International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307 OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/CSEIT251112374

From Batch to Streaming: Building Real-time Inference Pipelines for Machine Learning

Chirag Maheshwari

Courant Institute of Mathematical Sciences, New York University

ARTICLEINFO

Article History:

Accepted : 23 Feb 2025

Published: 25 Feb 2025

Publication Issue

Volume 11, Issue 1

January-February-2025

Page Number

3546-3555

ABSTRACT

Modern machine learning applications are experiencing a fundamental shift from traditional batch processing toward real-time inference pipelines, driven by the increasing demand for timely and context-aware predictions. This article comprehensively explores different training and serving architectures, ranging from conventional batch processing to sophisticated streaming approaches. It examines the evolution of ML pipelines, discussing the advantages and challenges of various architectural patterns, including batch training with batch predictions, batch training with streaming predictions, and fully streaming approaches. The article delves into the implementation considerations for each architecture, addressing critical challenges such as data freshness, concept drift, and model degradation. It also explores continual learning systems, representing the cutting edge of adaptive ML architectures. The article includes a detailed analysis of best practices for implementation, covering architecture selection, system design considerations, and operational excellence. Through this systematic examination,

the article provides practitioners with a structured framework for selecting and implementing appropriate ML pipeline architectures based on their specific requirements and constraints.

Keywords: Machine Learning Pipelines, Real-time Inference, Continual Learning, Stream Processing, MLOps Architecture

Introduction

Modern machine learning applications are fundamentally transforming how they process and analyze data. The shift from traditional batch processing to real-time analytics represents one of the most significant changes in the machine learning landscape, with organizations facing increasing pressure to process and analyze data streams in real time to maintain competitive advantage [1]. This transition is particularly crucial as businesses seek to leverage machine learning for immediate decision-making and responsive customer experiences.

The evolution toward real-time processing has been driven by the exponential growth in data generation and the increasing sophistication of machine learning applications [1]. Traditional batch processing, historically operated on daily or hourly cycles, is being supplemented or replaced by streaming architectures that can process and analyze data as it arrives. This transformation reshapes how organizations approach data processing and decision-making, particularly in time-sensitive applications [2].

The real-time analytics market has shown a remarkable growth trajectory, driven by the increasing adoption of IoT devices and the rise of digital transformation initiatives across industries [2]. North America maintains a dominant position in this market due to its advanced technological infrastructure and early adoption of innovative solutions, while other regions are rapidly expanding their real-time analytics capabilities [2]. This growth is supported by the proliferation of cloud computing services and the advancement of edge computing technologies, making real-time processing more accessible and cost-effective for organizations of all sizes.

Implementing real-time ML systems presents unique challenges in data quality, system reliability, and processing efficiency [1]. Organizations must carefully balance the trade-offs between data freshness and processing complexity while ensuring their systems can handle the increased computational demands of real-time analysis. Maintaining data quality while processing it at high speeds requires sophisticated monitoring and validation systems [1].

Financial services institutions have emerged as early adopters of real-time ML processing, implementing systems for fraud detection and trading algorithms that must operate with minimal latency [2]. In manufacturing, the Industrial Internet of Things (IIoT) has driven the adoption of real-time analytics for predictive maintenance and quality control, while healthcare organizations are leveraging real-time processing for patient monitoring and diagnostic support [1].

The transition from batch to streaming architectures manifests in several distinct patterns, each representing a different balance between processing immediacy and implementation complexity [1]. While still valuable for specific applications, traditional batch processing is augmented by hybrid approaches combining periodic training with real-time inference. The selection of appropriate architecture depends heavily on particular use cases and organizational requirements [2].

Understanding these approaches becomes increasingly critical as organizations evolve their ML infrastructure [1]. The success of real-time ML implementations relies on technical architecture, organizational readiness, and precise alignment with business objectives [2]. This evolution represents a significant shift in how organizations approach machine learning, requiring new skills, tools, and methodologies to leverage real-time capabilities effectively.

