SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Enterprise System Integration For Distributed And Interoperable Architectures

J. Jayaudhaya[1], Shanthi Vairavan[2], Sanjay Kumar Jena[3], Antonibiya S[4]

[1] Department of Electronics and Communication Engineering, R.M.D. Engineering College, Chennai, India.
[2] Professor & Principal, Computer Science, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: shanthiv@maher.ac.in
[3] Assistant Professor, Department of Computer Science and Engineering, Institute of Technical Education and Research, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, India. Email: sanjayjena@soa.ac.in, ORCID: 0009-0000-7079-1651
[4] Assistant Professor, Department of Mathematics, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: antonibiya@maher.ac.in

Abstract
Modern enterprise environments require seamless integration of heterogeneous systems and real-time decision-making across distributed and interoperable architectures. However, traditional Enterprise Resource Planning (ERP) systems remain constrained by monolithic designs, limited interoperability, and inefficient processing of distributed CSV-based enterprise data, thereby restricting scalability and adaptive decision support. Research proposes a distributed enterprise system integration method explicitly designed for interoperable architectures, incorporating a novel Capuchin Search Graph Neural Network (CS-GNN). The proposed method adopts a cloud-based distributed architecture supported by service-oriented and Application Programming Interface (API)-driven interoperability to enable seamless communication among heterogeneous enterprise modules. Evaluation is conducted using a structured CSV dataset comprising 13000 data points, which includes enterprise transactional and operational records distributed across multiple subsystems. Data preprocessing includes Z-score normalization and missing value imputation to ensure consistency across distributed nodes. Feature extraction is performed using Independent Component Analysis (ICA) to derive statistically independent components from high-dimensional data. These features are transformed into graph-structured representations to capture interdependencies among enterprise entities. The CS-GNN model integrates graph neural learning with CSO to enhance decision-making across distributed components. Experimental results demonstrate a precision of 98.5% and a response time of 0.30s, which is implemented in Python and demonstrates improved interoperability, efficient cross-system integration, and enhanced decision support performance in distributed environments. The findings establish that integrating CS-GNN with ICA within distributed, interoperable architectures enables scalable, adaptive, and intelligent enterprise decision-making.

Keywords: Enterprise System Integration, Distributed Architectures, Interoperability, Cloud Computing, Decision Support Systems.

1. INTRODUCTION

Advances in technology have changed the business landscape and the healthcare industry because they provide tools to create very complex and connected digital solutions [1]. Using IoT and ERP together helped increase efficiency in many processes because they provided seamless communication between different devices; thus, making improvements in processes such as inventory control, human resource management, and accounting [2]. In addition, the creation of EHR has enabled better health information exchange between doctors and hospitals; therefore, helping achieve policy goals like HITECH and reducing chances of patient injuries or medical errors [3]. Modern business platforms tend to use microservices and cloud computing. This is why businesses need to develop advanced architecture and automated monitoring systems [4]. AI, ML, and DL play an increasingly important role in solving this technological problem. In the field of medicine, one of the key problems encountered in implementing solutions based on ML is related to the difficulties associated with project management and software integration into HEIS, which require certain strategies to be used when dealing with advanced technology and regulatory requirements [5]. Data architectures capable of handling growing amounts of data through ingestion, storage, processing, and model deployment are essential in the

context of scalable ML projects. AI observability techniques as well as AIOps allow predicting potential incidents and increasing platform resilience. On top of that, transformer-based large language models can provide automation and efficiency of decision-making processes and efficient usage of resources in the enterprise through the Enterprise AI approach [6]. The implementation of advanced AI and ML algorithms allows extracting insights from data and enhancing decision-making processes in telecommunication networks.

Although these improvements have been achieved, there are several issues that arise during integration and operation. Legacy ERP solutions and medical systems find it hard to integrate due to various problems such as interoperability of data, inefficiency in workflow, and delayed decision making [7]. Machine learning and AI technology could not perform well because of poor project management practices, complex integration, and handling of extensive amounts of data. Distributed enterprise architecture faces problems such as unreliability, delayed detection of problems, and inadequate prediction models, whereas telecommunication companies face difficulties in dealing with infrastructure for continuous service provision [8]. Updating enterprise architecture and ensuring interoperability in healthcare IT solutions are also among other important problems.

