



Analyzing Request Volatility in Cloud-Based Machine Learning: A Review of Alibaba's Machine Learning as a Service Platform

Dr. Naresh Thoutam¹, Sanika Bhusari², Snehal Thakur²,
Disha Gangurde², Chaitanya Mahajan²

¹Assistant Professor, Department of Computer Engineering, Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India

²Department of Computer Engineering, Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India
naresh.thoutam@sitrc.org, sbhusari6105@gmail.com, snehalthakur2909@gmail.com, dishagangurde2@gmail.com,
chaitanyamahajan1018@gmail.com

Corresponding author- Dr. Naresh Thoutam

DOI: [10.5281/zenodo.23482053](https://doi.org/10.5281/zenodo.23482053)

Abstract: Cloud ML (MLaaS) services need to schedule mixed training, pre-processing and inference requests arriving by different processes for different users, models, tasks and time scales. This paper reviews the literature on Alibaba's Platform for AI (PAI), large GPU clusters, traces of cloud storage, serverless workloads, traffic self-similarity and elastic deep-learning serving, in order to clarify how request volatility should be quantified and translated into operation. The literature finds that simple means and memoryless inter-arrival distributions are inadequate: cloud ML traffic can combine abrupt bursts, long-range dependence, cross-layer amplification, resource-demand skewness and colocated job interference. The paper therefore develops a reproducible analytical framework based on multi-resolution aggregation, coefficient of variation, Fano factor, burstiness index, autocorrelation, variance-time analysis, Hurst calculation, peak-episode detection and workload-resource correlation. These metrics are related to a practical web-based system based on trace ingestion, pre-processing, MySQL store, a volatility analytics engine, dashboards, report generation and inference-based resource recommendations, with no performance gains claimed in code, data or procedures. The key conclusion is that volatility must be jointly assessed across hierarchy, time scale and resource type prior to choosing autoscalers or schedulers.

Keywords: Alibaba PAI, cloud computing, machine learning as a service, request volatility, workload characterization, burstiness, self-similarity, long-range dependence, GPU scheduling, resource utilization.

1 INTRODUCTION

MLas is shifting the model development environment from isolated workstations to collaborative cloud infrastructure offering elastic compute, accelerators, storage and managed pipelines. The trend was helped by normalization of distributed execution on heterogeneous hardware [22], [23] and production platforms increased to multi-tenant serving of preprocessing, training, tuning, deployment, and inference. Alibaba's PAI [1], [25] exemplifies this trend: requests ascend through multiple scheduling tiers and may grow from a single submitted job into a multitude of tasks and instances fighting over CPUs, GPUs, memory, network bandwidth, and storage.

The key challenge is not just the scale, but the distribution of arrivals. Arrivals for a platform can have benign daily averages, but also have surges at minute timescales, bursty correlation patterns, or extended periods of high demand [12]. Such time-

series behaviors break distribution-agnostic capacity plans, leading to queueing, scheduling latency, and missed deadlines, or alternatively, "gold-plated" excess accelerators. Like GPU-cluster workloads, serverless traces have time series with time-varying heterogeneity and demand skew [9], [10], [13], [21], as well as sensitivity to scheduling choice [21]. Serverless arrivals are also both sparse and tightly packed [20], also motivating distribution-aware analyses.

This report examines the provided literature using three questions: (1) how is volatility defined and estimated, (2) how does it vary by MLaaS hierarchy level and time scale, and (3) how can the found evidence inform storage, scheduling, elastic scaling, and monitoring? This document contributes (1) a unified taxonomy for short term burstiness and long-memory, (2) a measurement procedure to prevent the aggregation problem, (3) a- cness on the implementation architecture for the

academic community, and (4) an conservative translation of statistics into practice.

II. LITERATURE REVIEW AND WORKLOAD FOUNDATIONS

A. Cloud ML Workload Characterization

Zou et al. place the request volatility with his or her particular job, process, instance, and machine levels [1]. The relationship is important because a number of jobs diverge into a lot of executable units that local whips are absorbed or energized by batching, and retries, synchronization hurdles, and placement. Its tie-in research of I/O whiplash demonstrates that request arrivals and information motion should be considered in conjunction because storage pressure might be exerted after compute admission somewhat than with submission time [4]. Production GPU work clusters expand on this picture. Weng et al. study MLaaS scheduling in heterogenous clusters [9]; Li et al. define AI workload types on large accelerator systems [10]; Hu et al. define characterization and modeling for GPU datacenter workloads [13]; and Jeon et al. analyze multi-tenant DNN training clusters [21]. These works, together, suggest why it makes sense to keep properties like the required duration, number of accelerators, model family, and retry pattern for each job, rather than collapsing all requests into a single count series. HPC I/O studies also show that ML phases have unique access patterns [14].

