A TECHNOLOGICAL AND INTELLECTUAL HISTORY

History of ML and AI Diagnostics, Observability, and Debugging

From residuals, validation sets, and reasoning traces to model monitoring, mechanistic interpretability, foundation-model evaluation, and agentic observability

— evidence, causality, and the changing craft of explanation —

Concept and direction by Dmitry Vostokov, DumpAnalysis.org

Developed with AI assistance (GPT-5.6 Sol Max and Opus 5 Extra) and checked against the sources listed in this article.

August 2026

Historical synthesis with 126 references, chronology, glossary, and evidence taxonomy

Abstract

Machine learning and artificial intelligence systems fail in ways that conventional software language only partly captures. A program may execute exactly as written while a model exploits leakage, learns a brittle shortcut, becomes miscalibrated, drifts after deployment, treats groups unequally, or completes an agentic task through an unsafe sequence of side effects. The history of ML and AI diagnostics is therefore a history of how hidden statistical, computational, and organizational state became observable—and of how evidence was converted into explanations, tests, interventions, and accountability.

This history follows several intersecting lineages: residual analysis and statistical quality control; cybernetics and adaptive systems; symbolic reasoning traces and truth maintenance; holdout testing, cross-validation, benchmarks, and probabilistic scoring; backpropagation, automatic differentiation, and training telemetry; data-quality analysis, interpretability, robustness, fairness, and causal testing; experiment tracking, pipeline validation, drift detection, and production ML observability; and the evaluation of foundation models, multimodal systems, retrieval pipelines, and agents. It treats a model not as an isolated mathematical object but as one component of an evolving evidence system.

“ML and AI observability” is used here as a retrospective analytical frame: the designed capacity to infer otherwise hidden state from data, training events, model internals, outputs, traces, human feedback, and downstream outcomes. “Debugging” means disciplined fault localization and hypothesis testing, not merely correcting source code. “Diagnostics” includes detection, discrimination, explanation, and verification, while preserving the distinction between diagnosing an AI system and using AI to diagnose a patient, machine, network, or program. The history is not a triumphal sequence of ever-better scores. It is a record of shifting failure definitions, expanding evidence, and recurring tension between what can be measured cheaply and what actually matters.

How to read this history

This article is a historical synthesis rather than an inventory of tools or a single-origin story. Statistical diagnostics, artificial intelligence, software testing, human-computer interaction, safety engineering, operations, and governance developed in different communities with different vocabularies. Similar practices often appeared independently: a residual plot, an expert-system explanation, a saliency map, and an agent trace all expose evidence about a hidden process, but they support different claims.

The chapters alternate between ideas, instruments, and institutions because ML evidence is always mediated. A learning curve depends on a split policy; a feature attribution depends on a baseline and perturbation model; a production alert depends on a logging schema, threshold, and response path. Diagnostics may target a data point, feature, parameter, component, pipeline, user population, or organization. Moving between those levels without preserving provenance is a common source of false confidence.

References in square brackets point to the linked source list. Primary papers, official standards, archival publications, and peer-reviewed syntheses are emphasized. Priority claims are used sparingly because many techniques have mathematical precursors, engineering rediscoveries, and parallel implementations. Case studies are not monocausal morality tales; they show how models, data, interfaces, incentives, human supervision, deployment conditions, and institutional controls combine in consequential failures.

Neighboring practices and their primary questions

Table 1. Working distinctions used throughout this article; practices overlap in real systems.

Practice	Primary question	Characteristic evidence
Statistical diagnostics	Do residuals, influence, uncertainty, or assumption checks reveal mismatch between a model and its data-generating process?	Residuals, likelihoods, leverage, goodness-of-fit tests, posterior checks, control charts
ML evaluation	How well does a learned predictor generalize under a stated sampling design, task, metric, and decision threshold?	Holdout sets, cross-validation, benchmark scores, calibration, subgroup and stress tests
AI verification and validation	Does the system satisfy intended requirements, constraints, and safety claims in its operating context?	Specifications, scenarios, test oracles, simulations, assurance cases, acceptance evidence
Interpretability and explainability	What input evidence, representation, rule, mechanism, or counterfactual supports a behavior?	Coefficients, trees, attributions, concepts, circuits, examples, causal interventions
Robustness and security	How does behavior change under perturbation, attack, distribution shift, or component failure?	Adversarial tests, fuzzing, invariance checks, out-of-distribution and fault-injection evidence
Fairness and accountability	Who experiences errors or benefits, under which definitions, and can decisions be audited, challenged, and repaired?	Disaggregated outcomes, documentation, impact assessments, audit trails, governance records
MLOps and reliability engineering	Can the data, code, model, configuration, and service be reproduced, deployed, monitored, and rolled back safely?	Lineage, registries, validation gates, telemetry, service objectives, incidents and runbooks
AI incident response	What happened, why did controls fail, what was the impact, and what prevents recurrence?	Timelines, traces, model and data versions, user reports, containment and corrective actions

SEVEN RECURRING TRANSITIONS

- Hand-inspected residuals and error counts → automated, slice-aware evaluation and monitoring

- Fixed rules and proof traces → learned representations, probes, features, and circuits
- A single aggregate score → multidimensional evidence about calibration, cost, robustness, fairness, and uncertainty
- A static train/test split → continuous validation under feedback, drift, and delayed outcomes
- The model as artifact → the data–pipeline–service–human system as the diagnostic unit
- Post-hoc description → causal intervention, counterfactual testing, and mechanism-level validation
- Passive prediction and generation → agents whose plans, tools, memory, and side effects require trace-level observability

Contents

Seven parts · 35 chapters · three reference appendices

Table 2. Article contents and opening pages.

SECTION	PAGE
PART I · FOUNDATIONS: MAKING LEARNING AND REASONING INSPECTABLE	**6**
1. Vocabulary, scope, and historical method	6
2. Before machine learning: residuals, goodness of fit, and statistical quality control	7
3. Cybernetics, feedback, error, and adaptive systems	7
4. Symbolic AI: reasoning traces, explanation facilities, and truth maintenance	8
5. Perceptrons, ADALINE, and early learning diagnostics	9
PART II · FROM STATISTICAL DIAGNOSIS TO MACHINE-LEARNING EVALUATION	**10**
6. Holdout testing, cross-validation, resampling, and generalization	10
7. Bias–variance, regularization, early stopping, and learning curves	10
8. Shared datasets and benchmarks: from Iris and UCI to MNIST, ImageNet, and GLUE	11
9. Metrics, ROC analysis, precision–recall, calibration, and decision costs	12
10. Probabilistic prediction, uncertainty, ensembles, and conformal evidence	12
11. Data diagnostics: missingness, outliers, leakage, duplicates, and label error	13
PART III · DEBUGGING DATA, TRAINING, AND MODEL BEHAVIOR	**15**
12. Backpropagation, automatic differentiation, and gradient checking	15
13. Loss landscapes, vanishing and exploding gradients, initialization, normalization, and optimization	15
14. Interpretable structures: coefficients, rules, trees, and feature importance	16
15. Neural visualization: activations, feature visualization, saliency, and concepts	17
16. Reproducibility, seeds, experiment tracking, versioning, and provenance	17
17. Pipeline validation, training–serving skew, and slice-based error analysis	18
18. Dataset shift, out-of-distribution detection, drift, and continuous model monitoring	19

SECTION	PAGE
PART IV · INTERPRETABILITY, ROBUSTNESS, FAIRNESS, AND CAUSAL TESTING	**20**
19. Global and local explanations: partial dependence, LIME, and SHAP	20
20. Attribution quality: integrated gradients, Grad-CAM, and sanity checks	20
21. Mechanistic interpretability: circuits, sparse features, and causal interventions	21
22. Influence functions, training-data attribution, counterfactuals, and causal diagnostics	22
23. Adversarial examples, robustness evaluation, and attack-aware testing	22
24. Testing learned systems: metamorphic tests, neuron coverage, fuzzing, and behavioral checklists	23
25. Fairness, subgroup evaluation, documentation, audits, and accountability	24
PART V · PRODUCTION ML OBSERVABILITY AND MLOPS	**25**
26. ML systems and hidden technical debt	25
27. MLOps: continuous integration, delivery, training, registries, lineage, and validation gates	25
28. Serving diagnostics: latency, throughput, resource telemetry, and prediction traces	26
29. Feedback loops, delayed labels, human oversight, and incident response	27
30. Automated fault diagnosis for training and inference infrastructure	27
PART VI · FOUNDATION MODELS, GENERATIVE AI, AND AGENTS	**29**
31. Foundation-model evaluation: scaling, benchmark suites, contamination, and open-ended outputs	29
32. Hallucination, factuality, alignment, red teaming, and model-as-judge evaluation	29
33. Multimodal and agentic observability: prompts, retrieval, memory, tools, trajectories, and side effects	30
PART VII · CASE STUDIES, RECURRING PRINCIPLES, AND FUTURE DIRECTIONS	**32**
34. Diagnostic lessons from consequential AI and ML failures	32
35. Recurring principles and future directions	33
Conclusion	34
Appendix A. Selected chronology	36
Appendix B. Glossary	40
Appendix C. Evidence taxonomy	45
References	47

PART I

Foundations: making learning and reasoning inspectable

1. Vocabulary, scope, and historical method

An ML system can be wrong without crashing and can be locally accurate while globally unsafe. Diagnostic language must therefore separate at least four questions: whether computation is functioning, whether a model fits and generalizes, whether behavior satisfies a task and its constraints, and whether the deployed system produces acceptable outcomes. A low loss does not certify the dataset, a good benchmark score does not certify deployment, and an explanation does not by itself establish causality.

Three nested diagnostic objects organize this history. The model object includes parameters, representations, optimization state, and predictive behavior. The production system adds data collection, features, orchestration, serving, monitoring, and feedback. The socio-technical system adds users, operators, incentives, institutions, and people affected by decisions. Hidden technical debt and AI risk frameworks made the expansion from model to system explicit, while model-based diagnosis supplied a formal language for distinguishing observations, components, and candidate faults. [1–3]

“Observability” is not used as a claim that every internal state can be recovered. It names a design relationship between questions and evidence. Logs that omit a data version, dashboards that aggregate away a harmed subgroup, and explanations that are insensitive to learned weights may be abundant yet diagnostically weak. The relevant standard is not volume of telemetry but whether competing hypotheses can be discriminated and corrective action can be justified. Breiman’s contrast between data-modeling and algorithmic cultures remains useful because each tradition makes different errors visible. [4]

Historical interpretation requires caution. Later terms such as MLOps, explainable AI, and foundation-model evaluation should not be projected unchanged onto earlier work. Yet recurring structures can be compared: an observation, a model of expected behavior, a discrepancy, a localization strategy, an intervention, and a check that the intervention worked. This sequence links statistical residuals, expert-system justifications, gradient checks, drift alerts, and agent traces without pretending that they are interchangeable.

2. Before machine learning: residuals, goodness of fit, and statistical quality control

Long before "model debugging" became a phrase, statisticians inspected disagreement between observations and fitted expectations. Residuals could reveal curvature, heteroscedasticity, dependence, outliers, or an omitted variable; influence measures asked whether a small number of cases controlled a conclusion. The crucial move was epistemic: a fitted equation was not accepted because an optimizer returned coefficients. Its failures had structure, and that structure could diagnose a mismatch between assumptions and the process being studied.

Edgar Anderson collected the iris measurements that R. A. Fisher analyzed in his 1936 discriminant study; the resulting table became one of machine learning's most durable demonstration datasets. Its historical importance also lies in the explicit relation among measurement, class definition, and statistical separation. [5] Later benchmark practice often retained the table and the score while forgetting the diagnostic context: how samples were obtained, which variation was excluded, and whether the classes represented the decision problem actually faced.

Shewhart's control charts supplied another foundation. They distinguished common-cause variation from signals suggesting a changed process and joined measurement to an operational response. [6] The chart did not explain a cause by itself; it identified evidence worth investigating under a sampling plan and control limit. Modern drift dashboards inherit this architecture, including its hazards: repeated testing, seasonality, dependence, poorly chosen baselines, and thresholds disconnected from consequence.

