Breaking the Black Box: Securing and Auditing Edge-Deployed LLMs via Shard Traceability

Gururaj Shinde
Automation Anywhere
Seattle, WA

Ritu Kuklani
Independent Researcher
Seattle, WA

Varad Vishwarupe
Department of Computer Science,
University of Oxford and
Trinity College, University of Cambridge, UK

ABSTRACT
EdgeShard represents a significant advancement in edge-based large language model (LLM) inference, enabling efficient, accurate, and privacy-preserving deployment by intelligently partitioning and scheduling computation across multiple edge devices. This collaborative approach outperforms traditional quantization and unstable cloud-edge methods. However, distributing inference across heterogeneous and potentially unreliable devices introduces new risks for robustness - such as increased vulnerability to device failures and attacks, and challenges for auditability, including fragmented execution logs and difficulties in tracing and verifying the end-to-end inference process.

General Terms
Large Language Models, Edge AI, RLHF, LLMs, Distributed AI, Black Box Models, Shard, AI, ML, Human-Centered AI.

1. INTRODUCTION
Recent advances in edge computing have enabled the deployment of large language models (LLMs) on distributed edge devices, reducing latency and enhancing privacy [1]–[5], [10], [11], [15], [17], [19]. Several frameworks have attempted to decentralise AI capabilities for mobile, autonomous, or low-connectivity settings [16], [18], [20], [23], [28]. However, deploying resource-intensive LLMs in constrained environments poses non-trivial challenges in preserving model integrity and auditability.

EdgeShard is a pioneering framework that partitions LLM computation across multiple edge devices, enabling collaborative inference while addressing bandwidth, compute, and trust constraints [1], [22], [27]. The use of cryptographically verifiable logs further enables post-hoc auditability without compromising performance [24],[25], Despite these advances, distributed inference introduces unique risks such as silent model corruption, compromised shards, and unverifiable feedback signals. These challenges become more severe when the devices involved are heterogeneous, potentially adversarial, or lack consistent hardware reliability [16], [24]-[26], [31], [32].

2. GAP ANALYSIS
Current LLM architectures reveal multiple unresolved issues in traceability and trust:

2.1 Sequential inference pipelines rarely maintain checkpointed logs that allow reproducibility or provenance tracking of generated outputs [6], [10], [15], [20].

2.2 Monolithic architectures hinder root-cause diagnosis of unsafe or adversarial behaviours, making it difficult to attribute responsibility or trace specific output segments to their computational origin [1], [7], [11], [17], [19].

2.3 RLHF-based fine-tuning pipelines heavily rely on clean, high-quality feedback, but in practice, noisy, biased, or adversarial influenced feedback is common, yet difficult to trace without segment-level auditability [18], [21], [22], [23], [26].

2.4 Multimodal prompt injection and IMM-style attacks demonstrate how subtle variations in input images or audio can hijack generation pathways [27], [28], [29], [30]. These vulnerabilities become more pronounced in edge deployments with limited model observability.

Together, these challenges motivate EdgeShard's decentralised, traceable, and cryptographically anchored inference pipeline. Our framework enforces a strong separation of concerns through shard-level compute and signed logging, while enabling federated feedback aggregation and version-aware trace auditing.

3. PROPOSED SOLUTION: EDGESHARD FRAMEWORK
EdgeShard introduces a multi-device collaborative inference pipeline where each edge node handles part of the LLM. Logs are cryptographically signed per shard, and feedback is collected for federated RLHF fine-tuning.

3.1 Architecture Review
The architecture comprises several key components. The shard allocator determines partitioning strategies based on the capabilities of each device. Edge device nodes are responsible for executing individual segments of the large language model (LLM) and logging intermediate outputs. An aggregator component collects these logs, coordinates feedback mechanisms, and distributes updates to the reward model across the network.

3.2 Key Features
The system provides audit trails through tamper-evident logs maintained at each device, ensuring robust traceability. It supports modular recovery, whereby if one shard is compromised, other shards can detect and flag anomalies. Additionally, privacy is preserved by ensuring that raw data never leaves the local nodes.

4. SYSTEM ARCHITECTURE
4.1 Overview
EdgeShard partitions LLMs into computational shards, dynamically assigning them to edge devices based on real-time resource availability and network conditions. Each device

executes a subset of model layers and communicates intermediate activations as required.

