Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

Self-Optimizing Data Pipelines Using Machine Learning for Cloud Workloads

Velangani Divya Vardhan Kumar Bandi
Director AI/ML Engineering
divyavardhanbandi@gmail.com,
ORCID ID: 0009-0008-7949-5670

ARTICLE INFO

Received: 01 Nov 2025
Revised: 15 Dec 2025
Accepted: 25 Dec 2025

ABSTRACT

Cloud Data Pipelines enable enterprises to readily ingest, process, clean, and store large amounts of structured and unstructured data in cloud environments to drive analytics, business intelligence, and data-science workloads. However, designing and implementing such pipelines is non-trivial and challenging. Pipelines should be optimized for cost, speed, or any combination of the two but these objectives are at odds with each other. A data pipeline architecture that enables easy prototyping of data ingestion and transformation processes within any cloud platform is presented. Machine Learning (ML) is employed to inform scheduling and resource allocation decisions in order to reduce operational cost while ensuring acceptable latencies.The objectives of optimizing Ingest, Transformation, and Enhanced ETL cloud data pipelines in real-time for cost and latency are accomplished. The four cloud providers—Google, Amazon, Microsoft, and IBM—are supported, with data volumes ranging from a few megabytes to generating several gigabytes. Latency from minutes to hours can be supported without breaking the bank. ML models inform autoscaling groups, transformation resources, and scheduling. Cross-cloud portability through modular code-based connection-management further optimizes the development phases while improving code quality.

Keywords: Cloud Data Pipelines, Real-Time ETL Optimization, ML-Driven Resource Scheduling, Cost–Latency Trade-off, Cross-Cloud Data Architecture, Intelligent Autoscaling, Data Ingestion Frameworks, Transformation Pipelines, Enhanced ETL, Cloud Portability, Modular Connection Management, Multi-Cloud Support (AWS, GCP, Azure, IBM), ML-Based Workload Scheduling, Resource Allocation Optimization, Data Engineering Automation, Operational Cost Reduction, Low-Latency Analytics Pipelines, Structured and Unstructured Data Processing, Cloud-Native Data Platforms, Production-Grade Data Pipelines.

1. Introduction

The costs and complexities of executing cloud workload make optimizing resource management a priority. Classical approaches utilize careful capacity provisioning through autoscaling, but controlling Quality of Service (QoS) in such systems often relies on best efforts. QoS violations can introduce unexpected service downtimes, degrade user experiences, and decrease revenue. The portable, cloud-agnostic design of cloud data pipelines gives automated Machine Learning (ML) modellers a high-level view of the various influencing factors, offering data-driven solutions to resource allocation and scheduling problems. Tuning third-party services instead of application-managed services allows the use of less-knowledgeable yet easier-to-understand, can-be-executed-just-once ML models. Cloud data pipelines for batch workloads support scheduling decisions between service execution and aksing an ML model to supply the execution decision. Public clouds provide data from many different users, which can be collected, labelled, and used for training, testing, or validating a wide range of ML models used to support many aspects of data pipeline operation.

The presented work studies self-optimizing cloud data pipelines using externalized ML models to support many aspects of QoS management. It defines the modeling and experimentation process, detailing the introduction of ML models for decision-making in resource allocation, scheduling, and scaling. The experiments involve data transfer or storage in multiple clouds, examples of data ingestion and data processing with CNNs or Web servers from a third-party service in multiple clouds. The cloud-in-cloud approach supports the Tests As A Service concept. The current scope focuses on ML models for scheduling and resource allocations, pursuing the creation and usage of a diverse set of models. Research work in the past few years and expected improvements by the emergence of new areas in the ML field drive model creation.

1.1. Overview of the Study's Objectives and Scope

In-cloud optimization of data pipelines for data integration and transformation has several objectives: minimizing cost without affecting latency; minimizing latency without degrading performance; and achieving high throughput at a minimal cost. These goals are pursued via machine learning models capable of predicting, for a given data workload, the best schedule that incurs the least cost for a required latency, or that achieves the best quality metric (cost or latency) for a given throughput demand. The research focuses on integration and transformation processes deployed on cloud data pipelines and hosted on the Google Cloud Platform. Other services (e.g. data storage, caching, and messaging) are assumed to be provided by the cloud vendor management, and workloads are executed on large-scale cloud resources.

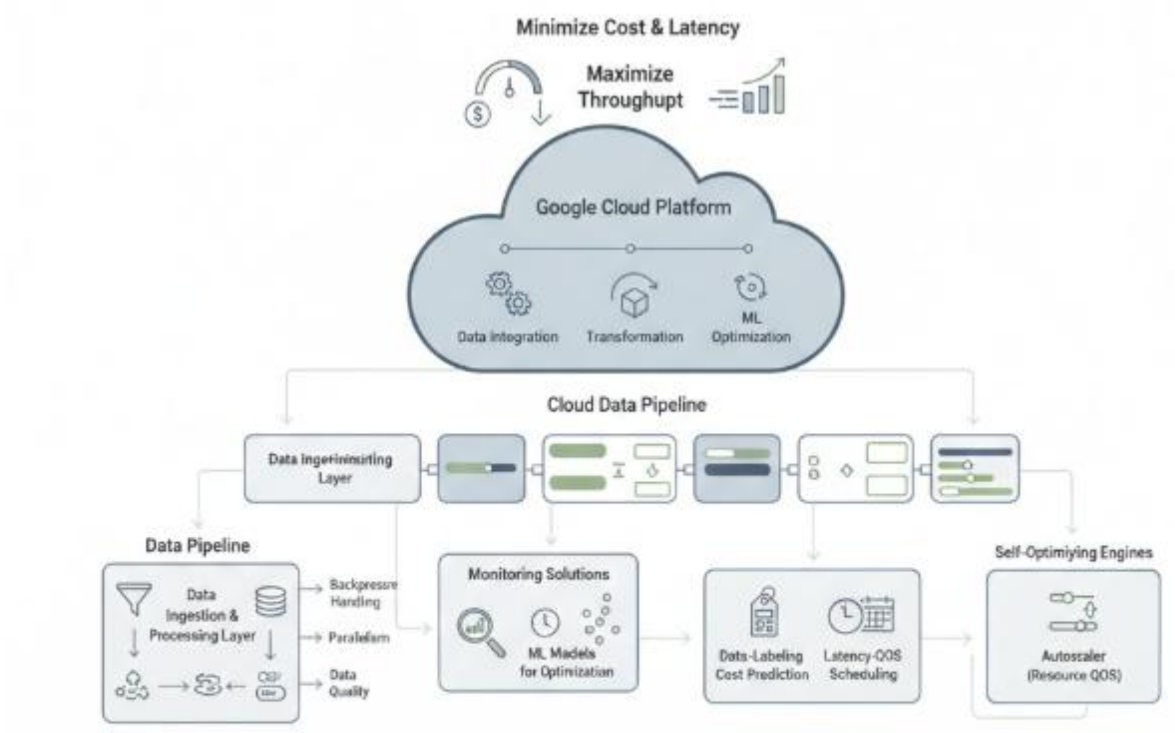

Fig 1: Autonomous Cloud Data Pipelines: A Machine Learning Framework for Multi-Objective Cost, Latency, and Throughput Optimization

Previous studies provided a modular architecture for data pipelines that run on any cloud platform. The architecture is implemented and the data ingestion and processing layer is described; it supports cloud-native streaming and batch services, handling backpressure, parallelism, and data quality. Monitoring solutions for the pipeline control plane collected data for training ML models that optimize data integration and transformation processes. Three kinds of ML-driven self-optimizing engines were designed and built. The first predicts how data-labeling requirements influence the overall pipeline cost; the second adjusts the scheduling of labels to minimize cost subject to fixed-latency QoS; and the third is a full-fledged autoscaler that controls pipeline resources based on user-defined QoS.

