International Journal of AI, BigData, Computational and Management Studies
Noble Scholar Research Group | Volume 7, Issue 1, 39-44, 2026
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V7I1P106

Original Article

AutoML-Enabled Infrastructure for Adaptive Personalization in High-Traffic Shopping Events

Udit Agarwal
Independent Researcher, USA.

Received On: 27/11/2025 ***Revised On: 29/12/2025*** ***Accepted On: 06/01/2026*** ***Published On: 17/01/2026***

Abstract: *The necessity for highly adaptable, ultra-low latency personalization systems is critical in high-traffic e-commerce environments, particularly during unpredictable demand surges such as high-volume shopping events. Traditional machine learning (ML) paradigms, constrained by manual feature engineering, model design, and slow iteration cycles, are fundamentally unsuitable for managing massive, unpredictable load and dynamically changing user intent during peak traffic events. This paper introduces a resilient architectural framework built upon the Automated Machine Learning (AutoML) paradigm, specifically optimized for deep recommender systems (AutoRecSys). The proposed architecture details an ultra-low latency feature pipeline integrating Apache Kafka for high-frequency ingestion, Apache Flink for stream processing, and dedicated Feature Stores (Redis/Delta Lake) to mitigate catastrophic train/serve skew and ensure comprehensive data consistency. The efficacy of AutoRecSys techniques such as sampling-based searches and multi-fidelity optimization is substantiated by demonstrating quantified production benefits, including architectural efficiency gains that result in up to a 25% Query per Second (QPS) increase and resource optimization that reduces embedding parameters by 50% to 95%. The holistic, integrated system consistently achieves an end-to-end latency below the critical 100-millisecond threshold, providing a robust, evidence-based blueprint for deploying adaptive, scalable ML solutions in high-stakes, time-constrained production environments.*

Keywords: *Automated Machine Learning, AutoRecSys, Recommender Systems, Low Latency, Feature Store, Adaptive Personalization, High-Traffic E-commerce, MLOps, Train/Serve Skew.*

1. Introduction

1.1. The Critical Challenge of Scalability and Latency in Digital Commerce

Digital commerce platforms operating at scale encounter immense pressure during scheduled or sudden high-traffic events, such as seasonal sales or flash promotions. These applications experience unpredictable and massive surges in demand, which necessitates a system architecture that is not merely reactive but intrinsically predictive and resilient.

A fundamental challenge in modern e-commerce is mitigating the impact of system latency. The time delay between a shopper's action and the system's response is directly correlated with customer satisfaction and conversion rates. Industry analysis reveals a critical performance benchmark: a delay of merely 100 milliseconds can translate to a 1% loss in sales revenue. This established metric underscores the operational imperative to maintain sub-second response times across the entire personalization pipeline. When response times degrade, customer frustration increases, undermining trust and leading to lost purchases. To maintain a smooth, trustworthy interaction flow, the system must deliver instant recommendations and real-time nudges, necessitating an ultra-low latency architecture engineered for speed at scale.

Traditional autoscaling methods, which rely on reactive rules (e.g., adding resources only after CPU usage exceeds a threshold), are often insufficient. This reactive latency causes performance degradation to occur before new resources can be fully integrated, leading to service degradation and revenue loss. Consequently, modern ML infrastructure must move beyond static, reactive scaling to incorporate predictive resource management and intrinsically efficient model deployment that minimizes the time required to serve personalized content.

1.2. Automated ML (AutoML) as the Driver for Adaptivity and Efficiency

The adoption of machine learning has broadened significantly, extending from foundational research into mainstream production environments across diverse industries. While ML offers considerable value, the complexity of deploying and maintaining high-performance models in production, especially deep recommender systems, remains substantial. The machine learning workflow, such as the CRISP-ML(Q) model, typically involves six phases, including model engineering, evaluation, deployment, and monitoring, necessitating continuous iteration. Data science experts often need to revisit earlier stages due to complexity, incurring significant time costs.

