

Light Water Reactor Sustainability Program

Evaluation of the feasibility of multimodal foundation artificial intelligence models for anomaly detection and development of the scientific basis



December 2025

U.S. Department of Energy

Office of Nuclear Energy

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

Evaluation of the feasibility of multimodal foundation artificial intelligence models for anomaly detection and development of the scientific basis

Mohammad G. Abdo (Principal Investigator)¹

Niharika Karnik¹

Linyu Lin¹

Congjian Wang¹

Tao Liu¹

Akshay Dave²

¹Idaho National Laboratory

²Argonne National Laboratory

December 2025

**Idaho National Laboratory
Light Water Reactor Sustainability
Idaho Falls, Idaho 83415**

<http://www.lwrs.inl.gov>

**Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517**

Page intentionally left blank

EXECUTIVE SUMMARY

As foundation models (FMs) in general, and large language models (LLMs) in particular, achieve unprecedented advancements, they are revolutionizing the technology industry across various sectors. One of the most significant markets for these technologies is the nuclear power plant (NPP) industry. However, NPPs are characterized by their complexity and stringent safety requirements. Consequently, any technological advancements that can enhance plant reliability and alleviate safety concerns from human operators are highly sought after and have the potential to provide substantial benefits.

Nuclear systems typically lack the luxury of immersive sensing capabilities, due to the harsh operational conditions, physical inaccessibility, and stringent cost constraints involved. These limitations result in streaming data that are often barely sufficient for comprehensive monitoring and analyses. Additionally, numerous text-based sources such as work orders, operator logbooks, engineering reports, manuals, and drawings are processed independently of any real-time dynamics. If these various data modalities could be integrated through a centralized intelligence system or coupled and correlated via agentic artificial intelligence (agentic AI), it would be possible to detect anomalies, off-normal operating conditions, degradation-induced drifts, and even preemptive accident scenarios well in advance of their occurrence. This integration could also significantly mitigate unplanned maintenance shutdowns, thereby enhancing the overall efficiency and safety of NPPs.

In line with the current U.S. administration’s vision for energy security, nuclear renaissance, and significant investments in artificial intelligence (AI), the U.S. Department of Energy’s Light Water Reactor Sustainability (LWRS) program has decided to invest in adapting open-source foundation models for nuclear applications. This strategic initiative is geared toward exploring and comparing different approaches, including fine-tuning, nuclear-specific retrieval-augmented generation (RAG), and mixture of agents (MoAs).

The present study was envisioned to perform an exhaustive literature review and comparative analysis of the available open-source flagship models. The primary objective is to utilize existing models without retraining via extensive hardware utilization. Instead, the focus is on investigating parallel routes for enhancing current open-source models. The central hypothesis being examined is whether development of multimodal nuclear-specific embeddings, incorporation of higher-quality nuclear-specific RAG, and/or utilization of low-rank adaptation (LoRA) fine-tuning can enhance multimodal foundation models (MFMs) to the point of outperforming traditional and classical anomaly detection techniques based solely on tabular sensor data.

Achieving this objective entails a series of methodical steps. First, we will synthesize a comprehensive dataset that mimics engineering processes, incorporating both synthetic and real-world data. Next, we will implement and fine-tune several models, evaluating their performance in various scenarios. The evaluation criteria will include the models’ ability to detect anomalies, predict off-normal operating conditions, identify degradation-induced drifts, and forecast potential accident scenarios.

Furthermore, the study will explore the benefits of integrating these advanced models into existing NPP systems. This will involve assessing the feasibility of real-time data integration, the potential for reducing human operator workload, and the overall impact on plant safety and reliability. Through detailed case studies and simulations, we aim to demonstrate the practical applicability and advantages of these AI-driven approaches.

In conclusion, this research substantiates the transferability attribute of foundation models. It demonstrates that these models, even when pre-trained on tasks not initially related to anomaly detection, can be effectively utilized for anomaly detection, following a degree of fine-tuning and augmentation. The findings of this study may pave the road to significant advancements in the monitoring and safety of NPPs, ultimately

contributing to the broader goals of energy security and sustainability.

ACKNOWLEDGMENTS

The authors wish to thank the Light Water Reactor Sustainability program for funding this effort. The visualizations and explanations of transformer concepts in this work were inspired by the educational content of 3Blue1Brown [1, 2], and many figures were created via the great animation library Manim [3, 4].