Research aim: Towards achieving a scalable enterprise integration strategy with enhanced interoperability, improved data processing efficiency, and decision making through the use of ICA feature extraction techniques and CS-GNN optimization.

Research organization: The current research paper consists of five main sections. The first section is related to integration problems of enterprises system and interoperability. In the second section, we are going to talk about the literature review on integration of enterprises, distributed computing, graph neural network, and optimization approaches. The third section talks about our suggested approach for feature extraction and CS-GNN. The forth section is dedicated to evaluation through experimental analysis.

2. RELATED WORKS

The current researches in terms of improving interoperability and integration between distributed ledger technology, enterprise systems, and healthcare systems are discussed in Table 1. These include layered architecture approach, API-driven integration, use of UML modelling, and real-time processing using cloud, AI, and edge computing, despite the existing limitations in scalability, security, and resource utilization.

Table 1: Recent research on enterprise system integration

Ref.	Objective	Techniques	Results	Limitations / Future Work
[9]	Enable Distributed Ledger Technologies (DLT) interoperability in Collaborative Enterprises.	Standardized DLT integration architecture; Design Science method	Supports communication among blockchains; improves operations	Limited cross-blockchain communication; enhance interoperability
[10]	Design an Enterprise Application Integration (EAI) model for managing applications and heterogeneous data.	Three-layer architecture; separation of business functions.	Unified data and applications; applicable to enterprise networks	Scalability, real-time implementation, extension to larger networks
[11]	Real-time data integration for Operational Business Intelligence	Federated systems, event-driven method, data mesh; cloud/AI/edge convergence	Improved responsiveness, decision-making, and operational efficiency	Scalability, compliance, algorithmic transparency, and data handling
[12]	Improve EHR interoperability	API-led integration method	Better data access, interoperability, and patient care	Privacy/security; extend to IoT, cloud
[13]	Enterprise data integration from legacy ERP	Entity-Relationship / Unified Modeling Language (ER/UML) models with integration patterns	Reusable models; rapid, maintainable integration	Complex data types; dynamic environments; automated pattern selection

[14]	Enable loosely-coupled communication among Information Technology (IT) systems.	Integration Flows diagrams; UML Profile with Apache Camel	Clear, extensible, precise communication flows	Refine UML profile; broader validation; align with dynamic requirements
[15]	DLT interoperability in Virtual Enterprises	Layered architecture with Internet of Things Application (IOTA) Tangle; Design Science; REST APIs	Cross-platform requests and data sharing; interoperable ecosystem	Resource-intensive; improve efficiency, scalability, and adoption
[16]	Enhance ERP systems with deep learning for adaptive decision-making and efficient data management.	Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Deep Reinforcement Learning (DRL)	RNN attaining 95% accuracy, CNN achieving 98% accuracy, and DRL delivering over 10% cost savings and reducing order processing time by 42.86%	Future work focuses on further improving deep learning models, enhancing adaptability, scalability, and decision-support capabilities in ERP environments.

Research gap: The existing methods of enterprise integration and interoperability face various constraints such as inadequate scalability, excessive consumption of resources, minimal cross-platform interaction, and absence of intelligent decision-making capabilities. Besides, current methods based on EAI and APIs do not provide effective mechanisms to model the complex interactions among data in the distributed enterprise system. In order to address the above problems, the suggested technique uses the combination of ICA and CS-GNN to extract features, model relationships, and perform intelligent decision-making.

3. METHODOLOGY

The methodology (from Figure 1) starts off with data collection in CSV format within a distributed enterprise, followed by the preprocessing step of normalization and missing data imputation to make sure that all data are consistent. The next step is performing feature extraction through ICA, followed by transforming the components into graph structures. In this regard, a combined approach involving the use of CS-GNN will be considered where both GNN and CSO techniques will be used.

