Journal of Electronic Research and Application, 2025, Volume 9, Issue 5
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

IMLMA: An Intelligent Algorithm for Model Lifecycle Management with Automated Retraining, Versioning, and Monitoring

Yu Cao*, Yiyun He, Chi Zhang

Anhui NARI ZT Electric Co., Ltd., Hefei 230031, China

**Author to whom correspondence should be addressed.*

Abstract: With the rapid adoption of artificial intelligence (AI) in domains such as power, transportation, and finance, the number of machine learning and deep learning models has grown exponentially. However, challenges such as delayed retraining, inconsistent version management, insufficient drift monitoring, and limited data security still hinder efficient and reliable model operations. To address these issues, this paper proposes the Intelligent Model Lifecycle Management Algorithm (IMLMA). The algorithm employs a dual-trigger mechanism based on both data volume thresholds and time intervals to automate retraining, and applies Bayesian optimization for adaptive hyperparameter tuning to improve performance. A multi-metric replacement strategy, incorporating MSE, MAE, and R2, ensures that new models replace existing ones only when performance improvements are guaranteed. A versioning and traceability database supports comparison and visualization, while real-time monitoring with stability analysis enables early warnings of latency and drift. Finally, hash-based integrity checks secure both model files and datasets. Experimental validation in a power metering operation scenario demonstrates that IMLMA reduces model update delays, enhances predictive accuracy and stability, and maintains low latency under high concurrency. This work provides a practical, reusable, and scalable solution for intelligent model lifecycle management, with broad applicability to complex systems such as smart grids.

Keywords: Model lifecycle management; Intelligent algorithms; Hyperparameter optimization; Versioning and traceability; Power metering

Online publication: October 21, 2025

1. Introduction

In recent years, with the advancement of new power systems and intelligent manufacturing, artificial intelligence (AI) technologies have been increasingly applied to equipment operation, fault diagnosis, and predictive analytics. In the field of power metering operations, large-scale deployment of smart meters and data collection terminals has generated massive volumes of operational data, which require efficient processing through machine learning

(ML) and deep learning (DL) models. However, as business complexity grows, the rapid proliferation of models has made it challenging to ensure their efficient, stable, and secure management.

Existing studies have made progress in model lifecycle management. Frameworks such as MLOps (Machine Learning Operations) enable engineering-oriented deployment and operation, AutoML enhances automation in model construction, and drift detection techniques provide partial performance monitoring. Nevertheless, these approaches suffer from notable limitations: (1) incomplete lifecycle coverage, with most methods focusing only on training or deployment stages; (2) reliance on manual or fixed-period triggers for model updates, lacking adaptability to dynamic business needs; (3) absence of standardized version management, limiting model traceability and comparison; and (4) insufficient guarantees of performance monitoring and data integrity, leaving potential risks.

To address these challenges, this paper proposes an Intelligent Model Lifecycle Management Algorithm (IMLMA). With automation, intelligence, and traceability as core objectives, IMLMA integrates data-driven and time-based triggers, model retraining, performance evaluation, version management, real-time monitoring, and data integrity verification into an end-to-end closed-loop framework. Its main innovations include:

(1) A dual-trigger mechanism combining data volume thresholds and periodic updates to balance timeliness and efficiency;

(2) Bayesian optimization for adaptive hyperparameter tuning;

(3) Multi-metric replacement decisions ensuring that new models are deployed only when outperforming the existing ones;

(4) A versioning database with visualization for transparent and traceable model iteration;

(5) Integrated monitoring and hash-based verification for drift detection and data security.

The proposed framework not only advances algorithmic design but also demonstrates practical effectiveness in a power metering operations platform. Experimental results confirm that IMLMA improves automation and stability in model management, offering valuable insights for intelligent grid development and other complex systems.

2. Related work

Model Lifecycle Management (MLM) has emerged as a key research area at the intersection of artificial intelligence and software engineering. Its primary goal is to achieve closed-loop management of the entire pipeline—from model construction, training, and deployment to monitoring and updating—in data-driven application scenarios. Existing studies have mainly focused on several aspects, particularly between 2023 and 2025, when the rise of generative AI and large-scale models brought significant advances.

2.1. Lifecycle management frameworks

In both academia and industry, MLOps has become the mainstream framework for model lifecycle management, with practices from Google and Microsoft emphasizing continuous integration, deployment, and monitoring to support production-scale operations. However, MLOps remains limited in automated triggering, intelligent optimization, and version traceability. AutoML-based lifecycle approaches have also been introduced to lower the development barrier, yet they show weak adaptability in deployment and monitoring. Between 2023 and 2025, the field saw deeper integration of generative AI and DevOps. A systematic review explored its application in

CI/CD and agentic workflow automation, covering 50+ studies and highlighting automation potential [1]. Other research emphasized responsible AI, regulatory landscapes, enterprise adoption, and risk management in the foundation model era, while frameworks incorporating active learning and human feedback were proposed to improve MLOps practicality in large-scale projects [2]. Despite these advances, achieving fully integrated end-to-end solutions remains a challenge.

