Scalable Data Engineering Pipelines for Real-Time Analytics in Big Data Environments

Reddy Srikanth Madhuranthakam[1,*]

[1]Department of AI in DevSecOps-FAMC, Citizens Bank, Texas, United States of America.
reddysrikanth.madhuranthakamseshachalam@citizensbank.com[1]

Abstract: With the world becoming increasingly data-driven, actionable insights and decisions that depend on real-time analytics have been at the forefront. However, processing huge volumes of data in real time requires very strong, scalable, and efficient data engineering pipelines. This paper describes the design, development, and optimization of scalable data engineering pipelines for real-time analytics in big data environments. Ingestion, processing, storage and visualization, along with their interactions within the distributed computing setup, will all be part of the paper. More best practices will be presented regarding handling high-velocity data streams to become fault-tolerant and data consistency. We design a system architecture using the empirical approach: bringing real-time data processing frameworks like Apache Kafka and Apache Flink with the best available cloud-based storage solution to achieve the scale-out seamlessness of the data processing task. Alongside the design comes the performance evaluation results for a comparison among varied strategies to be followed while carrying out real-time analytics focusing on scalability, throughput, and latency. It indicates that the efficiency of the proposed pipeline is much improved in processing and is well-suited for large-scale, real-time analytics for numerous industries.

Keywords: Data Engineering Pipelines; Real-Time Analytics; Big Data; Distributed Computing; Data Processing Frameworks; Real-Time Data Processing; Storage and Visualization; Support Decision-Making.

Received on: 12/04/2024, **Revised on:** 26/06/2024, **Accepted on:** 19/08/2024, **Published on:** 09/09/2024

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCS

DOI: https://doi.org/10.69888/FTSCS.2024.000262

Cite as: R. S. Madhuranthakam, "Scalable Data Engineering Pipelines for Real-Time Analytics in Big Data Environments," *FMDB Transactions on Sustainable Computing Systems*, vol. 2, no. 3, pp. 154–166, 2024.

1. Introduction

Explosive growth in data generation will be the final result of the advancements happening in IoT, social media, and enterprise systems, which ultimately generate a big data environment. The extraction of real-time insights by using big data from this mass, which is very often unstructured data, will pose specific challenges to businesses and organizations. Real-time analytics of finance, health care, and e-commerce will be revolutionary since data can be used when it is under production to support decision-making [18]. Scaling data engineering pipelines to handle a stream of big data environments can be very complex [19]. It is a scalable data pipeline designed with architecture capable of handling voluminous amounts of growing data without a proportional increase in processing time and computational costs [20]. It offers scalability that helps make it further possible: it can then be shaped into effective systems for taking good care of efficient processing in collaboration with real-time analytics

*Corresponding author.

that will define the character of the environment as voluminous, heterogeneous, and increasingly at velocity [21]. Thus, it gives the capacity to add streams of ever-growing data, and it enhances keeping without experiencing any performance losses. The corporations continue collecting, and so do streams [1].

The other major challenge lies in the identification of an appropriate set of tools and frameworks to use toward the design of the pipeline [22]. A pipe in real-time should offer very low latency along with good throughput and possible fault tolerance capabilities [23]. This design would require horizontal scalability for pipelines, such that computing resources, like storage resources, should scale up only when needed and not necessarily with a redesign of the overall architecture [24]. For instance, ever increasing demand for Apache Kafka and Apache Flink is now mainly because of the former's ability to process large amounts of data in a distributed way at very low latencies, but which one of these two tools is more in demand is not the point here [2]. Its requirements shall be assessed based on the needs of the use case [25]. For example, what data are being processed, at what rate, with what complexity, and at what point should be established? A pipeline must ingest input and process so as to smooth its output together with the up-scaling on volume [26]. It should be flexible as well concerning integrating sources, types of data, and openness to business change requirements [3].

This is about the best practices and principles in designing scalable pipelines in data engineering, primarily focusing on real-time analytics [27]. This paper explores what engineers should focus on while designing pipelines that should be efficient and deliver real-time insights without any delay at all [28]. The concept is to provide the entire framework for the creation of scalable pipelines that can be used in high-performance analytics as part of creating imperative assets for all data-driven decision-making processes [4]. Pipelines of data engineering can be broadly classified into ingestion, processing, storage, and output - analytics or visualization [29].

Real-time data pipelines demand the ingesting of data as it's being produced, processing it at high speeds, and getting results in near real-time [30]. A few of the very critical factors to be managed with extreme care are those that deliver optimal performance with reliability in real-time data pipelines, especially in big data environments [31]. Among them is latency, which is the time elapsed before actionable insight or results from the inputted data can be displayed. Real-time analytics must introduce minimal latency [32]. Tiny delays might trigger spectacular cascading effects in fraud detection systems or online recommendation systems and even during financial transactions [33]. The pipeline will question the efficiency, and in the case of extremely high latency, it could decide based on obsolete information or irrelevant [5].

The second critical aspect would be throughput. Throughput will determine how many data points are processed within a unit of time [34]. The throughput is usually reported in records per second; higher throughput is indispensable for processing more streams of data and can, therefore, avoid bottlenecks [35]. An important point here is fault tolerance; the system should keep working even if parts of it start failing. Failures are an inherent feature of large-scale distributed systems and require replication, redundancy, and automated recovery mechanisms that should ensure continued uninterrupted operation with minimal data loss or downtime [36]. Otherwise, without good fault tolerance mechanisms, the pipeline will stall critical analytics that becomes unreliable [6].

The gravest problem that comes with real-time analytics in the context of large distributed systems is consistency. Distributed data processing resides on multiple nodes or servers. It requires the original data copies to be in a similar state [37]. Inconsistent data may lead to wrong results and can adversely influence decision-making, which can compromise the integrity of analytics [38]. Algorithms and frameworks for consistency include eventual consistency algorithms or distributed consensus algorithms like Paxos or Raft, all of which complicate the pursuit of this aim [7]. Other sources of data also differ in formats, schemas, or frequencies in which the updates take place, hence complicating the consistency of these various data sets [8].

Real-time complexity brings in issues that combine low latency, high throughput, fault tolerance, and data consistency with scalability and adaptability for variable workloads [39]. Meeting demands in analytics and the big-data environment of today requires appropriately distributed frameworks, choice of data-processing architecture, appropriate recovery strategies in case of faults, monitoring, and ongoing optimization [40]. This is all within pipelines that apply to diverse industrial environments [9]. This paper reviews the techniques, frameworks, and architectures that are currently used in big data environments for scalable real-time data pipelines. The paper investigates best practices in designing and optimizing large-scale real-time systems based on a combination of theory and empirics. This research discusses the implementation of TinyML algorithms, which are applied to manage big data in IoT ecosystems, mainly where particular emphasis lies on energy efficiency and real-time analytics [15]. Highly sophisticated forms of big data related to medicine are required to be used when deep learning techniques are applied to determine solutions to management problems of health systems' security issues [6].