Automated Machine Learning (AutoML) systems simplify these labor-intensive tasks by automating components such as data preprocessing, feature creation, model selection, and hyperparameter tuning. The deployment of next-wave AutoML frameworks, such as Auto-sklearn 2.0 and FLAML, has demonstrated the capacity to achieve efficient and accurate solutions for large datasets under severe time constraints, accelerating computation time from hours to mere seconds on various benchmarks.

By automating the burdensome model engineering and optimization phases, AutoML releases human engineering resources. This capacity for accelerated, automated iteration is critical during high-traffic shopping events where rapid shifts in user behavior or data drift can quickly degrade model performance. AutoML provides the mechanism for near-instantaneous model recalibration or the efficient search and deployment of optimized model architectures, ensuring the system remains adaptive and high-performing when traffic spikes are at their peak.

1.3. Paper Organization

The remainder of this paper is structured to define the precise operational requirements for low-latency performance, detail the application and quantitative benefits of AutoRecSys in model optimization, and provide a thorough architectural analysis of the ultra-low latency MLOps feature pipeline required to support adaptive personalization in production.

2. The Operational Imperative: Defining Low-Latency Performance

2.1. Deconstructing End-to-End Latency in Real-Time Systems

In e-commerce, latency is not merely a theoretical metric but a direct measure of system reliability and user experience. End-to-end latency in a personalization system encapsulates several factors, including network transport, front-end rendering, and the time required for the AI model to generate a prediction (model inference).

To ensure a smooth, conversion-driving experience, the overall system response must adhere to stringent latency budgets. A production-ready architectural framework for real-time autonomous decision-making establishes an ambitious objective: the entire end-to-end latency must remain consistently under 100 milliseconds (ms).

2.2. The Centralized Vulnerability and Distributed Resilience

Designing systems for peak traffic requires addressing bottlenecks associated with monolithic or centralized architectures. Centralized systems inherently lack the distributed resilience required to tolerate large traffic spikes. Furthermore, a common architectural pitfall is co-locating large data representations (such as item embeddings) with the model serving infrastructure. This centralization leads to severe memory pressure, introduces single-node bottlenecks, and significantly degrades service reliability, particularly during model warm-up or sudden increases in user demand.

The analysis confirms a critical causal chain: Architectural centralization leads directly to excessive memory and service pressure, resulting in degraded reliability during unpredictable spikes. The engineering solution is architectural decoupling. This requires separating the high-volume data (features and representations) from the compute layer (model serving). A dedicated, distributed Feature Store is essential for handling high-throughput, ultra-low latency data access independently of the model server, thereby eliminating the single-node bottleneck and supporting the required resilience.

2.3. Strategic Adaptive Personalization

Effective personalization is not a one-size-fits-all solution; it requires matching the recommendation strategy to the specific stage of the customer journey. A successful three-stage personalization framework structures strategies for different user states: Strategic Segmentation for anonymous or new visitors (acquisition), Progressive Identification for engaged browsers (consideration), and Individual Personalization for known customers (retention).

This strategic requirement for adaptivity means that the underlying ML infrastructure must be capable of rapidly selecting, switching, or deploying different models optimized for varying contexts. For example, a high-speed, simpler model might be used for anonymous users, while a deep, feature-rich model may be reserved for known, high-value customers. This need for rapid, context-aware model management underscores the importance of a highly efficient model selection mechanism, a core function provided by AutoRecSys.

3. Automated Machine Learning for Deep Recommender Systems (AutoRecSys)

3.1. AutoRecSys: Specialized Automation for Recommender Systems

Recommender systems play a critical role in filtering information across digital platforms, including e-commerce. Deep recommender systems offer superior performance by capturing non-linear user-item relationships, but their design historically relies heavily on human experience and expert knowledge. Automated Machine Learning for Deep Recommender Systems (AutoRecSys) is introduced to automatically search for optimal candidates across the system’s building blocks, alleviating reliance on manual effort.

AutoRecSys differs from conventional AutoML techniques because it places particular emphasis on the optimization of the input component specifically the embedding matrix. For deep recommender systems, the embedding layer is a primary factor in memory consumption. The efficient learning of features from raw data dramatically influences all subsequent model components and is crucial to the final performance of the system.

