ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646/ Volume 6 Issue 4 October-December 2026
Paper Id: JETA-V6I4P101/ Page No: 01-09

Original Article

Predictive Infrastructure Drift Detection Framework for Cloud-Native Carrier Network Reliability Engineering

Prabhudoss Jayakumar
University of Madras, Chennai, India

Received Date: 25 September 2026 Revised Date: 30 September 2026 Accepted Date: 05 October 2026

Abstract: *Cloud-native carrier networks rely on declarative infrastructure, virtualized network functions, distributed orchestration, and dynamic service topologies. Divergence among intended configuration, deployed state, and runtime behaviour may create reliability risks that traditional alarms detect only after service degradation begins. This review examines the literature relevant to predictive infrastructure drift detection and links infrastructure-as-code analysis, network anomaly detection, automated log analysis, concept-drift detection, artificial-intelligence-driven operations and carrier-network automation. Existing research demonstrates strong capabilities for retrospective anomaly detection, failure localization, and workload forecasting, but provides limited evidence for predicting harmful divergence before operational impact occurs. Most existing approaches treat intended state, runtime state, telemetry, topology, and service-level objectives as largely independent analytical units. Carrier-specific validation remains very limited because public datasets do not typically contain configuration histories, dependency graphs, orchestration events and service consequences within a common timeline. The predictive framework needs to incorporate temporal state reconciliation, topology-aware feature construction, risk-driven forecasting with uncertainty calibration, and intervention policies. Advances in this area will help mitigate the problems of silent degradation, incorrect automated remediation, and loss of reliability in programmable carrier infrastructure systems.*

Keywords: *Aiops, Carrier Networks, Cloud-Native Reliability, Concept Drift, Infrastructure Drift, Predictive Detection*

I. INTRODUCTION

Carrier-network infrastructure is being virtualized, controlled by software, orchestrated by containers, managed by service meshes and declared in configuration repositories. The advantage of network-function virtualization is that it allows the separation of packet-processing functions from dedicated appliances, thereby affording significant operational flexibility, as well as expanding the surface area of configuration related to placement, chaining, scaling, acceleration, and lifecycle management [1]. Software-defined networking (SDN) provides logically centralized control and programmable forwarding; however, consistency must be maintained among controller intent, installed forwarding rules, discovered topology, and device state [2]. Network slicing introduces another coordination layer because isolation, resource allocation, service assurance, and orchestration policies must remain mutually consistent across shared physical and virtual infrastructures [3]. Machine-learning (ML)-based network management can identify patterns that are not captured by fixed rules, but the quality of the model is critically dependent on the stability of data semantics, varied traces used for training, and accurate labels for operation in the field [4].

In this context, infrastructure drift refers to divergence over time among service-level operating conditions, runtime state, deployed configuration, and authorized desired state. Skew can occur due to emergencies, rollout failures, defects in controller reconciliation, manual actions, stale dependencies, policy conflicts, or environment-specific defaults. Conventional configuration comparison will detect an existing mismatch but not determine if the mismatch is transient, benign, self-correcting, or is likely to cause a carrier-grade incident. Predictive drift detection, on the other hand, estimates the likelihood, extent and likely trajectory of harmful drift before a service level is breached. In carrier environments, this separation is important as drift can affect routing convergence, autoscaling policies, slice-isolation controls, stateful network functions, and redundancy mechanisms.

The research field of relevance to this problem is rife with fragmentation. Research on infrastructure-as-code focuses on defects, maintainability, security smells, and deployment automation, while operational anomaly research focuses on telemetry after deployment. Concept drift research investigates variability in data distribution and rarely considers explicit difference between desired and actual infrastructure state. While AIOps research brings together heterogeneous monitoring data, it typically measures incident detection or root cause localization but not forecasting risks of configurations. Carrier-network automation research focuses on resource prediction and closed-loop optimization, but configuration provenance and change causality are not as well discussed as traffic demand or policy enforcement.