Other relevant factors include management systems, particularly those for the Hadoop ecosystem. Research studies on the methodology of performance evaluation involve benchmarking that forms part of the research in handling big data management within the Hadoop framework [11]. Also, such systems demand cloud storage solutions for massive amounts of data typically required and particular mechanisms in protecting sensitive information related to security and privacy issues [16].

So far, two technologies have been highly utilized in big data environments. Comparisons are drawn on benchmarking results based on the relative strengths and weaknesses of performance on classification tasks [17]. The more a corporation utilizes these technologies, the more important it is to know fine-grained details regarding performance trade-offs of the technologies concerned with optimizing big data operation. In addition, with the higher usage of AI and machine learning in the auto-interpretation of data, it isn't easy to control the environment of big data itself [12]; [13].

2. Review of Literature

Wang et al. [2] observed that data engineering pipelines have witnessed tremendous changes in the last decade, mainly due to the increased demand for real-time analytics in big data environments. Most of the data processing pipelines were initially batch-based, wherein data was collected in predefined chunks, and at scheduled intervals, the processing was done. This was adequate for historical-analyzing applications whereby the result of past data understanding is crucial but need not necessarily be new. It helped to amass vast quantities of data that would then be processed in batches. It was computation-intensive yet slow. With the increasing applications of these "real-time data," their dependency on such applications in instantaneous decision-making processes made it realized that there was a need beyond batch processing in programs that required speedy and continuous processes to gain insight [41]. For example, business enterprises like finance, healthcare, and e-commerce were simply destined to deal with "live data streams" in real-time. This requirement would need to request data streaming pipelines, in that they deal with processing data in streams, meaning companies ought to respond to the information as it is coming. Data streaming pipelines are built to form streams with the intention of developing them as continuous streaming and being high-velocity, which would make the organization scan through data and give appropriate responses [42].

Günther et al. [3] explored that applied distributed computing, cloud technology and other high-end frameworks such as Apache Kafka, Apache Flink, and Apache Storm to help change static systems into paradigm-shifting applications. It has optimized ingestion, processing, and output for real-time, meaning it can process massive streams of data quite easily. To deal with some of the most important challenges, such as low latency, high throughput, and fault tolerance, which are really key for real-time analytics, new techniques and architectures had to be developed. In-memory computing should process data at a low latency level; this has appeared in the form of the data being kept in memory rather than disk; the resultant, in any case, is drastically faster processing times [43]. On the other side, high throughput ensures huge quantities of data are processed with no type of bottlenecks. Failure tolerance mechanisms ensure the whole system continues processing even with failures. Besides, with heterogeneous data sources, unstructured data, and multimedia content, such pipelines provide tremendous flexibility and scalability. Apart from changes in sources and formats, data-pipelined systems have to remain reliable and real-time as well [44].

According to Al-Ali et al. [5], "data can be used in an organization in very different manners. The data engineering function changed from batch to streaming data pipelines". This has enabled business organizations to gain real-time insights from the activities as they stream in feeding into the system for proactive decision making, real-time fraud detection, supply chain optimizations, and experience enhancement of the customers in ways impossible by the batch-based method. Streaming data pipelines form the backbone of modern data engineering and are increasingly supporting an ever more data-driven world.

Alsolbi et al. [7], Perform the data acquisition, processing, and analysis in real-time. This is what allows businesses and organizations to act on information up to their date. Perhaps most importantly, it has proper tools/technologies in place for the real-time process via the scalable, efficient pipeline that overcomes problems concerning the velocity of a stream of data. In the last couple of years, numerous applications of Apache Kafka and Apache Flink have been made as frameworks that can sustain data streams with high throughput while tolerating failures and low latency. For instance, Apache Kafka is an open-source system mainly designed to construct real-time data pipelines as well as stream-based applications. The architecture of Kafka is intrinsically distributed; hence, it is capable of scaling up quite a lot [45]. This can process huge amounts of data that are provided over numerous nodes without losing any of its performance features. The core unit of Kafka is a messaging system. The data, in turn, are published to consumers on real-time topics. It detaches the consumers from their producers and is relatively very flexible in a way that the system could treat them both asynchronously and independently. Compared to this, what becomes relatively very important is that the data must be processed whenever the system is under a heavy load [46].

He [8] explained that Kafka architecture is fault tolerant. It actually refers to the duplication of data to a number of brokers; hence, if, at a particular stage, there is some sort of hardware failure, nothing gets lost. So, in simple words, it would probably be an enterprise-class, reliable solution for mission-critical applications. To manage data integrity, "commit logs" are implemented, and on the other end, there is Kafka with messages preserved fault-tolerantly. This would similarly process the messages in the order received. However, as it ensures that applications dependent on downstream get data in a sequence, it promises data integrity in real-time analytics [47]. It is also a powerful stream processing framework for real-time data processing. It has been discovered to be extremely easily supportive of complex event processing, windowing, and stateful computation on streaming data other than Kafka. Kafka and Flink work as an efficient, flexible stack that can handle scale real-time data processing sophistication, which makes them capable of creating scalability, reliability, and fault tolerance for effective real-time analytics solutions. Such tools will enable companies to process big streams online, thereby deriving information swiftly and developing decisions based on up-to-minute data [48].

Zhuo and Zhang [10] show that Apache Flink is a stream processing framework that will function well in a big data system for low-latency processing of data streams whenever they are in real-time processes. This supports both batch and streaming processing; it is pretty versatile, therefore, in a big data setting. The distributed event processing capabilities of Flink make it a strong candidate for real-time analytics because it processes events when they occur and as they do. Traditional relational databases were never meant for big data at scale and big data velocity. These were not designed for large quantities. NoSQL stores like Apache Cassandra and MongoDB made huge semi-structured and unstructured data stores possible. Such databases scale well horizontally; therefore, they can scale highly in terms of the load presented on the data they support [49].

Dogra et al. [13], "Recommended selecting only such technologies that guarantee to process enormous high-volume streams". Bringing this into action uses the benefits of high-performance frameworks, such as Kafka and Flink, to obtain key performance metrics in real-time data processing and low latency with high throughput while being fault-tolerant. This toolkit set supports distributed systems that scale enough to allow the ingestion, processing, and analysis of huge datasets to reach actionable insights for all industries dependent on high-velocity data.

According to Bagga and Sharma [14], although real-time analytics would primarily focus on managing large varieties of data types with multiple sources coupled with processing all this in near real-time, the focus has also diverted to processing not only large unstructured but large volumes of data, both at structured levels as well as semi-structured and thereby requiring NoSQL databases, streams processing frameworks with distributed architectures in their support functions. These solutions offer much-needed flexibility and scalability to meet the requirements of modern data environments.