In high-traffic environments where computational resources and serving costs are highly constrained, the automation of input components (embedding dimension search) is paramount. If the embedding layer consumes prohibitive memory, it limits the total number of concurrent experiments that can be supported and strains the architecture's capacity to serve large-capacity models efficiently. AutoRecSys directly optimizes this memory and compute trade-off, thereby fundamentally enabling scalable deployment under high pressure.

3.2. Taxonomy of Search Spaces for Optimization

The AutoRecSys framework classifies its search methodologies into five distinct categories, designed to target optimization across the full spectrum of the deep learning pipeline.

Table 1: Autorecsys Taxonomy and Production Efficiency Impact

Search Category	Primary Focus Area	Quantified Impact on Production/Efficiency
Feature Selection Search (Auto-FSS)	Optimizing the input features used by the model.	Reduces computational complexity and feature noise in training.
Embedding Dimension Search (Auto-EDS)	Determining optimal dimension sizes for feature embeddings.	Reduces memory consumption and computation cost
Feature Interaction Search (Auto-FIS)	Identifying beneficial feature interactions within the model.	Enhances predictive accuracy by finding powerful non-linear relationships.
Model Architecture Search (Auto-MAS)	Automating the overall neural network structure.	Enables the discovery of models with high Return on Investment (ROI) and up to 25% Query per Second (QPS) increase.

Specific search focuses provide clear operational benefits. Auto-EDS addresses the common practice of using uniform dimension sizes, which often leads to excessive resource consumption and suboptimal representation. Similarly, Auto-FIS mitigates noise and training complexity that arises from calculating non-beneficial feature interactions. Research has demonstrated that optimizing embeddings using AutoML (AutoSrh) yields tangible efficiency results, achieving better predictive performance with lower training time cost while preserving model performance and reducing embedding parameters.

This optimized resource utilization provides a dual benefit: improving model accuracy while simultaneously serving as a tool for cost and efficiency optimization. Reducing the overall model size and training time cost dramatically lowers infrastructure operational expenditure, and critically, improves inference speed, helping the system remain within the strict latency budget of 50 ms during peak loads.

3.3. Production Efficiency through Sampling-Based Optimization

The integration of AutoML into real-time production ranking systems, such as those at Meta, demands extreme efficiency. Due to the large scale of models and tight production schedules, AutoML must surpass human-tuned baselines using a small budget of model evaluation trials.

A sampling-based AutoML method addresses these challenges by leveraging a lightweight predictor-based searcher and reinforcement learning to explore the vast neural architecture search space. This method has been shown to achieve outstanding Return on Investment (ROI) against human baselines. The results demonstrate performance gains of up to 0.09% Normalized Entropy (NE) loss reduction, or a significant **25% Query per Second (QPS) increase**, achieved by sampling only one hundred candidate models on average.

The realized QPS increase is of particular significance. This outcome proves that AutoML optimization is not solely an academic pursuit of accuracy but rather a practical mechanism for infrastructural scaling and capacity enhancement. By improving model efficiency, the system can handle a greater volume of requests with the same resources, directly increasing resilience during high-traffic peaks. This validation accelerates the adoption of AutoML in high-stakes production ranking systems.

4. Ultra-Low Latency Feature Infrastructure and MLOps

4.1. The Streaming Architecture for Real-Time Feature Engineering

To support adaptive personalization, the MLOps pipeline must transition from static batch data processing to handling high-velocity, real-time data streams. Real-time feature engineering is necessary to quickly cleanse, aggregate, and enrich user click data, sensor readings, or financial transactions on the fly.

Core streaming frameworks, such as Apache Kafka (for ingestion), AWS Kinesis, and Apache Flink, are essential for managing high-velocity data streams. Apache Flink is noteworthy because it functions as an open-source system capable of processing both streaming and batch data, unifying diverse data processing use cases under a single, fault-tolerant execution model.