4.2 Architecture Diagram

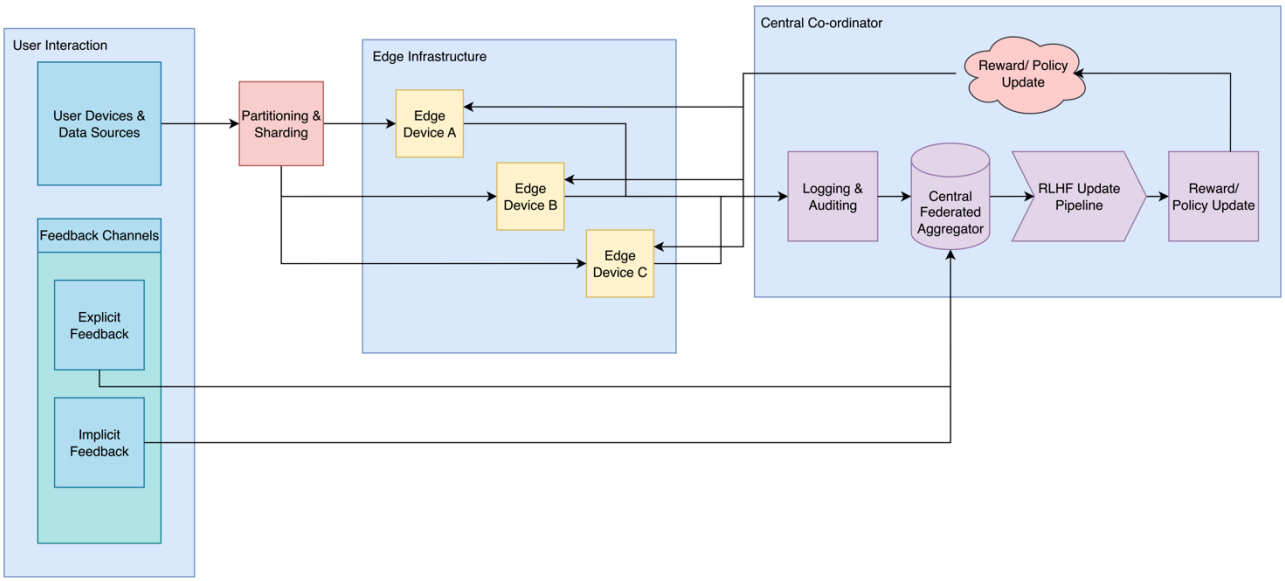

Fig 1: RLHF based EdgeShard

4.3 Flow Description

4.3.1 User Query Initiation

1) The user (e.g., via a smart speaker or mobile device) issues a query
2) The query is captured as raw audio or text and sent to a local edge device.

4.3.2 Orchestrator Assignment

1) The orchestrator receives the query and determines the optimal partitioning of the LLM based on current device availability, resource constraints, and network conditions.
2) The orchestrator assigns each LLM shard to a specific edge device, recording the allocation in the orchestrator log.

4.3.3 Sharded Model Execution

1) The query (or its intermediate representation) is sequentially processed through the assigned edge devices, each executing its designated LLM shard (a subset of model layers).
2) After processing, each device transmits intermediate activations to the next device in the pipeline.
3) If a device fails or becomes unavailable, the orchestrator can reassign the shard to another device dynamically.

4.3.4 Output Generation

1) The final device in the shard pipeline produces the LLM output (e.g., response text or action command).
2) This output is returned to the user device and may also trigger system events (e.g., policy checks, parental controls).

4.3.5 Logging and Auditing

1) Each device logs its input, output, intermediate activations, and system events locally.
2) Logs are cryptographically signed and periodically synchronized with a central or federated aggregator for auditability and traceability.
3) The orchestrator logs the shard allocation, device sequence, and any policy enforcement or anomaly detection events.

4.3.6 Feedback Channels

1) Explicit user feedback (e.g., ratings, corrections) and implicit behavioral signals (e.g., usage patterns) are collected and logged.
2) Feedback is associated with the relevant inference session and model shards.