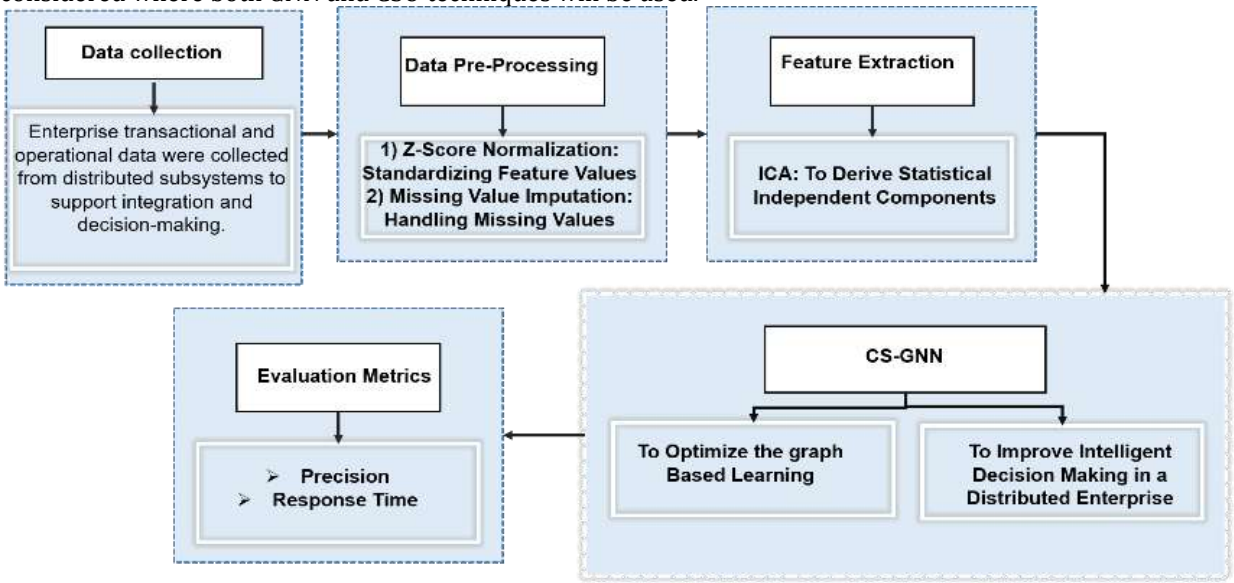

Figure 1: Methodology for the proposed method

3.1 Dataset description

The data used in this research was sourced from an open-source website (https://www.kaggle.com/datasets/colabsss/enterprise-integration-records) and consists of enterprise communication and operational subsystem records across cloud nodes, API services, and business modules for

enterprise system integration in distributed environments. This dataset comprises 13000 data points and includes communication flows, transaction events, resource utilization, interoperability quality, system dependencies, business unit activities, and decision-support status within distributed enterprise systems. The dataset is suitable for analyzing interoperability, cloud-based operations, subsystem coordination, API performance, and distributed decision-making patterns in enterprise environments, and it is divided into 70% for training and 30% for testing.

3.2 Preprocessing of data

Z-score standardization involves standardizing data through a process where features are scaled to a mean of 0 and standard deviation of 1, thus ensuring that all data is standardized and avoiding any domination of variables that operate on larger scales. In missing value imputation, missing data are estimated based on their statistical properties such as the mean, median, or mode.

3.2.1 Z-score normalization for feature standardization

The process of normalization using the Z-score technique takes place at the preprocessing stage in order to transform the system performance parameters into input values that would be used by the model to identify and rectify any faults in the resilient computing infrastructure. This helps to ensure that the system log information, resource utilization data, and transaction signals are normalized to the same scale.

$$X_{score} = \frac{rss(n)-\mu}{\sigma} \quad (1)$$

$$rss_{j,i} = [rss(1), rss(2), \dots rss(N)] \qquad for\ 1 \le n \le N \quad (2)$$

Equations (1) and (2) X_{score} define the Z-score for the nth observation, where $rss(n)$ is the residual sum of squares for that point. Here, μ represents the mean of all RSS values, and σ is their standard deviation. The vector notation $rss_{j,i} = [rss(1), rss(2), \dots rss(N)]$ lists all RSS values for indices j and i, where N is the total number of observations, allowing for the standardization of errors.

