International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Developing Big Data in Computing Applications with Lambda Architecture

M. Mohammed Thaha[1], Aizul Nahar Harun[2], Rafikullah Deraman[3], T. Jackulin[4], Vignesh. T[5]

Submitted: 22/09/2023 **Revised**: 26/10/2023 **Accepted**: 10/11/2023

Abstract*:* The IT industry and academics have both paid close attention to big data. Information generation and collection in the digital and computing worlds happen at a rate that quickly surpasses the limit. Globally, there are currently over 5 billion mobile phone owners and over 2 billion Internet users. There are projected to be 50 billion Internet-connected gadgets by 2020. At this point, data generation is predicted to increase 44 times over that of 2009. This application is distinct from hardware-centric evaluations in that it employs real-world data-intake scenarios to demonstrate the methodology's effectiveness. This work adds to the corpus of prior research on LA while simultaneously addressing a significant gap in the field. This work establishes a standard for further research in this field by providing a fresh, empirically validated technique for assessing LA, a methodology that may be used in various big-data architectures. It also advances past work that lacked empirical validation. The field's future research orientations are defined by the prospects and various unresolved difficulties in Big Data dominance. It is simpler to research the area and develop the most effective techniques for handling Big Data thanks to these lines of investigation.

Keywords: *Big Data, Computing Applications, Lambda Architecture, empirical validation, dominance*

1. Introduction

The primary cause of this is that electronic data is useless, or unavailable. In addition, it has become challenging to correlate data that might show patterns that are helpful in the medical profession due to the healthcare systems that house health-related information [1]. An additional issue is BDA intricacy. Many operations and actions are carried out in a pipeline style in a BDA process. A growing variety of proprietary and open-source tools are available to carry out each of these duties. Due to the many different tasks that must be finished, such as data upload, data transformation/cleaning, statistical evaluation, interaction between back-end and front-end GUIs, and multiple analytics and visualization activities, there is a shortage of qualified BDA pipeline developers. Each device has a learning curve, and the challenge gets bigger when BDA engineers have to integrate multiple technologies into one pipeline. Additionally, because the BDA pipeline must operate continuously until the analytical need is met, basic procedures like ETL and machine learning must be automated.

[1]*Assistant Professor (Sr.Grade), B.S.Abdur Rahman Crescent Institute of Science and Technology, GST Road, Vandalur, Chennai - 600 048, Tamilnadu, INDIA.*
Email: mohammedthaha@crescent.education
[2]*YU-MJIIT International Joint Intellectual Property Lab (YU-MJIIT IJIPL), Department of Management of Technology, Malaysia-Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, Malaysia.*
[3]*Project and Facilities Management Research Group, Faculty of Civil Engineering and Built Environment, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Malaysia.*
Email: rafikullah@uthm.edu.my
[4]*Associate Professor, Panimalar Engineering College, Chennai, India.*
Email: karthijackulin@gmail.com, Orcid: 0000-0003-4015-7718
[5]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P. – 522302. Email: vigneshthangathurai@gmail.com, ORCID: 0000-0003-2865-966X*

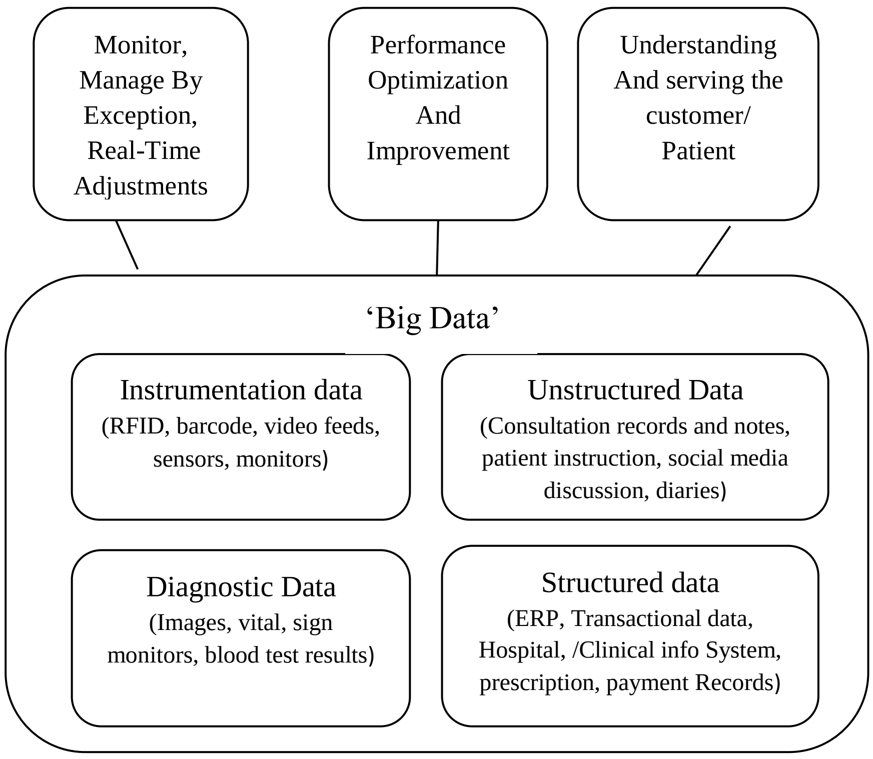
Monitor, Manage By Exception, Real-Time Adjustments
Performance Optimization And Improvement
Understanding And serving the customer/ Patient
'Big Data'
Instrumentation data (RFID, barcode, video feeds, sensors, monitors)
Unstructured Data (Consultation records and notes, patient instruction, social media discussion, diaries)
Diagnostic Data (Images, vital, sign monitors, blood test results)
Structured data (ERP, Transactional data, Hospital, /Clinical info System, prescription, payment Records)