This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

The central gap, therefore, is not the absence of anomaly-detection techniques. The gap is that there is no empirically validated framework that connects declarative intent, infrastructure changes, runtime topology, multivariate telemetry, and resulting carrier-service consequences. None of the articles in the reviewed journal corpus evaluates this whole chain in an operational cloud-native carrier network. The aim of this review is to evaluate the most relevant journal articles published, compare the methodological families, identify common findings, and establish a topic-specific framework that can be used in predictive infrastructure drift detection, without equating anomaly detection, configuration differencing or concept drift.

II. LITERATURE REVIEW

Research into infrastructure-as-code reveals that automated configuration is no longer a secondary deployment tool and has become a key operational artifact. The systematic mapping undertaken by Rahman et al. revealed a field with substantial testing activity, an industrial focus, limited industrial replication, and various quality measures that were not well agreed upon, but also indicated a field that was growing rapidly but methodologically uneven [5]. The main conclusion for predictive carrier-network drift detection is that declarative repositories contain good evidence of the intended behaviour, but they cannot be used to assert runtime realization. A valid detector should be able to distinguish between a defective declaration and a declaration that is divergent in deployment, an environmental override, a mutation of the controller and operationally necessary adaptation.

Table 1: Summary of Key Findings

Ref	Focus	Key Findings
[5]	Infrastructure-as-code research	Declarative automation supports testing and repeatability, but empirical standards, runtime linkage, and industrial replication remain limited.
[6]	Network anomaly detection	Statistical and machine-learning methods improve anomaly coverage, although false positives, adaptability, and feature quality remain persistent constraints.
[7]	Automated log analysis	Parsing and sequence analysis support anomaly identification, but template instability and incomplete event semantics reduce transferability.
[8]	AIOps failure management	Multisource monitoring enables prediction, correlation, diagnosis, and remediation; evaluation remains fragmented and commonly proprietary.
[9]	Cloud-microservice anomaly detection and root-cause analysis	Topology and causal reasoning improve localization, but public benchmarks and reproducible end-to-end validation remain scarce.
[10]	Deep anomaly detection	Learned representations capture nonlinear patterns but remain vulnerable to contaminated training data, scarce labels, and shifting normality.
[11]	Unified anomaly-detection theory	Deep and shallow models often implement related one-class, reconstruction, or density objectives with comparable assumption risks.
[12]	Learning under concept drift	Detection and adaptation must account for changing distributions, model aging, delayed labels, and recurring regimes.
[13]	Comparative concept-drift detection	Detector rankings depend on drift form, evaluation metric, stream construction, and acceptable detection delay.
[14]	Drift characterization	Distributional change requires explicit characterization of affected variables, severity, duration, and decision-boundary impact.
[15]	Multi-access edge orchestration	Distributed placement and low-latency control increase state-coordination demands across heterogeneous edge resources.
[16]	Predictive sliced-network management	Demand forecasting can enable proactive resource control, but configuration provenance and divergence attribution remain outside the prediction target.

The methodological foundations are given by the network and cloud anomaly research. In [6] Ahmed, Mahmood, and Hu presented a taxonomy of statistical, knowledge-based, and machine-learning approaches to network anomaly detection, focusing on the impact on the dimensions of adaptability, computational load, and false-alarm control. Automated log-analysis research has produced pipelines for log parsing, representation, anomaly detection, and diagnostic interpretation, while also demonstrating sensitivity to evolving message templates, incomplete logging, and changing labels [7]. While AIOps research broadened the range of analytics to cover multimodal monitoring, failure prediction, incident correlation and automated remediation, the focus on what constitutes evaluation remained dominated by proprietary environments and heterogeneous success criteria [8]. Surveys of microservice-based cloud applications also found substantial research on anomaly detection and root-cause analysis, but limited reproducibility, weak causal validation, and insufficient attention on topology evolution [9].

While deep anomaly detection studies have representational flexibility, they do not necessarily address infrastructure drift. Despite these challenges, Pang et al. demonstrated that deep models can be trained to learn nonlinear normality representations from high-dimensional observations [10]. Ruff et al. also showed that many deep and shallow techniques pursue similar objectives, such as density estimation, one-class classification, reconstruction, or representation compactness [11]. Such commensurability is important as apparent architectural novelty often masks similar assumptions of stationary normality. A telemetry-only carrier-network drift detector cannot reliably determine whether rare behaviour reflects configuration drift or a legitimate network event.

