Beyond Model-Centric AI: Embedding Governance as an Architectural Component with Measurable Efficiency

Waleed Al Shehri
Department of Computing, College of Engineering and Computing in Al-Lith
Umm Al-Qura University, Makkah, Saudi Arabia

***Abstract*—Production AI systems typically apply governance—compliance checking, auditing, and explanation—as an external control layer that observes outputs after they are produced. In this paper, embedded governance is proposed: a governance gate that forms part of the system architecture itself, positioned between inference and release, so that no output leaves the system without a measurable governance assessment. For each candidate output, a governance score is computed from three factors that are checked mechanically rather than judged by a model: intent conformance, formulation traceability, and verification. On the basis of this score, each output is released, remediated, or blocked. Governance efficiency η, the ratio of intercepted error cost to governance cost, is further defined, by which the claim that governance pays for itself becomes falsifiable rather than asserted. Weights and thresholds are calibrated by maximizing η on a validation split and reported on held-out data. Evaluation uses the full credit-card fraud detection dataset of 284,807 transactions under a temporal split, so that the evaluation period follows the training period and distribution drift arises naturally. The calibrated gate intercepts 99% (95% CI 94.6–99.8) of severely shifted requests at a mean overhead of 3.74 ms, and η reaches 4.52 when releasing unreliable outputs carries cost, while remaining below break-even under accuracy-only cost models. A severity sweep locates the detection boundary and shows interception falling to 2% under mild shift. An ablation identifies distributional checks as the load-bearing components and reveals that η can be inflated by a gate that governs less, so efficiency must be read alongside coverage. Under identical workload, external monitoring and post-hoc assessment detect the same errors but prevent none, whereas inline placement reduces released error cost by 82.**

***Keywords*—AI governance; software architecture; embedded governance; governance efficiency; ablation study; failure semantics; out-of-distribution detection; MLOps**

I. Introduction

The rapid adoption of AI across healthcare, finance, and enterprise applications has created strong demand for governance frameworks that guarantee fairness, transparency, and regulatory conformity [1], [2]. Machine learning models now influence decisions that shape access to services, opportunities, and resources [3], and this influence is increasingly codified in regulation: the European Union's Artificial Intelligence Act [4] imposes obligations on high-risk systems, and ISO/IEC 42001 [5] defines management-system requirements for organizations that provide or use AI. Organizations consequently face a practical tension between accelerating AI development and installing the safeguards that these instruments require.

In current practice, this tension is resolved architecturally in one predominant way: governance is implemented as a control layer external to the AI system. Compliance monitors observe outputs after they are produced, audit pipelines record decisions retrospectively, and evaluation harnesses assess models offline. The literature on governance overhead reflects this assumption, in that it measures the cost an external governance apparatus imposes on a running system [6]. What this framing leaves unexamined is the architectural position of governance itself. The governed system produces its output first, and governance reacts.

In this paper, a different position is taken. Governance should be a component of the AI system's design rather than an external observer of its behaviour. An architecture is proposed in which a governance gate Γ is placed inside the critical path, between inference and release, so that every candidate output must pass a measurable assessment before it exists outside the system. The assessment poses three questions of each output: what was asked, how the result was formulated, and how it was verified. Each question is answered by a computable quantity rather than by the model's own judgment of itself. This distinction matters. Self-refinement, Reflexion, and LLM-as-a-judge ask a model, or a second model, to evaluate output quality, and therefore inherit the reliability limits of the judge [7], [8], [9]. The proposed gate instead evaluates artifacts that can be checked mechanically: schema and constraint conformance, provenance completeness, distributional validity, and confidence calibration.

The scope of the contribution is stated precisely. Governance frameworks, explainability methods, anomaly detection, documentation practices, and control libraries are established prior work [1], [12], [14], [19], [20], [22], and the observation that governance imposes measurable latency on real-time pipelines is established in [6]. What is new here is the formalization of governance as an inline architectural gate over computable factors, a cost model that renders the value of governance falsifiable through a single ratio, and an empirical characterization of the conditions under which embedded governance does and does not repay its cost. The contributions are as follows:

- C1. An embedded governance model. A governance function $G(x, \hat{y})$ over three computable factors, a three-way gating rule, and a governance efficiency measure η whose break-even point makes the value of governance an empirically testable claim.

- C2. An architecture that places the gate in the critical path, together with a trust model for the artifacts the gate consumes and explicit failure semantics for the case in which the gate itself becomes unavailable.
- C3. An empirical characterization on the full credit-card fraud dataset under a temporal split, comprising interval estimates of interception, a severity sweep that locates the detection boundary, an ablation of individual checks, the efficiency cost of a mandated traceability floor, behaviour under concurrency, and a comparison of governance placements under identical workload.

Section II positions embedded governance against related work. Section III presents the formal model. Section IV describes the architecture, its trust model, and its failure semantics. Section V details the experimental methodology and Section VI the results. Section VII discusses interpretation, threats to validity, and limitations, and Section VIII concludes.

II. Related Work

A. Architectural Frameworks for ML-Enabled Systems

Traditional software architecture frameworks were developed for deterministic, specification-driven systems, whereas in ML-enabled systems the functionality is inferred from data rather than specified at design time [15]. An empirical study of experts across more than twenty-five organizations identified stakeholder groups absent from standard frameworks and motivated dedicated viewpoints for data pipelines, training, deployment, monitoring, and governance [15]. Generative AI introduces further architectural concerns of its own [16]. The operational consequences of treating ML systems as software have been documented since the identification of hidden technical debt in production ML [17] and the subsequent formulation of production-readiness rubrics [18]; both emphasize monitoring and testing, and both stop short of placing an enforcement point in the serving path. The present paper operationalizes the governance viewpoint by giving it a concrete inline component with defined interfaces and a measurable cost.

Eisenberg et al. [14] observed that AI governance is fragmented across isolated risk frameworks, divergent regulations, and high-level standards lacking implementation guidance, and proposed a unified control framework integrating a risk taxonomy with a library of controls. Documentation practices such as model cards [19] and datasheets for datasets [20] standardize what should be recorded about models and data. Such artifacts specify *what* governance should check and record; they are complementary to this work, which specifies *where* in the architecture checking occurs, what it costs, and what it prevents.

