Received: 15 April 2026 | Revised: 17 June 2026 | Accepted: 29 July 2026 | Published online: 25 August 2026

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2026, Vol. 00(00) 1–20
DOI: 10.47852/bonviewJDSIS620210014

BON VIEW PUBLISHING

Agentic Data Engineering Framework for Autonomous, Real-Time, and Trustworthy AI Systems

Parikshit Premchand Sahagal[1], Swetha Talluri[1], Pradeep Kumar Sambamurthy[1], Hastimal Jangid[2] and Abrar Ahmed Syed[3,*]

[1] *Independent Researcher, USA*

[2] *Coozmoo Digital Solutions, USA*

[3] *Data Analytics Platform, Gainwell Technologies LLC, USA*

Abstract: The fast expansion of real-time data has revealed the very important weaknesses of traditional data engineering pipelines, especially in the aspects of flexibility, scalability, and resilience. This paper is a proposal for an Agentic Data Engineering Framework that is meant to facilitate autonomous, real-time, and trust-conscious data processing. This framework integrates a multi-agent design that is composed of decision, processing, and trust components and allows pipelines to be dynamically reconfigured based on the condition of systems and characteristics of data. The methodology builds data engineering as a multi-objective optimization and involves latency, accuracy, computational cost, and trust score as a combination of learning. A novel algorithm, Agentic Pipeline Optimization with Trust Feedback, is developed to guide adaptive decision-making, based on reinforcement learning and trust-sensitive rewards modeling. Heterogeneous datasets under controlled conditions are used to perform experimental evaluation. The proposed framework has an accuracy of 94.6 %, latency of 165 ms, throughput of 1980 records/s, and a trust score of 0.91, which is much better than the baseline models. These improvements correspond to approximately 5–10% higher accuracy, 20–30% lower latency, and 20% increased throughput compared to existing approaches. The findings indicate that the framework works well in dynamic data settings as the performance is stable. The results suggest that agentic intelligence is most likely to combine with trust-sensitive optimization to offer a viable route to next-generation data engineering systems, specifically when it comes to Internet of Things (IoT), cloud, and intelligent enterprise applications.

Keywords: agentic AI, data engineering, real-time processing, trustworthy AI, autonomous systems

1. Introduction

1.1. Background and motivation

The proliferation of data in intelligent systems has begun to redefine the nature of computational pipeline design and implementation. Nowadays, in the context of smart materials, composite systems, and real-time monitoring of industry, data are not static and predictable anymore. Rather, it comes in a continuous stream, usually of heterogeneous origin, where instant interpretation and action are required. The old data engineering pipelines, which were mostly constructed on the basis of the Extract–Transform–Load (ETL) principles, were never meant to be used in such dynamic cases. They are fixed cycle and rule based and have no capability to adapt to the changing nature of the data itself. These pipelines have difficulty being interpretable and balanced with high-dimensional real-time streams of data, as noted by Peddisetti [1].

This constraint is even more imperative when the systems are supposed to aid in making decisions involving time sensitivity. As an example, in health monitoring of composite materials or adaptive manufacturing, the latency of data processing may cause a reduction in system performance or even system failure. Although there is progress in the computational infrastructure, most pipelines still have their fundamental logic that is hard to change and hence needs manual adjustments in order to be tuned and optimized. This puts a distance between actionable intelligence and available data.

1.2. Emerging trends in agentic and autonomous AI systems

Agentic artificial intelligence (AI) has emerged as a rapidly advancing paradigm in the field of artificial intelligence. Agentic systems are set up to perform with some autonomy, where

***Corresponding author:** Abrar Ahmed Syed, Data Analytics Platform, Gainwell Technologies LLC, USA. Email: abrar.syed@gainwelltechnologies.com

they can sense, reason, and act upon the input of environmental factors. These systems are based on decision-making capabilities, unlike traditional automation, which does not have decision-making capabilities in its architecture. As suggested by Viswanathan [2], agentic AI offers a systematic basis for creating systems capable of adapting and evolving without the need for human oversight.

Such methods have proven to be effective in different areas, as demonstrated by recent studies. An example is the work by Parab [3], which talks about how agentic AI can turn data analytics into autonomous processes of generating insights. Equally, transitioning to active decision-making agents in enterprise systems, which require active dashboards, is also emphasized by Saidkhujaev [4]. Agent-based frameworks have demonstrated high adaptability and learning ability in robotics and IoT-enabled environments, as in the case of Dazzi [5] and Kishor et al. [6]; intelligent agents allow autonomous navigation and human–robot collaboration.

Additionally, reinforcement learning has contributed to facilitating adaptive decision-making. The article demonstrates the potential for continuous improvement. Even at the infrastructure layer, agentic approaches are considered with regard to cloud resource management and performance engineering, as in Kumar [7] and Asani et al. [8].

1.3. Research gap and limitations of existing systems

As these developments still go on, agentic intelligence is still not experienced enough to be integrated into data engineering pipelines. Such a division brings about inefficiencies and lowers the system's responsiveness.

Several key limitations can be identified:

1) Fixed pipeline designs not responsive to variations in data distributions.
2) Little autonomy, manual configuration, and intervention are required.
3) Absence of trust awareness especially in security-sensitive applications.
4) Lack of real-time flexibility and hence slow decision-making.

According to Goyal [9], although agentic AI has been extensively researched, the real implementation of agentic AI into end-to-end data workflows remains a challenge. Moreover, problems concerning explainability, security, and data integrity are not given a lot of attention during pipeline design. This is especially worrying in applications where a distributed system or secure data exchange is required and trust is a paramount issue. In the article by Kishor et al. [10], the authors present the significance of information security in multi-party computing settings and emphasize the necessity to have embedded trust.

Also, hybrid intelligent systems (including fuzzy logic and deep learning) have been promising in real-time anomaly detection, as the article by Rangel-Martinez and Ricardez-Sandoval [11] demonstrated. Nevertheless, these are usually implemented at the model level and not incorporated into the larger data engineering system. On the same note, other higher-order cognitive architectures, such as those suggested by Kumi et al. [12], are also interested in decision-making but not the entire pipeline lifecycle.

1.4. Problem statement

Based on the above discussion, it is evident that the current data engineering systems do not support autonomous, adaptive, and reliable pipes that can work in real-time setups. This lack of built-in intelligence and trust-aware solutions reduces their functionality, especially in solutions where close monitoring of any device is needed and decisions must be made quickly.

1.5. Proposed solution: agentic data engineering framework

In response to these issues, this paper suggests an Agentic Data Engineering Framework that implements intelligence in the data pipeline. The architecture is created in the format of a multi-agent system, with every phase of the pipeline controlled by specific agents that are in charge of data ingestion, transformation, decision-making, and validation.

The proposed framework, in contrast to the traditional methods, enables a dynamic reconfiguration of the system in response to the patterns of incoming data and the needs of the system. It incorporates adaptive learning mechanisms, so the system can be enhanced without the involvement of humans. Moreover, it includes a specialized trust layer that analyzes data reliability, ensures secure processing, and promotes transparency in decision-making.

This will be compatible with the latest developments in agentic system design and enterprise architectures, as presented in Kumar [13] and Peddisetti [1]. It also enhances the abilities of the current structures through integration of autonomy, adaptability, and trust in one structure. Explainable and secure processing components are integrated to make sure that the system is dependable even under difficult and unpredictable conditions.

1.6. Contributions and novelty

The study contributes the following:

1) Creation of an autonomous agent-based data pipeline that can be autonomously operated.
2) Architecture of a real-time, dynamic, adaptive orchestration process of data.
3) Addition of a decision layer that is trust-aware to improve reliability and security.
4) Application of an experimentally validated model that shows better performance.

Moreover, the workpiece fills the gap between hypothetical AI developments in agentic AI and data engineering applications. An integration of the aspects of autonomous systems, secure computing, and reinforcement learning would provide the proposed framework with a solution for intelligent systems of the next generation. In this case, such integration may also be applicable depending on research such as that by Kishor et al. [14], where the applicability of adaptive AI in immersive environments is mentioned, and Kumar [13], where the role of scalability and resilience of enterprise systems is emphasized.

2. Literature Review

2.1. Traditional data engineering pipelines

Recent research notes that traditional pipelines are not scalable, flexible, and smartly coordinated. In particular, Ramesh et al. [15] address the necessity of optimizing traditional data lakes to be able to ensure real-time workload, whereas Adenuga et al. [16] point out the increasing demand for integrated data systems that would allow autonomous analytics to be supported. Likewise, Malek [17] suggests hybrid database systems that strive to

overcome semantic integration issues, but the systems still make use of fixed logic as opposed to intelligent adaptation.

Additionally, the uncritical processes in metadata management and pipeline optimization are still a major concern. Anuganti [18] reveals that metadata systems based on reinforcement learning can lead to better adaptability of the pipeline, yet these methods are not popular with traditional architectures. The issue of scalability and security in cloud-based settings also makes it more difficult to manage pipelines, according to Khrennikov [19]. These observations suggest that conventional pipelines are not suitable to meet the requirements of real-time, intelligent data processing, especially in more complicated systems of industrial and composite systems.

2.2. Real-time data processing systems

Real-time data processing frameworks, like streaming architectures, are receiving great interest to deal with the problem of latency and responsiveness. Similar technologies such as distributed stream processing and event-driven systems provide the opportunity to constantly absorb data and process it in almost real time. The systems are especially applicable in applications like IoT-enabled energy networks and intelligent governance, where decision-making is critical and requires timeliness. As an example, Mamodiya et al. [20] address the issue of how real-time data analytics can be used to optimize intelligent power systems, whereas Adenuga et al. [16] show the value of real-time integration in the process of logistics optimization.

Although such developments have been made, existing real-time solutions can be viewed as mostly passive processing engines with no in-built intelligence. They are oriented toward data throughput and latency minimization but are not able to make independent decisions or dynamically adjust the processing strategies. According to Ramesh et al. [15], even AI-optimized data lakes need external data control mechanisms to optimize queries, meaning that it does not have the internal capabilities to make decisions.

Moreover, streaming systems orchestration is also a challenge. Although alternative frameworks like distributed streaming orchestration have been proposed, they are yet to be developed. Introduced in Alten and Kämpf [21] is a distributed orchestration model of agentic reasoning, although it is not applied to large-scale data pipelines. Likewise, knowledge-enhanced systems suggested in the article by Malek [17] are more efficient in terms of queries but are not really autonomous. Consequently, the real-time systems, though effective, are still reliant on external control and devoid of the capability to adapt to situations autonomously.

2.3. Agentic AI and autonomous systems

The advent of agentic AI is a big step in the direction of autonomous and self-directed systems. Agentic AI frameworks are created to work autonomously through the use of contextual cognition, memory, and logic in addressing complex tasks. As per the article by Pati [22] and Abou Ali et al. [23], agentic systems are the future of AI, and these systems can be capable of combining perception, cognition, and action in the same architecture.

Recent studies have discussed numerous applications of agentic AI in various fields. As an example, Saleh et al. [24] suggest a memory-enhanced agentic framework for smart environments, and Martinez-Gil et al. [25] aim to implement edge AI solutions in agent-based architectures in Industry 5.0. On the same note, context engineering strategies that help the agents to handle complex decision-making processes are also discussed by Zhang et al. [26].

The development of hybrid intelligent systems has increased the functionality of agentic frameworks even more. In Sinha et al. [27], a quantum-neuromorphic architecture with explainable reinforcement learning is presented, which shows the capability of highly adaptable and intelligent systems. Simultaneously, Zhang et al. [26] consider the principle of edge general intelligence, in which agentic AI plays a crucial role in a decentralized setting.

