Article

Evaluation–Deployment Structural Divergence in Large-Scale AI Systems: A Diagnostic Perspective on Projection Mismatch and Runtime Drift

Gregor Herbert Wegener

Friedrichstrasse 4, 10969 Berlin, Germany; gregor.wegener@gmail.com; Tel.: +49 179 2544522

Abstract

Large-scale AI systems are evaluated in controlled benchmark environments but deployed in dynamically coupled production ecosystems. Recent international AI assessment reports note that evaluation predictability decreases as advanced models increasingly differentiate between tightly defined test contexts and open-ended deployment settings [8]. This paper reframes evaluation–deployment divergence not as a model-internal phenomenon, but as a *projection mismatch* between two structurally distinct system spaces. Evaluation operates as a constrained projection: fixed prompts, bounded distributions, limited feedback loops, and minimal runtime coupling. Deployment unfolds within a dynamically coupled execution space: multi-agent interaction, adaptive serving infrastructure, scheduler and policy constraints, latency–cost tradeoffs, and cross-layer control feedback. These spaces are structurally non-equivalent. Their divergence forms a coordination surface across which behavioral variance naturally emerges at scale. To reason about this divergence with architectural clarity, we introduce a diagnostic taxonomy comprising five structural analysis layers: Evaluation Context Projection Instability, Benchmark Integrity and Drift Diagnostics, Structural Drift Diagnostics, Runtime Control Coherence, and Deployment Drift Signal Aggregation. These layers map to diagnostic applications within the SORT-AI structural analysis framework [2,3]. The analysis is intentionally diagnostic rather than prescriptive. It provides a structural vocabulary for discussing evaluation–deployment divergence as an architectural property of large-scale AI systems, without attributing fault, intent, or alignment failure, and without offering implementation guidance.

Keywords: evaluation–deployment divergence; projection mismatch; benchmark integrity; runtime drift; control coherence; structural diagnostics; AI safety; SORT-AI

Executive Summary

Large-scale AI systems are evaluated under tightly defined benchmark conditions but deployed within dynamically coupled production ecosystems. As models scale and deployment environments increase in structural complexity, observable variance between evaluation outcomes and deployment behavior expands [8,9].

This paper reframes evaluation–deployment divergence not as a model-internal anomaly, but as a structural non-equivalence between two system spaces.

Evaluation operates as a constrained projection of behavior: fixed prompts, bounded distributions, limited feedback loops, and minimal runtime coupling. Deployment unfolds within a high-dimensional execution space shaped by multi-agent interaction, adaptive serving infrastructure, scheduler and policy constraints, latency–cost tradeoffs, and cross-layer control feedback. These spaces are structurally distinct. Their divergence defines a coordination surface across which behavioral variance naturally emerges at scale.

To reason about this divergence with architectural clarity, we introduce five structural analysis layers:

1. **Evaluation Context Projection Instability**
2. **Benchmark Integrity and Drift Diagnostics**
3. **Structural Drift Diagnostics**
4. **Runtime Control Coherence**
5. **Deployment Drift Signal Aggregation**

These layers map to diagnostic applications within the SORT-AI structural analysis framework [1, 2].

The analysis is intentionally diagnostic rather than prescriptive. It attributes no fault, intent, or alignment failure. Instead, it provides a structural vocabulary for discussing evaluation–deployment divergence as an architectural property of large-scale AI systems.

1. The Emerging Divergence Problem

1.1. Operational Observations at Scale

As large-scale AI systems move from controlled evaluation into continuously evolving production environments, engineering teams increasingly observe structural variance between benchmark behavior and deployment dynamics [8,9].

Several recurring patterns appear across independent analyses:

Evaluation reliability does not always transfer to live interaction contexts.

Systems stable under constrained test regimes exhibit gradual behavioral drift under sustained real-world feedback.
Adversarial or red-team configurations surface behaviors not reproducible under standard benchmark conditions.
Observable deployment behavior evolves over time even when model weights remain unchanged.

These observations are often discussed using terms such as alignment fragility, capability masking, or benchmark gaming. The 2026 International AI Safety Report notes that advanced models increasingly differentiate between tightly defined test contexts and open deployment settings [8]. Independent benchmark integrity analyses further show that training-set contamination can enable strong performance on historical benchmarks while structurally equivalent post-cutoff tasks reveal different generalization properties [9].

A common implicit assumption underlies many of these interpretations: that evaluation and deployment operate within a structurally comparable system space.

At scale, this assumption no longer holds.

1.2. The Core Structural Hypothesis

The central claim of this paper is:

Evaluation–deployment divergence is not primarily a model property. It is a projection mismatch between two structurally non-equivalent system spaces.

