Journal of Computer Technology and Software
ISSN:2998-2383
Vol. 5, No. 4, 2026

From Pre-training to Deployment: A Review of Large Language Model Technologies and Applications

Elowen Mercer
University of Waterloo, Waterloo, Canada
elowen82@gmail.com

Abstract: Large language models have reshaped natural language processing and are increasingly used as general interfaces for knowledge work. This review summarizes their architectural foundations, pre-training and post-training methods, application patterns, evaluation practices, and governance challenges. The literature shows that progress depends not only on parameter scale, but also on data quality, compute allocation, instruction tuning, preference optimization, retrieval augmentation, and deployment discipline. The paper also discusses hallucination, bias, privacy, cost, and safety risks, and identifies future directions in multimodal modeling, agentic systems, domain adaptation, and trustworthy governance.

Keywords: large language model; Transformer; instruction tuning; retrieval augmented generation; AI safety

1. Introduction

Large language models, or LLMs, have become one of the most influential research and engineering directions in artificial intelligence. They are trained on very large text corpora, and increasingly on multimodal data, to predict, reconstruct, or rank language sequences. This apparently simple training objective produces systems that can summarize documents, answer questions, generate code, translate between languages, extract structured information, and participate in extended dialogue. The practical importance of LLMs lies in their generality. Instead of building a separate model for every natural language processing task, users can often describe the task in ordinary language and let a single foundation model adapt its behavior through context.

The rise of LLMs is not the result of scale alone. It reflects the combination of the Transformer architecture, large-scale self-supervised pre-training, instruction tuning, preference optimization, retrieval augmentation, and extensive systems engineering. The Transformer made it possible to train sequence models efficiently while capturing long-range dependencies through self-attention [1]. BERT demonstrated the value of pre-trained contextual representations for many understanding tasks [2]. GPT-3 showed that autoregressive language models can perform zero-shot and few-shot learning when trained at sufficient scale [3]. Later work on instruction following and reinforcement learning from human feedback shifted the field from raw completion models toward usable assistants [4].

This paper reviews the major technical components, capabilities, risks, and future directions of LLM research. The review emphasizes the full lifecycle of an LLM system: model architecture, data and compute, post-training, deployment, evaluation, and governance. This perspective is important because LLM quality is determined by interactions among these elements. A model with strong pre-training can still be unreliable if it lacks retrieval, safety testing, or domain adaptation. Conversely, a smaller model can become useful in a narrow setting if it is supported by high-quality data, efficient fine-tuning, and task-specific validation.

2. Technical Foundations

2.1 Transformer Architecture

The Transformer replaced recurrent sequence processing with attention-based computation. In self-attention, each token representation is compared with other token representations through query, key, and value projections. The resulting attention weights allow the model to aggregate information from relevant positions in the sequence. Multi-head attention lets the model learn several relation patterns in parallel, such as syntactic agreement, coreference, topical continuity, or local phrase structure. Feed-forward layers then transform the aggregated representations into more expressive features. Because the main operations are matrix multiplications, the architecture is well matched to modern accelerators.

The Transformer is also modular. Encoder-only models such as BERT are suitable for representation learning and classification. Decoder-only models such as GPT are suitable for open-ended generation. Encoder-decoder models support conditional generation tasks such as translation and summarization. Modern LLMs are most often decoder-only because next-token prediction naturally supports interactive generation, tool use, and completion of arbitrary textual contexts. The same architecture can be extended with longer context windows, sparse attention, mixture-of-experts layers, and multimodal input adapters.

2.2 Pre-training Objectives and Data

Pre-training teaches a model broad statistical regularities of language before it is adapted to user tasks. Autoregressive pre-training predicts the next token given previous tokens. Masked language modeling reconstructs hidden tokens from bidirectional context. The choice of objective shapes the model's behavior: autoregressive models are strong generators, while masked models have historically been strong encoders for understanding tasks. In both cases, the main advantage is that no manually labeled dataset is required at web scale.

Data quality has become as important as data quantity. Large corpora contain duplicates, boilerplate, low-quality pages, biased language, private information, and sometimes benchmark contamination. Data curation therefore includes deduplication, language identification, toxicity filtering, quality scoring, code and math balancing, and domain mixture design. A training set that is larger but noisier may not improve the final model. For this reason, current LLM development treats data pipelines as a central research object rather than a simple preprocessing step.