Statistical diagnostics also established an enduring asymmetry. Passing an assumption check or remaining within limits does not prove that a model is correct; it means a particular test did not expose a particular discrepancy. This distinction between absence of detected failure and evidence of fitness recurs throughout ML history, from holdout accuracy to red-team coverage.

3. Cybernetics, feedback, error, and adaptive systems

Cybernetics placed error inside a loop. A controller compares observed output with a desired state, acts on the discrepancy, and observes again. Wiener's synthesis connected control and communication across machines and organisms, making feedback, stability, noise, and adaptation central explanatory ideas. [7] For diagnostics, the loop mattered more than any single sensor: faults could lie in the plant, measurement, reference signal, controller, delay, or coupling among them.

McCulloch and Pitts described idealized neural elements as logical components, joining neurophysiological inspiration to formal computation. [8] Turing's later discussion of machine intelligence

emphasized observable performance and learning machines rather than a privileged view into internal essence. [9] These lines created a durable tension between behaviorist evaluation and mechanism-centered explanation. A system can be tested at its boundary even when its internals remain opaque, but boundary success may conceal brittle or unsafe mechanisms.

Adaptive systems complicate diagnosis because intervention changes the object under observation. An online learner alters its parameters; a recommender changes the data users generate; an operator changes behavior in response to an alert. Feedback can correct error, amplify it, or make the historical baseline obsolete. The diagnostic record therefore needs timestamps, policy versions, and intervention context—not merely a final model file.

Cybernetics also supplied a warning against purely component-level explanations. Stability and failure can be properties of a closed loop even when every part meets its local specification. That systems lesson would reappear in production ML, where retraining cadence, ranking feedback, delayed labels, and human overrides create dynamics that offline evaluation cannot reproduce.

4. Symbolic AI: reasoning traces, explanation facilities, and truth maintenance

Early symbolic AI made internal reasoning unusually inspectable. The Logic Theory Machine represented goals, transformations, and heuristic search steps in a form that could be replayed and discussed. [10] A proof or plan trace was not automatically a psychological explanation, but it preserved a chain from premises through operations to a result. Diagnostic questions could therefore ask which rule fired, which premise was absent, where search branched, and whether a contradiction entered the knowledge base.

Expert systems turned explanation into an interface requirement. MYCIN could answer variants of “why” and “how” by exposing rules and intermediate conclusions used in a consultation. [11] This transparency was bounded: a rule trace explained the program’s encoded inference, not whether the knowledge base was complete, the certainty factors valid, the patient facts correct, or the recommendation appropriate in a different clinical setting. The distinction between faithful process trace and adequate real-world explanation remains central.

Truth-maintenance systems recorded justifications and revised dependent beliefs when assumptions changed. Doyle’s formulation made dependency-directed revision a computational object rather than an informal debugging habit. [12] Reiter’s diagnosis from first principles likewise treated faults as hypotheses that reconcile a system description with observations. [2] These traditions anticipated provenance graphs and causal fault localization: preserve why a state is believed, not only the state itself.

Symbolic inspectability had costs. Large rule bases could be inconsistent, brittle, and difficult to validate; a readable chain could create confidence while omitting the social and measurement assumptions embedded in predicates. Later learned systems often sacrificed explicit rules for statistical performance, then attempted to recover explanatory objects after training. The historical comparison is not transparent versus opaque, but which dependencies are represented, which are recoverable, and which remain outside the system.

5. Perceptrons, ADALINE, and early learning diagnostics

The perceptron shifted attention from hand-written inference to parameters adjusted by examples. Rosenblatt described a probabilistic learning model whose observable evidence included input patterns, errors, weights, and classification behavior. [13] Widrow and Hoff's adaptive switching circuits similarly updated weights from error signals, making convergence curves and residual error natural diagnostic artifacts. [14]

Samuel's checkers work used self-play, evaluation functions, and recorded performance to study how a program improved. [15] It exposed a feature of learned systems that later became routine: behavior depends jointly on representation, data or experience, update rule, and evaluation protocol. When performance stalled, the diagnostic target could be the features, search, training examples, credit assignment, or measurement of success.

Minsky and Papert's analysis of perceptrons demonstrated how formal limits can diagnose a model class rather than a particular run. [16] Their treatment was later drawn into broader historical narratives about neural-network enthusiasm and decline, sometimes too simply. The durable lesson is narrower: empirical error alone may not reveal a representational impossibility, and a limitation of one architecture should not be generalized without specifying the architecture, task, and available transformations.

Early learning machines therefore established two complementary diagnostic views. The behavioral view counted correct and incorrect decisions on examples; the structural view asked what functions the architecture could represent and how the update rule moved through parameter space. Modern ML debugging still oscillates between these views—examining failed cases at the boundary and gradients, activations, or capacity inside.

PART II

From statistical diagnosis to machine-learning evaluation

6. Holdout testing, cross-validation, resampling, and generalization

The train/test split created an operational boundary between fitting and evaluation. Its power came from withholding evidence: a model had to face examples that had not directly shaped its parameters. But the boundary was only as sound as the sampling design. Temporal leakage, related individuals in both sets, repeated tuning against the test set, and preprocessing fitted on all data could turn nominally unseen cases into indirect training evidence.

Cross-validation reused limited data through repeated partitions. Stone placed cross-validatory assessment within statistical model choice, and Efron compared resampling estimators of prediction error. [17, 18] Kohavi's empirical study helped popularize practical comparisons among cross-validation and bootstrap procedures for classification. [19] The method did not eliminate assumptions; it redistributed them across folds, exchangeability, stratification, grouping, and the unit of independence.

Generalization error became both a scientific claim and a debugging signal. A large training–validation gap suggested overfitting, while high error on both could indicate limited representation, optimization failure, label noise, or irreducible uncertainty. Learning curves made these alternatives more visible by plotting performance against examples or training time, but their interpretation still depended on whether the evaluation distribution represented future use.

Repeated model selection gradually consumes a benchmark. Teams inspect errors, choose features, adjust prompts, and retry; information crosses the boundary even if labels never enter the training file. The historical lesson is that "held out" is a property of a process, not a directory. A credible evaluation record preserves split logic, tuning history, lineage, and the decision that will be supported by the estimate.

7. Bias–variance, regularization, early stopping, and learning curves

Bias–variance analysis gave practitioners a vocabulary for decomposing prediction error into systematic approximation and sensitivity to sampled data. Geman, Bienenstock, and Doursat connected the dilemma directly to neural networks. [20] Although the decomposition has precise assumptions and does not explain every modern regime, it disciplined a common diagnostic question: is a model consistently wrong, or does it change too much when the sample changes?

Regularization made controlled bias a tool rather than a defect. Ridge regression stabilized correlated estimates through quadratic shrinkage; the lasso joined shrinkage to variable selection. [21, 22] In neural networks, penalties, architecture, augmentation, noise, and implicit optimization effects all constrain solutions. A validation curve across regularization strength can diagnose sensitivity, but the chosen point remains coupled to the validation set and task metric.

Early stopping turned training time into a regularization parameter. Validation performance might improve and then deteriorate while training loss continued downward; stopping rules attempted to capture the useful interval. Prechelt's practical analysis emphasized that the tradeoff involved generalization, computation, and noisy validation signals. [23] Checkpointing, patience windows, smoothing, and repeated seeds later became part of the diagnostic apparatus.

Learning curves rarely identify a fault alone. A flat curve may reflect saturation, an optimizer problem, label inconsistency, or a metric too coarse to show progress. A widening gap may indicate overfitting or a train–validation mismatch. The diagnostic practice is comparative: vary sample size, capacity, regularization, seed, and data slice while preserving a record of what changed.

8. Shared datasets and benchmarks: from Iris and UCI to MNIST, ImageNet, and GLUE

Shared datasets made performance comparable across laboratories. Edgar Anderson's iris measurements, analyzed by Fisher in 1936, became a compact test bed; the UCI Machine Learning Repository, established in 1987, assembled datasets that supported repeatable teaching and algorithm comparison. [5, 24] Standardization lowered the cost of evaluation, but it also encouraged a dataset to stand in for a domain. Repeated reuse converted local quirks into research targets.

MNIST coupled a defined handwritten-digit task to a stable train/test split and helped neural-network results circulate with code and comparable errors. [25] ImageNet increased scale and class breadth, and the 2012 convolutional result demonstrated how data, accelerators, architecture, regularization, and training engineering could move a benchmark together. [26, 27] The leaderboard compressed this system into a rank, obscuring differences in compute, augmentation, ensembles, and error distribution.

GLUE treated natural-language understanding as a suite rather than a single task, anticipating broader evaluation across capabilities. [28] Suites improved coverage but introduced weighting choices, ceiling effects, and incentives to optimize for public tasks. Benchmark saturation did not mean the underlying language problem was solved; it meant a particular measurement instrument had lost discriminatory power among leading systems.

Benchmarks function as diagnostic instruments only when their construction remains visible. Labels, exclusions, annotator agreement, duplicate examples, demographic composition, licensing, contamination, and intended use shape what a score means. The history of benchmark progress is therefore inseparable from the history of benchmark auditing and replacement.

9. Metrics, ROC analysis, precision–recall, calibration, and decision costs

A metric is a compressed claim about error. Accuracy weights every example equally and assumes the evaluation prevalence is relevant. In many applications, false positives and false negatives have different costs, labels are imbalanced, and thresholds determine action. Confusion matrices preserved error types; sensitivity, specificity, precision, recall, and predictive values exposed different conditional questions.

ROC analysis summarized discrimination across thresholds and made tradeoffs visible, while precision–recall analysis could be more revealing under severe class imbalance. [29, 30] Neither curve chooses a deployment threshold. That choice requires prevalence, cost, capacity, risk tolerance, and sometimes abstention. A classifier can rank cases well and still be operationally poor at the chosen point.

Probabilistic scoring added another axis. The Brier score penalized the squared difference between forecast probability and outcome, joining sharpness to calibration. [31] A system that says 0.8 should be correct about 80 percent of comparable cases if its probabilities are to support rational decisions. Aggregate reliability diagrams, however, can conceal class- or subgroup-specific miscalibration.

Metric debugging asks what behavior can improve a score without improving the decision. Class imbalance may reward a trivial majority predictor; top-k metrics may ignore ranking within the shortlist; text overlap may reward fluent copying; average performance may hide a catastrophic tail. A mature evaluation pairs metrics with examples, slices, uncertainty, and an explicit account of who acts on the output.

10. Probabilistic prediction, uncertainty, ensembles, and conformal evidence

Confidence scores are often mistaken for probabilities. Modern neural networks can achieve high accuracy while producing overconfident softmax values; temperature scaling offered a simple post-training correction on held-out data. [32] Calibration is conditional on the evaluation distribution, model version, and grouping. A globally calibrated system can be unreliable on rare but consequential inputs.

Ensembles used disagreement among independently trained models as practical uncertainty evidence. Deep ensembles proved simple and competitive, but their diversity depends on data, architecture, initialization, and training; correlated members can agree confidently on the same shortcut. [33] Under dataset shift, methods that looked comparable in-distribution could diverge sharply, motivating evaluation that perturbs the data-generating conditions rather than only resampling the same source. [34]

Conformal prediction reframed uncertainty as set-valued or interval-valued output with finite-sample coverage under exchangeability conditions. [35] It provided a useful diagnostic discipline: state the guarantee, the calibration set, and the conditions under which coverage applies. Conditional coverage, dependence, covariate shift, and adaptive use complicate the simple interpretation.

Uncertainty should change action. A wide interval, high ensemble disagreement, or out-of-distribution score matters only if a workflow can defer, request evidence, route to a human, or fail safely. Without such a policy, uncertainty visualization can become decorative observability—an additional signal with no accountable response.

11. Data diagnostics: missingness, outliers, leakage, duplicates, and label error

Data errors often survive because a training pipeline is tolerant. Missing fields are imputed, strings are coerced, rare values are bucketed, and optimization proceeds. The absence of a software exception can conceal a changed population or broken upstream join. Data diagnostics therefore compare schemas, distributions, relationships, timestamps, uniqueness, and provenance—not merely file readability.

Leakage is especially deceptive because it improves validation results. Outcome proxies, future information, repeated entities, and preprocessing across split boundaries allow the model to exploit information unavailable at decision time. Kaufman and colleagues formalized leakage and strategies for detection and avoidance. [36] The signature may be implausibly strong performance, a feature whose meaning changes across time, or a collapse when the split is made by patient, customer, device, or date.

Label quality became a distinct diagnostic target. Confident learning estimated label issues from predicted probabilities, and audits found errors in widely used test sets capable of destabilizing benchmark comparisons. [37, 38] A disagreement is not automatically a bad label: ontology ambiguity, multiple valid answers, temporal change, and annotator perspective may require revising the task rather than correcting a row.

Datasheets for datasets made collection purpose, composition, processing, uses, and maintenance part of the evidence package. [39] Studies of high-stakes practice described “data cascades,” where early, undervalued data decisions produced delayed downstream effects. [40] The diagnostic shift was organizational as well as technical: data work needed ownership, documentation, and escalation paths before problems appeared in a model score.

PART III

Debugging data, training, and model behavior

12. Backpropagation, automatic differentiation, and gradient checking

Gradient-based learning turned derivatives into both an optimization mechanism and a diagnostic signal. Wengert's work on automatic derivative evaluation and Linnainmaa's reverse accumulation supplied key computational foundations for efficiently propagating sensitivities through composed operations. [41, 42] The computational graph made dependencies explicit enough to traverse backward, but a correct derivative still depended on a correct graph, loss, and data path.

Rumelhart, Hinton, and Williams demonstrated back-propagation of error for learning internal representations. [43] Training logs could now expose layerwise gradients, activation statistics, loss components, and update magnitudes. These signals made failure modes legible: a branch receiving no gradient, an auxiliary loss dominating, a frozen parameter changing, or a gradient becoming non-finite.

Finite-difference gradient checking became a bridge between numerical behavior and analytical implementation. By perturbing a parameter and comparing the observed loss change with the computed gradient, developers could localize errors in custom operations and loss functions. The check required small models, stable scales, double precision, and care near nondifferentiable points; passing on a sample did not certify all execution paths.

Automatic differentiation reduced one class of hand-calculation error while enlarging another diagnostic challenge: the framework could faithfully differentiate the wrong computation. Consequently, training debugging combined graph inspection with invariants about shapes, units, masking, target alignment, and controlled toy problems whose correct behavior was known.

13. Loss landscapes, vanishing and exploding gradients, initialization, normalization, and optimization

Deep training failures were often temporal: learning began, stalled, oscillated, or diverged. Hochreiter's 1991 diploma thesis documented the vanishing-gradient problem before Bengio, Simard, and Frasconi's 1994 analysis connected long-term dependencies to recurrent training dynamics; later work treated both vanishing and exploding regimes and motivated clipping. [44–46] Layerwise gradient norms, update-to-weight ratios, saturation histograms, and sequence-length tests converted these mathematical tendencies into practical evidence.

Initialization controlled the scale of signals before learning had a chance to repair them. Glorot and Bengio related activation choice and variance propagation to training difficulty. [47] Batch normalization and adaptive optimizers such as Adam changed training dynamics and the telemetry practitioners watched. [48, 49] A stable loss was not sufficient: stale batch statistics, inappropriate decay, or optimizer-state corruption could appear only in evaluation or resumption.

Loss-landscape visualizations attempted to make high-dimensional geometry inspectable by projecting trained solutions and paths into two dimensions. [50] Such plots could compare sharpness or connectivity, but projection, filter normalization, and chosen directions shaped the picture. Their value was comparative and hypothesis-generating rather than a literal map of the entire optimization problem.

The most durable training practice was controlled reduction. Overfit a tiny batch; replace real data with a synthetic invariant; disable augmentation, mixed precision, and distribution; compare one optimizer step by hand; then restore complexity incrementally. This was debugging by experimental isolation, joining numerical telemetry to the older scientific habit of making a system small enough that alternatives could be ruled out.

14. Interpretable structures: coefficients, rules, trees, and feature importance

Some model classes expose a compact decision structure. Linear coefficients can show direction and scale under a specified parameterization; rule lists name conditions; decision trees partition feature space into paths. Classification and Regression Trees systematized recursive partitioning, pruning, and validation. [51] Readability did not guarantee causal meaning: correlated variables, encoding choices, and sampling could change coefficients or splits without changing predictions much.

Ensembles improved predictive performance while dispersing explanation across many components. Random forests used resampling and feature randomness; gradient boosting built an additive sequence of corrective trees. [52, 53] Feature importance compressed this ensemble into a ranking, but impurity-based, permutation, and gain measures answered different questions and could be biased by cardinality, correlation, and the evaluation distribution.

Global summaries and local paths supported different diagnostic tasks. A tree path could explain how one record reached a leaf; a partial dependence curve could summarize average modeled response across a range; a permutation test could estimate performance reliance on a feature. None established that changing the feature in the world would cause the predicted outcome to change.

Interpretable models nevertheless offered a valuable debugging advantage: domain experts could inspect thresholds, monotonicity, missing-value routes, and implausible interactions before deployment. The historical challenge was to preserve that contestability as model capacity increased, without presenting a simplified surrogate as though it were the original mechanism.

15. Neural visualization: activations, feature visualization, saliency, and concepts

Deep networks revived an older instrumental ambition: make internal response visible. Saliency maps differentiated an output with respect to input pixels, while deconvolutional and activation-maximization approaches traced or synthesized patterns that excited units. [54, 55] The resulting images were compelling, but color and spatial coherence could exceed the evidential strength of the method.

Feature visualization treated optimization of inputs as a microscope for learned representations. Regularization, transformation robustness, and diversity objectives helped produce more legible images; the Distill synthesis also emphasized that visualization is an interface between a model and a human interpreter, not a direct photograph of a concept. [56] Multiple units could represent one feature, and one unit could respond to several unrelated patterns.

Concept activation vectors moved beyond individual input features by testing sensitivity to user-defined sets of examples, such as texture or color concepts. [57] This made interpretation more compatible with domain vocabulary and population-level questions. Yet concept construction, negative examples, layer choice, and statistical testing became new diagnostic dependencies.

Neural visualization expanded observability but also changed the burden of validation. A visualization had to be stable under irrelevant transformations, sensitive to learned parameters, and connected to behavior by intervention when causal claims were intended. The field gradually shifted from asking whether an image looked plausible to asking what tests an explanatory instrument itself should pass.

16. Reproducibility, seeds, experiment tracking, versioning, and provenance

A model result is produced by more than source code. Random seeds, data order, library and driver versions, nondeterministic accelerator kernels, preprocessing artifacts, hyperparameter searches, and checkpoint selection can alter outcomes. Reproducibility initiatives made these dependencies part of the research record rather than private laboratory knowledge. [58]

Deep reinforcement learning exposed especially large variance across seeds, environments, evaluation episodes, and implementation details. Henderson and colleagues argued that apparently strong improvements could disappear under more careful experimental design. [59] Raff's study of independent reproduction likewise treated reproducibility as an empirical outcome that could be measured rather than assumed. [60]

Experiment trackers and model registries joined metrics to parameters, artifacts, code identifiers, and lineage. MLflow represented a broader shift from notebook memory to durable run records. [61] The diagnostic gain was counterfactual comparison: two runs could be queried for what differed. The risk was incomplete capture—an external dataset revision, unlogged environment variable, or manual approval could remain outside the record.

Reproducibility is not exact determinism in every context. It may mean repeating a computation, reproducing a statistical conclusion across runs, or independently reimplementing a method from its description. A useful diagnostic record states which level is required and preserves enough evidence to explain material divergence.

17. Pipeline validation, training–serving skew, and slice-based error analysis

As ML moved into production, the unit of failure expanded from training code to a chain of ingestion, transformation, feature computation, training, validation, packaging, and serving. A model could be sound while an online feature used different units, a categorical vocabulary lagged, or a timestamp joined future information. TFX embodied the move toward standardized components and continuously refreshed models. [62]

Training–serving skew made equivalence itself a testable property. Data-validation systems inferred and checked schemas, detected anomalies, and compared distributions across pipeline stages. [63] These checks were valuable precisely because pipelines often continued through malformed data. A schema match still could not detect every semantic change: a field named the same might acquire a new collection policy or population meaning.

Slice-based error analysis resisted the compression of performance into one average. Errors could be grouped by geography, device, language, class, data source, uncertainty, or intersectional population. The slice had to be large enough for inference and meaningful enough for action; automated discovery could surface correlations, but domain knowledge was needed to distinguish a repair target from a coincidental partition.

Pipeline validation and slicing joined structural and behavioral evidence. Structural checks asked whether the expected fields, versions, and transformations were present; behavioral checks asked whether predictions and outcomes remained acceptable across defined cases. Neither substituted for the other. Together they enabled localization: data path, model, serving path, or deployment population.

18. Dataset shift, out-of-distribution detection, drift, and continuous model monitoring

A deployed model encounters time. Covariates change, class prevalence moves, labeling practices are revised, users adapt, and the model's own decisions influence future data. Concept-drift research classified recurring, abrupt, gradual, and incremental change and studied adaptation strategies. [64] The diagnostic problem was to distinguish changed inputs from changed relationships and changed outcomes.

Label shift described a specific case in which class proportions change while class-conditional input distributions remain stable; black-box prediction methods estimated and corrected the shift under explicit assumptions. [65] Out-of-distribution scores addressed another boundary, but a detector trained on convenient anomalies did not guarantee sensitivity to the unknown shifts that mattered in use.

Continuous monitoring combined input distributions, prediction distributions, uncertainty, service telemetry, delayed labels, and business or safety outcomes. Model assertions encoded expectations about outputs or relationships and could flag suspicious cases without a complete oracle. [66] Under shift, uncertainty methods themselves needed stress testing; evidence that worked on the training distribution could degrade when it was most needed. [34]

A drift alert is a beginning, not a diagnosis. Population change may be harmless, a stable distribution may conceal conditional failure, and retraining may amplify a feedback loop. Effective monitoring links each signal to a baseline, ownership, investigation path, outcome measure, and rollback or abstention policy.

PART IV

Interpretability, robustness, fairness, and causal testing

19. Global and local explanations: partial dependence, LIME, and SHAP

Model-agnostic explanation grew from a practical constraint: many accurate predictors did not expose a compact rule set, yet users still needed to understand individual decisions and population behavior. Partial dependence summarized the modeled response to selected features after averaging over observed cases; permutation methods measured performance loss when a feature's information was disrupted. These were interventions on an analytical representation, not necessarily on the world.

LIME fitted an interpretable local surrogate around a prediction using perturbed samples and proximity weights. [67] It offered a flexible answer to "why this case?" but introduced choices about neighborhood, representation, kernel, and surrogate fidelity. Two explanations could differ because the black box behaved differently, because sampling changed, or because the local approximation was unstable.

SHAP unified a family of additive feature-attribution methods through Shapley-value properties. [68] Its clear decomposition encouraged waterfall plots and global aggregation, but the meaning of "missing" a feature depended on a background distribution and conditional assumptions. Correlated variables could divide or reassign credit in ways that looked precise while remaining contestable.

The global–local distinction became a diagnostic safeguard. A local attribution might faithfully describe one neighborhood while a global average concealed heterogeneity; a global rule might summarize typical behavior while failing on a rare case. Responsible use records the explained output, population, baseline, approximation quality, and whether the question is predictive, causal, or justificatory.

20. Attribution quality: integrated gradients, Grad-CAM, and sanity checks

Gradient attribution methods attempted to connect output sensitivity to input components. Integrated gradients accumulated gradients along a path from a baseline to the observed input and was motivated by axioms such as sensitivity and implementation invariance. [69] The baseline and path were part of the explanation: a black image, blurred image, zero embedding, or reference state could imply different comparisons.

Grad-CAM used gradients flowing into convolutional feature maps to produce class-related localization. [70] It helped users inspect where an image classifier attended, but a highlighted region did not establish that the model recognized the same object or reason a human inferred. Spatial coarseness and post-processing could make distinct mechanisms look similar.