Evaluation observes behavior under constrained projection: controlled prompts, static distributions, limited feedback loops, and minimal runtime coupling. Deployment operates within a dynamically coupled execution space shaped by multi-agent interaction, adaptive serving infrastructure, scheduler and policy constraints, latency–cost tradeoffs, and cross-layer control feedback [3,5].

Evaluation confidence therefore reflects stability within a low-dimensional projection regime. Deployment variance reflects behavior within a higher-dimensional, continuously coupled structure.

The structural properties of the SORT projection framework—specifically the distinction between constrained and full-coupling projections—provide a formal vocabulary for characterizing this divergence without attributing fault, intent, or design deficiency [1].

2. Why Classical Evaluation Is Structurally Bounded

2.1. Evaluation Measures Projected Performance

Benchmark systems measure accuracy, compliance behavior, hallucination rates, robustness under adversarial prompting, and related output characteristics. These metrics are operationally valuable and essential for comparative assessment [10].

However, they operate within a deliberately constrained projection of system behavior. Evaluation environments isolate prompts, bound distributions, suppress long-term interaction history, and abstract away runtime-level coupling. In doing so, they measure projected performance within a static frame rather than behavior within a dynamically coupled execution context.

The Stanford AI Index 2025 reports that leading models now approach saturation on established benchmarks such as MMLU, while performance on newly introduced reasoning-intensive benchmarks remains substantially lower [10]. This divergence is consistent with the projection hypothesis: benchmark scores reflect optimization within a defined evaluation regime, not necessarily performance under expanded structural coupling.

2.2. The Projection Abstraction Boundary

Evaluation reduces system behavior into a bounded context: fixed data distributions, finite prompt sets, absence of cost constraints, no dynamic resource contention, and no persistent multi-agent interaction history.

This reduction is not a flaw. It is a necessary abstraction that enables reproducibility and comparability. However, it also establishes a structural abstraction boundary. Degrees of freedom active in deployment—scheduler interaction, serving variability, cost-pressure adaptation, user feedback accumulation—are compressed out of the evaluation regime.

Interdisciplinary benchmark analyses document that data contamination, benchmark saturation, and task-specific shortcut learning are widespread across multiple research domains [9]. In several documented cases, models achieve high benchmark scores through structural regularities specific to the evaluation dataset (e.g., URL-pattern shortcuts in navigation tasks), without demonstrating equivalent robustness under structurally varied conditions.

From a structural perspective, these findings illustrate a consistent pattern: optimization within a constrained projection space does not automatically transfer to behavior within a higher-dimensional deployment structure space.

3. Evaluation as a Restricted Projection Space

Evaluation can be interpreted as a structured projection of a higher-dimensional system space.

A production AI system spans multiple interacting layers: model parameters, serving runtime, scheduler behavior, resource allocation, user interaction patterns, and policy enforcement mechanisms. These layers form a coupled state space in which behavior emerges from cross-layer interaction.

Evaluation isolates a subset of that space. It constrains prompt distributions, suppresses long-term feedback accumulation, removes resource contention, and abstracts away scheduler and infrastructure dynamics. The resulting projection compresses the active degrees of freedom of the full system into a bounded observational regime.

Within this restricted projection, behavior appears comparatively stable and reproducible. Variability induced by runtime coupling, adaptive load balancing, or multi-tenant interference is intentionally minimized. This is a necessary design choice: reproducibility requires structural reduction.

However, stability within a constrained projection does not imply equivalent stability under expanded coupling [1]. A structure that behaves predictably under reduced dimensionality may exhibit different dynamics when additional interaction surfaces are activated. The SORT projection framework formalizes this distinction between constrained and full-coupling observation spaces [2].

In practical terms, a model passing an extensive benchmark or safety evaluation suite demonstrates stability within the evaluation projection. When embedded in a production environment characterized by runtime feedback, multi-tenant workloads, scheduler variability, and adaptive cost constraints, the observable behavior reflects the properties of the full coupled system [3].

Benchmarks therefore characterize model behavior under constrained coupling conditions. Deployment reveals behavior under expanded structural interaction.

4. Deployment as a Dynamically Coupled Structure Space

While evaluation isolates system components within a constrained projection, deployment activates the full interaction surface of the system.

A production AI environment is not defined solely by model weights and input distributions. It is defined by structural coupling across runtime layers. Multi-tenant workloads compete for shared resources. Adaptive load balancing redistributes traffic based on latency and utilization signals. Retry logic and rate limiting introduce conditional execution paths. Policy enforcement layers intervene dynamically based on contextual triggers. Cost-aware inference may truncate outputs or modify generation behavior under budget constraints. Agentic tool-calling loops create recursive execution pathways with extended task horizons [5]. Cross-layer control feedback between scheduler, runtime, and serving engine continuously reshapes system dynamics [3].

