Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Scalability and Throughput Analysis: Benchmarking PostgreSQL, MongoDB, and Couch DB in Distributed Environments

Salahuddin*
Department of Computer Science, NFC
Institute of Engineering and technology,
Multan, Pakistan.
Email: msalahuddin8612@gmail.com

Muhammad Tanveer Meeran
Faculty of Computer Science and Mathematics,
University Malaysia Terengganu, Malaysia.

Bareerah Saeed
Department of Computer Science,
COMSATS University Islamabad,
Vehari Campus.

Hafiz Muhammad Ijaz
Department of Information Technology,
The Islamia University of Bahawalpur, Pakistan.

ABSTRACT

The rapid evolution of digital technologies has intensified the demands on modern web applications, particularly in managing high-volume, concurrent read and write operations. Traditional relational databases, which rely on vertical scaling, often face limitations in performance, scalability, and cost-efficiency when handling large-scale data. In contrast, NoSQL databases especially document-oriented systems—provide flexible, horizontally scalable solutions tailored for distributed environments. However, selecting the most efficient database system for optimal throughput and low latency remains a challenge due to the diverse architectures of available NoSQL options. This study presents a comparative performance analysis of one relational database (PostgreSQL) and two document-oriented NoSQL databases (MongoDB and CouchDB) under distributed system conditions. Using a custom benchmarking tool, we evaluate their read/write throughput, latency, and scalability to determine their suitability for high-demand applications. Our findings aim to guide developers in selecting the most appropriate database solution based on performance requirements.

Keywords: MongoDB, Couch DB, PostgreSQL, NoSQL, Distribution Environment

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Introduction

In today's digital era, application developers are continually exploring innovative approaches to efficiently store and manage data in meaningful ways within databases. As the complexity and volume of data continue to grow, numerous database management systems (DBMS) have emerged to simplify the processes of data storage and retrieval. This study focuses on two primary categories of databases: Relational Databases and NoSQL Databases. Each of these systems uses distinct mechanisms for storing, organizing, and processing data. Traditional relational databases often struggle with managing large volumes of unstructured data due to their rigid schema and reliance on vertical scaling, which is both costly and less adaptable. The emergence of NoSQL databases was driven by the need to handle "big data" more effectively through horizontal scaling and eventual consistency mechanisms [1]. Unlike relational databases, NoSQL solutions provide greater flexibility, high availability, and scalability. The wide array of available NoSQL databases each with its own strengths and limitations presents a challenge for developers in selecting the most suitable option. Given the increasing demand for systems capable of handling massive data loads (such as those used in social media, e-commerce, and blogging platforms), schema-less and scalable solutions are increasingly preferred. This study focuses on two widely used NoSQL document-oriented databases, MongoDB and CouchDB, both of which store data as documents identified by unique key-value pairs [2] using JSON notation.

Unlike traditional databases, NoSQL systems are built on distributed architectures where performance is not dependent on a single node but instead on the collective performance of all nodes within a database cluster. To assess these capabilities, this research performs a performance evaluation through various CRUD operations under timed conditions, measuring efficiency using throughput and latency metrics. To conduct this evaluation, a custom benchmark tool was developed to measure performance in a distributed environment, since most online tools fail to provide sufficient control or transparency over workload simulation. This tool enables a detailed analysis of how each database responds under different conditions.

The primary goal of this research is to evaluate and compare the performance specifically in terms of latency and throughput of different database systems. This is achieved by conducting experiments on one relational database (PostgreSQL) and two NoSQL document databases (MongoDB and CouchDB) under varying workloads in a distributed computing environment. With the ever-increasing number of online users, managing high volumes of concurrent requests has become challenging for relational databases due to their reliance on vertical scaling. In contrast, NoSQL databases offer horizontal scalability, allowing seamless scaling across multiple nodes without compromising data availability or incurring significant costs. NoSQL databases follow the BASE model Basically Available, Soft State, Eventually Consistent which ensures availability and resilience in distributed systems, especially in the event of node failures. Unlike the strict guarantees of the ACID model, this property ensures that the system remains available even during partial failures or when data is incomplete. Through data replication across nodes, the system continues to serve requests reliably.

This aspect refers to the system's ability to hold temporarily inconsistent data across

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

nodes. The state may change over time without external input, but cached data can still be served efficiently, allowing faster responses and reuse of resources [3]. Once data updates stop, the system eventually synchronizes all nodes, ensuring that each one has an up-to-date copy. This model does not enforce consistency before processing each transaction but instead ensures that consistency is achieved over time.

Related work

The reviewed literature was selected to emphasize studies that primarily deal with relational databases (RDBMS) and NoSQL document-oriented databases. Many of these works involve comparative analysis between relational and NoSQL systems, while others conduct experimental evaluations of NoSQL databases under limited hardware and relatively light workloads. The findings from several of these studies suggest that NoSQL systems tend to offer greater flexibility, reliability, and scalability compared to traditional relational databases. The core objective of our own research is to evaluate both relational and non-relational databases to determine which performs better in terms of scalability, space complexity, and consistency. Matallah et al. [19] conducted a comparative study of two NoSQL databases HBase and MongoDB in a non-distributed environment with a small workload, using the Yahoo! Cloud Serving Benchmark (YCSB). Their results indicated that MongoDB outperformed HBase in both read and write operations. Similarly, Alejandro Corbellini et al. [20] compared key features of RDBMS and NoSQL systems, illustrating their findings with application-based implementations. They concluded that NoSQL excels in data storage when replication and sharding are applied, making it preferable in scenarios where consistency is not essential but fast data retrieval is required. In another contribution, Neelima Kuderu et al. [21] proposed a mapping framework that assists developers in migrating from relational to non-relational databases without modifying the existing schema, while preserving database semantics. They observed that after migration, data in a NoSQL system can be accessed in the same manner as before, without altering the application code.

A further study [22] synthesized results from earlier research to compare the capabilities of NoSQL and RDBMS for modern application development. According to this work, traditional RDBMS offer ACID compliance, strong consistency, and robust security, but often suffer from lower performance, limited scalability, and higher costs. In contrast, NoSQL databases, which generally follow the BASE model, provide superior scalability, higher performance, and the capacity to handle massive volumes of unstructured data. The study concluded that NoSQL systems align better with the requirements of contemporary database-driven applications.

Chieh Ming Wu et al. [23] evaluated the performance of MS-SQL and MongoDB for handling website messaging operations. Their experiments showed that MongoDB significantly outperformed MS-SQL. This aligns with the structural differences between the two: RDBMS require predefined schemas for structured data, while NoSQL allows flexible, schema-less storage.

Other studies have used YCSB to benchmark multiple NoSQL systems under varying workloads in non-distributed settings. One such evaluation [24] found that Tarantool delivered the best execution times across different load conditions. Similarly, Yusuf Abubakar et al. [25] compared OrientDB, Elasticsearch, and Redis under constrained

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

resources. Their results showed Redis as the most efficient in both read and write operations.

Wumuti Naheman et al. [26] analyzed the advantages and disadvantages of both NoSQL and relational systems, concluding that while NoSQL offers advantages, it does not always outperform RDBMS in every context. In their tests, HBase performed much slower in a single-machine setup compared to distributed environments. They stressed that database selection should be tailored to the specific needs of the application.