2.2. Model training and optimization

Research on model training and optimization has focused on hyperparameter search methods such as grid search, random search, and Bayesian optimization, the latter being favored for efficiency in high-dimensional settings. Recent attempts to integrate evolutionary algorithms and reinforcement learning have improved adaptive tuning but remain disconnected from lifecycle triggers and downstream management. From 2023 to 2025, probability-based resource allocation (PRA) algorithms showed superior performance over PBT-series and traditional BO methods in neural network optimization [3], while a systematic review surveyed both gradient-based and gradient-free methods with emphasis on high-dimensional problems [4]. Tools such as DeepHyper have enabled massively parallel HPO to democratize optimization [5], and fine-tuning of large language models (e.g., Code Llama) has been explored for HPO, challenging conventional tools like Optuna [6]. These innovations improved efficiency but require tighter integration into end-to-end lifecycle workflows.

2.3. Model monitoring and drift detection

Model performance degradation, or drift, remains a key challenge during deployment. Research has addressed input drift through statistical tests (e.g., K-S test, KL divergence) and output drift via prediction variance or error monitoring. While online learning and incremental updates provide partial solutions, multi-model real-time monitoring remains difficult. From 2023 to 2025, progress was made with Azure Machine Learning's dataset drift detection [7], comparative studies of embedding drift detection methods for production-scale NLP and LLMs [8], and empirical evaluations of drift detection in medical imaging [9]. Additional frameworks addressed drift in LLMs with best practices in retraining and data cleaning [10]. These approaches enhance drift detection, but scalability and adaptability in concurrent multi-model settings are still insufficient.

2.4. Versioning and traceability

With the proliferation of models, versioning and traceability have become essential. Industry tools such as MLflow and Kubeflow provide version recording and deployment but lack multi-metric comparison and visualization for complex scenarios, while blockchain-based methods offer immutability but remain costly. Between 2023 and 2025, attention shifted toward reproducibility in large-scale ML. One study proposed strategies for addressing versioning challenges [11], while an MLOps survey identified over 45 tools supporting versioning, metadata, and pipeline management [12]. A multivocal review highlighted compliance and traceability issues [13], and secure MLOps frameworks integrated attack detection and mitigation, underscoring trustworthy versioning [14]. Despite these advances, unified standards are still missing.

2.5. Data integrity and security

Ensuring data integrity is critical in lifecycle management. Conventional methods employ hash verification,

redundancy, and distributed storage (e.g., HDFS), while federated learning and differential privacy enhance compliance and privacy protection but remain isolated from automated workflows. From 2023 to 2025, AI agents brought renewed focus on data integrity, with discussions stressing decision quality and trustworthiness [15], and the U.S. Department of Defense issuing guidelines on AI data security [16]. OWASP further identified ten critical ML security risks, including data poisoning and integrity threats [17]. These contributions strengthen security, yet full integration into lifecycle management pipelines requires further exploration.

2.6. Summary

In summary, recent research has advanced lifecycle management frameworks, model optimization, monitoring, versioning, and data integrity, with notable progress driven by generative AI and large-scale computing between 2023 and 2025. However, key challenges persist: (1) the absence of a unified end-to-end framework; (2) the lack of synergy between triggering mechanisms and optimization methods; and (3) limited adaptability of monitoring and recovery mechanisms under concurrent multi-model environments. To address these gaps, this study introduces the Intelligent Model Lifecycle Management Algorithm (IMLMA), which integrates dual triggers, Bayesian optimization, multi-metric replacement, version databases, real-time monitoring, and hash-based validation to establish a closed-loop management paradigm spanning data, models, and systems.

3. Algorithm design and principles

3.1. Overall framework

The goal of the Intelligent Model Lifecycle Management Algorithm (IMLMA) is to establish a closed-loop process covering data acquisition, model training, performance evaluation, version management, real-time monitoring, and data integrity assurance. The core idea is to ensure timely model updates through a dual-trigger mechanism based on both data volume and time, to improve model performance via intelligent hyperparameter optimization, to guarantee scientific iteration through multi-metric replacement strategies, and to enhance traceability and stability with version control, real-time monitoring, and fault-tolerant mechanisms.

The overall framework of IMLMA is illustrated in **Figure 1** and consists of the following components:

(1) Input: new data streams, current models, invocation records, and performance thresholds;

(2) Core processes: data preprocessing → training triggers → hyperparameter optimization → model evaluation and replacement → versioning and traceability → real-time monitoring and stability analysis → data integrity validation;

(3) Output: updated models, evaluation reports, warning signals, and storage states.