USE OF AI TOOLS AND LARGE LANGUAGE MODELS

This work made extensive use of large language models (LLMs) as assistive tools during the research and writing process. Specifically, the following AI systems were employed:

- **Claude Sonnet 4.5** (Anthropic) for literature synthesis, technical writing assistance, code development, and conceptual explanations
- **gpt-oss-120b**, **GPT-codex** (OpenAI) for report outlining, paraphrasing, command line integration, workflow automation, embeddings
- **bge-m3** for embeddings.

The LLMs were used for the following purposes:

- Paraphrasing and improving the clarity of technical explanations
- Generating initial drafts of code implementations and visualizations
- Identifying relevant literature and summarizing research papers
- Debugging and optimizing computational workflows
- Suggesting alternative phrasings and organizational structures.

All LLM-generated content was critically reviewed, fact-checked against primary sources, and significantly revised by the authors.

SCOPE AND INTENDED USE OF THIS DOCUMENT

This report serves a dual purpose: (1) to establish a comprehensive scientific foundation for applying multimodal foundation models to NPP anomaly detection, and (2) to demonstrate the practical feasibility and superiority of this approach through representative case studies on industrial benchmarks.

Rationale for Extensive Literature Coverage

Sections 2 and 3 constitute an in-depth technical exposition of foundation model architectures, multimodal integration strategies, and agentic workflows. This material is presented in tutorial style, along with substantial mathematical formalism and explanatory diagrams, for the following reasons:

1. **Knowledge Transfer to the Nuclear Domain:** Foundation models and LLMs represent rapidly evolving technologies that have not yet been systematically adapted for safety-critical nuclear applications. The detailed treatment bridges the gap between the AI/ML research community and nuclear engineering practitioners.
2. **Multi-Stakeholder Audience:** This document addresses a diverse readership, including:
 - Nuclear engineers and plant operators seeking to understand AI-driven monitoring tools
 - Computer scientists requiring domain context for anomaly detection in safety-critical systems
 - DOE program managers and federal decision-makers evaluating technology maturity and deployment pathways
 - Regulatory bodies assessing the scientific basis for AI-assisted operations.
3. **Long-Term Reference Value:** As foundation models continue to advance, this report is designed to serve as a self-contained technical reference that will remain useful for future LWRs program activities, technology transfer efforts, and academic collaboration.
4. **Reproducibility and Transparency:** The mathematical formulations and architectural details enable independent verification and extension of our methods by other research teams.

Organization and Reading Guidance

Readers may strategically approach this document in light of their specific background and objectives:

- **Executive Summary, Introduction, and Conclusions (Sections 1 and 7):** Provide high-level findings and recommendations for non-technical stakeholders.
- **Demonstration Benchmarks and Results (Sections 5 and 6):** Present empirical validation on industrial datasets (SKAB and Industrial Utility Benchmark) for practitioners interested in performance metrics and operational feasibility.
- **Technical Foundation (Sections 2 and 3):** Offer comprehensive treatment of underlying methods for researchers, developers, and advanced practitioners seeking a detailed understanding of—or planning to implement—similar systems.

Pedagogical Design Decisions

Several design choices prioritize accessibility and completeness over brevity:

- **Mathematical notation is introduced progressively**, with intuitive explanations preceding formal definitions.
- **Figures and diagrams are integrated inline** rather than relegated to appendices, allowing readers to grasp concepts without repeatedly cross-referencing supplementary material.
- **Nuclear engineering examples** (reactor types, component systems, operational scenarios) are woven throughout to maintain relevance and demonstrate domain-specific applications.
- **Background material is self-contained**, minimizing reliance on external references for core concepts, as is particularly important given the rapid evolution and fragmentation of foundation model literature.

Relationship to Milestone Objectives

This report fulfills the LWRs milestone requirement to “develop the scientific basis for multimodal foundation models in anomaly detection” by:

1. Systematically surveying architectural paradigms (encoder-decoder, decoder-only, diffusion models) and their suitability for tabular and time-series data
2. Establishing theoretical frameworks for multimodal fusion (early/late fusion, cross-modal attention, contrastive learning, unified tokenization)
3. Demonstrating empirically that pre-trained foundation models (TabPFN) outperform traditional ML methods (Random Forest, XGBoost, CatBoost) on nuclear circulating water system data, with multimodal inputs yielding superior accuracy and reducing false-negative rates
4. Prototyping agentic workflows that combine foundation models with retrieval-augmented generation and tool-use for explainable, context-aware anomaly diagnoses.