2. Background and Related Work

Data ingestion, integration, and transformation pipelines are a vital part of cloud workloads. They move data from multiple sources, such as web servers, application logs, third-party APIs, and monitoring systems, to storage solutions or data lakes designed for later analysis. Pipelines are often formed as a bouquet of cloud services offered by the main providers; these services cover data extraction from sources, transformation into the final format, and storage for further analysis. These flows consume resources and incur costs that can be tens to hundreds of dollars a day.

Thus, managing data flows should ideally be considered from a datacenter-wide perspective, in conjunction with other components such as batch jobs and machine learning training. However, these flows seldom get the attention they deserve. They are often coded as sets of ad-hoc scripts that get the job done for a specific need but without the guarantees of quality, maintainability, or reliability that one would expect from a production-grade system.

Existing systems take different approaches to optimizing data pipelines. Some of these systems leverage ML techniques to automatically make decisions, such as allocating resources or selecting the instance types of virtual machines that will run the data pipeline's components. Such optimizations may at times bring tangible gains, but they are not a panacea, since they focus on a single aspect of the flow and overlook others. A data pipeline can still benefit from automation and optimization even without the utilization of ML techniques—indeed, without ML techniques at all. The present work considers the entire data pipeline system, describing its architecture and implementation, and proposing key aspects of its operation that can be automated and optimized through ML techniques.

2.1. Related Research and Prior Art

Most existing data pipelines that consume cloud services focus the cost of data transfer and buffer storage while performing a simple best-fit scheduling of preprocessing tasks to minimize the use of cloud resources. A cost function is reduced by finding optimal transfer schedule to minimize the cost of data retrieval and data transfer. A smart queue service leverages the parallelism of cloud by redirecting the data stream from data producers to multiple data consumers based on the priority order and processing time. CloudCrane integrates multiple services of cloud infrastructure as a data segmentation framework which segments a data into several pieces and store into Cloud servers by retrieving the relevant credentials. Workload-aware scheduling considered multiple resource constraints such as time, budget, and data locality.

A framework called mCloud optimizes the job scheduling of multiple concurrent queries for data reservoir services provided by cloud service providers. With the considerations of monetary cost and Quality of Service (QoS) in job execution time, it assure that the fast response time with lower monetary cost. An end-to-end cloud-based framework is proposed to subsume the cost of data staging in pipeline pattern processing with reliable network. WhizCloud is a dynamic resource provisioning system for heterogeneous batch applications on clouds with dedicated resources. Instead of scheduling the resources for multiple users, it minimizes resource consumption for a single user who enforces batch jobs with very small temporal locality by utilizing the dedicated resources. Pipeline scheduling considered cost, delay and dependencies of cloud services by processing a specified request which includes known and unknown cloud services.

Equation 1) Pipeline performance model (latency, throughput, cost)

1.1 Notation (maps to the paper's concepts)

- Stages: $i \in \{1, \dots, N\}$
- Provisioned resources for stage i: r_i (e.g., number of workers/instances/slots)
- Arrival rate (workload intensity): λ (jobs/sec) — corresponds to workload pattern features
- Service rate per resource unit at stage i: μ_i (jobs/sec per unit)
- Total service capacity at stage i:

  $\mu_i^{\text{tot}}(r_i) = r_i \mu_i$

1.2 Queueing-based latency per stage (step-by-step)

A common approximation for a saturated processing stage is an M/M/1 queue (or M/M/m; we keep it simple and interpretable).

Step 1 (stability condition):

For stage i to not explode in backlog,

$$\lambda < \mu_i^{\text{tot}}(r_i) = r_i \mu_i$$

Step 2 (utilization):

$$\rho_i = \frac{\lambda}{r_i \mu_i}$$

Step 3 (mean time in system for M/M/1):

$$W_i = \frac{1}{\mu_i^{\text{tot}}(r_i) - \lambda} = \frac{1}{r_i \mu_i - \lambda}$$

Step 4 (include fixed overheads):

Real pipelines also have serialization, IO, orchestration, etc. Let that be d_i:

$$L_i(r_i) = W_i + d_i = \frac{1}{r_i \mu_i - \lambda} + d_i$$

Step 5 (end-to-end latency):

Summing stage latencies:

$$L_{\text{e2e}}(\mathbf{r}) = \sum_{i=1}^{N} L_i\,(r_i) = \sum_{i=1}^{N} \left(\frac{1}{r_i \mu_i - \lambda} + d_i \right)$$

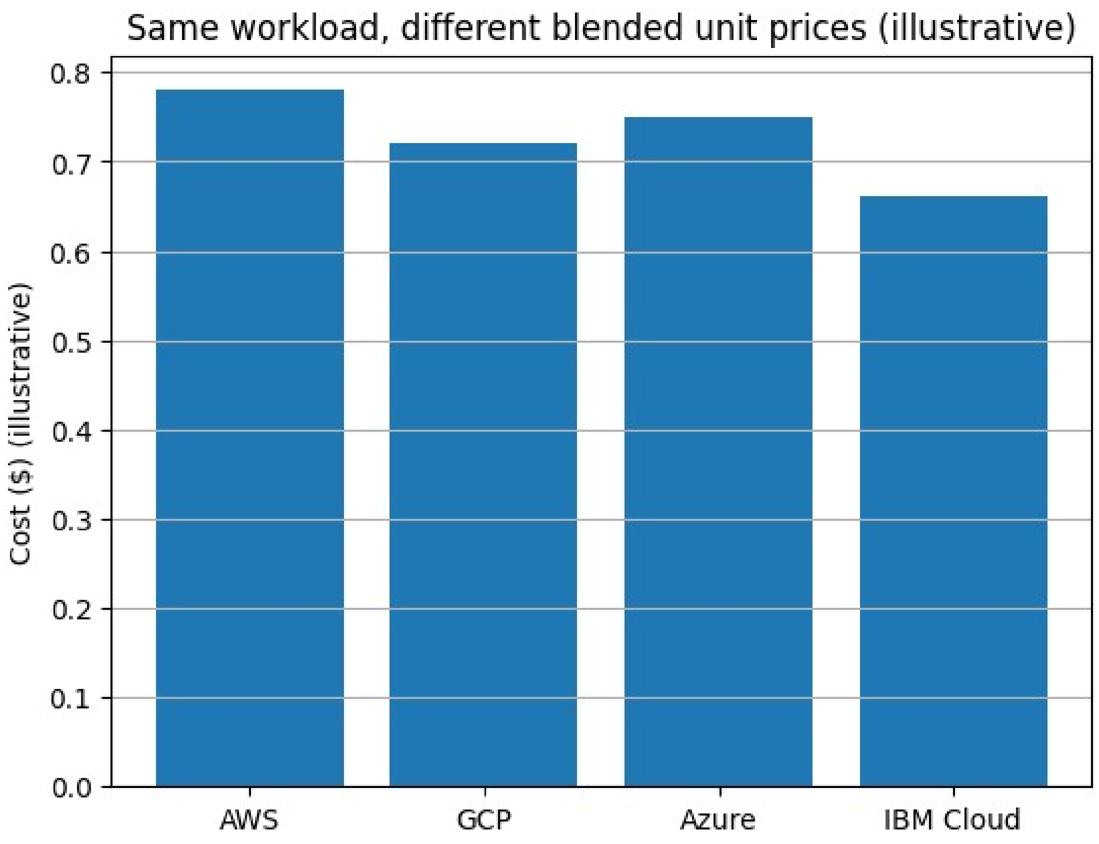

Fig 2: Queueing-Theoretic Performance Model for Stage-Wise Latency, Throughput, and Resource Utilization in Data Pipelines

3. Problem Formulation

An architect builds a house with wood. Later on, a carpenter fills the house with wood. This modularity allows specialization and portability to many regions with little effort. Data pipelines can be built upon similar modularity. Unit transactions must integrate, transform, store, and deliver data. The same pipeline is also usually used for a data warehouse load so scaling can be more efficient. Data always enters a data pipeline at the same point, but once it starts going through the house, it can take many paths. It may go through a streaming path or a batch path or some other process flow such as an ETL or ELT process for a data warehouse. During high-volume periods, backpressure can occur in the system, and the batch processing part of the system may have to run a few loads with a higher level of data latency to absorb the load. This part of the system will want to make sure it has available parallelism to assist with the necessary data loads. Data pipelines rarely have any events triggered or controled by the actual arrival of the data. A triggered event by the data arrival may push a follow-up alert or a warning into an event-driven system. A service mesh will be able to handle more fine-grained communication between all the microservice containers, but it has a heavier infrastructure load. Such setups may want to talk to the integration side during off-peak hours so the data handling can be done more seamlessly.