Understanding the Evolution of ML Pipelines

The evolution of machine learning pipelines from batch to streaming architectures represents a fundamental shift in how organizations process and utilize data for predictive analytics. This transformation has been driven by the complex challenges of real-time feature engineering, data freshness, and the need to maintain consistent performance across varying data volumes [3]. While still relevant for specific use cases, the traditional batch-processing approach is increasingly being complemented or replaced by streaming architectures that can handle continuous data flows and provide immediate insights.

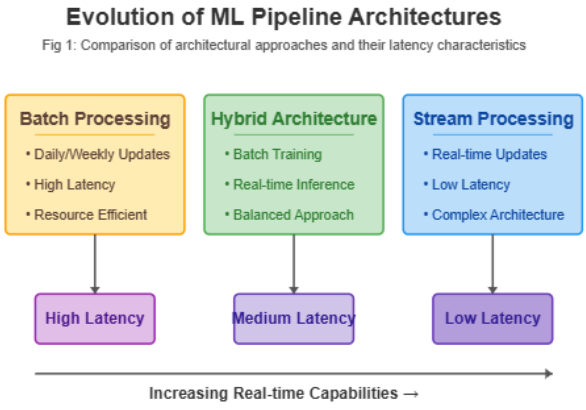
The journey toward real-time ML processing has introduced new challenges in data processing and system architecture. Organizations must address issues such as feature consistency between training and serving, handling out-of-order events, and managing the complexity of real-time feature computation [3]. These challenges become particularly acute when dealing with time-window aggregations and ensuring that features computed in real time match those used during model training.

Stream processing has emerged as a critical paradigm for handling continuous data flows, offering advantages regarding data freshness and processing latency [4]. Unlike batch processing, which operates on fixed chunks of historical data, stream processing enables organizations to process data as it arrives, leading to more timely insights and faster decision-making capabilities. This approach has become particularly valuable in use-cases requiring immediate action based on incoming data patterns.

Modern ML pipelines must balance the trade-offs between batch and stream processing approaches [4]. While stream processing offers benefits in terms of data freshness and reduced latency, it also introduces additional complexity in system design and maintenance. Organizations must consider data consistency, processing guarantees, and system scalability when choosing between batch and streaming approaches for their specific use cases.

The implementation of real-time ML pipelines requires careful attention to system architecture and data management practices [3]. Organizations must address challenges such as maintaining feature stores that can handle batch and streaming computations, ensuring consistent feature definitions across different processing paradigms, and managing the complexity of deploying models in real-time environments. These technical considerations are coupled with operational challenges such as monitoring system performance and maintaining data quality in a streaming context [4].

In the following sections, we detail three major architectural paradigms—batch training with batch predictions, batch training with streaming predictions, and fully streaming (stream training with stream predictions)—before discussing the advanced concept of continual learning.



Batch Training with Batch Predictions

The traditional approach to machine learning deployment—where both training and inference (predictions) occur in scheduled batches—continues to serve as a foundational architecture in many organizations. By processing data in bulk at regular intervals (e.g., nightly or weekly), teams can simplify operational workloads and dedicate well-defined windows for data validation, model retraining, and large-scale inference jobs. [5].

One of the primary strengths of batch processing lies in its ability to handle large datasets efficiently. Organizations can effectively manage memory constraints by processing data in predetermined batch sizes while maintaining training stability. This approach is particularly beneficial when dealing with deep learning models, where batch normalization and other batch-dependent techniques play crucial roles in model performance [5]. The architecture's systematic nature allows for better control over the training process and more consistent model updates.