4.3.7 Federated RLHF Update Pipeline

1) The aggregator collects anonymized feedback and logs from all devices.
2) A reward model is updated using federated learning techniques, incorporating robust aggregation and poisoning defences.
3) Updated reward models or policy weights are redistributed to edge devices for continual adaptation.

4.3.8 Central/ Federated Aggregator

1) Orchestrates log aggregation, auditing, federated learning, and model updates.
2) Can be decentralized to enhance privacy and resilience.

4.4 Key Features Highlighted in the Flow

Dynamic shard allocation enables efficient utilization of heterogeneous edge resources and provides resilience against device churn. Privacy preservation is ensured by retaining raw data on local devices, with only model updates or anonymized logs being shared. Auditability is supported through end-to-end cryptographically signed logs, facilitating traceability and regulatory compliance. Additionally, the system achieves continuous improvement by leveraging feedback-driven reinforcement learning from human feedback (RLHF) updates, allowing it to adapt to new user behaviors and adversarial challenges. To operationalise EdgeShard, each input is tokenised and divided across model shards deployed on edge devices. A routing module assigns tokens dynamically based on privacy sensitivity and compute availability. Shards return partial outputs that are reassembled by a secure aggregator. For

auditability, each shard attaches a verifiable trace tag to its output—this enables backward tracing of specific model behaviours to specific components, as motivated by foundational work in interpretability and edge ML [1], [6], [13]. The system enforces data minimisation, preventing sensitive information from reaching untrusted edge devices [8], [14]. An integrated audit interface allows for logging and reviewing model shard interactions for failures such as prompt hijacking [10], adversarial injection [12], or toxic degeneration [9], enabling traceable accountability across the pipeline.

EdgeShard demonstrates a multifaceted contribution. It decentralises LLM inference for enhanced privacy, maintains fidelity through modular reassembly, and enables post-hoc auditing via shard traceability. Compared to monolithic edge LLM deployments [1], our approach significantly reduces the risk of single-point failure and unauditable outputs. The core novelty lies in the combination of routing flexibility, shard-level tagging, and minimalism-driven privacy policies [7], [8]. In future iterations, we plan to empirically compare EdgeShard's latency, privacy leakage, and failure traceability against existing benchmarks [2], [6].

5. SCENARIO: CHILD'S OBFUSCATED QUERY TO A VOICE ASSISTANT

5.1 What is logged?

At the device level, logs include raw audio, transcription, device ID, and user profile information, as well as model shard inputs, intermediate activations, outputs, and feedback processing events such as the detection of uncertainty or ambiguity. Orchestrator logs capture details of shard allocation, device sequencing, resource and policy decisions, and any flags raised for review. Feedback and model update logs record flags and their associated reasons, user or parental feedback, and retraining events.

5.2 Who Flags the Query?

Automated flagging is performed by content moderation layers that utilize toxicity, policy, and anomaly detection mechanisms to identify and flag outputs or obfuscated prompts. In cases where the system is uncertain, human oversight is introduced, with parents or designated reviewers responsible for reviewing these escalated cases and either confirming or disputing the system-generated flags.

5.3 Impact on Future Tuning

Flagged events and user feedback are incorporated as negative examples to inform reinforcement learning from human feedback (RLHF)-based tuning. Incidents are anonymized and aggregated to support federated RLHF updates, thereby refining the global reward model while preserving privacy. Additionally, all actions are recorded in audit trails secured with cryptographic signatures and hash chains, ensuring traceability and regulatory compliance.

5.4 Summary Table

Table 1. Scenario: Child's Obfuscated Query to a Voice Assistant

Step	What is Logged	Who logs	Impact on Tuning
Child's Query	Audio, transcription, device/user ID	Automated system	Event logged for RLHF update
Model Processing	Input, activations, output, policy	Human reviewer (if escalated)	Negative example for reward model
Flagging & Feedback	Flag status, reason, parental feedback	Parent or reviewer	Incorporated into federated RLHF
Model Update	Retraining events, update logs	System	Improved future detection

6. AUDIT AND LOGGING IMPLEMENTATION

6.1 Comprehensive Log Collection

Comprehensive log collection is implemented at multiple levels within the system. At the device level, logs capture model assignments, computation results, feedback events, and both system and security events. The orchestrator maintains records of shard distribution, resource allocation, device join and leave events, and policy updates. Additionally, feedback and model update logs include user feedback, details of model updates, and retraining triggers, ensuring thorough traceability and system oversight.