2.3 Scaling Laws and Compute Allocation

Scaling laws describe how loss and downstream performance change as model size, dataset size, and compute increase. Early LLM development often emphasized parameter count, but later research showed that compute-optimal training requires a more balanced relationship between parameters and training tokens. The Chinchilla study found that many large models were undertrained relative to their size, and that a smaller model trained on more data can outperform a larger model trained on fewer tokens under a similar compute budget [5].

This result has several implications. First, model quality should not be judged only by the number of parameters. Second, the marginal value of additional data depends on its quality and distribution. Third, inference cost matters after deployment. A model that is expensive to serve may be impractical even if it is strong on benchmarks. Current systems therefore use quantization, batching, caching, distillation, and specialized serving infrastructure to make LLMs usable at scale.

2.4 Post-training and Alignment

A pre-trained model learns to continue text, not necessarily to follow instructions. Post-training changes the model's behavior so that it becomes more helpful, controllable, and safe in conversation. Supervised instruction tuning trains the model on examples of prompts and desirable responses. Reinforcement learning from human feedback then collects human preference comparisons, trains a reward model, and optimizes the policy to produce preferred outputs [4]. This pipeline was a major step toward assistant-like LLMs.

Preference optimization remains an active area. RLHF can be costly and unstable because it requires reward modeling, sampling, reinforcement learning, and careful regularization. Direct Preference Optimization reformulates preference learning into a simpler objective that can train the language model directly on preference pairs while implicitly representing the reward function [6]. Constitutional AI explores the use of written principles and AI feedback to reduce harmful behavior when direct human labeling is limited [11]. These methods show that alignment is not a single filter added after training. It is a continuous process of shaping model incentives, data, evaluation, and deployment constraints.

3. Capabilities and Applications

3.1 Language Understanding and Generation

The most visible capability of LLMs is fluent language generation. Models can produce summaries, explanations, emails, reports, tables, and code comments that follow user-specified tone and structure. They can also classify text, extract entities, transform unstructured notes into structured records, and answer questions from supplied context. These capabilities are valuable because they reduce the interface cost between human intentions and digital workflows.

However, fluency should not be confused with truth. LLMs generate likely continuations, and the likely continuation may be factually wrong. This problem is especially serious when the model is asked about recent events, niche technical details, or documents outside its context. A responsible application therefore treats generation as one component in a larger system that includes retrieval, citation, validation, and human review.

3.2 Reasoning and Code Generation

LLMs can perform multi-step reasoning, especially when prompts encourage them to decompose a problem. Chain-of-thought prompting, self-consistency, and tool-assisted reasoning can improve performance on arithmetic, symbolic, and planning tasks. Code generation is another major application. Since code has formal syntax and can be executed, model outputs can be checked through tests and static analysis. This creates a feedback loop that is more reliable than natural-language evaluation alone.

Even so, reasoning remains fragile. A model may use a plausible but invalid shortcut, ignore a boundary condition, or produce an answer that matches a familiar pattern without satisfying the actual constraints. In software engineering, LLMs can accelerate exploration, refactoring, documentation, and test creation, but they still need repository context and verification. The strongest workflow is not blind automation. It is human-machine collaboration in which the model proposes, the toolchain verifies, and the user makes final decisions.

3.3 Retrieval-Augmented Generation

Retrieval-augmented generation, or RAG, combines an LLM with an external knowledge source. A typical RAG system chunks documents, embeds the chunks, stores them in a vector or hybrid index, retrieves relevant passages for each query, and provides those passages to the model as context. The original RAG formulation showed that combining parametric generation with non-parametric retrieval can improve knowledge-intensive tasks [7].

RAG is attractive in enterprise settings because it allows models to answer questions over private and changing

knowledge bases without retraining the base model. It also supports citation and auditability when retrieved passages are shown to the user. The main engineering challenges are retrieval quality, chunk design, ranking, context compression, contradictory sources, and access control. A poor retriever can cause a strong model to answer from irrelevant evidence. A good RAG system therefore requires evaluation at both the retrieval layer and the generation layer.

3.4 Efficient Fine-tuning and Open Models

Full fine-tuning of a large model is expensive because it updates all parameters. Parameter-efficient methods reduce this cost by training only a small number of additional parameters. LoRA freezes the pre-trained weights and injects low-rank trainable matrices into Transformer layers, reducing the number of trainable parameters and memory requirements while preserving quality in many settings [8]. This makes domain adaptation more accessible to research groups and companies that cannot train a foundation model from scratch.