B. Governance of Production AI Systems

The performance cost of governance in production machine learning has been examined primarily under the assumption of an external governance apparatus. Vaddepalli [6] frames the resulting tension: runtime policy checks, lineage collection, and audit logging impose latency on real-time pipelines, which forces organizations to choose between slowing pipelines for compliance and disabling checks for speed. Metadata-driven automation is proposed as mitigation, with reported reductions in governance-related latency of up to 40%. Regulatory analyses similarly identify metadata obligations as a dominant share of compliance requirements [2], [4]. Throughout this literature governance is measured as an overlay on the serving path rather than as a component of it. The question addressed here is inverted: if governance is placed inside the serving path, what value does it return for what it costs? Section VI-F answers this question by comparing placements experimentally.

C. Self-Assessment, Guardrails, and Evaluation

A separate family of techniques improves or evaluates model outputs through self-assessment. Constitutional AI [10] instills normative constraints during training. Self-Refine [8] and Reflexion [7] prompt a model to critique and revise its own outputs at inference time. LLM-as-a-judge [9] employs a second model to grade outputs after the fact. Assurance frameworks such as the NIST AI Risk Management Framework [2] and ISO/IEC 42001 [5] operate at the level of organizational process. Guardrail toolkits are the closest prior art in placement: NeMo Guardrails [11] interposes programmable rails between a language model and its user and can block responses, which shares the inline position adopted here. It differs in two respects that matter for the present contribution: its rails are specified as dialogue policies for conversational systems rather than as a model-agnostic score over verifiable artifacts, and it carries no accounting of what the interposed check costs relative to the harm it prevents. Table I summarizes the comparison by mechanism. Two properties are jointly distinctive of the gate proposed here: assessment rests on mechanically computable artifacts rather than model judgements, and the architecture is accompanied by an explicit cost model. To the best of the author's knowledge, no prior approach combines inline pre-release gating with a quantified cost-of-governance accounting.

TABLE I. Differentiation by Mechanism

Approach	When	Basis	Blocks	Cost
Constitutional AI [10]	Training	Learned preferences	No	No
Self-Refine, Reflexion [8], [7]	Inference prompt	Model self-critique	No	No
LLM-as-a-judge [9]	Post-hoc	Second model	No	No
NIST RMF, ISO 42001 [2], [5]	Process	Documentation	No	No
Guardrails [11]	Inline	Dialogue policy	Yes	No
External governance [6]	Parallel	Policies, metadata	No	Latency
Embedded gate (this work)	**Inline**	**Computable artifacts**	**Yes**	**Full**

D. Verification Components

The checks composing the verification factor draw on established methods. Out-of-distribution detection using softmax confidence was established as a baseline by Hendrycks and Gimpel [21]; isolation forests provide an efficient density-free anomaly score [22]; and the literature on concept drift adaptation characterizes how input and target distributions evolve over time [23]. Explainable AI underpins transparency obligations [1], with SHAP [13] the de facto model-agnostic standard and SensX [12] reporting higher explanation accuracy at substantially lower cost for high-dimensional inputs. In the proposed architecture these methods are not contributions but components: they instantiate the verification factor and the remediation action, and any of them may be substituted without altering the gating mechanism or the efficiency accounting.

III. The Embedded Governance Model

A. Setting

An AI system is considered that, for each request x, produces a candidate output $\hat{y}$ together with two auxiliary records. The response record carries the prediction, the confidence vector, and the model version. The provenance record carries the model hash, the feature schema version, the dataset reference, the configuration hash, a timestamp, and a data-quality flag. The governance gate Γ is invoked after inference and before release, and the output does not leave the system without its decision.

B. The Governance Function

For each pair $(x, \hat{y})$ the gate computes:

$$G(x, \hat{y}) = w_I \, g_I(x, \hat{y}) + w_F \, g_F(x, \hat{y}) + w_V \, g_V(x, \hat{y}), \quad (1)$$

with $w_I + w_F + w_V = 1$. Each factor answers one governance question with a mechanically computable quantity in $[0, 1]$.

1) Intent conformance g_I: records the fraction of structural checks the response passes: required fields present, prediction within the valid label set, and probability vector well formed. These are deterministic schema and constraint checks and are independent of any model's judgment.

2) Formulation traceability g_F: records the fraction of required provenance artifacts that are present and valid for the specific output. A release whose lineage cannot be reconstructed scores low irrespective of its predictive quality. This factor has no analogue in self-assessment approaches, which do not inspect lineage.

3) Verification g_V: records the mean of a battery of validation checks: a normalized anomaly score obtained from an isolation forest fitted on the training data [22], the model's confidence for the predicted class [21], and the fraction of input features lying within tolerance-extended training bounds.

C. Gating Rule

Released outputs proceed unmodified. Blocked outputs are withheld and escalated. Remediated outputs receive additional treatment before or alongside release; in this work that treatment is explanation generation and flagging, while regeneration and human review are alternatives with higher cost. Every decision, including each release, is written to the audit trail, so audit coverage is complete by construction.

$$\mathrm{mode}(x, \hat{y}) = \begin{cases} \text{release}, & G \geq \theta_{\text{pass}} \\ \text{block}, & G \leq \theta_{\text{fail}} \\ \text{remediate}, & \text{otherwise.} \end{cases} \quad (2)$$

D. Governance Efficiency

Whether the gate repays its cost is an empirical question. Let $c_e(x)$ denote the cost of releasing an erroneous output under a declared error model, c_{fb} the cost of wrongly blocking a correct output, c_{rem} the cost of remediation, L_{gate} the total gate latency, and λ the cost per millisecond of added latency. The cost of governing and the value it returns are:

$$C_{\text{gov}} = \lambda L_{\text{gate}} + c_{fb} N_{\text{fb}} + c_{rem} N_{\text{rem}}, \quad (3)$$

$$\eta = \frac{\sum_{x \in \text{intercepted}} c_e(x)}{C_{\text{gov}}}. \quad (4)$$

Governance repays its cost when $\eta > 1$. The claim is falsifiable: for a given deployment, error model, and cost parameters, η is measured rather than asserted. The remediation term in (3) is necessary, since without it a configuration that remediates all traffic attains an arbitrarily favourable score, an artifact observed and eliminated during development.

Two cautions govern the use of η, and both are borne out experimentally in Section VI. First, η is defined relative to a declared error model, so values computed under different error models are not comparable. Second, and less obviously, η is a ratio whose denominator is the cost of governing, so a gate that governs less incurs less cost and may report a higher ratio while intercepting almost nothing. Efficiency must therefore be reported alongside a coverage measure such as the interception rate, and it should not be used as a sole optimization objective without a coverage constraint.