Cristina Băzăr et al. [27] compared NoSQL databases to identify those with the lowest latency, also highlighting reasons for transitioning from RDBMS to NoSQL. They found Couchbase to deliver the best latency for building highly interactive applications, with strong scalability, reliability, and consistency.

The concept of RDBMS was first introduced by E.F. Codd in 1970, with SQL as the query language. RDBMS excel at managing structured, table-based data and play a critical role in handling transactional workloads. In contrast, NoSQL databases vary in schema design, data storage, and architecture, allowing organizations to choose based on specific requirements. While RDBMS dominated in the 1970s, their limitations in handling today's ever-growing data volumes have driven adoption of NoSQL solutions [28].

When requirements demand ACID compliance and up-to-date data integrity, RDBMS remains the preferred choice. However, when dealing with large datasets requiring horizontal scaling across multiple distributed systems, NoSQL becomes more advantageous due to its scalability features [29]. For applications needing a combination of ACID compliance, integrity, scalability, and efficiency, either type may be suitable depending on the use case.

Pavel Seda et al. [30] compared Elasticsearch (NoSQL) and MySQL (RDBMS) to determine which is better suited for large-scale e-application development, focusing on space and consistency. Their benchmarks demonstrated that Elasticsearch was 5–6 times faster than MySQL for retrieving specific data. This study compares RDBMS and NoSQL databases across ten different features, summarized in Table 1.

Table 3-1: Difference between RDBMs & NoSQL Databases

Parameter	RDBMS	NoSQL Databases
Schemas	In relational databases, data is stored in a well-structured format. If a new product entry needs to be added to the software, the entire database schema must be altered, which typically requires taking the database offline during the update.	In contrast, NoSQL databases store data in an unstructured, schema-less format. New data can be added dynamically without interrupting the database's operation.
Scaling	Traditional relational databases are designed primarily for **vertical scaling**, meaning performance can	NoSQL databases support **horizontal scaling**, distributing workload and traffic across

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

	only be improved by adding more powerful hardware. As the number of users grows, vertical scaling becomes a costly and time-intensive process, and it may still fail to handle heavy traffic efficiently. High concurrent usage often results in performance degradation.	multiple nodes in a distributed environment [31]. This approach enables systems to serve a large number of concurrent users efficiently, as seen in large-scale services like Facebook and Amazon.
Big Data Handling	RDBMS solutions have limited capacity for managing large-scale datasets [22].	Distributed NoSQL systems are designed to handle massive volumes of data efficiently.
Complex Queries	RDBMS stores structured data in tables with rows and columns, requiring SQL (Structured Query Language) to manage complex queries, often involving multiple joins to retrieve data from different tables in a single view.	NoSQL databases are table-less, relying on document-oriented or key-value structures. Data is typically stored in formats like JSON or XML, eliminating the need for complex join operations and improving efficiency and consistency.
High Transactions Processing	RDBMS is highly suitable for applications requiring heavy transaction loads due to its **ACID** compliance, ensuring data integrity and stability.	While NoSQL can be used for transactional purposes, it is generally less suitable for complex, high-volume transactional workloads.
Cloud Environment	RDBMS is less optimal for cloud-based deployments because it scales vertically to a limited extent, which is neither stable nor cost-effective. Its rigid, predefined schema also limits flexibility.	NoSQL databases, being **non-ACID** compliant, are more adaptable to cloud environments, offering better scalability, high availability, and improved performance compared to RDBMS.
Data Model	In RDBMS, records are inserted according to a predefined schema. Data is stored in separate tables, and different data types are kept apart. Retrieving related data often requires join queries. For example, student records might be stored in one table, and classes in another, with joins used to link them. Any	NoSQL databases allow flexible data insertion without disrupting application functionality and without the need for predefined schemas. The data model can be modified at any time without downtime.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

	insertion or update operations may require downtime.	
Relationship Mapping of Data	Relationship mapping is a fundamental feature of RDBMS, supporting **one-to-one**, **one-to-many**, and **many-to-many** relationships, which simplifies data retrieval through queries.	NoSQL databases generally do not support relationship mapping. If the data is inherently relational, NoSQL may not be the optimal choice.
Data Migration Policy	In RDBMS, migrating a database often requires the use of MapReduce-like processes, which are not natively integrated and can be time-consuming, with potential risk of data loss.	NoSQL databases often have built-in migration functionality, making the process faster and more reliable. Using MapReduce, large datasets can be processed efficiently in three main steps: mapper, shard, and reducer.
Timely Support	Commercial RDBMS vendors provide professional, reliable, and timely technical support for deployment and troubleshooting.	NoSQL users typically rely on community-driven support and local user groups, which can still offer valuable assistance.

Methods and materials

The research methodology is composed of four primary stages, as illustrated in Figure 1, with each stage briefly described below.



Figure 1: Research Methodology

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Preliminary study

The first stage involved conducting a preliminary study to gather relevant information, identify the existing databases, and review prior work related to this research topic. This stage focused on selecting suitable databases, examining previous studies comparing various databases, and identifying the limitations in the current body of research.

Existing Databases: This study utilizes two major categories of database management systems: Relational and NoSQL. While multiple relational and NoSQL databases exist, three specific systems were selected for experimentation one relational database (**PostgreSQL**) and two non-relational databases (**CouchDB** and **MongoDB**).

Existing Literature on Database Comparison: An extensive review of previous research was conducted to assess the current knowledge base. As discussed in the preceding chapter, existing studies comparing different databases were critically examined, and key findings from these works were consolidated into the literature review.

Limitations of Existing Literature: Only a limited number of studies have addressed the performance evaluation of relational and NoSQL databases, particularly in non-distributed environments. Most of these works used minimal hardware configurations and relatively small datasets, leaving a gap in large-scale performance analysis.

Study design

The second stage in the research methodology was the design phase. This stage involved determining the specific methods and procedures for database model selection and dataset creation.

Database Model Selection: Given that the primary objective of this research is to compare and evaluate the performance of multiple databases, selecting the appropriate database models was a critical step in ensuring the validity and relevance of the results.

Dataset Collection: The next step in the design phase involved creating datasets. Two database collections **Users** and **Comments** were developed. The workload dataset was prepared in three different sizes: 5,000, 50,000, and 500,000 records. User comments were generated randomly to populate the datasets.