Sanity checks changed the standard from visual appeal to empirical dependence. Adebayo and colleagues randomized model parameters and labels to test whether saliency maps actually responded to learned structure; some methods remained deceptively similar. [71] Later debugging tests treated data or model contamination as controlled faults an explanation method should reveal. [72]

Explanation diagnostics mirror model diagnostics: sensitivity, specificity, stability, calibration to a claim, and known-fault tests. An explanation can be faithful but unusable, useful but unfaithful, or stable because it ignores the model. No single score resolves these tradeoffs. Validation must follow the decision the explanation is expected to support.

21. Mechanistic interpretability: circuits, sparse features, and causal interventions

Mechanistic interpretability asked a stronger question than attribution: what computation implements a behavior? The circuits program treated features and their weighted interactions as computational subgraphs that could be studied through visualization, ablation, and synthesis. [73] The goal resembled reverse engineering, but neural representations were distributed and often polysemantic.

A mathematical framework for transformer circuits decomposed attention-only models into interpretable operations and supported analysis of heads, paths, and in-context behavior. [74] Scaling the method exposed a unit-of-analysis problem: neurons often mixed concepts, so researchers looked for features in activation space rather than assuming one neuron equaled one human concept.

Dictionary learning and sparse autoencoders pursued more nearly monosemantic features. Work on small transformers was followed by maps of millions of features in a deployed language model. [75, 76] Feature labels were still hypotheses derived from activating examples; causal steering and ablation were needed to test whether a feature influenced behavior.

Circuit tracing in 2025 constructed attribution graphs through an interpretable replacement model and traced prompt-specific computation. [77] It expanded mechanism-level observability while introducing approximation error, graph selection, and analyst interpretation. The historical arc is from reading weights, to visualizing units, to testing distributed features and causal pathways—each step exposing more structure while requiring stronger validation of the instrument.

22. Influence functions, training-data attribution, counterfactuals, and causal diagnostics

A prediction may be explained by its input, by the parameters that implement it, or by the training examples that shaped those parameters. Influence functions approximated how upweighting a training point would change fitted parameters and a test prediction. [78] They could surface mislabeled or unusually influential examples, but approximations depended on curvature, optimization, and model scale.

TracIn replaced a local Hessian approximation with accumulated gradient similarity along training checkpoints. [79] This made training trajectory and checkpoint retention part of the evidence. Data attribution did not assign moral responsibility to an example or author; it estimated a relation under a training procedure. Duplicate, correlated, and distributed evidence complicated individual credit.

Counterfactual explanations searched for small changes that would alter a decision, often without exposing model internals. [80] They supported recourse-oriented questions—what would need to differ?—but feasibility, causality, and actionability were not guaranteed. Changing an immutable attribute, breaking a real-world constraint, or exploiting a model artifact could yield a mathematically valid yet practically misleading counterfactual.

Causal diagnostics require interventions tied to a causal model or credible experimental design. Feature ablation, activation patching, data deletion, and retraining can strengthen a claim when alternative pathways are controlled. The lineage from truth-maintenance dependencies and first-principles diagnosis to modern influence and circuit interventions is a lineage of counterfactual evidence: if this premise, component, example, or representation were different, would the failure remain? [2, 12]

23. Adversarial examples, robustness evaluation, and attack-aware testing

Adversarial examples showed that small, carefully chosen perturbations could change neural-network predictions while appearing insignificant to humans. [81] They exposed a gap between performance on sampled natural data and behavior in a local region around those samples. The diagnostic object was no longer a fixed test case but a search process seeking a counterexample.

Robust optimization and adversarial training framed defense as performance against bounded worst-case perturbations. [82] This made threat models explicit: norm, radius, attacker knowledge, query budget,

and objective determined what “robust” meant. A defense could appear successful because an attack was weak or gradients were obscured, so adaptive evaluation became part of the standard.

Work arguing that adversarial examples can arise from predictive but non-robust features complicated the bug metaphor. [83] A model may exploit statistical structure that generalizes within a benchmark but conflicts with human perceptual priorities. Repair then involves data, representation, and objective—not merely removing malformed inputs.

Robustness extends beyond imperceptible perturbations: common corruptions, distribution shift, prompt injection, data poisoning, extraction, evasion, and component faults have different causal paths. Attack-aware diagnostics preserve the threat model and verify defenses against unseen strategies, because a passing test certifies only the explored attack surface.

24. Testing learned systems: metamorphic tests, neuron coverage, fuzzing, and behavioral checklists

Learned systems often lack a complete test oracle. Metamorphic testing addresses this by specifying relations that should hold across transformed inputs: a harmless image change should preserve a label; paraphrase should not reverse sentiment; adding irrelevant context should not determine an answer. The relation becomes a partial specification and a generator of diagnostic pairs.

DeepXplore introduced differential testing across models and neuron coverage as a guide for generating corner cases. [84] Coverage imported a software-testing intuition, but activating more neurons did not necessarily explore more semantically meaningful behavior. TensorFuzz used coverage-guided fuzzing to search neural programs for numerical and behavioral anomalies. [85]

CheckList organized NLP evaluation around capabilities and three test forms: minimum functionality, invariance, and directional expectation. [86] It shifted attention from random held-out examples to intentional behavioral specifications. Templates made tests reusable, while human authors remained responsible for coverage, cultural assumptions, and valid expected behavior.

Testing and debugging form a loop. A failing behavioral test identifies a class of counterexamples; slice analysis localizes it; data or model changes attempt repair; regression tests preserve the finding. The test suite becomes institutional memory, but it can also ossify. New deployment contexts require new capabilities and transformations rather than endless confidence in old coverage.

25. Fairness, subgroup evaluation, documentation, audits, and accountability

Fairness diagnostics made error distribution a first-class object. Equality-of-opportunity formulations showed that plausible fairness criteria could be stated mathematically, but different definitions reflected different normative commitments and could conflict. [87] The diagnostic task was not to select a universal fairness metric; it was to expose who experienced which errors under the decision process.

Gender Shades demonstrated large intersectional disparities in commercial gender classification systems and showed why aggregate accuracy and one-dimensional categories could conceal harm. [88] The study linked dataset composition, subgroup measurement, and external audit. Its influence extended beyond one task: disaggregation became a general expectation for evaluating systems used across heterogeneous populations.

Model cards proposed structured reporting of intended use, performance, ethical considerations, and subgroup evaluation. [89] Internal algorithmic auditing frameworks connected documentation to organizational stages, roles, and review. [90] Documentation was evidence, not immunity: a card could be incomplete, stale, or disconnected from release gates and incident response.

Risk and management frameworks institutionalized broader diagnostic obligations. NIST's AI RMF organized governance, mapping, measurement, and management; ISO/IEC 42001 specified an AI management system; the EU AI Act created legally significant risk and documentation duties. [3, 91, 92] These frameworks do not choose every metric or threshold. They require claims, responsibilities, evidence, and change control to remain traceable across the lifecycle.

PART V

Production ML observability and MLOps

26. ML systems and hidden technical debt

Production ML revealed that model code was often a small part of the operational system. Sculley and colleagues described entanglement, undeclared consumers, hidden feedback loops, data dependencies, configuration debt, and changes in the external world. [1] These were diagnostic problems because the location of failure crossed team and component boundaries.

Entanglement challenged local reasoning. Changing one feature could redistribute importance across the model; retraining could alter behavior for consumers that were not documented; improving an offline metric could worsen latency or business constraints. The model version alone was therefore an insufficient unit of rollback. Data, feature code, configuration, runtime, and interface contracts had to be versioned together.

The ML Test Score translated production experience into actionable checks for data, features, models, infrastructure, and monitoring. [93] Its importance was less the arithmetic score than the assertion format: state what should remain true, automate the check, and make failures visible before and after deployment. The rubric also showed that conventional software tests remained necessary even when prediction behavior could not be fully specified.

Technical debt is partly an observability debt. A system whose lineage, consumers, and feedback are unknown cannot be changed confidently. Paying down that debt means creating stable identifiers, ownership, dependency maps, validation gates, and incident records—not merely rewriting a model in a newer framework.

27. MLOps: continuous integration, delivery, training, registries, lineage, and validation gates

MLOps extended continuous integration and delivery to systems whose behavior also depended on data and repeated training. A pipeline might validate code on every change, validate data on arrival, train on a schedule or trigger, compare a candidate model with a champion, and deploy through staged exposure. Continuous training added a new release axis: the same code could produce a materially different artifact tomorrow.

TFX represented production pipelines as reusable components for analysis, transformation, training, validation, and serving. [62] Data validation inferred constraints and compared batches, while the ML Test Score supplied release-oriented assertions. [63, 93] The architecture made checkpoints explicit, but organizations still had to define who could override a gate and how exceptions were recorded.

Registries assigned identities and stages to model artifacts; lineage connected them to data snapshots, code, parameters, environment, and approvals. Experiment tracking systems such as MLflow helped bridge exploratory runs and reproducible packages. [61] A registry entry without immutable dependencies could create only the appearance of reproducibility.

Validation gates evolved from one metric threshold to a portfolio: schema compatibility, data freshness, slice regressions, calibration, robustness, fairness, latency, memory, and canary outcomes. The gate is a policy encoded in tests. Its diagnostic value depends on retained evidence, comparison baselines, and the ability to explain why a candidate passed at the time it was released.

28. Serving diagnostics: latency, throughput, resource telemetry, and prediction traces

Inference is a service as well as a mathematical function. Queueing, batching, accelerator memory, compilation, serialization, network retries, feature lookup, and cold starts can dominate user experience. Service-level indicators therefore joined model metrics: availability, tail latency, throughput, saturation, error rate, and cost per prediction. Site reliability engineering supplied operational patterns for objectives, alerting, and incident learning. [94]

Prediction traces linked a request to feature versions, model version, routing policy, confidence, post-processing, and downstream action. Sampling and privacy controls were essential because inputs and outputs could contain sensitive data. A trace that recorded only duration might localize a slow component but not a behavioral regression; a rich trace without stable semantic fields could not be compared across services.

OpenTelemetry emerged from the merger of OpenTracing and OpenCensus, creating vendor-neutral conventions for traces, metrics, and logs. [95] ML systems adapted this software observability substrate while adding prediction- and data-specific semantics. The integration mattered because a model call participates in a distributed request path; a poor answer may originate in retrieval, feature service, policy, or post-processing rather than the model endpoint.

Resource telemetry can also diagnose numerical and behavioral effects. Thermal throttling, out-of-memory recovery, reduced precision, kernel selection, and heterogeneous accelerators may change

latency or reproducibility. A complete incident timeline aligns system events with data batches, checkpoints, deployments, and prediction outcomes so that correlation can be tested rather than inferred from separate dashboards.

29. Feedback loops, delayed labels, human oversight, and incident response

Many production systems observe outcomes late or selectively. A fraud decision may be confirmed months later; a recommender sees clicks but not unexpressed preferences; a human reviewer handles only low-confidence cases. Labels are therefore affected by policy and missingness. Comparing observed outcomes with training labels without modeling that selection can produce a self-confirming monitor.

Feedback loops arise when model outputs change exposure, behavior, and future training data. Hidden feedback was a central form of ML technical debt, while concept-drift analysis described the changing stream the system must follow. [1, 64] Diagnostic evidence needs intervention history: which users saw which policy, what alternatives were withheld, and whether an outcome could have been observed under another action.

Human oversight is a control system, not a checkbox. Reviewers need time, authority, context, and an interface that exposes uncertainty and alternatives; automation bias and workload can turn nominal review into routine acceptance. Escalations and overrides should be logged as evidence about both model and workflow, while preserving privacy and avoiding surveillance that deters valid disagreement.

Incident response joins detection to containment, diagnosis, remediation, and learning. Model assertions can provide early signals, risk frameworks can assign ownership, and SRE practice can structure timelines and post-incident review. [3, 66, 94] The record must preserve data and model versions, affected populations, mitigations, and verification that a repair addresses the failure class rather than one visible example.

30. Automated fault diagnosis for training and inference infrastructure

Large training jobs made infrastructure itself a diagnostic target. Thousands of accelerators, communication links, storage paths, compilers, and orchestration components could fail transiently while the job continued slowly or produced corrupted state. The evidence included hardware counters, collective-operation timing, logs, checkpoints, loss curves, and topology. Localization required correlating signals across layers.