Cloud Engineering permits for the much-needed auto-scaling for microservice containers over the Ingress and Event Delivery Track. Reliability comes more from reliability riding over three edges or other toppinging patterns. But by becoming visible and computable, observability can also support emergent logic as the monitoring systems draw attention to latent patterns. Enhancing the ties among

Schwartzian Analysis, SEO, SCHEMA, META-MODEL, and MASTER_MIND_TT is another future facet. All these other facets are generic add-ons to the architecture, allowing for different pipelining patterns while still allowing for the original situation where the domain-expert trained Data Quality Rules are the data pipeline flow controller.

Table 1: Impact of Resource Provisioning on Pipeline Latency and Operational Cost

Resource units (r)	Latency (sec) (illustrative)	Cost ($) (illustrative)
1	128.0	0.12
2	68.0	0.24
3	48.0	0.36
4	38.0	0.48
5	32.0	0.6
6	28.0	0.72
7	25.14	0.84

3.1. Key Concepts in Cloud Data Pipeline Architecture

Data pipeline architecture in cloud computing environments consists of multiple integrated layers, each responsible for a specific functionality. The data ingress layer contains all the components and strategies needed to ingest any type of data from heterogeneous sources into the cloud provider. The data streaming layer enables the platform to react immediately to any event. The data storage layer offers cost-efficient storage solutions and services adapted to the data working set size. The data batch-processing layer executes the ELT or ETL process for analytics workloads. Event-driven components allow processing on specific incoming data events without being triggered by user queries. Service meshes enforce and implement observability, reliability, and fault-tolerance primitives and rules for microservices deployed in the pipeline.

The modularity of the architecture design promotes isolation in the development cycle while supporting portability and compatibility with the services from different cloud providers. Each module can be implemented over the set of services offered by the cloud provider the organization has an agreement with. A GCP organization can use Pub/Sub and Dataflow for streaming data, whereas an AWS organization can use Kinesis and Lambda. For simpler data-pipeline use cases that do not need complex data processing, connector components to third-party providers, such as Fivetran and Striim, can be employed. Data-pipeline users are only required to satisfy the properties imposed by their choice of the modules.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

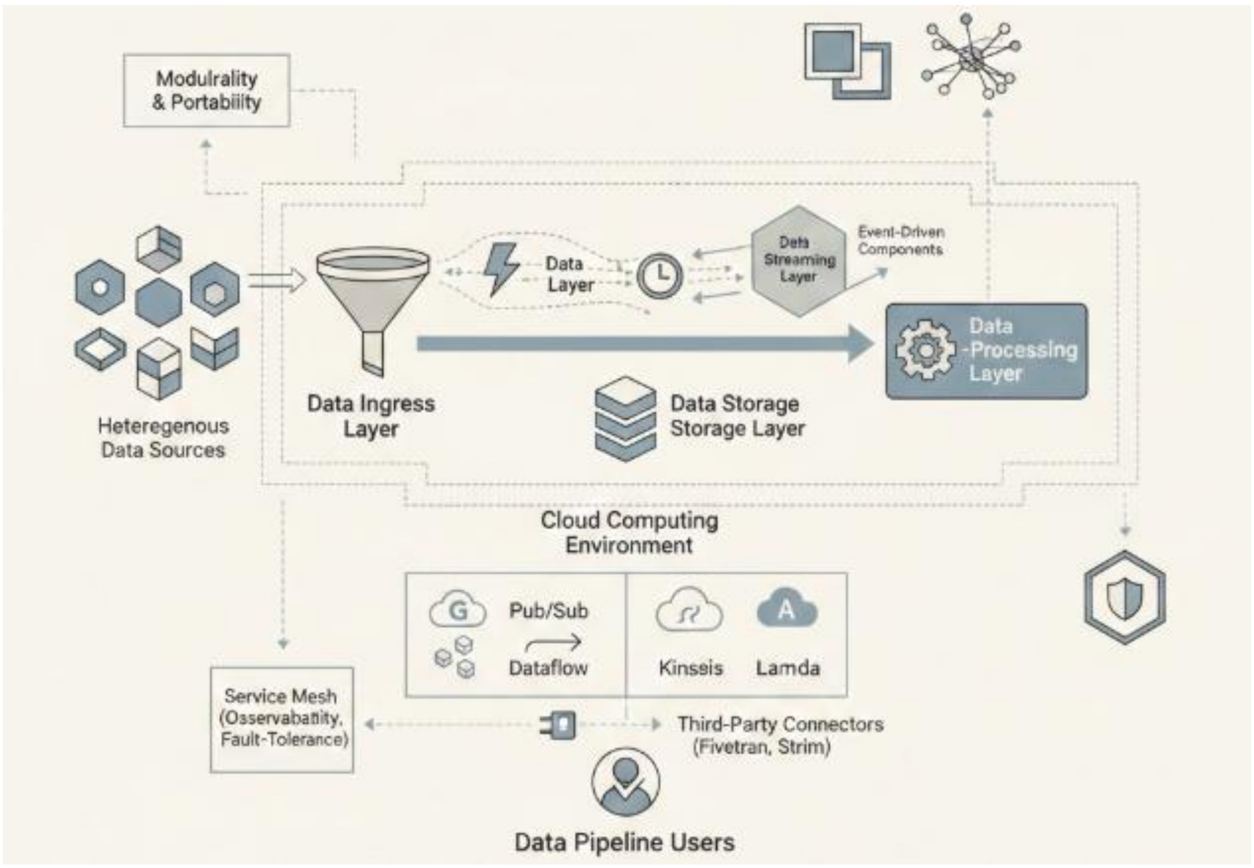
Modulrality & Portabiilty
Data Layer
Deta Streaming Layer
Event-Driven Components
Data -Processing Layer
Heteregenous Data Sources
Data Ingress Layer
Data Storage Storage Layer
Cloud Computing Environment
Pub/Sub
Dataflow
Kinseis
Lamda
Service Mesh (Osservabatiity, Fault-Tolerance)
Third-Party Connectors (Fivetran, Strim)
Data Pipeline Users

Fig 3: Poly-Cloud Architecture: A Modular Framework for Agnostic and Event-Driven Data Orchestration

4. Data Pipeline Architecture for Cloud Environments

The reference architecture of a data pipeline suitable for cloud deployment differentiates between four logical but closely interleaved layers: data ingress, processing, application, and control. Cloud systems are inherently multitenant, hence reliable and efficient separation of data is crucial. Each data-processing stage should flexibly span multiple cloud services, as the choice of service for a particular task may vary depending on the nature of the workload and the available cloud service at a given time. At the lowest layer, the control plane injects information, policies, and control commands to the data-pipeline system, allowing it to optimally manage the services comprising the architecture and their interactions.

Such an architecture can serve as a skeleton for building cloud data pipelines: each logical layer can be instantiated using cloud services already in place, with the support of an orchestration system. It holds specific patterns for modularizing new data integration-and-transformation processes that can be made available as additional cloud services. Many additional capabilities are already managed natively by the cloud vendors but may require a particular design and implementation. Three of these capabilities are data quality, data security, and monitoring. A cloud data pipeline built according to this architecture can flexibly span multiple cloud-services providers whenever required and can be deployed and operated according to the principles of chaos engineering and data-driven SRE.