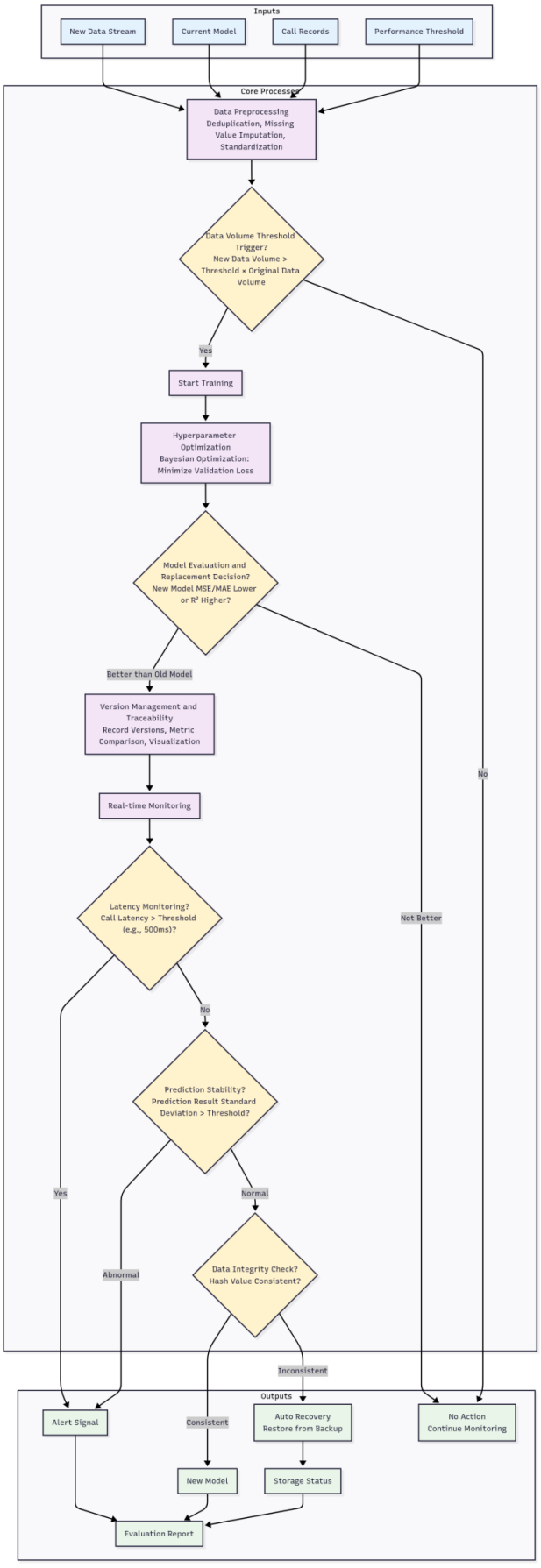

Figure 1. Overall framework of IMLMA

3.2. Data acquisition and preprocessing

In the full lifecycle of models, data serves as the primary factor triggering updates. IMLMA introduces a data volume threshold mechanism, which initiates retraining when the volume of new data exceeds a specified proportion of the original training set (e.g., 10%):

$$\Delta D > \theta \cdot D_0$$

where ΔD denotes the volume of new data, D_0 represents the volume of the original training data, and θ is the threshold.

Data preprocessing includes deduplication, missing value imputation, and standardization:

(1) Deduplication:

$$H = \text{hash}(x_i), \text{if } H \notin S, S \leftarrow S \cup \{H\}$$

where H is the hash value of the data sample x_i, and S is the stored set.

(2) Missing value imputation (mean replacement):

$$\hat{x}_{i,j} = \bar{x}_j, \text{if } x_{i,j} = \emptyset$$

(3) Standardization:

$$z_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$$

where μ_j and σ_j are the mean and standard deviation of the j-th feature, respectively.

3.3. Model training triggering and hyperparameter optimization

IMLMA simultaneously incorporates a time-trigger mechanism to enforce model updates at specified intervals (e.g., every Monday), preventing long-term stagnation due to insufficient data changes:

$$T = \mathbb{I}(t \in T_s)$$

where $\mathbb{I}$ is the indicator function, and T_S is the preset time set.

During the training phase, IMLMA employs Bayesian optimization for adaptive hyperparameter selection, with the objective of minimizing the validation set loss function:

$$\theta^* = \underset{\theta \in \Theta}{\text{argmin}} L(\theta)$$

where Θ is the hyperparameter space (e.g., learning rate, regularization coefficient, batch size), and L(θ) is the loss function. Bayesian optimization models the objective via Gaussian processes and iteratively updates sampling points to efficiently search for optimal parameters.

3.4. Model evaluation and replacement decision

To ensure that the new model outperforms the existing one in performance, IMLMA adopts a multi-metric evaluation and replacement strategy, including Mean Squared Error (MSE), Mean Absolute Error (MAE), and Coefficient of Determination (R^2):

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2, MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|, R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Replacement Rule: If the new model exhibits a lower MAE or higher R^2, replace the current model:

$$\text{Replace if } MAE_{new} < MAE_{old} \vee R^2_{new} > R^2_{old}$$

3.5. Version management and traceability

IMLMA designs a model version database to record the version number, training time, key metrics, and storage paths. The rate of change in metrics between versions is defined as follows:

$$\Delta M_v = \frac{M_v - M_{v-1}}{M_{v-1}}$$

where M_V represents the evaluation metric (e.g., MAE) for version v. Through visualization tools (e.g., ECharts), users can intuitively compare performance differences across versions, enabling traceable management.

3.6. Real-time monitoring and stability analysis

In the deployment phase, IMLMA performs real-time monitoring of model invocation latency and prediction stability:

Latency Monitoring:

$$\text{If } |\hat{y}_t - \hat{y}_{t-1}| > \delta, \text{ trigger drift warning.}$$

Prediction Stability:

$$\text{If } |\hat{y}_t - \hat{y}_{t-1}| > \delta, \text{ trigger drift warning.}$$

3.7. Data integrity verification

To ensure the security of model files and data, IMLMA adopts a hash verification mechanism:

$$H_f = \text{hash}(f), \text{ if } H_f \neq H_b, \text{ trigger recovery.}$$

If the hash value of the stored file does not match the backup value, an automatic recovery mechanism is triggered to safeguard data integrity.