B. Burstiness, Self-Similarity, and Long-Range Dependence

An extensive body of network-traffic research proved that demand represented as an aggregate can still be bursty over long time scales. Leland et al. explained why self-similar traffic differs radically from Poisson traffic [24]. Subsequent work draws on this lineage to analyze industrial Internet traffic [2], cloud-server traces [16], Open Flow saturation detection [18], and burst scheduling for data-center networks [15]. For MLaaS, this paternity suggests that if the traffic is long-range-dependent, then not only short observation windows but also independent arrivals models will underestimate the true persistent congestion risk.

Thus, a volatility profile covers no fewer than four distinct aspects: dispersion of the distribution around the mean, an auto-regression component that questions whether the high values are aggregated in time, the tail component measuring extreme outlier intensity, and a cross-resource coupling component posing whether the marked list of surges results in GPU, memory, network, or I/O load. Any sound review would never rely on a single statistical device.

C. Serving, Multiplexing, and Interference

Alpa Serve employs statistical multiplexing with model parallelism to enhance deep-learning serving [6]. MIGER combines multi-instance GPU and multi-process service mechanisms [3]. Interference-aware middleware and Horus illustrate that placement decisions should be informed by performance interactions between colocated jobs [7], [11]. Elastic scaling work demonstrates importance of quickly yet reliably tuning capacity [17]. These works are pertinent because (1) volatility can be translated directly into concurrency and contention, (2) predictability of bursts, co-variation of resources, and (3) latency of a scale operation are crucial factors in the benefits of shifting resources.

III. RESEARCH GAP AND REVIEW FRAMEWORK

Prior work optimizes only one layer-GPU scheduling, model serving, I/O, or traffic prediction: the system demands a harmonized end-to-end perspective. Three common gaps.1 Request counts are usually reported at one sample window, when the output flips with coarser aggregation.2 Job arrivals can be decoupled from downstream instance and resource decisions.3 Dashboards show peaks but not accompanying data coverage, error, or baseline sensitivity.

This review structure closes these gaps by considering the volatility as a measurement chain. $X_t^{(l, \Delta)}$ for requests at level l in interval Δ . Valid comparisons are only made when level, interval, timezone, missing-data policy, and request semantics are all held constant. For each series, the analysis considers central tendency, dispersion, tails, temporal dependence, peak episodes, and resource correlations. Then findings are tested across several Δ 's and request types before making recommendations.

IV. VOLATILITY ANALYSIS ARCHITECTURE

The architecture of Figure 1 decomposes the synopsis into six auditable layers. The user submits an Alibaba ML request trace via a secure portal. Ingestion retains its source and logs in schema, provenance and observation interval. Preprocessing aligns timestamps, de-duplicates data, annotates for missingness and generates job, task, instance and machine tables. A MySQL store partitions fixed raw metadata from variable time-series features. Analytics engine derives volatility metrics and identifies peak episodes. Dashboard makes visible trends and assumptions, while reports drive recommendations by revealing the evidence.