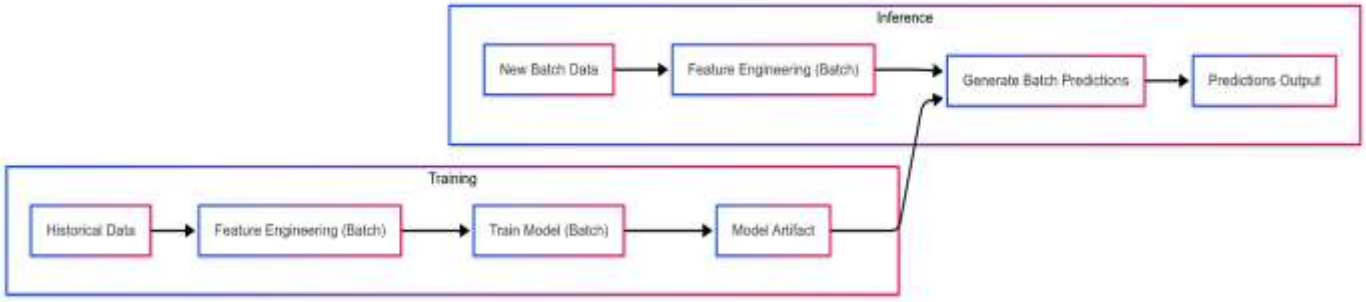
The batch training approach provides significant advantages regarding computational efficiency and resource utilization. When processing data in batches, systems can optimize memory usage and use parallel processing capabilities [6]. This becomes particularly important in large-scale data processing scenarios, where efficient resource management is crucial for maintaining system performance and controlling operational costs.

In practical applications, batch processing demonstrates its value through reliability and predictability. The architecture excels in scenarios where immediate real-time results aren't critical, such as periodic reporting systems and offline analytics [6]. By processing data in scheduled intervals, organizations can ensure thorough data validation and quality checks, making it particularly suitable for applications requiring high accuracy and comprehensive error checking.

Systematic batch processing provides natural support for model evaluation and validation. Data scientists can implement thorough testing procedures and maintain detailed performance metrics across different batches [5]. This becomes especially valuable when model stability and predictable performance are prioritized over real-time responsiveness. The architecture naturally accommodates comprehensive testing protocols and validation procedures.

However, organizations must carefully consider the inherent trade-offs of batch-processing architectures [6]. While batch processing offers advantages in terms of efficiency and stability, it introduces latency between data collection and result generation. This delay makes it less suitable for applications requiring immediate responses or real-time decision-making. Additionally, batch operations require careful scheduling and resource allocation to manage periodic processing loads effectively.

An example use case: A retail company that generates product recommendations once per day. Overnight, the system performs feature engineering on all sales and clickstream data from the previous day, trains or retrains the model, and uploads the resulting recommendations to the production database. This schedule ensures stable performance without the complexity of real-time data ingestion and inference, making it ideal for organizations that do not require immediate updates.



Batch Training with Streaming Predictions

The hybrid approach of combining batch training and real-time (streaming) predictions is gaining traction for its ability to bridge the gap between stable, offline model updates and immediate, low-latency inference. Organizations can maintain robust model quality by periodically retraining a model on large historical datasets and leveraging cost-efficient batch processes. At the same time, they can serve near real-time predictions using the latest streaming data, ensuring responsive decision-making. [7].

The foundation of this hybrid architecture relies on a well-structured feature engineering pipeline that can handle batch and streaming data effectively. The pipeline must address several key aspects: feature definition and computation, data validation and testing, and monitoring of feature distributions. The implementation typically involves creating reusable feature transformation components that can be applied consistently across batch training and real-time serving scenarios [7].

A crucial component of this architecture is the feature store, which serves as a central repository for managing feature definitions and computations. The feature store must handle batch-computed features from historical data and real-time feature computations, ensuring that features remain consistent between training and serving environments. This includes maintaining proper versioning of feature definitions and providing mechanisms for feature sharing across different models and applications [7].

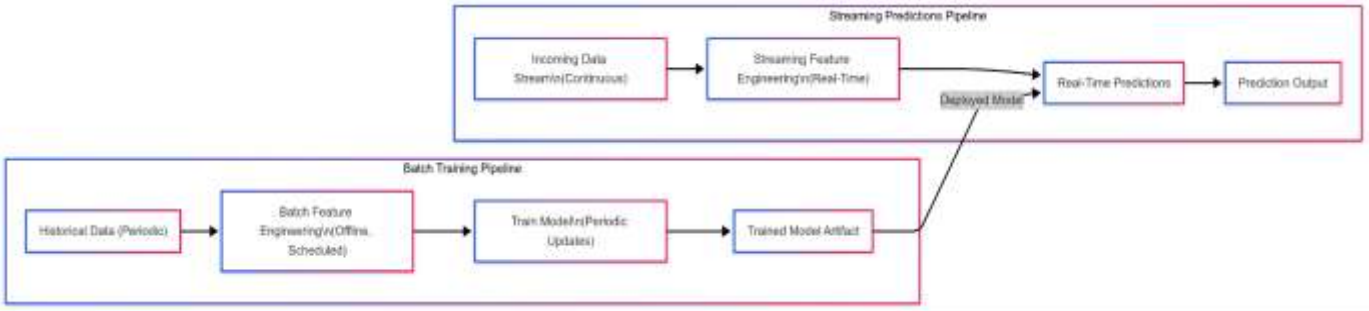
Data validation plays a vital role in maintaining the reliability of the hybrid system. The architecture must implement robust validation procedures to detect data quality issues and distribution shifts in batch and streaming contexts. This includes setting up automated testing frameworks that validate feature computations and ensuring that feature distributions remain consistent between training and serving environments [7].