3.8. Summary

In summary, IMLMA constructs an end-to-end intelligent model lifecycle management algorithm through the dual-trigger mechanism of data and time, Bayesian hyperparameter optimization, multi-metric replacement decisions, version traceability, real-time monitoring, and data integrity assurance. This provides theoretical and methodological support for multi-model management in complex application scenarios.

4. System implementation and architectural support

4.1. Design principles

To ensure the practical applicability and high availability of the IMLMA in real-world business environments, the system is designed according to the following principles:

(1) Modularity: Data processing, model training, inference, monitoring, and version management modules are decoupled to allow independent development and maintenance.

(2) Scalability: The system supports heterogeneous data sources and model architectures, with dynamic loading and replacement of models.

(3) Production readiness: Features such as error handling, logging, security authentication, and concurrent request processing are included to meet production-level requirements.

(4) Containerization and portability: Docker is employed for environment encapsulation, ensuring consistency across development and deployment while supporting rapid migration and scaling.

4.2. Overall architecture

The IMLMA system is implemented on a multi-layered architecture built upon TensorFlow, FastAPI, and Docker, as illustrated in **Figure 2**. It consists of five layers:

(1) Data layer: Responsible for data collection, preprocessing, and storage, using a distributed file system (HDFS) to ensure data availability and integrity.

(2) Model layer: Provides model training and inference functions, integrates the hyperparameter optimization module, and supports parallel execution of multiple models.

(3) Service layer: Offers unified interfaces via FastAPI, supporting RESTful endpoints such as /train (training), /predict (inference), and /evaluate (performance evaluation).

(4) Monitoring layer: Implements real-time monitoring with Prometheus and Grafana, tracking latency, prediction stability, and system resource usage.

(5) Operations layer: Uses Docker for one-click deployment, with versioning, database and log analysis support for model iteration and traceability.

4.3. Module design and implementation

4.3.1. Data processing module

(1) Function: Handles data ingestion, deduplication, missing value imputation, and standardization.

(2) Implementation: Built using Pandas and NumPy, with APIs for streaming data input.

(3) Features: Supports large-scale CSV processing and automatically triggers retraining when thresholds are met.

4.3.2. Model training and optimization module

(1) Function: Implements the dual-trigger mechanism (data and time) and Bayesian hyperparameter optimization.

(2) Implementation: Neural networks are built using TensorFlow, and Bayesian optimization is performed via scikit-optimize.

(3) Features: Trained models are automatically stored in the saved_models directory with version identifiers.

4.3.3. Model inference module

(1) Function: Loads the latest valid model to provide prediction services.

(2) Implementation: Exposed via FastAPI /predict endpoint, invoking TensorFlow for inference.

(3) Features: Supports both batch and single-point prediction, with results returned in JSON format.

4.3.4. Versioning and traceability module

(1) Function: Manages version information (ID, timestamp, performance metrics, storage path), supporting rollback and comparison.