Study implementation

In this stage, experiments were executed using the prepared datasets, applying a range of database operations. The benchmark approach chosen during the design phase was implemented here. A cluster of five computers was configured, with one serving as the master node and the remaining four as slave nodes. This phase comprises several sub-steps, illustrated in
Figure 2 and explained in detail below.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

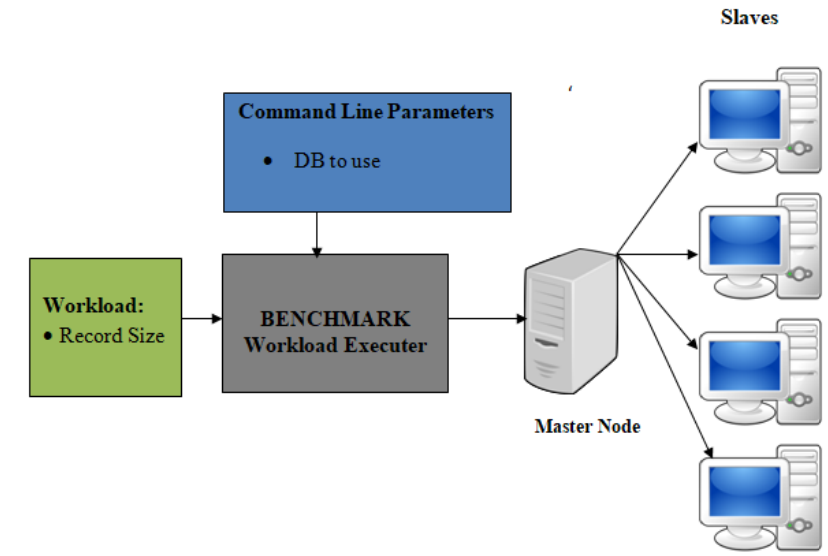


Figure 2: Experimental Clustering

Our experimental procedure focused on four primary CRUD operations **Insert, Update, Select, and Delete** executed on MongoDB, CouchDB, and PostgreSQL.

Insertion of Data: In the benchmark, new data was inserted into each database in three stages. Initially, a dataset containing 5,000 records was inserted, and the latency time was recorded. The process was repeated for datasets of 50,000 and 500,000 records, with latency measured at each stage to identify which database handled the operation most efficiently.

Updating of Records: Following data insertion, the second operation involved updating existing database records with new values. Workloads were processed in batches, and each CRUD operation (Insert, Update, Select, Delete) was only considered complete when all replica nodes in the distributed cluster matched the state of the master node. During insertion and updating, the replication process was automatically triggered on the master node, after which the updated data was synchronized across all slave nodes.

Selection of Data: The third operation involved retrieving data. Initially, 5,000 records were selected, and the latency (in milliseconds) was recorded to measure database efficiency. The same procedure was followed for datasets containing 50,000 and 500,000 records.

Deletion of Data: Finally, after insertion and updating, the delete operation was executed to remove specific records. This task was initiated on one cluster node, but due to the master–slave replication setup, deletion on the master node automatically propagated to all slave nodes. For instance, deleting a particular comment on the

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

master node instantly removed it from every slave node in the cluster.Our workload design consisted of two main components, explained as follows:

Database Size: The fundamental difference between the evaluated database management systems lies in how they manage data and scale. NoSQL databases are inherently more adaptable when dealing with very large datasets. To determine and compare the performance of each system, our benchmark executed tests on varying workload sizes and measured the execution time. The workload selection and database schema design played a critical role in ensuring fair and meaningful comparisons. Before running the benchmark, we created the table schemas according to the workload requirements. The database used in our experiment comprised two main tables: **users** and **comments**. The comments table contained all user-generated comments, with each comment linked to a user entry in the user's table.

Operation Count and Dataset: To assess performance, we conducted multiple CRUD operations, gradually increasing the load in steps. The throughput varied depending on the workload size. Our benchmark tested two collections **users** and **comments** where the comments collection stored remarks from various users.
The dataset was categorized into three sizes: **5,000**, **50,000**, and **500,000** records. CRUD operations were executed sequentially on each dataset in a distributed environment to determine which database demonstrated the lowest latency (response time). All datasets were processed and evaluated through our benchmark. To assess the performance of the three databases, we utilized a custom-developed benchmarking tool designed to be both flexible and standardized. This PHP-based tool was capable of handling various heavy workloads efficiently. The primary objective behind its development was to execute and evaluate multiple CRUD operations Create, Read, and Update across different workload sizes. The benchmark measured both latency and throughput for each workload scenario. It was implemented using the CodeIgniter framework, developed in PHP, and followed the MVC (Model-View-Controller) architecture [38]. The core functionality was contained within a database model

Results and Setup
The multi-node replication setup follows a master–slave architecture with several configuration options. In this arrangement, every server (node) maintains an identical copy of the dataset. One server acts as the **primary master node**, while the remaining ones function as **secondary nodes**. In MongoDB, a replica set ensures that all nodes retain synchronized copies of the data [39]. Although replication introduces a single point of failure, it significantly improves availability and supports high failover readiness.

Configuration of Databases
For this study, database versions that natively support clustering were selected. All three databases were installed on **Ubuntu 16.04**, and each cluster was configured with **five nodes**. The configurations for each database are as follows:

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

CouchDB Configuration: CouchDB, chosen from the NoSQL database family, is an open-source, object-relational database well-suited for handling heavy workloads. It is compatible with major operating systems and follows the **BASE** model. CRUD operations in CouchDB are executed using

JSON: Although CouchDB supports multiple replication methods, the **master–slave** approach was selected for this experiment. In this setup, a single master node handled all operations, which were automatically replicated to four slave nodes. If the master node failed, one of the slave nodes would automatically be promoted to master. Each slave maintained an up-to-date replica of the master database. For our experiments, a **five-node cluster** was deployed on Ubuntu 16.04 to achieve scalability and maintain consistency through replication.

MongoDB Configuration: MongoDB is a free, open-source, schema-less, document-oriented database written in C++. Like other NoSQL systems, it supports replication to ensure availability, scalability, and consistency. In our MongoDB replica set, one node acted as the **primary** while the others were **secondary** nodes. Write and read operations were directed to the primary node, while tasks such as backups were handled by secondary nodes. If the primary node became unavailable, the secondaries automatically elected a new primary and synchronized updates via asynchronous replication. MongoDB also employs locking mechanisms to maintain consistency, allowing multiple users to read and write concurrently.

PostgreSQL Configuration: PostgreSQL, an open-source relational database, uses **SQL** as its standard query language. Replication was configured using a **master–slave** model, distributing data across multiple machines. If the master server failed, a backup (secondary) server could take over.
For this study, **PostgreSQL version 9.6** was installed on each machine in the cluster. The master node supported both read and write operations, while the secondary nodes were read-only. The master was configured first, followed by the secondary nodes.

Cluster and Replication Performance: To assess the efficiency of the three databases in managing large datasets, each was scaled out to a **five-node master–slave cluster**. Replication performance was evaluated by varying the number of nodes within the cluster. This architecture proved well-suited for the workloads tested, as every node maintained an identical dataset and supported consistent operations across the cluster.

Hardware and Software Details
In this research, a performance evaluation was carried out for two database management systems (DBMSs). The experimental setup comprised a five-node cluster configured using replication. All nodes were equipped with identical hardware specifications and operated on the same operating system, as detailed in Table 3.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Table 3: Hardware and Software details of the Experiment

OS	Ubuntu (16.04) LTS x64
Ram	16 GB
Hard Disk	1 TB
CPU	2.4 GHz Intel Core i7
Number of Machines	5
Ethernet	Gigabit

All cluster nodes were connected within the same local area network (LAN) and assigned static IP addresses. For each database, a total of five machines were deployed using a **Master/Slave replication** setup. One node functioned as the **Master**, configured with master-specific settings, while the remaining four operated as **Slave nodes**, each with slave-specific configurations.

