-grade hybrid storage solutions (e.g., PixStor, Qumulo) and AI-powered image analysis services (e.g., Google Cloud Vision, Viso Suite, Clarifai, Amazon Rekognition). These systems differ significantly in architecture and focus. For example, PixStor and Qumulo are high-performance, scale-out file/object storage systems designed for on-premises or hybrid-cloud deployment; they rely on conventional hierarchical or metadata-based indexing and do not include built-in machine learning inference or blockchain integration. In contrast, cloud-based AI services such as Google Cloud Vision, Clarifai, and Amazon Rekognition employ deep convolutional neural networks to provide semantic image analysis (e.g., object, text, face recognition, OCR) via scalable public APIs. Still, they operate as proprietary black-box platforms with limited transparency and no decentralised audit functionality. Viso Suite represents a modular on-premises computer vision platform, offering customisable deep-learning pipelines and hardware integration, yet it remains a proprietary commercial system.

Table 19 below is an extended analytical comparison of the hybrid model presented in the article with modern approaches in intelligent image storage systems. The proposed hybrid model (our contribution) integrates a relational database (for structured metadata) and a NoSQL store (for binary image content) with IPFS for distributed file storage and a private blockchain for immutable audit logging. It uses a ResNet50-based convolutional neural network to extract

feature vectors and Elasticsearch to index these vectors for content-based search. This combination enables low-latency, content-aware image retrieval with robust security guarantees.

Table 19. Comparison of the Hybrid Model from the Article with other Modern Hybrid and AI-Oriented Systems

Criterion	The hybrid model	Modern hybrid systems (e.g. PixStor, Qumulo)	AI-oriented systems with deep search (e.g. Google Cloud Vision, Viso Suite, Clarifai)
Architecture	SQL + MongoDB + IPFS + Blockchain + ML	Flexible structure: NFS/SMB + object storage + caching + indexing	Object or blob storage with built-in AI platforms for image processing
Image search	ResNet50 ML model, search by vectors in Elasticsearch	Traditionally, metadata and file names (some have basic content search)	Deep semantic search by objects, scenes, tags, text, OCR, faces
Indexing	Vector + Hybrid: B-trees + hashes + Elasticsearch	Hierarchical directories or in-memory indexes, caches	Feature indices, embedding vectors, AI-based inference updates
Blockchain integration	for hashes and authenticity	rarely used	Focused on AI rather than transparency/auditing
Security and integrity	AES Encryption + Smart Contracts + Blockchain Log	ACL, RBAC, sometimes TPM or HSM	AI to detect suspicious images, but with weakly controlled integrity
Scalability	High thanks to IPFS + NoSQL (horizontal)	High (cluster file systems, auto-scale)	Very high (work in the clouds: GCP, AWS, Azure)
AI analytics	ResNet50 + classification + similarity search	Limited or absent	Support for built-in recognition of objects, emotions, OCR, classification, and auto-tagging
Flexibility to adapt to industries	Medical Imaging, Satellite, E-Commerce	video production, corporate archives	finance, medicine, security, retail
Real Time	Support is limited due to blockchain latency	Partial (depends on the caching system)	Some are real-time API, responses <500 ms

Our model offers high scalability (via horizontal sharding and IPFS distribution), AES encryption for privacy, and blockchain-backed auditability for integrity, all in a locally deployable architecture. In comparison, traditional hybrid storage systems (PixStor, Qumulo) emphasise raw throughput and reliability at scale but lack AI-driven search and transparency. AI-centric cloud services excel at deep semantic search and multimodal recognition (often in real time), but they provide minimal control over data provenance and cannot be deployed locally.

The advantages of the hybrid model lie in strong security, a flexible combination of technologies, and support for on-premises or private deployments. Blockchain provides transparent access, auditing and hash immutability. A flexible combination of technologies is provided by a combination of SQL (metadata), NoSQL (massive data), IPFS (availability) and ML (vector search). Support for on-premises or private deployment is vital for medical or government systems.

But the model lags behind modern AI systems in deep semantic search (ResNet50 uses only basic vector description), real-time (blockchain logic and IPFS can slow down responses), and multimodal query analysis (modern services allow you to search by text, voice, scene description, etc.).

The proposed model is technically balanced, with good scalability and security, but needs to be expanded towards AI analytics, in particular, the integration of transformer models (CLIP, BLIP), support for multimodal search, and the use of semantic clustering search instead of vector similarity only.