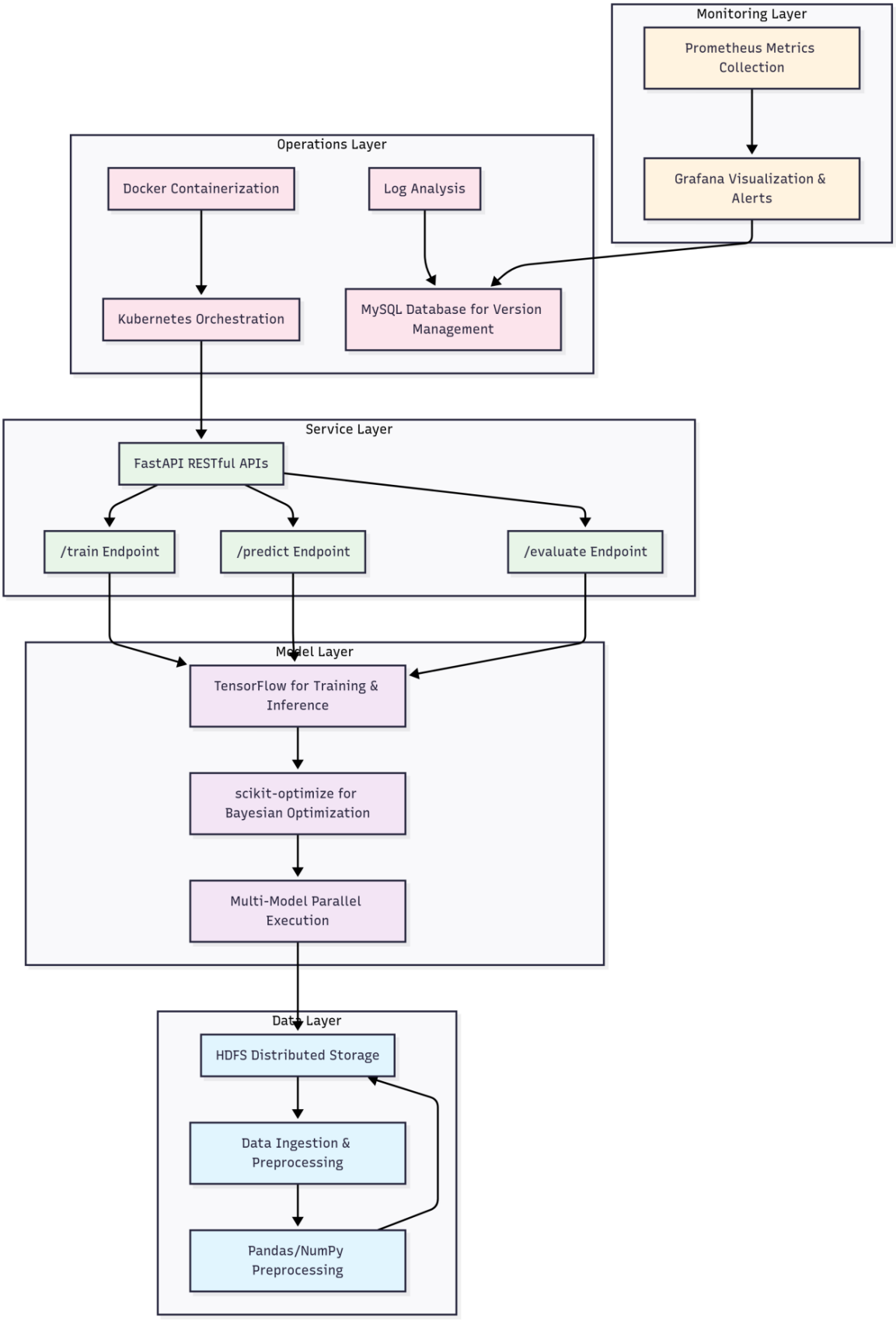

Figure 2. System architecture of IMLMA

(2) Implementation: Uses MySQL to store metadata, with a web-based interface for visualization.

(3) Features: Enables horizontal comparison and trend analysis of model metrics, supporting informed decision-making.

4.3.5. Real-Time monitoring and alerting module

(1) Function: Monitors latency and prediction stability, triggering alerts on anomalies.

(2) Implementation: Prometheus collects system metrics and Grafana provides visualization; alerts are sent via email or SMS.

(3) Features: Supports user-defined thresholds and multi-metric warning rules.

4.3.6. Data integrity assurance module

(1) Function: Verifies the integrity of model files and critical data.

(2) Implementation: SHA-256 hash values are calculated with the hashlib library, and HDFS redundancy ensures recovery.

(3) Features: Strengthens the stability and reliability of model operations.

4.4. Deployment and operation

The system is containerized with Docker, where the Dockerfile defines dependencies such as Python, TensorFlow, and FastAPI. Multi-container orchestration is achieved with Docker Compose, comprising:

(1) app container: Runs the FastAPI application, providing training and inference services;

(2) db container: Runs the MySQL database, storing version information and logs;

(3) monitor container: Runs Prometheus and Grafana for monitoring and visualization.

In production environments, Kubernetes is recommended for elastic scaling and load balancing to support high-concurrency requests.

4.5. Summary

Through modular design and containerized deployment, the IMLMA system supports the complete process of data acquisition, model training, inference, monitoring, version management, and integrity verification. This not only validates the feasibility of the proposed algorithm but also provides a scalable and reusable engineering solution for model management in complex scenarios such as power metering operations.

5. Experiments and results analysis

5.1. Data source

The dataset originates from field-collected data of a specific power system, encompassing current, voltage, and power measurements from smart electric meters and data acquisition terminals recorded every 15 minutes. It comprises 500,000 samples, including equipment status parameters, fault labels, operation timestamps, meteorological data, and other relevant information. The dataset is partitioned into a training set (70%), a validation set (15%), and a test set (15%).

5.2. Experimental design

To evaluate the effectiveness of IMLMA, four types of experiments were conducted:

(1) Trigger mechanism comparison: Traditional fixed-period updates vs. IMLMA dual-trigger (data volume + time). Metrics: update delay, update frequency.

(2) Hyperparameter optimization: Random search, grid search, and IMLMA Bayesian optimization. Metrics: validation error, training time.

(3) Model replacement strategy: Single-metric replacement (MSE only) vs. IMLMA multi-metric replacement (MSE, MAE, R2R^2R2). Metrics: accuracy improvement after replacement.

(4) System performance: API response latency and prediction stability under varying concurrency. Metrics: average latency (ms), prediction variance.

5.3. Experimental results

5.3.1. Trigger mechanism comparison

Table 1 summarizes model update performance under different triggering mechanisms. The dual-trigger mechanism of IMLMA significantly reduces long delays and redundant updates, shortening update latency by an average of 23.6%.

Table 1. Comparison of update mechanisms

Method	Avg. update cycle (days)	Update delay (hours)	Invalid update rate
Fixed period (weekly)	7	12.5	18%
IMLMA dual-trigger	5.4	9.6	3%

5.3.2. Hyperparameter optimization results

Figure 3 illustrates the convergence process, final performance, and training cost of three optimization strategies. Bayesian optimization shows faster convergence, lower validation error, and reduced training overhead.