Operating System-Ubuntu (Desktop Version)

In this study, the choice of operating system played a significant role; therefore, **Ubuntu 16.04** was selected. Ubuntu is a community-driven, free, visually appealing, and open-source Linux distribution designed to make Linux more accessible to users [40]. Its primary objective is to offer a user-friendly and feature-rich Linux environment. It comes pre-installed with a wide range of up-to-date software, simplifying setup and use. A notable advantage of Ubuntu is its strong security it is largely immune to typical viruses and spyware, making it one of the most secure operating systems available. Additionally, Ubuntu benefits from a vast global user community, providing quick and helpful responses to queries. Its popularity stems from its security, stability, and suitability for both personal and server-side applications. While Ubuntu support is available free of charge, commercial support can be obtained at a reasonable cost for a defined period. As an enterprise-grade Linux distribution, it delivers a reliable and stable environment for server development, ensuring that applications remain secure and functional without unexpected failures.

The first experiment focused on bulk insert operations (write operations), where records were inserted in large batches instead of one by one to improve efficiency. This was followed by bulk read operations. The resulting benchmarks provided valuable insights into database performance.

For the evaluation, three datasets were used:

Dataset 1: 5,000 records

Dataset 2: 50,000 records

Dataset 3: 500,000 records

Each workload was executed **11 times**, and the **average execution time** was calculated. The performance of MongoDB, CouchDB, and PostgreSQL for different operations under varying workloads is detailed below.

Performance Analysis

CRUD operations were used to measure **throughput** and **latency** for each database in a replicated cluster. The analysis for different workloads is as follows:

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Workload A – Bulk Insert (5,000 Records)
In the first workload, 5,000 records were inserted into each database in a single bulk operation rather than one at a time. Bulk inserts generally provide better efficiency compared to individual inserts.

Table 3: Average Performance of three Databases in WRITE Operation

Databases	Time Latency (ms) 5,000 Records
Mongodb	**433.5**
Couchdb	**385**
PostgreSQL	**556**

From Table 3, it is evident that CouchDB outperformed the other two databases, showing the lowest average latency after inserting 5,000 records and achieving the highest throughput among them. The results indicate that CouchDB handled heavy workloads more efficiently. MongoDB ranked second, performing slightly better than PostgreSQL. In contrast, the relational database (PostgreSQL) exhibited the lowest throughput and the highest average latency compared to the NoSQL databases.

Workload B (Bulk Insert 50,000 Records)
In the second workload of our experiment, 50,000 records were inserted into each database in bulk rather than one at a time. With the increased workload, the following results were obtained regarding average latency.

Table 4: Average performance of the three databases in the WRITE operation

Databases	Time Latency (ms) 50,000 Records
Mongodb	**4570**
Couchdb	**3330.25**
PostgreSQL	**5956.5**

From Table 4, it is evident that CouchDB once again delivered the best performance among the three databases, maintaining the lowest average latency after inserting 50,000 records and achieving the highest throughput. MongoDB performed marginally better than PostgreSQL, while the relational database (PostgreSQL) recorded the lowest throughput and the highest average latency when compared with the NoSQL databases.

Workload C (Bulk Insert 500,000 Records)
In the third workload of our experiment, 500,000 records were inserted into each database in bulk rather than individually. With this significant increase in workload, the following results were obtained for average latency.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Table 5: Average performance of the three databases in the WRITE operation

Databases	Time Latency (ms) 500,000 Records
Mongodb	**97661.5**
Couchdb	**44387.75**
PostgreSQL	**67145.5**

From Table 5, it is evident that CouchDB once again outperformed the other two databases, delivering the lowest average latency after the insertion of 500,000 records and achieving the highest throughput. MongoDB ranked slightly above PostgreSQL in performance, while the relational database (PostgreSQL) recorded the lowest throughput and the highest average latency compared to the NoSQL databases.

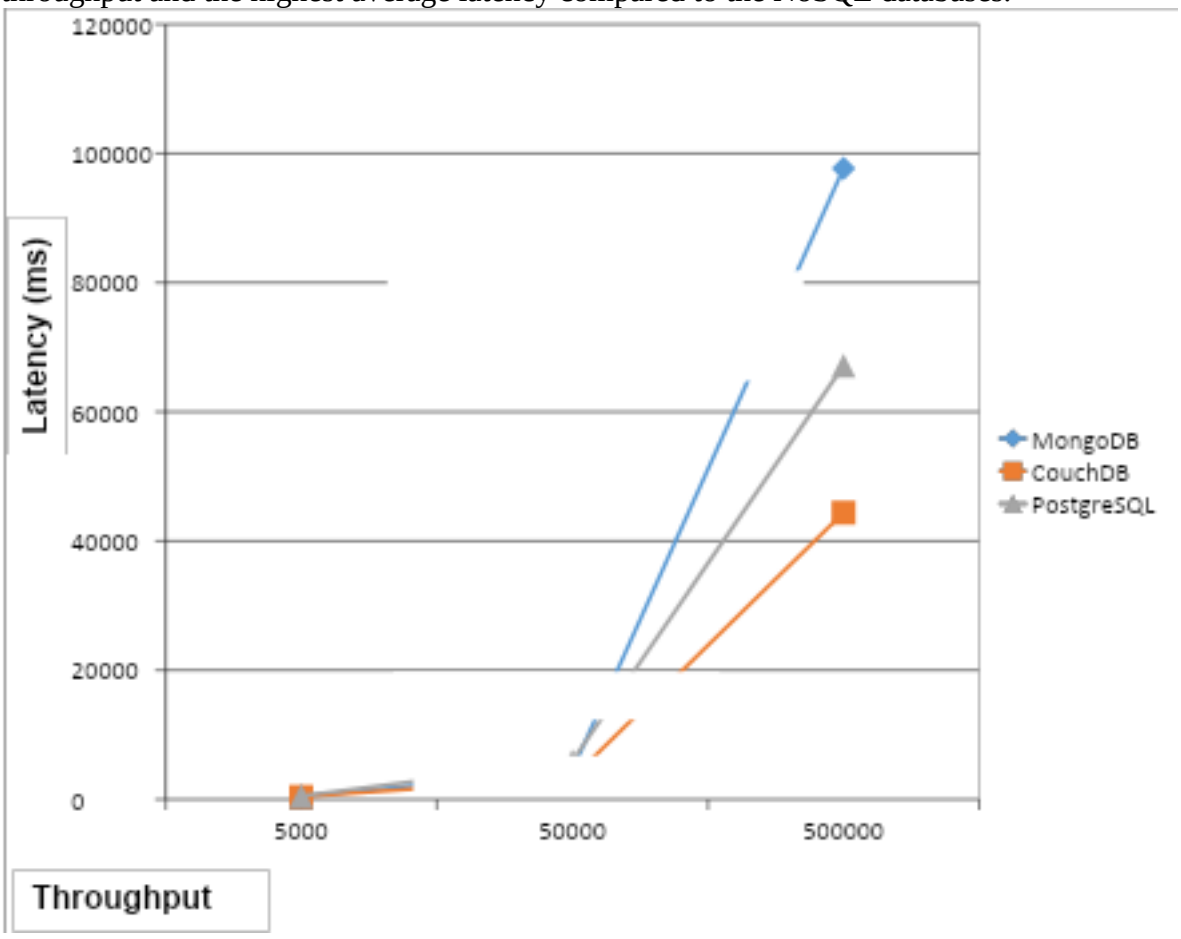


Figure 4: Write Operation of all three Databases on (500, 50,000 and 500,000)
The overall performance comparison of the three databases across three different dataset sizes for the insert operation is presented graphically in Figure 4.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Workload D (95% Read, 5% Write)
In Workload D, both MongoDB and CouchDB demonstrated low average latency and high throughput. For the UPDATE operation, CouchDB maintained superior performance, benefiting from its flexible design. MongoDB also showed better results than PostgreSQL in this workload.