3.2.2 Missing value imputation for data completeness

The missing value imputation technique will be used in the suggested blockchain-based logistics supply chain approach for dealing with the incomplete information related to shipments, IoT, warehouse, and transactions by using some estimated value in place of missing values. The purpose of missing value imputation is to maintain consistency in data while also ensuring that there is no problem with data transparency and traceability in the logistic chain.

$$\bar{Y}^i = Y^i \odot N^i + (1 - N^i) \odot \tilde{Y}^i \quad (3)$$

In Equation (3), $\bar{Y}^i$ represents the final imputed value for the i^{th} observation. Y^i is the original observed data value, while $\tilde{Y}^i$ denotes the imputed value for missing data. N^i is a weighting indicator that defines whether the data is present (1) or missing (0) in the observed data. The operator $\odot$ indicates element-wise multiplication.

3.3 Feature extraction using ICA for dimensionality reduction

ICA is an algorithm that helps in extracting features from multivariate data by transforming the data into its independent components. In relation to the proposed architecture, ICA is utilized to reveal any hidden structures in various forms of data related to transactions, problems, and system performance. The core aim of the ICA algorithm is to minimize redundancy within the data and extract independent components.

$$X = AS \quad (4)$$

$$S = WX \quad (5)$$

Equation (4) represents the observed data matrix X , where X is the input data, A is the mixing process, and $S$are the original independent sources to be recovered. Equation (5) performs source separation using an unmixing matrix W, learned by ICA, to transform X into statistically independent components S, capturing meaningful features.

3.4 Optimization and learning process in CS-GNN

The CS algorithm is a nature-inspired meta-heuristic algorithm designed to solve complex optimization problems based on the behavior of capuchin monkeys. On the other hand, the GNN model is a deep learning algorithm whose aim is to learn from graph-based data to capture interdependencies among different entities. The main purpose of combining these two algorithms is to optimize the performance of the algorithm by learning parameters more efficiently and hence improving predictions and decision-making.

3.4.1 GNN for learning complex graphs

GNN is a DL approach that processes graph-based data consisting of entities in form of nodes and their interconnections represented as edges. GNN is basically used for learning complicated dependencies and connections among interconnected entities. The application areas of GNN include recommendation, node classification, link prediction, and decision-making in a distributed system environments. These characteristics of GNN enable high efficiency in use cases involving enterprise integration, social media, healthcare systems, and knowledge graphs.

$$F^{(j+1)} = \sigma(\tilde{B}_{sym} F^{(j)} V^{(j)}) \quad (6)$$

$$F^{(i+1)} = \sigma(B_{sym} F^{(j)} V^{(j)}) \quad (7)$$

$$X = \tilde{B}_{sym}\sigma(\tilde{B}_{sym} Y V^{(0)}) V^{(1)} \quad (8)$$

The above equations (6-8) denote the graph neural network operates on a node feature matrix $F^{(j)}$, where each row represents a graph node, and each column corresponds to a feature associated with that node. During learning, these features are iteratively updated to generate the feature matrix $F^{(j+1)}$, which is obtained by aggregating information from neighboring nodes through the graph structure and applying a nonlinear activation function $\sigma(\cdot)$. The graph connectivity is represented using the symmetrically normalized adjacency matrices $\tilde{B}_{sym}$ and B_{sym}. Feature transformation at each layer is controlled by trainable weight matrices $V^{(j)}$, with $V^{(0)}$ mapping the initial input feature matrix Y into a hidden representation space and $V^{(1)}$ further refining these hidden representations to produce more informative node embeddings. After multiple graph convolution and transformation operations, the network generates the final output feature matrix X. The indices j and i denote the iteration number.