Fig. 1.1. Examples of Big Data

The utilization of data from many easily accessible sensors and the absence of people from the decision-making procedure are two more problems associated with big data in Figure 1.1. BDA is a costly, resource-intensive, and complicated procedure with multiple problems that frequently outcome in large project failures across various sectors. In 2017, up to 85% of BDA efforts were failing, according to Gartner. Only 5% of the 273 telecommunication companies engaged in BDA are receiving more than 10% of the rewards, according to a McKinsey report that evaluated the impact of telecom companies' expenditures in BDA projects on the actual benefits. Furthermore, the implementation of BDA generated losses for 75–80% of the businesses [2]. As a result, development happens much quicker than acceptance. Since most of these technologies are open-source projects, utilizing them in an operational setting immediately calls for specific expertise and skills. The majority of firms are discouraged from making investments in BDA and adoption is slowed down during this learning time.

The creation of BDA pipelines over the last ten years has been greatly helped by the advancement and current dominance of the Python programming language as a pipelining language. A few technologies, such as MongoDB, Redis, Hbase, Spark as a tool Flink, and Hadoo, have also advanced and helped to pave the way for the emergence of the BDA applications in the telecom industry. These technologies are causing a rise in BDA applications in the telecommunications sector, which is expected to continue. As an example, BDA minimizes processing complexity and delay from data by recognizing traffic delay sensitivity and precisely identifying small packet traffic.

The following is the arrangement of the essay's succeeding sections. In Section 2, the research on the relevant earlier work is provided. Section 3 covers the aspects of the proposed system, including its suggested design, implementation model, components of the graph-based method, and data analysis. The effectiveness of the system is assessed and the implementation environment is described in Section 4. Section 5 presents the resolution.

2. Literature Review

Munshi, A. A., et.al [3] The primary concept behind Yarn is to divide the two main tasks of the MapReduce JobTracker/TaskTracker into distinct entities. For distributed application management, Yarn primarily comprises of a per-node slave NodeManager and a global ResourceManager. Yarn's ApplicationMaster collaborates with the NodeManager(s) to supervise and carry out component task execution, negotiating resources from the ResourceManager. It is the duty of every ApplicationMaster to negotiate suitable resource containers with the scheduler, keep track of their progress, and monitor their status.

Persico, V., et.al [4] The proposal goes into further depth on a data model, a synthetic data generator, and a description of the associated workload. BigBench sought to surpass earlier large data benchmarking initiatives. A few years ago, there was insufficient information on real-world use cases to adequately build generic solutions, so the scientific literature recommended a careful approach to creating "big data benchmarks" for the purpose of fairly and appropriately evaluating big data systems. Nonetheless, a number of big data benchmarking systems have been put forth in more recent times.

Katkar, J. et.al [5] An easier paradigm that can logically store and analyse vast amounts of data is suggested by lambda architecture. In addition to working with internal and external information sources, big data systems also handle unstructured, semi-organized, and raw data. Unlike traditional data warehousing systems, which are intended for structured internal data, analytical ecosystem architecture is a relatively new concept that was initially put forth by Nathan Marz. Business may also need big data frameworks with batch and real-time processing capabilities.

Roukh, A., et.al [6] Big data is being used in agriculture for a wide range of purposes, including sustainable farming, farm-to-fork traceability, yield prediction, supply chain management, risk mitigation, and loss reduction. Furthermore, large-scale data systems for agriculture can be classified into three primary categories: systems for sophisticated sensor technology; (ii) systems for risk management; and (iii) systems for agricultural control. Since our work falls under this category of systems, we primarily address the third category in this section. There have been recent efforts to establish farming platforms that gather the information required by intelligent farming decision support systems.

Basanta-Val, P. et.al [7] In order to satisfy various applications needs complex applications need to combine both online and offline technologies. Particularly in the industrial big-data space, most technologies are undergoing a consolidation process that will continue into the upcoming years. A current trend indicates a desire for a more efficient infrastructure: it seeks efficiency in map-reduce interactions. Industrial systems that must deliver faster responses to outside events even as system performance rises are particularly interested in the trend.

Kastouni, M. Z., et.al [8] while the idea of data analytics and its applications may seem novel to some, an examination of the literature revealed that the notion is not new at all, since data analytics can be summed up as the use of data to support business choices and activities. In fact, tales like the one about the Roman emperor Caesar, who rejected data from analysts predicting that March would be a "down month," or Michelangelo, who calculated the quantity of paint required to cover the Sistine Chapel using an intricate abacus, demonstrate the foundations of analytical thinking in forecasting results and assisting in business decisions.

Casado, R., et.al [9] Big Data is a broad term for large amounts of structured, semi-structured, and/or unstructured data that may be too large to handle and process using conventional databases and software. The high volume, high velocity, and variety of information found in big data can be used to enhance decision-making, forecasting, business analysis, customer experience and loyalty, and process optimisation across a variety of industries, businesses, and online social networks. As such, it calls for new management and processing techniques. The enormous volume, velocity, and variety (3Vs) of Big Data make traditional software unmanageable.