4.1. Data Integration and Transformation Processes

Data connectors enable efficient interaction with external data sources, while schema management systems support the auto-discovery of input and output data structures. Data quality rules detect anomalies in data generated by unreliable sources, whereas transformation pipelines enforce specific business rules on the data prior to utilization. Since data pipelines often rely on data from external partners, idempotency is an essential attribute for all pipeline components. By accumulating knowledge of external service responses, data pipelines can tolerate failures from those services without iteration.

Data pipelines sourced from external services should also exhibit fault tolerance. For example, change-data-capture components consuming Kafka streams from unreliable partners may skip events referencing unstable partitions or registers with inconsistent states. To divert initialization failures into an alternative control stream and attempt a new initialization attempt after N successful cycles in the main information topology, the originator must also keep track of the state from the last successful partition read. The usage of a service mesh transparently solves many of these issues and brings added observability and reliability primitives embedded into the architecture.

Equation 2) Cost model (what you optimize against latency)

2.1 Cost per stage

Let:

- p_i = price rate ($ per resource-unit per second) for stage i
- T = time window (sec) you keep resources allocated

Then stage cost:

$$C_i(r_i) = p_i\, r_i\, T$$

Total cost:

$$C_{\text{tot}}(\mathbf{r}) = \sum_{i=1}^{N} p_i\, r_i T$$

4.2. Data Storage Solutions for Cloud Systems

Architectural variety offers a rich palette of data storage options, shaped by factors such as scale, dataset structure and access modes. The classic dichotomy of file systems (e.g., NTFS, HDFS) vs. databases (relational or NoSQL) becomes an oversimplification in the cloud. Object storage provides a S3-like interface to retrieve and store virtually unlimited volumes of unstructured data with low latency and reasonable economic costs; however, its lack of indexing and partitioning capabilities makes it unsuitable for large-scale ad-hoc analysis. Columnar storage (e.g., Parquet, ORC) overcomes these limitations and fuels parallel data warehouses, such as Redshift, BigQuery, and Synapse, in which many concurrent queries can each quickly scan a large dataset. Different consistency models fit various use cases and access patterns: read-after-write consistency for the latest data (e.g., social-media timelines); eventual consistency for highly-scalable read-heavy workloads (NoSQL systems); and configurable consistency between the two extremes (e.g., Amazon S3). Replication across datacenters supports disaster recovery and ultra-low-latency content delivery, while also exposing a cost-performance trade-off: latency and read throughput advantages for read-heavy workloads against increased expense for write-heavy loading patterns—read replicas can be stale, allowing lower storage costs per replica.

Layered storage provides a logical alternative: only the most recent copy is kept in hot S3 storage, while historical data with less frequent access is archived in warmer (e.g., S3-IA) and frozen (e.g., Glacier) storage tiers. Increasing the number of available layers enables a smoother cost-performance trade-off; however, a sparse dataset with only a few layers might not yield optimal cost savings with the same granularity.

Continued growth of fully-managed cloud data warehousing services like Amazon Redshift and Google BigQuery blurs the lines between online analytical processing (OLAP) systems and online transaction processing (OLTP) databases. Serverless architectures decouple resource provisioning from consumption, dynamically allocating resources to make ad-hoc queries responsive when using an analytic data warehouse. Reinforcement learning can be invoked to choose an appropriate cold storage layer throughout the pipeline, and the schedule can be combined with the number of layers to determine costs for data load-and-query use cases.

5. Machine Learning for Optimization

Machine learning (ML) techniques assist in determining configurations that result in suboptimal performance of data pipelines, such as tuning parameters, selecting an adequate pattern for scheduling component execution, and appropriate resources for data connector components. Training pipelines require collecting heterogeneous data from all layers of the architecture and modeling all processes controlled by ML algorithms or engines. Continuous monitoring and validation are essential to ensure model accuracy, and input data need to be periodically validated and relabeled before being retrained.

Features reflect the current workload pattern, the quality seals applied in upstream processes, resource consumption levels collected from monitoring solutions, data locality across cloud regions and availability zones, or historical information on the interaction of these characteristics with the QoS defined to execute specific data pipeline flows (for instance, latency, throughput, and adherence to service level agreement rules). Data collected from running the declared pipelines, such as resource utilization, progress of ongoing executions, and workload completion time, feed the subsystems in charge of making predictions or selecting the execution plans of the data flows. Based on these features, different ML models—such as reinforcement learning (RL), Markov decision processes, or supervised predictors—can be implemented to address the integration and processing of data pipelines, supporting predictive execution planning and responsive resource allocation. Model accuracy is determined by the modeling purpose. Predictors are trained using historical information to classify the outcome argument, while RL and Markov decision processes are trained through interactions with the environment.

Table 2: Comparative Operational Cost of Pipeline Execution Across Major Cloud Providers

Cloud	Cost ($) (illustrative)
AWS	0.78
GCP	0.72
Azure	0.75
IBM Cloud	0.66

5.1. Feature Engineering and Data Collection

Feature selection and customization are key elements of any supervised ML task. Although previous work on self-optimizing ML models has focused exclusively on using resource consumption and environment features (e.g., workload type and resource availability) to predict optimal values of parameters for client-driven services, a much broader set of features can be engineered to produce accurate predictions for pipeline scheduling and resource allocation. Such features can be grouped into the following four categories.

First, workload pattern features—such as incoming traffic rates, temporal traffic concentration, bursting patterns, and user/device locality—indicate how load will change over time and the balancing requirements across machines. The second group of features describes resource consumption or utilization for executors and data stores (long-term, short-term, or current values): CPU, memory, network, and disk throughput. The third group collects resource location information: the distance or transfer delay between clients and data stores or between two different data stores. Finally, quality-of-service (QoS) requirement features indicate the minimum acceptable latency/throughput values for a given job. Their unfulfillment can generate cost penalties or loss of service-level agreements.

Constructing a dataset for the model training, validation, and deployment phases requires using a tracer tool to record the model features, labeling, and resource usage when executing pipelines built as data flows, including data processing with Stream Processing Engines (SPEs) and Data Flow Engines (DFEs). The data from these pipelines can then be used to build, validate, and deploy ML models that predict the runtime cost (in monetary units or time) of pipeline components and resource schedules needed to satisfy QoS requirements. As an additional option, the model can also be trained to predict the optimal values for parameters selected beforehand.