The extensive literature review component (approximately 60%–70% of the report) was originally conceived as the primary deliverable, but has been augmented with practical demonstrations (Sections 5–7) at program leadership’s request so as to validate theoretical findings and establish performance baselines for future deployments.

Limitations and Future Milestones

While this report provides a comprehensive foundation, it does not constitute deployment-ready guidance. Operational implementation would require:

- Regulatory engagement to establish acceptance criteria for AI-assisted monitoring
- Cybersecurity assessments for integration with plant digital infrastructure
- Human factors studies to optimize operator interaction with agentic systems
- Validation on additional nuclear-specific datasets beyond circulating water systems
- Addition of other modalities such as images, vibration/acoustics, etc., so as to be able to capture a more integrated fusion of correlated data
- Creation of modality-specific embeddings or a proper way to project/encode them before passing them to the FM
- Multimodal finetuning.

The second half of these activities are being addressed for a subsequent LWRs program milestone, and they are informed by the scientific basis established herein.

Page intentionally left blank

CONTENTS

EXECUTIVE SUMMARY	v
1 INTRODUCTION	1
2 OUTLOOK ON FOUNDATION MODELS	3
2.1 Taxonomy of Modern AI Systems	3
2.1.1 Foundation Models and Architectural Families	3
2.1.2 Capability Tiers and Deployment Classes	4
2.1.3 Learning Paradigms and Adaptation Strategies	4
2.1.4 Key Technical Concepts	5
2.2 Typical workflow	6
2.2.1 Tokenization	7
2.2.2 Embeddings	9
2.2.3 Attention Mechanisms	20
2.2.4 Finetuning	29
2.2.5 Retrieval-Augmented Generation	32
2.3 Foundation model architectures.	34
2.3.1 Encoder-Decoder Overview	35
2.3.2 Encoder-Decoder Tokenization and Embeddings	36
2.3.3 Encoder-Decoder Attention Stack	36
2.3.4 Encoder-Decoder Decoding and Alignment	36
2.3.5 Decoder-Only Overview	36
2.3.6 Decoder-Only Tokenizer and Positional Encodings	36
2.3.7 Decoder-Only Attention and Masking	38
2.3.8 Decoder-Only Output Head	38
2.3.9 Stable Diffusion Overview	38
2.3.10 Stable Diffusion Tokenizer and Conditioning	40
2.3.11 Stable Diffusion Latent Attention	40
2.3.12 Stable Diffusion Sampling and Decoding	40
2.4 Multimodal Integration Approaches	41
2.4.1 Early Fusion vs. Late Fusion	42
2.4.2 Cross-Modal Attention	44
2.4.3 Contrastive Learning	45
2.4.4 Unified Tokenization Spaces	47

2.5	Foundation Models for Anomaly Detection	48
2.5.1	Zero-Shot and Few-Shot Anomaly Detection	49
2.5.2	Reconstruction-Based Approaches	51
2.5.3	Embedding-Based Approaches	53
2.5.4	Generative Approaches	55
2.6	Evaluation Frameworks and Benchmarks	57
2.6.1	Benchmark Datasets and Evaluation Suites	60
2.6.2	Evaluation Protocols and Best Practices	62
2.6.3	Domain-Specific Evaluation Considerations	63
2.7	AI Agents And Agentic Orchestration	65
2.7.1	Tool Calling	65
2.7.2	Agent vs. LLM	66
2.7.3	Beyond the Baseline	67
3	BASE MODEL	70
3.1	Architecture Overview	70
3.2	Feature and Label Tokenization	71
3.3	In-Context Inference	72
3.4	Training Regimen and Deployment Considerations	74
4	PHYSICS-BASED ANOMALY DETECTION	75
4.1	Introduction	75
4.2	The Problem of Biofouling and Debris Accumulation	75
4.3	Operational and Economic Consequences	76
4.4	The Shift to Physics-Based Diagnostics	76
4.5	System Description	77
4.5.1	CWS Role in the PWR Thermal Cycle	77
4.5.2	System Component Specifications	78
4.5.3	Circulating Water Intake Structure	78
4.5.4	Circulating Water Pumps	79
4.5.5	Main Condenser	79
4.5.6	Sensor Set Adopted	79
4.6	Simulation Model Development in Dymola	79
4.6.1	Pressure Loss Models	81
4.6.2	Circulating Water Pump Model	81
4.6.3	Condenser Model	82