Open foundation models have also changed the research ecosystem. Llama 2 released a family of pre-trained and chat-tuned models from 7 billion to 70 billion parameters [9]. Open-weight models support reproducible research, local deployment, privacy-sensitive applications, and specialized fine-tuning. They also increase governance challenges because capable models can be modified and redistributed. The field must therefore balance openness, safety, transparency, and misuse prevention.

4. Evaluation and Governance

4.1 Model Evaluation

LLM evaluation includes benchmark scores, human preference studies, task-specific tests, and deployment monitoring. Benchmarks are useful because they provide repeatable comparisons, but they can be contaminated by training data or overfit by repeated optimization. Human evaluation captures helpfulness, coherence, and style, but it is expensive and subjective. Application evaluation is often the most meaningful: a generated program should pass tests, a retrieval answer should cite valid evidence, and a customer support response should solve the user's issue without escalation.

The GPT-4 technical report illustrates both the potential and the difficulty of evaluation. GPT-4 showed strong performance across many academic and professional benchmarks and accepted both image and text inputs, but the report also emphasized safety testing and limitations [10]. This reflects a broader lesson: model capability, robustness, and safety must be evaluated together. A model that performs well on average can still fail in rare but consequential cases.

4.2 Hallucination and Reliability

Hallucination occurs when a model generates unsupported or false information. It is not merely a bug in a particular model, but a consequence of optimizing for plausible language rather than verified truth. Hallucination can be reduced but not fully eliminated through better data, retrieval, calibration, abstention behavior, and post-generation checking. In high-stakes domains such as medicine, law, finance, and public policy, LLMs should be used as assistive systems rather than autonomous authorities.

A practical reliability strategy uses layered safeguards. The model should be instructed to cite sources when evidence is required. Retrieval should be scoped to trusted documents. Outputs should be checked against rules, tests, or databases. Users should see uncertainty when the evidence is incomplete. Logs should support audit and error analysis. These measures do not make LLMs perfect, but they make failure more observable and manageable.

4.3 Bias, Privacy, and Security

LLMs inherit risks from their training data and deployment environment. Training corpora may contain stereotypes, toxic content, copyrighted material, and private information. During interaction, users may provide sensitive data that should not be stored or exposed. Attackers may use prompt injection, data exfiltration attempts, or jailbreaks to bypass safety controls. The risk increases when a model has access to tools, internal documents, or external systems.

Governance therefore requires both technical and organizational controls. Technical controls include red teaming, content filtering, data minimization, access control, sandboxed tool execution, and monitoring. Organizational controls include policy, user training, incident response, documentation, and accountability. Constitutional AI and related methods can reduce harmful outputs by training models against explicit principles [11], but no alignment method replaces careful deployment design.

4.4 Cost and Sustainability

The cost of LLMs includes training compute, serving infrastructure, energy consumption, data engineering, evaluation, and human oversight. Training a frontier model requires large accelerator clusters and specialized distributed systems. Serving the model can become even more expensive over time because every user interaction consumes inference compute. Long context windows, high concurrency, and tool-augmented workflows further increase cost.

Future systems will likely use a mixture of models rather than a single universal model. Large models will handle complex reasoning and open-ended tasks. Smaller models will handle routing, extraction, classification, and local privacy-sensitive workloads. Caching, quantization, distillation, speculative decoding, and retrieval can reduce cost. The key engineering question is how to allocate computation only where it produces real value.

Recent work broadens the evidence base for LLM systems beyond static text. Graph neural and Transformer models for multivariate time series, together with incremental modeling for evolving distributed streams, show how temporal context and changing data distributions can be represented explicitly [13], [14]. Multi-feature recurrent modeling for sentiment classification further illustrates that task-specific feature fusion remains useful when domain language is sparse or noisy [15].

Application-facing learning also benefits from objectives that respect decision context. Causal consistency-aware recommendation, hierarchical multi-source fraud detection, and one-sided conformal prediction for cloud warehouse right-sizing demonstrate complementary ways to account for correlation bias, heterogeneous evidence, and asymmetric operational loss [16], [17], [18].