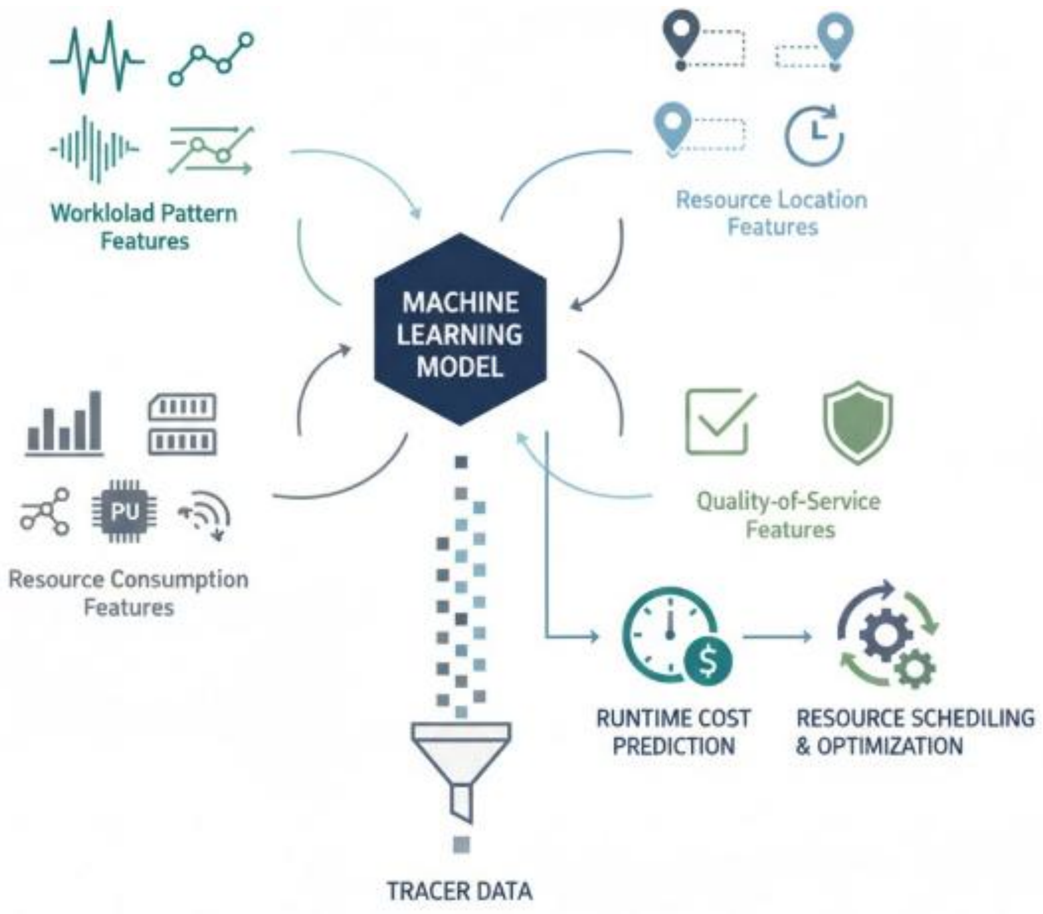
Worklolad Pattern Features
Resource Location Features
MACHINE LEARNING MODEL
Resource Consumption Features
PU
Quality-of-Service Features
RUNTIME COST PREDICTION
RESOURCE SCHEDILING & OPTIMIZATION
TRACER DATA

Fig 4: A Multi-Dimensional Feature Engineering Framework for Holistic Resource Allocation and QoS-Aware Scheduling in Distributed Data-Flow Engines

5.2. Models for Resource Allocation and Scheduling

A range of ML approaches, notably in computer vision, reinforcement learning, and time-series forecasting, present themselves as candidates for optimizing pipeline performance. Two categories stand out. SP algorithms presume the optimal solution for all distinct combinations of input constraints resides in an external repository. For active RL and Markov decision processes, the decision-making intelligence can be internal to the data pipeline; learning proceeds via repeated interaction with the environment. The training is accomplished offline when a high-fidelity multicloud simulator of the proposed framework is available. Supervised predictors appear particularly suited for task scheduling and function placement in a data pipeline because of the multiprocessing architecture and large volumes of training data that will be generated. Training relies on workload characterization enriched with QoS requirements and resource utilization patterns and levels, complemented by resource budgets, deployment and scheduling latencies, cloud-provider specifications, resource allocation policies, and service-level agreements (SLAs). Health monitoring of the service mesh coupled with system-layer observability data will yield training and validation labels. The chosen modeling approach will thus depend on the level of feature engineering and available training and validation data.

ML also supports dynamic and adaptive optimization, balancing performance as effectively as possible while operating under continuous or transient changes in conditions; it does so by steering the scheduling and resource allocation functions toward one or more of the following goals: lowering execution latency (or latency for a batch of requests); maximizing throughput over time within an SLA (considering resource budgets); maximizing the number of servable request paths over time (or servable paths at a given instant) within SLAs and resource budgets; minimizing the end-to-end cost of the pipeline; minimizing the execution cost of the pipeline (equivalent to cost per request when operating at maximum throughput). Data-driven and AI-based techniques (including system synthesis and ML) address scheduling such that performance is improved or path latency reduced without triggering alert conditions.

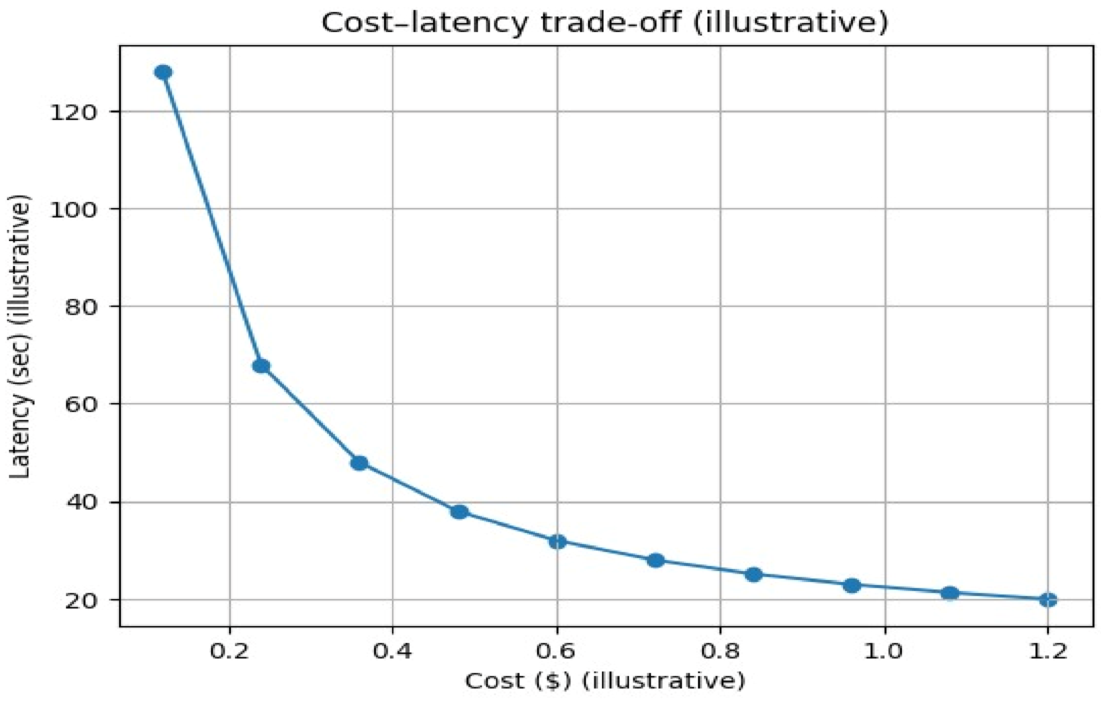

Fig 5: ML-Based Closed-Loop Optimization Architecture for Adaptive Scheduling and Resource Control

6. System Design and Implementation

The supervised prediction model utilizes randomly generated workloads covering the decision space as training data, while a reinforcement learning approach learns resource-logical patterns through interaction with the pipeline. Supervised prediction models govern batch operations and the assessment of data quality rules, and predict the number of connectors required for each phase; other batch functions must also be weighted by the loading–unloading data time. Resource localization is labeled by the administrator, and the pipeline's control plane enriches its description with exam plans for ML embeds. The reinforcement learning agent controls the parallelism level for thermal memory-free in-memory processes, and also governs the local approach to data stash and un-stash components located on nearby nodes. The deployment phase is designed to be as simple as possible.

Although the system presented in this study is limited to the scheduling of data pipelines, all processes of the described architecture and the declared end-to-end process can benefit from auto-tuning of input parameters. Models explore the usage of data locality, the resource trajectory, and the time to finish the process against the operations being executed at all process locations through direct interactions. Such models are particularly suited for reinforcement learning. Patterns arising from the combination of pipeline components, detected by a model considering all system logs, can also compose a Markov decision process that tracks the use of resources on the system graph. Finally, the QoS requirements expressed in declarative schemas can be 0-1 labels that predict the local and global backpressure in order to generate a near-optimum routing and an optimum level of resource parallelism.