Table 6: Average performance of the three databases after the UPDATE operation

Databases	Time Latency (ms) 5,000 Records
Mongodb	**390**
Couchdb	**272**
PostgreSQL	**444.5**

From Table 6, it is evident that both NoSQL databases delivered superior performance in the UPDATE operation compared to PostgreSQL within a distributed cluster environment.

Workload E (95% Read, 5% Write)
In this workload, MongoDB and CouchDB achieved the lowest average latency and the highest throughput. Once again, CouchDB demonstrated better performance in the UPDATE operation, benefiting from its flexible architecture. MongoDB also outperformed PostgreSQL in this scenario.

Table 7: Average performance of the three databases after the UPDATE operation

Databases	Time Latency (ms) 50,000 Records
Mongodb	**2916**
Couchdb	**1114.8**
PostgreSQL	**19662.5**

From Table 7, it is evident that both NoSQL databases outperformed PostgreSQL in the UPDATE operation within a distributed cluster environment.
Workload F (95% Read, 5% Write) In this workload, MongoDB and CouchDB recorded the lowest average latency and the highest throughput. CouchDB once again led in the UPDATE operation, benefiting from its flexible design, while MongoDB also delivered better results than PostgreSQL.

Table 8: Average Performance of three Databases after the UPDATE operation

Databases	Time Latency (ms) 500,000 Records
Mongodb	**48708.5**
Couchdb	**30657.75**
PostgreSQL	**210101.5**

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Table 8 clearly shows that both NoSQL databases achieved superior performance in the UPDATE operation compared to PostgreSQL within a distributed cluster environment.

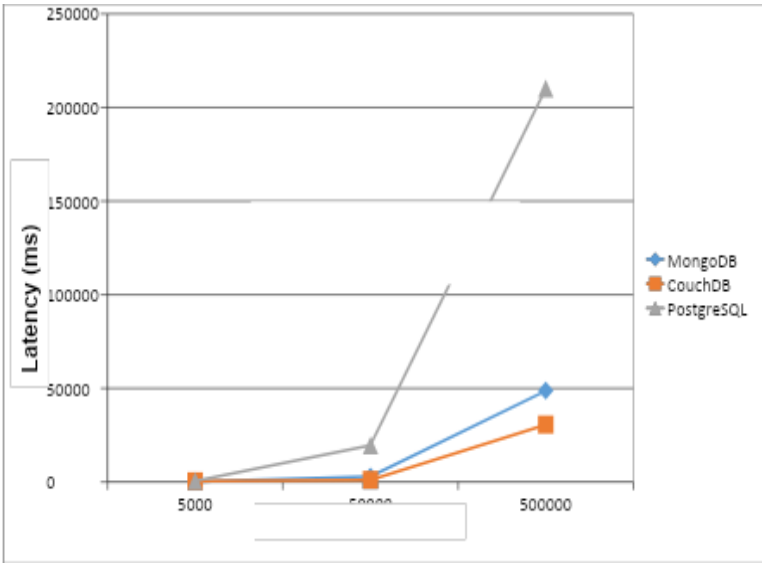


Figure 5 Update Operation of all three Databases on (500, 50,000, and 500,000 Records)

The overall performance comparison of the three databases for the update operation, using datasets of varying sizes, is illustrated in Figure 5.

Workload G (95% read & 5% Delete)

In Workload G, both MongoDB and CouchDB exhibited low average latency and high throughput. The DELETE operation requires scanning the entire database; however, MongoDB achieved the lowest latency among the three databases. CouchDB also performed better than PostgreSQL in this operation.

Table 9: Average performance of the three databases after the DELETE operation

Databases	Time Latency (ms) 5,000 Records
Mongodb	**132**
Couchdb	**90**
PostgreSQL	**434**

From Table 9 it is evident that both NoSQL databases outperformed PostgreSQL in the DELETE operation within a distributed cluster environment.

Workload H (95% Read, 5% Delete)

In this workload, MongoDB and CouchDB achieved low average latency and high throughput. Since the DELETE operation involves reading the entire database, MongoDB recorded the lowest latency among the three databases. CouchDB also

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

demonstrated better performance compared to PostgreSQL.

Table 10: Average performance of the three databases after the DELETE operation

Databases	Time Latency (ms) 50,000 Records
Mongodb	**1607**
Couchdb	**800.65**
PostgreSQL	**10203**

\From Table 10, it is evident that both NoSQL databases delivered superior performance in the DELETE operation compared to PostgreSQL in a distributed cluster environment.

Workload I (95% Read, 5% Delete)

In this workload, MongoDB and CouchDB demonstrated low average latency and high throughput. As the DELETE operation requires scanning the entire database, MongoDB achieved the lowest latency among the three. CouchDB also outperformed PostgreSQL in this operation.

Table 11: Average performance of the three databases after the DELETE operation

Databases	Time Latency (ms) 500,000 Records
Mongodb	**39073**
Couchdb	**11436.1**
PostgreSQL	**82326**

Table 12 clearly indicates that both NoSQL databases outperformed PostgreSQL in the DELETE operation within a distributed cluster environment.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

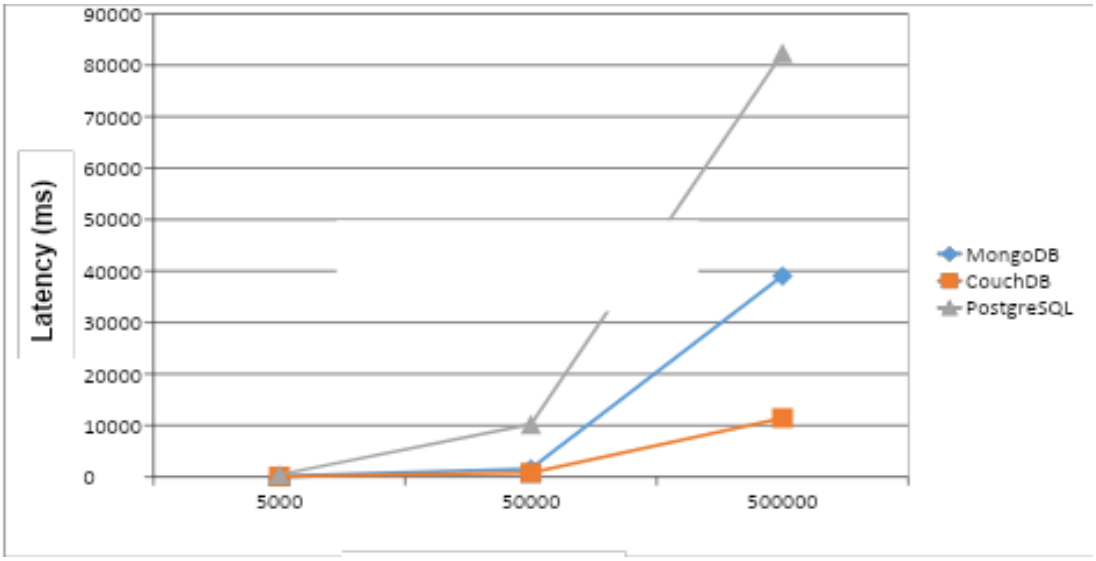


Figure 12: Delete Operation of All Three Databases on (500, 50,000, and 500,000 Records)

The overall performance comparison of the three databases for the delete operation, using datasets of varying sizes, is illustrated in Figure 12.

Workload J (100% Read with 5,000 Records): In this workload, both MongoDB and CouchDB recorded low average latency and high throughput. However, in the SELECT operation, MongoDB achieved the lowest latency among the three databases. CouchDB also performed better than PostgreSQL.

Table 12: Average performance of the three databases after the SELECT operation

Databases	Time Latency (ms) 5000 Records
Mongodb	**5**
Couchdb	**10.7**
PostgreSQL	**18.5**

From Table 12, it is evident that both NoSQL databases outperformed PostgreSQL in the SELECT operation within a distributed cluster environment.

Workload K (100% Read with 50,000 Records): In this workload, MongoDB and CouchDB demonstrated low average latency and high throughput. Overall, MongoDB achieved the lowest latency in the SELECT operation among the three databases, while CouchDB also showed better performance compared to PostgreSQL.

Table 13: Average performance of the three databases after the SELECT operation

Databases	Time Latency (ms) 50,000 Records
Mongodb	**30**
Couchdb	**152.2**
PostgreSQL	**97**

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

From Table 13, it is evident that both NoSQL databases delivered superior performance in the SELECT operation compared to PostgreSQL in a distributed cluster environment.

Workload L (100% Read with 500,000 Records): In this workload, MongoDB and CouchDB achieved low average latency and high throughput. Overall, MongoDB recorded the lowest latency in the SELECT operation among the three databases, while CouchDB also outperformed PostgreSQL.

Table 14: Average performance of the three databases after the SELECT operation

Databases	Time Latency (ms) 500,000 Records
Mongodb	**217.5**
Couchdb	**1537.7**
PostgreSQL	**3317.5**

Table 14 clearly shows that both NoSQL databases outperformed PostgreSQL in the SELECT operation within a distributed cluster environment.

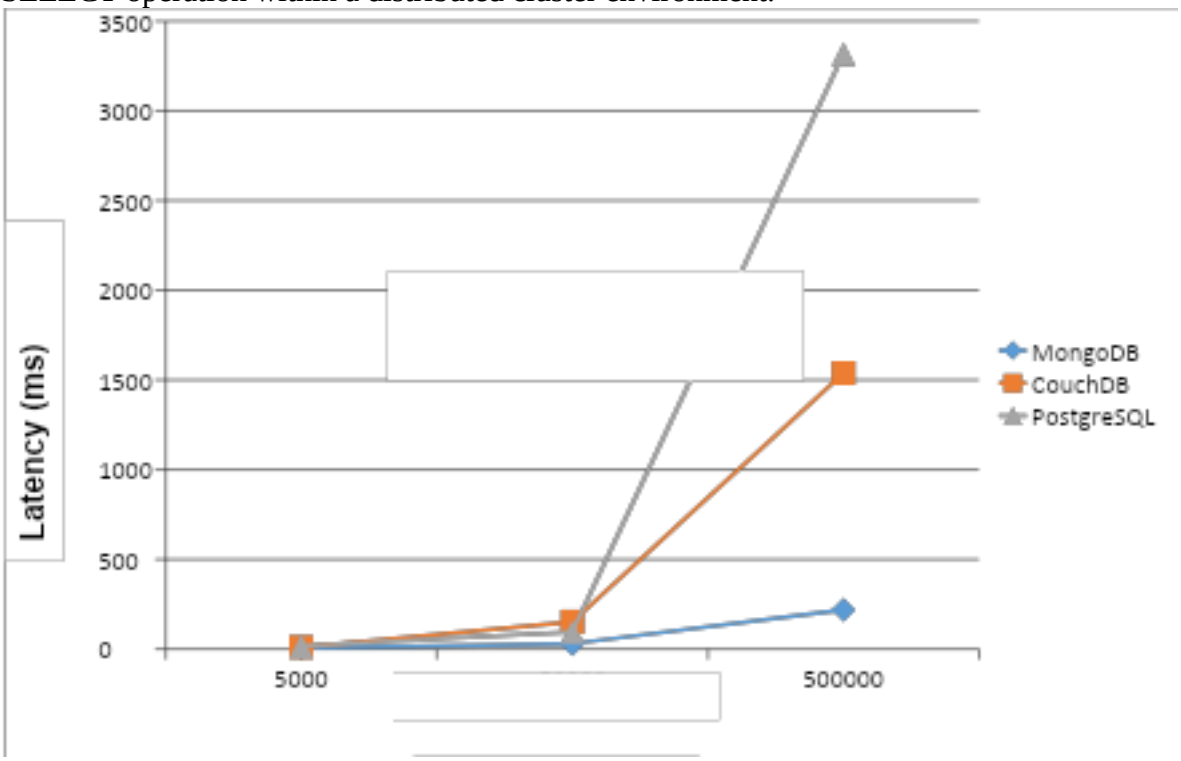


Figure 13: Select Operation of All Three Databases on (500, 50,000, and 500,000 Records)

The overall performance comparison of the three databases for the SELECT operation, using datasets of different sizes, is presented in graphical form in Figure 13.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Conclusion

In application development, the database plays a crucial role. Each database comes with its own set of features and design patterns, making it essential to assess their efficiency to determine the most suitable option for a specific application. In this research, two open-source NoSQL databases (CouchDB and MongoDB) and one relational database (PostgreSQL) were evaluated. The experimental results revealed that the NoSQL databases (CouchDB and MongoDB) outperformed the relational PostgreSQL (RDBMS) system. Various CRUD operations were conducted on different datasets in a distributed environment using master–slave replication. Overall, CouchDB delivered the best performance in terms of read, write, and update operations, and proved to be faster than MongoDB among document-oriented databases. NoSQL databases demonstrated strong performance in scenarios requiring high scalability, efficiency, and consistency. This study also provided insights into the behavior of NoSQL databases in a distributed environment, showing that their performance improves over time as the dataset grows. This research indicates that NoSQL databases are highly suitable in scenarios where the database must scale seamlessly as data volume increases. In future work, this experiment can be extended using much larger datasets. Performance evaluation could also be carried out in a geographically distributed environment, allowing for assessment when data is spread across multiple machines located in different network locations. Such an approach would provide deeper insights into database behavior under real-world distributed conditions.