The non-stationarity problem is tackled more directly in the concept-drift literature. Lu et al. identified the following adaptation strategies: drift detection, model updating, ensemble management, and data reuse [12]. A large-scale comparison found that for abrupt changes, gradual changes, recurring changes and incremental changes no detector could perform consistently better than the others, and that its performance depends on delay, false alarms, magnitude of drift, and stream characteristics [13]. Webb and colleagues proposed to consider drift as changes in the joint distributions and decision boundaries, rather than as a single undifferentiated event [14]. Sethi and Kantardzic also demonstrated that there are explicit safeguards needed to detect reliably in unlabelled streams in order to avoid the problems of novelty, sampling variation, and real concept change [17]. These findings warn against treating every shift in telemetry distribution as infrastructure drift.

The literature on carrier-network softwarization provides the relevant operational context, but there is little evidence of direct drift. The introduction of multi-access edge computing presents new and more significant challenges for inconsistent state, including the introduction of distributed orchestration, heterogeneous resource pools, mobility-sensitive placement, and low-latency constraints [15]. In the field of sliced mobile networks, DeepCog showed that learning-based prediction can help in proactively managing resources by predicting demand and adjusting resources in advance of the onset of service degradation [16]. For drift detection, its value is in demonstrating the operational value of predictive control, but workload forecasting does not detect divergence from intended configuration, nor does it predict risk based on a specific change. The main evidence clusters are organized in Table 1. The table shows that there is high methodological maturity for anomaly recognition and stream adaptation, and low integration of configuration provenance and carrier-specific causal validation.

Four conclusions are thus supported by the literature. Intended-state artifacts and the operational telemetry are complementary, not substitutable evidence. Second, drift prediction requires temporal reasoning; the drift may be harmless convergence or harmful divergence and can be initially seen as state mismatch. Third, carrier reliability needs topology/service-path context because the same configuration change can have different impacts based on redundancy, locality, traffic state and function criticality. Fourth, the current journal-based studies do not have a common reference point that connects change events, infrastructure state, telemetry, labels of incidents and service-level outcomes.

III. METHODOLOGY

The literature reviewed yielded five families of methods applicable to predictive infrastructure drift detection. Static configuration analysis examines declarations, dependency structures, policy violations and version differences. The main strength is its direct access to intended state and change provenance, and the main limitation is its lack of runtime mutation visibility and service consequences. A number of statistical streaming methods have been developed to identify departures from counters, traces, latency, resource utilization, or control-plane performance, and are often computationally efficient for online operation [6, 13]. These techniques can detect emerging instability but may confound configuration drift with workload fluctuations, maintenance operations, and software changes.

Representation-learning methods are designed to learn complex normality by either autoencoding, embedding, predicting sequences or using one-class objectives [10], [11]. These methods are appealing in the context of carrier telemetry where the observation space is multivariate and nonlinear. They suffer from low interpretability, training contamination sensitivity, lack of site stability, and limited configuration intent encoding capabilities. Log-oriented AIOps methods bring event semantics, cross-source correlation and incident context [7], [8]. Log templates and monitoring schemas, however, change over time, as does the infrastructure, leading to a second-order drift issue where the input to the detector changes independently from the condition being inferred.

Topology-aware and causal methods link symptoms together in an interdependent manner, across services, components, and propagation paths [9]. This is particularly applicable to network slices, service-function chains, edge placements and stateful virtual network functions. A topology model can help mitigate spurious attribution by separating what is shared from local configuration issues. However, causal claims can be challenging when interventions are not applied in all settings, labels are not applied immediately, and there are differences in what is observed across administrative domains. Most research on predictive resource management provides closed-loop decision logic but typically focuses on

optimizing the match between demand and capacity rather than on reliability of the configuration [16]. Table 2 is a comparison of these approaches with the requirements of predictive carrier-network drift detection. None of these methods satisfies all five requirements: intent awareness, temporal adaptation, causal localization, uncertainty estimation, and safe operational control.