3. Methods and Materials

3.1 Current Data Processing Solutions

To design and develop decision support systems that optimize the underlying infrastructure, data analytics is a prerequisite. This entails searching for information online as well as digesting it. For specific occurrences, but additionally, historical information sources could be required to recognize data patterns that affects judgments. In addition, managing elasticity by dynamically assigning resources to meet growing demand presents computational challenges.

He proposed an affordable virtual machine provisioning solution that aimed to meet all SLA requirements while simultaneously executing dynamic data analytics workloads. The study emphasized how a better infrastructure would not be considered a service level agreement and would instead depend more on the provider performing trials [10].

The writers present a proposal for commercial warehousing or biological protein analysis. Image analysis in the medical domain is a further example of a combination. He discussed how the existing Twitter APIs were expanded to allow other academics to use their information analysis to enhance company processes on Twitter feeds. Nonetheless, novel approaches that enable multiple individuals with diverse backgrounds to develop and implement the best data processing programs are still needed. For online and bulk data processing, customized solutions are nonetheless needed, and these solutions must take into account non-functional factors like cost and complexity of the network.

Similar data processing tools have been used in subsequent research in applications related to smart grids where real-time redistribution of resources and forecasting are critical. Some of the shortcomings of the Hadoop system of processing have been addressed by the industry's current focus on using Spark SQL to facilitate even faster processing.

3.2 Lambda Architecture

The lambda architecture integrates batch and online processing into one framework and is offered as a software development pattern. When data must be legitimate for online use as soon as it arrives, the pattern is appropriate for applications where dashboards have temporal delays in gathering information and availability. To the needs of users, the pattern additionally allows batch analysis of older data sets to identify behavioral patterns.

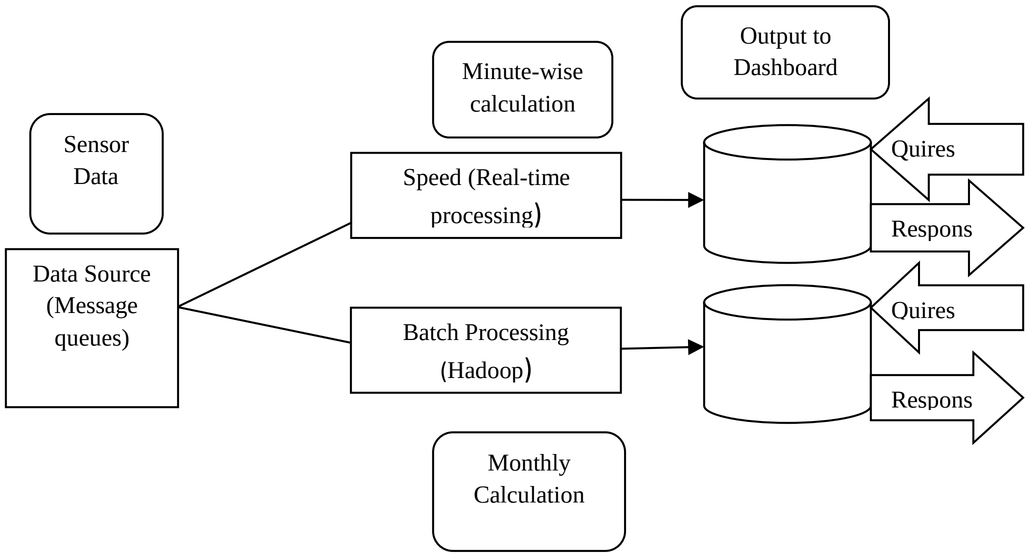

Fig. 3.1. Basic lambda architecture for speed and batch processing

The basic structure of the lambda architecture is seen in Figure 3.1 [11]. It meets a variety of needs. (1) Using batch processing to precompute big data sets (2) Real-time computing speed to reduce latency by performing calculations in real-time as data comes in; and (3) a query reply layer that interfaces with queries and returns the calculation results.