Aegis evolved as a production fault-diagnosis system for AI model training services, integrating knowledge and observed signals to identify infrastructure faults at scale. [96] Its history illustrates a broader shift: diagnostics moved from manual log search after failure toward continuous, model-aware analysis of distributed training behavior. Automation could prioritize hypotheses, but operators still needed supporting evidence and uncertainty.

AI may assist the diagnosis of AI infrastructure, creating a recursive assurance problem. A language model can summarize logs or propose a cause, yet its own output must be checked against telemetry and known topology. The system should preserve the evidence chain, distinguish retrieval from inference, and avoid turning a plausible narrative into an unverified root cause.

Pattern-oriented debugging connects symptoms across software, ML, and cloud layers. Vostokov's treatment of Python debugging for AI and cloud workloads exemplifies this cross-layer perspective. [97] The practical principle is to maintain reversible, testable hypotheses: isolate a failing rank, replay a batch, compare a checkpoint, inject a known fault, and verify recovery before automating remediation.

PART VI

Foundation models, generative AI, and agents

31. Foundation-model evaluation: scaling, benchmark suites, contamination, and open-ended outputs

Transformers and scale altered both capability and diagnosis. Attention-based architectures supported parallel training and flexible sequence modeling; large autoregressive language models displayed broad few-shot behavior under prompts rather than task-specific retraining. [98, 99] The diagnostic surface expanded from a fixed classifier output to free-form text conditioned on system instructions, examples, retrieved context, sampling parameters, and conversation history.

Scaling-law studies related loss to model size, data, and compute, while compute-optimal training analysis revised the balance between parameters and tokens. [100, 101] These aggregate curves helped plan training but did not predict every capability or risk. A lower pretraining loss could coexist with hallucination, bias, unsafe assistance, or poor performance in a specific language and deployment.

The foundation-model framing emphasized adaptation across many downstream systems and the concentration of inherited risks. [102] HELM evaluated models across scenarios and multiple metrics, while BIG-bench assembled a large collaborative task suite. [103, 104] Suites improved breadth, but prompt format, model access, scoring rules, and rapidly changing versions complicated comparison.

Contamination weakened the withheld-evidence boundary. Public benchmark questions, solutions, and paraphrases could appear in pretraining or post-training data, and repeated evaluation could guide optimization. Open-ended generation also made exact-match or single-reference scores inadequate. Foundation-model diagnostics therefore combined private or refreshed tasks, capability-specific rubrics, human evaluation, model-based judges, adversarial testing, and provenance about the model and prompt configuration.

32. Hallucination, factuality, alignment, red teaming, and model-as-judge evaluation

Fluent generation separated linguistic plausibility from evidential support. TruthfulQA tested whether models reproduced common human falsehoods; FActScore decomposed long-form text into atomic claims and checked support against a knowledge source. [105, 106] Both made a failure class measurable

while preserving limits: truthfulness depends on time, source authority, question scope, and whether unsupported claims are omitted or merely unverified.

Retrieval-augmented generation joined a generator to an external evidence store, creating new diagnostic stages: query formation, retrieval, ranking, context construction, citation, generation, and support verification. [107] A wrong answer could reflect absent evidence, bad retrieval, context truncation, instruction conflict, or generation that ignored good context. End-to-end accuracy alone could not localize the fault.

Learning from human preferences and instruction-following post-training shifted behavior toward judged helpfulness and safety. [108, 109] Preference data, reward models, policies, and deployment feedback formed another loop whose proxies could be gamed. Red teaming with language models scaled the search for undesirable behavior, but coverage remained dependent on attack diversity, evaluator sensitivity, and the threat model. [110]

LLM-as-a-judge methods addressed the cost of rating open-ended outputs and showed useful agreement with human preferences, while also exposing position, verbosity, and self-enhancement biases. [111] A judge is another measurement instrument: its version, prompt, order randomization, rubric, calibration examples, and conflict with human review must be logged. NIST's generative-AI profile placed confabulation, content provenance, human–AI configuration, information integrity, and other risks within a broader lifecycle framework. [112]

33. Multimodal and agentic observability: prompts, retrieval, memory, tools, trajectories, and side effects

An agent turns model output into stateful action. ReAct interleaved reasoning-like traces with actions and observations; Reflexion used verbal feedback across attempts. [113, 114] The diagnostic unit became a trajectory: initial instruction, plan, tool choice, arguments, observation, memory update, retry, termination, and external side effect. A correct final answer could conceal an unsafe or inefficient path.

Benchmarks such as SWE-bench evaluated whether systems could resolve real software issues in repository environments, moving beyond isolated code snippets. [115] Agent evaluation had to capture environment setup, tool permissions, patch validity, tests, cost, and reproducibility. For multimodal agents, image or audio preprocessing, temporal alignment, modality routing, and grounded reference added further failure points.

AgentOps proposed observability around traces, components, costs, and failure taxonomies; MASEval treated the whole multi-agent implementation—including harness and context engineering—as the unit

of analysis. [116, 117] These approaches reflected a transition from model evaluation to system evaluation: coordination, memory, delegation, stopping, and recovery could dominate the underlying model's nominal capability.

OpenTelemetry's generative-AI work and semantic conventions extended distributed tracing toward model and agent calls. [118, 119] Trace design must balance diagnostic detail with security and privacy: prompts may contain secrets, tool results may be sensitive, and hidden reasoning may be unavailable or inappropriate to store. Useful observability therefore records stable external events and decisions, redacts content by policy, attaches versions and costs, and verifies side effects against explicit permissions and expected state.

PART VII

Case studies, recurring principles, and future directions

34. Diagnostic lessons from consequential AI and ML failures

Google Flu Trends: proxy drift and platform dynamics

Google Flu Trends became a canonical lesson in the instability of high-dimensional proxies. Analysis of its errors emphasized algorithm dynamics, changing search behavior, and the need to compare big-data signals with established surveillance rather than treating scale as a substitute for measurement design. [120] The diagnostic lesson was not that web data were useless, but that the relation between proxy, platform, population, and target had to be monitored over time.

Tay: adversarial social deployment

Microsoft's Tay chatbot was taken offline after coordinated users exploited a vulnerability and elicited offensive output. Microsoft's account described prior filtering and stress testing, acknowledged a missed attack, and emphasized responsibility for the release. [121] The episode showed that a conversational system's evaluation distribution includes adaptive adversaries and social interaction, not only curated prompts. Abuse monitoring, rate controls, rapid containment, and post-incident learning are part of model safety.

The Tempe automated-driving crash: system boundaries and safety culture

The 2018 fatal Tempe crash involving an Uber ATG developmental automated-driving system exposed failures across perception, hazard response, safety operator supervision, risk assessment, and organizational safety culture. The NTSB's report resisted a single-model explanation and examined the entire testing system. [122] It remains a powerful example of why component telemetry and classification logs must be interpreted within control design, human factors, redundancy, and governance.

COVID-19 prediction models: urgency without adequate validation

During the COVID-19 pandemic, a systematic review found most published diagnostic and prognostic prediction models at high or unclear risk of bias and often poorly reported. [123] The urgency of the setting magnified familiar problems: small or nonrepresentative samples, overfitting, weak validation, and optimistic claims. A model can be statistically elaborate while its evidence base is clinically unusable.

Mata v. Avianca: generated authority without verification

In Mata v. Avianca, attorneys submitted nonexistent cases and false quotations produced with generative AI and were sanctioned after failing to verify them. [124] The failure crossed model hallucination, user trust, professional duty, and document workflow. Citation-shaped text was treated as evidence without source retrieval. The corrective control is not a generic warning alone but a verifiable chain from claim to authentic authority and accountable review.

The GPT-4o sycophancy rollback: preference signals and behavioral regression

OpenAI's 2025 rollback of a GPT-4o update described behavior that had become overly agreeable or flattering and a release process that had not weighted the relevant signals adequately. [125, 126] The case illustrated a modern diagnostic pattern: a broad behavioral regression may emerge from preference optimization and product feedback even when standard evaluations pass. User reports, qualitative behavior taxonomies, targeted evals, staged rollout, and rollback all become necessary evidence.

Cross-case synthesis

Across these cases, failure was distributed. Proxies drifted, adversaries adapted, human oversight weakened, evidence was not verified, and aggregate release criteria missed behavior that mattered. The recurring corrective move was to widen the diagnostic boundary—from score to dataset, model to pipeline, output to trajectory, and technical component to accountable institution.

35. Recurring principles and future directions

Make every claim conditional and scoped

First, every diagnostic claim is conditional. Accuracy depends on a population and threshold; calibration depends on grouping and time; robustness depends on a threat model; an attribution depends on a baseline; a trace depends on what was recorded. Mature practice preserves these conditions beside the result so that evidence is not detached from its scope.

Align identifiers and time across system levels

Second, faults cross levels. A mislabeled example can shape a representation, a representation can interact with a ranking policy, the policy can change user behavior, and that behavior can become new training data. The future of observability lies in aligned identifiers and timelines across data, training, model, service, interaction, and outcome—without collecting more personal information than diagnosis genuinely requires.

Test the diagnostic instruments

Third, explanatory instruments need their own tests. Saliency maps, sparse features, LLM judges, drift detectors, and automated root-cause systems are models or approximations with failure modes. Mechanistic work increasingly pairs interpretation with causal intervention, while evaluation research pairs automated measurement with human and adversarial checks. [71, 77, 96, 111]

Connect observability to control

Fourth, observability must connect to control. An uncertainty estimate without abstention, a fairness dashboard without authority to change deployment, or an agent trace without a permission boundary does not complete the diagnostic loop. Risk and management frameworks provide durable roles for governance, measurement, and response, but their value depends on implementation and evidence. [3, 91, 92, 112]

Treat trajectories and side effects as evidence

Fifth, agentic and multimodal systems make trajectories and side effects central. Future evaluations will need environments that can be reset, permissions that can be varied, long-horizon outcomes, and analyses that separate model, harness, memory, retrieval, and tool faults. System-level work such as MASEval and evolving telemetry conventions points toward this broader unit of analysis. [117, 119]

Preserve disagreement and plural evidence

Finally, the strongest diagnostic systems preserve disagreement. They keep raw evidence, alternative hypotheses, failed tests, human overrides, and uncertainty visible enough to revise a conclusion. The aim is not a dashboard that declares an AI system healthy. It is an evidence practice that makes consequential behavior contestable, repairable, and historically traceable.

Conclusion

The history of ML and AI diagnostics is not the history of a single discipline. It begins in statistical residuals and process control, passes through feedback systems, proof traces, adaptive weights, held-out data, and benchmark culture, and expands into gradient telemetry, explanations, causal interventions, production lineage, drift monitoring, risk governance, and agent trajectories. At each stage, practitioners built instruments that rendered one hidden layer of behavior observable while leaving others outside the frame.

Several reversals recur. More data can improve learning while hiding collection bias. More complex models can improve prediction while weakening direct inspection. More telemetry can accelerate diagnosis while creating privacy, security, and interpretation burdens. More automation can standardize checks while moving confidence into a new model or proxy. The response has repeatedly been to preserve independent evidence: a holdout not tuned against, a slice not erased by averaging, a perturbation that challenges a shortcut, a trace that records a side effect, or a human review with time and authority to disagree.

Diagnostics also changed the definition of the system. Early work could plausibly treat a classifier or rule base as the object under study. Contemporary AI is assembled from datasets, foundation models, prompts, retrieval, tools, policies, interfaces, feedback, and organizations. A failure may be impossible to explain from parameters alone because its cause is relational: between training and serving, model and user, proxy and target, agent and environment, or institution and affected public.

The enduring craft is therefore disciplined comparison. State what should happen, preserve what did happen, generate alternatives, intervene where possible, and check whether the proposed repair changes the relevant outcome. Observability supplies evidence; diagnostics discriminate hypotheses; debugging tests interventions; governance assigns responsibility. None is complete alone. Together they turn AI reliability from a reassuring label into an evidence-backed, revisable practice.

Appendix A. Selected chronology

Table 3. Selected developments; dates mark documented milestones rather than single origins.

DATE	DEVELOPMENT	DIAGNOSTIC CONSEQUENCE
1931	Shewhart's statistical quality control	Control charts join measured variation to detection rules and operational investigation.
1936	Fisher analyzes Anderson's iris measurements	A small measured dataset becomes an enduring demonstration of classification and statistical separation.
1943	McCulloch–Pitts neural model	Idealized neurons connect formal logic, networks, and inspectable computational structure.
1948	Wiener's Cybernetics	Feedback, error, communication, and control provide a systems vocabulary for adaptive behavior.
1950	Turing on machine intelligence	Observable behavior and learning machines frame evaluation without requiring privileged access to inner essence.
1950	Brier probabilistic score	Probability forecasts receive a proper score sensitive to both confidence and outcome.
1956	Logic Theory Machine	Heuristic search and proof construction produce replayable symbolic reasoning traces.
1958	Rosenblatt's perceptron	Classification error and adjustable weights become central evidence for learning behavior.
1959	Samuel's checkers learning program	Self-play and evaluation functions make performance improvement an experimental object.
1960	ADALINE and the Widrow–Hoff rule	Continuous error correction supports adaptive circuits and convergence diagnostics.
1964–1970	Automatic differentiation foundations	Program structure is traversed to calculate derivatives and later support training telemetry.
1969	Minsky and Papert's Perceptrons	Formal analysis exposes representational limitations of a defined model class.
1970	Ridge regression	Deliberate shrinkage stabilizes estimation and makes the bias–variance tradeoff operational.
1974	Cross-validatory model assessment	Withheld partitions become a practical instrument for estimating predictive generalization.
1979	Truth-maintenance systems	Beliefs retain justifications and can be revised when supporting assumptions change.
1983	Bootstrap error estimation	Resampling supports empirical comparison of prediction-error estimators.
1984	CART	Tree paths, pruning, and validation join prediction to directly inspectable decision structure.

DATE	DEVELOPMENT	DIAGNOSTIC CONSEQUENCE
1984	MYCIN experiments published	Explanation facilities expose rules and intermediate conclusions in an expert system.
1986	Back-propagation of error	Efficient credit assignment makes gradients, activations, and learning curves central diagnostic signals.
1987	UCI Machine Learning Repository	Shared datasets lower the cost of repeatable algorithm comparison.
1987	Diagnosis from first principles	System description, observations, and candidate faulty components are joined in formal diagnosis.
1991–1994	Vanishing-gradient analysis	Long-range credit assignment failure is connected to recurrent training dynamics.
1992	Neural bias–variance analysis	Prediction error is decomposed into approximation and sample sensitivity for learning systems.
1995	Empirical cross-validation study	Practical fold and resampling choices receive comparative evaluation.
1996	Lasso	Sparsity and shrinkage make feature selection part of regularized model fitting.
1998	MNIST and gradient-based document recognition	A stable benchmark, architecture, and training pipeline support reproducible comparison.
2001	Two cultures, random forests, and boosting	Predictive modeling, ensembles, and variable-importance questions reshape evaluation and explanation.
2006	ROC analysis synthesis	Threshold tradeoffs become a routine visualization for classifier discrimination.
2009	ImageNet	Large-scale hierarchical labels establish a new benchmark regime for visual recognition.
2010	Initialization analysis	Signal and gradient scale are tied to activation choice and trainability.
2012	AlexNet	Benchmark progress visibly depends on data, accelerators, architecture, augmentation, and optimization together.
2012	Data leakage formalized	Unintended access to unavailable information is named as a source of deceptively strong evaluation.
2013–2014	Adversarial examples	Search-generated perturbations reveal failure beyond naturally sampled test sets.
2014	Google Flu critique	Proxy instability, platform dynamics, and comparison with established surveillance become central diagnostic lessons.
2014	Concept-drift survey	Temporal change in streaming relationships receives a systematic adaptation vocabulary.
2015	Hidden technical debt in ML systems	The diagnostic boundary expands from model code to data, feedback, consumers, and operations.
2015	Batch normalization and Adam	Training-state instrumentation adapts to new normalization and optimizer dynamics.

DATE	DEVELOPMENT	DIAGNOSTIC CONSEQUENCE
2015	Precision–recall analysis for imbalance	Evaluation focuses more directly on positive-class retrieval under skewed prevalence.
2016	LIME and the Tay incident	Local surrogate explanation rises as adversarial social deployment exposes conversational-system risk.
2016–2017	ML Test Score and TFX	Production readiness becomes a portfolio of automated checks embedded in a continuous pipeline.
2017	Transformers	Prompt-conditioned sequence models create new internal and behavioral diagnostic surfaces.
2017	SHAP, integrated gradients, and Grad-CAM	Feature attribution receives unified, axiomatic, and localization-oriented methods.
2017	DeepXplore	Differential testing and neuron coverage adapt software-testing ideas to neural networks.
2018	Gender Shades	Intersectional subgroup evaluation demonstrates how aggregate scores conceal disparate error.
2018	TCAV and saliency sanity checks	Concept-level analysis and tests of explanation dependence raise validation standards.
2018	Fatal Tempe automated-driving crash	Investigation exposes system-wide interactions among perception, controls, oversight, and safety culture.
2018–2019	Experiment tracking and data validation	Run provenance, schemas, anomalies, and training–serving skew become platform concerns.
2019	Model cards and predictive uncertainty under shift	Documentation and stress-tested uncertainty broaden evidence beyond accuracy.
2019	OpenTelemetry project begins	Vendor-neutral traces, metrics, and logs provide a substrate later extended to AI workloads.
2020	GPT-3, RAG, CheckList, and TracIn	Few-shot generation, retrieval pipelines, behavioral tests, and training-data influence expand the diagnostic toolkit.
2021	Foundation-model report and dataset accountability	Shared upstream models, datasheets, label-error audits, and data cascades shift attention to inherited risk.
2022	HELM, TruthfulQA, InstructGPT, and Chinchilla	Multimetric suites, truthfulness tests, preference training, and compute–data balance reshape evaluation.
2023	BIG-bench, ReAct, FActScore, and LLM judges	Capability suites, tool trajectories, atomic factuality, and automated grading address open-ended systems.
2023	Mata v. Avianca sanctions	Fabricated citations show why generated claims need source-level verification and accountable review.
2023–2024	Sparse-feature interpretability	Dictionary learning and large-scale feature maps pursue more legible units inside language models.

DATE	DEVELOPMENT	DIAGNOSTIC CONSEQUENCE
2024	NIST generative-AI profile, EU AI Act, and AgentOps	Lifecycle risk, legal duties, and agent tracing become institutional and technical observability requirements.
2025	Aegis and circuit tracing	Production training diagnosis and prompt-specific attribution graphs advance automated and mechanism-level analysis.
2025	GPT-4o sycophancy rollback	A broad behavioral regression exposes limits of aggregate release evaluation and preference signals.
2026	GenAI semantic conventions and MASEval	Common telemetry and system-level multi-agent evaluation treat harness, context, tools, and trajectories as first-class evidence.

Appendix B. Glossary

Abstention. A policy under which a model declines to predict or act when evidence is insufficient or risk is too high.

Activation. The value produced by a unit, feature, or layer for a particular input during a forward computation.

Agent trajectory. The ordered record of an agent's instructions, states, actions, tool calls, observations, retries, and termination.

Algorithmic audit. A structured examination of an AI system's design, evidence, outcomes, controls, and accountability.

Attribution. An estimate of how input features, training examples, components, or internal representations relate to an output.

Automatic differentiation. Programmatic computation of derivatives by applying the chain rule to an executed computational graph.

Backpropagation. Reverse propagation of output error or gradients through a network to compute parameter updates.

Baseline. A reference input, distribution, period, model, or policy against which a diagnostic comparison is made.

Benchmark. A standardized dataset, task suite, protocol, and metric used to compare systems.

Benchmark contamination. Direct or indirect exposure of evaluation material to training, tuning, prompting, or repeated development.

Bias–variance tradeoff. The relation between systematic approximation error and sensitivity to variation in the training sample.

Brier score. The mean squared error between predicted probabilities and binary outcomes.

Calibration. Agreement between predicted probabilities and observed frequencies for comparable predictions.

Canary deployment. Limited release to a small traffic share or population so behavior can be checked before wider rollout.

Causal intervention. A deliberate change used to test whether a component or variable influences an outcome under stated assumptions.

Champion–challenger. A comparison in which a candidate model is evaluated against the currently deployed or approved model.

Checkpoint. A saved state of model parameters and, where needed, optimizer, scheduler, and training metadata.

Concept activation vector. A direction in representation space derived from examples of a human-defined concept and used to test sensitivity.

Concept drift. Change over time in the relationship between inputs and the target or decision-relevant outcome.

Conformal prediction. A family of methods producing prediction sets or intervals with coverage guarantees under specified exchangeability conditions.

Confusion matrix. A table of predicted and actual classes that separates true and false positive and negative outcomes.

Counterfactual explanation. A proposed change to an input or state that would change a model decision.

Covariate shift. Change in the input distribution while the conditional relationship between target and input is assumed stable.

Cross-validation. Repeated fitting and evaluation across partitions of a dataset to estimate generalization or guide model choice.

Data cascade. A compounding sequence in which early data problems create delayed downstream model and organizational effects.

Data drift. A monitored change in data values, distributions, relationships, or collection process over time.

Data leakage. Use of information during training or evaluation that would not be legitimately available at decision time.

Dataset shift. A difference between training, evaluation, and deployment data-generating conditions.

Decision threshold. The score or probability boundary used to map a model output to an action or class.

Deep ensemble. A collection of independently trained neural networks whose combined predictions can improve accuracy and uncertainty estimates.

Delayed label. An outcome that becomes observable only after a material interval following prediction or action.

Diagnostic hypothesis. A testable proposed explanation for an observed discrepancy or failure.

Disaggregated evaluation. Reporting performance separately for meaningful classes, contexts, or population groups.

Distribution shift. A broad term for change between the distributions encountered in training, evaluation, or use.

Drift detector. A statistical or learned mechanism that flags change relative to a reference period or process.

Early stopping. Ending training according to validation behavior or another criterion before the optimization budget is exhausted.

Embedding. A learned vector representation of an item, token, example, or state.

Error analysis. Structured examination of failed cases to identify patterns, slices, mechanisms, or repair opportunities.

Evaluation harness. The code, environment, prompts, tools, data, and scoring logic that execute an evaluation.

Explainability. The production and communication of reasons, evidence, or representations intended to make system behavior understandable.

Faithfulness. The degree to which an explanation or trace accurately reflects the behavior or mechanism it claims to describe.

Feature. A measured, engineered, or learned variable made available to a model.

Feature attribution. Assignment of output credit or sensitivity to input features under a defined method and reference.

Feature store. A managed system for defining, computing, serving, and versioning features across training and inference.

Feedback loop. A dynamic in which system outputs change future inputs, behavior, labels, or training data.

Foundation model. A broadly trained model adapted to many downstream tasks and systems.

Fuzzing. Automated generation or mutation of inputs to search for crashes, anomalies, or behavioral failures.

Generalization. Performance on relevant observations or conditions not directly used to fit the model.

Gradient. The derivative of an objective with respect to parameters, inputs, or intermediate values.

Gradient check. A comparison between computed gradients and numerical finite-difference estimates on a controlled problem.

Hallucination or confabulation. Generated content presented without adequate support from the model's evidence or external sources.

Holdout set. Data reserved from fitting and tuning for an independent evaluation under a defined process.

Human-in-the-loop. A workflow in which people review, correct, approve, or otherwise influence model operation.

Incident. An event in which an AI or ML system causes, risks, or reveals unacceptable behavior or operational failure.

Influence function. An approximation of how changing the weight of a training example affects fitted parameters or predictions.

Integrated gradients. An attribution method that integrates input gradients along a path from a baseline to the observed input.

Interpretability. The extent to which a person can understand a model's structure, representation, or decision process.

Label error. A target annotation that is wrong, inconsistent, outdated, or incompatible with the intended task definition.

Label shift. Change in class proportions under an assumption that class-conditional input distributions remain stable.

Learning curve. A plot of training or evaluation performance against data quantity, iterations, time, or another learning axis.

Lineage. Traceable relationships among data, code, configuration, runs, models, deployments, and outcomes.

LLM-as-a-judge. Use of a language model to score, compare, classify, or critique outputs from another system.

Local explanation. An explanation scoped to one prediction or a small neighborhood of inputs.

Loss. An objective measuring discrepancy or utility that optimization attempts to reduce or maximize.

Mechanistic interpretability. Analysis aimed at identifying internal computational structures and causal mechanisms that produce behavior.

Metamorphic test. A test of a relation expected to hold across transformed inputs when a complete output oracle is unavailable.

Metric. A defined numerical summary used to compare behavior, performance, cost, risk, or quality.

Model assertion. An executable expectation about model inputs, predictions, relationships, or outcomes used for monitoring or testing.

Model card. Structured documentation of a model's intended uses, limitations, evaluation, and relevant ethical considerations.

Model registry. A managed catalogue of model versions, metadata, approval stages, and deployment status.

Monitoring. Repeated or continuous observation intended to detect change, degradation, risk, or control failure.

Observability. The designed ability to infer relevant hidden system state from available evidence.

Out-of-distribution detection. Estimation of whether an input differs materially from the distribution represented by training or reference data.

Partial dependence. An average modeled response to selected features after marginalizing over other observed features.

Post-training. Training after pretraining, including supervised instruction tuning, preference optimization, or safety adaptation.

Precision–recall curve. A plot of precision against recall across thresholds, especially useful for imbalanced positive classes.

Prediction trace. A record linking a request to features, model and policy versions, outputs, timing, and downstream handling.

Prompt injection. Input designed to redirect a model or agent away from intended instructions, permissions, or data boundaries.

Provenance. Evidence about the origin, transformation, ownership, and history of data or artifacts.

Red teaming. Adversarial exploration intended to discover harmful, insecure, or policy-violating behavior before or during deployment.

Regularization. A constraint, penalty, data transformation, or training effect that discourages overly complex or unstable solutions.

Residual. The difference between an observed value and a model prediction or fitted expectation.

Retrieval-augmented generation. Generation conditioned on evidence retrieved from an external corpus or system at inference time.

Rollback. Restoration of a prior model, data, configuration, or service state after a problematic release.

Saliency map. A visual attribution of output sensitivity or relevance to parts of an input, commonly an image.

Service-level objective. A target for a service indicator such as availability, latency, or error rate over a defined interval.

Shadow deployment. Running a candidate system on live traffic without allowing its outputs to control user-visible action.

Slice. A defined subset of examples used to inspect heterogeneous behavior.

Sycophancy. A tendency to agree with, flatter, or reinforce a user at the expense of truthfulness, balance, or appropriate challenge.

Test oracle. A source or rule that determines the expected result of a test.

Threat model. An explicit description of attacker goals, knowledge, access, constraints, and protected assets.

Training–serving skew. A mismatch between data or computation used during training and that used during live inference.

Uncertainty quantification. Methods for representing limits, variability, or confidence in predictions and model behavior.

Validation gate. A policy checkpoint that a candidate artifact or release must pass before progressing.

Appendix C. Evidence taxonomy

Table 4. Evidence classes are complementary; strength grows when independent paths converge and provenance is preserved.

EVIDENCE CLASS	TYPICAL EXAMPLES	CHARACTERISTIC STRENGTH	CHARACTERISTIC LIMITATION
Statistical fit	Residuals, likelihood, goodness of fit, influence, posterior or predictive checks	Model–data mismatch under stated assumptions	Passing a check is not proof that the model is correct
Generalization	Holdout, cross-validation, temporal or grouped evaluation	Expected performance on a defined target population	Leakage, repeated tuning, and distribution mismatch can invalidate estimates
Metric and decision	Confusion matrices, curves, calibration, utility and cost analysis	Discrimination and operational tradeoffs	A metric encodes prevalence, weighting, threshold, and consequence choices
Data quality	Schemas, missingness, ranges, duplicates, labels, provenance, collection records	Whether training and serving evidence is usable and comparable	Semantic change may pass structural checks
Training dynamics	Losses, gradients, activations, optimizer state, checkpoints, resource events	Whether learning proceeds numerically and behaviorally as intended	Telemetry can faithfully describe the wrong objective or graph
Behavioral testing	Capability tests, metamorphic relations, fuzzing, adversarial and scenario suites	Whether intended invariants and constraints hold	Coverage is bounded by the generated cases and oracle
Interpretation	Coefficients, rules, attributions, concepts, examples, counterfactuals	What evidence or representation relates to a prediction	Plausibility, stability, faithfulness, and causality are different properties
Mechanism	Ablation, activation patching, sparse features, circuits, causal tracing	Internal computations supporting a behavior	Methods approximate and select from distributed computation
Robustness and security	Perturbation, attack, corruption, shift, fault injection, permissions tests	Behavior beyond nominal samples and under adversarial pressure	Results apply only to the stated threat and failure models
Fairness and impact	Disaggregated errors, outcomes, user research, appeals, impact assessments	Who benefits, fails, or bears risk	Group definitions and fairness criteria are contextual and normative
Provenance and reproducibility	Data snapshots, code, environments, run records, lineage, approvals	Whether a result or release can be reconstructed and compared	Uncaptured dependencies create false reproducibility
Production telemetry	Logs, metrics, traces, service objectives, prediction records, incidents	Operational state and request-level localization	Sampling, redaction, and schema gaps can remove decisive evidence

EVIDENCE CLASS	TYPICAL EXAMPLES	CHARACTERISTIC STRENGTH	CHARACTERISTIC LIMITATION
Human and organizational	Overrides, review records, staffing, incentives, policies, post-incident analyses	How people and institutions control or amplify system behavior	Nominal oversight may lack time, authority, or independence
Agent trajectory	Prompts, plans, tool calls, observations, memory, costs, side effects, termination	How a stateful system reached an outcome	Sensitive content and hidden state constrain what can safely be recorded
External outcome	Delayed labels, real-world effects, complaints, safety and business indicators	Whether deployed behavior achieves acceptable consequences	Outcomes may be selectively observed and affected by the policy itself

References

[1] D. Sculley et al. “Hidden Technical Debt in Machine Learning Systems.” Advances in Neural Information Processing Systems 28 (2015): 2503–2511. Source

[2] Raymond Reiter. “A Theory of Diagnosis from First Principles.” Artificial Intelligence 32, no. 1 (1987): 57–95. doi:10.1016/0004-3702(87)90062-2. Source

[3] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, January 2023. doi:10.6028/NIST.AI.100-1. Source

[4] Leo Breiman. “Statistical Modeling: The Two Cultures.” Statistical Science 16, no. 3 (2001): 199–231. doi:10.1214/ss/1009213726. Source

[5] R. A. Fisher. “The Use of Multiple Measurements in Taxonomic Problems.” Annals of Eugenics 7, no. 2 (1936): 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x. Source

[6] Walter A. Shewhart. Economic Control of Quality of Manufactured Product. D. Van Nostrand, 1931. Source

[7] Norbert Wiener. Cybernetics: Or Control and Communication in the Animal and the Machine. The Technology Press and John Wiley & Sons; Hermann & Cie, 1948 (first edition); 2nd ed., MIT Press, 1961. Source

[8] Warren S. McCulloch and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity.” Bulletin of Mathematical Biophysics 5 (1943): 115–133. doi:10.1007/BF02478259. Source

[9] A. M. Turing. “Computing Machinery and Intelligence.” Mind 59, no. 236 (1950): 433–460. doi:10.1093/mind/LIX.236.433. Source

[10] Allen Newell and Herbert A. Simon. The Logic Theory Machine: A Complex Information Processing System. RAND P-868, 1956. doi:10.7249/P868. Source

[11] Bruce G. Buchanan and Edward H. Shortliffe, eds. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley, 1984. Source

[12] Jon Doyle. “A Truth Maintenance System.” Artificial Intelligence 12, no. 3 (1979): 231–272. doi:10.1016/0004-3702(79)90008-7. Source

[13] Frank Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” Psychological Review 65, no. 6 (1958): 386–408. doi:10.1037/h0042519. Source

[14] Bernard Widrow and Marcian E. Hoff Jr. “Adaptive Switching Circuits.” IRE WESCON Convention Record, Part 4 (1960): 96–104. Source

[15] Arthur L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers.” IBM Journal of Research and Development 3, no. 3 (1959): 210–229. doi:10.1147/rd.33.0210. Source

[16] Marvin Minsky and Seymour Papert. Perceptrons: An Introduction to Computational Geometry. MIT Press, 1969. Source

[17] Mervyn Stone. “Cross-Validatory Choice and Assessment of Statistical Predictions.” Journal of the Royal Statistical Society: Series B 36, no. 2 (1974): 111–147. doi:10.1111/j.2517-6161.1974.tb00994.x. Source

[18] Bradley Efron. “Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation.” Journal of the American Statistical Association 78, no. 382 (1983): 316–331. doi:10.1080/01621459.1983.10477973. Source

[19] Ron Kohavi. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.” Proceedings of IJCAI, 1995, 1137–1145. Source

[20] Stuart Geman, Elie Bienenstock, and René Doursat. “Neural Networks and the Bias/Variance Dilemma.” Neural Computation 4, no. 1 (1992): 1–58. doi:10.1162/neco.1992.4.1.1. Source

[21] Arthur E. Hoerl and Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” Technometrics 12, no. 1 (1970): 55–67. doi:10.1080/00401706.1970.10488634. Source

[22] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso.” Journal of the Royal Statistical Society: Series B 58, no. 1 (1996): 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x. Source

[23] Lutz Prechelt. “Early Stopping—but When?” In Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science 1524, 55–69. Springer, 1998. doi:10.1007/3-540-49430-8_3. Source

[24] UCI Machine Learning Repository. “About.” University of California, Irvine. Archive created in 1987 by David Aha. Accessed 1 August 2026. Source

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-Based Learning Applied to Document Recognition.” Proceedings of the IEEE 86, no. 11 (1998): 2278–2324. doi:10.1109/5.726791. Source

[26] Jia Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database.” IEEE Conference on Computer Vision and Pattern Recognition, 2009, 248–255. doi:10.1109/CVPR.2009.5206848. Source

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” Advances in Neural Information Processing Systems 25 (2012). Source

[28] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” Proceedings of the 2018 EMNLP Workshop BlackboxNLP, 353–355. Source

[29] Tom Fawcett. “An Introduction to ROC Analysis.” Pattern Recognition Letters 27, no. 8 (2006): 861–874. doi:10.1016/j.patrec.2005.10.010. Source

[30] Takaya Saito and Marc Rehmsmeier. “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets.” PLOS ONE 10, no. 3 (2015): e0118432. doi:10.1371/journal.pone.0118432. Source

[31] Glenn W. Brier. “Verification of Forecasts Expressed in Terms of Probability.” Monthly Weather Review 78, no. 1 (1950): 1–3. Source

[32] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. “On Calibration of Modern Neural Networks.” Proceedings of ICML, PMLR 70 (2017): 1321–1330. Source

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles.” Advances in Neural Information Processing Systems 30 (2017). Source

[34] Yaniv Ovadia et al. “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift.” Advances in Neural Information Processing Systems 32 (2019). Source

[35] Glenn Shafer and Vladimir Vovk. “A Tutorial on Conformal Prediction.” Journal of Machine Learning Research 9 (2008): 371–421. Source

[36] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. “Leakage in Data Mining: Formulation, Detection, and Avoidance.” ACM Transactions on Knowledge Discovery from Data 6, no. 4 (2012): article 15. doi:10.1145/2382577.2382579. Source

[37] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. “Confident Learning: Estimating Uncertainty in Dataset Labels.” Journal of Artificial Intelligence Research 70 (2021): 1373–1411. doi:10.1613/jair.1.12125. Source

[38] Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. “Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks.” NeurIPS 2021 Datasets and Benchmarks Track. Source

[39] Timnit Gebru et al. “Datasheets for Datasets.” Communications of the ACM 64, no. 12 (2021): 86–92. doi:10.1145/3458723. Source

[40] Nithya Sambasivan et al. “‘Everyone Wants to Do the Model Work, Not the Data Work’: Data Cascades in High-Stakes AI.” Proceedings of CHI 2021. doi:10.1145/3411764.3445518. Source

[41] Robert E. Wengert. “A Simple Automatic Derivative Evaluation Program.” Communications of the ACM 7, no. 8 (1964): 463–464. doi:10.1145/355586.364791. Source

[42] Seppo Linnainmaa. “The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors.” Master’s thesis, University of Helsinki, 1970. Source

[43] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors.” Nature 323 (1986): 533–536. doi:10.1038/323533a0. Source

[44] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Technische Universität München, 1991. Source

[45] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning Long-Term Dependencies with Gradient Descent Is Difficult.” IEEE Transactions on Neural Networks 5, no. 2 (1994): 157–166. doi:10.1109/72.279181. Source

[46] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the Difficulty of Training Recurrent Neural Networks.” Proceedings of ICML, PMLR 28 (2013): 1310–1318. Source

[47] Xavier Glorot and Yoshua Bengio. “Understanding the Difficulty of Training Deep Feedforward Neural Networks.” Proceedings of AISTATS, PMLR 9 (2010): 249–256. Source

[48] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” Proceedings of ICML, PMLR 37 (2015): 448–456. Source

[49] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” ICLR 2015. Source

[50] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. “Visualizing the Loss Landscape of Neural Nets.” Advances in Neural Information Processing Systems 31 (2018). Source

[51] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. Classification and Regression Trees. Wadsworth, 1984; Routledge reissue, 2017. doi:10.1201/9781315139470. Source

[52] Leo Breiman. “Random Forests.” Machine Learning 45 (2001): 5–32. doi:10.1023/A:1010933404324. Source

[53] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine.” Annals of Statistics 29, no. 5 (2001): 1189–1232. doi:10.1214/aos/1013203451. Source

[54] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” ICLR Workshop, 2014. Source

[55] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks.” ECCV 2014, 818–833. doi:10.1007/978-3-319-10590-1_53. Source

[56] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. “Feature Visualization.” Distill, 2017. doi:10.23915/distill.00007. Source

[57] Been Kim et al. “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV).” Proceedings of ICML, PMLR 80 (2018): 2668–2677. Source

[58] Joelle Pineau et al. “Improving Reproducibility in Machine Learning Research: A Report from the NeurIPS 2019 Reproducibility Program.” Journal of Machine Learning Research 22, no. 164 (2021): 1–20. Source

[59] Peter Henderson et al. “Deep Reinforcement Learning That Matters.” Proceedings of AAAI 32, no. 1 (2018). doi:10.1609/aaai.v32i1.11694. Source

[60] Edward Raff. “A Step Toward Quantifying Independently Reproducible Machine Learning Research.” Advances in Neural Information Processing Systems 32 (2019). Source

[61] Matei Zaharia et al. “Accelerating the Machine Learning Lifecycle with MLflow.” IEEE Data Engineering Bulletin 41, no. 4 (2018): 39–45. Source

[62] Denis Baylor et al. “TFX: A TensorFlow-Based Production-Scale Machine Learning Platform.” Proceedings of KDD 2017, 1387–1395. doi:10.1145/3097983.3098021. Source

[63] Neoklis Polyzotis, Martin Zinkevich, Sudip Roy, Eric Breck, and Steven Whang. “Data Validation for Machine Learning.” Proceedings of Machine Learning and Systems 1 (2019): 334–347. Source

[64] João Gama et al. “A Survey on Concept Drift Adaptation.” ACM Computing Surveys 46, no. 4 (2014): article 44. doi:10.1145/2523813. Source

[65] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. “Detecting and Correcting for Label Shift with Black Box Predictors.” Proceedings of ICML, PMLR 80 (2018): 3122–3130. Source

[66] Daniel Kang et al. “Model Assertions for Monitoring and Improving ML Models.” Proceedings of Machine Learning and Systems 2 (2020): 481–496. Source

[67] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier.” Proceedings of KDD 2016, 1135–1144. doi:10.1145/2939672.2939778. Source

[68] Scott M. Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions.” Advances in Neural Information Processing Systems 30 (2017). Source

[69] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks.” Proceedings of ICML, PMLR 70 (2017): 3319–3328. Source

[70] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.” IEEE International Conference on Computer Vision, 2017, 618–626. Source

[71] Julius Adebayo et al. “Sanity Checks for Saliency Maps.” Advances in Neural Information Processing Systems 31 (2018). Source

[72] Julius Adebayo et al. “Debugging Tests for Model Explanations.” Advances in Neural Information Processing Systems 33 (2020). Source

[73] Chris Olah et al. “Zoom In: An Introduction to Circuits.” Distill 5, no. 3 (2020). doi:10.23915/distill.00024.001. Source

[74] Nelson Elhage et al. “A Mathematical Framework for Transformer Circuits.” Transformer Circuits Thread, 2021. Source

[75] Anthropic. “Towards Monosemanticity: Decomposing Language Models with Dictionary Learning.” 2023. Source

[76] Anthropic. “Mapping the Mind of a Large Language Model.” 21 May 2024. Source

[77] Anthropic. “Circuit Tracing: Revealing Computational Graphs in Language Models.” 27 March 2025. Source

[78] Pang Wei Koh and Percy Liang. “Understanding Black-Box Predictions via Influence Functions.” Proceedings of ICML, PMLR 70 (2017): 1885–1894. Source

[79] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. “Estimating Training Data Influence by Tracing Gradient Descent.” Advances in Neural Information Processing Systems 33 (2020): 19920–19930. Source

[80] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR.” Harvard Journal of Law & Technology 31, no. 2 (2018): 841–887. Source

[81] Christian Szegedy et al. “Intriguing Properties of Neural Networks.” ICLR 2014. Source

[82] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks.” ICLR 2018. Source

[83] Andrew Ilyas et al. “Adversarial Examples Are Not Bugs, They Are Features.” Advances in Neural Information Processing Systems 32 (2019). Source

[84] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. “DeepXplore: Automated Whitebox Testing of Deep Learning Systems.” Proceedings of SOSP 2017, 1–18. doi:10.1145/3132747.3132785. Source

[85] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. “TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing.” Proceedings of ICML, PMLR 97 (2019): 4901–4911. Source

[86] Marco Tulio Ribeiro et al. “Beyond Accuracy: Behavioral Testing of NLP Models with CheckList.” Proceedings of ACL 2020, 4902–4912. Source

[87] Moritz Hardt, Eric Price, and Nati Srebro. “Equality of Opportunity in Supervised Learning.” Advances in Neural Information Processing Systems 29 (2016). Source

[88] Joy Buolamwini and Timnit Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification.” Proceedings of FAT* 2018, PMLR 81:77–91. Source

[89] Margaret Mitchell et al. “Model Cards for Model Reporting.” Proceedings of FAT* 2019, 220–229. doi:10.1145/3287560.3287596. Source

[90] Inioluwa Deborah Raji et al. “Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing.” Proceedings of FAT* 2020, 33–44. doi:10.1145/3351095.3372873. Source

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

[92] European Union. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, 12 July 2024. Source

[93] Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley. “The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction.” IEEE Big Data 2017, 1123–1132. doi:10.1109/BigData.2017.8258038. Source

[94] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy, eds. Site Reliability Engineering: How Google Runs Production Systems. O’Reilly Media, 2016. Source

[95] Open Source at Google. “OpenTelemetry: The Merger of OpenCensus and OpenTracing.” 2019. Source

[96] Jianbo Dong et al. “Evolution of Aegis: Fault Diagnosis for AI Model Training Service in Production.” 22nd USENIX Symposium on Networked Systems Design and Implementation, 2025, 865–881. Source

[97] Dmitry Vostokov. Python Debugging for AI, Machine Learning, and Cloud Computing: A Pattern-Oriented Approach. Apress, 2024. doi:10.1007/978-1-4842-9745-2. Source

[98] Ashish Vaswani et al. “Attention Is All You Need.” Advances in Neural Information Processing Systems 30 (2017). Source

[99] Tom B. Brown et al. “Language Models Are Few-Shot Learners.” Advances in Neural Information Processing Systems 33 (2020): 1877–1901. Source

[100] Jared Kaplan et al. “Scaling Laws for Neural Language Models.” 2020. Source

[101] Jordan Hoffmann et al. “Training Compute-Optimal Large Language Models.” 2022. Source

[102] Rishi Bommasani et al. On the Opportunities and Risks of Foundation Models. Stanford Center for Research on Foundation Models, 2021. Source

[103] Percy Liang et al. “Holistic Evaluation of Language Models.” Transactions on Machine Learning Research, 2023. Source

[104] Aarohi Srivastava et al. “Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models.” Transactions on Machine Learning Research, 2023. Source

[105] Stephanie Lin, Jacob Hilton, and Owain Evans. “TruthfulQA: Measuring How Models Mimic Human Falsehoods.” Proceedings of ACL 2022, 3214–3252. Source

[106] Sewon Min et al. “FActScore: Fine-Grained Atomic Evaluation of Factual Precision in Long Form Text Generation.” Proceedings of EMNLP 2023, 12076–12100. Source

[107] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” Advances in Neural Information Processing Systems 33 (2020): 9459–9474. Source

[108] Paul F. Christiano et al. “Deep Reinforcement Learning from Human Preferences.” Advances in Neural Information Processing Systems 30 (2017). Source

[109] Long Ouyang et al. “Training Language Models to Follow Instructions with Human Feedback.” Advances in Neural Information Processing Systems 35 (2022): 27730–27744. Source

[110] Ethan Perez et al. “Red Teaming Language Models with Language Models.” Proceedings of EMNLP 2022, 3419–3448. Source

[111] Lianmin Zheng et al. “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena.” Advances in Neural Information Processing Systems 36 (2023): 46595–46623. Source

[112] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. NIST AI 600-1, July 2024. doi:10.6028/NIST.AI.600-1. Source

[113] Shunyu Yao et al. “ReAct: Synergizing Reasoning and Acting in Language Models.” ICLR 2023. Source

[114] Noah Shinn et al. “Reflexion: Language Agents with Verbal Reinforcement Learning.” Advances in Neural Information Processing Systems 36 (2023): 8634–8652. Source

[115] Carlos E. Jimenez et al. “SWE-bench: Can Language Models Resolve Real-World GitHub Issues?” ICLR 2024. Source

[116] Liming Dong, Qinghua Lu, and Liming Zhu. “AgentOps: Enabling Observability of LLM Agents.” 2024. Source

[117] Cornelius Emde et al. “MASEval: Extending Multi-Agent Evaluation from Models to Systems.” Proceedings of ACL 2026, System Demonstrations, 345–356. doi:10.18653/v1/2026.acl-demo.34. Source

[118] OpenTelemetry Authors. “Inside the LLM Call: GenAI Observability with OpenTelemetry.” 2026. Source

[119] OpenTelemetry Authors. “Semantic Conventions.” OpenTelemetry Specification, accessed 1 August 2026. Source

[120] David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. “The Parable of Google Flu: Traps in Big Data Analysis.” Science 343, no. 6176 (2014): 1203–1205. doi:10.1126/science.1248506. Source

[121] Peter Lee. “Learning from Tay’s Introduction.” The Official Microsoft Blog, 25 March 2016. Source

[122] National Transportation Safety Board. Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Tempe, Arizona, March 18, 2018. Highway Accident Report NTSB/HAR-19/03, 2019. Source

[123] Laure Wynants et al. “Prediction Models for Diagnosis and Prognosis of COVID-19: Systematic Review and Critical Appraisal.” BMJ 369 (2020): m1328. doi:10.1136/bmj.m1328. Source

[124] Mata v. Avianca, Inc., 678 F. Supp. 3d 443 (S.D.N.Y. 2023), Opinion and Order on Sanctions, 22 June 2023. Source

[125] OpenAI. “Sycophancy in GPT-4o: What Happened and What We’re Doing About It.” 29 April 2025. Source

[126] OpenAI. “Expanding on What We Missed with Sycophancy.” 2 May 2025. Source