Table 2: Method comparison

Ref	Method	Strengths	Limitations
[5]	Static infrastructure-as-code analysis and repository mining	Preserves intended state, version history, ownership, and explicit change sets	Cannot confirm runtime realization or quantify service impact without operational linkage
[6]	Statistical and machine-learning network anomaly detection	Supports online monitoring across high-volume numerical telemetry	Rare behaviour can be misclassified; causal relation to configuration remains uncertain
[7]	Log parsing, event sequencing, and log-based anomaly detection	Captures control-plane events, failures, and component-specific operational semantics	Template evolution, missing events, and inconsistent timestamps weaken longitudinal analysis
[8]	Multimodal AIOps correlation and failure management	Combines metrics, logs, traces, alerts, and incident records	Proprietary datasets and inconsistent evaluation targets restrict external validity
[9]	Topology-aware anomaly localization and root-cause analysis	Represents service dependencies and probable propagation paths	Dynamic topology and hidden dependencies can invalidate inferred causal structure
[10], [11]	Deep representation, reconstruction, and one-class modelling	Captures nonlinear interactions in large multivariate observation spaces	Calibration, interpretability, contamination, and domain transfer remain unresolved
[12]–[14], [17]	Concept-drift detection and adaptive learning	Detects changing distributions and supports model updating	Distribution shift is not equivalent to infrastructure-state divergence
[16]	Forecast-driven network resource control	Enables pre-emptive intervention under predicted demand	Prediction target excludes configuration integrity and change causality

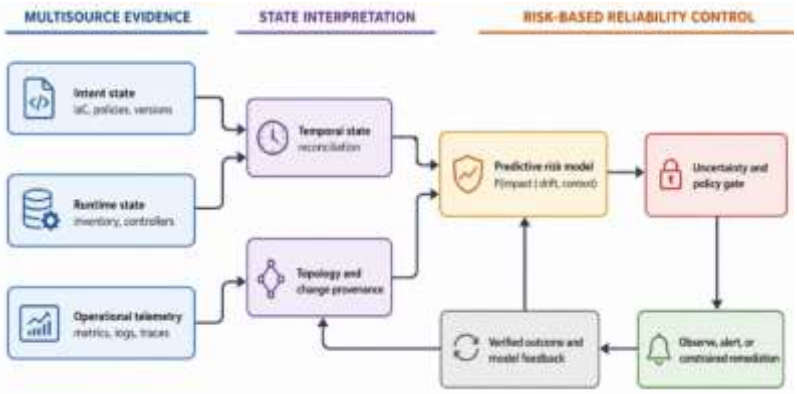

Figure 1: Predictive Infrastructure Drift Detection Framework for Cloud-Native Carrier Reliability

A conceptual architecture is presented in Figure 1 that combines the best of the families of methods that were transferable. The framework is useful because it provides the foundation for understanding the relationship between authorized intent, realized state, time context, topology, and risk of the services offered, which means that it is able to identify harmful drift without relying on any one anomaly score. The framework handles the intent of the declaration and the runtime state observed, enriches mismatches with change provenance and topology, and provides predictions of service risk and supports taking constrained action for high-confidence cases. The risk model and the state-reconciliation layer are updated based on feedback from verified incidents and remediation outcomes.

IV. DISCUSSION

The literature consistently indicates that, although prospective reliability forecasting is still in its early stages, retrospective detection is much more mature. Anomaly-detection systems can identify deviations from learned patterns, whereas infrastructure-as-code analysis can identify unusual declarations and configuration changes [5], [6]. However, neither result alone establishes that a service failure will occur. Not every change in the declaration is bad and even anomalous telemetry might be a legitimate demand, maintenance, failover or traffic engineering. Predictive drift detection, therefore, requires a conditional target: the probability that a particular divergence will produce a specified reliability consequence within a defined time horizon.

A further issue arises from the assumptions underlying reported anomaly studies. While efficient in adapting to the statistics, statistical detectors more often than not assume that current data is a good representation of normal operation [6]. For deep methods, the linearity assumption is dropped, but they must still have a training distribution that is normal enough [10, 11]. In cloud-native carrier environments, this requirement is not met due to the constant changes in software versions, placement policies, traffic mixes, scaling rules and infrastructure generations. An adaptive detector can minimize the aging of the model, but conversely; adaptation can normalize an evolving configuration fault. As shown in the concept-drift research [12]- [14] detector performance in terms of sensitivity and detection delay depends on the form of drift and the evaluation conditions. Models should therefore distinguish at least three classes of change: benign observation-schema changes, planned policy changes, and unauthorized or unsuccessful state changes that create divergence.