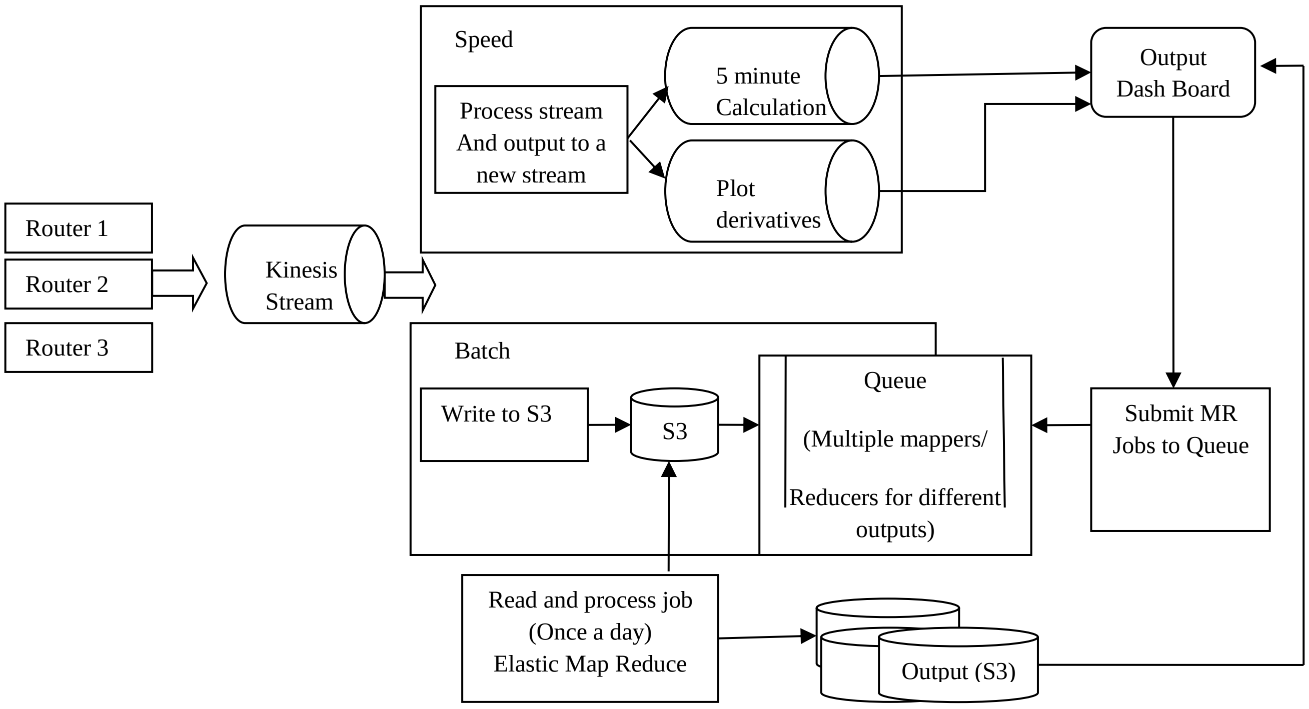
Speed
Process stream And output to a new stream
5 minute Calculation
Plot derivatives
Output Dash Board
Router 1
Router 2
Router 3
Kinesis Stream
Batch
Write to S3
S3
Queue (Multiple mappers/ Reducers for different outputs)
Submit MR Jobs to Queue
Read and process job (Once a day) Elastic Map Reduce
Output (S3)

Fig. 3.2. Main lambda architecture implemented on Amazon web services

By identifying which aspects of the data require batch or online processing, lambda architecture enables users to reduce their data processing expenses in Figure 3.2 [12]. Before further processing, the live stream may be used to identify data anomalies and verify its precision. Once data has been verified, it may be put into databases and analyzed over time utilizing batch scripts that execute once a day or once a month. Breaking the issue down into absorbed phases can assist users in reducing expenses associated with executing these scripts on bigger data sets.

They may also modify the data analysis algorithms to meet their specific needs. A great deal of data may be processed efficiently via this architecture for collecting and analyzing real-time sensor data.

3.3 Lambda Architecture-Based Big-Data Organization

While batch processing and analytics are done offline, real-time data processing & analytics are done on the speed layer. A conceptual depiction of the LA is displayed in Figure 3.3 [13].

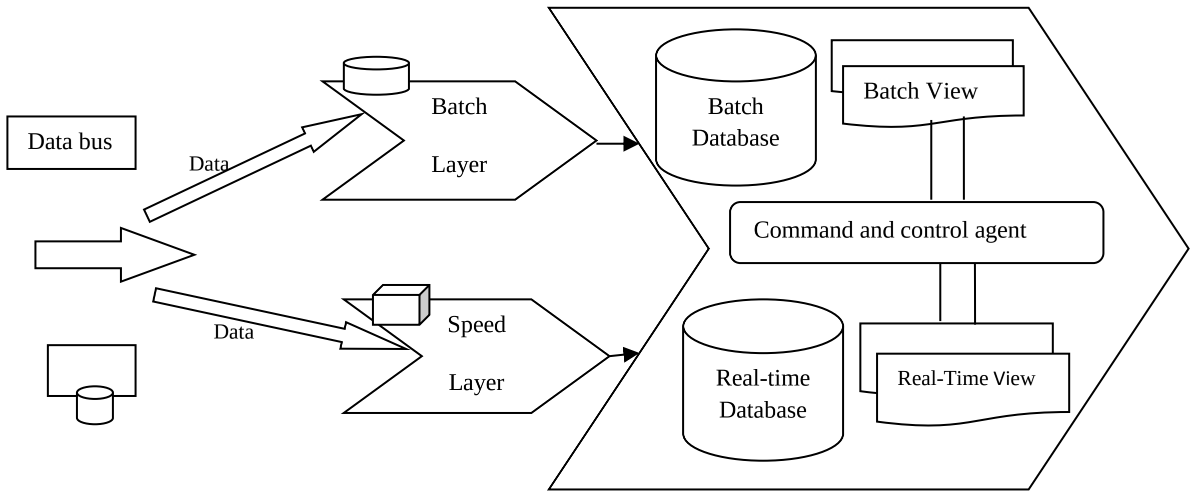

Fig. 3.3. Conceptual diagram of Lambda Architecture Common Technologies for Lambda Architecture Coatings

An effective substitute for conventional relational database management systems (RDBMS) is offered by the NoSQL storage of data. Organizations, however, can find it challenging to decide on the best course to take fast. Understanding the unique needs of the program that a relational

database system is unable to meet is crucial to choosing the best NoSQL database. It might not be required to use a NoSQL storage platform if an RDBMS can handle the data well. MongoDB, Redis, Apache HBase, and Apache Cassandra may all be used as speed layer databases. These databases have the capability of random writing and reading operations in addition to real-time data drinking. MongoDB, CouchbaseDB, and SploutSQL may all be utilized as batch-layer database records. Large volumes of data may be imported into these databases to generate and offer batch views.