4.7	PBD Model Development and Implementation	82
4.7.1	Physics-based Diagnostic Models	83
4.7.2	Available Diagnostic Models and Residuals Based on Sensor Configurations	83
4.7.3	Explainability using LLMs	84
4.8	Simulation Results: Startup and Fault Scenarios	84
4.8.1	Startup Simulation	84
4.8.2	Fault Scenario Simulations	85
4.8.3	Trash Rack Fouling	85
4.8.4	Waterbox Fouling	86
4.8.5	CWP Discharge Pressure Sensor Fault	87
4.9	Fault Diagnosis Results	87
4.9.1	Fault Scenario 1: Trash Rack Fouling	88
4.9.2	Fault Scenario 2: Waterbox Fouling	91
4.10	Fault Scenario 3: Pump Discharge Pressure Sensor Fault	93
4.10.1	Discussion of LLM Explainability	95
4.11	Physics-based Discussion and Conclusion	95
5	BENCHMARKS	97
5.1	Skoltech Anomaly Benchmark (SKAB)	97
5.2	Industrial Utility Benchmark	98
6	RESULTS	101
6.1	SKAB Dataset	101
6.1.1	Single Modality Binary Classification	102
6.1.2	Multimodal Binary Classification	103
6.1.3	Comparison of Classification Models	105
6.1.4	RAG-Based Additional Features	106
6.1.5	Agentic Workflow	109
6.2	Industrial Utility Benchmark Dataset	118
6.2.1	Single-Modality Binary Classification	119
6.2.2	Multimodal Binary Classification	120
6.2.3	Multi-Class Classification	122
7	CONCLUSIONS	125
	REFERENCES	128

LIST OF FIGURES

Figure 1.	Schematic of multimodal data acquisition in an NPP (sensor streams, operator logs, and maintenance records) and its ingestion by a pre-trained foundation model. The model processes the combined modalities and then outputs anomaly scores either as a classification label or a regression-based severity estimate, perhaps subsequently triggering automated control actions or scheduled component maintenance.	2
Figure 2.	Transformer workflow predicting the next word	6
Figure 3.	Tokenization process in LLMs, with each word in a sentence being broken down into individual tokens. The LLM processes these tokens through its layers to generate a prediction for the subsequent word, demonstrating the core mechanism of next-word prediction in language modeling.	8
Figure 4.	Each word in the vocabulary is represented as a high-dimensional vector in a continuous vector space.	9
Figure 5.	Embedding space visualization of nuclear reactor types. Vector proximity encodes similarity: pressurized-water reactor (PWR) and boiling-water reactor (BWR) cluster tightly (both light-water reactors), high-temperature gas-cooled reactor (HTGR) and Advanced Gas Reactor (AGR) group together (gas-cooled designs), while liquid fluoride thorium reactor (LFTR) and sodium fast reactor (SFR) share commonalities despite belonging to different reactor families (molten-salt reactor vs. fast neutron reactor). The embeddings capture both taxonomic relationships and shared operational characteristics.	10
Figure 6.	Multimodal embeddings correlations.	11
Figure 7.	2D projection of reactor type embeddings. Semantically similar reactor types cluster together: pressurized-water reactor (PWR) and boiling-water reactor (BWR) (both light-water reactors), high-temperature gas-cooled reactor (HTGR) and Advanced Gas Reactor (AGR) (gas-cooled), and sodium fast reactor (SFR) and liquid fluoride thorium reactor (LFTR) (Generation IV designs). Vector proximity encodes multiple dimensions of similarity: coolant type, operating temperature, neutron spectrum, and safety philosophy.	20
Figure 8.	Comparison of embedding methodologies. (a) TF-IDF produces sparse, frequency-based vectors with no semantic awareness. (b) Word2Vec learns dense embeddings via neural prediction, capturing semantic similarity. (c) BERT generates contextualized embeddings that adapt to word usage, handling polysemy (“plant” as flora vs. facility).	20
Figure 9.	Skip-gram (Word2Vec) architecture. Given a target word (“pump”), the model predicts surrounding context words within a fixed window. The learned embedding $\mathbf{v}_{\text{pump}}$ encodes semantic relationships: words appearing in similar contexts receive similar embeddings.	21
Figure 10.	Positional embeddings in Transformers. Each token receives a word embedding $\mathbf{v}_w$ (capturing semantics) and a positional embedding $\mathbf{p}_t$ (encoding sequence position). The final representation $\mathbf{e}_t = \mathbf{v}_w + \mathbf{p}_t$ combines both. Right: sinusoidal positional encodings use different frequencies across embedding dimensions to uniquely encode each position.	22