E. Calibration

The weights and thresholds are hyperparameters of the gate. They are selected by maximizing η on a validation split of the evaluation stream, and all results are reported on the disjoint holdout split. Because the selection depends on the distribution of the traffic being governed, the calibration must be repeated whenever the dataset, the split, or the workload changes; values fitted to one distribution are not transferable to another. This point is not incidental. During the preparation of this revision, gate parameters fitted to a 20,000-transaction subsample were applied unchanged to the full dataset under a temporal split, and the resulting gate remediated approximately 2,825 requests, so that remediation cost dominated the denominator of (4) and interception fell to 40%. The behaviour

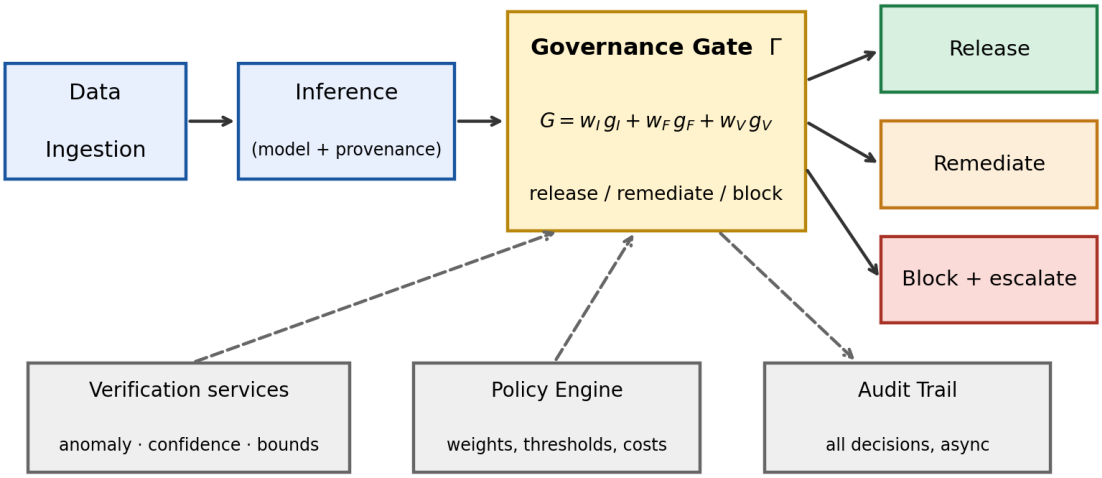

Fig. 1. Embedded governance architecture. The gate Γ occupies the critical path; policy configuration and audit recording are out of band.

was an artifact of stale calibration rather than a property of the architecture, and it illustrates why re-calibration is part of the method rather than a detail of its application.

IV. Architecture

Fig. 1 shows the architecture. The serving path runs from data ingestion through inference to the governance gate Γ and thence to one of release, remediation, or blocking. Three components surround the gate without occupying the path: verification services supply the inputs to g_V; a policy engine supplies weights, thresholds, and cost parameters, so that governance intensity is an operational configuration rather than a code change; and an audit trail records every decision asynchronously.

This placement is the substantive architectural commitment of the paper. Prior governance architectures separate data flow from control and audit flows precisely so that governance never blocks the data path [6]. The separation of control flow and audit flow is retained here, but the governance decision itself is deliberately placed in the data path, because blocking is the mechanism by which an unreliable output is prevented from reaching a consumer, and (4) is the instrument that determines whether that mechanism is worth its cost. The layered organization of ML-enabled systems is retained around the gate, and the stakeholder groups identified in [15] map onto it: data engineers own ingestion and the provenance artifacts consumed by g_F; machine learning engineers own inference and the verification services; compliance officers and auditors own the policy engine and the audit trail; and affected communities are served by the remediation path, which attaches explanations to borderline releases.

A. Trust Model

The gate consumes artifacts produced by other components: predictions and confidence vectors from the inference service, and provenance records from the pipeline. Its guarantees are therefore conditional on assumptions about those components, which are stated here explicitly.

The gate assumes that the inference service and the preprocessing pipeline are not adversarial, and that the artifacts they emit describe the computation that actually occurred. Under this assumption the gate detects three failure classes: malformed or incomplete responses, through g_I; missing or unresolvable lineage, through g_F; and inputs that lie outside the training distribution or on which the model is not confident, through g_V.

Three classes of compromise are not detected, and the distinction matters for any deployment in a regulated setting. First, a compromised inference component can emit a well-formed response with a fabricated confidence vector; because g_V consumes the reported confidence, an output that is wrong yet reported as confident, and whose input lies within the training distribution, will pass. Second, a compromised pipeline can emit syntactically complete provenance containing false values, such as a model hash that does not correspond to the model that served the request; g_F verifies presence and format, not authenticity. Third, an adversary able to craft inputs that are simultaneously misclassified and unremarkable to the anomaly detector defeats the verification factor by construction, since the factor is a statistical test rather than a security control.

Two mitigations follow directly and are recommended for deployments with a stronger threat model. Provenance records can be signed at the point of production, so that g_F verifies a signature rather than the presence of a field, which converts the second class from undetectable to detectable. Confidence can be recomputed by the gate from the model output rather than accepted from the caller, which converts the first class from undetectable to detectable at the cost of a second inference pass. Neither mitigation is evaluated here, and the empirical results in Section VI should be read as holding under the non-adversarial assumption stated above. Reporting this boundary explicitly is preferable to leaving the trust assumptions implicit, since a governance component whose assumptions are undocumented invites precisely the misplaced confidence that governance exists to prevent.

B. Failure Semantics

A component placed on the serving path must define its behaviour when it is unavailable, times out, or receives malformed metadata. Two policies bound the design space. Under *fail-open*, requests bypass the gate and are released ungoverned, which preserves availability and forfeits safety for the affected requests. Under *fail-closed*, requests are blocked, which preserves safety and forfeits availability. The gate is a dependency of the serving path in the sense described by the stability patterns literature [24], so a circuit breaker around Γ is the natural implementation: repeated timeouts trip the breaker into whichever policy the operator has declared, rather than allowing per-request timeouts to accumulate into a queue collapse.

Three refinements are available and are noted for completeness. Malformed metadata need not be treated as gate failure at all, since an absent or unparseable provenance record is exactly the condition g_F is designed to score, and such a request should be assessed with a low traceability score rather than bypassed. Loss of a single verification service can be treated as partial degradation, in which the affected check is neutralized and the remaining checks continue, which is the ablation configuration evaluated in Section VI-C and is therefore already characterized quantitatively. Finally, the policy can be made risk-dependent, with fail-closed applied to high-consequence request classes and fail-open to the remainder, though this requires a request classifier and is not evaluated here.