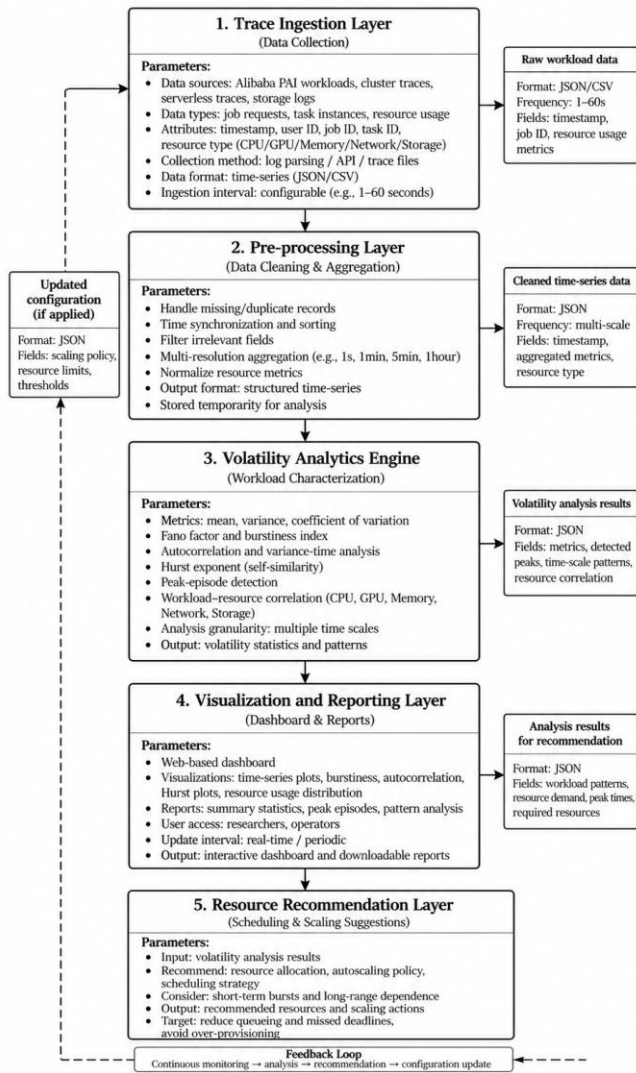


Figure 1: Proposed cloud ML request-volatility analysis architecture.

A. Data Integrity and Hierarchical Semantics

A trace row must not be assumed to equal an independent user request. A submitted job may create tasks, and tasks may create replicated or distributed instances. Retries may repeat logical work. The data model should retain stable identifiers and parent-child relationships so analysts can distinguish arrival volatility from execution fan-out. Mobile file-usage and block-storage studies further illustrate that application behavior and storage behavior can diverge [8], [19].

B. Dashboard and Recommendation Boundaries

The dashboard should show raw counts, smoothed context, distribution summaries, detected peaks, and resource overlays. Recommendations remain conditional. For example, a short GPU-bound burst may justify warm capacity or batching; a long correlated episode may justify predictive scaling; and an I/O-heavy surge may require caching or staging rather than

more GPUs. The system should never infer causal benefit from correlation alone.

V. ANALYTICAL METHODOLOGY

A. Multi-Resolution Time-Series Construction

After validation, events are binned at several intervals such as one minute, five minutes, one hour, and one day. Empty bins are represented explicitly rather than removed. For a series $x_1, \dots, x_n$, the coefficient of variation is

$$CV = \sigma_x / \mu_x, \quad (1)$$

where μ_x and σ_x are the sample mean and standard deviation. CV is scale-normalized but unstable when the mean is near zero. The Fano factor $F = \sigma_x^2 / \mu_x$ compares variance with the mean, while the bounded burstiness index

$$B = (\sigma_x - \mu_x) / (\sigma_x + \mu_x) \quad (2)$$

summarizes relative dispersion when both terms are well-defined. These measures should be accompanied by median, interquartile range, 95th and 99th percentiles, maximum-to-median ratio, and the fraction of zero intervals.

B. Temporal Dependence and Self-Similarity

Autocorrelation at lag k tests whether high or low demand persists. Variance-time plots aggregate x_t into non-overlapping blocks and inspects how variance changes with scale. A slowly decaying autocorrelation function and Hurst estimate H greater than 0.5 are consistent with long-range dependence, but estimators are sensitive to trends, seasonality, finite samples, and nonstationarity [16], [24]. Agreement across estimators and segmented windows is therefore more credible than a single H value.

C. Peak Episodes and Cross-Layer Propagation

A peak episode can be defined by a transparent threshold, such as the 95th percentile of a stable baseline window, followed by merging adjacent exceedances separated by a short gap. Each episode is described by start time, duration, peak amplitude, excess request volume, request mix, concurrency, and resource pressure. Cross-correlation between hierarchy levels tests whether job bursts precede instance or machine demand. Lagged association is useful for capacity lead time, but it does not establish causality.

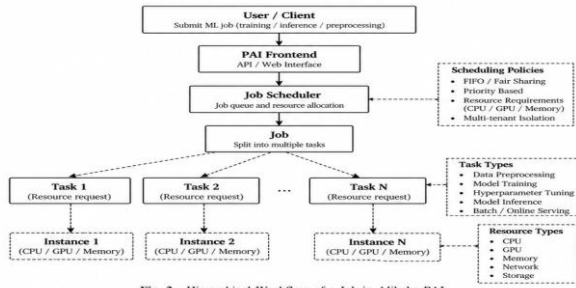


Fig. 2. Hierarchical Workflow of a Job in Alibaba PAI.