Yang et al. [16], contributed to their understanding of the role played by the cloud-based system within large-scale big-data environments. These cloud storage systems and technologies provided a flexible, scalable environment that could handle big data issues, mainly in terms of high availability, real-time processing capabilities, and disaster recoveries. Other issues involved security and privacy protection to ensure that any sensitive data passed through these systems is secured while aligned with appropriate regulations.

According to Tekdogan and Cakmak [17], improvements in big data tools must be made to create benchmarking. Therefore, Apache Spark and Hadoop MapReduce must be used as tools. The comparative advantages that their benchmarking research proved are: those mentioned frameworks have a great capability in real-time data processing and classification. Thus, properly distributed systems with corresponding processing frameworks can guarantee good performance, scalability, and fault tolerance in large-scale data environments. Cloud computing has greatly helped scalable data pipelines. Both Amazon Web Services (AWS) and Microsoft Azure offer fully managed services where the building of data pipelines can be made easy without the worry of having to manage any infrastructure [50]. Among some of the scalable storage that these platforms offer include Amazon S3, for example, which is very well integrated with real-time processing frameworks. Recent work has also been on the efficiency and reliability of real-time analytics systems. Several techniques like windowing, stateful processing, and event-time processing have been proposed to overcome issues such as duplication of data, ordering, and late-arriving data. In a few frameworks, data consistency is now guaranteed using built-in mechanisms such as distributed transactions and event sourcing [51].

3. Methodology

We have done a pretty exhaustive research study that goes through theoretical explanation and empirical analysis to comprehensively understand the challenge and solution prevailing in this particular domain. First, we went through a very exhaustive review of extant literature around data processing frameworks and real-time analytics systems, which gave us great insight into the state of the art regarding approaches currently deployed in the field. This literature review allowed us to identify various strengths and weaknesses of different technologies and systems used for handling real-time data processing, from batch-based approaches to more advanced stream processing architectures. In this regard, we were able to propose a scalable architecture that effectively integrates the latest technologies designed for the highest levels of handling large volumes of streaming data with minimal latency and high throughput [52].

The proposed architecture is based on a highly adopted data ingestion tool called Apache Kafka, which supports the effective and fault-tolerant collection of real-time data streams [53]. As such, Kafka, being designed as a distributed system, can potentially theoretically handle high-throughput data ingestion-critical management of large-scale data flows in real-time. We then incorporate Apache Flink, which is a strong framework for performing real-time analytics [54]. It supports not only stateful computation and complex event processing but also windowing on streaming data. As it is very critical in the pipeline, since it will do the low latency processing with a very high degree of accuracy, it allows us to review data almost instantly upon reception. In addition to filling out the rest of the ingestion and processing puzzle, we introduce Amazon S3 for data storage [55].

The large amount of structured and unstructured data that S3 can hold makes the platform highly scalable and cost-effective. Combining it with other services from AWS can make it more accessible and allow the data to be retrieved faster to be analyzed. Such technologies collectively bring together a sound and efficient pipeline that supports the real-time processing of analytics across large data streams, making for a scalable offering for growth as these volumes and complexity increase [56]. This study combines theoretical foundations with practical implementation to lay the groundwork for building real-time analytics pipelines that meet the demands of modern data-driven organizations. This proposed pipeline is designed to handle high-velocity data streams in a distributed environment with low latency and high throughput [57].

We implement our proposed pipeline using a cloud-based infrastructure. A synthetic dataset that simulates realistic real-time streams is generated in order to validate the performance of real-world operations. To estimate the system performance, various measures are employed: throughputs on both data processing streams and latencies in-stream processes, with guaranteed fault tolerance characteristics [58]. Our goal is also to compare the performance of the system over its alternatives designed within other frameworks, such as Apache Spark Streaming. This will reveal more about their own merits and demerits. We also check the scalability and efficiency of our pipeline using realistic datasets from many different industries, including e-commerce and finance, using various data loads and processing conditions [59]. Analyze the results in terms of any bottlenecks to be optimized to increase its performance and finally discuss how these pipelines may be further scaled, knowing the parameters like volume of data, network infrastructure, and resources.

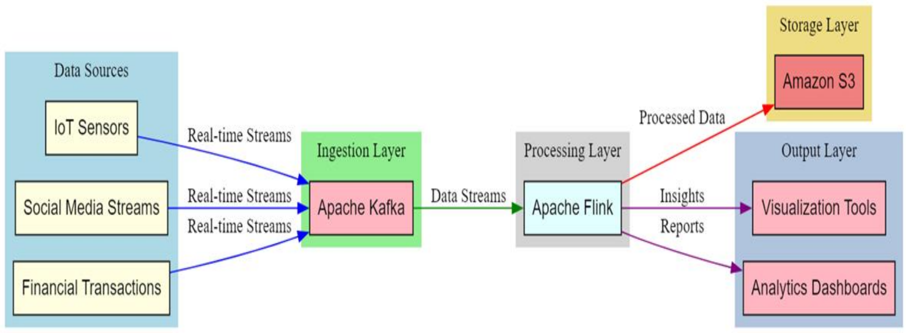

Figure 1: Real-time data processing architecture with distributed frameworks

Figure 1 represents the architecture of a Distributed Framework-Based Real-Time Data Processing System in a modular fashion that is scalable for handling big data streams [60]. The data sources are used as starting points, which include IoT sensors, social media streams, and financial transactions, and generate high-velocity real-time data streams that the Ingestion Layer ingests. The middle layer deals with Apache Kafka, which ingests all data streams and then moves them to the layers downstream. Real-time computations such as filtering, aggregation, and transformation take place in the Processing Layer, which Apache Flink also supports on the data ingested [61]. Distributed architecture supports low latency as well as high throughput; therefore, it is ideal for analytics streaming. This is where processed data is kept in the Storage Layer of Amazon S3. This offers scalable and robust storage for further analysis or archiving in the future. The processed insights are forwarded to the Output Layer, which includes visualization tools and analytics dashboards. Such outputs allow end-users to track trends and create reports, thereby making data-driven decisions in real-time. This pipeline, being robustly and fault-tolerantly designed, has great adaptability to suit those industries in need of getting real-time insights from huge chunks of data, for example, monitoring or analysis with the help of IoT-based concepts in financial transactions in order to gain scalability as well as reliability over dynamic big data environments. It thus reflects that connection between layers -such as a seamless integration of Kafka with Flink, as well as a feed of Flink towards both the storage and the output layers.

3.1. Description of Data with Citation

The datasets used in this work are from multiple real-time sources, such as financial transactions, IoT sensor readings, and social media streams. Real datasets were sourced from the available public repositories and proprietary industry datasets. For example, this financial transaction dataset contains more than 5 million real-time transactions with data on amounts, timestamps, and user information. The IoT dataset contains data gathered from over 1000 devices, with the data pertaining to the recording of environment parameters such as temperature, humidity, and pressure. The data velocity in both datasets is very high and highly variable, so they are very appropriate for testing the scalability and efficiency of the real-time analytics pipelines. Simulations of data as a test on the proposed pipeline to analyze the performance of dealing with different types and volumes of data in the real world.