3.4.2 CS in Optimization Processes

The CS algorithm is inspired by nature as it replicates the intelligence of capuchin monkeys in foraging and searching solutions. CS algorithms are mostly employed to improve convergence and avoid local optima in difficult optimization problems. In this paper, the CS algorithm is coupled with a GNN to optimize parameters of the machine learning model and improve its learning performance. The purpose of CS in this research is to increase decision-making precision and improve system efficiency through finding optimal solutions to the GNN algorithm.

$$v_j^i(t+1) = \rho v_j^i(t) + b_1(x_{best\,j}^i(t) - x_j^i(t))r_1 + b_2(Best_j - x_j^i(t))r_2 \quad (9)$$

$$x_j^i(t+1) = x_j^i(t) + v_j^i(t+1) \quad (10)$$

$$x_j^i(t+1) = Best_j + \frac{P_{bf} v_j^i(t+1)^2 sin\,(2\theta)}{2} \quad (11)$$

$$x_j^i(t+1) = \frac{x_j^i(t+1) + x_j^i(t)}{2} \quad (12)$$

From the above equations (9-12), $v_j^i(t)$ denotes the velocity of the i^{th} candidate solution in the j^{th} dimension at iteration t, while $x_j^i(t)$ represents its corresponding position. The term ρ is the inertia weight. The coefficients b_1 and b_2 are acceleration factors, $x_{best\,j}^i(t)$ is the best solution found by the i^{th} agent, and the global best position $Best_j$ is the best solution. The random variables r_1 and r_2 introduce stochastic behavior in the search process to enhance exploration. The updated velocity $v_j^i(t+1)$ is then used to update the position $x_j^i(t+1)$, where the second equation represents standard positional movement. In the third expression, P_{bf} is scaling factor, $v_j^i(t+1)$is again the updated velocity term, and θ is a random angle parameter, $sin\,(2\theta)$ is a nonlinear perturbation for better search diversity.

4. RESULTS AND DISCUSSION

Experimental evaluation of the proposed CS-GNN framework for the purpose of distributed enterprise integration is presented in this section. The comparison between the proposed CS-GNN framework and the traditional deep learning models, such as CNN, RNN, and DRL [16] is also conducted to analyze its advantages in terms of decision-making and interoperability. All experiments are executed using the Python programming language.

4.1 Transaction Performance and Decision Analysis

Feature magnitude totals of 120 sequential records are provided in Figure 2. The data consistency index falls within the range of 20 to 95; the data quality score varies from 30 to 90; failed requests lie between 0 and 100; the effect of entity node degree on cumulative peak values above 170 units is significant. Sequential records possess variable operational behavior, which is evident from the magnitude variations.

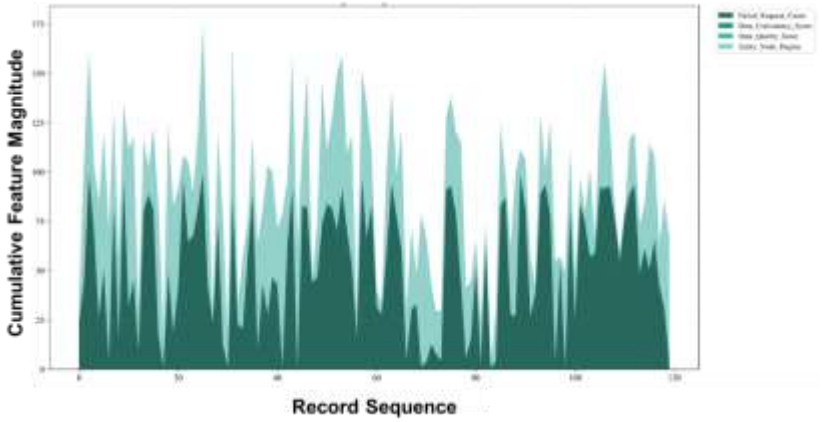

Figure 2: Cumulative feature magnitude analysis

The scores of decision support for different interoperability are illustrated in Figure 3(a). These scores range between 0.30 and 1.00, wherein the highest number of observations can be noted in low interoperability, an intermediate level of observations can be noted in moderate interoperability, while a small number of observations with comparatively higher scores can be noted in high interoperability. Service latency values between approximately 0 ms to 1,800 ms are depicted in Figure 3(b). Although a slightly higher density is noted between 1,200 ms to 1,600 ms, the overall distribution pattern remains relatively even.