Section VI-G quantifies the trade-off between the two bounding policies at several failure rates, so that the choice can be made against a declared cost model rather than by intuition.

V. Experimental Methodology

A. Test Environment

All reported experiments were executed on a virtual private server running Debian GNU/Linux (kernel 6.12), Python 3.13.5, scikit-learn 1.9.0, and NumPy 2.5.1, with two shared virtual cores. Every run writes a manifest recording configuration, environment, and random seeds; complete logs, per-sample records, and machine-readable results accompany the code.

B. Dataset and Model

The full Credit Card Fraud Detection dataset is used: 284,807 transactions with 30 features and 492 fraud cases (0.173%). Earlier work on this architecture used a 20,000-transaction subsample, which left only about seven fraud cases in the test split and rendered rate estimates statistically fragile. The full dataset is partitioned temporally rather than at random: the model is trained on the earliest 227,845 transactions and evaluated on the latest 56,962, so that the evaluation period follows the training period exactly as it would in deployment. The test split contains 75 fraud cases. Because fraud patterns evolve over time, this split introduces genuine distribution drift rather than the synthetic perturbation used previously (Table II).

TABLE II. Dataset, Split, and Model Configuration

Property	Value
Dataset	Credit Card Fraud Detection (Kaggle), full file
Transactions / features	284,807 / 30
Fraud cases	492 (0.173%)
Split	Temporal: train on earliest 227,845, test on latest 56,962
Fraud cases, train / test	417 / 75
Model	Random forest, 50 estimators, depth 10
Accuracy / precision / recall	99.953% / 0.962 / 0.667
Misclassifications, clean test	27 of 56,962

C. Fault Injection and Severity Levels

Two controlled fault classes are injected into the evaluation stream. Out-of-distribution requests are constructed by scaling donor feature vectors by a factor s with additive noise. Rather than fixing a single value of s, the severity is swept over $s \in \{1.25, 1.5, 2, 3, 5, 8\}$, so that gate behaviour is characterized across the range from mild to extreme shift instead of at one favourable operating point. Provenance is degraded for 5% of requests by removing three of the seven required lineage artifacts. All injections are seeded and logged.

D. Error Models

η depends on what counts as an erroneous release. Three error models are evaluated. The reliability model captures the governance-relevant position that releasing an unreliable output is itself a harm, even if the prediction happens to be correct; the accuracy-only models charge nothing for unreliable-but-lucky releases. Cost parameters are $c_{fb} = 1$ per false block, $c_{rem} = 0.5$ per remediation, and $\lambda = 0.001$ per millisecond of gate latency. All η components are reported raw, so η can be recomputed under any alternative cost assignment without re-running experiments.

E. Calibration Protocol

Weights and thresholds are hyperparameters of the gate and are re-calibrated for the dataset under evaluation: a weight sweep over the simplex (step 0.05) followed by a threshold grid ($\theta \in [0.50, 0.99]$, step 0.02) selects the configuration maximizing η on a validation half of the evaluation stream, with all results reported on the disjoint holdout half. Calibration on the full dataset selected $w = (0.10, 0.00, 0.90)$ and $\theta = (0.68, 0.72)$, with validation $\eta = 4.03$ and mean gate latency 3.74 ms. It is worth recording that this configuration is almost identical to the one selected independently on the 20,000-transaction subsample, which provides incidental evidence that the calibration procedure is stable across sampling regimes even though the resulting efficiency values are not transferable (Fig. 2).

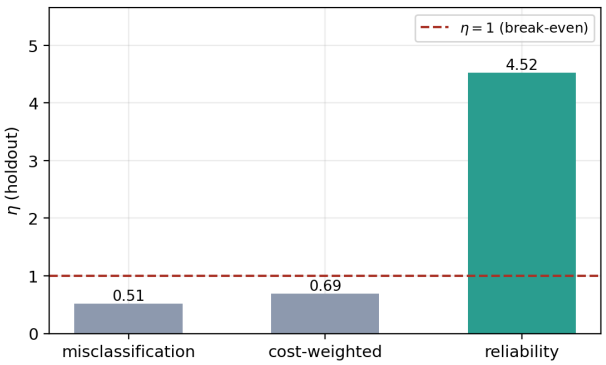

Fig. 2. Governance efficiency on the holdout stream under the three error models. Only the reliability model exceeds break-even.

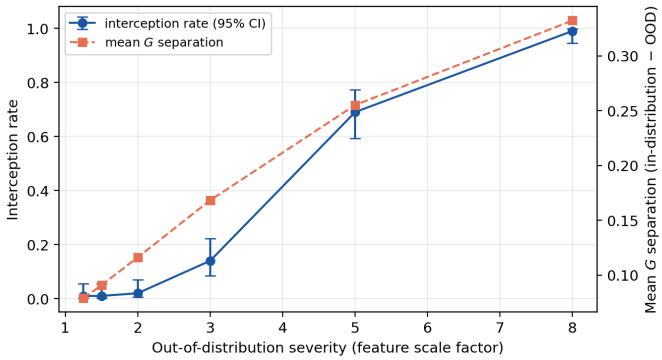

Fig. 3. Interception rate with 95% confidence intervals and mean governance-score separation across shift severity.

VI. Results

A. Interception and Efficiency on the Full Dataset

Table III reports gate behaviour on the holdout stream. Of 38 erroneous outputs, 16 were intercepted, a rate of 0.42 (95% CI 0.28–0.58); of 100 severely shifted out-of-distribution requests, 99 were intercepted (95% CI 0.95–1.00). The gate released 56,938 requests, remediated 25, and blocked 99. Under the reliability error model $\eta = 4.52$, comfortably above the break-even value of 1. Under accuracy-only models η is 0.51 and 0.69, below break-even: the classifier commits too few outright errors on this dataset for interception value to cover the cost of governing 57,000 requests, and blocked out-of-distribution outputs whose predictions happened to be correct are charged as false blocks. The conclusion drawn previously is therefore unchanged in direction and more precisely bounded: embedded governance is justified when the cost model treats unreliable releases as harmful, and not otherwise.