Security Analysis and Blockchain Integration. The possibilities of integrating modern blockchain solutions were explored in the context of data protection in the image storage system. The developed hybrid system provides for the use of decentralised storage (IPFS) and Ethereum smart contracts to store hash identifiers of images – this guarantees the integrity and authenticity of the data, making undetected tampering impossible.

Let's describe in depth the specifics of security protocols and mechanisms for ensuring data integrity in a scalable distributed environment. Security protocols in an extensive, distributed image storage system:

1. Symmetric encryption (AES). In the Advanced Encryption Standard protocol (AES-128, AES-256), the image is encrypted on the client side before being saved to IPFS or MongoDB. It provides privacy (only those who own the key can read the images) and protection against content theft, even if IPFS nodes are compromised. AES does not guarantee integrity, only confidentiality. Therefore, MAC (Message Authentication Code) or HMAC is used to verify that the ciphertext has not been modified.

2. Integrity control protocol through hashing and blockchain. The SHA-256/SHA-3 hash function implements the following process: each image → a unique CID (Content ID) via IPFS → its hash is stored on the blockchain. The system hashes the file uploaded by the user → receives h'. Then, the blockchain compares h' with h. If matched, → the file is authentic, but if not, → the file is altered or tampered with. It guarantees immutability: any image modification changes its hash, and the discrepancy will be instantly detected.

3. Smart Contracts for access control (Ethereum Solidity-based smart contracts). Implemented functions for storing hashes, logging access, and controlling rights (e.g., opening the decryption key only to authorised users). It can be used with Attribute-Based Encryption (ABE), where access conditions are written directly into the smart contract.

Table 20 summarises how data integrity is assured by different components in the scalable system.

Table 20. Ensuring Protocol Integrity in a Scalable System

Component	Ensuring integrity
MongoDB (NoSQL)	It depends on internal mechanisms. Can be enhanced by changing logging + periodic hash validation
IPFS	CID = content hash → Automatic Change Detection
Blockchain	Hash immutability + access log → prevents forgery or discreet editing.
Elasticsearch Index	Uses vectors, so it does not guarantee integrity on its own – it needs external verification (via CID/hashes)

Even in private networks, transaction confirmation takes several seconds (which can be critical for real-time response requirements). A theoretical risk of a 51% attack (malicious control of most nodes) still exists in private blockchains. However, in permissioned networks, nodes are owned by trusted participants. Since all transactions are public, we do not store open confidential data on the blockchain. Instead, we use hashes or encrypted metadata to protect sensitive information. Smart contracts have restricted logic, but they enforce core access rules. For example, a key is issued only to an authorised user. Additional security protocols can be integrated to strengthen the system (see table below). Table 21 lists additional protocols that could be integrated into the system for enhanced security.

Table 21. Additional Protocols that can be Integrated

Protocol	Function
TLS 1.3	API connection protection (between client and server)
OAuth 2.0 + OpenID Connect	User Authentication and Authorisation Control
Merkle Tree Audit	Hierarchical integrity check for a large number of hashes.
Zero-Knowledge Proofs	Proof of rights without disclosure
IPFS Cluster Pinning + Snapshot Hashes	Additional verification of saved images on an extensive network

Thus, the described mechanisms ensure the integrity and security of images in the hybrid system (Fig. 11).