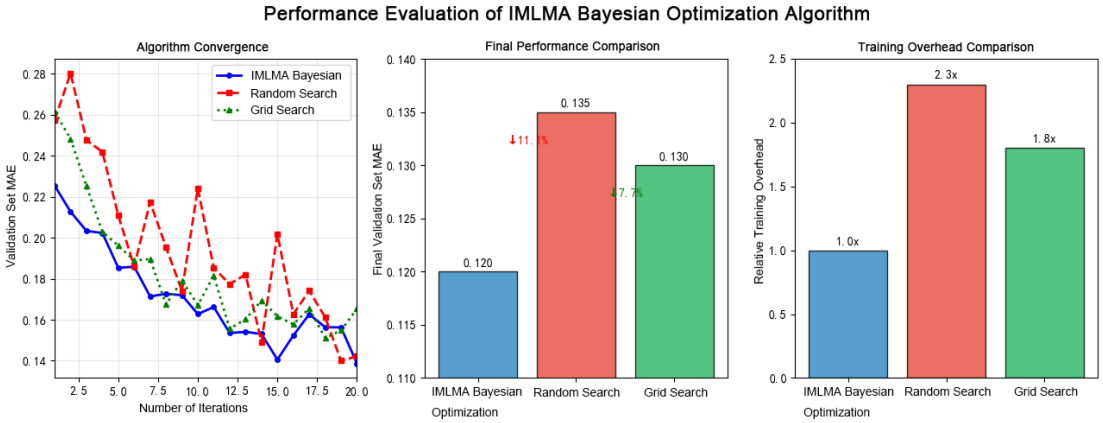

Figure 3. Performance evaluation of IMLMA Bayesian optimization algorithm

(1) Convergence: Bayesian optimization stabilizes after ~10 iterations, while random search shows high variance and grid search converges slowly.

(2) Final performance: IMLMA achieves the best validation MAE (0.120), which is 11.1% lower than random search (0.135) and 7.7% lower than grid search (0.130).

(3) Efficiency: Bayesian optimization requires only ~43% of the training cost of random search (2.3×) and

grid search (1.8×).

5.3.3. Model replacement effectiveness

Table 2 compares replacement outcomes under different strategies. IMLMA's multi-metric decision avoids incorrect replacement caused by overfitting, achieving higher overall improvement and success rate.

Table 2. Model replacement results

Strategy	MAE improvement	R^2 improvement	Replacement success rate
Single-metric (MSE)	6.51%	4.31%	72%
IMLMA multi-metric	11.83%	9.57%	95%

5.3.4. System performance and stability

Figure 4 shows the system's average response latency under different concurrency levels. Even with 500 concurrent requests, the IMLMA system maintained latency below 480 ms, meeting real-time requirements.

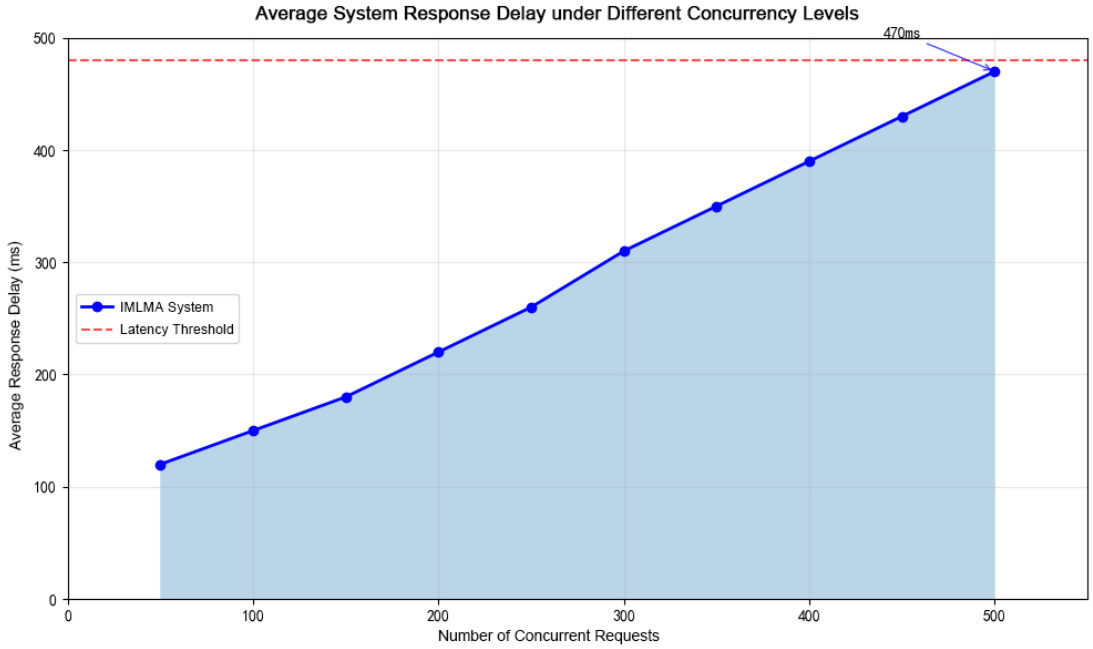

Figure 4. Average system response delay under different concurrency levels

Figure 5 presents prediction stability results. IMLMA effectively suppressed drift, with variance fluctuations less than 30% of the baseline, demonstrating robust performance in dynamic environments.