TABLE III. Gate Behaviour and Efficiency on the Holdout Stream

Metric	Value
Released / remediated / blocked	56,938 / 25 / 99
Errors in stream / intercepted	38 / 16
Error interception rate (95% CI)	0.42 (0.28–0.58)
OOD interception at $s = 8$ (95% CI)	0.99 (0.95–1.00)
Mean gate latency	3.74 ms
η, misclassification model	0.51
η, cost-weighted model	0.69
η, reliability model	4.52

B. Detection Boundary Across Shift Severity

Fig. 3 and Table IV report interception as a function of shift severity. Interception is 0.01 at $s = 1.25$, 0.02 at $s = 2$, 0.14 at $s = 3$, 0.69 at $s = 5$, and 0.99 at $s = 8$, while the mean separation between in-distribution and shifted governance scores grows monotonically from 0.08 to 0.33. Two conclusions follow. First, the high interception rate reported for severe shift is not representative of mild shift: a single severity level overstates what the gate detects. Second, the detection boundary on this configuration lies between $s = 3$ and $s = 5$, which is a concrete and reusable characterization rather than a single headline figure. Mild covariate shift is not reliably detected by the verification battery employed here, and this is a property of the checks rather than of the architecture: substituting a more sensitive drift detector would move the boundary without altering the gating mechanism or the efficiency accounting.

TABLE IV. Interception as a Function of Out-of-Distribution Severity

Severity s	Intercepted	Rate (95% CI)	G separation
1.25	1 / 100	0.01 (0.00–0.05)	0.079
1.50	1 / 100	0.01 (0.00–0.05)	0.091
2.00	2 / 100	0.02 (0.01–0.07)	0.116
3.00	14 / 100	0.14 (0.09–0.22)	0.168
5.00	69 / 100	0.69 (0.59–0.77)	0.255
8.00	99 / 100	0.99 (0.95–1.00)	0.332

C. Contribution of Individual Checks

Table V and Fig. 4 report the effect of disabling each check in turn. Removing the anomaly check collapses out-of-distribution interception from 0.99 to 0.01; removing the confidence check reduces it to 0.62 and lowers η from 57.6 to 20.4; removing the bounds check reduces interception to 0.93 and η to 46.8. Schema validation and provenance verification leave interception unchanged and alter η by less than 1%. Distributional checks are therefore the load-bearing components of the verification battery on this dataset, while the intent and traceability factors contribute governance value that this efficiency measure does not price.

The ablation also exposes a limitation of η itself, which merits explicit statement. With the anomaly check disabled, η rises to 480 even as interception falls to 0.01. The reason is structural: η is a ratio whose denominator is the cost of governing, so a gate that governs almost nothing incurs almost no cost, and any interception it happens to make yields a large ratio. Efficiency alone can thus be maximized by inaction. η must therefore be read together with a coverage measure such as the interception rate, and it should not be used as a sole optimization objective without a coverage constraint. This limitation was surfaced by the ablation and was not visible in the aggregate results reported previously.

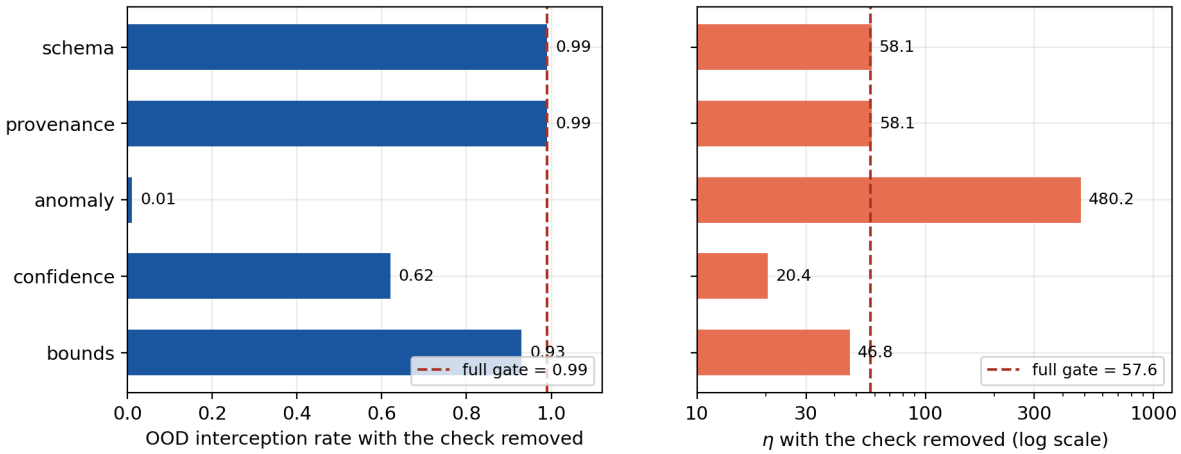

Fig. 4. Ablation. Left: out-of-distribution interception with each check removed. Right: η with each check removed; the increase when anomaly detection is removed reflects a gate that governs less, not one that governs better.

TABLE V. ABLATION OF INDIVIDUAL CHECKS (RELIABILITY ERROR MODEL)

Configuration	η	OOD int.	ms/req
Full gate	57.6	0.99	0.028
Without schema validation	58.1	0.99	0.025
Without provenance check	58.1	0.99	0.024
Without anomaly detection	480.2	0.01	0.000
Without confidence check	20.4	0.62	0.024
Without bounds checking	46.8	0.93	0.025

TABLE VI. CONSTRAINED CALIBRATION UNDER POLICY FLOORS ON w_F

Floor	(w_I, w_F, w_V)	η	OOD int.	Cost
0.0	(0.05, 0.10, 0.85)	5.14	0.97	—
0.1	(0.05, 0.10, 0.85)	5.14	0.97	0.0%
0.2	(0.00, 0.20, 0.80)	4.86	0.93	5.5%
0.3	(0.00, 0.30, 0.70)	3.88	0.84	24.5%

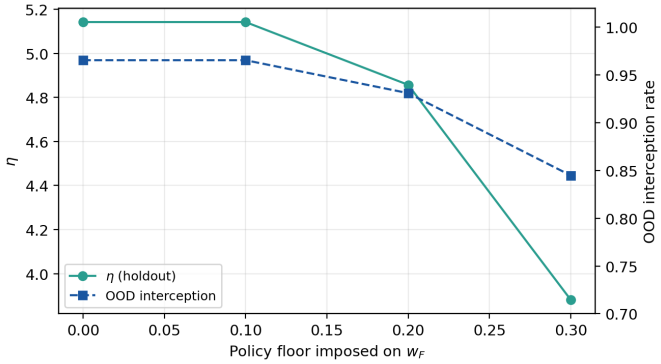

Fig. 5. Efficiency and interception as a function of the policy floor imposed on formulation traceability.