The homogeneity afforded by Flink's architecture is crucial for MLOps consistency. It allows the identical transformation logic, used to create crucial features, to be consistently applied across both historical training data (batch) and live inference events (stream). Utilizing versioned feature definitions and supporting incremental updates (maintaining rolling averages or counters reflecting the latest events) provides the necessary defense against feature inconsistencies and data drift. This architectural consistency is the first necessary step in minimizing *train/serve skew* before the data reaches the serving layer.

4.2. The Enterprise Feature Store: Consistency and Governance

Deploying real-time ML systems in production presents significant challenges, including massive engineering overhead, difficulties in scaling, and exorbitant costs associated with managing complex data pipelines. The Feature Store addresses these challenges by serving as a centralized, scalable platform for managing and serving features.

The Feature Store's primary architectural achievement is the resolution of train/serve skew. This critical failure mode occurs when subtle differences emerge between the features used for offline model training and those used for online inference, leading to a catastrophic decline in model performance. The Feature Store mitigates this by providing a unified serving layer. It ensures consistent data is available in the offline environment (historical data, optimized for batch retrieval in data lakes like Delta Lake) and the online environment (latest feature values, optimized for ultra-low latency lookup in high-speed stores like Redis).

Beyond consistency, the Feature Store reduces engineering overhead by automating batch and streaming data pipelines, enabling feature reuse across models and eliminating the duplication of data engineering efforts.

Table 2: Feature Store Role in Mitigating Production Challenges

Challenge	Feature Store Solution	Key Benefit in High-Traffic E-commerce
Training/Serving Skew	Consistent storage (Offline/Online stores) and unified serving APIs.	Ensures reliable model performance under dynamic load and maintains model calibration.
Massive Engineering Overhead	Automated Batch/Streaming pipelines; centralized management, feature reuse.	Reduces time spent creating bespoke, difficult-to-scale pipelines; accelerates ML project velocity.
Data Quality & Governance	Feature versioning, lineage tracking, and monitoring capabilities.	Provides accountability and traceability, crucial for debugging data quality issues during high-velocity streams.
Low Latency Feature Access	Feature Serving Layer optimized for low latency and high throughput (e.g., using Redis).	Supports the sub-100 ms prediction budget required for real-time personalization.

The Feature Store is also vital for operational resilience during critical traffic events. High-velocity streams are vulnerable to data quality anomalies and potential corruption. The Feature Store's governance capabilities including data lineage tracking, feature versioning, and monitoring/observability features provide insights into feature usage, performance, and data quality. This serves as an essential safety layer; should performance degrade due to a feature inconsistency introduced during a traffic spike, the Feature Store provides the necessary clear audit trail and rapid rollback mechanism, minimizing service disruption in high-stakes environments.

4.3. Benchmark Performance: Meeting the Sub-100ms Guarantee

To validate the architectural choices necessary for real-time personalization, a detailed analysis of an ultra-low latency data pipeline was conducted, integrating state-of-the-art technical components.

Table 3: Ultra-Low Latency Pipeline Architecture and Performance Benchmarks

Pipeline Stage	Core Component(s)	ML Role / Function	Target Latency (Lx)	Achieved End-to-End Latency
Data Ingestion	Apache Kafka	High-frequency data input for real-time events.	< 20 ms	80 – 95 ms
Processing/Enrichment	Apache Flink	Real-time feature transformation and aggregation.	< 30 ms	
Feature Retrieval	Redis / Delta Lake	Low-latency feature lookup from online store.	Included in Processing/Inference	
Model Inference	NVIDIA Triton Inference Server	GPU-accelerated model prediction serving.	< 50 ms	
End-to-End Total Latency	Total System Architecture	Autonomous Decision/Prediction Loop	Target < 100 ms	80 – 95 ms

The core stack integrating Apache Kafka for data ingestion, Apache Flink for stream processing, Redis (a common high-speed feature store component) and Delta Lake for data storage, and the NVIDIA Triton Inference Server for model serving was tested rigorously. The pipeline consistently achieved end-to-end latency in the range of 80–95 milliseconds.

This consistent performance validates that the architecture selection is capable of meeting the critical sub-

100 ms requirement for the total personalization loop, covering ingestion, processing, feature lookup, and inference.