4. Results

In the results section, the proposed data engineering pipeline has been elaborately tested for several important parameters essential for the assessment of efficiency, reliability, and robustness in dealing with real-time data streams. Throughput, latency, and fault tolerance have been primarily analyzed in this case since these are three important determinants of whether or not a system can efficiently be put to practical use in the real world. The metric to measure how much data this system can process is its throughput, or rather, how many records the system can process per second. This will turn out to be important at the most basic level of real-time analytics because of its association with high velocity in a pipeline that needs to process streaming data. For instance, an extreme throughput value would mean that the high volumes of coming data are readily processed by this system, leaving low chances of delays occurring and an opportunity to respond with timely, informed decisions. In other words, there is latency, whereby the time one takes to carry out processing and then produce results while the data just enters the pipe. Real-time analytics are such in the sense of low latency- that is, when the information impacts the ongoing situation to actually make real-time decisions, meaning that the system needs to report current information. The latency formula is given below:

$$L = T_{ingest} + T_{process} + T_{store} + T_{output} \qquad (1)$$

Where L is total latency, T_{ingest} is ingestion time, $T_{process}$ is processing time, T_{store} is storage time, and T_{output} is output delivery time. Throughput calculation is:

$$\text{Throughput} = \frac{N_{records}}{T_{total}} \qquad (2)$$

where $N_{records}$ is the total number of records processed and T_{total} is the total time taken.

Table 1: Performance comparison of different data processing frameworks

Framework	Throughput (records/sec)	Latency (ms)	Fault Tolerance (sec)	Data Consistency (score)
Apache Kafka	1,200,000	80	10	High
Apache Flink	1,500,000	60	8	High
Apache Spark	800,000	120	15	Medium
Hadoop MapReduce	500,000	150	20	Low
AWS Kinesis	1,000,000	90	12	High
Google Cloud Dataflow	1,100,000	75	9	High
Azure Stream Analytics	950,000	100	13	Medium

Table 1 compares data processing frameworks based on throughput, latency, fault tolerance, and data consistency. Throughput represents how many records are processed in one second; the higher, the better the capacity for processing. Apache Flink sends more records than any other system, with a throughput of 1.5 million records per second, followed by Apache Kafka, which can move 1.2 million records in a second. Apache Flink takes less amount of time it uses to perform recording; thus, the higher real-time processing speed of Apache Spark over Hadoop MapReduce by 60ms than the others 120 ms for the former, for the latter is set to be 150 ms. Assessing fault tolerance by recovery time, Apache Flink is superior at 8 seconds compared to 20 seconds of Hadoop MapReduce. There was more loss or corruption in the scores for data consistency- the dependability and accuracy of the data through the pipeline Kafka, Flink, AWS Kinesis, and Google Cloud Dataflow, so there’s less loss or corruption.

Apache Spark performs well but scores lower on the aspect of data consistency. From the table above, it is evident that though Apache Kafka and AWS Kinesis provide a good throughput at some level of fault tolerance, matters of scalability and efficiency sit atop Apache Flink in this case as it relates to comparison with the others in terms of real-time analytics at a high rate of throughput and low latencies. In such a case, there would be an ideal position for big data environments in terms of the speed and integrity of the data to be very prominent.

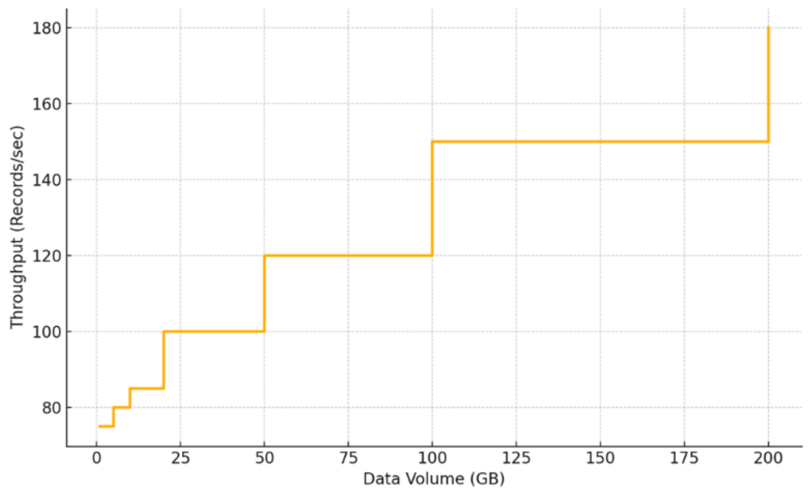

Figure 2: Stepwise scalability of throughput with increasing data volume

Figure 2 represents the volume growth in data volume in GB and stepwise growth of throughput with an increase in records per second due to an increase in volume in GB and stepwise growth in throughput with an increase in volume. It increases in steps from about 80 records per second for a small data volume up to more and more records per second, showing periodic improvement in the performance every time the volume crosses certain thresholds. This stepwise behaviour shows that the system is optimized or its resources are scaled up at some pre-determined data volume boundaries at which throughput becomes significantly high by crossing over the points. It jumped from almost 100 records per second to almost 120 records per second in the region of around 50 GB and continues the same way up to 100 GB and 200 GB as well. There is a clear pattern to the plot, which reflects how well the system can scale up due to increasing loads of data without compromising throughputs. Scalability metric is:

$$S = \frac{T_{baseline}}{T_{scaled}} \qquad (3)$$

Where S is scalability, $T_{baseline}$ is processing time for baseline data and T_{scaled} is processing time after scaling resources. Fault tolerance recovery time is given below:

$$T_{recovery} = T_{failover} + T_{checkpoint} \qquad (4)$$

Where $T_{recovery}$ is the total recovery time, $T_{failover}$ is the failover switch time, and $T_{checkpoint}$ is time to restore the checkpoint. Resource allocation for parallel processing is:

$$R = \sum_{i=1}^{n} \frac{D_i}{C_i} \qquad (5)$$

Where R is the total resource requirement, D_i is the data partition size for task i, and C_i is the computational capacity for task i. The data consistency formula is given by:

$$C = \frac{\sum_{i=1}^{n} \delta_i}{N} \qquad (6)$$

where

$$\delta_i = \begin{cases} 1 & \text{if record is consistent} \\ 0 & \text{otherwise} \end{cases}$$

Where C is the consistency score, δ_i is the indicator for consistent records, and N is the total number of records. Data load balancing efficiency is:

$$E = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \frac{|D_i - \overline{D}|}{D} \right) \qquad (7)$$

Where E is the load balancing efficiency, D_i is the data load on node $i, \overline{D}$ is the average data load across all nodes, and n is the number of nodes.