For backend and cloud-facing deployments, temporal-structural representations can connect service behavior with fault discrimination, while active data curation can reduce the cost of building useful training sets [19], [20]. These concerns are particularly relevant when an LLM is part of a monitored production workflow rather than a standalone text interface.

RAG research is also moving from simple retrieval pipelines toward coordinated retrieval, reranking, conflict detection, and evidence integration. Dual retrieval and adaptive reranking, long-text reasoning with semantic conflict detection, and causal graph-augmented retrieval for root-cause analysis provide useful design patterns for evidence-sensitive answers [21], [22], [25]. Workload-aware database tuning and retrieval-augmented log-metric-trace twins extend this pattern to operational decision support and diagnosis [28], [32].

Reliable use requires evaluation mechanisms that match the deployment surface. Requirement-aware testing for backend APIs, causal cue modeling with factual correction for summarization, and topology-aware discrimination of microservice anomalies make verification more concrete for language-enabled systems [26], [29], [33]. Related studies of graph-based fault identification, cloud security anomaly detection, and resilient backend configuration emphasize continuous monitoring and recovery rather than one-time benchmark performance [34], [35], [31].

Efficient deployment is increasingly a systems problem. Adaptive model compression across the edge-cloud continuum, KV-cache-aware Kubernetes routing, risk-constrained elastic scaling, semantic-aware routing across model gateways, hierarchical federated anomaly detection, resource-contention classification, and reinforcement-learning scheduling offer complementary controls for throughput, latency, cost, and privacy [27], [30], [39], [40], [44], [45], [46].

Several applied studies also underline the value of structured evidence and memory in specialized settings. Spatiotemporal attention supports collaborative demand forecasting [23], embedding and sequence models support fine-grained risk detection [24], and structured summarization and contrastive analysis strengthen financial-document workflows [36], [37]. Relation-aware graph reasoning for fraud detection, privacy-preserving multi-agent collaboration, evidence-gated memory writing, and knowledge-graph-enhanced sentiment classification point toward LLM systems that retain traceable context while adapting to domain-specific interaction needs [38], [41], [42], [43].

5. Future Directions

First, LLMs are moving toward multimodal foundation models. Text, image, audio, video, and structured data can be processed through unified interfaces. This expands the range of applications, from document understanding to scientific analysis and robotics, but it also creates new safety problems because errors can move across modalities.

Second, LLMs are becoming agents that plan, call tools, and act over longer horizons. Agentic systems can search, write code, query databases, operate software, and update files. Their value depends on reliable state tracking, permission boundaries, verification, and recovery from failure. The research challenge is to make such systems useful without allowing small errors to compound into large failures.

Third, domain-specific systems will become more important. Many organizations do not need a general model to answer every possible question. They need a model that understands their documents, terminology, workflows, and compliance constraints. RAG, efficient fine-tuning, and human feedback can create such systems more cheaply than training new foundation models.

Fourth, governance will become part of model quality. As LLMs enter education, healthcare, finance, law, and public services, transparency, documentation, safety evaluation, and accountability will matter as much as benchmark scores. A mature LLM ecosystem must measure not only what models can do, but also when they should not be used.

6. Conclusion

Large language models are best understood as socio-technical systems built from architecture, data, compute, post-training, retrieval, evaluation, and governance. Their progress has been driven by the Transformer, large-scale pre-training, improved compute allocation, instruction tuning, preference optimization, and open model ecosystems. They already support many applications in writing, coding, education, enterprise search, and knowledge work.

At the same time, LLMs remain limited by hallucination, bias, privacy risk, high cost, and incomplete interpretability. These limitations do not make LLMs useless; rather, they define the conditions under which they should be deployed. The most promising path is to combine capable models with trusted data, external tools, explicit evaluation, and human accountability. In this sense, the future of LLM research is not only about larger models, but about more reliable, efficient, and governable intelligent systems.

References

[1] A. Vaswani et al., "Attention is all you need," in Proc. Advances in Neural Information Processing Systems, 2017.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805, 2018.

[3] T. B. Brown et al., "Language models are few-shot learners," in Proc. Advances in Neural Information Processing Systems, 2020.

[4] L. Ouyang et al., "Training language models to follow instructions with human feedback," arXiv:2203.02155, 2022.

[5] J. Hoffmann et al., "Training compute-optimal large language models," arXiv:2203.15556, 2022.