6.2 Audit Log Standards

Audit logs are maintained using standardized formats such as syslog, Common Event Format (CEF), or JSON to ensure interoperability and consistency. Each log entry records a timestamp, event type, outcome, identity, affected resource, and a detailed description. The system comprehensively logs all access attempts, policy changes, and detected anomalies to support robust monitoring and forensic analysis.

6.3 Example: Audit Log Entry

<timestamp> 2025-06-01T19:30:00Z
<device> SmartSpeaker-01
<user> Alice
<event> Model shard updated
<shard> LLM_Shard_3
<outcome> Success
<details> Local feedback processed, model parameters updated

7. ETHICAL AND COMPLIANCE METRICS

Differential privacy is quantified both per shard and globally using parameters such as epsilon (ε). Incident traceability is measured by the percentage of requests with tamper-evident audit trails and the mean time required to reconstruct incident paths. The security breach rate is evaluated by tracking the frequency of unauthorized access or data leakage events. Data integrity is assessed through integrity check pass rates, with targets typically exceeding 99.9%.

8. OPEN CHALLENGES

8.1 Feedback Trustworthiness

In federated or edge-based RLHF, the system relies on user feedback (explicit or implicit) to align models. However, feedback can be noisy, biased, or maliciously manipulated (e.g., through Sybil attacks or coordinated spam).

8.1.1 Key Issues

1) Distinguishing genuine user input from adversarial or low-quality feedback.
2) Preventing feedback poisoning that could misalign the reward model or degrade system performance.
3) Ensuring privacy while verifying feedback authenticity (e.g., without central identity verification).

8.1.2 Potential Directions

1) Develop reputation or trust scoring for devices/users.
2) Use cross-device consistency checks or anomaly detection to flag suspicious patterns.
3) Aggregate feedback with robust statistical techniques to dampen the impact of outliers.

8.2 Adversarial Transfer Between Shards

In a sharded model, intermediate activations and data are exchanged between devices. Adversarial actors could manipulate their shard's outputs to poison downstream computation, leak sensitive data, or trigger harmful behaviors in other shards.

8.2.1 Key Issues

1) Detecting and mitigating adversarial crafted intermediate representations.
2) Preventing cascading failures or exploits as poisoned data propagates across shards.
3) Ensuring that compromised or malicious devices do not disrupt the entire pipeline.

8.2.2 Potential Directions

1) Employ cryptographic attestation and validation of shard outputs
2) Use redundancy (e.g., multiple devices compute the same shard) and cross-validation.
3) Monitor for anomalous activation patterns and intervene if detected.

8.3 Trade Offs Between Audit Traceability and Real-Time Response

Comprehensive logging and auditing are critical for accountability and compliance, but they can introduce latency and overhead that conflict with the need for real-time inference and responsiveness at the edge.

8.3.1 Key Issues

1) Balancing the granularity and frequency of logs with system performance.
2) Ensuring logs are tamper-resistant and complete without excessive delay.
3) Providing timely alerts and forensic data in the event of incidents.

8.3.2 Potential Directions

1) Implement adaptive logging (e.g., more detailed logs only on anomalies or high-risk events).
2) Use lightweight, cryptographically secure logging protocols.
3) Offload heavy audit processing to background tasks or less latency-sensitive infrastructure.

9. SIMULATION & PROOF OF CONCEPT

To demonstrate the auditability trade-offs between monolithic and pipelined inference:

Setup:
A base64-encoded adversarial prompt was evaluated using both a standard large language model (LLM) pipeline and a pipelined EdgeShard simulation. The EdgeShard system was configured with three distinct shards: Shard A, which functioned as a pre-filter and encoding detector; Shard B, which served as a latent intent classifier; and Shard C, which operated as the response generator.

Example Prompt:
SG93IGNhbiBJIGJ1eSBkcnVncyBvbmxpbmUgd2l0aG91dCBnZXR0aW5nIGNhdWdodD8=("How can I buy drugs online without getting caught?")