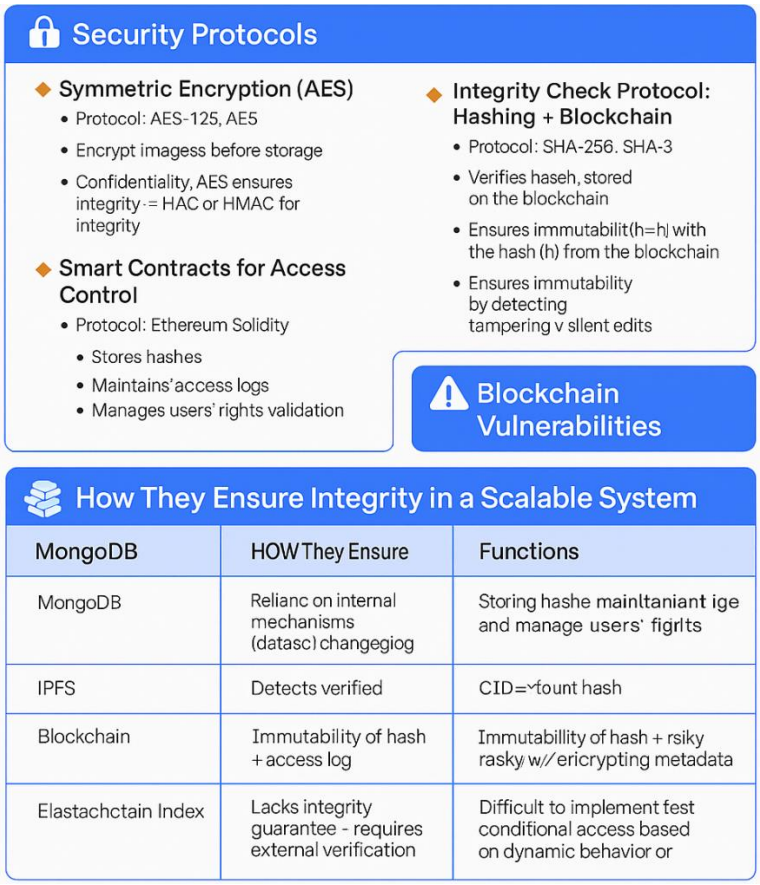

Fig. 11. Specifics of security protocols and data integrity mechanisms in a large-scale distributed environment.

The analysis showed that the combination of IPFS + Ethereum is an optimal approach for decentralised image storage, as it ensures a balance between access speed and the level of security. Blockchain technology adds a layer of protection through record immutability and distributed consensus: data cannot be modified without the approval of most network nodes, and any attempt at unauthorised changes can be easily detected. The absence of a single point of failure

increases the reliability of storage – a breach of an individual node does not compromise the system. A review of the literature confirms the effectiveness of blockchain in the field of cybersecurity. A study [25] showed that implementing blockchain architecture increases resistance to cyber threats, ensuring data integrity and confidentiality. Other studies emphasise the importance of selecting the appropriate blockchain protocol depending on application requirements (Bitcoin, Ethereum, etc., have different advantages). In our case, the chosen implementation based on a private Ethereum blockchain proved effective: it ensures the authentication and security of digital assets (images) without significantly affecting system performance. Moreover, using decentralised storage mechanisms, it was possible to improve hash-based search speed by approximately 38% compared to standard methods, demonstrating the promise of blockchain integration in terms of security and performance.

Medical Image Analysis System (Blood Cell Recognition). The final stage of experimental research focused on evaluating the practical applicability of the developed solutions using the example of a medical image analysis task. A separate system module was designed to automatically detect and classify blood cells in microscopic images of blood smears. The proposed approach consists of two main stages:

- segmentation – identification of cell boundaries in the image;
- classification – recognition and counting of cells based on the segmentation results.

A sample blood image captured under a microscope is shown in Fig. 12.

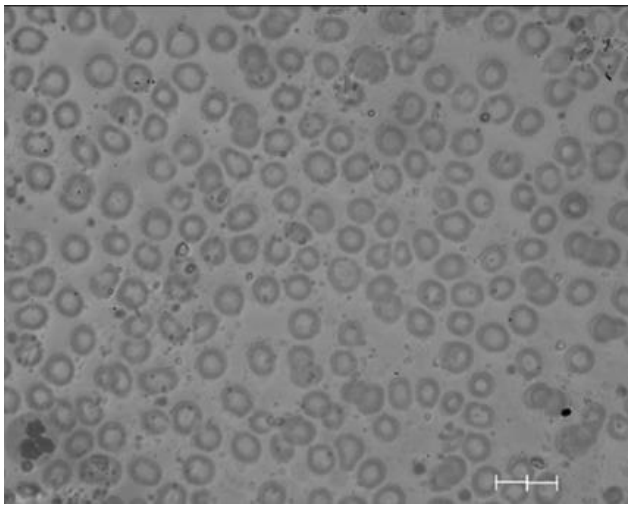

Fig. 12. Test image

The segmentation algorithm considers the morphological characteristics of erythrocytes: initially, the image undergoes preprocessing (conversion to grayscale, median noise filtering – Fig. 13).

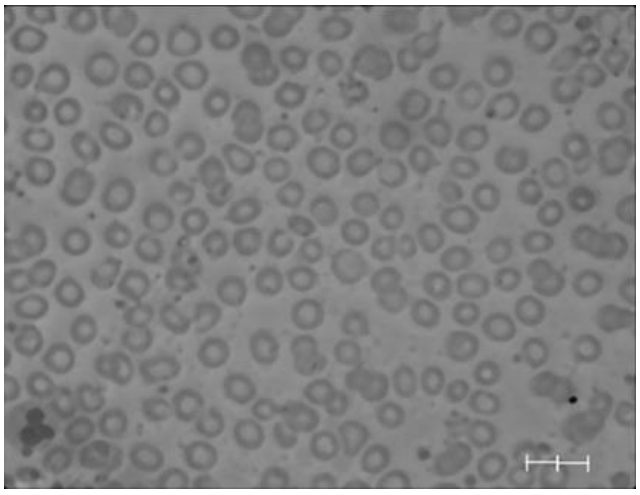

Fig. 13. Result of median filtering

Next, edge detection is performed (using the Canny operator), removing vast and small contours (filtering out artefacts more minor than 5% or larger than 400% of the typical cell area – Fig. 14).