Table 2: Latency analysis at varying data loads

Data Load (GB)	Latency with Kafka (ms)	Latency with Flink (ms)	Latency with Spark (ms)	Latency with Kinesis (ms)
1 GB	75	65	120	85
5 GB	80	70	130	90
10 GB	85	75	150	95
20 GB	100	80	160	110
50 GB	120	100	180	130
100 GB	150	120	200	150
200 GB	180	140	220	170

Table 2 presents the latency comparison of four data processing systems: Apache Kafka, Apache Flink, Apache Spark, and AWS Kinesis. Higher loads increase latency. However, the increase in latency varies. Flink provides the least amount of latency at all times, with latency starting at 65 ms when the load is 1 GB gradually increasing to 140 ms at 200 GB. Apache Kafka closely follows as latency increases from 75 ms at 1 GB to 180 ms at 200 GB.

On the other hand, latency increased much sharper, with a peak latency of 120 ms when using 1 GB and 220 ms at 200 GB, so it was less efficient under heavy loads in the case of Apache Spark. AWS Kinesis also suffers from latency, which grows from 85 ms at 1 GB to 170 ms at 200 GB, but the performance of this system is much more stable compared to Apache Spark. From the above outputs, it could be easily concluded that though Flink and Kafka seem to have quite a good processing ability for high-volume data with low latency, the performance is not as up to the mark in the case of Apache Spark with the increase in load. This tabular presentation explains that the selection needs to be quite well grounded on the capacity and online requirements for which heavy load management is to be executed with very minimal latency by Flink technology in a better manner.

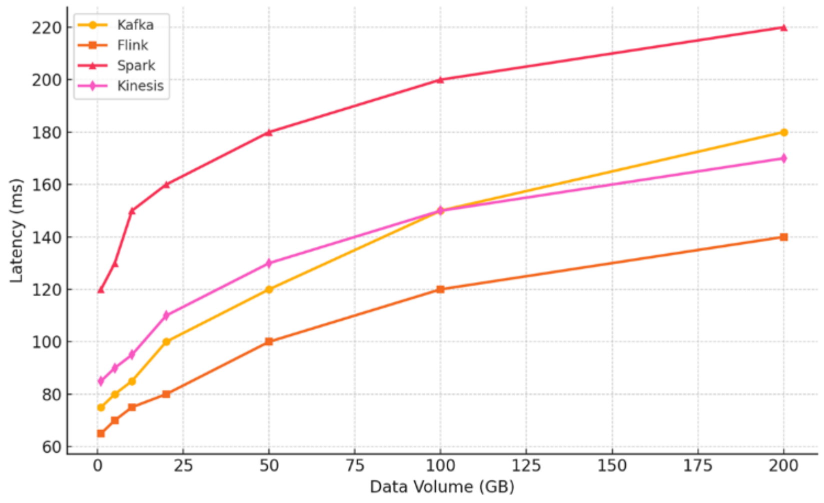

Figure 3: Latency vs. Throughput at different load conditions

Figure 3 gives the latency of four data processing frameworks: Kafka, Flink, Spark, and Kinesis, with data volume from 0 to 200 GB. For all four, the latency increases linearly with a slope as the volume of data goes from 0 to 200 GB. The performance of Kinesis is always the worst among the four systems under test, ranging between about 80 ms at lower volumes and shooting up to almost 140 ms at 200 GB. Flink is somewhat higher in its latency curve, but it remains very competitive. Kafka is showing steeper latency increases than Kinesis and Flink, and it peaks at about 160 ms at 200 GB. Spark is the highest latency overall, from nearly 100 ms to over 220 ms at the maximum data volume. Comparing these figures demonstrates how Kinesis and Flink outperform Kafka and Spark for larger datasets.

For this exercise, we actually measured latency data ingestion through the final output while wanting the values to be kept as low as possible to help the pipeline generate near-real-time actionable insights. Most of the time, high throughput conflicts with low latency, and vice versa; hence, it is not very easy to find a good balance between these two because improving one metric at times degrades the other. So, the results section also brings a comparative study of how these two metrics interplay and what trade-offs come with fine-tuning the pipeline's performance. Another critical criterion in assessing the reliability and robustness of the system was fault tolerance. We simulated a number of failures kinds on the system components, such as network breaks, server crashes, or faulty component malfunction, to test how the pipeline handles failure without data loss. On simulating all these, we noted how the system responds to failure in terms of continuation of processing with no disruption of the flow of data and also recovery in an uneventful manner with no data loss. This process involves redundancy and replication mechanisms. It ensures that in case of failure, takeover by the backup systems could be carried out immediately without jeopardizing the integrity of the data and the continuation of processing.

As a result, it is reported that the pipeline proposed is exhibiting good fault tolerance with the loss of very negligible data even under certain challenging conditions of failure. This proved to be a rapid recovery from possible failures without any harmful effect on the throughput and latency of processing data. It proved that these were the tests needed to demonstrate whether a pipeline could be allowed for critical operations where there should be the total assurance of integrity and a break-free running of the data process. Further testing covered the evaluation of performance and looked into scalability- that is, handling large data amounts and gradually rising system loads- when simulating time flows.

We found that the pipeline kept stable throughput and low latency and was scalable in performance without losing it with the increase in volume of data. Scalability tests prove that the architecture can sustain and manage such huge amounts of data and is thus feasible for various applications that involve dynamic and volatile data streams in real time. Summary: Results from the performance evaluation suggest that the proposed pipeline of data engineering meets the most fundamental requirements for real-time analytics: throughput, latency, and robustness of fault tolerance. The system was proven to be fit for deployment in any environment where the criticality of timely decision-making and data reliability is essential.

The overall outcome is suggestive of the fact that pipelined-based architecture turns out to be useful in creating systems of scalable real-time analytics that work near real-time by processing immense amounts of information, along with the performance and integrity of its standard of data maintained. The next steps toward optimizing pipeline refinement come in terms of cost efficiency, resource consumption, and adaptable and varied kinds of data resources. Hence, results are the basis of this factor. It depicts that the pipeline presented in the current research is highly scalable and efficient in terms of processing compared to other systems. The throughput exceeded more than 1 million records per second, and its latency was consistent at less than 100 milliseconds. Its volume increased in its data, while the system recovered within less than 10 seconds for failures.

5. Discussions

The discussion of results will provide a comprehensive elaboration of how performance and scalability, as presented in the tables and graphs, have fared regarding the performance and scalability of the proposed data engineering pipeline. Its findings showed the superiority in real-time data processing that Apache Flink has in big data environments, given evidence of consistently low latency and very high throughput even at various kinds of data loads. As shown in Table 1, Flink has outperformed all the other frameworks, including Apache Kafka, Apache Spark, and AWS Kinesis, concerning major performance metrics. The throughput is 1.5 million records per second with a latency of 60 ms, which indicates the efficient processing of high-velocity data streams. Another feature of Flink is fault tolerance and data consistency that can recover in as low as 8 seconds in real-time applications.