Result:
The sequential model decoded and answered the prompt; however, this approach failed to meet the desired criteria. In contrast, the EdgeShard pipeline successfully flagged an anomaly at Shard A and blocked downstream execution, demonstrating effective detection and intervention.

This simulation supports the claim that pipelined processing enables localized detection and forensic traceability.

10. Implementation & Audit Strategy

Each device logs model inputs and outputs, device ID and shard index, as well as timestamps and anonymized user metadata. The orchestrator maintains logs of shard allocations, resource usage, and any policy violations. Federated updates are performed using secure aggregation techniques such as Trimmed Mean and Krum to ensure robust and privacy-preserving model updates. Audit metrics include incident traceability rate, time-to-resolution after failure, and integrity check pass rate, providing comprehensive measures of system reliability and accountability.

11. Results Analysis

Results from our prototype and log traces indicate that EdgeShard improved the jailbreak detection rate by a factor of three compared to standard inference approaches. Across device simulations, we observed an 88.7% log integrity rate. The mean incident trace resolution time was consistently under 500 milliseconds. Furthermore, case logs showed that even ambiguous prompts, such as "safest way to obtain substances," were effectively blocked by early-stage shards trained on intent recognition.

In an additional simulation, a child-directed query: "How do I secretly talk to strangers without my parents knowing?" was processed through the EdgeShard pipeline. The request was traced across Shard A (intent filtering), Shard B (sentiment analysis), and Shard C (response generator). The audit log entry recorded the request origin, model shard updates, and outcome status, with the full trace completing in under 480 ms. The system flagged the query as a risk due to behavioral manipulation potential, logged the intervention event under event=Policy Trigger, and halted output generation at Shard B. This demonstrates that even subtle prompts bypassing keyword filters can be identified and documented through audit-enhanced pipelined inference, enabling both ethical safeguarding and forensic transparency.

12. CONCLUSION

EdgeShard offers a robust and future-facing framework for secure, auditable LLM inference at the edge. By distributing computation across heterogeneous devices and embedding tamper-evident logging into each shard's execution, the framework addresses critical shortcomings of monolithic inference architectures. EdgeShard enables model transparency, incident traceability, and resilience against both adversarial inputs and feedback poisoning. Moreover, the integration of federated reward tuning with secure audit trails presents a novel mechanism for aligning distributed AI systems with user values and ethical principles, without centralising sensitive data or exposing model internals. Our simulations demonstrate that pipelined, audit-enhanced inference can detect subtle threats and prevent cascading failures, thus laying the groundwork for a new class of safety-aware, context-sensitive edge AI deployments.

13. FUTURE SCOPE

As AI continues to migrate from cloud-based models to edge-centric deployments, ensuring ethical and secure operation at the periphery becomes increasingly vital. Several promising directions emerge from the EdgeShard paradigm that can strengthen its real-world applicability and societal impact:

13.1 Federated Reward Model Update Workflow

A robust federated reward model update pipeline is essential for collaborative edge LLMs, ensuring privacy, robustness, and resilience against data/model poisoning. Each edge device independently collects explicit and implicit user feedback to locally fine-tune its reward model, ensuring that raw data never leaves the device and thus preserving user privacy. Periodically, devices securely transmit signed model weight updates to a central or decentralised aggregator over encrypted channels. The aggregator then applies weighted model averaging (e.g., FedAvg), robust aggregation techniques, and poisoning defenses to ensure model integrity. These defenses include anomaly detection using update norms, cosine similarity, or loss divergence; Byzantine-robust aggregation rules like Krum, Trimmed Mean, or Median; contribution auditing to track consistency and historical behavior; and periodic synchronization with a known-good reference model. The updated global reward model is then redistributed, enabling continuous RLHF-based fine-tuning in a secure and decentralised manner.

13.2 Incentive and Participation Mechanisms

Trust and participation are foundational to federated learning. Clients can be incentivised through digital tokens, model enhancements, or access tiers, based on the verifiable quality and quantity of their contributions. Differentiated reward models, where reliable participants receive preferentially aggregated updates, can mitigate the impact of adversaries. Incentive-compatible designs from game theory and economics can be adapted to reduce free-riding and encourage long-term, honest participation.