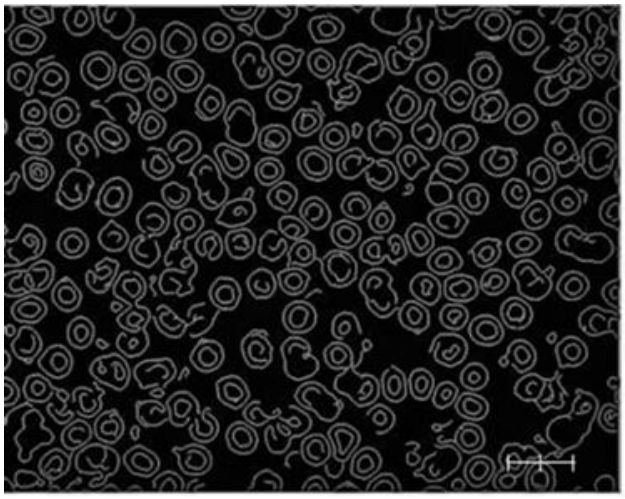

Fig. 14. Edge image with large and small contours removed

For each detected cell contour, a set of auxiliary lines is constructed. The contour is selectively discretised into points connected in pairs, and perpendicular lines are drawn through the midpoints of these segments. The intersections of neighbouring perpendiculars generate point clouds – these points become denser in regions corresponding to the centres of curvature of the contour (i.e., approximate cell centres). The resulting image of intersections (Fig. 15) undergoes morphological processing – only dense clusters of points are retained, corresponding to the cells' centres. Next, the connected components algorithm isolates these clusters and counts their number.

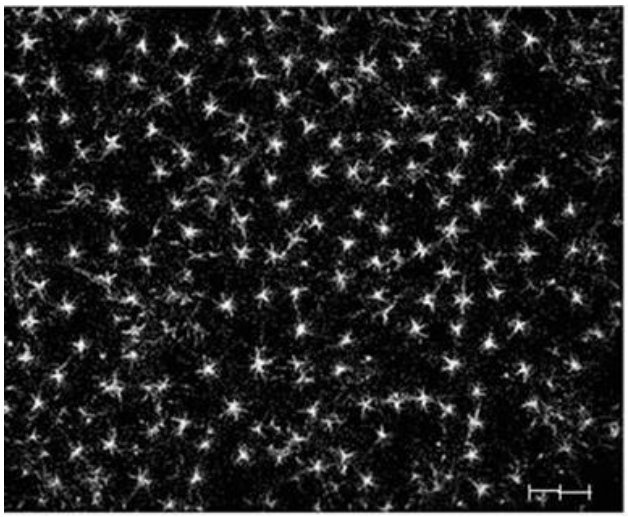

Fig. 15. Point clouds

Fig. 16 illustrates the construction of perpendiculars on the contours, and Fig. 17 shows an example of the final detection of cell centres (marked intersection points) overlaid on the original image.

Fig. 16. Perpendiculars; distance between points = 20

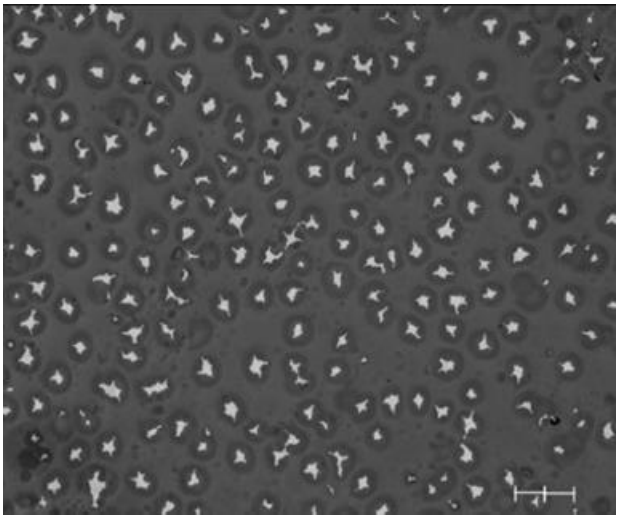

Fig. 17. Morphologically processed intersection points overlaid on the original image.