Reference

Khan, S.U.R., Asif, S., Bilal, O. et al. Lead-cnn: lightweight enhanced dimension reduction convolutional neural network for brain tumor classification. Int. J. Mach. Learn. & Cyber. (2025). https://doi.org/10.1007/s13042-025-02637-6.

Khan, S. U. R., Asim, M. N., Vollmer, S., & Dengel, A. (2025). Robust & Precise Knowledge Distillation-based Novel Context-Aware Predictor for Disease Detection in Brain and Gastrointestinal. arXiv preprint arXiv:2505.06381..

Hekmat, A., et al., Brain tumor diagnosis redefined: Leveraging image fusion for MRI enhancement classification. Biomedical Signal Processing and Control, 2025. 109: p. 108040.

Khan, Z., Hossain, M. Z., Mayumu, N., Yasmin, F., & Aziz, Y. (2024, November). Boosting the Prediction of Brain Tumor Using Two Stage BiGait Architecture. In 2024 International Conference on Digital Image Computing: Techniques and Applications (DICTA) (pp. 411-418). IEEE.

Khan, S. U. R., Raza, A., Shahzad, I., & Ali, G. (2024). Enhancing concrete and pavement crack prediction through hierarchical feature integration with VGG16 and triple classifier ensemble. In 2024 Horizons of Information Technology and Engineering (HITE)(pp. 1-6). IEEE https://doi. org/10.1109/HITE63532.

Khan, S.U.R., Zhao, M. & Li, Y. Detection of MRI brain tumor using residual skip block based modified MobileNet model. Cluster Comput 28, 248 (2025). https://doi.org/10.1007/s10586-024-04940-3

Khan, U. S., & Khan, S. U. R. (2024). Boost diagnostic performance in retinal disease

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

classification utilizing deep ensemble classifiers based on OCT. Multimedia Tools and Applications, 1-21.

Raza, A., & Meeran, M. T. (2019). Routine of encryption in cognitive radio network. Mehran University Research Journal of Engineering & Technology, 38(3), 609-618.

Al-Khasawneh, M. A., Raza, A., Khan, S. U. R., & Khan, Z. (2024). Stock Market Trend Prediction Using Deep Learning Approach. Computational Economics, 1-32.

Khan, U. S., Ishfaque, M., Khan, S. U. R., Xu, F., Chen, L., & Lei, Y. (2024). Comparative analysis of twelve transfer learning models for the prediction and crack detection in concrete dams, based on borehole images. Frontiers of Structural and Civil Engineering, 1-17.

Khan, S. U. R., & Asif, S. (2024). Oral cancer detection using feature-level fusion and novel self-attention mechanisms. Biomedical Signal Processing and Control, 95, 106437.

Asim, M. N., Ibrahim, M. A., Malik, M. I., Dengel, A., & Ahmed, S. (2020). Enhancer-dsnet: a supervisedly prepared enriched sequence representation for the identification of enhancers and their strength. In Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 23–27, 2020, Proceedings, Part III 27 (pp. 38-48). Springer International Publishing.

Khan, M. A., Khan, S. U. R., Rehman, H. U., Aladhadh, S., & Lin, D. (2025). Robust InceptionV3 with Novel EYENET Weights for Di-EYENET Ocular Surface Imaging Dataset: Integrating Chain Foraging and Cyclone Aging Techniques. International Journal of Computational Intelligence Systems, 18(1), 1-26.

Dai, Q., Ishfaque, M., Khan, S. U. R., Luo, Y. L., Lei, Y., Zhang, B., & Zhou, W. (2024). Image classification for sub-surface crack identification in concrete dam based on borehole CCTV images using deep dense hybrid model. Stochastic Environmental Research and Risk Assessment, 1-18.

Muhammad, N. A., Rehman, A., & Shoaib, U. (2017). Accuracy based feature ranking metric for multi-label text classification. International Journal of Advanced Computer Science and Applications, 8(10).

Mehmood, F., Ghafoor, H., Asim, M. N., Ghani, M. U., Mahmood, W., & Dengel, A. (2024). Passion-net: a robust precise and explainable predictor for hate speech detection in roman urdu text. Neural Computing and Applications, 36(6), 3077-3100.

Khan, S.U.R.; Asif, S.; Bilal, O.; Ali, S. Deep hybrid model for Mpox disease diagnosis from skin lesion images. Int. J. Imaging Syst. Technol. 2024, 34, e23044.

Khan, S.U.R.; Zhao, M.; Asif, S.; Chen, X.; Zhu, Y. GLNET: Global–local CNN's-based informed model for detection of breast cancer categories from histopathological slides. J. Supercomput. 2023, 80, 7316–7348.

Saleem, S., Asim, M. N., Van Elst, L., & Dengel, A. (2023). FNReq-Net: A hybrid computational framework for functional and non-functional requirements classification. Journal of King Saud University-Computer and Information Sciences, 35(8), 101665.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Hekmat, Arash, Zuping Zhang, Saif Ur Rehman Khan, Ifza Shad, and Omair Bilal. "An attention-fused architecture for brain tumor diagnosis." Biomedical Signal Processing and Control 101 (2025): 107221.

Khan, S.U.R.; Zhao, M.; Asif, S.; Chen, X. Hybrid-NET: A fusion of DenseNet169 and advanced machine learning classifiers for enhanced brain tumor diagnosis. Int. J. Imaging Syst. Technol. 2024, 34, e22975.

Shahzad, I., Khan, S. U. R., Waseem, A., Abideen, Z. U., & Liu, J. (2024). Enhancing ASD classification through hybrid attention-based learning of facial features. Signal, Image and Video Processing, 1-14.

Khan, S. U. R., Asif, S., Zhao, M., Zou, W., Li, Y., & Xiao, C. (2026). ShallowMRI: A novel lightweight CNN with novel attention mechanism for Multi brain tumor classification in MRI images. Biomedical Signal Processing and Control, 111, 108425.

Khan, S. U. R., Asif, S., Zhao, M., Zou, W., Li, Y., & Li, X. (2025). Optimized deep learning model for comprehensive medical image analysis across multiple modalities. Neurocomputing, 619, 129182.

Khan, S. U. R., Asif, S., Zhao, M., Zou, W., & Li, Y. (2025). Optimize brain tumor multiclass classification with manta ray foraging and improved residual block techniques. Multimedia Systems, 31(1), 1-27.

Raza, Asif, Inzamam Shahzad, Muhammad Salahuddin, and Sadia Latif. "Satellite Imagery Employed to Analyze the Extent of Urban Land Transformation in The Punjab District of Pakistan." Journal of Palestine Ahliya University for Research and Studies 4, no. 2 (2025): 17-36.

Waqas, Muhammad, Muhammad Atif Tahir, Sumaya Al-Maadeed, Ahmed Bouridane, and Jia Wu. "Simultaneous instance pooling and bag representation selection approach for multiple-instance learning (MIL) using vision transformer." Neural Computing and Applications 36, no. 12 (2024): 6659-6680.