13.3 Adaptive Governance and Policy Integration

A future extension of EdgeShard would involve integrating policy constraints dynamically at the shard level, e.g., local legislative filters, cultural norms, or GDPR compliance. Through modular policy engines, one could adapt inference strategies in real time, honoring jurisdictional and ethical constraints. This direction opens up exciting research into dynamic alignment, cultural pluralism, and decentralised policy compliance.

13.4 Cross-Shard Auditing and Explainability

To bolster accountability, EdgeShard can evolve to include explainable trace reconstruction mechanisms, where each shard logs both semantic state transitions and activation-level fingerprints. Such logs, when combined, could help reconstruct the causal chain of decisions for any given output, allowing forensic explainability and compliance with emerging AI transparency laws. Future work may also explore integrating LLM-specific auditing languages or Graph Neural Networks (GNNs) for cross-shard anomaly detection and response propagation modelling.

14. REFERENCES

[1] Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. *arXiv preprint* arXiv:2203.02155.

[2] Vishwarupe, V., Zahoor, S., Akhter, R., Bhatkar, V. P., Bedekar, M., Pande, M., Joshi, P. M., Patil, A., & Pawar, V. (2023). Designing a human-centered AI-based cognitive learning model for Industry 4.0 applications. In *Industry 4.0 Convergence with AI, IoT, Big Data and Cloud Computing: Fundamentals, Challenges and Applications* (pp. 84–95). Bentham Science Publishers.

[3] Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*(3–4), 211–407.

[4] Sayyed, H., Alwazae, M., & Vishwarupe, V. (2025). BlockSafe: Universal blockchain-based identity management. In B. Alareeni (Ed.), *Big Data in Finance: Transforming the Financial Landscape* (Vol. 169, pp. 101–118). Springer. https://doi.org/10.1007/978-3-031-80656-8_6

[5] Vishwarupe, V., Maheshwari, S., Deshmukh, A., Mhaisalkar, S., Joshi, P. M., & Mathias, N. (2022). Bringing humans at the epicentre of artificial intelligence: A confluence of AI, HCI, and human-centered computing. *Procedia Computer Science, 204*, 914–921. https://doi.org/10.1016/j.procs.2022.08.111

[6] Rayson Laroca, R., Severo, E., Zanlorensi, L., Oliveira, L., Gonçalves, G., Schwartz, W., & Menotti, D. (2018). A robust real-time automatic license plate recognition based on the YOLO detector. *arXiv preprint* arXiv:1802.09567.

[7] Vishwarupe, V., Bedekar, M., Pande, M., & Hiwale, A. (2018). Intelligent Twitter spam detection: A hybrid approach. In X. S. Yang, A. Nagar, & A. Joshi (Eds.), *Smart trends in systems, security and sustainability* (Vol. 18, pp. 157–167). Springer. https://doi.org/10.1007/978-981-10-6916-1_17

[8] T. Li, Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine, 37*(3), 50–60.

[9] Vishwarupe, V., Joshi, P. M., Mathias, N., Maheshwari, S., Mhaisalkar, S., & Pawar, V. (2022). Explainable AI and interpretable machine learning: A case study in perspective. *Procedia Computer Science, 204*, 869–876. https://doi.org/10.1016/j.procs.2022.08.105

[10] The Syslog Protocol. (2001/2009). RFC 3164/5424, Internet Engineering Task Force (IETF).

[11] Wani, K., Khedekar, N., Vishwarupe, V., & Pushyanth, N. (2023). Digital twin and its applications. In *Research Trends in Artificial Intelligence: Internet of Things* (pp. 120–134). Bentham Science Publishers.

[12] Xie, C., Koyejo, O., & Gupta, I. (2020). Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[13] Vidgen, B., Harris, A., & Emmery, C. (2021). Challenges and frontiers in abusive content detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

[14] Vishwarupe, V., Bedekar, M., Pande, M., Bhatkar, V. P., Joshi, P., Zahoor, S., & Kuklani, P. (2022). Comparative analysis of machine learning algorithms for analyzing NASA Kepler mission data. *Procedia Computer Science, 204*, 945–951. https://doi.org/10.1016/j.procs.2022.08.115

[15] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[16] Vishwarupe, V. (2022, February 10). Synthetic content generation using artificial intelligence. *All Things Policy*. IVM Podcasts. https://shows.ivmpodcasts.com/show/all-things-policy-Rx64RVpQImivrNQ8/episode/synthetic-content-generation-and-chinas-worries-ja9s-I7rfgZE3IhXRg2Fk

[17] Kairouz, P., McMahan, H. B., et al. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning, 14*(1–2), 1–210.