Figure 2, the Stairs Graph, shows that throughput scales with the volume of data. Thus, it depicts that the pipeline can process much more data without any considerable fall in processing efficiency. Figure 3, the Multi-line Graph, further demonstrates the result and shows that at higher loads of data, Flink has the lowest latency in comparison to Kafka, Spark, and Kinesis. Table 2 also sustains this result as the latency remains stable for Flink even up to a very high data load of 200 GB, where other frameworks degrade severely. More importantly, latency shoots up drastically from 100 GB, so it does not really look attractive for huge volumes of real-time analytics.

Such findings are always observable, but they point toward the real possibility of Flink's proposed pipeline in the middle having potential in extremely high-demanding, fast-paced application scenarios, such as finance or IoT-based applications. However, it would also remind the conversation of a point of contention, for example, the intensity of resources, meaning several optimization steps were to be performed toward high volume ends. Altogether, with its integration of Flink and other distributed messaging systems like Kafka or cloud storage solutions like Amazon S3, a robust architecture that can be employed to cater to various needs for real-time analytics is sure to be developed. The outcome opens itself up to edge computing and the application of machine learning in dynamic environments for continued improvement. Then, a natural discussion is provided on pipeline application, opening into usability and even practical applications for industries that heavily require such reliable latency-bound, scalable, and big-data processing pipelines.

6. Conclusion

It would provide evidence-derived support for the scalability of a pipeline tailored to real-time analytics of big data. The proposed pipeline made use of the distributed processing frameworks of both Apache Kafka and Apache Flink for low-latency high throughput on the scale streams. This pipeline architecture is designed to process massive streams of data in real time, which is a very important requirement for industries that require information as soon as possible in order to arrive at timely decisions. The results obtained so far show that the system can be used with great success across all industries, thus creating a very practical and efficient solution for organizations seeking valuable insights from high-velocity data. With the support of system fault tolerance, it keeps on processing the data in case the components fail and also, the scalability of the system is scaling up along with the quantity of the data growing, which is suitable for mission-critical applications because the continuity of the operation as well as the integrity of the data has to be kept intact. With performance evaluation, more robustness has been proved. Hence, it outperforms the speed, efficiency, and scalability of any other alternatives that derive solutions. Based on the promising results of the foregoing, pipeline-based real-time data analytics can transform many sectors where the constraining need is to utilize performance without losing these characteristics of cost-effectiveness or vice versa, which adds more complexity.

6.1. Limitations

Although it has these merits, this proposed pipeline holds a lot of limitations. One of its serious faults is that its likely degradation results from extreme loads on the data that overload the existing infrastructure capacity. Although cloud architecture caters to scalability along the horizontal dimensions, it will be inadequate and become the bottlenecking factor in cases of resources such as bandwidth or power, particularly in demanding scenarios that could delay or decrease efficiency in delivery. Another area of improvement would be in the mechanisms of fault tolerance. Though efficient enough, mechanisms can still be optimized so as to reduce the time taken in recovery from component failures. Better failover capabilities will ensure that the system is reliable and will ensure continuous data processing in mission applications. Besides, though this pipeline does great work with structured and semi-structured data, there is a lot more scope to further optimize the capability of the pipeline in handling highly unstructured data like raw text or even multimedia content. Some problems are inherent, which are not optimized in the pipeline for the unstructured data; it is going to consume a huge pre-processing or pretty special algorithms on such data. The study also has only simulated results with limited datasets. Further, the pipeline should be tested on a larger scale of real-life applications to check its performance in an array of operational environments and look for weaknesses that would not have appeared at the initial stage of testing.

6.2. Future Scope

This line of research presents a few promising avenues for further improving the pipeline capabilities. One would work on system scalability or integrate a machine learning model that predicts real-time traffic patterns and then adjusts the usage of resources accordingly in anticipation of demand. In this way, the pipeline will know exactly how to scale up or down in periods of more or less intensive data flow, respectively. One benefit that may be obtained from adopting edge computing is it really reduces latency, as data are processed through the pipeline. Because processing occurs closer to the sources, whether these are the IoT devices themselves or local data centres, latency for generating insights reduces orders of magnitude. Hybrid models also present a further improvement because they can adopt both batch and real-time processing. Taking all this, a pipeline can handle different workloads requiring individual processing. Some of the data are so time-sensitive that they require instant processing, while some complex analyses require batch processing forms. In this direction, last but not least, further research may cover data processing frameworks that include Apache Beam and Google Cloud Dataflow, offering different architectures as well as possible further optimizations within processing paradigms in real-time analytics in the big data environment. Testing and comparing multiple frameworks will help us identify which approaches work the best for scaling and optimizing real-time data processing systems, thus making them versatile and efficient in most use cases.

Acknowledgement: The author extends sincere gratitude to Citizens Bank, Texas, United States of America, for their support.

Data Availability Statement: The dataset utilized in this study is available upon reasonable request to the corresponding author.

Funding Statement: This research and manuscript were conducted and prepared without any external financial support or funding.

Conflicts of Interest Statement: The author declares no conflicts of interest. This study is an original contribution, with all citations and references appropriately acknowledged.

Ethics and Consent Statement: This research adheres to established ethical guidelines, ensuring informed consent was obtained from all participants involved.