3.4 Description of the Approach

The balance between complexity and latency is the main trade-off in lambda architecture. Batch processing is renowned for producing outcomes that are correct and trustworthy. It does, however, come at the price of more storage and maintenance needs as well as lengthier execution times. Although stream processing produces results fast and in real-time, the addition of partial or out-of-order data may cause it to be less accurate and reliable. To balance these trade-offs, the Lambda Architecture makes use of each level and merges them into the service layer. However, including these additional components in the data pipeline also makes it more complex and wasteful. Using each level and integrating it into the service layer, the Lambda Architecture achieves a compromise between these problems. The data pipeline is made more ineffective and convoluted by adding these additional components, though. Developers can streamline and get rid of redundancies in the data flow by using this method. This implies that the only set of information and logical structures that developers require to concentrate on creating, testing, and maintaining is that particular set. Without the need for separate implementations, group and stream processing may be changed with ease when using one platform. By doing this, development time decreases and consistent data processing throughout the architecture's levels is guaranteed. Moreover, since developers are simply able to monitor the movement of data and shifts inside a single codebase, having a unified framework makes debugging and troubleshooting simpler.

3.5 An Explanation of Lambda Architecture's Coordinated Operation of its Various Layers

Certain statistics, particularly those that depend on the rate of bulk operation and the length of the data gathering, must be expressly deleted by the batch operation. The "incomplete/lost statistics" stated before can be included in the streaming layer [14]. Equation (1) can be utilized for modeling this operation.

$$\lambda_{\mathrm{filtered}} = \mathrm{e}_{\mathrm{discard}}(\lambda_{\mathrm{raw}} \exists (\tau_{\mathrm{time}} - \tau_{\mathrm{interval}})) \tag{1}$$

It is shown by the formula statement in Equation (1) when the monitoring events should be omitted from the calculation. The method $\mathrm{e}_{\mathrm{discard}}$ () shows how events are eliminated in the provided equation. Before implementing the event selection (filter), the variable λ_{raw} indicates the full amount of raw events. τ_{time} Variable suggests the time spent executing the batch. In addition, τ_{interval} denotes the period that events are eliminated from the batch. Last but not least, the formula $\tau_{\text{time}} - \tau_{\text{interval}}$ denotes a time interval τ interval subtracted from the batch's execution time τ_{time}. The computation establishes the time required for event selection and emission of all events meeting the specified need. The computation establishes the time required for event selection and emission of all events meeting the specified requirement.

$$\lambda_{\text{batch}} = \lambda_{\text{filtered}}^{e_{\text{map(key,value)}} \to e_{\text{reduced(key,value)}}} \to\to \Delta_{\text{batch}}^{\text{data}}. \quad (2)$$

Equation (2) describes how key and value pairs are created by mapping the chosen events, represented as $\lambda_{\text{filtered}}$, via a mapping process, represented as $e_{\text{map}}()$. The value relates to each value of the matrices connected to the corresponding key, whilst the key serves as an individual identification for the statistical data. The data will next go through the reduction procedure, known as e_{reduce} (), which combines the values depending on the key for each distributed node. Following that, these uniform values are kept in a special storage folder referred to as $\Delta_{\text{batch}}^{\text{data}}$. The new file produced by the batch process will be stored on the distributed file system of Hadoop (HDER) in a specified folder. It is essential to keep in mind that there are choices available for replacing this storage layer.

If pre-computed statistics are available, they should be obtained from the serving layer and used in the unified bulk and stream layers. Equation (3) may be used to explain this operation:

$$\lambda_{\text{stats}}^{\text{storage}} = e_{\text{load}}^{\text{storage}}(\tau_{\text{current}}, \tau_{\text{from}}) = \{ \lambda_{\text{filtered}}^{\text{storage}} = \left(\tau_{\text{input}}^{\text{storage}} > (\tau_{\text{current}} - \tau_{\text{from}})\right). \quad (3)$$

The pre-computed goal statistics imported from the service layer are expressed by the variable $\lambda_{\tau_{\text{from}}}^{\text{stare}}$ in Equation (3). The current timestamp is indicated by the variable τ_{current}, and the date and time from which the data will be downloaded is represented by τ_{from}.

The process of loading information into the serving layer is done using the function $e_{\text{load}}^{\text{storage}}()$. When the variation between the current timestamp τ_{current} and the timestamp τ_{from} is greater than the input data, also referred to as $\tau_{\text{input}}^{\text{storage}}$, which includes every statistic transmitted from the serving layer to the database load method, and then the statistics $\lambda_{\text{filtered}}^{\text{storage}}$ should be selected and returned.

$$\lambda_{\text{processed}}^{\text{storage}} = \lambda_{\text{stats}}^{\text{storage}^{e_{\text{map(k,u)}}}} \to\to \prod_{\text{stats}}^{\text{storage}} \quad (4)$$

The value of $\lambda_{\text{processed}}^{\text{storage}}$, or the statistics that have been mapped and saved from the serving layer's $\tau_{\text{stats}}^{\text{storage}}$ into the memory, is expressed by equation (4).

Equation (5) describes the procedure in the streaming layer:

$$\lambda_{\text{processed}}^{\text{stream}} = e_{\text{transformation}}^{\text{data}}(\lambda_{\text{sream}})^{e_{\text{map(k,u)}} \to e_{\text{reduced(k,u)}}} (5)$$

The total number of monitored incidents within the stream is shown by the variable λ_{sream}. The events are filtered and converted by the function $e_{\text{transformation}}^{\text{data}}$ () before being mapped by the method $e_{\text{map}}()$, which produces key/value pairs. In the final stage, the data journey through the reduction process $f_{\text{reduced}}()$ to combine the values according to the key.