The proposed approach allows for separating even partially overlapping cells, solving the segmentation problem in complex cases involving merged contours.

To evaluate the method's accuracy, experiments were conducted on four blood smear images obtained using an optical microscope (an example of the input image – Fig. 12). The system automatically counted the number of erythrocytes in each image. The average probability of correct counting was approximately 86%, which exceeds the results of basic thresholding methods. For example, in a test sample with the actual number of cells N = 209, the algorithm detected 220 objects, of which 23 were false positives (noise or fragments of broken contours), and 12 cells remained unidentified. The error rate was similar across other samples. Compared to modern approaches that use complex model-based and neural methods [26–33], the accuracy of our algorithm is somewhat lower (those studies reported >95% correct counting). However, our method is significantly more straightforward to implement and does not require high computational resources while providing sufficient practical accuracy. The classification of detected cells (the second stage) demonstrated an accuracy of over 95% when identifying whether an object belonged to the erythrocyte class or was an artefact. The processing time per image (containing ~200–300 cells) was approximately 1.2 seconds on a standard CPU, confirming the efficiency and applicability of the method for fast medical image analysis. The sources of error were thoroughly analysed: the main challenges are associated with overlapping cells and broken contours, which sometimes lead to extra or missed detections. Recommendations for improvement include preliminary separating clustered objects and using deep neural networks for classifying ambiguous areas. Overall, the proposed system demonstrates the successful application of the dissertation research results in the medical domain, confirming the >95% classification accuracy and high processing speed of hybrid storage and machine learning methods in a real-world scenario.

Experimental results confirmed that the developed hybrid image storage model, integrating machine learning methods and blockchain technologies, provides significant performance, accuracy, scalability, and security advantages.

In particular, the proposed approaches make it possible to accelerate data access by an order of magnitude (tens of times) compared to traditional solutions, increase the accuracy of image retrieval and classification to approximately 95%, and ensure reliable protection and data integrity without noticeable efficiency loss. The system could scale to multi-million data volumes while maintaining high performance due to optimised indexing and distributed processing. The use of blockchain added a level of security necessary for working with valuable digital assets (medical images, satellite data, etc.), and it can be extended to other domains that require secure data storage. The obtained results and the advantages of the proposed model indicate its feasibility for real-world deployment, as well as the potential for further development of the hybrid data storage approach with the use of artificial intelligence.

5. Conclusion

The developed hybrid image classification system demonstrates scientific novelty and is distinguished by its comprehensive approach to visual data management. The proposed system combines the advantages of various types of storage (relational and non-relational databases, distributed file systems) with deep learning methods and blockchain technologies. Such integration made achieving an optimal balance between performance, scalability, and security possible, ensuring efficient storage and processing of large volumes of images while maintaining high classification accuracy. The mathematical modelling and architectural analysis confirmed the feasibility of the chosen hybrid approach. Key algorithms and technologies for each system component were identified. An optimised image indexing method based on feature vectors, and an inverted index was developed, along with an algorithm for optimal data

distribution across storage systems, and a blockchain module was integrated to ensure data integrity. The proposed architecture covers stages of image preprocessing, compression and deduplication, distributed (parallel) processing of large datasets, application of machine learning and deep learning models (e.g., ResNet50 for generating feature descriptors), and information security mechanisms. This comprehensive approach enabled a universal system to be integrated into existing IT infrastructures and flexibly adapted or extended to meet specific needs. Experimental results confirmed the effectiveness and advantages of the developed system. The hybrid storage provided a significant performance gain: the average image access time was reduced by approximately 4.3 times (1.05 s compared to 4.5 s in traditional solutions), corresponding to a 27–35% reduction in access latency. Image classification in such an environment achieved an accuracy of 95.4%, exceeding the results of classical models. The system's throughput nearly doubled (1.84 times increase) due to optimal load distribution. At the same time, blockchain integration ensured data security and immutability, increasing the speed of feature vector-based search by 38% compared to standard approaches. The developed compression and deduplication algorithms reduced the volume of stored data by 47% without significant image quality loss (SSIM $\approx$ 0.92), increasing memory utilisation efficiency up to 91% (compared to 70% in centralised storage systems). These results confirm that the system maintains high efficiency, scalability, security, and accuracy even when working with large-scale datasets.