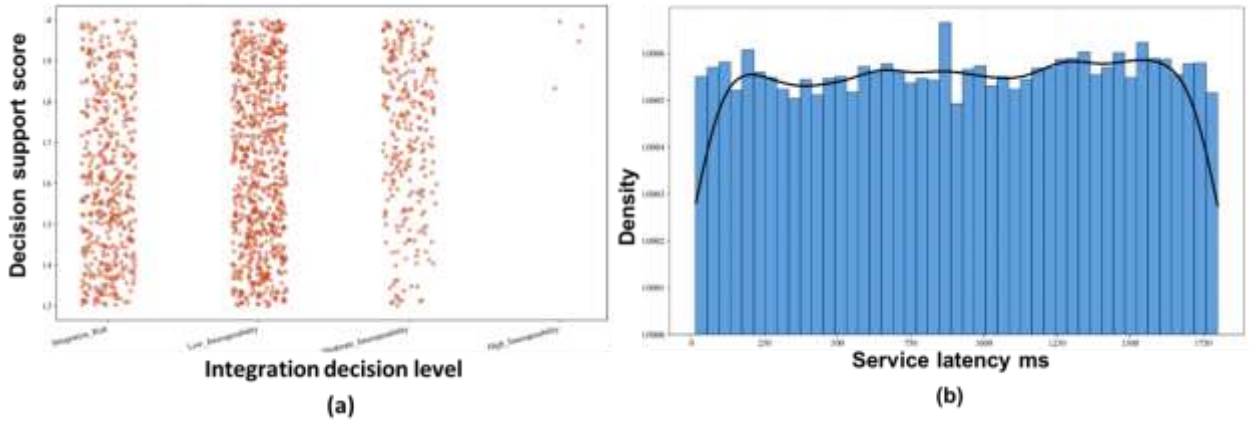

Figure 3: Analysis of (a) Integration decision support analysis; (b) Service latency distribution analysis

4.2 Metrics explanation

Precision: It refers to how accurate a system's decisions turn out to be for predicting that something is important. That is, the percentage of correct identification of all relevant outputs that have been classified as relevant.

Response Time: It is the total amount of time required by a computer to respond after an input and return an output. In distributed computing systems, the term response time implies effectiveness, and a low response time is a measure of quick response time.

4.3 Performance evaluation

Training dataset for the proposed model uses the Multi-Industry Enterprise Business Dataset [16] with a dataset size of 1.2 million records. This training data contains data from transactions, finances, customers, inventories, and macroeconomics. These datasets are used to train models for testing and ensuring that our model has scalability and robustness when deployed into the real business world. Table 2 and Figure 4 present the precision and response time of the proposed CS-GNN model compared with other models (CNN, RNN, and DRL) [16].

Table 2: Performance evaluation of the proposed model is trained using the existing dataset

Metrics	Precision (%)	Response time (s)
CNN [16]	95.5	0.8
RNN [16]	93.5	0.5
DRL [16]	96.2	1.0
CS-GNN [proposed]	97.2	0.4

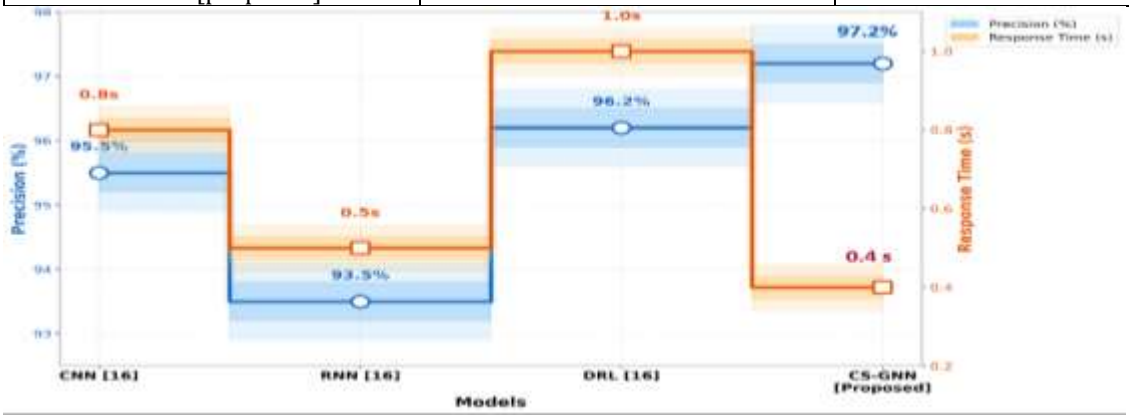

Figure 4: Graphical representation of the existing and proposed model trained in the existing dataset.