Equation (6) describes the batch information reading procedure in the batch layer:

$$\lambda_{\text{loaded}}^{\text{batch}} = e_{\text{batch}}^{\text{load}}(\lambda_{\text{batch}})^{e_{\text{map(k,u)}}} \to \quad (6)$$

Equation (6) utilizes the term $\lambda_{\text{loaded}}^{\text{batch}}$ to refer to the statistics that are mapped from storage, and λ_{batch} to refer to the precomputed statistics that come from Equations (1) and (2). The batch must be loaded utilizing the function $e_{\text{batch}}^{\text{load}}()$. The function loads just the set of statistics that are already precomputed and are regarded as "new." Upon the successful completion of the loading procedure, the file is designated as "old."

$$\lambda_{\text{joined}} = \left(\lambda_{\text{processed}}^{\text{storage}} \cup \lambda_{\text{loaded}}^{\text{batch}} \cup \lambda_{\text{processed}}^{\text{stream}} \right) \quad (7)$$

The statistics acquired from the service layer are shown as $\lambda_{\text{processed}}^{\text{storage}}$ in Equation (7).

The data that are estimated from streaming information is designated as $\lambda_{\text{processed}}^{\text{stream}}$, while the data that are obtained from batch calculations are designated as $\lambda_{\text{loaded}}^{\text{processed}}$. Once these information sets are merged, an additional set of data called λ_{joined} is generated.

Equation (8) describes how the statistical value in the memory has developed:

$$\lambda_{\text{state}}^{\text{memory}} = e_{\text{state}}^{\text{update}}\left(\lambda_{\text{joined}} \right) = \begin{Bmatrix} \text{insert}, & \text{if storage} = \text{True} \wedge \text{state} \\ \text{overwrite}, & \text{if batch} = \text{True} \\ \text{update} \vee \text{insert}, & \text{if storage}' \vee \text{batch}' \end{Bmatrix} \quad (8)$$

$$\left(\lambda_{\text{processed}}^{\text{stream}} \cup \lambda_{\text{loaded}}^{\text{batch}} \right) \infty \; \lambda_{\text{state}}^{\text{memory}} \forall e_{\text{serving-layer}}^{\text{update or insert}} \quad (9)$$

Equation (9) explains how the $\lambda_{\text{state}}^{\text{memory}}$ variable is paired with the $\lambda_{\text{processed}}^{\text{stream}}$ and $\lambda_{\text{loaded}}^{\text{batch}}$ variables, and then a left-join operation with the symbol ∞ is performed.

4. Implementation and Experimental Results

4.1 Environment Setup

We developed the deep ensemble-based IDS model in Python 3.7 using Tensorflow 2.6 in order to validate the efficacy of the suggested architecture. The experiment was performed using a Core i5 PC with 16GB RAM and a 64-bit operating system (OS). The software stack featured the Java programming language (JDK), Flash v3.0, Hadoop 2.7, Pyspark 3.0, and Kafka 2.6.

4.2 Evaluation Metrics

The proposed IDS, which employ deep learning models for attack detection, was assessed utilizing the most significant metrics for performance. The following list includes the most often-used assessment parameters by researchers:

1.Remember This assessment metric, which may be calculated using Eq. 10 [15], quantifies the percentage of actual positive outcomes that have been displayed to be positive.

$$\text{recall} = \frac{TP}{TP+EM} \qquad (10)$$

2.Precision Precision is used to characterize the performance model in several areas, such as information retrieval, data mining, and machine learning. We compute it concerning Eq. 12.

$$\text{precision} = \frac{\text{TP}}{\text{TP+EP}} \qquad (11)$$

3.Accuracy Accuracy may be stated as your overall performance of the categorization model. Eq. 3 can be used to calculate it.

$$\text{Accuracy} = \frac{\text{(TP+TM)}}{\text{TP+EP+TM+FM}} \qquad (12)$$

4.The F-measure, or F1-score, takes recall and precision into account. Equation 11 may be utilized for calculating the F1-score.

$$\text{E1} = 2 \times \frac{(\text{precision} \times \text{recall})}{\text{precision} + \text{recall}}$$

5.Throughput It talks of the quantity of results produced in a certain amount of time. In our situation, the measurement is in units of flow.

4.3 Performance of Binary Classifiers

We independently evaluated and deployed each of the three deep learning algorithms to examine how well they worked with the testing dataset to assess the binary classifier. To select the best binary classification framework with the most effective evaluation parameters, the accuracy and loss of the model during training and validation were calculated. Table 1 shows all performance indicators for the binary classification framework's ANN, CNN, and LSTM-based learner.