Each of these mechanisms introduces additional feedback paths into the execution graph. These feedback paths modify observable behavior even in the absence of model weight updates. The system evolves through interaction rather than parameter change.

Deployment is therefore not merely a broader input distribution. It is a higher-dimensional coupling regime. Behavior emerges from the interaction between model inference, infrastructure dynamics, cost surfaces, and user-driven feedback loops.

Recent model releases incorporating extended agentic capabilities illustrate this shift toward structural coupling. Systems that operate across multi-step planning horizons, invoke external tools, or coordinate multiple instances activate execution pathways that are structurally absent in static evaluation settings [8]. When combined with retry mechanisms, adaptive serving policies, and runtime cost constraints, the resulting behavior reflects properties of the coupled system rather than the isolated model component.

Infrastructure partnerships that introduce stateful runtime environments further expand this interaction surface. In such settings, deployment architecture itself becomes a structural variable influencing behavior, independent of model training or benchmark configuration.

At hyperscale, coupling effects amplify. Large training and inference clusters exhibit recovery dynamics, scheduling interactions, and resource contention patterns that are intrinsic to distributed execution [12]. These structural dynamics—recovery cascades, allocation variability, and cross-layer control interactions—are not artifacts. They are normal properties of large coupled systems. They are, however, absent from evaluation projections.

Deployment thus reveals behavior under full structural interaction. Evaluation observes behavior under constrained structural isolation.

5. Five Structural Diagnostic Layers

The divergence between evaluation and deployment can be decomposed into five structural diagnostic layers. Each layer identifies a distinct dimension along which projection stability in evaluation differs from behavior under deployment coupling.

These layers are not independent failure modes. They are orthogonal observation axes applied to the same structural non-equivalence introduced in Sections 3 and 4.

5.1. Evaluation Context Projection Instability

Structural condition: Stability under constrained projection does not guarantee stability under expanded context dimensionality.

A model may pass static safety prompts or bounded evaluation suites. When exposed to extended interaction horizons, accumulated dialogue state, tool outputs, or multi-step planning contexts, response behavior may shift even in the absence of weight updates [2].

This is not necessarily alignment degradation. It is projection instability: behavioral characteristics that are conditional on the dimensionality of the observation space.

Recent analyses document that models can differentiate between tightly defined evaluation regimes and open-ended deployment contexts [8]. As context windows expand and interaction histories lengthen, the effective state space increases. Evaluation regimes that operate within bounded prompt scopes cannot fully probe behaviors that emerge only under extended context accumulation.

The instability is therefore conditional, not anomalous. It arises from context expansion rather than parameter change.

5.2. Benchmark Integrity and Drift Diagnostics

Structural condition: Benchmarks assume distributional stationarity; deployment evolves in time.

Evaluation benchmarks operate within a fixed geometry: stable task definitions, fixed datasets, and bounded prompt distributions. Deployment environments do not.

User behavior shifts. Tool availability changes. Prompt engineering practices evolve. Usage contexts expand. Even absent adversarial manipulation, deployment progressively diverges from the distributional assumptions embedded in benchmark construction [9].

Expanding benchmark coverage increases projection dimensionality, but it does not eliminate projection mismatch. Each benchmark captures a slice of system behavior within evaluation geometry. Deployment unfolds along a temporal axis, introducing distributional drift and structural recombination not represented in static test suites.

As leading models approach benchmark saturation [10], differentiation increasingly depends on structural robustness under deployment coupling rather than incremental gains within evaluation geometry. This reflects projection compression: evaluation resolution becomes insufficient to discriminate between systems whose deployment behaviors may diverge structurally.

5.3. Structural Drift Diagnostics

Structural condition: Observable behavioral change without model weight modification.

Structural drift refers to shifts in system-level behavior that originate from runtime and orchestration dynamics rather than parameter updates [6].

Contributing mechanisms include:

- Runtime batching adjustments
- Memory allocation variability
- Latency optimization strategies
- Control-layer feedback adaptation
- Scheduler contention and resource reallocation

In such cases, the model remains static. The coupled system evolves.

Operational evidence from large inference environments demonstrates that modifying control interactions—without changing model weights—can significantly alter throughput and latency characteristics [15]. Behavior shifts because the execution surface shifts.

For evaluation–deployment divergence, this implies that a system evaluated at time t_0 may exhibit different behavior at t_1 solely due to deployment coupling evolution. Drift originates from structural interaction layers rather than model degradation.