D. The Price of a Traceability Policy Floor

Unconstrained calibration assigns zero weight to formulation traceability, because the injected provenance faults are statistically independent of output errors and so weighting the factor produces false blocks without intercepting error cost. Compliance obligations, however, do not depend on predictive value. Table VI therefore reports calibration subject to a policy floor on w_F. A floor of 0.1 costs nothing measurable; a floor of 0.2 costs 5.5% of η and reduces interception from 0.97 to 0.93; a floor of 0.3 costs 24.5% of η and reduces interception to 0.84. The framework thus supports governance requirements that are mandated rather than earned, and quantifies their price instead of arguing about it, which is the intended use of the measure (Fig. 5).

E. Behaviour Under Load

Because the gate occupies the serving path, its behaviour under concurrency determines whether it is deployable. Table VII reports a concurrency ladder on the two-core test server. Throughput is 242 requests per second at concurrency 1 and remains between 202 and 260 across the range, indicating that the server saturates almost immediately; the additional concurrency is absorbed as queueing rather than as parallelism. Median latency rises from 3.9 ms to 102.6 ms and the 99th percentile from 6.9 ms to 837.7 ms between concurrency 1 and 64, while CPU utilisation remains near 107% of a single core and resident memory is stable between 374 and 379 MB. The mean latency of 3.74 ms reported earlier is therefore representative only of the unloaded case, and tail latency, not mean latency, governs capacity planning for this component. Memory stability across the ladder indicates that the gate holds no unbounded per-request state (Fig. 6).

TABLE VII. GATE BEHAVIOUR UNDER INCREASING CONCURRENCY

Conc.	req/s	p50	p95	p99	CPU	RSS
1	242.5	3.9	5.5	6.9	109%	374
2	207.4	4.7	36.8	87.2	108%	374
4	202.3	11.6	66.5	116.9	101%	374
8	205.9	15.4	137.8	223.2	107%	374
16	202.0	34.6	277.1	448.6	108%	375
32	250.0	60.3	384.9	561.0	109%	376
64	260.2	102.6	586.2	837.7	109%	379

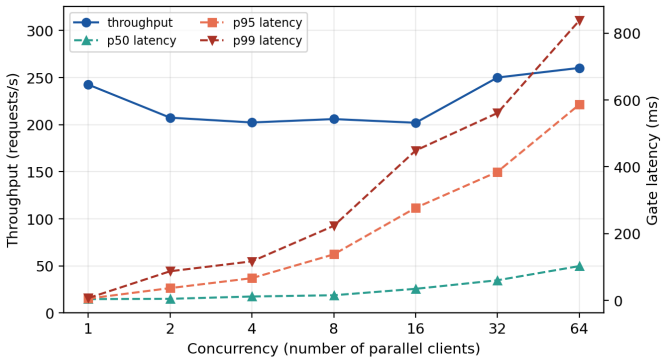

Fig. 6. Throughput and latency percentiles across the concurrency ladder.

F. Comparison of Governance Placements

Table VIII compares four architectural placements under an identical workload, identical cost parameters, and identical per-request assessments; only the position of governance relative to release differs. No governance releases 1,270 units of error cost. An external monitor sampling 10% of traffic detects 10 errors and prevents none, still releasing 1,270 units. Post-hoc assessment detects all 104 errors and prevents none, again releasing 1,270 units while incurring the full assessment cost. The inline gate detects the same 104 errors and prevents all of them, reducing released error cost to 230 units, a reduction of 82%. The comparison isolates the architectural variable: detection and prevention are separated by placement alone, and only inline placement converts assessment into avoided harm. This is the central claim of the paper, and it is here measured rather than argued (Fig. 7).

G. Failure Semantics

A component on the serving path must define its behaviour when it becomes unavailable. Two policies are evaluated: under fail-open, requests bypass the gate and are released ungoverned, preserving availability at the cost of safety; under fail-closed, they are blocked, preserving safety at the cost of availability. Table IX quantifies the trade-off at three failure rates. At a failure rate of 1%, fail-open releases 575 ungoverned outputs while fail-closed denies 574 correct ones. The asymmetry appears in efficiency: fail-open leaves η essentially unchanged between 4.29 and 4.52, because ungoverned releases cost only the errors they carry, whereas fail-closed degrades η from 3.62 to 0.34 as the failure rate rises to 5%, since every denied correct output is charged as a false block. Neither policy dominates. The choice is a deployment decision

TABLE VIII. GOVERNANCE PLACEMENTS UNDER IDENTICAL WORKLOAD AND COSTS

Placement	Released	Detect.	Prev.	η
No governance	1,270	0	0	—
External monitor	1,270	10	0	0.00
Post-hoc assessment	1,270	104	0	0.00
Inline gate (ours)	**230**	104	**104**	**4.52**

TABLE IX. FAIL-OPEN AND FAIL-CLOSED BEHAVIOUR UNDER GATE FAILURE

Rate	Policy	Ungov.	Denied	η
0.1%	fail-open	57	0	4.52
0.1%	fail-closed	0	57	3.62
1%	fail-open	575	0	4.48
1%	fail-closed	0	574	1.29
5%	fail-open	2,863	0	4.29
5%	fail-closed	0	2,855	0.34

determined by the relative cost of an unreviewed release and a denied service, and the framework makes that comparison explicit for a given cost model (Fig. 8).

VII. DISCUSSION

A. What the Results Establish?

Three findings are established by the experiments, and they are stated separately from the architectural implications that follow them, because the two carry different evidential weight.

First, placement determines prevention. Under an identical workload, identical cost parameters, and identical per-request assessments, external monitoring and post-hoc assessment detected errors without preventing any, while the inline gate prevented all of the errors it detected and reduced released error cost by 82% (Section VI-F). Since the only variable that differs across the four configurations is the position of governance relative to release, the difference is attributable to placement alone. This is the paper's central claim and it is now measured rather than argued.

Second, the value of governance is conditional on the cost model, and the condition is identifiable. On the same system and the same workload, η was 4.52 under a reliability-centric error model and 0.51 and 0.69 under accuracy-only models (Section VI-A). Embedded governance repays its cost when releasing an unreliable output is itself treated as a harm, and does not when only outright misclassification is charged. Since the reliability position is the operative assumption of regulatory instruments that require transparency and human oversight for high-risk systems [4], the condition is satisfied in precisely the settings where governance is mandated, but this should be stated as a conditional result rather than a general one.

Third, the gate's detection capability has a boundary that can be located. Interception was 0.99 for severe shift and 0.02 for mild shift, with the transition between severity factors three and five (Section VI-B). The high figure alone would have overstated the mechanism; the curve characterizes it.