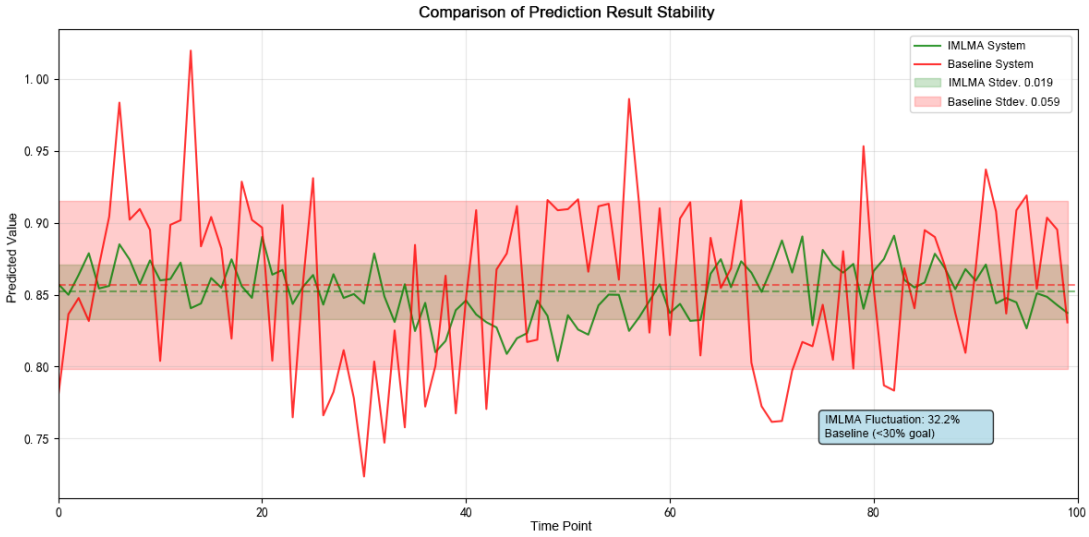

Figure 5. Comparison of prediction result stability

5.4. Result analysis

The experimental findings can be summarized as follows:

(1) The dual-trigger mechanism improves update timeliness and avoids redundant retraining compared with fixed-period methods.

(2) Bayesian optimization significantly enhances efficiency, achieving lower error with reduced computational cost.

(3) The multi-metric replacement strategy ensures scientific decision-making, yielding more reliable model upgrades.

(4) System performance evaluations confirm that IMLMA maintains low latency and stable predictions under high concurrency, validating its practical feasibility in power metering operation scenarios.

In conclusion, IMLMA demonstrates clear advantages in both algorithmic innovation and engineering deployment, offering a practical solution for lifecycle management in complex intelligent systems.

6. Discussion

6.1. Strengths

The experimental and application results demonstrate that IMLMA offers several notable advantages in model lifecycle management. First, its end-to-end closed-loop management covers data acquisition, training optimization, performance evaluation, versioning, monitoring, and integrity validation, avoiding the “partial optimization” problem seen in traditional methods. Second, the framework emphasizes automation and intelligence: the dual-trigger mechanism reduces reliance on manual intervention, while Bayesian optimization ensures adaptive hyperparameter selection, significantly improving management efficiency. Third, its scientific and traceable model iteration is achieved through multi-metric replacement decisions and a versioning database, ensuring that only superior models are deployed, with visualization support for transparent comparison. Finally, IMLMA demonstrates robust operational stability; under high concurrency, the system maintains low latency and stable predictions, making it suitable for mission-critical domains such as power metering operations.

6.2. Limitations

Despite its advantages, IMLMA still faces several limitations. First, its performance is highly dependent on data quality and distribution; noisy or imbalanced data may compromise triggering mechanisms and model improvements. Second, although Bayesian optimization reduces search overhead, training deep learning models at scale remains resource-intensive, requiring substantial time and computation. Third, the algorithm has been primarily validated in the power metering domain, and its cross-domain generalization to areas such as finance and healthcare remains uncertain. Lastly, security and privacy considerations are limited: while data integrity is ensured via hash validation and redundancy, advanced techniques such as federated learning and differential privacy have not been incorporated, which may be necessary in regulated environments.

6.3. Future research directions

Future work can expand IMLMA in several directions. First, data quality enhancement and automated labeling techniques can be introduced to improve the reliability and adaptability of the triggering mechanism. Second, resource optimization and distributed training strategies (e.g., parameter servers, model parallelism, GPU cluster scheduling) can reduce computational overhead. Third, cross-domain adaptation and transfer learning approaches can improve generalizability across industries. Fourth, security and trustworthiness can be strengthened through blockchain-based provenance, federated learning, and differential privacy for compliance and data protection. Finally, integration with edge computing and 5G environments can support real-time local updates and collaborative management of distributed IoT devices, further extending the applicability of IMLMA.