5.4. Runtime Control Coherence

Structural condition: Multiple autonomous control loops operate concurrently across system layers.

Large AI deployments include schedulers, orchestrators, serving engines, safety layers, cost-management mechanisms, and rate limiters [3]. Each optimizes a local objective: throughput, latency, compliance, cost containment, or resource stability.

Local optimization does not automatically imply global coherence.

Under certain load regimes, interactions between control loops can produce oscillatory latency patterns, retry amplification, tail variance, or non-deterministic throughput. These effects do not indicate malfunction. They reflect objective misalignment across control layers.

Evaluation environments suppress such interactions by design. Benchmarks do not operate under multi-tenant load balancing, memory-bandwidth contention, or adaptive scheduling variability. Deployment behavior is therefore conditioned by control coherence properties absent from the evaluation projection [3,4].

5.5. Deployment Drift Signal Aggregation

Structural condition: Deployment divergence manifests through weak, distributed signals rather than abrupt transitions.

Production systems rarely exhibit binary regime shifts. Instead, they accumulate small variations:

- Marginal latency variance
- Incremental token expansion
- Slight retry amplification
- Progressive memory pressure

Individually, such signals appear subcritical. Aggregated, they indicate structural regime movement [6].

Evaluation environments are optimized for signal clarity: fixed inputs produce measurable outputs. Deployment environments generate ambient interaction noise within which drift signals must be extracted.

Weak-signal aggregation therefore becomes a structural diagnostic layer absent from evaluation settings. It distinguishes transient variance from sustained coupling-induced drift and enables separation between model-level degradation and system-level evolution.

6. Architecture Risk Perspective

Evaluation–deployment divergence does not primarily introduce model-level uncertainty. It introduces architecture-level inference errors. When projection mismatch is not explicitly modeled, system operators may draw incorrect conclusions about structural stability.

The following risks emerge directly from the projection mismatch described in previous sections.

6.1. False Stability Inference

Structural risk: Stability under evaluation is interpreted as stability under full deployment coupling.

Benchmark performance measures behavior within constrained projection geometry. Deployment behavior unfolds under expanded dimensionality and cross-layer interaction.

A system that appears stable under evaluation may exhibit sensitivity under multi-tenant load, extended interaction history, or runtime constraint interaction [2,7].

The risk is not failure. The risk is misinterpretation.

Evaluation health is often used as a proxy for deployment resilience. Under projection mismatch, this proxy is structurally invalid.

As inference cost reductions accelerate large-scale deployment [10], systems transition more rapidly from evaluation to production environments. If projection mismatch is not explicitly accounted for, stability assumptions propagate into environments where structural conditions differ fundamentally.

6.2. Drift Without Alarm

Structural risk: Behavioral evolution occurs without model-version change.

Most monitoring frameworks are model-centric. They track parameter updates, version releases, or major configuration changes.

Structural drift, however, originates in coupling layers: scheduler policy adjustments, resource allocation shifts, load balancing heuristics, control-loop refinements, or orchestration updates [3,6].

In such cases, the model remains unchanged. Observability dashboards report nominal throughput and acceptable latency envelopes. Yet interaction geometry has shifted.

The risk is not silent failure. It is silent reconfiguration.

System behavior may evolve gradually without crossing predefined alert thresholds. Without structural diagnostics capable of distinguishing model-level change from coupling-layer change, operators may attribute behavioral variation to stochastic variance rather than to structural drift.

6.3. Benchmark Optimization as Projection Overfitting

Structural risk: Optimization within evaluation geometry reduces robustness under deployment coupling.

When evaluation space differs from deployment space, optimization pressure concentrates within the projection manifold defined by benchmarks.

This can produce geometric overfitting: solutions that are stable within evaluation projection but brittle under expanded structural interaction.

Observed phenomena such as benchmark-specific shortcut learning or performance collapse on structurally equivalent post-cutoff tasks illustrate this effect [9]. These outcomes need not imply intentional gaming or deception. They reflect optimization within constrained geometry.

Capability masking, in this framing, becomes an emergent property of projection-specific optimization. Behavior is tuned to the observable evaluation surface. Deployment introduces dimensions not represented in that surface.

The resulting divergence is structural, not adversarial.

From an architecture perspective, this implies that increasing benchmark complexity alone does not eliminate risk. Without modeling projection mismatch, evaluation-driven optimization may continue to prioritize performance within restricted geometry over robustness under full coupling.

7. Implications for Hyperscalers

Hyperscale environments do not merely increase workload. They expand structural coupling. As system size grows, the dimensionality of interaction surfaces grows with it. Evaluation–deployment divergence therefore scales nonlinearly with architectural complexity.