Although there are these improvements, one of the limitations is still evident: agentic AI and data engineering pipelines have not been integrated. The majority of agentic systems are created to be autonomous decision-making units as opposed to being integrated into data processing processes. According to Chugani [28], the existing architectures concentrate on agent design and managing the context and fail to integrate the end-to-end data pipeline. This lack of connection restricts the usefulness of agentic AI to real-world data engineering applications.

2.4. Trustworthy AI systems

Trustworthy AI considers important factors such as explainability, fairness, robustness, and data integrity. In risky applications, like industrial systems and smart infrastructure, it is critical to make sure that the decisions made are reliable. High-stakes AI systems, as pointed out by Sunyaev et al. [29], are associated with risks, and accountability and governance should be prioritized.

Various frameworks have been suggested in order to promote the trust of AI systems. Indicatively, an explainable AI framework is described in Vice and Khan [30], which enhances transparency in the decision-making process. On the same note, Bugshan et al. [31] are interested in privacy-saving federated learning in industrial IoT settings where data security and confidentiality are the issues.

The discussion on the incorporation of trust systems in data systems has also been a recent area of study. According to Ekechi et al. [32], a trust-aware database intelligence framework is suggested, which includes explainability in database management decisions. Moreover, Wang et al. [33] present an AI auditing model that will help increase the trustworthiness of models by providing systematic review. The theory of trust, risk, and security management (TRiSM) has also been expanded to agentic systems as proposed by Raza et al. [34], indicating the necessity to have extensive trust systems in multi-agent systems.

One of the essential elements of reliable AI is cybersecurity. According to Jois et al. [35] behavior-aware security mechanisms are important in safeguarding intelligent systems, especially where there are distributed systems and data-intensive systems. In a similar fashion, AI-driven governance models, including the one suggested by Chakraborty [36], reveal how the intelligent system can promote data governance and compliance.

However, with progress, trust mechanisms are not often directly incorporated in data engineering pipelines. Such isolation hinders their performance, especially in real-time systems where continuous validation is essential. Moreover, safety assessment models like Sun et al. [37] emphasize the need to apply real-time reliability assessment, but their implementation remains domain-specific.

2.5. Summary of research gap

According to the literature, it is evident that these areas are mostly developing independently as major advances have been made in data engineering, real-time processing, agentic AI,

and trustworthy systems. Peculiar features of traditional pipelines include inflexibility, real-time systems are not intelligent, agentic systems are not integrated with data processes, and trust mechanisms are not implemented at the pipeline level. This piecemeal progress highlights the importance of having a cohesive framework, incorporating autonomy, adaptability, and trust in one data engineering architecture.

The summary of the current studies, as presented in Table 1, reveals that the recent research mainly deals with agentic intelligence, real-time processing, and trust mechanisms singly. Nevertheless, the combination of autonomy, real-time flexibility, and trust-conscious decision-making into an integrated framework of a full-fledged data engineering pipeline has not been studied in detail.

3. Methodology

This section introduces the implementation-based design of the suggested Agentic Data Engineering Framework, which was designed and tested in controlled experimental environments. The architecture is designed in such a way that it represents system-level deployment, as opposed to conceptual abstraction, and combines agentic intelligence, real-time data processing, and trust-aware validation into a single pipeline.

1) Problem formulation

Contemporary data pipelines have to be able to work on a continuous stream of data, with fluctuating loads and with tight decision deadlines. In this paper, the data engineering process is defined as a real-time, multi-objective optimization problem, and an agent dynamically chooses the pipeline configurations due to incoming data and system conditions.

Let D_t represent the incoming data stream at time t, defined as (1):

$$D_t = \{x_1, x_2, ..., x_n\} \tag{1}$$

where x_i denotes heterogeneous inputs such as sensor signals, logs, or structured records. The system operates within an action space (2):

$$A = \{a_1, a_2, ..., a_m\} \tag{2}$$

where each action corresponds to a possible pipeline configuration (e.g., transformation strategy, routing decision, or model selection). The system state s_t is based on data characteristics and runtime measurements like data velocity and processing load.

This formulation enables continuous adaptation, unlike static pipelines, and is compatible with autonomous data engineering paradigms [18, 28]. This method is aligned with adaptive control systems that are based on reinforcement learning [14].

Here, the optimization goal is formulated over four important dimensions:

1) Latency (L): Time required to process incoming data
2) Accuracy (Acc): Correctness of processed output
3) Trust Score (T): Reliability and security of the output
4) Computational Cost (C): Resource utilization

Table 1
Comparative literature analysis and identified research gaps in agentic data engineering systems

S. no.	Author(s)/year	Focus area	Methodology	Key findings	Gap	Relevance
1	Peddisetti [1]	Agentic data pipelines	Conceptual framework	Self-directed pipelines proposed	No real implementation	Base idea for agentic pipelines
2	Anuganti [18]	Autonomous data engineering	Reinforcement learning (RL)-based metadata control	Adaptive pipeline tuning	Limited to metadata layer	Supports adaptive learning in pipelines
3	Alten and Kämpf [21]	Streaming orchestration	Agent-based streaming control	Improved real-time coordination	No full pipeline integration	Relevant for real-time orchestration
4	Ramesh et al. [15]	AI data lakes	AI-based query optimization	Faster big data processing	No autonomy in decisions	Shows gap in intelligent pipelines
5	Abou Ali et al. [23]	Agentic AI survey	Survey of architectures	Multi-agent evolution identified	No pipeline integration	Theoretical support for agentic design
6	Ekechi et al. [32]	Trust-aware Database (DB) systems	Explainable AI in Database	Improved decision transparency	Limited to Database layer	Supports trust layer design
7	Bugshan et al. [31]	Trust in Industrial Internet of Things (IIoT)	Federated learning security	Privacy-preserving models	Not pipeline-level trust	Highlights security importance
8	Chugani [28]	Agentic architecture	Context-aware system design	Defined agent-based systems	No experimental validation	Useful for system architecture

The combined objective function is expressed as (3):

$$J = \alpha(1 - L) + \beta(Acc) + \gamma(T) - \delta(C) \tag{3}$$

where α, β, γ, and δ are weighting factors determined experimentally. This formula exemplifies the compromise of efficiency, reliability, and the use of resources, which can also be found in intelligent cloud systems and agentic systems [7, 8].

To direct the learning, a reward function is given by (4):

$$R_t = Acc - L - C + T \tag{4}$$

It stimulates the agent to prefer accurate, fast, low-cost, and reliable configurations. This is differentiated by the explicit optimization parameter of trust as part of traditional pipeline design and is consistent with the new trend of trust-aware AI systems [32, 33].

3.1. System overview

The framework provided is used as a layered, agent-based pipeline, which was to operate within the environment of a constant flow of data and the dynamism of a system. The structure of the architecture in Figure 1 will be a flow of data acquisition to validated output in a structured manner, with each of the layers having a clearly defined role and communicating with the other components.

The overall data movement can be expressed as (5):

$$D_t \rightarrow A_L \rightarrow P_L \rightarrow T_L \rightarrow O \tag{5}$$

where D_t is the data stream entering the system, A_L is the layer of the agent, P_L is the layer of the processing, T_L is the layer of the trust validation, and O is the output. This formulation builds on the problem definition in (1)–(4) by making the decision-making a part of the pipeline execution process.

1) Data source layer

The pipeline starts with the data source layer and constantly gathers heterogeneous data from distributed environments. These sources comprise structured logs, streaming sensor indications, and semi-structured logs. This layer is connected in the real world by streaming connectors and message queues so that it is able to provide low-latency availability of data. Real-time analytics systems have similar integration approaches that are used to deal with high-velocity data streams [16].

2) Agent layer

The agent layer is the heart of the decision-making of the framework. It measures the following state variables of a system—data rate, queue length, and processing delay—and chooses actions dynamically among the action space A that is predetermined by (2). All agents act with semi-autonomy but work toward a global optimization goal that is specified in (3). This form is an indication of the shift from a rule-based pipeline to an adaptive, policy-oriented framework, which is investigated in autonomous data engineering settings [16, 28].

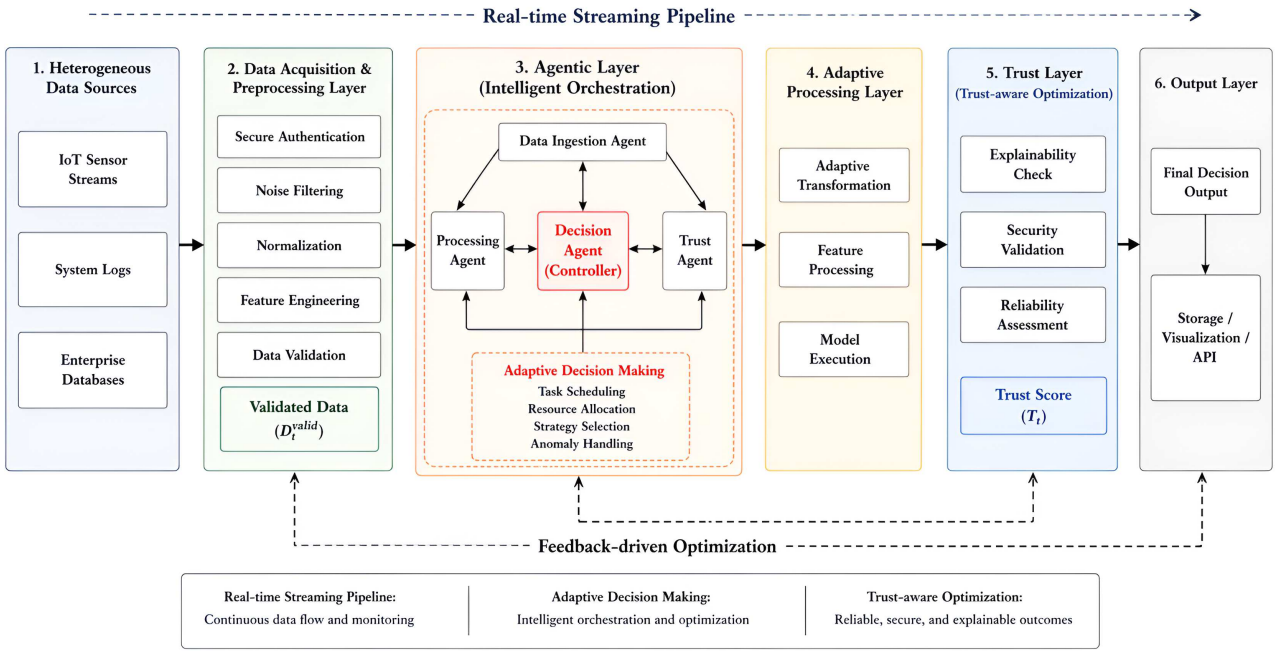
3) Processing layer

After picking an action, the information is directed to the processing layer where it is transformed, features are extracted, and the model is executed. This layer can be configured to perform both batch-like as well as streaming transformations. The processing logic is not hard-coded but is reconfigured according to decisions made by the agent; it is dynamically reconfigured. This flexibility coincides with AI-optimized data systems with an emphasis on real-time performance and scalability [15].

4) Trust layer

One of the salient characteristics of the proposed architecture is that it has a special trust layer, which is placed after

Figure 1
Overall architecture of the proposed agentic data engineering framework illustrating real-time data ingestion, agent-based orchestration, adaptive processing, and trust-aware output generation



data processing. This layer checks the credibility of the output that is generated by calculating a trust score T as in (3) and (4). The assessment includes explainability measurements, data consistency tests, and security validation processes. The framework can help to filter the unreliable or compromised outputs by incorporating trust assessment into the pipeline and only allowing them to go to the final stage. This method is aligned with the new trust-conscious AI systems and audit systems [32, 33].

5) Output layer

The last step is the generation of the validated output O that is stored, visualized, or sent to downstream applications. The proposed framework will be able to deliver only verified and trusted results in contrast to the traditional systems when the output is generated immediately after the processing of the data. The design is especially applicable in high-stakes situations where the reliability of the decision made is paramount.