Figure 2: Reproducible multi-scale workload-analysis workflow.

D. Reproducibility and Sensitivity Analysis

Results should be recomputed under alternative bin sizes, missing-value policies, percentile thresholds, and subgroup definitions. Exact software versions, trace hashes, SQL transformations, and analysis parameters should be recorded. This prevents a dashboard setting from becoming an undocumented analytical choice and makes comparisons with future Alibaba PAI traces defensible.

VI. COMPARATIVE SYNTHESIS

The literature can be organized by the operational question each source answers. Trace-characterization studies [1], [4], [8]-[10], [13], [14], [19]-[21] identify workload distributions and hierarchy. Traffic-dependence studies [2], [15], [16], [18], [24] supply methods for persistence and burst analysis. Scheduling and serving systems [3], [5]-[7], [11], [17] show how observations can inform placement, multiplexing, survivability, and elasticity. Framework and platform sources [22], [23], [25] define the execution environment.

Table 1: Literature synthesis by analytical role

Evidence group	Representative refs.	Primary signal	Review implication
Alibaba MLaaS	[1], [4], [25]	Cross-layer requests and I/O	Preserve hierarchy and align compute with I/O
GPU clusters	[9], [10], [13], [21]	Job shape, skew, concurrency	Stratify by resource demand and duration
Dependence	[2], [16], [24]	Self-similarity and long memory	Analyze multiple time scales and persistence
Cloud traces	[8], [19], [20]	Storage, files, functions	Do not generalize one subsystem to another
Scheduling	[3], [5]-[7], [11], [17]	Placement, multiplexing, elasticity	Match control latency to burst duration

A. Implications for Scheduling

If burst duration is smaller than provisioning delay reactive autoscaling will be too slow; admission control, queued batching, warm pools, statistical multiplexing will matter more [6], [17]. If bursts last many hours, we believe forecasting and scheduled capacity make sense. If volatility can be isolated to one tenant or model class, fairness and quotas may be better than platform-level scaling, and interference-aware schedulers should take into account both magnitude and colocation [7], [11].

B. Implications for Storage and Network

Compute requests will cause checkpoint lag, dataset reads, parameters, and model writes. The Alibaba I/O Workload [4], Storage Comparison [8], and HPC ML I/O Workloads [14] work show that compute and storage telemetry can be aligned but should be broken apart when you analyze it. Resource recommendations should determine which subsystem is bound; otherwise, increasing the GPU count will not affect the I/O bottleneck.

C. Prediction versus Description

Forecasting can be informative only once a stable descriptive baseline has been created. We motivate learned models by GRU-based survivable scheduling [5], GPU-workload prediction [13] and time-series prediction workloads [12], but evaluation must require chronological splits, reporting of calibration and tail error, and comparisons against seasonal or persistence baselines. The academic implementation should focus on clear descriptive measures before a complex predictor.

VII. EVALUATION FRAMEWORK

A. Analytical Validity

Testing should address if the pipeline preserves counts, reproduces extracted series, and captures what we already understand to be artificial perturbations. Tests for duplicates, gaps, timestamp shifts, single spikes, plateau, and bursting over time can all be included. The invariants to be verified include raw and grouped totals are equal after documented exclusions, and burst insertion matches estimated dispersion.

B. System Performance

Time required to ingest, preprocess, and the queries/query latency/dashboard render time, size of the database, memory usage should be measured as functions of trace size. These measurements are on implementation feasibility; they should not be counted as improvements over Alibaba's production platform. Concurrent-user experiments should be able to separate the interface latency from analytics time.

C. Decision Utility

A good recommendation can be traced to the evidence and can be robust under reasonable changes in parameters. Candidate scenarios are a short isolated burst, a daily recurring peak, a long-memory series, a GPU-heavy model mix, and an I/O-dominant workload. The reviewers need to confirm that the system recommends different mitigations and shows the rationale for each.

VIII. IMPLICATIONS, LIMITATIONS, AND FUTURE DIRECTIONS

A multi-scale perspective gives you a better handle on capacity planning – it isolates momentary spikes from persistent requirements, and it isolates pressure on the platform as a whole from local fan-out. It helps SDG 9 by urging the building of a more efficient digital infrastructure. It can bridge classwork involving cloud traces, database design, statistics, and resource management – without expecting classroom access to Alibaba's internal scheduler.

Interpreting these results faces several challenges. Public or provided traces may be de-identified, sampled, or incomplete, and not all are from a single operating period. Request counts reveal nothing about end-user intent, service level objectives, or causative scheduling decisions. Hurst estimates can be biased by trends and periodicity. Resource correlations can be a consequence of shared workload phases rather than dependence. And, a MySQL-backed academic prototype cannot mimic PAI's [25] scale or control plane.

Open problems include (i) testing for change-point detection, (ii) assessing online uncertainty estimates, (iii) determining multivariate attribution of peak, and (iv) performing privacy-preserving cross-tenant aggregation. Trace-driven simulators can be used to compare reactive scaling, predictive scaling, batching, and interference-aware placement under the same sequence of arrivals. A subsequent implementation could include explainable forecasting, following validation of the descriptive pipeline and data-quality controls.

A. Threats to Validity

Construct validity is bounded by what the trace calls a request. Submission records, scheduler events, container launches, RPCs, storage operations define different phenomena. The analysis should release a data dictionary, a library to read the traces, and various ways to aggregate time so different sources of data are comparable. Internal validity is limited by clock drift, duplicate retries, sampling, and gaps that look just like real quiet time. External validity is limited by publication of results from one provider, in one region, on one generation of hardware, with one product mix, which we do not expect to generalize to other provider regions. Conclusion validity suffers when we use thousands of correlated time bins as

evidence in numerous hypothesis tests.⁹ Block bootstrap confidence intervals and sensitivity analysis can ameliorate, but not eliminate, these risks.

B. Implementation Roadmap

A staged implementation makes the prototype testable. Stage one brings in a bounded trace, checks the schema and timestamps, and emits reconciled counts at all hierarchy levels stage two persists normalized events and materialized time bins in MySQL with indexes on time, request type, and parent id stage three implements descriptive statistics and peak episodes with automated tests stage four overlays resource telemetry and introduces conditional recommendations stage five unifies all self-service elements and multi-tenancy features. Forecasting and live ingestion should be add-on optional features until the batch pipeline can be reproduced, and the dashboard exposes all transformations.

C. Responsible Operational Use

Volatility analytics can regulate tenant quota and job priority governance. An alert would show the observation window, threshold, affected workload class, and confidence / sensitivity range. Recommendations should be advisory unless verified against service-level objectives and fairness constraints. Tenant labels should be pseudonymised, fine-grained traces should be kept only as long as needed, and aggregate views should be safeguarded against reidentification attacks. Operators should have an audit trail of what dataset and parameters created each report. These controls elevate a technically correct graph into a defensible decision tool.

D. Research Questions for Trace-Driven Experiments

The following experiments will address four specific issues. How long are the aggregation periods needed for burst rankings of workload classes to converge? Are bursts of the same class similar across workloads? To what extent do peaks at the job level carry through to tasks, instances, machines, and I/O? What combination of historical observations, calendar factors and workload composition helps us detect bursts of all durations while avoiding false positives? Which mitigation, warm capacity, batching, predictive scaling, interference-aware placement scenario best balances latency and utilization, for each burst-duration class? The experiments will require chronological evaluation, direct comparison of identical trace segments for each scenario and separate presentation of means and tails.

IX. CONCLUSION

Request Volatility in Cloud Machine Learning Request volatility is not one number. It is spread, it is tails, it is dependence, it is hierarchy, it is coupling. In the clouds in this survey (Alibaba PAI, GPU-cluster, storage, serverless,

networking, scheduling), averages and independent arrivals do not suffice. Reasonable analysis preserves job-task-instance relationships, looks at more than one time scale, measures heavy tails and autocorrelation, and links demand with resource telemetry.

This architecture implements this conclusion through auditable ingestion, preprocessing, MySQL storage, volatility analytics, visualization and evidence-bounded recommendation. It is valuable for reproducible characterization and improvement of scheduling decisions, not as a technique to claim production performance. It provides a solid basis on which to progress trace-driven experiments and implementation evaluation.

X REFERENCE

- [1] Q. Zou, Y. Deng, Y. Zhu, Y. Zhou, J. Cai, S. He, and L. Ge, "Analyzing request volatility in cloud-based machine learning: Insights from Alibaba's Machine Learning as a Service platform," *IEEE Trans. Netw. Service Manag.*, vol. 23, pp. 1449-1464, 2026.
- [2] Q. Li, S. Wang, Y. Liu, H. Long, and J. Jiang, "Traffic self-similarity analysis and application of Industrial Internet," *Wireless Netw.*, vol. 30, pp. 3571-3585, 2024.
- [3] B. Zhang, S. Li, and Z. Li, "MIGER: Integrating multi-instance GPU and multi-process service for deep learning clusters," in *Proc. 53rd Int. Conf. Parallel Process. (ICPP)*, 2024.
- [4] Q. Zou, Y. Deng, Y. Zhu, Y. Zhou, J. Cai, and S. He, "Dissecting I/O burstiness in machine learning cloud platform: A case study on Alibaba's MLaaS," in *Proc. 38th IEEE Symp. Mass Storage Syst. Technol. (MSST)*, 2024.
- [5] T. Yu et al., "Multi-Visual-GRU-based survivable computing power scheduling in metro optical networks," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 1, pp. 1302-1315, 2024.
- [6] Z. Li et al., "AlpaServe: Statistical multiplexing with model parallelism for deep learning serving," in *Proc. 17th USENIX Symp. Oper. Syst. Design Implementation (OSDI)*, 2023.
- [7] W. Chen et al., "Interference-aware multiplexing for deep learning in GPU clusters: A middleware approach," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, 2023.
- [8] J. Li, Q. Wang, P. Lee, and C. Shi, "An in-depth comparative analysis of cloud block storage workloads: Findings and implications," *ACM Trans. Storage*, vol. 19, no. 2, 2023.
- [9] Q. Weng et al., "MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters," in *Proc. 19th USENIX Symp. Networked Syst. Design Implementation (NSDI)*, 2022.
- [10] B. Li et al., "AI-enabling workloads on large-scale GPU-accelerated systems: Characterization, opportunities, and implications," in *Proc. IEEE HPCA*, 2022.
- [11] G. Yeung, D. Borowiec, R. Yang, A. Friday, R. Harper, and P. Garraghan, "Horus: Interference-aware and prediction-based scheduling in deep learning systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 88-100, 2022.
- [12] M. Jain, S. Ghosh, and S. P. Nandanoori, "Workload characterization of a time-series prediction system for spatio-temporal data," in *Proc. ACM Int. Conf. Comput. Frontiers*, 2022.
- [13] Q. Hu et al., "Characterization and prediction of deep learning workloads in large-scale GPU datacenters," in *Proc. SC*, 2021.
- [14] A. K. Paul, A. M. Karimi, and F. Wang, "Characterizing machine learning I/O workloads on leadership scale HPC systems," in *Proc. IEEE MASCOTS*, 2021.
- [15] A. Yu, H. Yang, K. Nguyen, J. Zhang, and M. Cheriet, "Burst traffic scheduling for hybrid E/O switching DCN: An error feedback spiking neural network approach," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 882-893, 2021.
- [16] S. Gupta and A. Dileep, "Long range dependence in cloud servers: A statistical analysis based on Google workload trace," *Comput. Commun.*, vol. 102, pp. 1031-1049, 2020.
- [17] V. Saxena, K. R. Jayaram, S. Basu, Y. Sabharwal, and A. Verma, "Effective elastic scaling of deep learning workloads," in *Proc. IEEE MASCOTS*, 2020.
- [18] Z. Li, W. Xing, S. Khamaseh, and D. Xu, "Detecting saturation attacks based on self-similarity of OpenFlow traffic," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 607-621, 2020.
- [19] C. Ji et al., "Inspection and characterization of app file usage in mobile devices," *ACM Trans. Storage*, vol. 16, no. 4, 2020.
- [20] M. Shahrad et al., "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2020.
- [21] M. Jeon et al., "Analysis of large-scale multi-tenant GPU clusters for DNN training workloads," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2019.

[22] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. OSDI, 2016.

[23] T. Chen et al., "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," arXiv:1512.01274, 2015.

[24] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," IEEE/ACM Trans. Netw., vol. 2, no. 1, pp. 1-15, 1994.

[25] Alibaba Cloud, "Machine Learning Platform for AI (PAI) documentation." [Online]. Available: <https://www.alibabacloud.com/help/en/pai/> . Accessed: Sep. 12, 2026.