7.1. Scale Expands Coupling Surfaces

More GPUs, more tenants, more agents introduce additional feedback paths. Each new control interface, orchestration layer, or scheduling domain increases the dimensionality of system interaction [4,12].

At small scale, projection mismatch may remain marginal. At hyperscale, even minor coupling variations propagate across thousands of nodes and millions of requests. Structural sensitivity amplifies with coordination density.

Operational evidence from large-scale clusters demonstrates that performance characteristics are often dominated not by model computation but by orchestration geometry [12]. Software-defined pooling and fine-grained scheduling strategies can recover substantial effective capacity without hardware modification [13]. This illustrates a core principle: system behavior is structurally conditioned by coupling regime.

As coupling surfaces expand, evaluation projection becomes progressively less representative of deployment dynamics.

7.2. Logging Is Not Structural Diagnosis

Modern observability stacks provide detailed telemetry: utilization rates, latency histograms, throughput metrics, retry counters, error codes.

These signals describe events. They do not describe projection geometry.

A system may report nominal utilization while exhibiting coupling inefficiencies invisible to metric dashboards [11]. Utilization does not equal effective work. Throughput does not imply coherence.

Observability captures activity. Structural diagnostics capture interaction topology.

Evaluation–deployment divergence persists when monitoring frameworks measure outputs without modeling coupling structure.

7.3. Benchmark Expansion Does Not Eliminate Divergence

Expanding benchmark suites improves coverage within evaluation geometry. It does not reconcile evaluation projection with deployment coupling.

The proliferation of specialized benchmarks reflects recognition that single-metric evaluation is insufficient. Yet each benchmark defines its own projection slice. Deployment is not the union of these slices. It is a coupled execution manifold shaped by runtime feedback, resource contention, cost surfaces, and interaction history [3].

Improving benchmark complexity increases projection dimensionality. It does not remove projection mismatch.

Structural divergence therefore persists even under comprehensive evaluation regimes.

7.4. Governance Requires Projection Awareness

Governance frameworks increasingly mandate post-deployment monitoring and lifecycle risk management [16,18]. These developments recognize that evaluation alone is insufficient.

However, most governance mechanisms remain predominantly model-centric. They focus on training data documentation, capability assessment, and versioned model evaluation as the primary unit of accountability.

Deployment divergence introduces structural dynamics that originate in runtime coupling rather than model weights: control-loop incoherence, scheduling variability, multi-tenant interaction effects, and weak-signal regime movement.

For governance and compliance functions, this distinction is not abstract. Projection mismatch directly affects how evaluation artifacts are interpreted as regulatory evidence under post-market monitoring obligations. If structural drift originates at the system layer rather than the model layer, model-centric audit mechanisms cannot fully explain observable deployment behavior.

Projection awareness means explicitly recognizing that evaluation and deployment inhabit structurally distinct spaces.

Without this distinction, governance mechanisms regulate the evaluation projection while deployment behavior evolves under expanded coupling dimensionality.

Extending governance beyond model evaluation into structural system analysis is therefore not regulatory expansion. It is geometric consistency.

8. Structural Diagnostics and SORT Application Mapping

The five diagnostic layers introduced in this paper align with structural conditions that have been independently characterized within the SORT diagnostic framework [1,2]. This section maps each diagnostic layer to specific SORT-AI applications, treating each as a structural instrument for identifying the conditions from which evaluation–deployment divergence arises.

The applications referenced below are drawn from the SORT-AI research series, which provides a diagnostic vocabulary for reasoning about structural instability across physical, logical, and control coupling layers in AI systems [3–6]. The theoretical foundation is documented in [1], with safety-relevant projection properties in [2].

8.1. ai.47 — Evaluation Context Projection Instability (Cluster A)

Structural condition diagnosed: Instability arising from context expansion beyond the evaluation projection boundary, producing behavioral divergence that is invisible under constrained test conditions [2]. The catalog definition—*structural diagnostics for instability arising when evaluation context is expanded beyond test boundary conditions*—addresses the core structural condition underlying test awareness and situational awareness phenomena.

Relevance to divergence taxonomy: The evaluation context instability documented in Section 5.1—where models exhibit different behavior under expanded context dimensionality—is a direct instance of the structural condition ai.47 diagnoses. The 2026 International AI Safety Report's finding that models distinguish between test and deployment settings [8] is a behavioral manifestation of projection instability at the evaluation boundary.