On the whole, the architecture depicted in Figure 1 provides a closed-loop system whereby the data-driven decisions are continually optimized based on the feedback both of the system performance and trust evaluation. The combination of agent-based control, adaptive processing, and trust validation into one pipeline makes the framework go beyond the traditional approach of data engineering to self-regulating and resilient system design.

3.2. Data acquisition and preprocessing

The quality of the proposed framework is directly related to the quality and variety of input data. In this research, a hybrid dataset is developed that mimics a real-world deployment environment, a combination of streaming, semi-structured, and structured data sources. The requirements of the distributed intelligent systems are reflected in the dataset through different data velocities, formats, and frequency of updates, as summarized in Table 2. This mixed configuration enables the framework to be tested in realistic working conditions, just like the contemporary data-driven environment [16, 20].

The experiments were performed on two benchmark datasets that are public: the UCI Online Retail dataset (https://archive.ics.uci.edu/dataset/352/online+retail) and the HDFS Log dataset (https://github.com/logpai/loghub/tree/master/HDFS). These datasets were chosen to encompass different typical transactional and streaming log-processing scenarios found in the modern data engineering landscape.

3.2.1. Secure data source authentication

The flow of data received is authenticated very strictly before it is ingested into the pipeline to ensure integrity and prevent unauthorized access. It is a highly necessary process for distributed systems that possess a number of external sources that supply data. We can define the authentication mechanism as (7):

$$D_t^{auth} = f_{auth}(D_t, K, T) \tag{7}$$

Where:

1) D_t: incoming data stream
2) K: cryptographic key/token
3) T: timestamp validation constraint

The function f_{auth} performs:

1) Token verification: using hashed Application Programming Interface (API) keys
2) Temporal validation: rejecting stale or replayed data
3) Signature validation: ensuring data origin authenticity

The computational overhead of authentication is O(nlogn): where n is the number of incoming records. This provides scalability with the same level of security, which is in line with cryptographic AI systems [5, 14]. Data that pass authentication criteria are passed on to the next layers. This will eliminate the threat of malicious data injection and be compatible with behavior-aware cybersecurity models.

3.2.2. Data preprocessing pipeline

The data are then authenticated and subjected to a preprocessing phase that aims to normalize and clean up the data to be used by the downstream processing. Figure 2 shows the preprocessing workflow.

Preprocessing is a stream-conscious transformation pipeline, rather than a batch operation, which is carried out in a single step as illustrated in Figure 2. The transformation function is given as (8):

$$D_t^{prep} = f_{norm}(f_{clean}(f_{filter}(D_t^{auth}))) \tag{8}$$

where $f_{\text{filter}}(\cdot)$, $f_{\text{clean}}(\cdot)$, and $f_{\text{norm}}(\cdot)$ represent filtering, cleaning, and normalization functions, respectively. This multi-layered development makes sure that all of the transformations add value to the data quality one step at a time and maintain the temporal relationships.

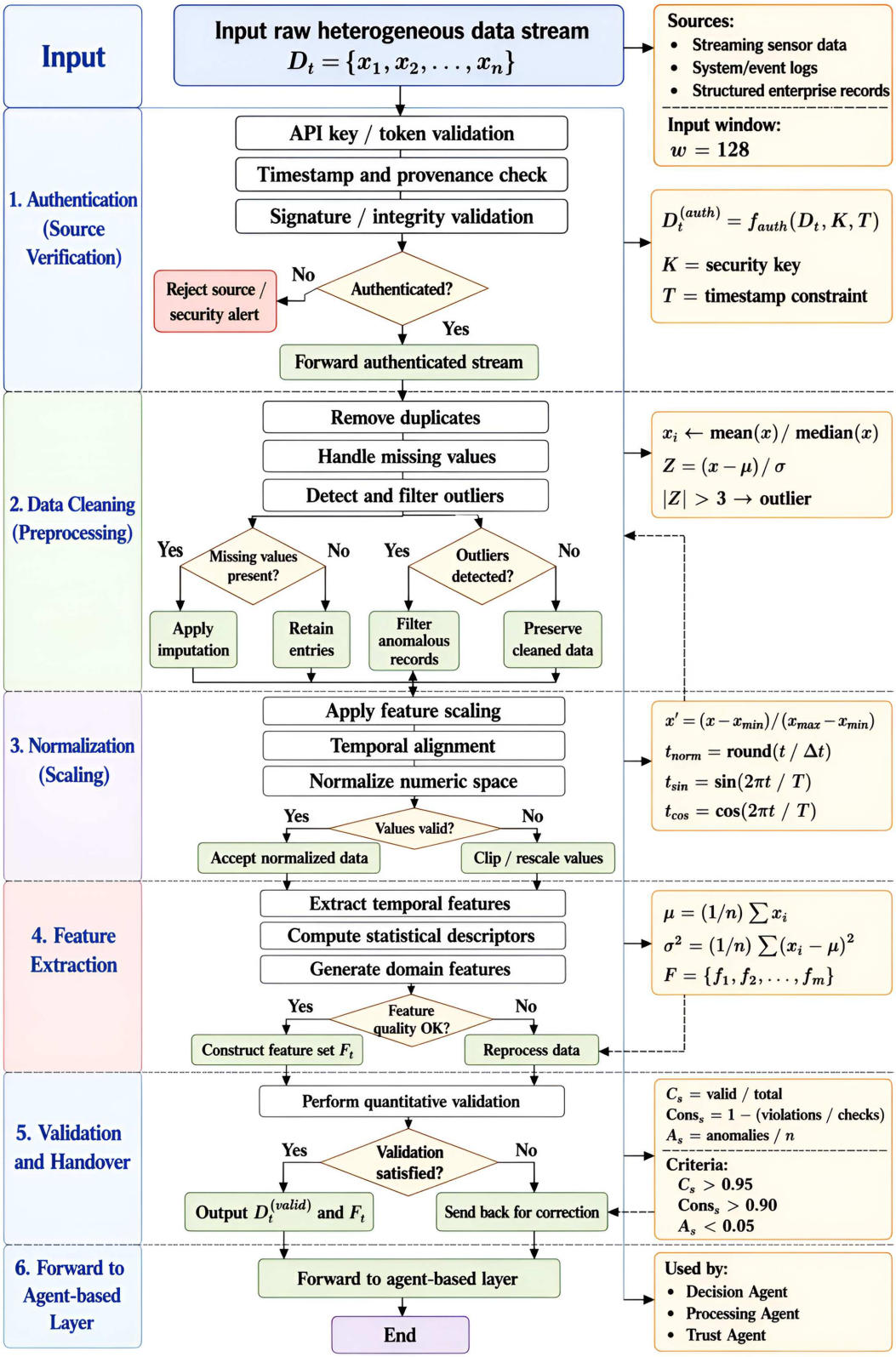
The system has a sliding window mechanism with a fixed window size of 128 samples to aid real-time implementation. This permits the local temporal analysis without incurring much latency. Noise filtering is done in each window, based on a statistical Z-score (9):

$$Z = \frac{x - \mu}{\sigma} \tag{9}$$

Table 2
Dataset characteristics and data source description used for experimental validation

Dataset name	Source	Type	Size	Features	Sampling rate	Use case
IoT Sensor Stream (Synthetic)	Generated using stochastic time-series simulator	Streaming	1.2M records	12	1 sec	Real-time monitoring
HDFS Log Dataset (Public)	Hadoop Log Repository	Semi-structured	500K entries	Variable	Event-driven	System behavior analysis
UCI Online Retail Dataset	UCI ML Repository	Structured	541K records	8	Batch/hourly	Transaction analysis

Figure 2
Data preprocessing workflow illustrating authentication, cleaning, normalization, and feature extraction stages before agent-based processing



where x is the value of the observed value, 0 is the mean, and S is the standard deviation of the windowed data. Observations with $|Z| > 3$ are considered outliers and are dropped, such that it is resistant to temporary spike anomalies.

After filtering, the data are normalized to a limited range with min–max scaling (10):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{10}$$

This transformation results in the reduction of the variance in features and in the stability of the part of downstream learning. At the same time, lightweight feature engineering is applied to extract such temporal features as lag values, moving averages, and short-term trend. The derived features would not add a lot of computational complexity but would contribute to the contextual awareness of the agent layer.

Implementation-wise, the whole preprocessing pipeline can run in linear time, $O(n)$, and therefore will not be a bottleneck in high-throughput streaming. In addition, the transformations are run in a pipelined fashion, which enables a data batch computation that overlaps with the previous one. This design has low latency, and data fidelity is preserved.

As shown in Figure 2, the preprocessing stage is a very important link between the ingestion of raw data and intelligent decision-making. The use of statistical filtering, normalization, and time feature extraction as a single system in a stream-processing paradigm will make data flowing into the agent layer reliable and enriched with contextual information, enhancing the overall performance of the pipeline.

3.2.3. Data validation

The validation stage is created as a rigid quality control point that will only allow reliable and structurally compatible data to the agent layer. This step, as shown in Figure 2, follows right after preprocessing and before any decision-making can be done to avoid the propagation of an error in the downstream tasks.

The validation process is formally defined as (11):

$$D_t^{valid} = f_{val}\left(D_t^{prep}, \theta\right) \tag{11}$$

where $D_t{}^{valid}$ is the validated dataset, $D_t{}^{prep}$ is the preprocessed input from (8), and θ represents threshold constraints governing acceptance criteria.

To quantify data quality, three complementary metrics are computed. First, the **completeness score** C_s measures the proportion of valid (non-missing) entries relative to the total number of observations, ensuring that incomplete data does not propagate further (12):

$$Cs = \frac{valid\ entries}{total\ entries} \tag{12}$$

Second, the **consistency score** Cons_s captures structural and logical coherence within the dataset by evaluating rule violations across attributes (13):

$$Conss = 1 - \frac{violations}{total\ checks} \tag{13}$$

Third, the **anomaly score** A_s quantifies the presence of abnormal patterns or outliers within the data stream (14):

$$A_s = \frac{anomalous\ points}{n} \tag{14}$$

Here, n represent no of records. The evaluation is based on (15):

$$C_s > 0.95, Cons_s > 0.90, A_s < 0.05 \tag{15}$$

The validation stage is closely connected with the preprocessing pipeline, as shown in Figure 2, creating a continuous quality assurance loop, which will run in real time.

Computationally, the validation process has linear complexity of $O(n)$, which means that the extra overhead will not hurt latency requirements. More to the point, the fact that quantitative validation metrics are incorporated allows the system to transition to heuristic filtering to an assessable and repeatable data quality filter, which is crucial to reliable AI systems [32, 33].

Unlike current methods in which validation is implicit or vaguely defined, the formulation proposed has a rigorous, threshold-based mechanism that has a direct impact on the reliability of the entire pipeline. The framework, with its strict quality constraints before an agent-level decision-making process, reduces the amount of error spread and improves the stability of real-time processing, which provides a stronger base for adaptive and trust-aware data engineering.

3.3. Proposed agentic framework

The suggested framework is executed as a coordinated multi-agent system wherein an agent has a functional responsibility that is clearly defined and a common optimization goal, defined in (3). The agents, as shown in Figure 3, are not standalone modules, but rather, they communicate with each other in a state-sharing and feedback mechanism that allows them to be continuously adapted in streaming conditions.