Latif, Aafia, Sadia Latif, and Furqan Jamil. "UTILIZING BIG DATA ANALYTICS TO OPTIMIZE INFORMATION COMMUNICATION TECHNOLOGY STRATEGIES." Kashf Journal of Multidisciplinary Research 1, no. 12 (2024): 122-140.

Khan, S. R., Asif Raza, Inzamam Shahzad, & Hafiz Muhammad Ijaz. (2024). Deep transfer CNNs models performance evaluation using unbalanced histopathological breast cancer dataset. Lahore Garrison University Research Journal of Computer Science and Information Technology, 8(1).

O. Bilal, Asif Raza, S. ur R. Khan, and Ghazanfar Ali, “A Contemporary Secure Microservices Discovery Architecture with Service Tags for Smart City Infrastructures ”, VFAST trans. softw. eng., vol. 12, no. 1, pp. 79–92, Mar. 2024

Waqas, Muhammad, Syed Umaid Ahmed, Muhammad Atif Tahir, Jia Wu, and Rizwan Qureshi. "Exploring multiple instance learning (MIL): A brief survey." Expert Systems with Applications 250 (2024): 123893.

S. ur R. Khan, Asif. Raza, Muhammad Tanveer Meeran, and U. Bilhaj, “Enhancing Breast Cancer Detection through Thermal Imaging and Customized 2D CNN Classifiers”, VFAST trans. softw. eng., vol. 11, no. 4, pp. 80–92, Dec. 2023.

Ashraf, M., Jalil, A., Salahuddin & Jamil, F. (2024). DESIGN AND

Online ISSN 3007-3197
Print ISSN 3007-3189

IMPLEMENTATION OF ERROR ISOLATION IN TECHNO METER. Kashf Journal of Multidisciplinary Research, 1(12), 49-66.

M. Waqas, Z. Khan, S. U. Ahmed and A. Raza, "MIL-Mixer: A Robust Bag Encoding Strategy for Multiple Instance Learning (MIL) using MLP-Mixer," 2023 18th International Conference on Emerging Technologies (ICET), Peshawar, Pakistan, 2023, pp. 22-26.

Mahmood, F., Abbas, K., Raza, A., Khan,M.A., & Khan, P.W. (2019). Three Dimensional Agricultural Land Modeling using Unmanned Aerial System (UAS). International Journal of Advanced Computer Science and Applications (IJACSA) [p-ISSN : 2158-107X, e-ISSN : 2156-5570], 10(1).

Meeran, M. T., Raza, A., & Din, M. (2018). Advancement in GSM Network to Access Cloud Services. Pakistan Journal of Engineering, Technology & Science [ISSN: 2224-2333], 7(1).

Khan, S. U. R., Asim, M. N., Vollmer, S., & Dengel, A. (2025). AI-Driven Diabetic Retinopathy Diagnosis Enhancement through Image Processing and Salp Swarm Algorithm-Optimized Ensemble Network. arXiv preprint arXiv:2503.14209.

Khan, Z., Khan, S. U. R., Bilal, O., Raza, A., & Ali, G. (2025, February). Optimizing Cervical Lesion Detection Using Deep Learning with Particle Swarm Optimization. In 2025 6th International Conference on Advancements in Computational Sciences (ICACS) (pp. 1-7). IEEE.

Khan, S.U.R., Raza, A., Shahzad, I., Khan, S. (2025). Subcellular Structures Classification in Fluorescence Microscopic Images. In: Arif, M., Jaffar, A., Geman, O. (eds) Computing and Emerging Technologies. ICCET 2023. Communications in Computer and Information Science, vol 2056. Springer, Cham. https://doi.org/10.1007/978-3-031-77620-5_20

Hekmat, A., Zuping, Z., Bilal, O., & Khan, S. U. R. (2025). Differential evolution-driven optimized ensemble network for brain tumor detection. International Journal of Machine Learning and Cybernetics, 1-26.

Khan, S. U. R. (2025). Multi-level feature fusion network for kidney disease detection. Computers in Biology and Medicine, 191, 110214.

Khan, S. U. R., Asif, S., & Bilal, O. (2025). Ensemble Architecture of Vision Transformer and CNNs for Breast Cancer Tumor Detection From Mammograms. International Journal of Imaging Systems and Technology, 35(3), e70090.

Khan, S. U. R., & Khan, Z. (2025). Detection of Abnormal Cardiac Rhythms Using Feature Fusion Technique with Heart Sound Spectrograms. Journal of Bionic Engineering, 1-20.

Khan, M.A., Khan, S.U.R. & Lin, D. Shortening surgical time in high myopia treatment: a randomized controlled trial comparing non-OVD and OVD techniques in ICL implantation. BMC Ophthalmol 25, 303 (2025). https://doi.org/10.1186/s12886-025-04135-3

Shahzad, I., Raza, A., & Waqas, M. (2025). Medical Image Retrieval using Hybrid Features and Advanced Computational Intelligence Techniques. Spectrum of engineering sciences, 3(1), 22-65.

Raza, A., Shahzad, I., Ali, G., & Soomro, M. H. (2025). Use Transfer Learning VGG16, Inception, and Reset50 to Classify IoT Challenge in Security Domain

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

via Dataset Bench Mark. Journal of Innovative Computing and Emerging Technologies, 5(1).

Raza, A., & Shahzad, I. (2024). Stock Market Trend COVID-19 Using Chest Radiographs. Spectrum of engineering sciences, 2(3), 367-396.

[48] Raza, A., Soomro, M. H., Shahzad, I., & Batool, S. (2024). Abstractive Text Summarization for Urdu Language. Journal of Computing & Biomedical Informatics, 7(02).

HUSSAIN, S., RAZA, A., MEERAN, M. T., IJAZ, H. M., & JAMALI, S. (2020). Domain Ontology Based Similarity and Analysis in Higher Education. IEEEP New Horizons Journal, 102(1), 11-16.

Raza, A., & Meeran, M. T. (2019). Routine of encryption in cognitive radio network. Mehran University Research Journal of Engineering & Technology, 38(3), 609-618.

Khan, U. S., & Khan, S. U. R. (2025). Ethics by Design: A Lifecycle Framework for Trustworthy AI in Medical Imaging From Transparent Data Governance to Clinically Validated Deployment. arXiv preprint arXiv:2507.04249.

Maqsood, H., & Khan, S. U. R. (2025). MeD-3D: A Multimodal Deep Learning Framework for Precise Recurrence Prediction in Clear Cell Renal Cell Carcinoma (ccRCC). arXiv preprint arXiv:2507.07839.

Khan, S. U. R., Asim, M. N., Vollmer, S., & Dengel, A. (2025). FOLC-Net: A Federated-Optimized Lightweight Architecture for Enhanced MRI Disease Diagnosis across Axial, Coronal, and Sagittal Views. arXiv preprint arXiv:2507.06763.

Bilal, O., Hekmat, A., & Khan, S. U. R. (2025). Automated cervical cancer cell diagnosis via grid search-optimized multi-CNN ensemble networks. Network Modeling Analysis in Health Informatics and Bioinformatics, 14(1), 67.