Figure 11.	Multimodal embedding architecture. Different modalities (text, thermal images, vibration signals) are encoded by specialized neural networks into a shared embedding space. Contrastive learning ensures that semantically related inputs—regardless of modality—cluster together, enabling cross-modal retrieval (“find thermal images matching this acoustic signature”).	22
Figure 12.	Word embedding analogies capture semantic relationships. The vector arithmetic $\mathbf{v}_{\text{PWR}} - \mathbf{v}_{\text{water}} + \mathbf{v}_{\text{sodium}} \approx \mathbf{v}_{\text{SFR}}$ demonstrates that embeddings encode conceptual transformations: replacing water coolant with sodium transforms a pressurized-water reactor (PWR) into an sodium fast reactor (SFR). This linear substructure enables analogical reasoning in the embedding space.	23
Figure 13.	Self (scaled dot-attention) from the original paper: “Attention is All You Need.”[6]	23
Figure 14.	Multi-head attention from the original paper: “Attention is All You Need.”[6]	24
Figure 15.	Scaled dot-product self-attention.	25
Figure 16.	MHA: parallel heads, concatenation, and output projection.	26
Figure 17.	Soft attention (scaled dot-product self-attention).	27
Figure 18.	Hard attention	27
Figure 19.	Spatial attention learns 2D attention maps to emphasize salient spatial locations. The data/structures are shown by light blue, while light pink represents the operations. . . .	28
Figure 20.	Channel attention: learns channel-wise weights to emphasize important feature channels.	28
Figure 21.	Temporal attention: models dependencies across time steps in sequential data.	28
Figure 22.	Global attention: full-range attention across all spatial, channel, and temporal positions.	29
Figure 23.	Comparison of model adaptation strategies: (1) Full pre-training of the LLM from scratch, (2) conventional fine-tuning that updates all model weights on a downstream task, and (3) PEFT approach that modifies only a small set of task-specific parameters while keeping the bulk of the pretrained weights frozen.	30
Figure 24.	Transformer architecture from the original paper: “Attention is All You Need.”[6]	35
Figure 25.	Transformer encoder-decoder translation pipeline. Light-blue nodes denote input representations, light-green nodes capture outputs that feed subsequent modules, and light-pink nodes highlight transformational operations; the flowing attention strands mirror how a query distributes weight across the sequence.	37
Figure 26.	Tokenizer, embedding, and attention components within a transformer encoder-decoder system. Dashed arrows indicate how outputs from one component feed the next stage of the stack.	38
Figure 27.	Decoder-only transformer step highlighting the causal attention block and its masked dependency on observed tokens. The curved attention ribbons emphasize the active query focus.	39
Figure 28.	Tokenizer, embedding, and causal attention subcomponents used in decoder-only transformers.	40

Figure 29.	Stable Diffusion pipeline showing text conditioning, latent denoising, and image synthesis. Inputs appear in light blue, latent outputs in light green, and detailed operations in light pink.	41
Figure 30.	Tokenizer, conditioning embeddings, and cross-attention components within the Stable Diffusion architecture.	42
Figure 31.	The ROC curve’s shape determines the AUC, which defines the true and false positive rates.	59
Figure 32.	Scheme of multi-agent framework.	68
Figure 33.	Evolving agent architecture with human-in-the-loop oversight (adopted from [197]). .	69
Figure 34.	TabPFN workflow (source: “Accurate predictions on small data with a tabular foundation model” [200].)	70
Figure 35.	TabPFN inference pipeline. Support examples become context tokens that, together with a query token, are processed by a transformer trained to approximate Bayesian posteriors over synthetic tabular tasks [199].	71
Figure 36.	Tokenizer, transformer, and prediction head components within TabPFN. Tokenization enforces permutation invariance while exposing feature semantics, the transformer amortizes Bayesian updates, and the prediction head maps the query representation to class probabilities.	72
Figure 37.	TabPFN 2D attention (source: “Accurate predictions on small data with a tabular foundation model”) [200].	72
Figure 38.	Simplified pressurized water reactor (PWR) cooling circuit (left) showing the role of the CWIS in providing cooling water to the main condenser, and detailed cutaway view of the CWIS (right) illustrating the primary filtration defenses—trash racks and traveling screens—that protect downstream equipment from debris and aquatic life before water enters the circulating water pumps.	78
Figure 39.	Dymola process flowsheet of the Circulating Water System. The model captures the essential flow path from estuary intake (source) through debris filtration (trash rack), pumping (pump), heat exchange (waterbox and condenser), and discharge (sink). The boundary component provides thermal reservoir conditions representing the Estuary, while the system defaults block establishes gravitational reference.	80
Figure 40.	Circulating water pump performance curve in the stable region (reconstructed). . . .	81
Figure 41.	Main operating variables during the startup ramp.	85
Figure 42.	Main operating variables during the simulated trash rack fouling scenario.	86
Figure 43.	Main operating variables during the simulated waterbox fouling scenario.	86
Figure 44.	Main operating variables during the simulated discharge pressure sensor fault scenario. .	87
Figure 45.	Water-cooling pump testbed used to acquire the SKAB dataset. The rig contains the centrifugal pump, reservoir, flow and pressure sensors, and controlled valves for injecting faults.	98
Figure 46.	Circulating Water Pump Plant layout.	99
Figure 47.	Diagram of a CWP motor, showing sensor placements for vibration and temperature monitoring.	100