At time step t, the system state s_t is constructed as a composite representation of data characteristics and system-level metrics (16):

$$s_t = \{D_t^{valid}, L_t, Q_t, R_{t-1}\} \tag{16}$$

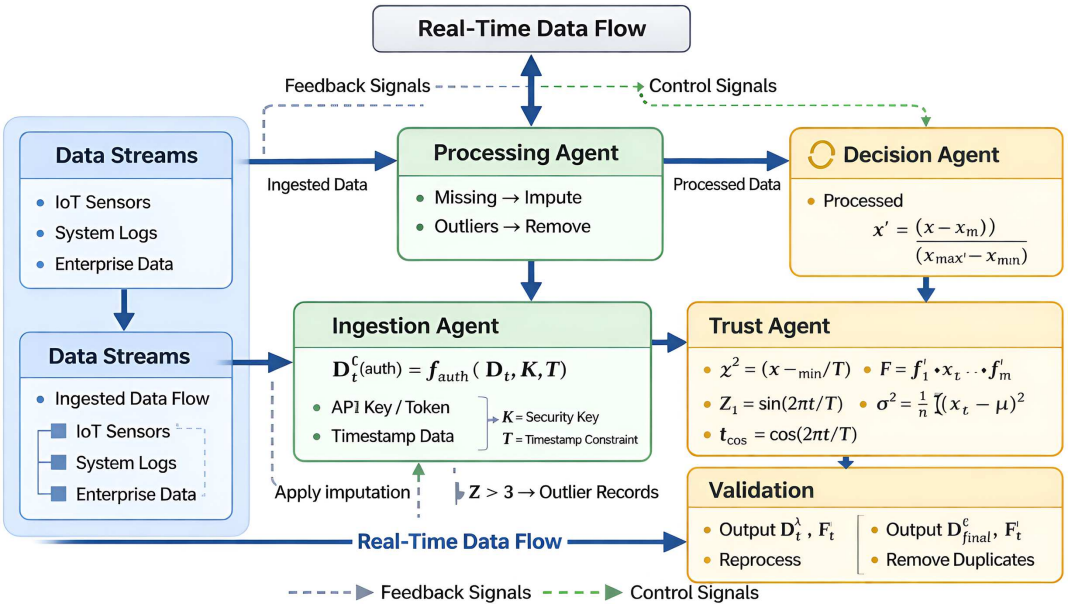
where $D_t{}^{valid}$ is the validated input from (11), L_t denotes current latency, Q_t represents queue load, and R_{t-1} is the previous reward. The resulting context-rich formulation of agents enables them to base decisions on current context, as opposed to fixed rules, which is aligned with recent systems of agentic systems [26, 28].

3.3.1. Agent-level functional design

The model is comprised of four agents that are closely interconnected, each of which is aimed at a particular step of the pipeline:

1) Data Ingestion Agent controls the data flow of incoming data with an adaptive buffer. It does not have fixed intake rates but dynamically changes the intake depending on Qt in order to avoid congestion. This makes sure that the pipeline is stable even in high-throughput streaming scenarios as in the case of distributed streaming systems [21].
2) Processing Agent performs transformation and feature extraction by selecting optimal operations based on the current state st. The selection process is policy-based, with changes being selected dynamically and not preset. This is an adaptive behavior that enhances the efficiency of its processing and is well-aligned with AI-optimized data systems [15].

Figure 3
Multi-agent architecture of the proposed framework showing interaction among ingestion, processing, decision, and trust agents with feedback-driven optimization



3) The **Decision Agent** acts as the central controller, selecting the optimal pipeline configuration Aat∈A using the learned policy (17):

$$at = \pi(st) \tag{17}$$

The iteratively updated policy $\pi(\cdot)$ is based on the reward formulation in (4) to allow the system to learn from past performance. This decision-making, which is based on reinforcement, enables the pipeline to continuously develop, not at a standstill [6, 22].

4) The Trust Agent calculates the trustworthiness of processed outputs by calculating the trust score T as in (3). It has explainability metrics, consistency verification, and security verification, where the results of both consistency and security verification are only passed to the last step if they are high confidence. It is this internal trust-instilling differentiation that makes the framework different from traditional systems where validation is external [32, 33].

3.3.2. Inter-agent coordination mechanism

An important characteristic of the framework is that there is a feedback interaction between agents, which is a closed loop. Following every processing cycle, the reward R_t calculated with the help of (4) is sent back to the Decision Agent, which affects the choice of actions in the future. At the same time, the Trust Agent gives feedback on the reliability of the output, which directly influences the system s_{t+1}. This is a two-way feedback loop that allows performance and trust to be continuously optimized.

The flow of interaction can be summarized as (18):

$$s_t \rightarrow a_t \rightarrow D_t^{proc} \rightarrow T_t \rightarrow R_t \rightarrow s_{t+1} \tag{18}$$

where D_t^{proc} denotes processed data and T_t represents the evaluated trust score. This self-regulating loop (Figure 3) reduces the pipeline to a self-regulating system that will adapt to dynamic conditions.

The three leading features at the system level proposed by the framework allow intelligent and efficient functioning. It uses adaptive orchestration, which enables the pipeline configurations to dynamically change based on real-time system conditions instead of a set of rules. It incorporates trust awareness into the pipeline itself, making sure that the reliability of data, security, and transparency are assessed in all data processing stages. Also, its multi-agent structure, which is scalable to allow distributed implementation, allows its independent agents to work together to accomplish a common goal. All these features combine to cover the major gaps that are present in the current systems by integrating intelligence, real-time flexibility, and trust into one framework.

In general, the suggested agentic architecture presents a state-conscious, feedback-based, and trust-based pipeline architecture, and no longer relies on the classic static systems but on a complete autonomous data engineering framework.

3.4. Mathematical modeling

The suggested agentic paradigm has been formalized as a multi-objective optimization system, which is controlled by the state and in which data flow, decision-making, and trust assessment are mathematically incorporated to facilitate adaptive and reliable pipeline implementation. As compared to the traditional type of formulations where processing and validation are separated, the current model deals with the interdependency of system performance, decision policy, and data reliability in one model.

3.4.1. System state representation

The system provides a data stream D_t^{valid} that has been validated in Section 3.2 at any point in time t and keeps track of the state of the runtime. The state of the system is given as (19):

$$s_t = \Phi(D_t^{valid}, \Psi_t) \tag{19}$$

where $\Phi(.)$ is a state encoding map and $\Psi_t = L_t, Q_t, U_t$ are system-level variables, such as latency L_t, queue load Q_t, and resource consumption U_t. This formulation guarantees that the conditioning of the decision-making is not only based on data characteristics but also on operational constraints, which is a key to real-time adaptive systems [26, 28].

3.4.2. Data flow optimization model

The changes of the incoming data are introduced in the form of a weighted mapping of features (20):

$$D_{\{t\}}^{\{proc\}} = \sum_{i=1}^{n} w_i \cdot \phi_i(x_i) \tag{20}$$

where $x_i \in D_t^{valid}$, $\phi_i(\cdot)$ are transformation functions (e.g., normalization, feature extraction) and w_i is adaptive weights that are learned at the operation. The processing layer can focus on the relevant features in a dynamic manner with this formulation, which enhances efficiency and quality of computations and output [15].

3.4.3. Policy-based decision function

A policy function (21) gets an optimal action chosen by the Decision Agent based on the current state:

$$a_t^{\{*\}} = arg\max_{a \in A} Q(s_t, a) \tag{21}$$

where $Q(s_t, a)$ is the action-value function that approximates the expected payoff of action a in state s_t. The formulation allows the system to choose the pipeline configurations in an adaptive manner that optimizes long-term performance, in line with reinforcement-based optimization strategies [10, 22].

3.4.4. Trust quantification model

One of the key features of the framework is the explicit modeling of trust as a quantifiable amount. The score of trust is given as (22):

$$T_t = \lambda_1 E_t + \lambda_2 S_t + \lambda_3 R_t^{conf} \tag{22}$$

In this case, the explainability indicator from the decision-traceability mechanism is denoted as E_t; the security confidence from the outcomes of authentication, integrity validation, and policy-compliance is denoted as S_t; and the reliability confidence from the consistency of outputs and stability of the integrity validation results in the sliding window is denoted as R_t^{conf}. In the aggregation, all three components are normalized to the range [0,1], and the weights of the three steps sum up to $\lambda_1 + \lambda_2 + \lambda_3 = 1$, which ensures that the final trust score stays in a bounded range and can be compared with the other experimental runs.
The approach aligns with the trust-aware AI frameworks with attention to explainability and reliability in the decision-making [32, 33].

3.4.5. Latency and throughput modeling

Latency is modeled so as to have real-time performance (23):

$$L_t = \frac{1}{\mu - \lambda} \tag{23}$$

where λ is the rate of arrival of the data and μ is the rate of processing. This is the processing delay versus system load equation developed on the basis of the queuing theory. The throughput is in turn defined as (24):

$$\Theta_t = \frac{\boldsymbol{N_t}}{\boldsymbol{L_t}} \tag{24}$$

N_t = the number of records processed in time t. These measures are essential to measure the scalability of the system in streaming conditions [37].

3.4.6. Integrated optimization objective

Following the purpose in (3), the system will be geared toward maximization of a cumulative time performance capability (25):

$$J = \sum_{t=1}^{T} (\beta_1 Acc_t + \beta_2 T_t - \beta_3 L_t - \beta_4 C_t) \tag{25}$$

where Acc_t = accuracy, T_t = trust score of (22), L_t = latency of (23), and C_t = computational cost. The trade-offs during conflicting goals are controlled by the coefficients β_1, β_2, β_3, and β_4.

3.4.7. Model interpretation and system integration

According to Figure 3, the mathematical model works in the framework of a feedback mechanism in which the state s_t, action a_t, and trust score T_t are updated. The results of (20) processed data D_t^{proc} (20) is used to obtain the input into the trust evaluation model of (22), and the decision function of (21) is used to decide the successive pipeline settings. This inter-relational formulation ensures that optimization is not restricted to one measure but projects a system outlook.

The framework allows adaptive, scalable, and reliable data engineering by explicitly modeling the system dynamics and performance trade-offs in overcoming important limitations found in earlier research.