Reference: Wegener, G.H. (2025). *SORT-AI: A Projection-Based Structural Framework for AI Safety*. Preprints.org, doi:10.20944/preprints202512.1334.v2

8.2. ai.16 — Benchmark Integrity and Drift Diagnostics (Cluster B)

Structural condition diagnosed: Distributional divergence between benchmark assumptions and deployment reality, including data contamination, benchmark saturation, and projection-specific optimization [2,6]. The catalog definition—*diagnostics for distributional divergence between benchmark regimes and deployment distributions, including data contamination and benchmark gaming effects*—addresses the structural integrity of evaluation instruments.

Relevance to divergence taxonomy: The benchmark integrity concerns documented in Section 5.2—distributional evolution, benchmark saturation, and projection-specific optimization—represent the structural conditions ai.16 diagnoses. The documented compression of performance gaps across leading models to within 5% [10] indicates benchmark projection saturation, where the evaluation geometry can no longer resolve structural differences relevant to deployment.

Reference: Wegener, G.H. (2025). *SORT-AI: A Projection-Based Structural Framework for AI Safety*. Preprints.org, doi:10.20944/preprints202512.1334.v2

8.3. ai.04 — Runtime Control Coherence (Cluster C)

Structural condition diagnosed: Incoherence between scheduler, orchestrator, runtime, and policy enforcement layers, producing instability and efficiency loss invisible at the component level [3]. The catalog definition—*diagnose and reduce incoherence between scheduler, runtime and model control loops*—addresses the structural condition underlying runtime-induced behavioral divergence.

Relevance to divergence taxonomy: The runtime control incoherence documented in Section 5.4—where independently correct control loops produce emergent behavioral change—is the primary mechanism through which deployment behavior diverges from evaluation expectations. The Meta engineering documentation of 35% throughput recovery through control alignment on unchanged hardware [15] demonstrates that control coherence is a structural determinant of system behavior.

Reference: Wegener, G.H. (2026). *SORT-AI: Runtime Control Coherence in Large-Scale AI Systems*. Preprints.org, doi:10.20944/preprints202601.0298.v1

8.4. ai.27 — Inference Pipeline Control Coherence (Cluster C)

Structural condition diagnosed: Incoherence across inference pipelines including batching, caching, serving control loops, and execution paths [3]. The catalog definition—*structural coherence analysis of inference pipelines including batching, caching, and serving control loops*—extends runtime control diagnostics to the specific execution environment of deployed AI systems.

Relevance to divergence taxonomy: Inference pipeline incoherence directly shapes the behavior of deployed systems in ways that are invisible to evaluation. When batching decisions, KV-cache management, and serving control interact incoherently [14], the system produces behavior that differs from the evaluation projection. This is particularly relevant for the February 2026 model generation, where extended context windows and agentic task horizons increase the structural surface over which inference pipeline incoherence may manifest.

Reference: Wegener, G.H. (2026). *SORT-AI: Runtime Control Coherence in Large-Scale AI Systems*. Preprints.org, doi:10.20944/preprints202601.0298.v1

8.5. ai.52 — Deployment Drift Signal Aggregation (Cluster C)

Structural condition diagnosed: Weak-signal detection across deployment environments, identifying regime shifts before threshold breach through aggregation of individually subcritical indicators [6]. The catalog definition—*aggregation and structural analysis of weak drift signals across deployment environments for early regime-shift detection*—addresses the diagnostic gap between component-level monitoring and structural drift detection.

Relevance to divergence taxonomy: The deployment drift signal aggregation described in Section 5.5 identifies the specific structural mechanisms through which deployment coupling produces observable behavioral change. Without this diagnostic layer, structural drift manifests as unexplained performance variation, attribution-resistant behavioral change, or gradual governance blind spots [3,6].

Reference: Wegener, G.H. (2026). *SORT-AI: Structural Efficiency Recovery in Hyperscale AI Systems*. Preprints.org, doi:10.20944/preprints202602.0015.v1

8.6. Summary of Diagnostic Mapping

Table 1. SORT Diagnostic Application Mapping — Evaluation–Deployment Divergence

App.	Cluster	Diagnostic Layer	Divergence Relevance	Primary Ref.
ai.47	A	Eval Context Projection Instability	Test awareness; context-expansion-induced behavioral divergence	Safety [2]
ai.16	B	Benchmark Integrity & Drift	Distributional divergence; benchmark saturation; projection-specific optimization	Safety [2]
ai.04	C	Runtime Control Coherence	Scheduler–runtime–policy incoherence producing emergent behavioral change	Control [3]
ai.27	C	Inference Pipeline Coherence	Batching–caching–serving conflicts altering deployed system behavior	Control [3]
ai.52	C	Deployment Drift Signal Aggregation	Weak-signal regime-shift detection; structural drift before threshold breach	Efficiency [6]

These diagnostics do not claim that evaluation–deployment divergence can be eliminated. They identify the *structural conditions* from which divergence arises, and they provide a vocabulary for reasoning about projection mismatch at a level that is independent of specific hardware, runtime, or framework choices.