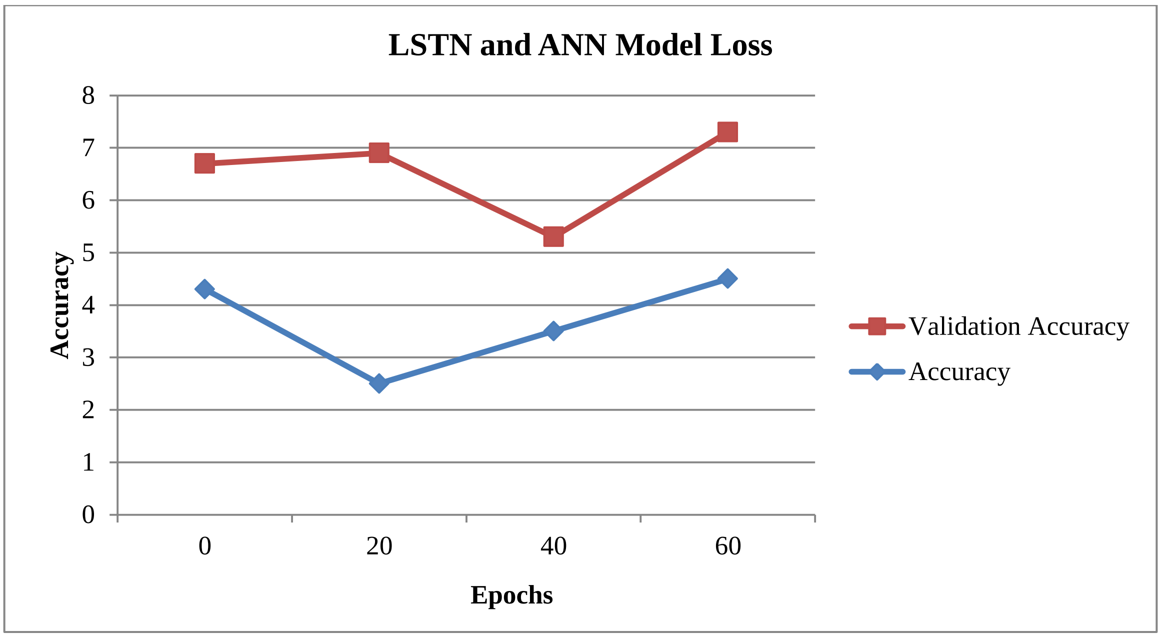

Fig. 4.1. Loss of LSTM and ANN during training and validation

4.4 Performance of Multi-class Classifiers

Figure 4.1 show the LSTM classifier's accuracy and loss during training and validation, respectively. Clear that all three models perform equally to one another throughout the multi-classification stage. In addition, LSTM's performance isn't as strong as its performance in the binary classifiers. This inspired us to use all three deep learning models in an ensemble-based multiclass classification stage. The performance data for the multi-class classifier with each of the three deep learning methods deployed separately can be seen in Table 2. It shows how well they can identify the various types of network assaults in an Internet of Things setting. Dividing the detection process into two phases—Binary and multi-class for attack-only transport—improves detection accuracy more than approaching the issue as one multiclass issue with four different attack types and regular traffic.

Table 1. Results for attack-only data—attack classification in batch mode using ensemble classifiers

Model	Type	Precision	Recall	F1-score	Accuracy	Time(ms)
Weighted ensemble	DDoS	93.6	99.2	96.4	99.7	0.8
	Okiru	99.2	93.3	96.2		

	Port Scan	1.1	2	2		
	C &C	2	2	2		

Table 2. Results for attack-only data—attack classification in batch mode

Model	Type	Precision	Recall	F1-score	Accuracy	Processing time (ms)
ANN	DDoS	89.6	99.7	94.4	97.1	0.08
	Okiru	99.7	88.3	93.6		
	Port Scan	100	2	100		
	C &C	2	2	2		
CNN	DDoS	89.7	99.6	94.5	97.2	0.06
	Okiru	99.8	88.9	93.7		
	Port Scan	100	2	100		
	C &C	2	2	2		
LSTM	DDoS	94.1	99.3	96.6	98.3	0.09
	Okiru	99.3	93.7	96.5		
	Port Scan	100	1.1	100		
	C &C	1.0	1.1	2		

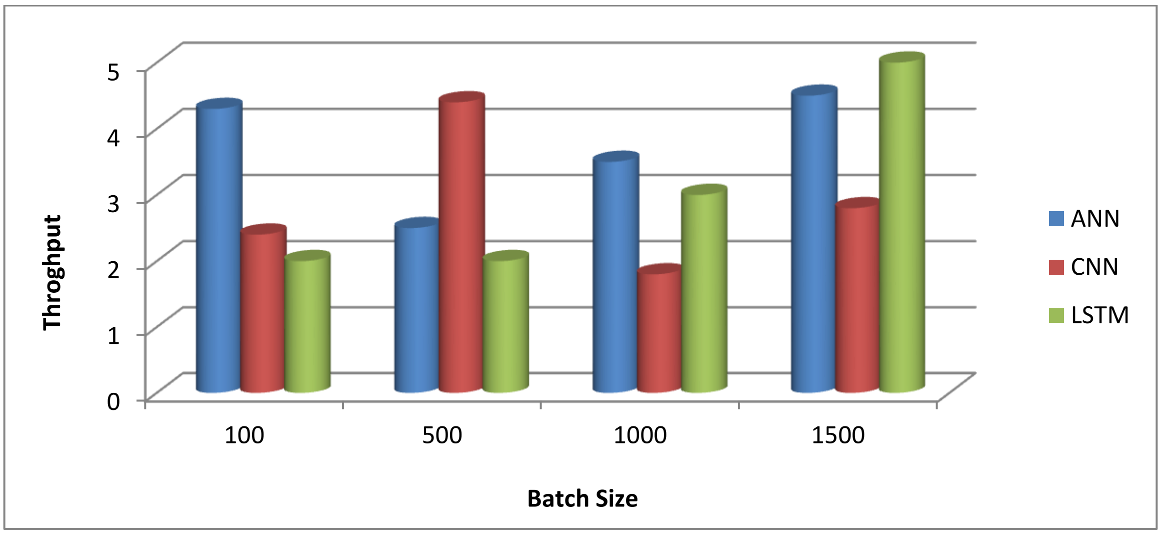

Fig. 4.2. Batch size-dependent binary and multi classification throughputs

Additionally, as Table 3 shows, binary classification in binary mode is significantly quicker than batch mode or without making use of Lambda architecture.