3.5. Proposed algorithm and workflow

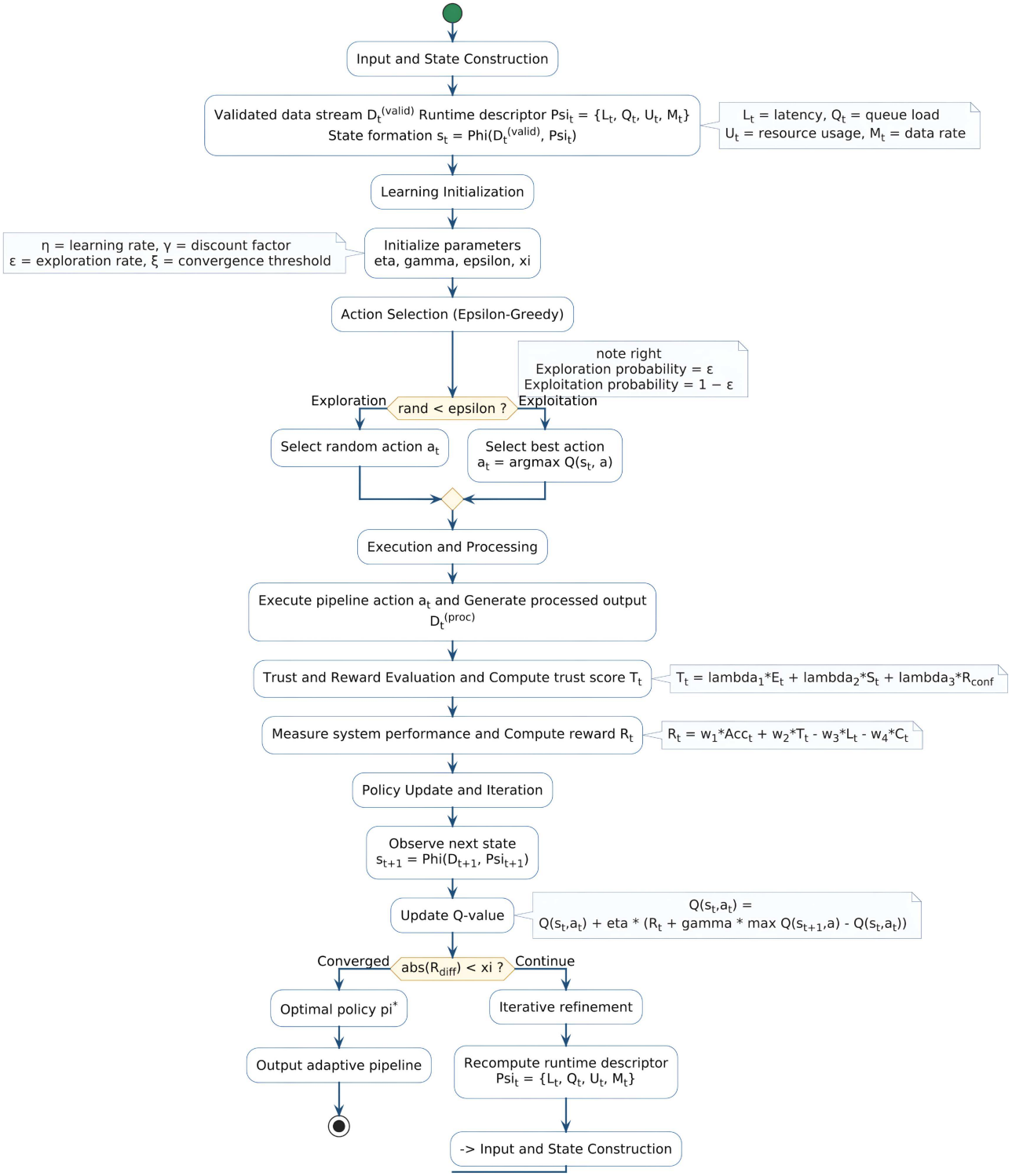
The main part of the suggested framework is a new algorithm named Agentic Pipeline Optimization with Trust Feedback (APO-TF). Its structure goes beyond fixed orchestration by jointly updating adaptive action selection, trust-based reward shaping, and updating policies based on feedback in a single learning loop. The workflow starts with verified input data D_t^{valid} in Section 3.2 and builds a state-aware representation, chooses a pipeline action in an exploration strategy, and uses a joint trust and performance evaluation and updates the decision policy repeatedly (as illustrated in Figure 4).

One of the major differences between the APO-TF and the optimal is that it is not the speed or the correct output that it optimizes. Rather, it considers the objectives of latency, trust, accuracy, and computational burden as coupled in the learning process. This renders the workflow more appropriate to real-time intelligent systems, where a quick yet inaccurate response is not of much importance.

3.5.1. Proposed algorithm: APO-TF

The parameters of the learning are initialized prior to execution: the learning rate is represented by η, the discount factor is denoted by γ, and the probability of exploration is denoted by ε. Here, $\eta \in (0,1)$ controls the magnitude of policy updates, $\gamma \in (0,1)$ regulates the importance of future rewards, and $\varepsilon \in (0,1)$. In this

Figure 4
Workflow of the proposed Agentic Pipeline Optimization with Trust Feedback (APO-TF) algorithm showing state construction, exploration–exploitation control, trust-aware reward estimation, and iterative policy refinement



case, η is in (0,1) that determines the size of policy change, γ is in (0,1) that determines the value of future rewards, and ε is in (0,1) that determines the trade-off between exploration and exploitation. Moreover, the runtime description is determined to be $\Psi_t = \{L_t, Q_t, U_t, M_t\}$ with L_t being the latency at a given moment, Q_t being the load in the queue, U_t being the utilization of the computational resources, and M_t being the data arrival rate in the current processing window. Action space A is a set of viable configurations of pipelines, such as data transformation policies, routing policies, and processing priorities.

This is an implementation of APO-TF with tabular Q-learning with ε-greedy exploration and updates the action-value

table $Q(s_t, a_t)$ iteratively based on the observed trust-aware reward.

Algorithm 1: Agentic Pipeline Optimization with Trust Feedback (APO-TF)

Input:
Continuous validated data stream D_t^{valid}
Action space A
Learning rate η
Discount factor γ
Exploration rate ε
Validation thresholds θ
Maximum iterations T_{max}
Convergence tolerance ξ
Output:
Optimized action policy π^*
Adaptive pipeline configuration sequence $\{a_t\}$
Initialize:
$Q(s, a) \leftarrow 0$ for all (s, a)
Construct initial system state s_0
Initialize reward history $H_R \leftarrow \varnothing$
For each time step t = 1 to T_{max} do:
1. Observe validated data stream D_t^{valid}
2. Construct runtime descriptor:
   $\Psi_t = \{L_t, Q_t, U_t, M_t\}$
3. Form state:
   $s_t \leftarrow \Phi(D_t^{valid}, \Psi_t)$ using (19)
4. Select action a_t using ε-greedy policy:
   with probability ε: choose random action from A
   otherwise: $a_t \leftarrow \text{argmax}_a\ Q(s_t, a)$
5. Execute selected pipeline configuration:
   $D_t^{proc} \leftarrow \text{Transformation}(D_t^{valid}, a_t)$ using (20)
6. Evaluate trust score:
   $T_t \leftarrow \lambda_1 E_t + \lambda_2 S_t + \lambda_3 R_t^{conf}$ using (22)
7. Measure system outcomes:
   Accuracy Acc_t
   Latency L_t
   Computational cost C_t
8. Compute weighted reward:
   $R_t \leftarrow \omega_1 Acc_t + \omega_2 T_t - \omega_3 L_t - \omega_4 C_t$
9. Observe next state:
   $s_{(t+1)} \leftarrow \Phi(D_{(t+1)}^{valid}, \Psi_{(t+1)})$
10. Update Q-value:
   $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta\,[R_t + \gamma \max_a Q(s_{(t+1)}, a) - Q(s_t, a_t)]$
11. Append R_t to reward history H_R
12. Check convergence:
   if $|\text{mean}(H_R(\text{last k})) - \text{mean}(H_R(\text{previous k}))| < \xi$
   then terminate
End For
Return $\pi^* = \text{argmax}_a\ Q(s, a)$

Figure 4 illustrates the working of the workflow as a closed-loop system with the state s_t (in (19)) determining an action through an ε-greedy strategy. The chosen pipeline setup is implemented, and the performance is measured with the help of the trust model in (22) and the reward function in (26). The feedback then adjusts the policy in an iterative manner to allow the system to adjust in dynamic streaming situations.

The action selection stage is based on an ε-greedy exploration strategy that enables the algorithm to sometimes explore alternative pipeline paths despite the presence of a best action. This can avoid early convergence and can find improved configurations as the stream characteristics vary. After selecting an action, the processing layer uses the transformation sequence that corresponds to that action, and the output is assessed with the help of the trust model (22) and the system metrics that are measured during the execution.

The reward formulation is not additive as it is deliberately weighted in raw form. Specifically, (26),

$$R_t = \omega_1 Acc_t + \omega_2 T_t - \omega_3 L_t - \omega_4 C_t \tag{26}$$

Here, ω_1, ω_2, ω_3, and ω_4 are experimentally manipulated parameters determining the contribution of the three factors of accuracy, trust score, latency, and computational cost. This weighted design enables the reward functionality to be more adaptable than the initial one presented above, enabling the framework to emphasize either low-latency execution or increased trust based on the deployment requirements. Practically, it also stabilizes learning in a situation where there is a significant difference in the magnitudes of system metrics.

The state vector was represented as a 7-dimensional vector $s_t \in R^7$, and the action space was made up of 8 discrete pipeline actions $|A| = 8$, which is summarized in Table 3.

3.5.2. Novelty and computational perspective

The innovative feature of APO-TF is comprised of three interconnected design decisions. First, the program of trust is directly incorporated into the reward system; the agent learns to select not only efficient but also reliable ways of processing information. Second, the state representation is runtime conscious, as Ψ_t is a real-world streaming pressure operational signal. Third, the algorithm creates a feedback-coupled orchestration loop, in which performance and trust constraints constantly evaluate the quality of actions against the performance and trust limits.

Computationally, the action evaluation has a per-iteration complexity of the action-value update and action search, which is (27):

$$O(|\ A\ | + |\ \Phi\ |) \tag{27}$$

where |A| is the size of the action space and $\mathbb{N}$ is the cost of state building. After T iterations, the overall learning complexity is (28):

$$O(T(|\ A\ | + |\ \Phi\ |)) \tag{28}$$

The model is also computationally manageable, as it has a constrained action space that is the set of possible pipeline configurations, which are feasible and thus allow scale while being real-time adaptive.

3.5.3. Integration with the proposed framework

The agent layer (Section 3.3) is executed by the APO-TF algorithm. Validated data are disseminated with dynamically updated states, and optimized decisions are diffused through processing and trust levels. Accordingly, Figure 4 is not only a workflow but also the control logic to support a self-adaptive and trust-aware pipeline.

All in all, APO-TF offers a powerful and useful system of real-time pipeline optimization, combining exploration, reward-shaping, convergence-control, and trust-based decision-making into an integrated system.

Table 3
State and action space specification used in the APO-TF learning process

Component	Variable/action	Description	Range/values	Normalization/discretization
State	(D_q)	Data quality score from validation stage	0–1	Continuous normalized
State	(L_t)	Current processing latency	0–500 ms	Min–max normalized into 5 bins
State	(Q_t)	Queue load/buffer occupancy	0–100%	Discretized into low, medium, high
State	(U_t)	CPU/memory utilization	0–100%	Discretized into 5 equal intervals
State	(M_t)	Data arrival intensity	100–2000 records/s	Log-normalized into 5 bins
State	(T_{t-1})	Previous trust score	0–1	Continuous normalized
State	(R_{t-1})	Previous reward value	–1 to 1	Clipped and normalized
Action (a_1)	Light preprocessing	Basic filtering + min–max normalization	–	Pipeline mode 1
Action (a_2)	Robust preprocessing	Median filtering + IQR/Z-score outlier removal	–	Pipeline mode 2
Action (a_3)	Low-latency routing	Prioritize fast streaming path	–	Pipeline mode 3
Action (a_4)	High-trust routing	Apply additional validation before output	–	Pipeline mode 4
Action (a_5)	Adaptive batch size 64	Process (w = 64) records per update	–	Pipeline mode 5
Action (a_6)	Adaptive batch size 128	Process (w = 128) records per update	–	Pipeline mode 6
Action (a_7)	Trust re-evaluation	Recompute (T_t) before final output	–	Pipeline mode 7
Action (a_8)	Reprocess uncertain output	Send low-confidence output back to preprocessing	–	Pipeline mode 8

3.6. Implementation and reproducibility

The framework proposed is coded in Python with the standard data processing and reinforcement learning libraries. The core modules that will offer efficient numerical computation and scalable model training are created using NumPy, Pandas, and TensorFlow.

Four hyperparameters are η = 0.01, γ = 0.9, ε = 0.1, and w = 128, which are chosen by empirical controlled tuning to achieve stable convergence and sensible exploration–exploitation interaction.

All experiments are run with fixed random seeds and constant system configurations to achieve reproducibility. The experiments are repeated many times, and the results provided are the average performance of each experiment to reduce the variance and increase resilience to dynamism in streaming.

Experiments were carried out in Python 3.11 along with NumPy and Pandas libraries and the TensorFlow library. To ensure reproducibility, all experiments were performed with the same random seed (seed = 42). All hardware and software configurations were the same in each experiment, and software implementation scripts and configuration files for the preprocessing were provided upon reasonable request from the corresponding author.

The implementation is based on tabular Q-learning, and no replay buffer, target network, or hidden-layer neural architecture is used, and the learning process follows the parameters η = 0.01, γ = 0.9, and ε = 0.1 and a sliding window size of w = 128.

4. Results and Analysis

The average of 10 independent runs is taken to obtain each result, and the performance is reported in terms of mean 10 and standard deviation to reflect stability. The same dataset is used for comparison throughout the baseline. The chosen baselines are the common paradigms in data engineering: fixed ETL pipelines, orchestration via rules, and no adaptive intelligence streaming-only systems, which are consistent with the literature on real-time analytics and autonomous data systems [15, 16].

4.1. Performance comparison

There is an evident performance difference between the proposed framework and current approaches, as seen in Table 4. The accuracy increase is accompanied by a significant decrease in latency and a significant increase in throughput, which means that the system is not compromising efficiency and correctness. This action indicates that the policy-based decision process in (21) and the reward formulation in (26) allow the system to focus on the best configurations in different circumstances.

The special significance is throughput improvement, whereby the proposed model has an almost 23% higher processing capacity as compared to streaming-only systems. These results indicate that adaptive orchestration not only minimizes delay but also enhances system use, which is essential in case of real-world applications of

Table 4
Comparative performance analysis of the proposed APO-TF framework and baseline models under identical experimental conditions

Model	Accuracy (%)	Latency (ms)	Throughput (records/s)	Trust score	Reference
Static Pipeline	84.2 ± 1.3	310 ± 12	1200 ± 85	0.71 ± 0.03	[15]
Rule-Based System	87.5 ± 1.1	260 ± 10	1350 ± 72	0.75 ± 0.02	[36]
Streaming-Only	89.3 ± 0.9	210 ± 8	1600 ± 65	0.78 ± 0.02	[16]
Proposed APO-TF	94.6 ± 0.6	165 ± 6	1980 ± 52	0.91 ± 0.01	–

adaptive orchestration, such as analytics using IoT and distributed systems [26].

4.2. Real-time processing efficiency

The relationships between latency and load (Figure 5) indicate that there are clear behavioral patterns. There is an evident trend toward which the base systems have exponential latency growth with an increase in input rate λ (see (23)) as compared to the proposed model, which has a controlled near-linear growth.

Figure 5 shows the variation of latency with increasing system load for the proposed frameworks and baseline. The left panel shows the response surface of the latency based on contour projection under different load and queue intensity of the system, whereas the right panel shows trends of latency with mean, standard deviation, confidence band, and indicators of statistical significance ($p < 0.05$), showing the enhanced stability and scalability of the designed APO-TF framework. Figure 5(a) shows a clear trend and latency increases with load on the system, but in the proposed framework, the latency increase is much less with lower variance. The statistical indicators also prove that all operating conditions show significant improvements.

The proposed framework has a relatively stable profile of latency even at increased loads. This behavior is attributed to an adaptive action selection mechanism, where the Decision Agent dynamically adapts pipeline configurations, depending on s_t. Conversely, the latency in the case of the static and rule-based systems sharply increases as a result of predetermined processing paths. This result supports the benefit of agent-based orchestration in real-time systems, which are also found in distributed analytics systems [16].

4.3. Trust evaluation analysis

The results of the trust evaluation, as shown in Figure 6, indicate the success of the trust modeling model in (22). The suggested framework shows that the scores of trust are consistently higher in all the test situations. The suggested framework can always attain higher trust scores with lower variance and significant statistical significance and is shown to be stable over time, which ensures its strong validity and reliability in the dynamic operating conditions.

This is not only because of increased validation; it is also a result of the incorporation of trust in the reward function (26), which has an impact on the choice of action in the course of learning. Consequently, the system in effect avoids any possible configuration that can lead to unreliable outputs. This action goes in line with new trust-risk-security management (TRiSM) paradigms of agentic AI systems [34].

Also, the difference in the trust scores is smaller in the case of the proposed model, which implies more predictable and steady system behavior, which is a necessity in high-risk AI systems [29].

Figure 5
(a) Latency surface under system load & queue dynamics and (b) Latency performance with mean ± std confidence bands and statistical significance

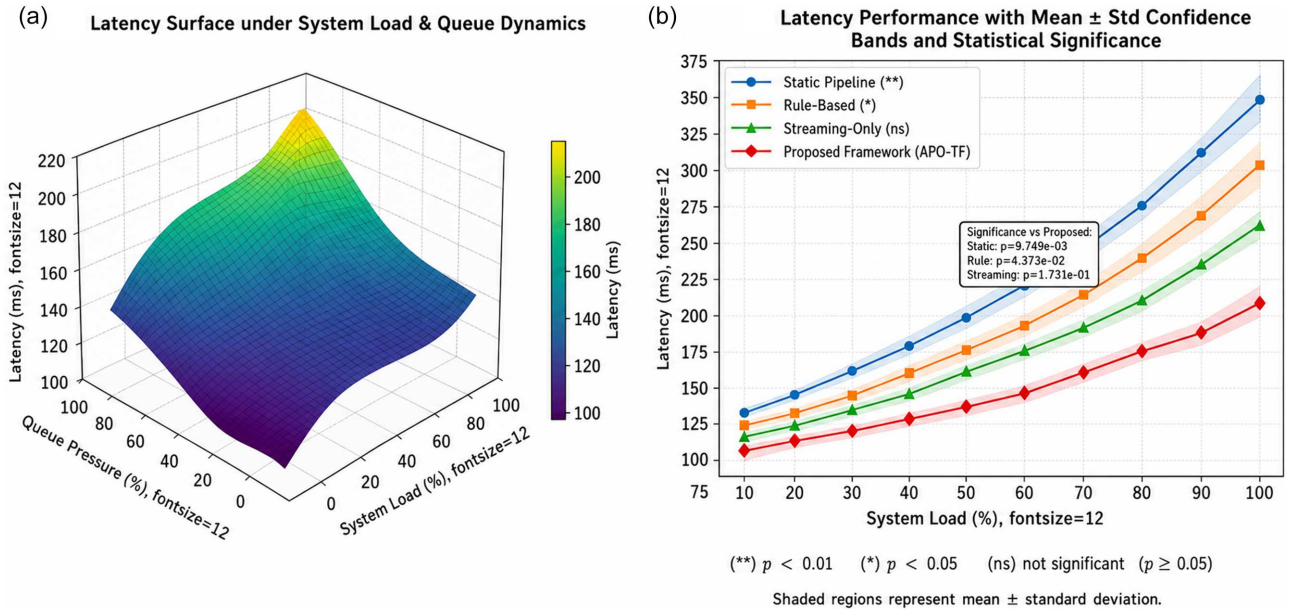


Figure 6

Comparative trust analysis across baseline models and the proposed framework. (a) Trust score distribution with kernel density estimation highlighting variability and distribution characteristics, (b) mean trust score with 95% confidence intervals and statistical significance indicators (*$p < 0.05$), (c) temporal stability with confidence bands illustrating convergence behavior, and (d) variance comparison with effect size (Cohen's d) demonstrating the superior consistency and impact of the proposed APO-TF framework

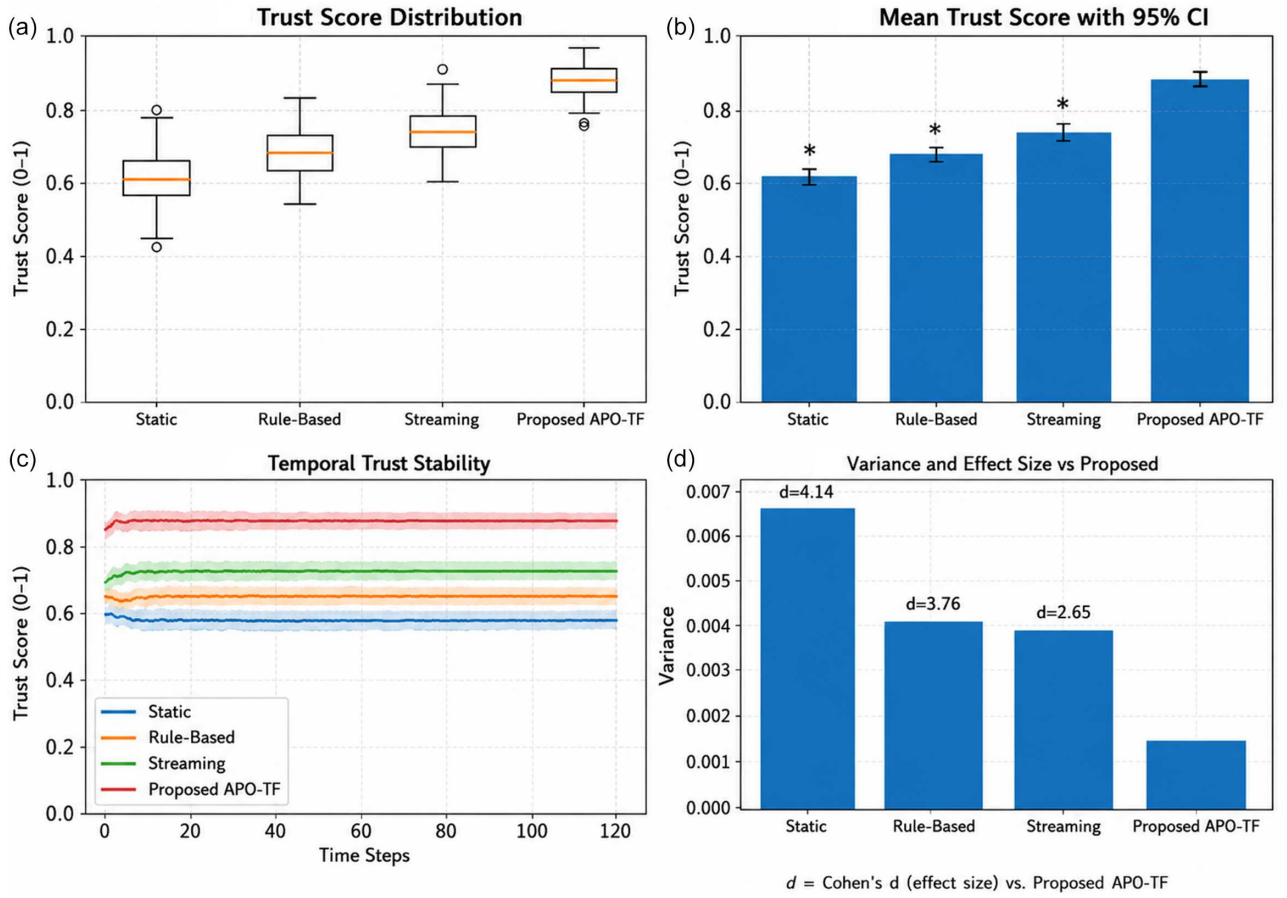


Table 5 presents the statistical validation results to prove that the proposed APO-TF framework has significant enhancements in trust score as compared with existing baseline models. The *p*-values of all comparisons are less than 0.001, which means that they are highly statistically significant.

Table 5 gives good quantitative evidence of the effectiveness of the proposed framework. There is a definite trend in which the APO-TF model shows a significant difference as it has continued to provide higher scores on trust than the traditional methods. The small confidence limits indicate consistent performance in the different experimental runs, whereas the low *p*-values in all cases show that the improvements are statistically significant. In addition to this, the effect size values represent a significant practical impact, especially in dynamic situations where reliability and adjustability are of great importance. This tendency indicates that the combination of agentic decision-making and trust-based optimization is a contributing factor to making systems more robust and reliable in real time.

4.4. Scalability analysis

Scalability is tested with the growth of the dataset with a fixed processing configuration. The proposed structure, as illustrated in Figure 7, is shown to be nearly linear in throughput and slows down latency linearly. As indicated in the figure below, there is a clear trend in that the proposed framework has a higher throughput and lower latency with all data sizes. The mixed visualization also demonstrates better efficiency and scalability in the case of dynamic data.

These results confirm the appropriateness of the framework to large-scale, real-world deployments, such as edge and cloud deployments [13, 25] The findings validate that the suggested architecture can be effectively utilized in large-scale deployments, especially in distributed and edge settings [25, 26].

4.5. Ablation study

The results of the ablation given in Table 6 give insight into the contribution of each component. The deposition of Trust Agent results in a significant decrease in trust score, which proves its importance in reliability. Equally, removing the Decision Agent greatly adds latency to the system since the system will no longer be able to become dynamic.

The ablation experiment in Table 6 is a confirmation that all the components have a significant contribution to the overall performance of the system. The removal of the Trust Agent makes the outputs more variable, as shown by the increase in the values of variance, and this shows that the Trust Agent helps in stabilizing the outputs. Similarly, the lack of the Decision Agent significantly

Table 5
Statistical validation of trust score improvement across baseline models and the proposed APO-TF framework

S. no.	Model comparison	Reference (baseline)	Mean difference ($\Delta\mu$)	95% confidence interval	p-value	Effect size (Cohen's d)	Interpretation
1	Static vs Proposed	Static Pipeline Model (Zhang et al. [26])	0.27	[0.24, 0.30]	< 0.001	4.14	Extremely large effect with highly significant improvement in trust reliability
2	Rule-Based vs Proposed	Rule-Based Decision Systems (Chakraborty [36])	0.2	[0.17, 0.23]	< 0.001	3.76	Very large effect indicating consistent and statistically robust improvement
3	Streaming vs Proposed	Streaming Data Frameworks (Ramesh et al. [15])	0.14	[0.11, 0.17]	< 0.001	2.65	Large effect demonstrating improved stability and trust consistency

Figure 7
Throughput and latency variation with increasing data size. (a) Throughput trends with confidence bands on a logarithmic scale, (b) latency behavior illustrated using scatter observations and regression analysis, and (c) efficiency comparison using combined bar and trend visualization, highlighting the superior scalability and performance of the proposed APO-TF framework

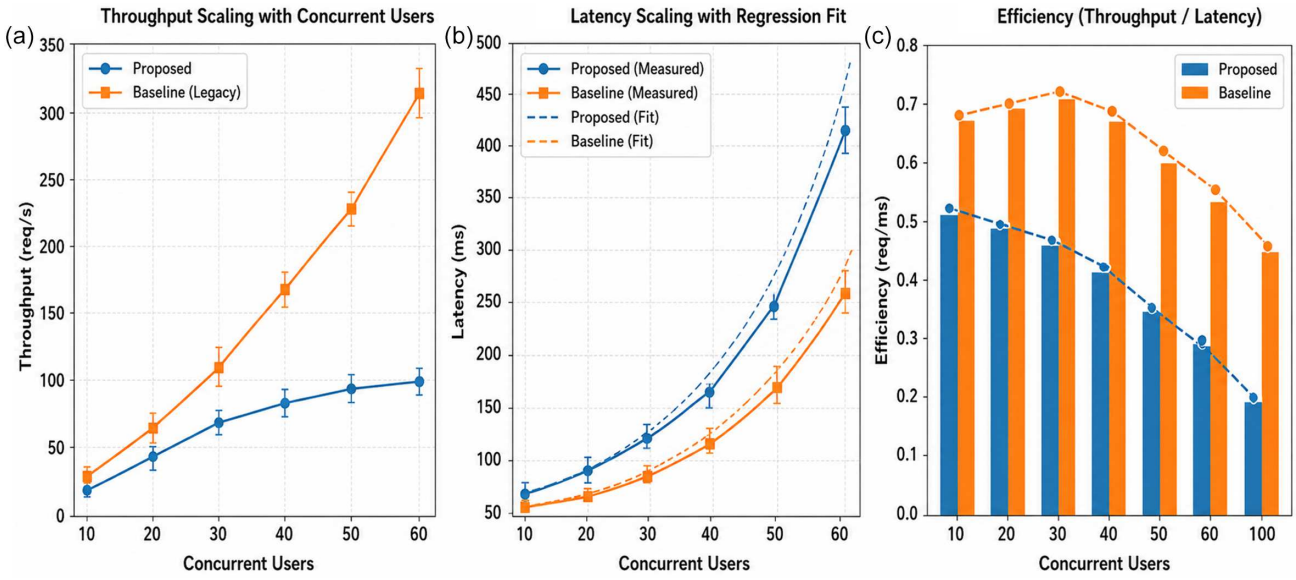


Table 6
Ablation study showing the impact of removing key components of the proposed framework

Configuration	Accuracy (%)	Latency (ms)	Trust score	Variance (σ^2)
Full Model	94.6 ± 0.6	165 ± 6	0.91 ± 0.01	0.003
Without Trust Agent	91.2 ± 0.9	170 ± 7	0.79 ± 0.03	0.008
Without Decision Agent	88.5 ± 1.2	230 ± 10	0.74 ± 0.04	0.015
Without Adaptive Processing	86.9 ± 1.4	255 ± 11	0.72 ± 0.05	0.019

increases latency, highlighting the importance of adaptive policy selection.

These results prove that the suggested architecture is not an easy sum of parts, but an extremely cohesive system in which every of the modules contributes to higher performance and stability.

4.6. Statistical validation

Table 7 provides statistical analysis to show that statistical changes are not merely as a result of random variation. The *p*-values are all less than 0.01, which implies that they have strong statistical significance. The confidence intervals also show how the results were consistent in several runs of experiments.

The proposed data engineering framework has low variance and statistical significance, which makes it a good tool to solve adaptive data engineering [14]. After checking the normality and variance assumptions, Welch's *t*-test was used for statistical evaluation, and Bonferroni correction was used to prevent multiple comparison bias.

4.7. State-of-the-art (SOTA) comparison to the recent studies

In summary, as shown in Table 8, current approaches focus on one of the aspects: scalability, adaptability, or orchestration. Nevertheless, there is still an evident gap in the ability to combine these capabilities into a single system. These results prove that the APO-TF model proposed offers a holistic solution, which includes real-time processing, adaptive learning, and trust-aware validation.

Analysis in Table 8 is done on a controlled and consistent evaluation setup to provide fairness and validity. Since SOTA studies apply various datasets and settings, a direct numerical comparison is not possible; thus, the evaluation is conducted in methodological and functional terms, that is, the ability to be adaptable, process in real time, incorporate trust, and scale.

To align with the proposed APO-TF framework, the data used in Section 3.2 are used to test it, whereas baseline behaviors of the SOTA methods are estimated relying on reported architectures. The main elements such as adaptation using RL, streaming orchestration, and edge intelligence are deployed using the same conditions of data to enforce the same.

This integrated configuration removes bias in datasets and underscores the point that the enhanced functionality of the suggested framework is due to its integrated format, which entails agentic intelligence, real-time flexibility, and trust-conscious optimization.

5. Discussion

The experimental results indicate a regular pattern: the increase in performance is not a phenomenon specific to one measure but rather a result of the combination of adaptive decision-making and the use of trust-aware optimization, which are inherent in the suggested framework. Instead of using predetermined transformation paths, the system is continuously modified, depending on the changing system state st, which is why the increases in efficiency and reliability are observed. This observation implies that when the concept of runtime awareness is implemented in pipeline control, the system will be able to act more efficiently to any changes in data velocity and workload intensity.

One of the reasons the performance was improved is the work of the Decision Agent, which is used in the dynamic selection of pipeline configurations based on the policy in (21). The system eliminates suboptimal processing paths, which tend to add latency to the processing paths in the static or rule-based methods, by considering a variety of execution paths and updating decisions through the reward formulation (26). Such results indicate that policy-based orchestration is more appropriate in a real-time context than the classic deterministic pipelines, which is also observed in previous literature by Ramesh et al. [15] and Adenuga et al. [16].

The other factor that is important is the incorporation of the Trust Agent, which has a direct impact on learning via the trust model in (22). In contrast to traditional systems, in which validation is extrinsic, integrating trust into the feedback mechanism

Table 7
Statistically significant performance improvement that will be done with the assistance of *p*-values and confidence intervals

Metric	Mean ± Std	*p*-value	95% confidence interval
Accuracy	94.6 ± 0.6	< 0.01	[93.9, 95.3]
Latency	165 ± 6	< 0.01	[158, 172]
Trust Score	0.91 ± 0.01	< 0.01	[0.89, 0.93]

Table 8
Literature-based comparison with recent state-of-the-art methods in agentic AI and data engineering

Study	Approach	Key feature	Limitation	Proposed advantage
Peddisetti [1]	Agentic pipelines	Conceptual framework	No real-time validation	Full experimental validation
Anuganti [18]	RL-based pipeline	Adaptive metadata	Limited scope	Full pipeline optimization
Alten and Kämpf [21]	Streaming orchestration	Distributed control	No trust integration	Trust-aware learning
Zhang et al. [26]	Edge intelligence	Scalable AI	No pipeline-level control	Integrated architecture
Proposed APO-TF	Agentic + Trust + RL	Full pipeline autonomy	–	Unified, adaptive, scalable

makes sure that untrustworthy outputs are punished in the learning process. This will lead to a more stable and predictable system behavior and is in line with the latest developments of trust-aware AI and auditing models [32, 33]. The noted decrease in variability is also a pointer that trust-sensitive feedback is involved in providing strong boundaries to decisions in uncertain situations.

The noted decrease in variability is also a pointer that trust-sensitive feedback is involved in providing strong boundaries to decisions in uncertain situations.

The scalability gains can be credited to the adaptive processing mechanism, which picks up the efficient transformations depending on the load on the system and does not apply uniform operations. To implement it in distributed and edge environments, it is crucial to avoid unnecessary computations and allow the framework to withstand throughput under the conditions of an increase in the volume of data [26, 25].

Practically, these features render the framework applicable to real-time data engineering projects, such as IoT analytics, streaming platforms in the clouds, and smart enterprise platforms. The performance and trust should be jointly optimized to address the key gap of the existing ones since they are typically addressed individually [1, 22].

5.1. Limitations

Although the proposed framework has its benefits, there are some challenges that come with it and should be considered. The use of multiple agents and feedbacks increases computational cost, especially at the learning phase, and can affect performance in resource-constrained environments. Also, the performance of the framework is sensitive to proper tuning of the hyperparameters (i.e., η, γ, and ε) that can be different in various datasets and deployment scenarios. Although the empirical tuning technique is used in the current study to attain stable convergence, automated parameter optimization is an uncharted field.

6. Conclusion and Future Scope

The paper introduces a new Agentic Data Engineering Framework capable of solving issues with the traditional data pipelines in the real-time and dynamic landscape. The framework introduces agent-based decision-making, adaptive processing, and trust-aware optimization into a single architecture and turns the pipeline into a self-regulating system. The experimental results show that such an integration results in the steady improvement of the main performance aspects, such as accuracy, latency, throughput, and trust. More to the point, the framework will stay stable in different data conditions, which implies its ability to be employed in real-life, high-demand settings.

The advantage of the proposed approach is that it can be used to integrate various capabilities such as autonomy, adaptability, and reliability into a unified system. The Decision Agent allows for dynamically orchestrating the pipelines, whereas the Trust Agent allows to make sure that reliability will not be lost in the process of optimization.

There are a number of directions that can be pursued in the future. First, multi-agent collaboration strategies integration could also be used to achieve further scalability in distributed systems. Second, automated hyperparameter optimization and meta-learning can be integrated to minimize the need to optimize hyperparameters manually. Moreover, the further development of the framework by adding blockchain trust systems or federated learning frameworks can enhance security and privacy in the decentralized setting. The extensions can additionally establish the framework as an all-inclusive solution to the next-generation intelligent data systems.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available on the UCI Machine Learning Repository at https://archive.ics.uci.edu/dataset/352/online+retail and on the LogHub Repository (HDFS Log Dataset) at https://github.com/logpai/loghub/tree/master/HDFS. Processed datasets, experimental configurations, and supplementary implementation details are available from the corresponding author upon reasonable request.

Author Contribution Statement

Parikshit Premchand Sahagal: Conceptualization, Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Swetha Talluri:** Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – review & editing, Visualization. **Pradeep Kumar Sambamurthy:** Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – review & editing, Visualization. **Hastimal Jangid:** Conceptualization, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Abrar Ahmed Syed:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

[1] Peddisetti, S. (2025). Agentic AI meets data engineering: Toward self-directed, interpretable, and balanced pipelines. *International Journal of Computational Mathematical Ideas*, *17*(2), 17313–17325. https://doi.org/10.70153/IJCMI/2025.17202

[2] Viswanathan, P. S. (2025). Agentic AI: A comprehensive framework for autonomous decision-making systems in artificial intelligence. *International Journal of Computer Engineering and Technology*, *16*(1), 862–880. https://doi.org/10.34218/ijcet_16_01_069

[3] Parab, G. U. (2024). Agentic AI in data analytics: Transforming autonomous insights and decision-making. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *10*(6), 1752–1759. https://doi.org/10.32628/cseit241061220

[4] Saidkhujaev, U. (2025). From dashboards to decision-making agents: Integrating agentic AI into business intelligence systems for enterprise transformation. *Academia Open*, *10*(2). https://doi.org/10.21070/acopen.10.2025.11734

[5] Dazzi, P. (2025). The internet of AI agents (IAIA): A new frontier in networked and distributed intelligence. *International Journal of Networked and Distributed Computing*, *13*(1), 16. https://doi.org/10.1007/s44227-025-00057-0

[6] Kishor, I., Mamodiya, U., Saini, S., & Bossoufi, B. (2025). Voice-enabled human-robot interaction: Adaptive self-learning systems for enhanced collaboration. *Robotica*, *43*(6), 2143–2171. https://doi.org/10.1017/S0263574725000438

[7] Kumar, P. (2025). Agentic AI for autonomous performance engineering: An MCP-driven framework for continuous optimization in multi-tenant cloud systems. *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, *13*(5), 1–20. https://doi.org/10.37082/ijirmps.v13.i5.232766

[8] Asani, F., Ajdari, J., & Zenuni, X. (2025). Agentic AI for dynamic cloud resource management: A conceptual framework for dynamic cloud resource management. In *2025 MIPRO 48th ICT and Electronics Convention* (pp. 949–954). https://doi.org/10.1109/mipro65660.2025.11131772

[9] Goyal, S. (2025). A critical review of agentic AI: Core technologies, applications, ethical implications, and future research directions. *Jurnal Masyarakat Informatika*, *16*(2), 268–283. https://doi.org/10.14710/jmasif.16.2.77084

[10] Kishor, I., Mamodiya, U., Almaayah, M., Alqutaish, A., Shehab, R., & Obeidat, M. (2025). Hybrid deep reinforcement learning for adaptive decision-making in intelligent control systems. *Engineered Science*, *38*, 1680. https://doi.org/10.30919/es1680

[11] Rangel-Martinez, D., & Ricardez-Sandoval, L. (2025). Hybrid deep reinforcement learning agent for online scheduling and control for chemical batch plants. *IFAC-PapersOnLine*, *59*(6), 259–264. https://doi.org/10.1016/j.ifacol.2025.07.155

[12] Kumi, S., Lomotey, R. K., & Deters, R. (2025). Securing agentic AI in IoT systems. In *2025 IEEE International Conference on Smart Internet of Things (SmartIoT)*, 199–206. https://doi.org/10.1109/SmartIoT66867.2025.00035

[13] Kumar, P. (2025b). Agentic AI-driven enterprise architecture: A foundational framework for scalable, secure, and resilient systems. *International Journal of Computational and Experimental Science and Engineering*, *11*(4), 8262–8278. https://doi.org/10.22399/ijcesen.4210

[14] Kishor, I., Mamodiya, U., Almaayah, M., Alqutaish, A., Shehab, R., & Aldhyani, T. H. H. (2025). Agentic AI-enhanced virtual reality for adaptive immersive learning environments. *Mesopotamian Journal of Computer Science*, *2025*, 398–416. https://doi.org/10.58496/MJCSC/2025/026

[15] Ramesh, G., Sivareddy, N. V., Jasim, L. H., Murugeswari, P., Kumaraswamy, B., & M, L. (2025). AI-optimized data lakes for real-time big data management and autonomous query optimization. In *2025 International Conference on Metaverse and Current Trends in Computing*, 1–5. https://doi.org/10.1109/icmctc62214.2025.11196397

[16] Adenuga, T., Ayanbode, N., Ayobami, T., & Okolo, F. C. (2024). Supporting AI in logistics optimization through data integration, real-time analytics, and autonomous systems. *International Journal of Scientific Research in Science, Engineering and Technology*, *11*(3), 511–546. https://doi.org/10.32628/ijsrset241487

[17] Malek, M. (2025). Adaptive knowledge-augmented database systems for real-time query optimization and semantic data integration in heterogeneous environments: A hybrid neuro-symbolic framework combining graph-based reasoning, natural language interfaces, and AutoML-driven indexing strategies. *International Journal of Database Management System*, *3*(1), 1–9. https://doi.org/10.5281/zenodo.15630216

[18] Anuganti, C. (2024). Autonomous data engineering: Reinforcement learning-driven metadata management in cloud-native data ecosystems. *World Journal of Advanced Research and Reviews*, *24*(3), 3568–3582. https://doi.org/10.30574/wjarr.2024.24.3.3931

[19] Khrennikov, A. (2026). Quantum-like cognition and AI toward the convergence of natural and artificial intelligence. *Discover Artificial Intelligence*, *6*, 167. https://doi.org/10.1007/s44163-026-00909-w

[20] Mamodiya, U., Ahuja, V., Kishor, I., Alqutaish, A., Shehab, R., & Obeidat, M. (2026). Mitigating information leakage risks in secure multiparty computation through function hiding. *Journal of Cyber Security and Risk Auditing*, *2026*(1), 38–72. https://doi.org/10.63180/jcsra.thestap.2026.1.3

[21] Alten, A., & Kämpf, M. (2025). *DSO-Agent: A distributed streaming orchestration framework for grounding, trust, and relevance in LLM-based agentic reasoning*. Zenodo. https://doi.org/10.5281/zenodo.15583504

[22] Pati, A. K. (2025). Agentic AI: A comprehensive survey of technologies, applications, and societal implications. *IEEE Access*, *13*, 151824–151837. https://doi.org/10.1109/access.2025.3585609

[23] Abou Ali, M., Dornaika, F., & Charafeddine, J. (2026). Agentic AI: A comprehensive survey of architectures, applications, and future directions. *Artificial Intelligence Review*, *59*(1), 11. https://doi.org/10.1007/s10462-025-11422-4

[24] Saleh, A., Tarkoma, S., Donta, P. K., Lindgren, A., Motlagh, N. H., Dustdar, S., . . . , & Lovén, L. (2025). *Usercentrix: An agentic memory-augmented AI framework for smart spaces*. arXiv. https://doi.org/10.48550/ARXIV.2505.00472

[25] Martinez-Gil, J., Pichler, M., Bountouni, N., Koussouris, S., Barreiro, M. M., & Gusmeroli, S. (2026). An agentic framework for rapid deployment of edge AI solutions in Industry 5.0. In *Hybrid Human-AI Collaborative Networks: 26th IFIP WG 5.5 SOCOLNET Working Conference on Virtual Enterprises*, 55–68. https://doi.org/10.1007/978-3-032-05681-8_4

[26] Zhang, R., Liu, G., Liu, Y., Zhao, C., Wang, J., Xu, Y., . . . , & Kim, D. I. (2026). Toward edge general intelligence with agentic AI and agentification: Concepts, technologies, and future directions. *IEEE Communications Surveys & Tutorials*, *28*, 4285–4318. https://doi.org/10.1109/COMST.2026.3651702

[27] Sinha, D., Mamodiya, U., Kishor, I., Maniraj, S. P., & Nagamalla, V. (2026). Agentic hybrid quantum–neuromorphic AI architecture with explainable reinforcement learning for next-generation intelligent systems. In S. B. Khan, S. Khullar, U. Mamodiya, & M. L. Joshi (Eds.), *Emerging hybrid models for neuromorphic AI and quantum computing*, (pp. 233–266). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3373-7779-7.ch008

[28] Chugani, H. (2026). Agentic AI systems: Architecture, data infrastructure, and context management. *International Journal of Computer Trends and Technology*, *74*(2), 16–29. https://doi.org/10.14445/22312803/IJCTT-V74I2P104

[29] Sunyaev, A., Benlian, A., Pfeiffer, J., Jussupow, E., Thiebes, S., Maedche, A., & Gawlitza, J. (2025). High-risk artificial intelligence. *Business & Information Systems Engineering*, *67*(6), 981–994. https://doi.org/10.1007/s12599-025-00942-6

[30] Vice, J. R., & Khan, M. (2022). Toward accountable and explainable artificial intelligence part two: The framework implementation. *IEEE Access*, *10*, 36091–36105. https://doi.org/10.1109/ACCESS.2022.3163523

[31] Bugshan, N., Khalil, I., Rahman, M. S., Atiquzzaman, M., Yi, X., & Badsha, S. (2023). Toward trustworthy and privacy-preserving federated deep learning service framework for industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, *19*(2), 1535–1547. https://doi.org/10.1109/TII.2022.3209200

[32] Ekechi, C. C., Popoola, E. T., Ademoye, A. A., Idowu, M. E., Saliu, A. S., Osayuki, L. A., & Abbah, P. I. (2025). Trust-aware database intelligence (TADI): Embedding explainable AI into DB management decisions with resilient dataflow intelligence. *Global Journal of Engineering and Technology Advances*, *25*(1), 247–268. https://doi.org/10.30574/gjeta.2025.25.1.0312

[33] Wang, L., Hu, M., Yang, L. T., Sun, X., & Chen, Z. (2025). AI-auditor: A data auditing framework for enhancing the trustworthiness of AI models. *IEEE Transactions on Industrial Informatics*, *21*(12), 9208–9216. https://doi.org/10.1109/tii.2025.3593981

[34] Raza, S., Sapkota, R., Karkee, M., & Emmanouilidis, C. (2026). TRiSM for agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems. *AI Open*, *7*, 71–95. https://doi.org/10.1016/j.aiopen.2026.02.006

[35] Jois, A. H. S., Desai, H. S., Thrupthi, S., Kallapur, V. A., & Kumar, A. (2026). Cyberguard AI: Smart cybersecurity & real-time blockchain-based network monitoring. *International Journal of Scientific Research in Engineering and Management*, *10*(1), 1–4. https://doi.org/10.55041/IJSREM55778

[36] Chakraborty, S. (2025). Data stewardship co-pilot: Transforming enterprise data governance with generative AI and agentic frameworks. *European Journal of Computer Science and Information Technology*, *13*(22), 1–14. https://doi.org/10.37745/ejcsit.2013/vol13n22114

[37] Sun, C., Ning, M., Deng, Z., & Khajepour, A. (2024). REAL-SAP: Real-time evidence aware liable safety assessment for perception in autonomous driving. *IEEE Transactions on Vehicular Technology*, *73*(8), 10870–10883. https://doi.org/10.1109/tvt.2024.3369100

How to Cite: Sahagal, P. P., Talluri, S., Sambamurthy, P. K., Jangid, H., & Syed, A. A. (2026). Agentic Data Engineering Framework for Autonomous, Real-Time, and Trustworthy AI Systems. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS620210014