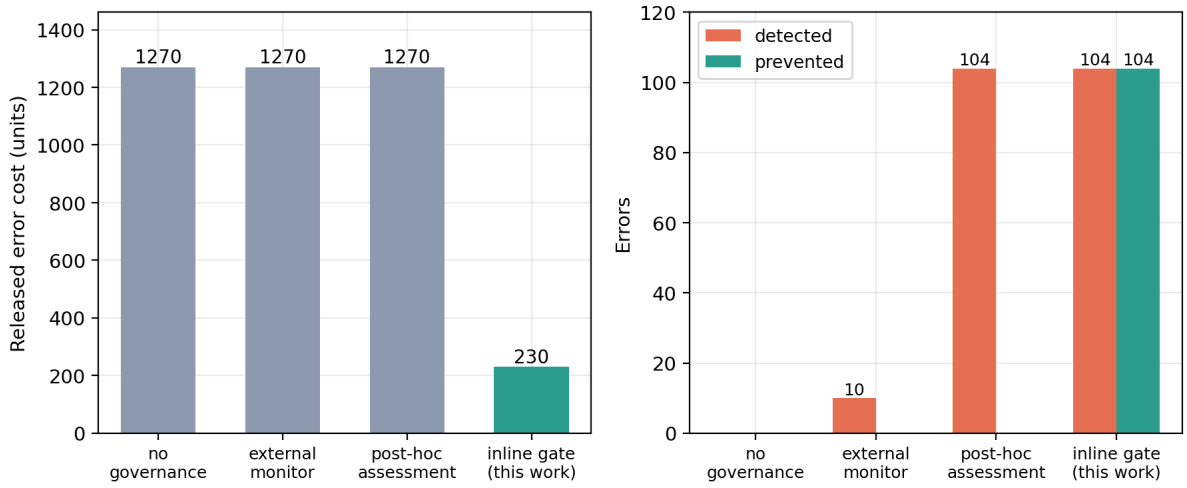

Fig. 7. Governance placements under identical workload. Left: released error cost. Right: Errors detected versus errors prevented.

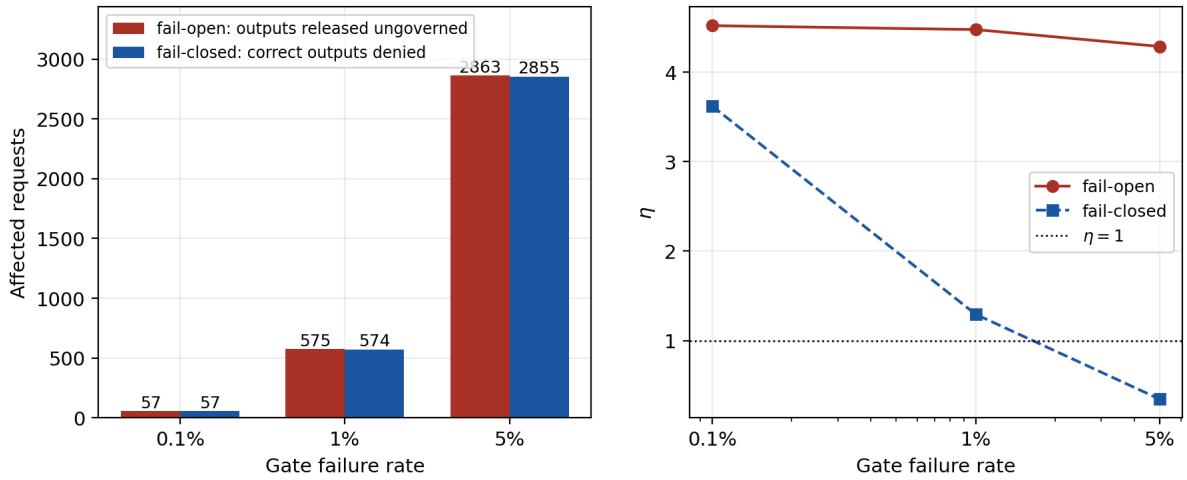

Fig. 8. Gate failure. Left: requests affected under each policy. Right: Efficiency under each policy as the failure rate increases.

The boundary is a property of the verification checks rather than of the architecture, and the drift-detection literature offers more sensitive alternatives [23] that would move the boundary without changing the gating mechanism or the efficiency accounting.

B. *Efficiency Must be Read with Coverage*

The ablation exposed a limitation of the efficiency measure proposed in this paper. Disabling anomaly detection raised η from 57.6 to 480 while out-of-distribution interception fell from 0.99 to 0.01 (Section VI-C). The cause is structural rather than incidental: η divides intercepted value by the cost of governing, and a gate that governs almost nothing incurs almost no cost, so any interception it happens to make produces a large ratio.

Two consequences follow for practitioners. Efficiency should never be reported without a coverage measure beside it, and the two together, not either alone, characterize a governance configuration. Where η is used as an optimization objective, it should be maximized subject to a coverage constraint, for example a minimum interception rate on a held-out set of known unreliable inputs, since unconstrained maximization admits the degenerate solution of governing nothing. This limitation was not visible in the aggregate results of the previous version and was surfaced only by the ablation; it is reported here because a measure whose failure modes are undocumented is more dangerous than one whose failure modes are known.

C. *Utility and Policy in Factor Weighting*

Unconstrained calibration assigned zero weight to formulation traceability, because in the fault model employed provenance degradation was statistically independent of output error, so that weighting the factor produced false blocks without intercepting error cost. This is a genuine tension rather than an artifact. Utility-maximizing calibration discards factors that do not predict errors, whereas provenance completeness is required by regulation and by documentation practice [19], [20], [5] irrespective of its predictive value, and an auditor

cannot accept the explanation that an optimizer removed lineage checking.

The framework accommodates the tension directly and prices it. Weight selection becomes a constrained optimization in which policy sets a floor and η is maximized within it. On the present data a floor of 0.1 cost nothing measurable, a floor of 0.2 cost 5.5% of efficiency, and a floor of 0.3 cost 24.5% and reduced interception from 0.97 to 0.84 (Section VI-D). The recommendation for practitioners is therefore concrete: mandated checks should be expressed as floors on their factor weights, or enforced as hard vetoes outside G where the obligation is absolute, and the resulting efficiency cost should be recorded as the measured price of compliance rather than left as an unexamined assumption.