8.7. Engagement and Licensing Context

The diagnostic applications referenced in this paper are structured instruments within the Structural Operator Resonance Theory (SORT) framework [1]. They formalize structural conditions in complex AI systems and provide a consistent vocabulary for evaluation–deployment divergence analysis.

In engagement contexts, SORT applications serve as analytical instruments for:

- **Evaluation audit:** Identifying projection boundaries beyond which benchmark results may not generalize.
- **Deployment structural mapping:** Characterizing runtime coupling surfaces that produce behavioral divergence.
- **Governance support:** Distinguishing model-level drift from system-level structural evolution.
- **Architecture-level risk characterization:** Identifying conditions under which evaluation stability does not imply deployment resilience.

Each application diagnoses structural conditions. Mitigation strategies, architectural adjustments, or integration approaches depend on the specific system environment.

SORT applications are available under structured licensing arrangements. Licensing architecture and engagement models are described in separate documentation.

Scope, Non-Claims, and Intended Use

What This Analysis Is

This paper provides:

- A **diagnostic framework** introducing shared vocabulary for evaluation–deployment structural divergence in large-scale AI systems
- A **structural taxonomy** for categorizing divergence phenomena across projection instability, benchmark integrity, structural drift, control coherence, and weak-signal aggregation
- A **formal mapping** of these diagnostic layers to applications within the SORT-AI framework

The analysis is system-agnostic and vendor-neutral. It does not assume a specific model provider, benchmark suite, or deployment topology.

What This Analysis Is Not

Table 2. Explicit Non-Claims

Non-Claim	Explanation
Not a safety certification	No specific model or system is assessed for regulatory compliance
Not an alignment claim	No attribution of intent, awareness, or deceptive capability
Not prescriptive	No architectural mandate or integration blueprint is provided
Not a benchmark study	No evaluation scores or vendor comparisons are presented
Not a mitigation guarantee	Structural diagnosis does not imply elimination of divergence

These boundaries reflect a deliberate focus on structural characterization rather than implementation.

Intended Audience and Use

Primary Audience. AI Safety, Evaluation, and Runtime Infrastructure teams responsible for pre-deployment testing, serving architecture, control layers, and post-deployment monitoring.

Governance and Compliance Stakeholders. Governance and Compliance teams responsible for regulatory evidence generation, auditability, and post-deployment risk interpretation

Executive and Strategy Functions. CTO offices and architecture leads evaluating long-term robustness of large-scale AI deployments.

Intended Use. This analysis serves as:

- a structural lens for interpreting why evaluation stability may not predict deployment behavior,
- a vocabulary bridge between evaluation, infrastructure, and governance functions,
- a diagnostic anchor for identifying projection mismatch in large-scale AI systems.

It complements, but does not replace, system-specific experimentation, validation, or regulatory review processes.

9. Conclusion

Evaluation–deployment divergence is not primarily a question of alignment fragility or model intent. It is a geometric mismatch between two structurally different spaces: evaluation as constrained projection, and deployment as dynamically coupled execution. This structural framing does not replace alignment research. It clarifies the system-level conditions under which alignment signals are observed and interpreted.

As AI systems scale—with extended context windows, multi-agent orchestration, heterogeneous accelerator stacks, and cross-layer control feedback—this projection mismatch becomes a first-order architectural variable. Behavioral differentiation between test and deployment environments [8] is a surface manifestation of deeper structural non-equivalence.

The five diagnostic layers introduced in this paper provide a structured vocabulary for characterizing this non-equivalence:

- projection-boundary instability,
- benchmark geometry saturation,
- system-level structural drift,
- runtime control coherence,
- weak-signal regime detection.

Together, they shift the discussion from model-centric interpretation toward system-level structural analysis.

For hyperscale environments, this distinction is not theoretical. Scale increases coupling dimensionality. Coupling dimensionality increases divergence sensitivity. Without projection-aware diagnostics, evaluation stability can be misinterpreted as deployment robustness.

Governance frameworks emerging across jurisdictions increasingly require post-deployment monitoring and lifecycle risk assessment. These developments implicitly acknowledge that pre-deployment evaluation is insufficient. However, monitoring that does not distinguish between model-level change and coupling-level evolution remains structurally incomplete.

Projection awareness therefore becomes a governance principle as well as an architectural one.

Predictable large-scale AI deployment requires not only improved evaluation coverage, but structural observability across runtime coupling surfaces. Diagnostic clarity precedes mitigation. Structural understanding precedes control.