Another common limitation is a lack of labels. Incident tickets and alarm histories provide inadequate supervision because they record alarm detection or escalation rather than the actual onset of the fault. Labels of root causes may also signal organizational instead of technical causality. Although these records are useful, their limitations become more pronounced in unlabelled drift detection, which requires safeguards against novelty and sampling variation [17]. In carrier applications, the following, in addition to incident categories, can offer more powerful weak labels: a change in a window, a change in controller transactions, a rollout stage change, a commit of a policy, and a failure in the reconciliation process. A model learned from these temporally-aligned events may be able to predict hazard for a configuration transition, as well as a normal or abnormal observation.

Multimodal AIOps moves toward integration, but practical challenges remain Numerical metrics arrive as continuous streams, symbolic log events occur irregularly, traces provide only partial path observations, and configuration commits represent sparse state transitions [7, 8]. Simple feature concatenation can introduce temporal leakage and allow high-frequency metrics to dominate. Additional features of a more defensible architecture are explicit missingness indicators, source-specific encoders, and event-time alignment. These configuration changes should be documented as structured deltas, which should record the affected resources, policy scope, dependency relationships, authorizing mechanism, rollout phase and reversibility. The pre-change regime should then be employed to condition the telemetry and compared with a predicted post-change trajectory. Topology is important because the severity of a configuration deviation is not intrinsic; it depends on context. A mismatch occurring on an idle replica is different from a mismatch occurring on an active replica in a low-redundancy service-function chain. Although carrier topology is more complex than microservice topology (because there are physical links, virtual functions, slices, transport tunnels, edge sites and control hierarchies), the study of microservices indicates that dependency graphs can be used for the localization of symptoms [9]. Configuration snapshots should therefore be paired with contemporaneous topology snapshots. If the topology of historical events is not stored, then retrospective bias can occur, e.g., if a later graph shows dependency, which may not have been the case during an incident.

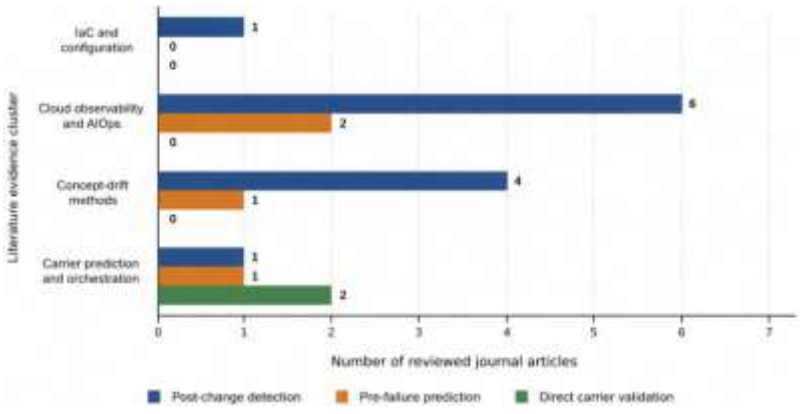

Figure 2: Distribution of Evidence Across Infrastructure-Drift Capabilities

Closed-loop operation brings an extra difference between predictive accuracy and value of reliability. The example of DeepCog demonstrates that demand forecasting can be a valuable tool for proactive allocation in sliced networks [16]. The decision policy is particularly important because remediation based on predicted drift can itself disrupt services. A high anomaly score should not directly trigger rollback or reconfiguration. The level of intervention should be based on a calibrated estimate of the likelihood of impact and the level of confidence, blast radius, reversibility, existing redundancy and delayed action costs. Staged rollback or canary reconciliation and intensified observation may be warranted for low-confidence cases, and bounded blast radius cases for high confidence cases. Automation, however, must be both safe and supported with decision-theoretic evaluation.

Figure 2 compares the analytical emphasis of the reviewed journal articles. Post-change anomaly detection receives substantially more attention than explicit pre-failure prediction or direct carrier-network validation. The grouped bars present review-coded counts of articles addressing each capability; a single paper may contribute to more than one category.