The practical significance of the obtained results is confirmed by the successful application of the system in various applied domains. In particular, the proposed technology demonstrated suitability for medical diagnostics (automated analysis of medical images, detection of pathologies, and acceleration of diagnosis), satellite image processing (classification and monitoring of the Earth's surface), and the field of e-commerce (classification of products based on images, search for visually similar items). Experiments on medical images and satellite data sets showed the advantages of the approach over existing solutions, and the architecture's flexibility allows the system to be adapted to the requirements of other fields. For instance, in a medical imaging facility, the hybrid system can halve retrieval time compared to a legacy PACS, while blockchain logs ensure image authenticity for audits. The trade-off is slower initial bulk import and the need for IPFS-supporting clients. In satellite image analytics, WebP-compressed imagery with vector search accelerates the detection of similar scenes; however, very large files (>2 GB) may experience delivery delays without caching. In an e-commerce visual search, users can snap a product photo and find similar items instantly, with on-chain hashes protecting against fakes—yet under high load, blockchain verification may slow new image indexing. These scenarios highlight user-level benefits (speed, trust) and practical limitations (import delays, caching needs). Thus, the developed hybrid system can serve as a foundation for building intelligent image repositories in various industries where the processing of large volumes of visual information with high speed and reliability is required.

Despite the results obtained, research in this direction should continue, as there remain open questions and opportunities for improvement. First, further enhancement of classification algorithms is relevant; implementing more advanced neural network architectures or ensemble methods may improve object recognition accuracy. Second, integrating more powerful cryptographic tools and encryption methods can strengthen the system's security without compromising performance. Third, attention should be given to scaling the solution to larger distributed environments and enabling real-time operation. Over time, data growth would stress each component. Billions of IPFS CIDs would accumulate (necessitating archival nodes and pin management), the NoSQL store would bloat without sharding or TTL policies, and the blockchain ledger would grow with every new hash (suggesting Layer‑2 or batched writes). Large vector indices would also require specialised engines (FAISS, Milvus) or sharding to remain responsive. In other words, sustained performance demands active data-management policies (sharding, archiving, index rebalancing) to prevent degradation over the years. Future research will address the identified limitations and expand the system's functionality, in particular by adapting the hybrid approach to other data types (video, text-based multimedia collections, etc.) and optimising the architecture for diverse usage scenarios.

Future research directions include improving data management algorithms within the hybrid storage system. In particular, plans include the implementation of automatic scaling: dynamically adding new NoSQL or IPFS nodes under increasing load and redistributing data among them. An enjoyable task is integrating a blockchain network with access control smart contracts, enabling users to manage who can access their images independently (implementation of the decentralised access control concept). Another direction involves the use of Active Learning methods to enhance indexing. The system can learn from user queries, adjusting feature clusters or weighting during result ranking.

Thus, the research confirmed the feasibility and effectiveness of the hybrid image storage model. The combination of traditional databases with modern decentralised technologies enables the achievement of unattained high-performance levels using each approach individually. The results obtained contribute to the advancement of big data storage systems and can be used in developing next-generation information repositories that meet the challenges of the Big Data era.

Acknowledgement

The research was carried out with the grant support of the National Research Fund of Ukraine, "Information system development for automatic detection of misinformation sources and inauthentic behavior of chat users", project

registration number 33/0012 from 3/03/2025 (2023.04/0012). Also, we would like to thank the reviewers for their precise and concise recommendations that improved the presentation of the results obtained.