However, both the current dataset and the proposed dataset are learned using the CS-GNN model. The performance and efficiency of the CS-GNN model are demonstrated in Table 3 and Figure 5, where the CS-GNN model surpasses current models in terms of enterprise integration by providing higher prediction precision (98.5%) and lower response time (0.30 seconds). It implies that the CS-GNN offers better performance in terms of accuracy and decision efficiency concerning enterprise integration datasets. In summary, the performance achieved using Table 3 clearly reveals that the proposed model performs well.

Table 3: Comparison of existing and proposed models trained on enterprise integration records

Metrics	Precision (%)	Response time (s)
CNN	96.3	0.72
RNN	94.5	0.45
DRL	97.4	0.61
CS-GNN [proposed]	98.5	0.30

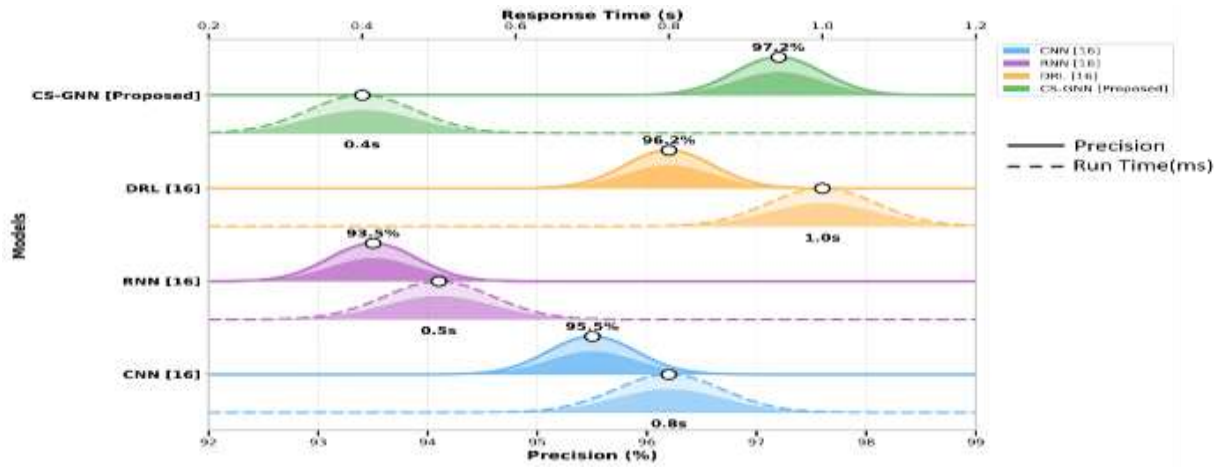

Figure 5: Representation of the existing and proposed models trained on enterprise integration records

DISCUSSION

Existing approaches to enterprise and distributed system integration face limitations such as poor interoperability, high resource consumption, scalability issues, and difficulty in handling heterogeneous and distributed data sources. On the other hand, the proposed technique is able to overcome the above challenges because of its cloud-based distributed approach involving CS-GNN and ICA. The technique will facilitate the effective extraction of features, decision-making, and interoperability among enterprise components.

5. CONCLUSION

It is anticipated that the enterprise system integration method relying on CS-GNN and ICA will bring improvements in connection, data processing, and smart decisions in different applications. Normalization, imputation, ICA-based feature extraction, and graph learning are employed for enhancing the capacity of representation of enterprise components. According to experiments, precision of the proposed framework amounts to 98.5%, while the response time equals 0.30 seconds using Python language. Nevertheless, some drawbacks include the use of structured databases, additional computing resources requirements, and difficulty adjusting to changing conditions. For future improvements, one can consider improving data processing, scalability, security, and optimization techniques.