D. Deployment Implications

Three operational points follow from the results and are stated as guidance rather than as findings. Capacity planning for the gate should use tail latency rather than mean latency: on the two-core test server the mean of 3.74 ms described only the unloaded case, while the 99th percentile reached 837.7 ms at concurrency 64 (Section VI-E). Because throughput saturated almost immediately and additional concurrency was absorbed as queueing, admission control at the gate is preferable to unbounded queueing, and horizontal replication of the gate is the appropriate response to load, since the component holds no unbounded per-request state, as indicated by the stable resident memory across the ladder. Finally, the failure policy should be declared explicitly and provisioned for: fail-open preserved efficiency and released 2,863 ungoverned outputs at a 5% failure rate, whereas fail-closed preserved safety and drove efficiency below break-even (Section VI-G), so neither is a safe default in the absence of a declared cost model.

E. Threats to Validity

1) Construct validity: The quantity η operationalizes the value of governance in terms of a declared error model and declared cost parameters. Different declarations yield different values, and the measure is meaningful only relative to them. All components are reported raw so that η can be recomputed under alternative assumptions without repeating the experiments.

2) Internal validity: Out-of-distribution requests and provenance faults are synthetic and seeded, which permits controlled severity but does not reproduce the structure of natural shift. The temporal split mitigates this in part, because the evaluation period genuinely follows the training period, but the injected faults remain artificial. The independence between injected provenance faults and output errors is an artifact of the fault model, and a deployment in which lineage degradation correlates with model malfunction would assign formulation traceability a higher calibrated weight.

3) External validity: Evaluation uses one tabular dataset, one classifier family, and one two-core server. Absolute latencies and throughput are environment-specific, and the calibrated weights and thresholds are specific to this distribution, as Section III-E demonstrates directly. The architecture and the efficiency accounting are not tied to the modality, since the three factors are defined over artifacts that exist in other settings, but this is an architectural argument and not an empirical result.

4) Statistical conclusion validity: The full dataset provides 75 fraud cases in the test period, against approximately seven in the subsample used previously. All rates are reported with Wilson intervals, and the interval on the error interception rate, 0.28 to 0.58, remains wide because it is estimated over 38 erroneous outputs. Conclusions are drawn from the direction and magnitude of that interval rather than from its point estimate.

F. Limitations and Future Work

The principal limitation is that validation is confined to tabular classification. Extension to text, vision, and generative outputs is the natural next step, and it is a substantial one rather than a formality: g_I would generalize to constraint and schema conformance of structured generations, g_V would require detectors appropriate to unstructured inputs, and the error model would need to express harms that are not simply misclassification. The trust model in Section IV-A identifies signed provenance and gate-side confidence recomputation as mitigations that are specified but not evaluated. Further directions include learned rather than swept weights under policy constraints, coverage-constrained optimization of η as recommended above, integration of scalable explainers [12] as the remediation action, more sensitive drift detectors [23] to move the detection boundary, and coordination among multiple gates in pipelines of models.

VIII. Conclusion

This paper reframed AI governance as an architectural component rather than an external control layer. A gate placed between inference and release scores every candidate output on computable factors of intent, traceability, and verification, and a single efficiency measure renders the value of governance falsifiable. On the full credit-card fraud dataset under a temporal split, with weights and thresholds calibrated on a validation split, the gate intercepted 99% of severely shifted outputs at 3.74 ms of mean overhead and returned 4.52 times its cost under a reliability-centric error model, while falling below break-even under accuracy-only models. Compared under an identical workload, external monitoring and post-hoc assessment detected the same errors and prevented none, whereas inline placement reduced released error cost by 82%. The ablation showed that efficiency alone can be inflated by a gate that governs less, so efficiency and coverage must be reported together. As regulation increasingly requires governance of deployed systems, architectures should be able to state not only that they govern, but what governing costs, what it prevents, and under which assumptions those claims hold.

Declaration on Generative AI

Generative AI tools were used to assist with language refinement, implementation of experiment tooling, and drafting under the author's direction. All scientific ideas, the experimental design, and the conclusions were developed, critically reviewed, and validated by the author, who takes full responsibility for the content of this manuscript.

Data Availability

The Credit Card Fraud Detection dataset is publicly available on Kaggle (mlg-ulb/creditcardfraud). The experiment code, configurations, run logs, and machine-readable results supporting all figures and tables are available from the author upon reasonable request.

References

[1] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[2] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, 2023.

[3] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California Law Review*, vol. 104, pp. 671–732, 2016.

[4] European Parliament and Council, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)," *Official Journal of the European Union*, 12 July 2024.

[5] International Organization for Standardization, *ISO/IEC 42001:2023 — Information technology — Artificial intelligence — Management system*, 2023.

[6] R. K. Vaddepalli, "Smart governance for AI: Can metadata automation keep up with real-time ML pipelines?," *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 6, no. 2, pp. 119–124, 2025.

[7] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[8] A. Madaan *et al.*, "Self-Refine: Iterative refinement with self-feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] L. Zheng *et al.*, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

[11] T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen, "NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails," in *Proc. EMNLP (System Demonstrations)*, pp. 431–445, 2023.

[12] M. Aggarwal, N. Cogan, and V. Periwal, "Sensitivity based model agnostic scalable explanations of deep learning," bioRxiv 2025.02.21.639516, 2025.

[13] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] I. W. Eisenberg, L. Gamboa, and E. Sherman, "The Unified Control Framework: Establishing a common foundation for enterprise AI governance, risk management and regulatory compliance," arXiv:2503.05937, 2025.

[15] A. Moin, A. Badii, S. Günnemann, and M. Challenger, "Enhancing architecture frameworks by including modern stakeholders and their views/viewpoints," arXiv:2308.05239, 2024.

[16] M. Esposito, X. Li, S. Moreschini, N. Ahmad, T. Cerny, K. Vaidhyanathan, V. Lenarduzzi, and D. Taibi, "Generative AI for software architecture: Applications, challenges, and future directions," *Journal of Systems and Software*, 2026.

[17] D. Sculley *et al.*, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2503–2511, 2015.

[18] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction," in *Proc. IEEE Int. Conf. Big Data*, pp. 1123–1132, 2017.

[19] M. Mitchell *et al.*, "Model cards for model reporting," in *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAT*)*, pp. 220–229, 2019.

[20] T. Gebru *et al.*, "Datasheets for datasets," *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021.

[21] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2017.

[22] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, pp. 413–422, 2008.

[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.

[24] M. T. Nygard, *Release It! Design and Deploy Production-Ready Software*. Pragmatic Bookshelf, 2007.