Performance claims in the literature should be interpreted cautiously because they are based on different prediction targets and datasets. Typical concept drift benchmarks are detection delay, false alarms, missed changes, and classification degradation [13]. In the field of anomaly-detection research, precision, recall, area under the receiver operating characteristic curve, and reconstruction-based separation are often reported [10], [11]. Other outcomes of AIOps and root-cause studies could also be localization accuracy, ranking quality, incident coverage, or time reduction [8], [9]. Resource efficiency, demand forecast error or service-level violations are all measures that can be used to assess carrier prediction [16]. These measures are not directly interchangeable. Low detection delay in a synthetic stream does not mean that an incident would provide useful early warning for a configuration-induced carrier incident. Table 3 therefore presents outcomes according to the operational question being addressed rather than imposing an artificial common performance scale. The comparison reveals that each of the literature clusters addresses only a small portion of the predictive drift issue.

Table 3: Results Comparison

Ref	System / Context	Metric / Basis of Comparison	Outcome
[5]	Infrastructure-as-code research corpus	Research-topic coverage, empirical method, validation setting	Strong growth in automated configuration research, but limited runtime-state integration and uneven industrial evidence
[6]	Network traffic and operational monitoring	Detection coverage, adaptability, false-positive exposure	No universally superior anomaly method; effectiveness depends on feature representation and operating regime
[7]	Software-system logs	Parsing stability, sequence modelling, anomaly-identification basis	Log analysis supports reliability diagnosis, but evolving templates and incomplete semantics impair longitudinal consistency
[8]	AIOps failure-management studies	Prediction, detection, correlation, diagnosis, remediation coverage	Broad functional coverage is reported, although benchmark fragmentation restricts cross-study comparison
[9]	Microservice-based cloud applications	Anomaly detection and root-cause localization	Dependency-aware methods improve localization potential, but causal validity and public reproducibility remain limited
[10]	High-dimensional anomaly datasets	Representation quality, label requirement, contamination sensitivity	Deep models capture nonlinear structure but do not remove dependence on stable training assumptions
[11]	Deep and shallow anomaly-method families	Objective-function and assumption comparison	Method families share core one-class and density assumptions; additional depth does not guarantee operational superiority
[13]	Synthetic and real concept-drift streams	Delay, false alarms, missed drift, classifier impact	Detector ranking changes across drift forms and evaluation conditions
[14]	Formal concept-drift characterization	Distribution component, severity, duration, recurrence	Drift interpretation improves when affected distributions and decision effects are explicitly specified
[16]	Sliced mobile-network resource management	Forecast-informed capacity control and service demand	Prediction supports proactive allocation, but configuration divergence is not modelled as the causal target

[17]	Unlabelled streaming data	Reliability of unsupervised change indication	Safeguards are required to separate genuine drift from novelty and sampling variation

There are a number of technical implications that follow. First, it is necessary to have an explicit prediction horizon. Immediate post-change risk may be related to deployment health and reconciliation status, and longer-horizon risk may be related to traffic growth or state accumulation, certificate expiry, resource exhaustion, or interaction with later changes. These mechanisms are concealed under the umbrella of one binary label. Static classification might not be the most appropriate method for survival analysis, temporal point process, or multi-horizon hazard models, which can model censoring and time-to-impact.

Second, negative examples need to be carefully constructed. It is possible that a configuration change may have no incident during a low-traffic period but is unsafe under high traffic. Labelling a change as safe solely because no incident occurred can introduce confounding. Matching changes by topology, workload regime, software version, and redundancy state can help minimize the impact of bias in a counterfactual evaluation. Mechanistically informed examples can also be created through controlled fault injection and/or digital-twin experiments as long as the effects of the fault are clearly distinguished from the operational evidence.

Third, calibration and decision cost should be part of the model evaluation. In situations with low prevalence of incidents and an asymmetric cost of remediation, precision and recall are not enough. Reliability engineering needs aspects like reduced service impact, prevented violation minutes, false intervention rate, calibration error, time gained before impact and performance across sites or software releases. Report confidence intervals over time partitions, rather than random record partitions, as it can introduce recurring configuration patterns into both the training set and the test set.

Fourth, interpretability should be related to actionable infrastructure objects. Measurements of generic features are not useful when an operator needs to decide to stop a rollout, to roll back a policy, to isolate a site or to inspect a controller. Explanations should include the configuration delta, service path affected, sequence of symptoms, similar historical transitions, and uncertainty source. Such output can verify without claiming causality that is not supported.

In Figure 3, method families are compared against the required capabilities for carrier-class predictive drift detection. The reason why the visualization is important is that it explains the benefit of the hybrid approach: a highly interpretable policy differencing is not capable of early-warning capacity, while deep representation learning has a stronger early-warning capacity and a stronger prediction capability with weaker topology awareness and operational transparency. Each of the following method families is scored ordinally from 1 to 5 for the five domains of early warning, interpretability, label independence, topology awareness, and deployment readiness, as shown in the heatmap. Scores are intended to be used for comparison purposes, not for inferential statistics, and are a summary of method properties indicated by the articles reviewed.

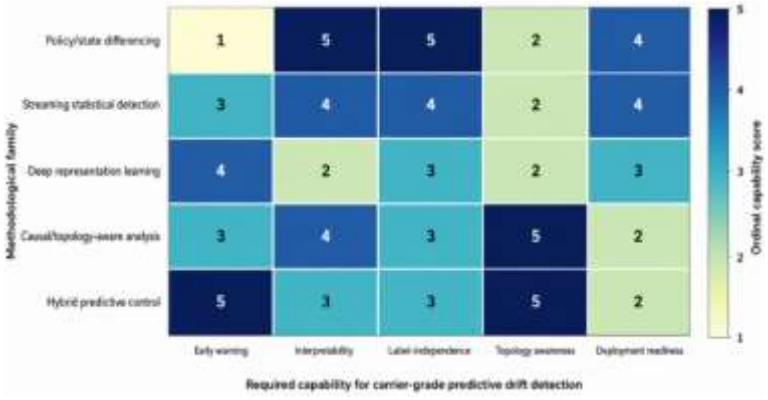

Figure 3: Capability Profile of Methodological Families for Predictive Carrier-Network Drift Detection

Finally, the resulting framework should be deterministic state reconciliation coupled with a probabilistic prediction, not a replacement of configuration controls with one machine learning model. By defining the difference between desired and realized states and compliance with an explicit policy, deterministic logic can determine whether a change is acceptable or not. The divergence can be captured by statistical and deep models to estimate if the divergence will continue or yield abnormal service behaviour. Topology-aware reasoning can provide propagation and blast-radius estimation. Then a policy gate can choose between observation, escalation, staged rollback, or automated remediation, based on risk calibration and operational constraints.

The main gap in the research is empirical. Typically, public anomaly datasets do not contain the coordinated artifacts needed for this or public infrastructure-as-code repositories do not contain public telemetry or verified service results. Existing carrier studies also favour the protection of operational traces as configurations and topologies provide security-sensitive information. Comparative evaluation will remain based on varying proxies where there are no shared longitudinal datasets. Therefore, privacy-preserving benchmark construction, standardised event schemas, and multi-operator evaluation are core research infrastructure needs, not secondary data-management concerns.

V. FUTURE DIRECTIONS

Future research should define infrastructure drift as a time-indexed relationship among intended state, applied state, observed state, and expected service behaviour. Each of these typed drift categories should distinguish among the following: Unauthorized change, failed convergence, version skew, partial rollout, controller inconsistency, environmental override, semantically incompatible but textually valid configuration. In such a representation it would not be possible for models to collapse multiple failure mechanism anomaly labels into a single failure anomaly label.

The development of benchmarks should capture within a shared clock, configuration commits, deployment events, topology snapshots, metrics, logs, traces, service-level indicators, intervention records and outcome verification. Warning time, calibration, false-intervention cost, prevented service-impact minutes, and traditional classification metrics [8], [12], [13] should all be considered during evaluation.

Future methodological development should prioritize hybrid architectures. Service-function chaining, slice dependency, sharing of physical resources and edge placement can be encoded in a graph-based approach. The time of a divergence becoming dangerous can be estimated using temporal hazard models and conformal or Bayesian models can provide insights into predictive uncertainty. In addition to intervention evidence, the motivation of causal discovery is to support intervention, so causal discovery should be used in conjunction with it, unless there are natural experiments set up by operational changes. Digital twins and controlled fault injection may be used to support testing of mechanisms, but results of the simulations must be validated by comparing to operational traces before the automation authority is expanded.

Research on carrier-grade remediation policies also remains limited. Redundancy state, rollback safety, regulatory obligations, maintenance windows and cross-domain ownership are all aspects of intervention policies. Multiple operators could compare the behaviour of the models without providing topology or configuration information. The effects of explanations on diagnostic accuracy, and the effects of automation on correlated operational errors, should be studied from a human-factors perspective. The goal of the research is not to increase the sensitivity to the anomalies but to prove the reduction of the service risk without inducing any unsafe control action.

VI. CONCLUSION

Over the next few years, predictive infrastructure drift detection will fall somewhere between declarative configuration management, cloud observability, adaptive anomaly detection, topology-aware diagnosis, and carrier-network reliability engineering. We already have well-established methods from the literature for configuration analysis, operational anomaly detection, log interpretation, distribution-shift detection, and forecast-driven network control. Unfortunately, these capabilities are not well-integrated, as intended state, runtime state, change provenance, topology and service consequences are usually assessed by different models and datasets.

Any valid framework should first establish the nature and duration of state divergence and then estimate the state-dependent service risk under the relevant topology and load conditions. Thus, deterministic reconciliation, probabilistic forecasting, graph-based context, calibrated uncertainty and constrained intervention are complementary. Carrier-grade validation, however, should not only be based on anomaly-classification accuracy but also include the reduction of the impact on the service, the cost of false intervention, transferability across sites and releases, and the warning time.

A key limitation of the field is the absence of common longitudinal evidence of configuration transitions and documented outcomes. Progress will depend on the construction of privacy-friendly benchmarks, understanding of the semantic nature of common events, preservation of historical topology and realistic studies of temporal change. If the bounds of predictive confidence and remediation authority are kept as explicit bounds, then a successful predictive framework can be used to move configuration management from a compliance inspection exercise to an anticipatory reliability function.

VII. REFERENCES

[1] Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., & Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials, 18*(1), 236–262.

[2] Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE, 103*(1), 14–76.

[3] Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., & Flinck, H. (2018). Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials, 20*(3), 2429–2453.

[4] Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *Journal of Internet Services and Applications, 9*(1), Article 16.

[5] Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. *Information and Software Technology, 108*, 65–77.

[6] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications, 60*, 19–31.

[7] He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. *ACM Computing Surveys, 54*(6), Article 130.

[8] Notaro, P., Cardoso, J., & Gerndt, M. (2021). A survey of AIOps methods for failure management. *ACM Transactions on Intelligent Systems and Technology, 12*(6), Article 81.

[9] Soldani, J., & Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. *ACM Computing Surveys, 55*(3), Article 59.

[10] Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys, 54*(2), Article 38.

[11] Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., & Müller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE, 109*(5), 756–795.

[12] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering, 31*(12), 2346–2363.

[13] Barros, R. S. M., & Santos, S. G. T. C. (2018). A large-scale comparison of concept drift detectors. *Information Sciences, 451–452*, 348–370.

[14] Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery, 30*(4), 964–994.

[15] Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., & Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials, 19*(3), 1657–1681.

[16] Bega, D., Gramaglia, M., Fiore, M., Banchs, A., & Costa-Pérez, X. (2020). DeepCog: Cognitive network management in sliced 5G networks with deep learning. *IEEE Transactions on Mobile Computing, 19*(2), 267–281.

[17] Balaji, R. (2024) 'Threat-Model-Driven Zero-Trust Authorization for Microservice APIs in Kubernetes-Based Cloud-Native Applications', ESP Journal of Engineering & Technology Advancements, 4(4), pp. 223–234. doi: 10.56472/25832646/JETA-V4I4P128.

[18] Sethi, T. S., & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabelled data. *Expert Systems with Applications, 82*, 77–99.