Implementing this hybrid approach requires careful attention to system design and operational considerations. Organizations must establish clear protocols for feature development, testing, and deployment. This includes implementing comprehensive testing strategies covering batch and streaming components and ensuring that feature computations remain consistent across different processing modes [7].

Monitoring becomes a critical aspect of maintaining system health in this hybrid architecture. Organizations must implement monitoring systems that can track feature distributions, detect data quality issues, and identify potential performance degradation. This includes setting up alerts for significant deviations in feature distributions and maintaining detailed logs of feature computation processes [7].

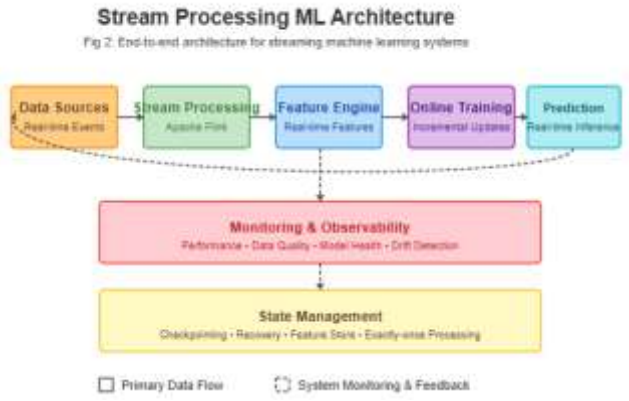
An example use-case: Consider an e-commerce company that retrains its recommendation model nightly using a batch pipeline that processes historical transactions. The newly trained model is then deployed to an online inference endpoint. Meanwhile, real-time user actions (product views, clicks) are fed into a streaming feature pipeline that generates predictions on the fly. Thanks to a shared feature store, the transformations used to generate user/item features in the batch pipeline are identical to those used during real-time inference, ensuring consistent and accurate recommendations.

This approach allows the business to combine the accuracy benefits of training on rich historical data with the agility of real-time inference, ultimately delivering personalized experiences to end users without incurring the complexities of continuous online model updates.



Stream Training with Stream Prediction

Stream training with stream prediction represents the most advanced approach to real-time machine learning, enabling continuous model updates alongside immediate inference. Rather than relying on static, periodically retrained models, online learning algorithms process new data as soon as it arrives, incrementally refining model parameters and drastically reducing the time between data ingestion and model improvement. [8]. The fully streaming approach requires sophisticated infrastructure and careful system design to maintain reliability and performance across various use cases.



The fundamental architecture of a streaming ML system relies on key capabilities provided by stream processing frameworks. Apache Flink's stateful stream processing enables organizations to maintain and update model states efficiently while processing incoming data streams. The framework's ability to handle event time processing and late-arriving data makes it particularly suitable for real-world applications where data arrival patterns may be unpredictable [8]. These capabilities become crucial when implementing online learning algorithms that must update model parameters incrementally.