Equation 3) Core optimization problems (cost–latency trade-off)

3.1 Constrained optimization: minimize cost subject to latency QoS

$$\min_{\mathbf{r}} \ C_{\text{tot}}(\mathbf{r}) \quad \text{s.t.} \quad L_{\text{e2e}}(\mathbf{r}) \le L_{\max}, \quad r_i \in \mathbb{Z}_{\ge 1}$$

3.2 Lagrangian (step-by-step)

Step 1 (write Lagrangian):

$$\mathcal{L}(\mathbf{r},\eta) = \sum_{i=1}^{N} p_i\, r_i T \ + \ \eta\left(\sum_{i=1}^{N}\left(\frac{1}{r_i\mu_i - \lambda} + d_i\right) - L_{\max}\right)$$

where $\eta \ge 0$ is the Lagrange multiplier.

Step 2 (differentiate w.r.t. each r_i, continuous relaxation):

$$\frac{\partial \mathcal{L}}{\partial r_i} = p_i T \ + \ \eta \cdot \frac{\partial}{\partial r_i}\left(\frac{1}{r_i\mu_i - \lambda}\right)$$

Now,

$$\frac{\partial}{\partial r_i}\left(\frac{1}{r_i\mu_i - \lambda}\right) = -\frac{\mu_i}{(r_i\mu_i - \lambda)^2}$$

So:

$$\frac{\partial \mathcal{L}}{\partial r_i} = p_i T \ - \ \eta\frac{\mu_i}{(r_i\mu_i - \lambda)^2}$$

Step 3 (set to zero at optimum interior point):

$$p_i T = \eta\frac{\mu_i}{(r_i\mu_i - \lambda)^2}$$

Step 4 (solve for r_i):

$(r_i\mu_i - \lambda)^2 = \eta\frac{\mu_i}{p_i T}\, r_i\mu_i - \lambda = \sqrt{\eta\frac{\mu_i}{p_i T}}\, r_i = \frac{\lambda}{\mu_i} + \frac{1}{\mu_i}\sqrt{\eta\frac{\mu_i}{p_i T}}$

6.1. Data Ingestion and Processing Layer

Pipelines manifest in either streaming or batch paths. Streaming flows transfer data continuously, incurring negligible delay before reaching consumers. Streaming ingestors—e.g. Apache Kafka Connect and Amazon Kinesis Data Firehose—serve data as soon as it's available. Sources must apply backpressure, preventing data production from exceeding transfer capacity. Latency budget dictates whether shortcuts may be taken at the expense of strong consistency; for replica pools, strong read-after-write consistency requires all writes to read from a single writer, impacting availability. The throughput-per-query bottleneck for a service mesh executing data enrichment may govern parallelism setup, promoted—or demoted—relative to throughput provided more direct paths connecting sources to consumers.

Batch jobs read data from file systems; data-in products like Amazon S3, backed by CDNs or caches, offer data-serving latency at the same order of magnitude as data-transfer latency. Natural groupings of clients with similar locality requirements allow for a drop in streaming parallelism in favour of batched requests. The batch-oriented nature of data lakes allows temporary data-discrepancy models to linearly reduce storage costs and serve data-read requests with different levels of data quality; rules of thumb for creating dedicated datasets in a layered storage architecture, plus a set of cost-performance benchmarks, improve cost optimization exponentially. A cloud-agnostic streaming engine like Apache Flink, supporting both streaming and batch modes, would facilitate the definition of autoscaling policy rules that account for backpressure, support the dynamic addition of new data sinks, and enable datacentre-best-positioned operations—e.g. ETL/ELT data-moulding operations requiring interaction with data consumers—to run close to clients-minimizing costs on data access.

6.2. Control Plane and Orchestration

The control plane is responsible for declarative descriptions of the data pipeline, such as schemas, data flow policies, and ML model metadata. Policy engines analyze these abstractions and trigger adaptation and resource provisioning operations. A dedicated set of rules evaluates resource, connectivity, and QoS requirements triggered by the pipeline definition; the results feed the autoscaling unit for proper deployment of the connectors and transformations. Data quality rules, such as completeness or timeliness, are instantiated in the data storage system, often as SQL-based checks in the defined monitoring environment. The policies implementing over- or under-provisioning of resources rely on a reaction matrix based on workload patterns detected over time. Failure and incident events caught by the observability layer drive archival procedures. Finally, the CI/CD scheme for ML components enables robustness of the data similarity-based data-locality and scheduling predictors.

For produced data in a streaming manner, a backpressure management module controls the source speed while preventing data loss in sinks. The streaming parallelism strategy relies on a user-defined factor for the connector and a per-node inbound throughput estimation multiplied by the pipelines at each level. The ability to dynamically choose the processing engine is a crucial feature of the architecture, allowing performance tuning and cost reduction. In the polling procedure, both the latency and the expected execution consumption determine whether the batch engine is engaged. For installations relying on cloud-service connectors, the selection depends on the connector's price. Finally, the choice of streaming engines follows three criteria: the ability to handle exactly-once processing, the supported processing patterns and functions, and the integration support for the CICD platform.

7. Conclusion

Data pipelines within cloud deployments can automatically adapt to workload evolution and resource availability by employing ML on relevant historical artifacts. The end-to-end architecture supporting this concept is modular, ensuring compatibility with all major cloud providers. Key components are either observability extensions to existing cloud services or implementations of well-established design patterns. Automatic behavior tuning relies on reinforcement learning, while predictive ML techniques support high-level scheduling decisions. Empirical evidence from a representative cloud provider points to an effective realization of these optimizations.

The objectives stated above are met, and important practical steps for real-world deployment and integration of the proposed ideas into a production-grade system are explored. Results from a streaming ETL application reveal a limited degree of optimization in a single cloud environment and suggest that a unified solution across heterogeneous clouds remains an unmapped territory. Gaps in the experimental evidence thus lead to a clear set of directions for future work. Generalization and validation of the mechanisms in an unsupervised manner are necessary, as is the inclusion of user relations in the learning processes. Specialized strategies to address data residency and privacy concerns along with emergent ML-based approaches for model self-optimization represent promising avenues for the adaptive optimization of data pipelines.

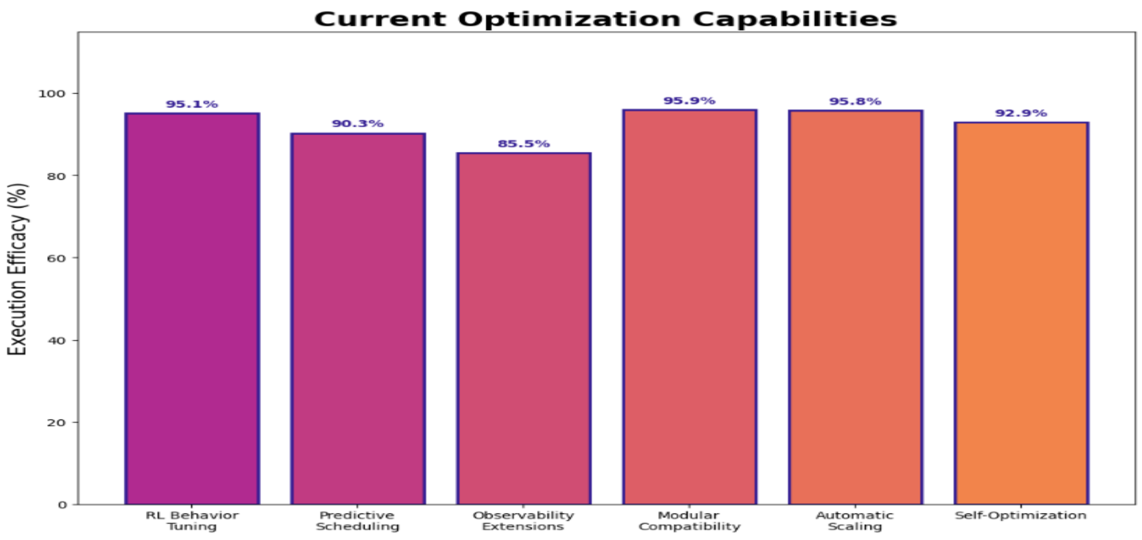

Fig 6: Current Optimization Capabilities

7.1. Future Work and Research Directions

Model generalization and transfer, cross-cloud portability, privacy protection, and the application of ongoing advancements in ML to support self-optimizing strategies represent particularly colourful yet less-explored directions. While CustardNet achieves state-of-the-art performance for the Azure data pipeline in terms of training time, pipeline cost, QoS latency, and QoS throughput, the audiovisual dataset is urban and, thus, its workload characteristics manifest in forested geographical areas as seen by the Kaliningrad cluster. Hence, CustardNet is trained solely in Azure, a cloud provider environment having non-point data locality and its visual workload. Using it on a diurnal distributed workload for urban stealthy detection on the German cluster is merely hoped to yield acceptable cost and QoS predictability without retraining, which has not yet been possible. Yet, CustardTransfer, a transfer-learning-based approach for cross-cloud data pipeline workload optimization, and the Poor-Transfer scenario—where the model pretrained on the source cloud provider environment achieves worse performance than another model pretrained on the target cloud provider environment and is nonetheless applied to mitigate the absence of training on the target cloud provider environment—seem particularly encouraging.