Table 3. Results for multi-classification without using lambda architecture—5 traffic types: 1 for benign and 4 for attacks

Model	Type	Precision	Recall	F1-score	Accuracy	Processing time (ms)
ANN	Normal	77.7	98.1	86.7	92.8	0.43
	DDoS	94.1	76.2	84.5		
	Okiru	98.9	90.4	94.5		
	Port Scan	99.5	99.0	99.3		
	C &C	99.6	1.0	99.9		
CNN	Normal	78.1	97.7	86.8	93.2	0.65
	DDoS	95.5	76.3	85.1		
	Okiru	98.7	91.9	95.1		
	Port Scan	98.9	99.7	99.4		
	C &C	100	1.1	1.1		
LSTM	Normal	78.8	96.35	86.7	92.8	0.13
	DDoS	91.2	79.1	84.7		
	Okiru	98.9	90.0	94.3		
	Port Scan	99.9	98.5	99.3		
	C &C	100	1.0	100		

The data has been streamed over a range of periods, spanning 1 to 15 s, as seen in Figure 4.2. As estimated, in all window frame durations for binary classification scenarios, LSTM processes the data quicker than other models.

5. Conclusion

There is no denying that big data analytics is extremely important and has a lot to offer the telecom sector. Through a comprehensive review of the literature, we discovered in this study that the limited use of the newest technologies in a developing technology stack and a lack of architecture restrict the practical applications of BDA to telecom in academic research. To address this and other challenges, we have created and published LambdaTel, innovative lambda architecture for BDA deployments in the telecom sector. Furthermore, we optimised the training of binary and multi-class classifiers at the batch layer by leveraging the Lambda architecture. At parallel, real-time Internet of Things traffic at the low-latency speed layer is evaluated and examined through the use of model inferences. Additionally, we show that, in comparison to the simple strategy, the ensemble approach yields higher detection precision and accuracy. Additionally, we show that the Lambda architecture improves throughput and overall performance.

In the future, to using additional deep-learning techniques in the ensemble model to enhance detection accuracy and system performance, we intend to test the suggested framework in an actual commercial IoT environment in order to further validate its performance. The hyperparameters may be tuned using automated machine learning approaches, which is a crucial objective.

References

[1] Zahid, H., Mahmood, T., Morshed, A., & Sellis, T. (2019). Big data analytics in telecommunications: literature review and architecture recommendations. *IEEE/CAA Journal of Automatica Sinica*, *7*(1), 18-38.

[2] Chen, C. M. (2016). Use cases and challenges in telecom big data analytics. *apSIpa transactions on Signal and Information processing*, *5*, e19.

[3] Munshi, A. A., & Mohamed, Y. A. R. I. (2018). Data lake lambda architecture for smart grids big data analytics. *IEEE Access*, *6*, 40463-40471.

[4] Persico, V., Pescapé, A., Picariello, A., & Sperlí, G. (2018). Benchmarking big data architectures for social networks data processing using public cloud platforms. *Future Generation Computer Systems*, *89*, 98-109.

[5] Katkar, J. (2015). STUDY OF BIG DATA ARHITECTURE LAMBDA ARHITECTURE.

[6] Roukh, A., Fote, F. N., Mahmoudi, S. A., & Mahmoudi, S. (2020). Big data processing architecture for smart farming. *Procedia Computer Science*, *177*, 78-85.

[7] Basanta-Val, P. (2017). An efficient industrial big-data engine. *IEEE Transactions on Industrial Informatics*, *14*(4), 1361-1369.

[8] Kastouni, M. Z., & Lahcen, A. A. (2022). Big data analytics in telecommunications: Governance, architecture and use cases. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), 2758-2770.

[9] Casado, R., & Younas, M. (2015). Emerging trends and technologies in big data processing. *Concurrency and Computation: Practice and Experience*, *27*(8), 2078-2091.

[10] Kiran, M., Murphy, P., Monga, I., Dugan, J., & Baveja, S. S. (2015, October). Lambda architecture for cost-effective batch and speed big data processing. In *2015 IEEE international conference on big data (big data)* (pp. 2785-2792). IEEE.

[11] Alghamdi, R., & Bellaiche, M. (2023). An ensemble deep learning based IDS for IoT using Lambda architecture. *Cybersecurity*, *6*(1), 5.

[12] Kim, Y. G., Ahmed, K. J., Lee, M. J., & Tsukamoto, K. (2022, August). A Comprehensive Analysis of Machine Learning-Based Intrusion Detection System for IoT-23 Dataset. In *International Conference on Intelligent Networking and Collaborative Systems* (pp. 475-486). Cham: Springer International Publishing.

[13] Imran, S., Mahmood, T., Morshed, A., & Sellis, T. (2020). Big data analytics in healthcare− A systematic literature review and roadmap for practical implementation. *IEEE/CAA Journal of Automatica Sinica*, *8*(1), 1-22.

[14] Ge, M., Bangui, H., & Buhnova, B. (2018). Big data for internet of things: a survey. *Future generation computer systems*, *87*, 601-614.

[15] Silva, B. N., Diyan, M., & Han, K. (2019). Big data analytics. *Deep learning: convergence to big data analytics*, (2), 13-30.