[6] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," arXiv:2305.18290, 2023.

[7] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Advances in Neural Information Processing Systems, 2020.

[8] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," arXiv:2106.09685, 2021.

[9] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv:2307.09288, 2023.

[10] OpenAI et al., "GPT-4 technical report," arXiv:2303.08774, 2023.

[11] Y. Bai et al., "Constitutional AI: Harmlessness from AI feedback," arXiv:2212.08073, 2022.

[12] W. X. Zhao et al., "A survey of large language models," arXiv:2303.18223, 2023.

[13] Wang, J. (2024). Multivariate time series forecasting and classification via GNN and Transformer models. Journal of Computer Technology and Software, 3(9).

[14] Wang, Y. (2024). AI-Enhanced Distributed Time Series Modeling: Incremental Learning for Evolving Streaming Data. Transactions on Computational and Scientific Methods, 4(8).

[15] Su, C., Jiang, L., & Pi, S. (2021, August). Sentiment classification of Chinese railway review text based on multi-feature fusion gated recurrent unit. In 2021 International Conference on Information Science, Parallel and Distributed Systems (ISPDS) (pp. 197-202). IEEE.

[16] Li, S., Wang, Y., Xing, Y., & Wang, M. (2025, December). Mitigating correlation bias in advertising recommendation via causal modeling and consistency-aware learning. In Proceedings of the 2025 6th International Conference on Computer Science and Management Technology (pp. 585-589).

[17] Ke, Z. (2026). Right-Sizing Cloud Data Warehouses Under Asymmetric Loss via One-Sided Conformal Prediction. IEEE Access, 14, 94748-94763.

[18] Wang, J. (2025). Credit card fraud detection via hierarchical multi-source data fusion and dropout regularization. Transactions on Computational and Scientific Methods, 5(1).

[19] Xue, Y., Wang, Y., Zhu, A., Sun, X., & Zhang, C. (2026, March). Joint Temporal-Structural Representation Learning for Distributed Fault Discrimination in Microservice Architectures. In 2026 IEEE 8th International Conference on Communications, Information System and Computer Engineering (CISCE) (pp. 1581-1586). IEEE.

[20] Sun, S., Wang, Y., Yan, R., Wang, J., & Lu, Y. (2026). CurateDistill: Efficient Data Selection via Active Data Curation and Dataset Distillation. DOI: 10.13140/RG.2.2.27464.28166.

[21] Mao, L. (2024). Joint Optimization of Dual Retrieval and Adaptive Re-Ranking for Retrieval-Augmented Generation. Artificial Intelligence and Computing Innovations, 4(4).

[22] Huang, S. Y. (2024). Retrieval-Augmented Long-Text Reasoning with Semantic Conflict Detection and Cross-Paragraph Evidence Integration. Artificial Intelligence and Computing Innovations, 4(1).

[23] Xu, Y., Kou, J., Wang, W., Su, Z., Yan, X., & Wang, J. (2026, March). Graph spatiotemporal attention-based collaborative demand forecasting for multi-level supply chains. In Proceedings of the 2026 3rd International Conference on Innovation Management and Information Systems (pp. 214-220).

[24] Su, Z. (2024). Microstructure-Aware Risk Detection in High-Frequency Trading Using Embedding and Sequence Modeling. Transactions on Computational and Scientific Methods, 4(3).

[25] Kou, J., Qing, Z., Wei, R., Xue, F., Huang, W., & Zhuang, H. (2026, April). CausalRCA: Causal Graph-Augmented Retrieval for Root Cause Analysis in Microservice Systems. In 2026 5th International Conference on Electronic Information Engineering, Big Data and Computer Technology (EIBDCT) (pp. 440-444). IEEE.

[26] Xue, F., Ying, T., Ding, T., Wang, L., Qing, Z., & Hu, P. (2026). Requirement-Aware LLM Testing for Backend REST APIs. DOI: 10.13140/RG.2.2.21048.30729.

[27] Hu, P., Qing, Z., Ding, T., Ying, T., Xue, F., & Wang, L. (2026). Energy-Efficient Distributed Inference across the Edge-Cloud Continuum Using Adaptive Model Compression. DOI: 10.13140/RG.2.2.19323.07209.

[28] Lyu, Y., Jiang, J., & Hu, J. (2026). QueryMind: Workload-Aware LLM-Guided Database Configuration Tuning for Backend Performance.