Experience with CustardNet in other cloud provider environments highlights risks in data privacy, especially for image classification. Reinforcement learning explores non-gaming environments. So-called emergent behaviours have developed from players' interactions with learning agents in DotA 2 and StarCraft II. Such emergent behaviours have been simulated during training but have never been achieved naturally like in CustardNet and CustardTransfer. Applying reinforcement-learning game-changing strategies to training data pipeline workload optimizers, along the lines of CustardNet and CustardTransfer, could lead to supporting players via constant automatic training.

References

1. Silver, E. A., Pyke, D. F., & Peterson, R. (1998). Inventory management and production planning and scheduling.

2. Sudhakar, A. V. V., Inala, R., Verma, A. K., Nag, K., Pandey, V., & Anand, P. S. (2025). Hybrid Rule-Based and Machine Learning Framework for Embedding Anti-Discrimination Law in Automated Decision Systems. In 2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT) (pp. 1–6). 2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT). IEEE. https://doi.org/10.1109/icicnct66124.2025.11232861.

3. Chopra, S., & Meindl, P. (2016). Supply chain management: Strategy, planning, and operation.

4. Nagabhyru, K. C., Garapati, R. S., & Aitha, A. R. (2025). UNIFIED INTELLIGENCE FABRIC: AI-DRIVEN DATA ENGINEERING AND DEEP LEARNING FOR CROSS-DOMAIN AUTOMATION AND REAL-TIME GOVERNANCE. Lex Localis, 23(S6), 3512-3532.

5. Nahmias, S., & Olsen, T. (2015). Production and operations analysis.

6. Paleti, S., Baliyan, M., Aitha, A. R., Reddy, B. A., Bhadauria, G. S., & Sing, S. A. (2025). Graph—LSTM Hybrid Model for Improving Fraud Detection Accuracy in E-Commerce Financial Services. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1-6).

7. Whitin, T. M. (1953). The theory of inventory management.

8. Rani, P. R. S., Kummari, D. N., Yellanki, S. K., Meda, R., Reddy Koppolu, H. K., & Inala, R. (2025). Blockchain and AI for Securing Electrical Infrastructure. In 2025 2nd International Conference on Computing and Data Science (ICCDS) (pp. 1–6). 2025 2nd International Conference on Computing and Data Science (ICCDS). IEEE. https://doi.org/10.1109/iccds64403.2025.11209487.

9. Hadley, G., & Whitin, T. M. (1963). Analysis of inventory systems.

10. Vajpayee, A., Khan, S., Gottimukkala, V. R. R., Sharma, D., & Seshasai, S. J. (2025). Digital Financial Literacy 4.0: Consumer Readiness for AI-Driven Fintech and Blockchain Ecosystems. International Insurance Law Review, 33(S5), 963-973.

11. Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2009). Effective forecasting and judgmental adjustments.

12. Garapati, R. S. (2025). Real-Time Monitoring and AI-Based Control of Industrial Robots Using Cloud-Hosted Web Applications. Available at SSRN 5612491.

13. Makridakis, S., Wheelwright, S., & Hyndman, R. (1998). Forecasting: Methods and applications.

14. Amistapuram, K. (2025). GENERATIVE AI FOR CLAIMS EXCEPTIONS AND INVESTIGATIONS: ENHANCING RESOLUTION EFFICIENCY IN COMPLEX INSURANCE PROCESSES. Available at SSRN 5785482.

15. Taylor, J. W., & Letham, B. (2018). Forecasting at scale.

16. Kumar, K. M., Banu S, P., Parasar, A., Walia, A., Inala, R., & Thulasimani, T. (2025). Enhancing Risk Management Strategies in Financial Institutions Using CNN and Support Vector Regression. In 2025 5th Asian Conference on Innovation in Technology (ASIANCON) (pp. 1–6). 2025 5th Asian Conference on Innovation in Technology (ASIANCON). IEEE. https://doi.org/10.1109/asiancon66527.2025.11280947

17. Gardner, E. S. (2006). Exponential smoothing: The state of the art.

18. Guntupalli, R. (2025, August). 5G and AI-Powered Cloud Security: Safeguarding Ultra-Low Latency Networks. In 2025 International Conference on Artificial Intelligence and Machine Vision (AIMV) (pp. 1-4). IEEE.

19. Petropoulos, F., et al. (2022). Forecasting: Theory and practice.

20. Ord, J. K., Koehler, A. B., & Snyder, R. D. (1997). Estimation and prediction for a class of dynamic nonlinear models.

21. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory.

22. Nagabhyru, K. C. (2025). Beyond Automation: The 2025 Role of Agentic AI in Autonomous Data Engineering and Adaptive Enterprise Systems.

23. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning.

24. Aitha, A. R., & Jyothi Babu, D. A. (2025). Agentic AI-Powered Claims Intelligence: A Deep Learning Framework for Automating Workers Compensation Claim Processing Using Generative AI. Available at SSRN 5505223.

25. Lim, B., et al. (2021). Temporal fusion transformers for interpretable multi-horizon forecasting.

26. Lebcir, I., Shah, C. A., Nagubandi, A. R., Dhoke, S. M., sikh, G. S. & Mishra, M. K. (2025). FinTech and Financial Inclusion in Emerging Economies: An Empirical Assessment. Advances in Consumer Research, 2(6), 2005-2011.

27. Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using RNNs.

28. Deep Learning-Driven Optimization of ISO 20022 Protocol Stacks for Secure Cross-Border Messaging. (2024). MSW Management Journal, 34(2), 1545-1554.

29. Seeger, M., et al. (2016). Bayesian intermittent demand forecasting.

30. Rao, A. N., Garapati, R. S., Suganya, R. T., Kaliappan, A., & Kamaleshwar, T. (2025, August). Smart Solar Harvesting and Power Management in IoT Nodes Through Deep Learning Models. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1-6). IEEE.

31. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction.

32. Powell, W. B. (2011). Approximate dynamic programming.

33. Bertsekas, D. P. (2017). Dynamic programming and optimal control.

34. Nagubandi, A. R. (2024). Breakthrough Real-Time AI-Driven Regulatory Intelligence for Multi-Counterparty Derivatives and Collateral Platforms: Autonomous Compliance for IFRS, EMIR, NAIC, SOX & Emerging Regulations. Journal of Information Systems Engineering and Management, 9.

35. Chen, F. (1999). Decentralized supply chains subject to information delays.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

36. Guntupalli, R. (2025, August). AI-Enhanced Data Encryption Techniques for Cloud Storage. In 2025 International Conference on Artificial Intelligence and Machine Vision (AIMV) (pp. 1-6). IEEE.