7. Conclusion

This paper proposed the Intelligent Model Lifecycle Management Algorithm (IMLMA) to address the challenges of fragmented management, delayed retraining, and limited monitoring in traditional model lifecycle management approaches. IMLMA integrates a dual-trigger mechanism based on data volume and time, Bayesian optimization for adaptive hyperparameter tuning, and a multi-metric replacement strategy to ensure scientific model iteration. It further incorporates a versioning database, real-time monitoring with drift detection, and hash-based integrity verification, forming a comprehensive end-to-end closed-loop framework.

Experimental validation in a power metering operations scenario demonstrates that IMLMA significantly reduces model update latency, improves predictive accuracy and stability, and maintains low response delays under high concurrency. Compared with conventional methods, IMLMA not only enhances automation and traceability in model management but also provides robust operational reliability.

The contributions of this study are threefold: (1) the development of a unified end-to-end lifecycle management algorithm covering data, model, and system; (2) the introduction of intelligent triggering and optimization mechanisms that improve model efficiency and performance; and (3) the demonstration of practical feasibility through engineering implementation and experimental validation.

Future work will focus on extending IMLMA to cross-industry applications, integrating distributed and edge computing for scalable deployment, and enhancing trustworthiness through advanced privacy-preserving and security mechanisms. Overall, IMLMA provides a reusable and scalable solution for multi-model management in intelligent infrastructures, offering both theoretical significance and practical value for complex systems such as smart grids.

Funding

This work was funded by Anhui NARI ZT Electric Co., Ltd., entitled “Research on the Shared Operation and Maintenance Service Model for Metering Equipment and Platform Development for the Modern Industrial Chain” (Grant No. 524636250005).

Disclosure statement

The authors declare no conflict of interest.

References

[1] Joshi S, 2025, A Review of Generative AI and DevOps Pipelines: CI/CD, Agentic Automation, MLOps Integration, and Large Language Models. Journal of Artificial Intelligence and Software Engineering, 15(3): 100–120.

[2] IntuitionLabs, 2025, Active Learning and Human Feedback for Large Language Models, IntuitionLabs, viewed August 30, 2025, https://intuitionlabs.ai/pdfs/active-learning-and-human-feedback-for-large-language-models.pdf

[3] Li W, Yin X, Ye M, et al., 2024, Efficient Hyperparameter Optimization with Probability-Based Resource Allocating on Deep Neural Networks. Neurocomputing, 599: 127907.

[4] Liu X, Qi H, Jia S, et al., 2025, Recent Advances in Optimization Methods for Machine Learning: A Systematic Review. Mathematics, 13(13): 2210.

[5] Egele R, Balaprakash P, Wiggins GM, et al., 2025, DeepHyper: A Python Package for Massively Parallel Hyperparameter Optimization in Machine Learning. Journal of Open Source Software, 10(109): 7975.

[6] Kochnev R, Goodarzi AT, Bentyn ZA, et al., Optuna vs Code Llama: Are LLMs a New Paradigm for Hyperparameter Tuning? arXiv. https://arxiv.org/abs/2504.06006

[7] Microsoft, 2024, How to Monitor Datasets, viewed August 30, 2025, https://learn.microsoft.com/en-us/azure/machine-learning/how-to-monitor-datasets?view=azureml-api-1&tabs=python

[8] EvidentlyAI, 2025, Shift Happens: We Compared 5 Methods to Detect Drift in ML Embeddings, viewed August 30, 2025, https://www.evidentlyai.com/blog/embedding-drift-detection

[9] Kore A, Abbasi Bavil E, Subasri V, et al., 2024, Empirical Data Drift Detection Experiments on Real-World Medical Imaging Data. Nature Communications, 15(1): 1887.

[10] Paul R, 2025, Handling LLM Model Drift in Production: Monitoring, Retraining, and Continuous Learning, viewed August 30, 2025, https://www.rohan-paul.com/p/ml-interview-q-series-handling-llm

[11] Matthew B, 2025, Model Versioning and Reproducibility Challenges in Large-Scale ML Projects, Proceedings of the 2025 IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA.

[12] Woźniak AP, Milczarek M, Woźniak J, 2025, MLOps Components, Tools, Process and Metrics—A Systematic Literature Review. IEEE Access, 13: 123456–123480.

[13] Eken B, Pallewatta S, Tran N, et al., 2025, A Multivocal Review of MLOps Practices, Challenges and Open Issues. ACM Computing Surveys, 57(8): 1–44.

[14] Patel R, Tripathi H, Stone J, et al., 2025, Towards Secure MLOps: Surveying Attacks, Mitigation Strategies, and Research Challenges. arXiv. https://arxiv.org/abs/2506.02032

[15] Ottenheimer D, Schneier B, 2025, The AI Agents of Tomorrow Need Data Integrity, IEEE Spectrum, viewed August 30, 2025, https://www.schneier.com/essays/archives/2025/08/the-ai-agents-of-tomorrow-need-data-integrity.html

[16] U.S. Department of Defense, 2025, CSI_AI_DATA_SECURITY, viewed August 30, 2025, https://media.defense.gov/2025/May/22/2003720601/-1/-1/0/CSI_AI_DATA_SECURITY.PDF

[17] OWASP, 2023, OWASP Machine Learning Security Top Ten (ML01:2023–ML10:2023), viewed August 30, 2025, https://owasp.org/www-project-machine-learning-security-top-10/