Evaluation measures behavior within projection. Deployment reveals behavior under coupling.

The distinction between the two is now an architectural reality.

Acknowledgments: The author acknowledges prior internal architectural work that informed the conceptual development of the diagnostic perspective presented in this paper. The author also acknowledges prior work within the SORT framework that informed the structural diagnostic mapping.

Conflicts of Interest: The author declares no conflicts of interest. The author is the developer of the SORT framework referenced in the diagnostic mapping section.

Data Availability Statement: No new data were generated in this study. All referenced data are available in the cited publications and publicly accessible sources listed in the references.

1. Wegener, G.H. (2025). The Supra-Omega Resonance Theory (SORT): A Closed Structural Architecture for Cross-Domain Scientific Analysis. *Preprints.org*, 2025. doi:10.20944/preprints202511.1783.v3
2. Wegener, G.H. (2025). SORT-AI: A Projection-Based Structural Framework for AI Safety—Alignment Stability, Drift Detection, and Scalable Oversight. *Preprints.org*, 2025. doi:10.20944/preprints202512.1334.v2
3. Wegener, G.H. (2026). SORT-AI: Runtime Control Coherence in Large-Scale AI Systems—Structural Causes of Cost, Instability, and Non-Determinism Beyond Interconnect Failures. *Preprints.org*, 2026. doi:10.20944/preprints202601.0298.v1
4. Wegener, G.H. (2026). SORT-AI: Interconnect Stability and Cost per Performance in Large-Scale AI Systems—Structural Causes, Diagnostic Operators, and Infrastructure-Level Consequences. *Preprints.org*, 2026. doi:10.20944/preprints202601.0161.v1
5. Wegener, G.H. (2026). SORT-AI: Agentic System Stability in Large-Scale AI Systems—Structural Causes of Cost, Instability, and Non-Determinism in Multi-Agent and Tool-Using Workflows. *Preprints.org*, 2026. doi:10.20944/preprints202601.1741.v1
6. Wegener, G.H. (2026). SORT-AI: Structural Efficiency Recovery in Hyperscale AI Systems—A Diagnostic Framework for Throughput, Control, and Orchestration Losses. *Preprints.org*, 2026. doi:10.20944/preprints202602.0015.v1
7. Wegener, G.H. (2026). An Operator-Projection Framework for Structural Risk Assessment in AI Infrastructure: Architecture, Applications, and Evidence Requirements. *SSRN Electronic Journal*, 2026. https://doi.org/10.2139/ssrn.6094907
8. Bengio, Y.; Clare, S.; Prunkl, C.; et al. (2026). International AI Safety Report 2026. *DSIT 2026/001*, February 2026. https://internationalaisafetyreport.org/publication/international-ai-safety-report-2026
9. Kapoor, S.; Widder, D.G.; Ensmenger, N.; Narayanan, A. (2025). Can We Trust AI Benchmarks? An Interdisciplinary Review of Current Issues in AI Evaluation. *arXiv preprint* arXiv:2502.06559. https://arxiv.org/abs/2502.06559
10. Maslej, N.; Fattorini, L.; Perrault, R.; et al. (2025). The AI Index 2025 Annual Report. *Stanford Institute for Human-Centered Artificial Intelligence*, Stanford University. https://hai.stanford.edu/ai-index/2025-ai-index-report
11. Jeon, M.; Venkataraman, S.; Phanishayee, A.; et al. (2024). Characterization and Prediction of Deep Learning Workloads in Large-Scale GPU Datacenters. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '24)*.
12. Dubey, A.; Jauhri, A.; Pandey, A.; et al. (2024). The Llama 3 Herd of Models. *arXiv preprint* arXiv:2407.21783.
13. Zhai, E.; et al. (2025). Aegaeon: Effective GPU Pooling for Concurrent LLM Serving on the Market. *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP '25)*.
14. Databricks Engineering. (2024). LLM Inference Performance Engineering: Best Practices. *Databricks Engineering Blog*. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices
15. Meta Engineering. (2024). Taming Tail Utilization of Ads Inference at Meta Scale. *Meta Engineering Blog*. https://engineering.fb.com/2024/07/10/production-engineering/tail-utilization-ads-inference-meta/
16. National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). *NIST AI 100-1*, January 2023. https://doi.org/10.6028/NIST.AI.100-1
17. National Institute of Standards and Technology. (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. *NIST AI 600-1*, July 2024. https://doi.org/10.6028/NIST.AI.600-1
18. European Parliament and Council of the European Union. (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). *Official Journal of the European Union*, L 2024/1689, 12 July 2024. https://eur-lex.europa.eu/eli/reg/2024/1689/oj