37. Shang, K. H., & Song, J. S. (2003). Newsvendor bounds and heuristic policies.

38. Kumar, B. H., Nuka, S. T., Recharla, M., Chakilam, C., Suura, S. R., & Pandugula, C. (2025). Addressing Ethical Challenges in AI-Driven Health Predictions. In 2025 2nd International Conference on Computing and Data Science (ICCDS) (pp. 1–6). 2025 2nd International Conference on Computing and Data Science (ICCDS). IEEE. https://doi.org/10.1109/iccds64403.2025.11209545

39. Zipkin, P. H. (2008). On the structure of lost-sales inventory models.

40. Amistapuram, K. (2025). Agentic AI for Next-Generation Insurance Platforms: Autonomous Decision-Making in Claims and Policy Servicing. Journal of Marketing & Social Research, 2, 88-103.

41. Manyika, J., et al. (2011). Big data: The next frontier for innovation.

42. Nagabhyru, K. C., Rani, M., Reddy, D. S., Krishnaraj, V., G, Renukaprasad., & V, Praveen. (2025). Machine Learning-Driven Fault Detection in Electric Vehicles via Hybrid Reinforcement Learning Model. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1–6). 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS). IEEE. https://doi.org/10.1109/iacis65746.2025.11211492.

43. Provost, F., & Fawcett, T. (2013). Data science for business.

44. Segireddy, A. R. (2025). GENERATIVE AI FOR SECURE RELEASE ENGINEERING IN GLOBAL PAYMENT NETWORK. Lex Localis: Journal of Local Self-Government, 23.

45. Inmon, W. H. (2005). Building the data warehouse.

46. Sriram, H. K., Challa, K., & Gadi, A. L. (2025). AI and Cloud-Driven Transformation in Finance, Insurance, and the Automotive Ecosystem: A Multi-Sectoral Framework for Credit Risk, Mobility Services, and Consumer Protection. Anil Lokesh and singreddy, Sneha, AI and Cloud-Driven Transformation in Finance, Insurance, and the Automotive Ecosystem: A Multi-Sectoral Framework for Credit Risk, Mobility Services, and Consumer Protection (March 15, 2025).

47. Stonebraker, M., et al. (2018). Data curation at scale.

48. Zaharia, M., et al. (2016). Apache Spark: A unified engine for big data processing.

49. Kreps, J. (2014). I heart logs.

50. Amistapuram, K. (2024). Generative AI in Insurance: Automating Claims Documentation and Customer Communication. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 15(3), 461–475. https://doi.org/10.61841/turcomat.v15i3.15474

51. Demchenko, Y., et al. (2014). Architecture framework and components for big data analytics.

52. Pandiri, L. (2025, May). Exploring Cross-Sector Innovation in Intelligent Transport Systems, Digitally Enabled Housing Finance, and Tech-Driven Risk Solutions A Multidisciplinary Approach to Sustainable Infrastructure, Urban Equity, and Financial Resilience. In 2025 2nd International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE) (pp. 1-12). IEEE.

53. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey.

54. Vadisetty, R., Polamarasetti, A., Goyal, M. K., Rongali, S. K., kumar Prajapati, S., & Butani, J. B. (2025, May). Cloud-Based Immersive Learning: The Role of Virtual Reality, Big Data, and Generative

AI in Transformative Education Experiences. In 2025 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) (pp. 1-6). IEEE.

55. Zikopoulos, P., et al. (2011). Understanding big data.

56. Reddy Segireddy, A. (2024). Federated Cloud Approaches for Multi-Regional Payment Messaging Systems. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 15(2), 442–450. https://doi.org/10.61841/turcomat.v15i2.15464.

57. Breck, E., et al. (2017). The ML test score.

58. Rongali, S. K., & Varri, D. B. S. (2025). AI in health care threat detection. World Journal of Advanced Research and Reviews, 25(3), 1784-1789.

59. Villalobos, J. R., et al. (2018). Data quality in analytics pipelines.

60. Guntupalli, R. (2025). Federated Deep Learning for Predictive Healthcare: A Privacy-Preserving AI Framework on Cloud-Native Infrastructure. Vascular and Endovascular Review, 8(16s), 200-210.

61. Otto, A., & Kotzab, H. (2012). Does supply chain visibility affect supply chain performance?

62. Polamarasetti, S., Kakarala, M. R. K., Goyal, M. K., Butani, J. B., Rongali, S. K., & kumar Prajapati, S. (2025, May). Designing Industry-Specific Modular Solutions Using Salesforce OmniStudio for Accelerated Digital Transformation. In 2025 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) (pp. 1-13). IEEE.

63. Christopher, M. (2016). Logistics and supply chain management.

64 Challa, K., Sriram, H. K., & Gadi, A. L. (2025). Leveraging AI, ML, and Gen AI in Automotive and Financial Services: Data-Driven Approaches to Insurance, Payments, Identity Protection, and Sustainable Innovation.

65. Min, H. (2010). Artificial intelligence in supply chain management.

66. Kumar, M. V. K., Kannan, S., Annapareddy, V. N., Adusupalli, B., Paleti, S., & Challa, S. R. (2025). Transforming Underground Electric Cable Management with AI in Smart Cities. In 2025 2nd International Conference on Computing and Data Science (ICCDS) (pp. 1–6). 2025 2nd International Conference on Computing and Data Science (ICCDS). IEEE. https://doi.org/10.1109/iccds64403.2025.11209611

67. Choi, T. M., Wallace, S. W., & Wang, Y. (2018). Big data analytics in operations management.

68. Varri, D. B. S. V. (2025). Human-AI collaboration in healthcare security.

69. Dubey, R., et al. (2019). Big data analytics and artificial intelligence in supply chains.

70. Recharla, M., & Nuka, S. T. (2025). Translational Approaches To Commercializing Neurodegenerative Therapies: Bridging Laboratory Research With Clinical Practice. South Eastern European Journal of Public Health, 121–144.

71. Grewal, D., et al. (2020). Retailing in a post-pandemic world.

72. Nagubandi, A. R. (2025). Advanced Predictive Autonomous Agents for Multiportfolio Risk Analytics and Real-Time Enterprise P&L Decisioning: Self-Learning AI Systems for Multi-counterparty Derivatives, Collateral Valuation, and Accounting Reconciliation. Collateral Valuation, and Accounting Reconciliation (December 01, 2025).

73. Hübner, A., Holzapfel, A., & Kuhn, H. (2016). Operations management in multi-channel retailing.

74. Piotrowicz, W., & Cuthbertson, R. (2014). Introduction to the special issue on information technology in retail.

75. Pantano, E., et al. (2018). Competing during a pandemic? Retailers' ups and downs during COVID-19.

76. Raj, M. S., Kaulwar, P. K., Raja, P. S., Pokhriyal, S., Ponnusamy, S., & Ramani, G. G. (2025, May). Future Proof Civic Participation Platforms with Behavioral Insight Driven Policy Making Artificial Intelligence and Big Data Analytics. In International Conference on Sustainability Innovation in Computing and Engineering (ICSICE 2024) (pp. 648-660). Atlantis Press.

77. Brynjolfsson, E., Hu, Y., & Rahman, M. (2013). Competing in the age of omnichannel retailing.

78. Paleti, S., Baliyan, M., Aitha, A. R., Reddy, B. A., Bhadauria, G. S., & Sing, S. A. (2025). Graph—LSTM Hybrid Model for Improving Fraud Detection Accuracy in E-Commerce Financial Services. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1–6). 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS). IEEE. https://doi.org/10.1109/iacis65746.2025.11210906

79. Cao, L., & Li, L. (2015). The impact of cross-channel integration.

80. kumar Kakarala, M. R., & Rongali, S. K. (2025). Existing challenges in ethical AI: Addressing algorithmic bias, transparency, accountability and regulatory compliance.

81. Kache, F., & Seuring, S. (2017). Challenges and opportunities of digital information at the intersection of big data analytics.

82. Balaji Adusupalli. (2025). Integrated Financial Ecosystems: AI-Driven Innovations in Taxation, Insurance, Mortgage Analytics, and Community Investment Through Cloud, Big Data, and Advanced Data Engineering. Journal of Information Systems Engineering and Management, 10(36s), 1103–1117. https://doi.org/10.52783/jisem.v10i36s.6709

83. Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey.