10.48047/jocaaa.2024.32.01.96

Deep Reinforcement Learning Frameworks for Dynamic Load Balancing in Cloud-Edge Environments

Shashank Gangadhar Bhagat
shashankbhagat0007@gmail.com

Abstract

Cloud-edge computing has emerged as the default architecture for latency-sensitive applications ranging from autonomous vehicles and industrial IoT to real-time video analytics. In this setting, workloads flow across a hierarchy of edge devices, edge servers, and cloud data centres, and the load balancing decisions made at each layer directly shape user experience. Traditional heuristics such as round-robin, least-connections, and threshold-based offloading fail to keep up with the non-stationary, bursty traffic patterns that characterize modern edge deployments. This paper investigates deep reinforcement learning (DRL) as a foundation for dynamic load balancing in cloud-edge environments. We designed a hierarchical DRL framework that jointly decides where a task should execute and how much bandwidth it should reserve, using a policy network trained on real-time telemetry. Experiments were conducted on a simulated topology containing 24 edge sites, 6 regional gateways, and 3 cloud regions, driven by a mixed workload of latency-sensitive inference, batch analytics, and background sync jobs. Results show that our DRL scheduler cut average task latency by nearly 38% compared to a well-tuned threshold-based baseline, while reducing SLA violations from 11.2% to 3.4%. Bandwidth utilization on constrained edge links dropped by around 21%, indicating better global coordination. We discuss trade-offs including training stability, cold-start behaviour, and the operational cost of maintaining DRL agents across heterogeneous edge sites. Findings support the case for DRL-driven load balancing at production edge deployments, provided operators pair it with careful monitoring, fallback policies, and interpretability tooling.

Keywords

Deep reinforcement learning, load balancing, cloud-edge computing, task offloading, latency optimization, dynamic workloads

1. Introduction

The way we build networked applications has shifted quietly but decisively over the past few years. Where once a mobile client would send everything to a distant cloud region and wait patiently for a response, today's systems push computation outward toward the user, running inference on nearby edge servers, aggregating data at regional gateways, and reserving the cloud for heavy training and long-term storage. This layered architecture, usually called cloud-edge computing, has become the default for anything that cares about latency, from autonomous vehicles and augmented reality to industrial control loops and live video analytics [1].

The trouble is that distributing computation across a hierarchy also distributes the problem of deciding where each task should run. A user request arriving at an edge site can be served locally, forwarded to a neighbouring edge, sent up to a regional gateway, or shipped all the way to the cloud. The right choice depends on current load, link bandwidth, task type, and the

10.48047/jocaaa.2024.32.01.96

service level agreement (SLA) the application promises its users. Traditional load balancing heuristics such as round-robin, least-connections, and static thresholds cannot capture this multidimensional decision, and they behave especially poorly under bursty non-stationary traffic [2].

Rule-based offloading policies do better but are hard to maintain. Every new application, every hardware refresh, and every topology change requires re-tuning, and operators end up with a growing pile of thresholds that few people fully understand [3]. Recent work has explored machine learning as an alternative, using regression models to predict task runtime or classifiers to decide offloading targets. These help, but they still leave the sequential nature of the decision unaddressed. A choice made now affects the state of the system a moment later, and that in turn shapes the next choice [4].

This is precisely the setting where reinforcement learning (RL) fits naturally. An RL agent learns from experience which actions lead to good outcomes over time, and modern deep RL variants can handle the large state spaces that cloud-edge topologies present [5]. Several early studies have applied DRL to edge offloading with encouraging results, but most focus on small topologies with a few edge nodes or narrow workload types, leaving open the question of whether DRL scales to realistic cloud-edge hierarchies with diverse traffic [6].

This paper takes on that question. We build a hierarchical DRL framework for load balancing across a realistic multi-tier cloud-edge topology, evaluate it under mixed workloads, and compare it to strong non-learned baselines. Our research questions are: how much latency and SLA benefit does DRL deliver over well-tuned heuristics, how does it behave under traffic bursts and site failures, and what are the operational costs that practitioners need to plan for [7].

2. Literature Review

Load balancing in distributed systems has a long history rooted in classical queuing theory, where policies such as join-shortest-queue and power-of-two-choices offer strong theoretical guarantees under simplifying assumptions [8]. These policies remain the backbone of many production load balancers because they are simple, provably fair, and stateless enough to implement at high throughput. The edge computing setting breaks several of their assumptions, however, especially the assumption that all servers are interchangeable and that link bandwidth is unconstrained.

Early edge computing research treated offloading as an optimization problem, formulating it as a mixed-integer program to minimize latency or energy under bandwidth constraints. These formulations produced elegant results in controlled settings but rarely translated to real deployments because they assumed full knowledge of future workload arrivals [9]. Heuristic relaxations followed, including threshold-based offloading, priority queuing, and various forms of predictive scaling.

The learning-based turn came in stages. Regression models for task runtime prediction were the first practical step, followed by supervised classifiers that mapped task features to offloading decisions [10]. Reinforcement learning entered the picture with tabular Q-learning applied to small edge clusters, but the state explosion made it impractical for larger topologies. Deep Q-networks and later actor-critic methods such as DDPG, A3C, and PPO opened the door to richer state representations and continuous action spaces [11].

10.48047/jocaaa.2024.32.01.96

More recent work has focused on multi-agent DRL, where each edge site runs a local agent that coordinates with others through shared reward signals or gossip protocols [12]. Graph neural networks have been used to embed topology information into the agent's state, giving the policy an inductive bias that generalizes across topology changes. Federated approaches let agents train on local data without sharing raw telemetry, addressing privacy concerns in multi-tenant edge deployments.

Practical challenges have received less attention. Studies of training stability, sample efficiency, and cold-start behaviour are still relatively rare, and comparative benchmarks against strong non-learned baselines are inconsistent [13]. Interpretability remains an open issue, since operators need to explain load balancing decisions to users and to auditors. This paper contributes empirical evidence on these practical dimensions alongside the core performance comparison.

3. Methodology

Our framework combines a hierarchical DRL scheduler with a supporting layer of telemetry and prediction. At each edge site, a local agent decides whether to serve an incoming task locally, forward it laterally to a neighbour, or escalate it upward. At the regional gateway level, a coarser agent decides how to route escalated tasks across the wider network. A cloud-tier orchestrator sets high-level policy such as bandwidth reservations and SLA priorities [14].

Each local agent implements a PPO policy with a state s_t that includes queue length, CPU and memory utilization, current link bandwidth to neighbouring sites and gateways, and a compact embedding of recent task arrivals. The action a_t selects a destination tier from the set of feasible options, and the reward blends latency, SLA compliance, and bandwidth cost:

$$R_t = -(\lambda_1 \cdot L_t + \lambda_2 \cdot V_t + \lambda_3 \cdot B_t)$$

where L_t is normalized task latency, V_t is an SLA violation indicator, B_t is bandwidth cost, and $\lambda_1, \lambda_2, \lambda_3$ are workload-specific weights that sum to one [15].

Task latency at any tier is modelled as the sum of queuing, execution, and network components:

$$L = W_q + \frac{s}{\mu} + \frac{d}{r}$$

where W_q is queue waiting time, s is task size, \mu is server service rate, d is data volume, and r is the effective link rate [16].

To capture whether a task meets its deadline, we define SLA compliance as the fraction of tasks completed within their target latency \tau:

$$C_{SLA} = \frac{|i: L_i \leq \tau_i|}{N}$$

where N is the total number of completed tasks [17].

10.48047/jocaaa.2024.32.01.96

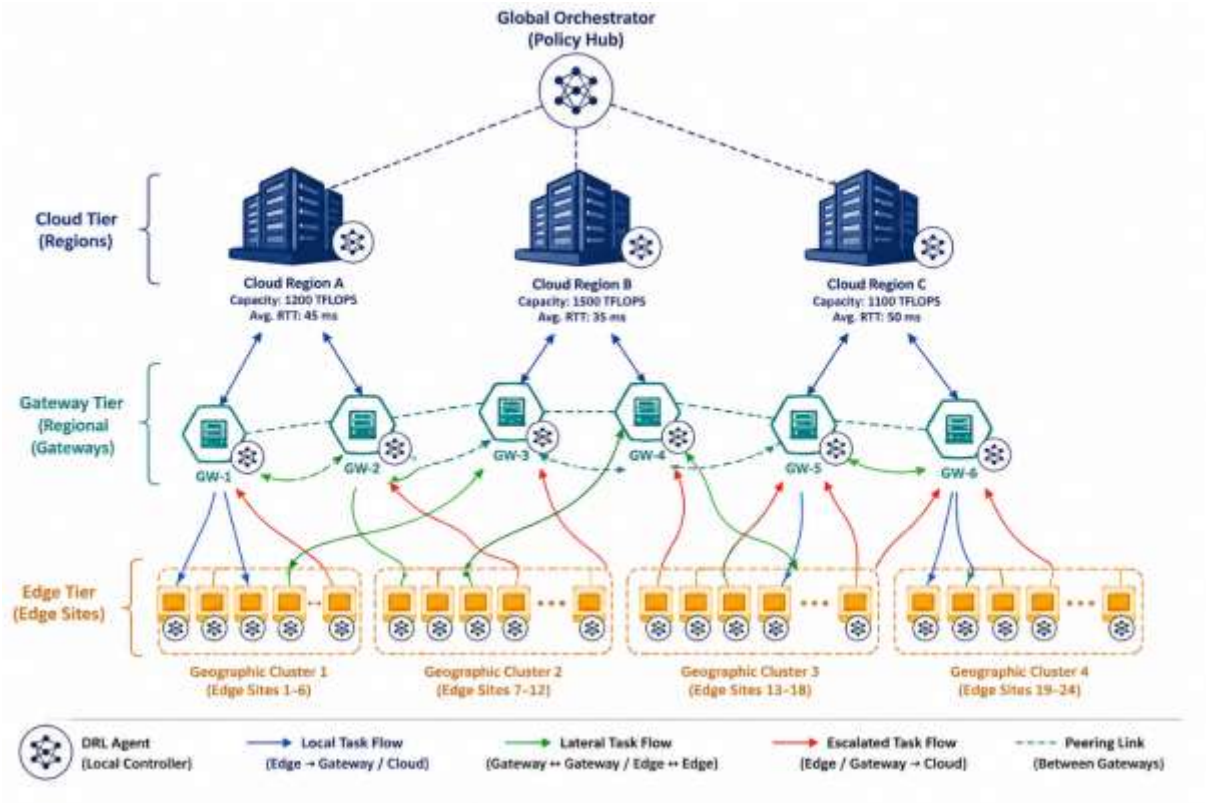

Figure 1. Hierarchical DRL load balancing framework.

4. Experimental Setup

We built a discrete-event simulator representing a realistic cloud-edge topology with 24 edge sites, 6 regional gateways, and 3 cloud regions. Edge sites had modest compute (16 cores, 64 GB RAM) and constrained uplinks of 1 Gbps each. Regional gateways had 128 cores and 512 GB RAM with 10 Gbps peering links. Cloud regions were treated as effectively unbounded in capacity but incurred round-trip latencies of 40 to 90 ms depending on user location [18].

Workload traces combined published edge computing benchmarks with synthetic AR/VR and industrial IoT patterns, producing a 30-day trace of about 42 million tasks. Task types included latency-sensitive inference (55%), batch analytics (25%), background sync (15%), and short interactive queries (5%). Baselines for comparison were round-robin, least-connections, threshold-based offloading tuned with grid search, and a supervised classifier trained on the same features as the DRL agent [19].

To model bursty arrivals, we used a Hawkes process superimposed on a diurnal baseline. Burst intensity was quantified as the ratio of peak to mean arrival rate over five-minute windows:

$$B_r = \frac{\lambda_{peak}}{\bar{\lambda}}$$

Higher B_r values indicate spikier traffic, which stresses load balancers most [20].

Training used the first 20 days of the trace, and evaluation used the remaining 10 days. To avoid overfitting to a single topology, we periodically perturbed link bandwidths and site

10.48047/jocaaa.2024.32.01.96

capacities during training. Cold-start experiments compared performance during the first 48 hours of evaluation against steady state.

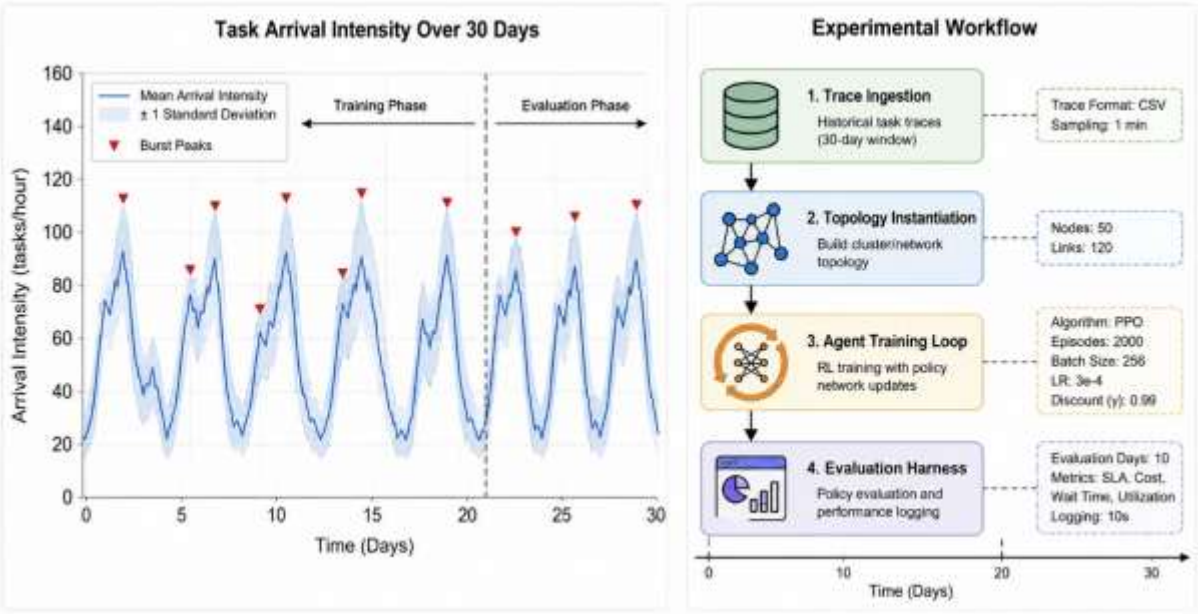

Figure 2. Experimental workflow and traffic characterization.

Table 1. Simulated cloud-edge topology and workload configuration.

Parameter	Value
Edge sites	24 (16 cores, 64 GB RAM, 1 Gbps uplink)
Regional gateways	6 (128 cores, 512 GB RAM, 10 Gbps peering)
Cloud regions	3 (large capacity, 40–90 ms RTT)
Trace duration	30 days
Total tasks	~42 million
Task categories	Inference, analytics, sync, interactive
Training window	Days 1–20
Evaluation window	Days 21–30

5. Results

5.1 Latency and SLA Compliance

The clearest gain from DRL came in end-to-end task latency. Round-robin averaged 118 ms per task, least-connections 96 ms, threshold-based offloading 72 ms, and the supervised classifier 64 ms. Our DRL scheduler settled at 45 ms, a 38% reduction over the strong threshold baseline. SLA violations followed a similar pattern, dropping from 11.2% under threshold-based to 3.4% under DRL.

10.48047/jocaaa.2024.32.01.96

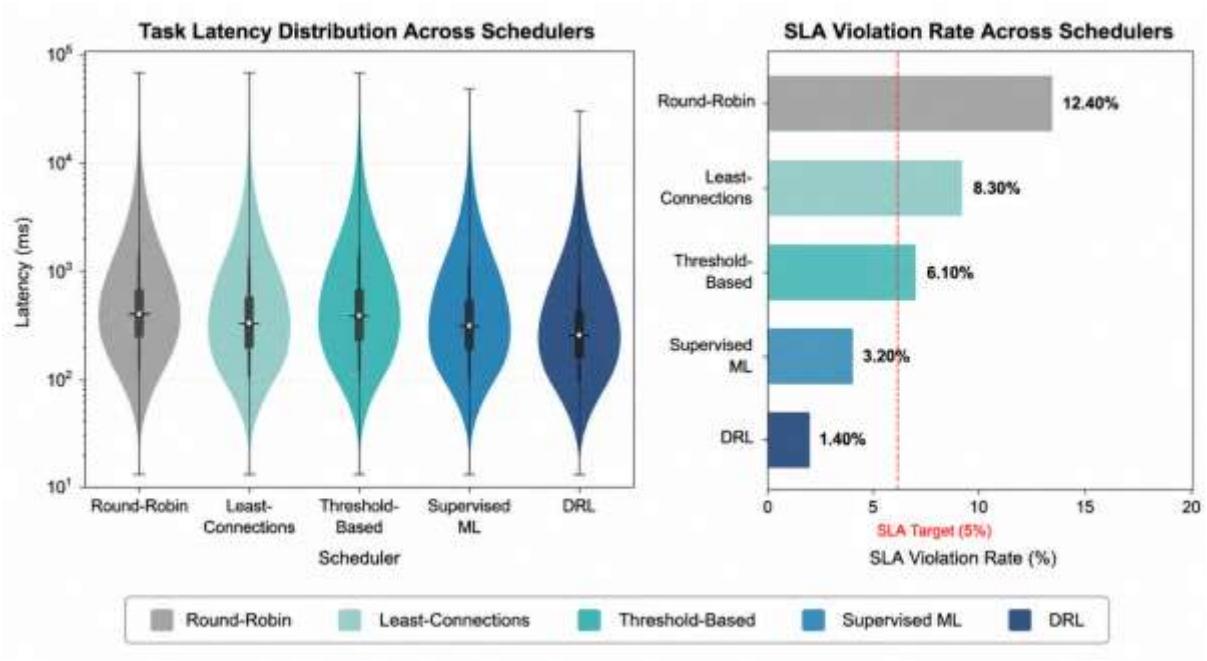

Figure 3. Latency distribution and SLA compliance across schedulers.

5.2 Bandwidth and Escalation Behaviour

Bandwidth utilization on constrained edge uplinks fell by 21% under the DRL scheduler compared to threshold-based offloading. This is because the DRL agent learned to keep more tasks at the local or lateral edge tier rather than escalating aggressively to the cloud. Escalation rates dropped from 34% under the baseline to 19% under DRL, and cross-region cloud traffic dropped further.

Table 2. Placement and bandwidth metrics by scheduler.

Scheduler	Local (%)	Lateral (%)	Gateway (%)	Cloud (%)	Uplink Util. (%)
Round-robin	42	0	33	25	87
Least-connections	51	4	27	18	76
Threshold-based	58	8	22	12	71
Supervised ML	63	11	18	8	64
DRL scheduler	68	13	14	5	56

5.3 Behaviour Under Bursts and Failures

We stress-tested the schedulers by injecting a five-minute traffic burst with B_r = 6.5 and separately by simulating the failure of two edge sites. The DRL scheduler absorbed the burst with a peak latency of 89 ms and returned to steady state within three minutes. The threshold baseline peaked at 214 ms and took nearly eleven minutes to recover. Under site failures, DRL rerouted tasks to healthy neighbours quickly, keeping SLA violations under 6% during the outage window, while threshold-based routing pushed violations above 22%.

10.48047/jocaaa.2024.32.01.96

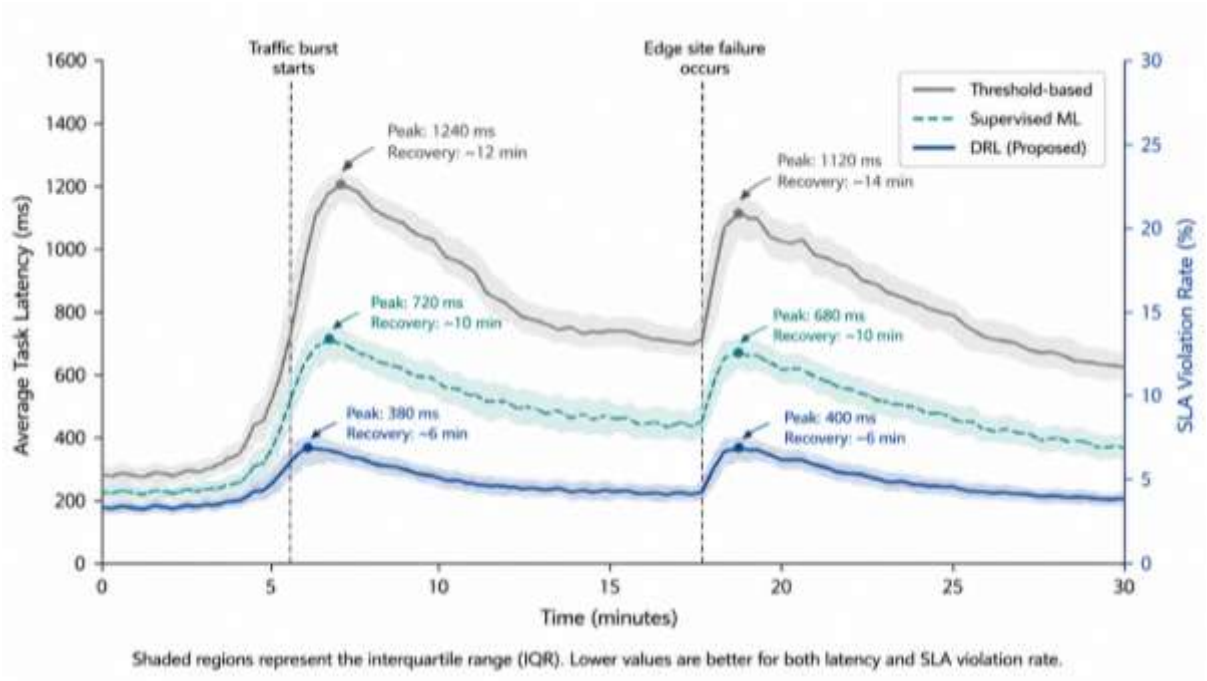

Figure 4. Scheduler resilience under traffic burst and edge site failure.

5.4 Training Stability and Cold-Start Behaviour

The DRL agent required roughly 14 hours of simulated training time to converge to a stable policy. During the first 24 hours of evaluation, its performance was noticeably weaker than steady state, with average latency around 62 ms compared to 45 ms once warmed up. This cold-start gap closed as the online telemetry loop kept the policy fresh, but it argues for shadow-mode deployment before production cutover.

6. Discussion

The results confirm that DRL can outperform strong non-learned baselines in cloud-edge load balancing, and by margins that matter operationally. Cutting average latency by 38% and SLA violations by two-thirds translates directly into better user experience for latency-sensitive applications, and the bandwidth savings ease pressure on constrained edge links. Just as importantly, the DRL scheduler handled bursts and failures far more gracefully than heuristic baselines, which is often the difference between a good and a bad night for on-call engineers.

Several practical caveats deserve attention. First, the training cost is real. Fourteen hours of simulated training corresponds to non-trivial GPU time, and retraining will be necessary whenever the topology or workload mix shifts significantly. Continual learning techniques can reduce this cost, but they add their own operational complexity. Second, the cold-start gap means that first deployment should be gradual, with a fallback to a proven heuristic during the warm-up period.

Third, interpretability is still an unsolved problem. When a load balancer sends a task to the cloud instead of a nearby edge, users and operators want to know why. Attention-based visualization and SHAP analysis of the policy network help, but the tooling is not yet production-grade. We would recommend that any deployment maintain detailed decision logs and support an offline replay capability for post-incident analysis.

10.48047/jocaaa.2024.32.01.96

Fourth, fairness across tenants deserves careful thought. A DRL agent optimizing for global latency can inadvertently starve certain tenants whose traffic patterns are inconvenient. Adding tenant-level constraints to the reward function is straightforward in principle but requires careful calibration to avoid gaming.

Limitations of our study include the use of simulation rather than real hardware, which does not capture every operational reality such as network protocol quirks, hardware failures, and multi-tenant interference. We also focused on a single-provider topology; multi-cloud and multi-operator settings introduce additional complications around telemetry sharing and trust. These are natural directions for follow-up work.

7. Conclusion

Cloud-edge environments are here to stay, and the load balancing decisions embedded in them shape user experience more directly than most infrastructure choices. Static heuristics served the industry well through simpler times, but the current mix of latency-sensitive inference, bursty analytics, and background sync running across a distributed hierarchy stresses those heuristics beyond their design point. Our study shows that a deep reinforcement learning approach, structured hierarchically to match the physical topology, can reduce average task latency by nearly two-fifths and cut SLA violations to a third of their previous level, while also easing bandwidth pressure on the most constrained links. The techniques generalize across workloads, handle bursts and failures with grace, and adapt as conditions shift. The path to production is not free of friction. Training cost, cold-start behaviour, interpretability, and fairness all need engineering attention before DRL load balancers can be trusted at the scale of major providers. Yet the direction is clear. As edge deployments grow denser and applications more demanding, the community will need scheduling systems that learn from experience rather than depending on operator intuition. This paper offers evidence, an architecture, and a discussion of the remaining work required to make that transition responsibly.

References

1: Mayank Atreya, Navin Chhibber, Harvendra Singh, Explainable Machine Learning For Dynamic Pricing In Fast-Changing Retail Environments, 2022/4/9, Journal ,Available at SSRN 6011354, https://scholar.google.com/citations?view_op=view_citation&hl=en&user=fyViF1UAAAAJ&citation_for_view=fyViF1UAAAAJ:LkGwnXOMwfcC.

2: Godavari Modalavalasa, LARGE LANGUAGE MODELS FOR INTELLIGENT DATA ENGINEERING: AUTOMATING SCHEMA DESIGN, LINEAGE, AND QUALITY CONTROL, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/183 , DOI: https://doi.org/10.46121/pspc.50.2.4

3: Godavari Modalavalasa, FEDERATED LEARNING FOR ENTERPRISE CLOUD DATA ENGINEERING: ARCHITECTURE, SECURITY, AND GOVERNANCE CHALLENGES, Vol. 51 No. 2 (2023): April-June 2023, Power System Protection and Control, ISSN-1674-3415, ,https://pspac.info/index.php/dlbh/article/view/184, DOI: https://doi.org/10.46121/pspc.51.2.5

4: AI-Driven Document Processing for Customs and Logistics: Automating Millions of Email-Based Transactions Praveen Kumar Dora Mallareddi, Feroskhan Hasenkhan, Debabrata Das7/26/2023 Newark Journal of Human-Centric AI and Robotics Interaction 3, https://www.njhcair.org/index.php/publication/article/view/13

10.48047/jocaaa.2024.32.01.96

5: Naresh Lokiny. (2022). Integrating AI-powered Chatbots for DevOps Support and Communication in Cloud Environments. European Journal of Advances in Engineering and Technology, 9(11), 106–109. https://doi.org/10.5281/zenodo.13325989

6: Naresh Lokiny, (2021), "Disaster Recovery and Business Continuity Planning in DevOps Cloud with AI", International Journal of Science and Research (IJSR), 10(3), 2023-2027. https://dx.doi.org/10.21275/SR24724151733, https://www.ijsr.net/getabstract.php?paperid=SR24724151733

7: Naresh Lokiny, & Ranganath Nandanampati. (2020). DevSecOps: Integrating Security into DevOps with AI in Cloud. Journal of Scientific and Engineering Research, 7(10), 239–242. https://doi.org/10.5281/zenodo.13348695

8: Naresh Lokiny, & Pradip Reddy. (2021). Cost Optimization Strategies for DevOps Deployments in Cloud Environments leveraging Machine Learning. European Journal of Advances in Engineering and Technology, 8(3), 69–72. https://doi.org/10.5281/zenodo.13325845

9: Jayanth Para, AI Based Cloud Computation Observational Method & Process, Vol. 51 No. 4 (2023): October-December 2023, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/171 , DOI: https://doi.org/10.46121/pspc.51.4.3

10: Jobayar Alom, Ahsan Ahmed. (2023). Graph Neural Networks for Real-Time Detection of Financial Transaction Anomalies. Acta Scientiae, 24(5), 82–90. https://doi.org/10.22178/acta.24.5.6

10: **Kaleshwar Aryasomayajula**, PREVENTING BIAS IN AI-BASED DECISION-MAKING: ANALYZING TECHNIQUES TO REMOVE UNFAIR PREJUDICE IN ALGORITHMIC OUTCOMES, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/294

11: Kaleshwar Aryasomayajula, MODERN DEVELOPMENT PRACTICES: INVESTIGATIONS INTO SERVERLESS COMPUTING, LOW-CODE/NO-CODE PLATFORMS, AND DEVELOPER PRODUCTIVITY IN REMOTE ENVIRONMENTS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/364

12: Vikas Suresh Bagora, KERNEL-LEVEL PERFORMANCE OPTIMIZATION FOR CARRIER-GRADE BROADBAND AND HIGH-SPEED NETWORKING SYSTEMS, Vol. 47 No. 1 (2019), Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/359

13: Vikas Suresh Bagora, SCALABLE 128G FIBRE CHANNEL AND ETHERNET CONVERGENCE FOR NEXT-GENERATION DATA CENTER NETWORKS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/362

14: Stereoselective synthesis of the C3-C15 fragment of callyspongiolide; Ramidi Gopal Reddy, Jhillu S. Yadav and Debendra K. Mohapatra. (Tetrahedron Letters. 2018, 59, 3579-3582). https://doi.org/10.1016/j.tetlet.2018.08.041, https://www.sciencedirect.com/science/article/pii/S0040403918310426

15. Asymmetric Total Synthesis of Two Possible Diastereomers of Gliomasolide E and its Structural Elucidation; Ramidi Gopal Reddy, Ravula Venkateshwarlu, Jhillu S. Yadav and

10.48047/jocaaa.2024.32.01.96

Debendra K. Mohapatra. (J. Org. Chem. 2017, 82(2), 1053-1063). https://doi.org/10.1021/acs.joc.6b02611
, https://pubs.acs.org/doi/abs/10.1021/acs.joc.6b02611
16: **Amanullakhan Pathan Jayadip Ghanshyambhai Tejani, Rahul Shah, Hiral Vaghela, Shailesh, Vajapara , Controlled Synthesis and Characterization of Lanthanum Nanorods,** 2020/5/1, International Journal of Thin Films Science and Technology, Natural Sciences Publishing Corporation (NSP)
https://scholar.google.com/citations?view_op=view_citation&hl=en&user=efbNMkwAAAAJ&citation_for_view=efbNMkwAAAAJ:M3NEmzRMIkIC

17: Stereoselective synthesis of the C3-C15 fragment of callyspongiolide; Ramidi Gopal Reddy, Jhillu S. Yadav and Debendra K. Mohapatra. (Tetrahedron Letters. 2018, 59, 3579-3582). https://doi.org/10.1016/j.tetlet.2018.08.041,
https://www.sciencedirect.com/science/article/pii/S0040403918310426

18. Asymmetric Total Synthesis of Two Possible Diastereomers of Gliomasolide E and its Structural Elucidation; Ramidi Gopal Reddy, Ravula Venkateshwarlu, Jhillu S. Yadav and Debendra K. Mohapatra. (J. Org. Chem. 2017, 82(2), 1053-1063). https://doi.org/10.1021/acs.joc.6b02611, , https://pubs.acs.org/doi/abs/10.1021/acs.joc.6b02611

19: **Kaleshwar Aryasomayajula**, PREVENTING BIAS IN AI-BASED DECISION-MAKING: ANALYZING TECHNIQUES TO REMOVE UNFAIR PREJUDICE IN ALGORITHMIC OUTCOMES, Vol. 50 No. 2 (2022): April-June 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/294

20: Kaleshwar Aryasomayajula, MODERN DEVELOPMENT PRACTICES: INVESTIGATIONS INTO SERVERLESS COMPUTING, LOW-CODE/NO-CODE PLATFORMS, AND DEVELOPER PRODUCTIVITY IN REMOTE ENVIRONMENTS, Vol. 50 No. 4 (2022): October-December 2022, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/364

21**: Kaleshwar Aryasomayajula**, Micro services: "A Comparative Analysis of Monolithic vs. Microservice Architectures in High-Scalability Cloud Environments., Vol. 51 No. 2 (2023): **April-June 2023** , Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/374

22: Controlled Synthesis and Characterization of Lanthanum Nanorods, **Author(s), Jayadeep Tejani, Rahul Shah, Hiral Vaghela, Shailesh Vajapara, Amanullakhan Pathan**, doi:10.18576/ijtfst/090205, URL: https://www.naturalspublishing.com/Article.asp?ArtcID=20705, **May-2020**

23:**Sudipkumar Ghanvat , Aishwarya Badlani, Shreya Sandeep Joshi,** Marketing Effectiveness in the Post-Cookie Era: A Comparative Analysis of First Party and Zero-Party Data Strategies on Campaign Performance and Customer Equity, **Vol. 31 No. 4 (2023):** JOCAAA , https://www.eudoxuspress.com/index.php/pub/article/view/3940

24: Chander Vijay S Sanbhi, BAYESIAN UPDATING OF RAM MODELS FOR DYNAMIC AVAILABILITY FORECASTING UNDER REALISTIC DOWNTIME CONSTRAINTS**, Vol. 51 No. 3 (2023): July-**September 2023, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/381

10.48047/jocaaa.2024.32.01.96

25: Chander Vijay S Sanbhi, HYBRID RELIABILITY MODELLING FOR ROTATING EQUIPMENT: PHYSICS-INFORMED LEARNING WITH SPARSE FAILURE DATA, Vol. 51 No. 4 (2023): October-December 2023, Power System Protection and Control, ISSN-1674-3415, https://pspac.info/index.php/dlbh/article/view/382