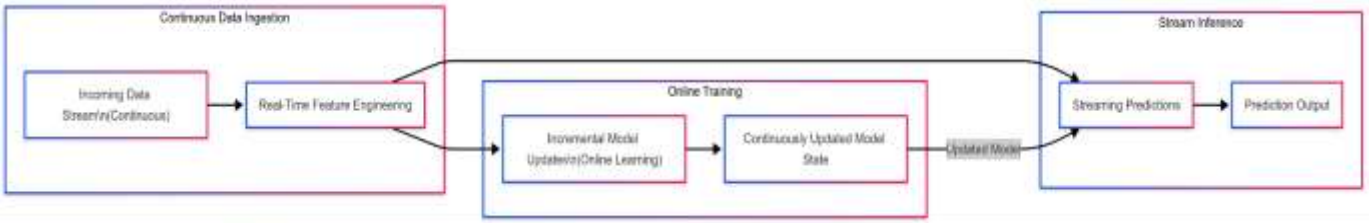
Modern stream processing architectures must address several architectural challenges to maintain reliable performance in production environments. Integrating machine learning models into streaming pipelines introduces complexities around model serving, versioning, and monitoring [9]. Organizations must carefully consider managing the model lifecycle, handling model updates, and maintaining consistent performance across distributed serving instances while dealing with the inherent challenges of stream processing.

Performance monitoring in streaming ML systems requires specialized approaches that differ significantly from traditional batch processing scenarios. The architecture must support real-time tracking of both system health and model performance metrics [9]. This includes tracking data distribution shifts, detecting anomalies in model predictions, and maintaining visibility into the stream processing pipeline's performance characteristics, such as throughput and latency.

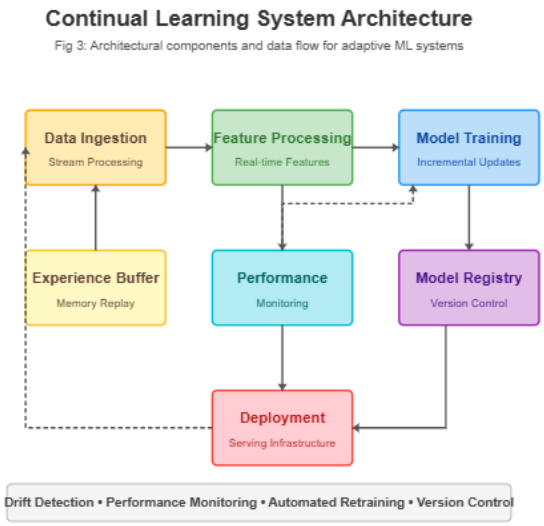
The implementation of streaming ML systems must address several technical challenges inherent to real-time processing. Apache Flink's checkpointing mechanism helps ensure fault tolerance and state consistency, which becomes crucial when maintaining model state across distributed environments [8]. The system must also implement appropriate windowing strategies to handle time-based aggregations and feature computations while efficiently processing out-of-order events.

Organizations implementing streaming ML architectures face challenges related to data quality, model stability, and system scalability [9]. The architecture must incorporate robust validation procedures that can operate in real-time, implement appropriate error-handling mechanisms, and maintain system reliability under varying load conditions. This includes developing strategies for handling edge cases, managing resource utilization, and ensuring consistent model performance across different deployment scenarios.

An example use-case: A fraud detection system ingests millions of transactions per day. Each incoming transaction event is processed using a streaming framework such as Apache Flink to extract features (e.g., transaction amount and user velocity). Incrementally updates a classifier with online learning methods. With every checkpoint, the model state is saved, ensuring fault tolerance. Because the model updates continuously, it quickly adapts to emerging fraud patterns instead of waiting for an overnight batch job. As a result, latency is minimized, and the system remains resilient to constantly shifting transaction behaviors. By combining real-time feature engineering with incremental model updates, stream training with stream prediction allows organizations to act on evolving data patterns immediately, keeping the model relevant and responsive in high-velocity domains where every second of delay can be costly.



Continual Learning Systems



Continual learning represents an adaptive approach to machine learning where systems can learn from a continuous stream of data while maintaining performance on previously learned tasks. This methodology addresses the fundamental challenge of learning in dynamic environments where data distributions evolve. Modern continual learning systems must balance the stability-plasticity dilemma, ensuring that models can acquire new knowledge without compromising their existing capabilities [10].