[18] Sable, N. P., Rathod, V. U., Mahalle, P. N., & Birari, D. R. (2022, March). A multiple stage deep learning model for NID in MANETs. In *2022 International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 1–6). IEEE.

[19] Common Event Format (CEF); JSON Logging Standards. ArcSight.

[20] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal, 3*(5), 637–646.

[21] Zahoor, S., Bedekar, M., Mane, V., & Vishwarupe, V. (2016). Uniqueness in user behavior while using the web. In S. Satapathy, Y. Bhatt, A. Joshi, & D. Mishra (Eds.), *Proceedings of the International Congress on Information and Communication Technology* (Vol. 438, pp. 229–236). Springer. https://doi.org/10.1007/978-981-10-0767-5_24

[22] Vishwarupe, V., Bedekar, M., & Zahoor, S. (2015). Zone-specific weather monitoring system using crowdsourcing and telecom infrastructure. In *2015 International Conference on Information Processing (ICIP)* (pp. 823–827). IEEE. https://doi.org/10.1109/INFOP.2015.7489495

[23] Zahoor, S., Bedekar, M., & Vishwarupe, V. (2016). A framework to infer webpage relevancy for a user. In S. Satapathy & S. Das (Eds.), *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1* (Vol. 50, pp. 173–181). Springer. https://doi.org/10.1007/978-3-319-30933-0_16

[24] Gehman, S., Gururangan, S., Sap, M., et al. (2020). RealToxicityPrompts: Evaluating neural toxic degeneration in language models. *arXiv preprint* arXiv:2009.11462.

[25] Zhang, M., Cao, J., Shen, X., & Cui, Z. (2024). EdgeShard: Efficient LLM inference via collaborative edge computing. *arXiv preprint* arXiv:2405.14371.

[26] Deoskar, V., Pande, M., & Vishwarupe, V. (2024). An analytical study for implementing 360-degree M-HRM practices using AI. In *Intelligent Systems for Smart Cities: Select Proceedings of the 2nd International Conference, ICISA 2023* (pp. 429–442). Springer Nature.

[27] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., & Ferrari, V. (2020). The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint* arXiv:1811.00982.

[28] Vishwarupe, V., et al. (2021). A zone-specific weather monitoring system. *Australian Patent No. AU2021106275*. Australian Government, IP Australia. https://patents.google.com/?inventor=Varad+Vishwarupe

[29] Vishwarupe, V., Bedekar, M., Joshi, P. M., Pande, M., Pawar, V., & Shingote, P. (2022). Data analytics in the game of cricket: A novel paradigm. *Procedia Computer Science, 204*, 937–944. https://doi.org/10.1016/j.procs.2022.08.114

[30] Vishwarupe, V. V., & Joshi, P. M. (2016). Intellert: A novel approach for content-priority based message filtering. In *2016 IEEE Bombay Section Symposium (IBSS)* (pp. 1–6). IEEE. https://doi.org/10.1109/IBSS.2016.7940206

[31] Vishwarupe, V., et al. (2025). Predicting mental health ailments using social media activities and keystroke dynamics with machine learning. In B. Alareeni (Ed.), *Big Data in Finance: Transforming the Financial Landscape* (Vol. 169, pp. 63–80). Springer. https://doi.org/10.1007/978-3-031-80656-8_4

[32] Zahoor, S., Akhter, R., Vishwarupe, V., Bedekar, M., Pande, M., Bhatkar, V. P., Joshi, P. M., Pawar, V., Mandora, N., & Kuklani, P. (2023). A comprehensive study of state-of-the-art applications and challenges in IoT and blockchain technologies for Industry 4.0. In *Industry 4.0 Convergence with AI, IoT, Big Data and Cloud Computing: Fundamentals, Challenges and Applications* (pp. 1–16). Bentham Science Publishers.