REFERENCES

1. Uysal, M.P., 2022. Machine learning-enabled healthcare information systems in view of Industrial Information Integration Engineering. Journal of Industrial Information Integration, 30, p.100382. https://doi.org/10.1016/j.jii.2022.100382
2. Kumar, N., 2022. IoT-enabled real-time data integration in ERP systems. International Journal of Scientific Research in Science, Engineering and Technology, 9(6), pp.393-410. https://doi.org/10.32628/IJSRSET2215479
3. Vajpayee, A., Mohan, R., and Chilukoori, V.V.R., 2024. Building scalable data architectures for machine learning. International Journal of Computer Engineering and Technology (IJCET), 15(4), pp.308-320. https://doi.org/10.5281/zenodo.13234031
4. Pinheiro, C.R., Guerreiro, S.L.P.D. and Mamede, H.S., 2024. A lightweight ontology for enterprise architecture mining of API gateway logs. IEEE Access, 12, pp.128585-128601. https://doi.org/10.1109/ACCESS.2024.3456119
5. Jaiswal, I.A., 2024. AI-Powered Observability and Incident Prediction in Distributed Enterprise Platforms. Scientific Journal of Artificial Intelligence and Blockchain Technologies, 1(1), pp.1-14. https://doi.org/10.63345/sjaibt.v1.i1.201
6. Bhaskaran, S.V., 2025. EnterpriseAI: A transformer-based framework for cost optimization and process enhancement in enterprise systems. Computers, 14(3), p.106. https://doi.org/10.3390/computers14030106

7. Zeydan, E. and Mangues-Bafalluy, J., 2022. Recent advances in data engineering for networking. IEEE Access, 10, pp.34449-34496. https://doi.org/10.1109/ACCESS.2022.3162863
8. Adel, E., El-Sappagh, S., Barakat, S., Kwak, K.S., and Elmogy, M., 2022. Semantic architecture for interoperability in distributed healthcare systems. IEEE Access, 10, pp.126161-126179. https://doi.org/10.1109/ACCESS.2022.3223676
9. Anthony Jnr, B., 2024. Enhancing blockchain interoperability and intraoperability capabilities in a collaborative enterprise, from a standardized architecture perspective. Enterprise Information Systems, 18(3), p.2296647. https://doi.org/10.1080/17517575.2023.2296647
10. Yu, H.Y., Ogbeyemi, A., Lin, W.J., He, J., Sun, W. and Zhang, W.J., 2023. A semantic model for enterprise application integration in the era of data explosion and globalisation. Enterprise Information Systems, 17(4), p.1989495. https://doi.org/10.1080/17517575.2021.1989495
11. Palli, S.S., 2023. Real-time data integration architectures for operational business intelligence in global enterprises. International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), 9 (1), pp.361-371. https://doi.org/10.32628/CSEIT2391548
12. Mishra, R., Kaur, I., Sahu, S., Saxena, S., Malsa, N., and Narwaria, M., 2023. Establishing a three-layer architecture to improve interoperability in Medicare using smart and strategic API led integration. SoftwareX, 22, p.101376. https://doi.org/10.1016/j.softx.2023.101376
13. Petrasch, R.J. and Petrasch, R.R., 2022. Data integration and interoperability: Towards a model-driven and pattern-oriented approach. Modelling, 3(1), pp.105-126. https://doi.org/10.3390/modelling3010008
14. Górski, T., 2023. Integration flows modeling in the context of architectural views. IEEE Access, 11, pp.35220-35231. https://doi.org/10.1109/ACCESS.2023.3265210
15. Bokolo, A.J., 2022. Exploring interoperability of distributed Ledger and Decentralized Technology adoption in virtual enterprises. Information Systems and e-Business Management, 20(4), pp.685-718. https://doi.org/10.1007/s10257-022-00561-8
16. Zhang, L.S., 2024. Deep learning-based optimization of cloud enterprise resource planning (ERP) systems for adaptive decision support and management effectiveness analysis. IEEE Access, 12, pp.193402-193415. https://doi.org/10.1109/ACCESS.2024.3514879