The core strength of continual learning systems lies in their ability to adapt to concept drift and distribution shifts while maintaining model performance. These systems employ sophisticated mechanisms to prevent catastrophic forgetting, which occurs when new learning interferes with previously acquired knowledge. Research has shown that careful management of model parameters and selective synaptic plasticity can help maintain performance across different tasks while allowing for continued learning [10]. This balance is crucial for maintaining system reliability in production environments.

Implementing effective continual learning systems requires careful consideration of practical deployment aspects in production environments. Organizations must develop robust monitoring systems that track model performance and learning dynamics over time

[11]. This includes implementing sophisticated metrics for detecting concept drift, measuring catastrophic forgetting, and evaluating the system's ability to maintain performance across different tasks and periods.

Production deployment of continual learning systems demands specialized architectures that can handle the complexities of incremental learning. These systems must incorporate mechanisms for efficient parameter updates and knowledge retention, often utilizing techniques such as experience replay or elastic weight consolidation [11]. The architecture must also include robust evaluation frameworks that assess immediate performance and long-term learning stability.

Monitoring becomes particularly critical in continual learning contexts, as systems must track not only current performance but also the evolution of model behavior over time. Organizations must implement comprehensive monitoring solutions that detect various types of drift, including feature, concept, and label drift [10]. This monitoring infrastructure must be capable of triggering appropriate interventions when significant deviations are detected.

The success of continual learning systems in production environments depends heavily on adequately implementing fallback mechanisms and safety protocols. Organizations must establish clear procedures for handling performance degradation and implementing model rollbacks when necessary [11]. This includes maintaining versions of well-performing models that can be deployed quickly if the continually learning system encounters issues.

Best Practices for Implementation

Implementing machine learning systems in production environments requires a systematic approach to MLOps practices and design patterns. Organizations must establish robust processes for developing, deploying, and maintaining ML systems while ensuring reliability, scalability, and operational efficiency [12].

7.1. Architecture Selection

The foundation of successful ML system implementation begins with MLOps practices that emphasize automation and reproducibility. Organizations must establish transparent processes for continuous integration and continuous delivery (CI/CD) of machine learning models, ensuring that model deployment and updates can be handled efficiently and reliably [12]. This includes implementing automated testing and validation procedures to verify model performance before deployment.

Data pipeline architecture plays a crucial role in successful ML implementations. Teams must implement robust data versioning and feature engineering practices that ensure reproducibility and traceability of model training [13]. This includes establishing transparent processes for data validation, feature computation, and model validation that can scale with increasing data volumes and complexity.

Organizations must also consider the operational aspects of their ML architecture. This includes implementing proper monitoring and logging systems that track model performance, data quality, and system health [12]. The architecture should support automated retraining pipelines that update models when performance degrades, or new data patterns emerge.

7.2. System Design Considerations

Effective system design in ML requires careful attention to technical and operational requirements. Modern ML systems must implement comprehensive monitoring capabilities that can track model accuracy, feature distributions, prediction latency, and system resource utilization [13]. This includes setting up proper logging frameworks to help diagnose issues and track system behavior over time.

Scalability considerations must be embedded in the system design from the start. This includes implementing proper model-serving infrastructure that can handle varying loads and implementing feature stores that can serve features efficiently for

training and inference [12]. The design should also include proper testing frameworks to validate model performance across different scenarios and data distributions.

Version control in ML systems extends beyond traditional code versioning, including model artifacts, data schemas, and hyperparameters. Organizations must implement MLOps practices that ensure reproducibility of model training and deployment [13]. This includes maintaining proper documentation of model architectures, feature definitions, and deployment configurations.

7.3. Operational Excellence

Operational excellence in ML systems requires robust monitoring, maintenance, and continuous improvement processes. Organizations must implement comprehensive monitoring systems to detect model drift, data quality, and system performance problems [12]. This includes setting up appropriate alerting mechanisms to notify teams when interventions are needed.

Performance metrics and service level objectives (SLOs) must be clearly defined and monitored. This includes tracking technical metrics like prediction latency and business metrics that measure the actual impact of ML models [13]. Organizations should implement proper capacity planning procedures that ensure sufficient resources are available for training and serving workloads.