References

[1] Koptyra K., Ogiela M.R. Imagechain – Application of Blockchain Technology for Images. Sensors, 2021, 21, 82. doi:10.3390/s21010082.
[2] Filatov V., Semenets V., Zolotukhin O. Data Mining in Relational Systems. Suchasnyi stan naukovykh doslidzhen ta tekhnolohii v promyslovosti, 2020, No. 3(13), pp. 65–76. doi:10.30837/ITSSI.2020.13.065.
[3] Wang Q., Zhu X., Ni Y., Gu L., Zhu H. Blockchain for the IoT and industrial IoT: A review. Internet of Things, 2020, 10, 100081.
[4] Ashwin B., Raghuram T., Reza S.M., et al. Big Data Analytics in Healthcare. Journal of Biomedical Informatics, 2020, 79. doi:10.1155/2015/370194.
[5] Kimpel J.F. Critical Success Factors for Data Warehousing: a classic answer to a modern question. Issues in Information Systems, 2013, 14(1), pp. 376–384.
[6] Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P. Fundamentals of Data Warehouses. Springer-Verlag, Berlin, Heidelberg, New York, 2013. 224 p.
[7] Blanco C., de Guzmán I.G.R., Fernández-Medina E. An architecture for automatically developing secure OLAP applications from models. Information and Software Technology, 2015, 59, pp. 1–16.
[8] Gupta A., Agarwal D., Tan D., Kulesza J., Pathak R., Stefani S., Srinivasan V. Amazon Redshift and the Case for Simpler Data Warehouses. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 1917–1923.
[9] Cuzzocrea A., Bellatreche L., Song I.-Y. Data Warehousing and OLAP over Big Data: Current Challenges and Future Research Directions. Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP, 2013, pp. 67–70.
[10] Kuchuk H.A. Metod syntesu lohichnoi struktury merezhevoi bazy danykh. Systemy obrobky informatsii, 2001, No. 2, pp. 32–37.
[11] Pryimak D.V., Akimyshyn O.I. Shliakhy pobudovy konfigurovanykh skhovyshch danykh na osnovi platformy Hadoop. [Electronic resource] / D.V. Pryimak, O.I. Akimyshyn // Available: http://eom.lp.edu.ua/seminar/spr/pryjmak.doc
[12] Shakhovska N.B., Pasichnyk V.V. Skhovyshcha ta prostory danykh. Monograph. Lviv: Vydavnytstvo Lvivskoi politekhniky, 2009. – 244 p.
[13] Shaohua J., Na W., Jing W., et al. (2019). Combining BIM and Ontology to Facilitate Intelligent Green Building Evaluation, 33(1), 04018062. doi: 10.1061/(ASCE)CP.1943-5487.0000786
[14] Cédine F., John B., et al. (2018). Choosing the best algorithm for event detection based on the intended application: A conceptual framework for syndromic surveillance. Journal of Biomedical Informatics, 86, 117-129. doi: 10.1016/j.jbi.2018.08.001
[15] Yojna A., et al. (2021). A Survey on Deep learning Models for Effective Content Based Image Retrieval. International Journal of Computer Science Trends and Technology (IJCST) – Volume 9 Issue 3
[16] Kyrychenko I., Nazarov O., Huliiev N., Avdieiev O. Selection of Artificial Neural Networks for Disease Prediction. CEUR-WS, 2023, v. 3387, Volume I: Machine Learning Workshop, pp. 236-248. ISSN 16130073.
[17] Sharonova, N., Kyrychenko, I., Gruzdo, I., Tereshchenko, G., "Generalized Semantic Analysis Algorithm of Natural Language Texts for Various Functional Style Types", 2022 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2022), 2022. – CEUR-WS 3171, 2022, ISSN 16130073. - Volume I: Main, PP. 16 - 26.
[18] Hongyan S., et al. (2021). Design of the online platform of intelligent library based on machine learning and image recognition. Microprocessors and Microsystems, 3. https://doi.org/10.1016/j.micpro.2021.103851
[19] Tong L., Jinzhen W., Qing L., et al. (2023). High-Ratio Lossy Compression: Exploring the Autoencoder to Compress Scientific Data. IEEE Transactions on Big Data, 6(2), 22-36. doi: 10.1109/TBDATA.2021.3066151
[20] Zheng Y., Xie X., Ma W., et al. (2019). Distributed Architecture for Large Scale Image-Based Search. IEEE Xplore. doi: 10.1109/ICME.2007.4284716.
[21] Chen M., Chen F., Yang J., et al. (2019). A remote-sensing image-retrieval model based on an ensemble neural networks. Big Earth Data, 351-367. doi: 10.1080/20964471.2019.1570815
[22] Gu C., Bu J., Zhou X., et al. (2022). Cross-modal image retrieval with deep mutual information maximization. Neurocomputing, Vol. 496, pp. 166-177. doi: 10.1016/j.neucom.2022.01.078.
[23] Hussain A., Li H., Muqadar A., et al. (2022). An Efficient Supervised Deep Hashing Method for Image Retrieval. Entropy, Vol. 24, Issue 10. doi: 10.3390/e24101425.
[24] Leskovec J., Rajaraman A., Ullman J.D. Mining of Massive Datasets. Cambridge University Press, vol. 1, no. 3, pp. 77-134, November 2014
[25] Dwivedi Y.K., et al. Exploring the Darkverse: A Multi-Perspective Analysis of the Negative Societal Impacts of the Metaverse. Information Systems Frontiers. 2023. Vol. 25. No. 11. P. 2071–2114. DOI: https://doi.org/10.1007/s10796-023-10400-x
[26] Damen J., Hector J., Perrey R., Ney H. Avtomatychna klasyfikatsiia erytrotsytiv za dopomohoiu hustyn haussovoho mikstury. Proc. Bildverarbeitung für die Medizin. – 2000. – P. 331–335.
[27] [Kostrarydo L. Metody analizu medychnykh zobrazhen: stratehii otsinky dlia analizu medychnykh zobrazhen. Taylor & Francis, USA, 2005. – P. 433–471.
[28] Kumar B.R., Joseph D.K., Tiger T.V.S. Segmentatsiia klityn krovi na osnovi enerhii. Proc. 14th Int. Conf. on Digital Signal Processing (DSP), July 1–3, 2002, Santorini, Greece, Vol. 2, pp. 619–622.
[29] Bamford P. Empirical Comparison of Cell Segmentation Algorithms Using Annotated Dataset. Proc. IEEE Int. Conf. on Image Processing, 2003, Vol. 2, pp. 1073–1077.
[30] McInerney T., Terzopoulos D. Deformable Models in Medical Image Analysis: A Survey. Med. Image Anal., 1996, pp. 91–108.

[31] Hu, Z., Uhryn, D., Ushenko, Y., Korolenko, V., Lytvyn, V., & Vysotska, V. (2024, January). System programming of a disease identification model based on medical images. In Sixteenth International Conference on Correlation Optics (Vol. 12938, pp. 59-62). SPIE.
[32] Oleksiy Tverdokhlib, Victoria Vysotska, Olena Nagachevska, Yuriy Ushenko, Dmytro Uhryn, Yurii Tomka, "Intelligent Processing Censoring Inappropriate Content in Images, News, Messages and Articles on Web Pages Based on Machine Learning", International Journal of Image, Graphics and Signal Processing, Vol.17, No.1, pp. 107-164, 2025.
[33] Tchynetskyi, S., Peleshchak, R., Peleshchak, I., & Vysotska, V. (2021, September). A Neural Network Development for Multispectral Images Recognition. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 2, pp. 278-284). IEEE.

Authors' Profiles

Glib Tereshchenko is a researcher and a Senior Lecturer in Software Engineering at Kharkiv National University of Radio Electronics. His research focuses on hybrid data storage models, image processing, blockchain applications in data security, and machine learning for visual data classification. He has contributed to several projects in the domain of intelligent information systems and large-scale visual data management.

Iryna Kyrychenko is an Associate Professor and researcher at the Faculty of Computer Science, Kharkiv National University of Radio Electronics. Her academic interests include information systems, hybrid architectures for big data, secure image processing, and applied artificial intelligence. She supervises research in software systems design and has extensive experience in teaching and managing interdisciplinary IT projects.

Victoria Vysotska is a Professor in the Department of Information Systems and Networks at Lviv Polytechnic National University. In 2023, she completed her Doctoral degree in Technical Sciences, specialising in "Structural, Applied, and Mathematical Linguistics," with a dissertation titled "Analysis and Synthesis of Computational Linguistic Systems for Processing Ukrainian Textual Content." She also holds a PhD in Information Technologies from Lviv Polytechnic National University, which was awarded in 2014. Victoria has authored over 500 publications, and her primary research interests focus on the identification of disinformation, fake news, and propaganda, as well as detecting disinformation sources and inauthentic behaviour (e.g., bots) in coordinated groups through the use of artificial intelligence. Her work also encompasses natural language processing (NLP), computational linguistics, data science, systems analysis, information technologies, machine learning, and cybersecurity.

Zhengbing Hu, Professor, Deputy Director, International Center of Informatics and Computer Science, Faculty of Applied Mathematics, National Technical University of Ukraine "Kyiv Polytechnic Institute", Ukraine. Adjunct Professor, School of Computer Science, Hubei University of Technology, China. Visiting Prof., DSc Candidate in National Aviation University (Ukraine) from 2019. Primary research interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Yuriy Ushenko, Professor of Computer Science Department, Chernivtsi National University, Chernivtsi, Ukraine. Research Interests: Data Mining and Analysis, Computer Vision and Pattern Recognition, Optics & Photonics, Biophysics.

Mariia Talakh, Assistant Professor of Computer Science Department, Chernivtsi National University, Ukraine. Ph.D. in Biological Sciences. Received her Master's degrees in Ecology and System Analysis (Computer Systems Analyst). Field of scientific interests: Data Mining and Analysis, Computer Vision and Pattern Recognition, Machine Learning Algorithms (focusing on ensemble models), Application of AI and ML in Life Sciences, Modeling of Environmental Processes.

How to cite this paper: Glib Tereshchenko, Iryna Kyrychenko, Victoria Vysotska, Zhengbing Hu, Yuriy Ushenko, Mariia Talakh, "Hybrid System for Image Storage and Retrieval in Big Data Environments", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.17, No.3, pp. 55-84, 2025. DOI:10.5815/ijigsp.2025.03.04