5. Adaptive Model Updates and Continuous Learning

5.1. Minimizing Data Staleness and Feature-Label Mismatch

Adaptive personalization requires not only fast inference but also features that are fresh and reflective of current user context. A critical vulnerability is feature-label mismatch, which results from the delayed propagation of updated representations (e.g., changes in item descriptions or popularity) into training or serving systems. This mismatch can significantly undermine calibration and reduce model effectiveness.

To maintain high-quality alignment between user behavior signals and item semantics, the system must aggressively minimize the end-to-end latency from content change to the feature's availability at the serving layer. This constraint confirms the critical role of streaming frameworks that support incremental updates, ensuring that if a product suddenly gains popularity during a flash sale, its representation is instantly updated, avoiding the use of stale features for subsequent recommendation requests. The capability to maintain real-time alignment is essential for continuous adaptation.

5.2. Multimodal Representation Learning for Enhanced Adaptivity

Modern product understanding in e-commerce has advanced beyond sparse, ID-based features to leverage rich multimodal content, including visual images and textual titles. These high-quality features are the foundation of adaptive models.

The MOON architecture, which utilizes a Multimodal Large Language Model (MLLM)-based method for product representation learning, proposes a sophisticated three-stage paradigm: “Pretraining, Post-training, and Application”. This framework sequentially fulfills two learning objectives: generative-model-based multimodal representation learning and downstream user visual preference modeling. Experimentally, this approach has demonstrated powerful capability for multimodal product content understanding, leading to an overall +20.00% improvement on Click-Through Rate (CTR) prediction in online A/B tests.

The quantified performance improvement (+20% CTR) validates the high reliance of adaptive ML models on the freshness and accuracy of these complex multimodal representations. This operational reality mandates that the AutoRecSys pipeline, particularly through its Feature Selection Search (Auto-FSS) mechanism, must be capable of efficiently selecting, managing, and retrieving these complex features while remaining strictly within the ultra-low latency budget established in Table 2.

5.3. Meta-Learning for Dynamic Model Adaptation

Traditional learning methods, frequently relying on classical algorithms like Stochastic Gradient Descent (SGD), often lack the dynamic adaptability necessary to handle rapidly changing data environments and diverse learning tasks efficiently.

The next major step in adaptive personalization involves integrating meta-learning (learning to learn) methods. Meta-learning allows the model to autonomously adapt, accelerating the continuous learning process. Research is actively progressing in scaling AutoML through multi-fidelity optimization, transfer learning, and the exploitation of user priors. This direction suggests moving beyond fully automated approaches toward a human-centered approach that combines expert knowledge with efficient automation to increase the efficiency of ML workflows.

This combined strategy provides the necessary mechanisms for rapid adaptation to new trends or cold starts that emerge instantly during peak shopping hours. The capacity to quickly train models on sequential, live events without requiring a full, disruptive retraining cycle is essential for maintaining personalization efficacy when data patterns change abruptly.

6. Conclusion

Adaptive personalization in high-traffic shopping events is only sustainable through the disciplined and synergistic integration of technological pillars. This report confirms that resilient, high-throughput model delivery is predicated on the convergence of three foundational components:

- Automated Machine Learning (AutoRecSys): Used for proactive architectural optimization, AutoRecSys enables significant efficiency gains, including a documented 25% Query per Second (QPS) increase in large-scale production ranking systems. Furthermore, its ability to automate the Embedding Dimension Search leads directly to resource optimization and reduction in embedding parameters. These gains reduce operational cost and critically minimize the model size.
- Ultra-Low Latency Architecture: This requires a streaming-first approach, utilizing resilient frameworks such as Apache Kafka and Apache Flink, combined with GPU-accelerated serving (e.g., NVIDIA Triton). This stack has been benchmarked to consistently deliver end-to-end performance within the critical 80–95 millisecond range.
- The Enterprise Feature Store: This platform provides essential data consistency, acting as the centralized source for feature governance, lineage, and versioning. Its core function is the mitigation of training/serving skew, which ensures model reliability and accurate predictions under dynamic and high-velocity load conditions.