Security and governance considerations must be integrated throughout the ML lifecycle. This includes implementing proper access controls, audit logging, and model governance practices [12]. Organizations should establish transparent processes for model validation, deployment approval, and ongoing monitoring of model performance and behavior.

Conclusion

The evolution from batch to streaming machine learning pipelines represents a transformative shift in how organizations approach real-time decision-making and model deployment. While traditional batch processing continues to serve essential use cases, the emergence of hybrid and streaming architectures has expanded the possibilities for real-time inference and continuous learning. Success in implementing these systems requires careful consideration of various factors, including data characteristics, operational requirements, and organizational capabilities. As the field continues to mature, we observe increasing convergence between different architectural approaches, with organizations adopting hybrid solutions that combine the benefits of both batch and streaming paradigms. The key to successful implementation lies in understanding the inherent trade-offs between different approaches and selecting architectures that align with specific use cases while maintaining an appropriate balance between complexity and maintainability.

References

[1]. Nina Zumel, "Real-Time Data in Machine Learning: Challenges and Solutions," Dataversity, 2022. [Online]. Available: https://www.dataversity.net/real-time-data-in-machine-learning-challenges-and-solutions/

[2]. Aarti Dhapte, "Real-Time Analytics Market Research Report: By Application (Marketing Analytics, Financial Analytics, Operational Analytics, Customer Analytics, Supply Chain Analytics), By Deployment Model (On-Premises, Cloud-Based, Hybrid), By Technology (Streaming Analytics, Complex Event Processing, Data Mining, Data Warehousing), By End Use (Retail, Healthcare, Manufacturing, Telecommunications, Banking and Financial Services) and By Regional (North America, Europe, South America, Asia Pacific, Middle East and Africa) - Forecast to 2034," Market Research Future, Feb. 2025. [Online]. Available: https://www.marketresearchfuture.com/reports/real-time-analytics-market-37074

[3]. Nick Parsons and Aditya Nambiar, "Challenges of Building Realtime Machine Learning Pipelines," Fennel.ai, 2022. [Online]. Available: https://fennel.ai/blog/challenges-of-building-realtime-ml-pipelines/

[4]. Daniel Greenberg, "Optimizing Data Pipelines: Understanding Batch Processing vs. Stream Processing," Rivery, Jan. 2025. [Online]. Available: https://rivery.io/blog/batch-vs-stream-processing-pros-and-cons-2/

[5]. CloudFactory, "Everything you need to know about batches in Machine Learning," CloudFactory. [Online]. Available: https://wiki.cloudfactory.com/docs/mp-wiki/training-parameters/everything-you-need-to-know-about-batches-in-machine-learning

[6]. GeeksforGeeks, "What is the difference between batch processing and real-time processing?" GeeksforGeeks, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-the-difference-between-batch-processing-and-real-time-processing/

[7]. Paul Iusztin, "A Framework for Building a Production-Ready Feature Engineering Pipeline," Towards Data Science, 2023. [Online]. Available: https://towardsdatascience.com/a-framework-for-building-a-production-ready-feature-engineering-pipeline-f0b29609b20f/

[8]. Sahil Sharma, "Apache Flink for Real-Time Stream Processing," Medium, 2023. [Online]. Available: https://medium.com/@DataEngineeer/apache-flink-for-real-time-stream-processing-e83335a70cfe

[9]. Chip Huyen, "Real-Time Machine Learning: Architecture and Challenges," InfoQ, 2022. [Online]. Available: https://www.infoq.com/presentations/ml-arch-challenges/

[10]. Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu, "A Comprehensive Survey of Continual Learning: Theory, Method and Application," 2023. [Online]. Available: https://arxiv.org/abs/2302.00487

[11]. Vincenzo Lomonaco, "Continual Learning for Production Systems," Medium - Continual AI, Medium, 2019. [Online]. Available: https://medium.com/continual-ai/continual-learning-for-production-systems-304cc9f60603

[12]. AWS, "What is MLOps?," AWS. [Online]. Available: https://aws.amazon.com/what-is/mlops/

[13]. Eugeneyan, "More Design Patterns For Machine Learning Systems," Eugene Yan's Blog, 2023. [Online]. Available: https://eugeneyan.com/writing/more-patterns/