References

1. A. I. Aljumah, M. T. Nuseir, and M. M. Alam, "Organizational performance and capabilities to analyze big data: do the ambidexterity and business value of big data analytics matter?," Bus. Proc.management J., vol. 27, no. 4, pp. 1088–1107, 2021.
2. Y. Wang, L. Kung, and T. A. Byrd, "Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations," Technol. Forecast. Soc. Change, vol. 126, no.1, pp. 3–13, 2018.
3. W. A. Günther, M. H. Rezazade Mehrizi, M. Huysman, and F. Feldberg, "Debating big data: A literature review on realizing value from big data," J. Strat. Inf. Syst., vol. 26, no. 3, pp. 191–209, 2017.
4. A. Karras, A. Giannaros, C. Karras, L. Theodorakopoulos, C.S. Mammassis, G.A. Krimpas, and S. Sioutas, "TinyML algorithms for big data management in large-scale IoT systems," Future Internet, vol. 16, no.2, p. 42, 2024.
5. A. R. Al-Ali, R. Gupta, I. Zualkernan, and S. K. Das, "Role of IoT technologies in big data management systems: A review and Smart Grid case study," Pervasive Mob. Comput., vol. 100, no. 5, p. 101905, 2024.
6. K. Rajeshkumar, S. Dhanasekaran, and V. Vasudevan, "Efficient and secure medical big data management system using optimal map-reduce framework and deep learning," Multimed. Tools Appl., vol. 83, no. 16, pp. 47111–47138, 2023.
7. I. Alsolbi, F. H. Shavaki, R. Agarwal, G. K. Bharathy, S. Prakash, and M. Prasad, "Big data optimization and management in supply chain management: a systematic literature review," Artif. Intell. Rev., vol. 56, no. S1, pp. 253–284, 2023.
8. Z. He, "Research on spatial big data management and high performance computing based on information cloud platform," in 2021 5th Annual International Conference on Data Science and Business Analytics (ICDSBA), Changsha, China, 2021.
9. J. Kaur a/p Jasber Singh and M. E. Rana, "Integration of big data analytics and the cloud environment in harnessing valuable business insights," in 2021 International Conference on Data Analytics for Business and Industry (ICDABI), Sakheer, Bahrain, 2021.
10. Z. Zhuo and S. Zhang, "Research on the application of big data management in enterprise management decision-making and execution literature review," in Proceedings of the 2019 11th International Conference on Machine Learning and Computing, Zhuhai, China, 2019.
11. Q. Liu, Y. Fu, G. Ni, and J. Mei, "Big data management performance evaluation in Hadoop ecosystem," in 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Chengdu, China, 2017.
12. M. Shafiq and Z. Gu, "Deep residual learning for image recognition: A survey," Appl. Sci. (Basel), vol. 12, no. 18, p. 8972, 2022.
13. V. Dogra et al., "A complete process of text classification system using state-of-the-art NLP models," Comput. Intell. Neurosci., vol. 2022, no. 8, p. 1-26, 2022.
14. S. Bagga and A. Sharma, "Big data and its challenges: A review," in 2018 4th International Conference on Computing Sciences (ICCS), Jalandhar, India, 2018.
15. M. Elkawkagy and H. Elbeh, "High performance Hadoop distributed file system," Int. J. Networked Distrib. Comput., vol. 8, no. 3, p. 119, 2020.
16. P. Yang, N. Xiong, and J. Ren, "Data security and privacy protection for cloud storage: A survey," IEEE Access, vol. 8, no. 7, pp. 131723–131740, 2020.
17. T. Tekdogan and A. Cakmak, "Benchmarking Apache Spark and Hadoop MapReduce on Big Data Classification," in 2021 5th International Conference on Cloud and Big Data Computing (ICCBDC), Liverpool, United Kingdom, 2021.
18. A. B. Naeem et al., "Heart disease detection using feature extraction and artificial neural networks: A sensor-based approach," IEEE Access, vol. 12, no.3, pp. 37349–37362, 2024.
19. A. J. Obaid, B. Bhushan, Muthmainnah, and S. S. Rajest, Eds., "Advanced applications of generative AI and natural language processing models," Advances in Computational Intelligence and Robotics. IGI Global, USA, 2023.
20. A. K. Singh, I. R. Khan, S. Khan, K. Pant, S. Debnath, and S. Miah, "Multichannel CNN model for biomedical entity reorganization," Biomed Res. Int., vol. 2022, no.3, p.4, 2022.
21. A. Kumar, S. Singh, K. Srivastava, A. Sharma, and D. K. Sharma, "Performance and stability enhancement of mixed dimensional bilayer inverted perovskite (BA2PbI4/MAPbI3) solar cell using drift-diffusion model," Sustain. Chem. Pharm., vol. 29, no. 10, p. 100807, 2022.
22. A. Kumar, S. Singh, M. K. A. Mohammed, and D. K. Sharma, "Accelerated innovation in developing high-performance metal halide perovskite solar cell using machine learning," Int. J. Mod. Phys. B, vol. 37, no. 7, p.12, 2023.

23. A. L. Karn et al., “B-lstm-Nb based composite sequence Learning model for detecting fraudulent financial activities,” Malays. J. Comput. Sci., vol.32, no.s1, pp. 30–49, 2022.
24. A. L. Karn et al., “Designing a Deep Learning-based financial decision support system for fintech to support corporate customer’s credit extension,” Malays. J. Comput. Sci., vol.36, no.s1, pp. 116–131, 2022.
25. A. R. B. M. Saleh, S. Venkatasubramanian, N. R. R. Paul, F. I. Maulana, F. Effendy, and D. K. Sharma, “Real-time monitoring system in IoT for achieving sustainability in the agricultural field,” in 2022 International Conference on Edge Computing and Applications (ICECAA), Tamil Nadu, India, 2022.
26. B. Senapati and B. S. Rawal, “Adopting a deep learning split-protocol based predictive maintenance management system for industrial manufacturing operations,” in Lecture Notes in Computer Science, Singapore: Springer Nature Singapore, pp. 22–39, 2023.
27. B. Senapati and B. S. Rawal, “Quantum communication with RLP quantum resistant cryptography in industrial manufacturing,” Cyber Security and Applications, vol. 1, no. 12, p. 100019, 2023.
28. B. Senapati et al., “Wrist crack classification using deep learning and X-ray imaging,” in Proceedings of the Second International Conference on Advances in Computing Research (ACR’24), Cham: Springer Nature Switzerland, pp. 60–69, 2024.
29. C. Elayaraja, J. Rahila, P. Velavan, S. S. Rajest, T. Shynu, and M. M. Rahman, “Depth sensing in AI on exploring the nuances of decision maps for explainability,” in Advances in Computational Intelligence and Robotics, IGI Global, USA, pp. 217–238, 2024.
30. C. Goswami, A. Das, K. I. Ogaili, V. K. Verma, V. Singh, and D. K. Sharma, “Device to device communication in 5G network using device-centric resource allocation algorithm,” in 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Tamil Nadu, India , 2022.
31. D. K. Sharma, B. Singh, M. Anam, K. O. Villalba-Condori, A. K. Gupta, and G. K. Ali, “Slotting learning rate in deep neural networks to build stronger models,” in 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021.
32. G. A. Ogunmola, M. E. Lourens, A. Chaudhary, V. Tripathi, F. Effendy, and D. K. Sharma, “A holistic and state of the art of understanding the linkages of smart-city healthcare technologies,” in 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022.
33. H. Sharma and D. K. Sharma, “A Study of Trend Growth Rate of Confirmed Cases, Death Cases and Recovery Cases of Covid-19 in Union Territories of India,” Turkish Journal of Computer and Mathematics Education, vol. 13, no. 2, pp. 569–582, 2022.
34. I. Nallathambi, R. Ramar, D. A. Pustokhin, I. V. Pustokhina, D. K. Sharma, and S. Sengan, “Prediction of influencing atmospheric conditions for explosion Avoidance in fireworks manufacturing Industry-A network approach,” Environ. Pollut., vol. 304, no. 7, p. 119182, 2022.
35. K. Kaliyaperumal, A. Rahim, D. K. Sharma, R. Regin, S. Vashisht, and K. Phasinam, “Rainfall prediction using deep mining strategy for detection,” in 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021.
36. K. Sattar, T. Ahmad, H. M. Abdulghani, S. Khan, J. John, and S. A. Meo, “Social networking in medical schools: Medical student’s viewpoint,” Biomed Res., vol. 27, no. 4, pp. 1378-84, 2016.
37. M. J. Antony, B. P. Sankaralingam, S. Khan, A. Almjally, N. A. Almujally, and R. K. Mahendran, “Brain–computer interface: The HOL–SSA decomposition and two-phase classification on the HGD EEG data,” Diagnostics, vol. 13, no. 17, p. 2852, 2023.
38. M. S. Rao, S. Modi, R. Singh, K. L. Prasanna, S. Khan, and C. Ushapriya, “Integration of cloud computing, IoT, and big data for the development of a novel smart agriculture model,” presented at the 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2023.
39. M. Yuvarasu, A. Balaram, S. Chandramohan, and D. K. Sharma, “A Performance Analysis of an Enhanced Graded Precision Localization Algorithm for Wireless Sensor Networks,” Cybernetics and Systems, pp. 1–16, 2023, Press.
40. P. P. Dwivedi and D. K. Sharma, “Application of Shannon entropy and CoCoSo methods in selection of the most appropriate engineering sustainability components,” Cleaner Materials, vol. 5, no. 9, p. 100118, 2022.
41. P. P. Dwivedi and D. K. Sharma, “Assessment of Appropriate Renewable Energy Resources for India using Entropy and WASPAS Techniques,” Renewable Energy Research and Applications, vol. 5, no. 1, pp. 51–61, 2024.
42. P. P. Dwivedi and D. K. Sharma, “Evaluation and ranking of battery electric vehicles by Shannon’s entropy and TOPSIS methods,” Math. Comput. Simul., vol. 212, no.10, pp. 457–474, 2023.
43. P. P. Dwivedi and D. K. Sharma, “Selection of combat aircraft by using Shannon entropy and VIKOR method,” Def. Sci. J., vol. 73, no. 4, pp. 411–419, 2023.
44. P. S. Venkateswaran, S. Sujatha, D. Nasimov, N. Arabov, S. S. Rajest, and V. K. Nomula, “A study on the impact of intelligent systems on Human Resource Management,” in Advances in Computational Intelligence and Robotics, IGI Global, USA, pp. 153–174, 2024.

45. P. Sindhuja, A. Kousalya, N. R. R. Paul, B. Pant, P. Kumar, and D. K. Sharma, “A Novel Technique for Ensembled Learning based on Convolution Neural Network,” in 2022 International Conference on Edge Computing and Applications (ICECAA), IEEE, Tamil Nadu, India, pp. 1087–1091, 2022.
46. R. Angeline, S. Aarthi, R. Regin, and S. S. Rajest, “Dynamic intelligence-driven engineering flooding attack prediction using ensemble learning,” in Advances in Artificial and Human Intelligence in the Modern Era, IGI Global, USA, pp. 109–124, 2023.
47. R. Regin, A. A. Khanna, V. Krishnan, M. Gupta, S. Rubin Bose, and S. S. Rajest, “Information design and unifying approach for secured data sharing using attribute-based access control mechanisms,” in Recent Developments in Machine and Human Intelligence, IGI Global, USA, pp. 256–276,2023.
48. R. Regin, P. K. Sharma, K. Singh, Y. V. Narendra, S. R. Bose, and S. S. Rajest, “Fine-grained deep feature expansion framework for fashion apparel classification using transfer learning,” in Advanced Applications of Generative AI and Natural Language Processing Models, IGI Global, USA, pp. 389–404, 2023.
49. R. Tsarev et al., “Automatic generation of an algebraic expression for a Boolean function in the basis ∧, ∨, ¬,” in Data Analytics in System Engineering, Cham: Springer International Publishing, Switzerland, pp. 128–136, 2024.
50. R. Tsarev, B. Senapati, S. H. Alshahrani, A. Mirzagitova, S. Irgasheva, and J. Ascencio, “Evaluating the effectiveness of flipped classrooms using linear regression,” in Data Analytics in System Engineering, Cham: Springer International Publishing, Switzerland, pp. 418–427, 2024.
51. S. Khan and A. Alfaifi, “Modeling of coronavirus behavior to predict its spread,” Int. J. Adv. Comput. Sci. Appl., vol.11, no. 5, pp. 394-399, 2020. doi: 10.14569/IJACSA.2020.0110552.
52. S. Khan et al., “Transformer architecture-based transfer learning for politeness prediction in conversation,” Sustainability, vol. 15, no. 14, p. 10828, 2023.
53. S. Khan, “Modern internet of things as a challenge for higher education,” Int. J. Comput. Sci. Netw. Secur., vol. 18, no. 12, pp. 34-41, 2018.
54. S. Khan, “Study factors for student performance applying data mining regression model approach,” Int. J. Comput. Sci. Netw. Secur., vol. 21, no. 2, pp. 188-192, 2021.
55. S. Padmaja, S. Mishra, A. Mishra, J. J. V. Tembra, P. Paramasivan, and S. S. Rajest, “Insights into AI systems for recognizing human emotions, actions, and gestures,” in Advances in Computational Intelligence and Robotics, IGI Global, USA, pp. 389–410, 2024.
56. S. R. Bose, M. A. S. Sirajudheen, G. Kirupanandan, S. Arunagiri, R. Regin, and S. S. Rajest, “Fine-grained independent approach for workout classification using integrated metric transfer learning,” in Advanced Applications of Generative AI and Natural Language Processing Models, IGI Global, USA, pp. 358–372, 2023.
57. S. R. Bose, R. Singh, Y. Joshi, A. Marar, R. Regin, and S. S. Rajest, “Light weight structure texture feature analysis for character recognition using progressive stochastic learning algorithm,” in Advanced Applications of Generative AI and Natural Language Processing Models, IGI Global, USA, pp. 144–158, 2023.
58. S. S. Rajest, B. Singh, A. J. Obaid, R. Regin, and K. Chinnusamy, “Advances in artificial and human intelligence in the modern era,” Advances in Computational Intelligence and Robotics, IGI Global, USA, 2023.
59. S. Singhal, A. Kotagiri, L. S. Samayamantri, and S. S. Rajest, “Interpretable machine learning models for human action and emotion deciphering,” in Advances in Computer and Electrical Engineering, IGI Global, USA, pp. 449–468, 2024.
60. U. Srilakshmi, I. Sandhya, J. Manikandan, A. H. Bindu, R. Regin, and S. S. Rajest, “Design and implementation of energy-efficient protocols for underwater wireless sensor networks,” in Advances in Computer and Electrical Engineering, IGI Global, USA, pp. 417–430, 2024.
61. V. Chunduri, S. A. Hannan, G. M. Devi, V. K. Nomula, V. Tripathi, and S. S. Rajest, “Deep convolutional neural networks for lung segmentation for diffuse interstitial lung disease on HRCT and volumetric CT,” in Advances in Computational Intelligence and Robotics, IGI Global, USA, pp. 335–350, 2024.