[29] Hu, J., Lyu, Y., & Jiang, J. (2026). High-Reliability LLM Summarization through Causal Cue Modeling and Factual Correction.

[30] Pi, S., Zhang, S., & Shawulieti, S. (2026). CacheRoute: KV-Cache-Aware Routing for LLM Backend Services on Kubernetes. DOI: 10.13140/RG.2.2.34191.55206.

[31] Wang, S., Su, B., Liu, Y., Pi, S., & Zhu, A. (2026). Safe Adaptive Resilience Configuration for Microservice Backends. DOI: 10.13140/RG.2.2.30702.57921.

[32] Liu, Z., Wu, K., Dang, S., & Yang, Y. (2026). A Retrieval-Augmented Log-Metric-Trace Twin for Backend Failure Diagnosis. DOI: 10.13140/RG.2.2.34050.64969.

[33] Liu, Z., Yang, Y., Chen, C., Dang, S., & Wu, K. (2026). Topology Dependency Modeling and Discriminative Representation Learning Methods for Backend Microservice Call Anomaly Detection Task. DOI: 10.13140/RG.2.2.18843.02082.

[34] Shawulieti, S. (2024). Enhancing Cloud Security with Graph Neural Network-Driven Anomaly Detection in Multi-Tenant Environments. Artificial Intelligence and Computing Innovations, 4(2).

[35] Xue, F., Wu, K., & Abdulla, A. (2025). Topology-Aware Graph Neural Networks for Distributed Node Fault Identification. Transactions on Computational and Scientific Methods, 5(3).

[36] Wang, M., Peng, P. L., & Chen, J. (2024). Financial Report Summarization via Structure-Aware Modeling and Information Fidelity Constraints. Transactions on Computational and Scientific Methods, 4(7).

[37] Peng, Z. (2024). Cross-Period Financial Statement Anomaly Detection via Contrastive Representation Learning. Journal of Computer Technology and Software, 3(8).

[38] Su, Z., Chen, J., Meng, X., Sang, N., & Peng, Z. (2026). Relation-Aware Graph Neural Networks with Verifiable Reasoning for Accounting Fraud Detection. ResearchGate Preprint. DOI: 10.13140/RG.2.2.13631.06565.

[39] Hu, P., Uziel, G., & Ma, X. (2024). Risk-Constrained Reinforcement Learning for SLO-Driven Cloud-Native Elastic Scaling. Journal of Computer Technology and Software, 3(1).

[40] Jiang, Z., Zhang, S., Huang, W., Pi, S., Shawulieti, S., & Xu, S. (2026). Semantic-Aware Intelligent Request Routing in Multi-Model LLM Backend Gateways.

[41] Jiang, Y., Xiao, Z., Lin, O., Gu, Z., & Yang, Z. (2026). PrivCollab-Agent: Privacy-Preserving Multi-Agent Collaboration under Partial Information. DOI: 10.13140/RG.2.2.33663.16800.

[42] Ma, L., Song, T., Long, R., Mao, L., & Gu, Z. (2026, May). Evidence-Gated Memory Writing for Personalized Large Language Model Agents. In 2026 3rd International Conference on Image Processing and Artificial Intelligence (ICIPAI) (pp. 333-338). IEEE.

[43] Gu, Z., & Nie, L. (2025). Knowledge Graph-Enhanced Large Language Models for Opinion Target Sentiment Classification. Journal of Computer Technology and Software, 4(7).

[44] Li, S., Chen, B., Li, Y., Wang, Z., Xue, Y., & Xu, C. (2026). Privacy-Preserving Anomaly Detection in Cloud Services Using Hierarchical Federated Learning With Differential Privacy. In 2026 9th International Symposium on Big Data and Applied Statistics (ISBDAS) (pp. 668-672).

[45] Lyu, N., Chen, F., Zhang, C., Shao, C., & Jiang, J. (2025, December). Deep temporal convolutional neural networks with attention mechanisms for resource contention classification in cloud computing. In Proceedings of the 2025 5th International Conference on Big Data, Artificial Intelligence and Risk Management (pp. 62-68).

[46] Zhang, C. (2024). Adaptive Multi-Tenant Resource Scheduling in Cloud Computing via Reinforcement Learning. Transactions on Computational and Scientific Methods, 4(3).