By automating the historically labor-intensive phases of feature and model engineering, the integrated platform gains the speed, scale, and flexibility necessary to react instantaneously to rapid shifts in user behavior that define peak shopping events. The architecture detailed here moves ML deployment from a complex, manually intensive operation to a high-throughput, automated, and continuously adaptive system. This integrated framework is mandatory for maintaining competitive relevance and maximizing revenue potential in high-stakes, real-time digital commerce environments.

References

1. P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. (2015). Apache Flink™: Stream and Batch Processing in a Single Engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 28–38.
2. R. Zheng, L. Qu, B. Cui, Y. Shi, and H. Yin. (2021). AutoML for Deep Recommender Systems: A Survey. *ACM Transactions on Information Systems*, 1, 1, Article 1.
3. J. Hu et al. (2023). AutoML for Large Capacity Modeling of Meta's Ranking Systems. *arXiv preprint arXiv:2311.07870*.
4. K. Stankeviciute, M. Alaa, A., & M. van der Schaar. (2021). Conformal time-series forecasting. *European Journal of Operational Research*, 314, 111–121. URL: https://www.sciencedirect.com/science/article/pii/S0377221723006859.
5. G. Li, S. Wu, and C. Xiao. (2024). MOON: Generative MLLM-based Multimodal Representation Learning for E-commerce Product Understanding. *arXiv preprint arXiv:2508.11999v2*.
6. J. Smith. (2024). Designing Ultra-Low Latency Data Pipelines to Power ML Models for Real-Time Autonomous Decision Making. *ResearchGate*. URL: https://www.researchgate.net/publication/395477469_Designing_Ultra-Low_Latency_Data_Pipelines_to_Power_ML_Models_for_Real-_Time_Autonomous_Decision_Making.
7. F. Hutter et al. (2025). Automated Machine Learning (AutoML) review. *arXiv preprint arXiv:2505.18243v1*.
8. D. Liu et al. (2025). Multimodal Representation Learning Challenges and Solutions for E-commerce Recommendation Systems. *arXiv preprint arXiv:2511.11305v2*.
9. R. Zhang et al. (2025). Taxonomy of Challenges in Metaverse Systems. *arXiv preprint arXiv:2510.09621v1*.
10. S. Li et al. (2024). Human-Centered AutoML: Goals, Techniques, and Future Directions. *arXiv preprint arXiv:2406.03348v1*.
11. M. Zhai et al. (2024). A Study on Ranking-Based Loss Functions in Neural Architecture Search. *arXiv preprint arXiv:2506.05869*.
12. H. Lee. (2024). Comparison of AutoML Libraries and Recommendation Systems for E-commerce Tasks. *arXiv preprint arXiv:2402.04453*.
13. L. Cheng-Ju et al. (2020). Machine learning-based e-commerce platform repurchase customer prediction model. *Plos One*, 15(12):e0243105.
14. M. W. El-Said, and M. I. Aly. (2024). Real-Time Data Processing in E-Commerce: Concepts, Challenges, and Cloud Solutions. *International Journal of Advanced Engineering Technologies and Innovations*, 1(3), 298–307.
15. B. H. M. S. Waseem, H. U. Zaman, F. H. Khan. (2022). Real-Time Analytics: Concepts, Architectures, and ML... *IEEE Xplore*.
16. P. A. Alagumalai et al. (2022). An efficient anomaly detection and classification approach for rare events in audio data. *IEEE Xplore*.
17. Y. Xu, B. Cai, M. Ding. (2023). A Comprehensive Survey on Automated Machine Learning for Recommendations. *ResearchGate*. DOI: 10.1007/978-981-99-4328-9_13.
18. Hemish Prakashchandra Kapadia, Krunal Bharatbhai Thakkar. (2024). AI based user behavior prediction for web navigation, International Journal of Research and Analytical Reviews - (IJRAR), 11(3), 397-405, https://www.ijrar.org/papers/IJRAR24C2956.pdf