Figure 48.	Acceleration, current, pressure, temperature, voltage, and volume flow rate, as measured for a water pump system in which varying the flow from the outlet valve introduced anomalies in the system's operations.	101
Figure 49.	Single Modality-based anomaly detection results in a high ROC AUC of 0.9208 for unseen data. It has an accuracy of 95.46% but cannot accurately estimate the start of the anomaly.	102
Figure 50.	The confusion matrix displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The single-modality TabPFN classifier misclassified 49 (12.22%) anomalous instances as normal and three (0.40%) normal instances as anomalous.	102
Figure 51.	Adding a modality such as text descriptions along with sensor measurements increases the ROC-AUC from 0.9208 to 0.9865 on test data, with a classifier accuracy of 96.42%. This indicates a lower rate of false positives and shows that it can capture the start of the anomaly despite underestimating the end.	104
Figure 52.	The confusion matrix displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The Multimodal TabPFN classifier misclassified 39 (9.73%) anomalous instances as normal, and two (0.27%) normal instances as anomalous.	104
Figure 53.	Three-fold cross-validation comparing four models for classifying anomalous vs. normal data. TabPFN achieves the highest ROC-AUC with both single and multimodal data. The additional step of encoding string data is required for CatBoost, Random Forest, and XGBoost. For all four models, adding a modality improves the ROC-AUC for classification.	106
Figure 54.	The confusion matrix for Baseline A displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The TabPFN classifier misclassified two (4.3%) anomalous instances as normal and eight (10.7%) normal instances as anomalous.	108
Figure 55.	The confusion matrix for Baseline B displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The TabPFN classifier misclassified one (2.2%) anomalous instances as normal and six (8.0%) normal instances as anomalous.	109
Figure 56.	Simple agent workflow with tool access for single modality and binary classification.	110
Figure 57.	Sensor data, including motor stator temperature, inlet/outlet temperature, inlet pressure, motor and pump age, and number of active CWPs, are used to predict classification labels indicating anomalous conditions such as water fouling and normal healthy states for binary classification.	118
Figure 58.	Sensor measurements used for multi-class classification, illustrating the two anomaly types: water-box fouling and CWP diffuser fault.	119
Figure 59.	Single modality based anomaly detection results in a high ROC-AUC of 0.9860 for unseen data. Its accuracy is 9.46%, but it cannot accurately estimate the start of the anomaly.	120

Figure 60. Adding a modality such as operator logs, along with sensor measurements, increases the ROC-AUC from 0.9860 to 0.9961 on the test data, and the classifier has an accuracy of 99.70%. This reduces the number of false positives from 176 to 45, and the start and end of the anomalies are successfully captured. 121

Figure 61. TabPFN accurately identifies classes in multi-class classification and achieves zero false negatives on unseen data. This makes TabPFN a reliable model for safety-critical systems, as anomalies do not go undetected. 123

Figure 62. Three-fold cross-validation evaluates four models on a multi-class anomaly-detection task (anomalous vs. normal). 124

EVALUATION OF THE FEASIBILITY OF MULTIMODAL FOUNDATION ARTIFICIAL INTELLIGENCE MODELS FOR ANOMALY DETECTION AND DEVELOPMENT OF THE SCIENTIFIC BASIS

1 INTRODUCTION

Recent years have shown a rise in the use of historical and real-time data generated from engineering systems for forecasting time-series data, detecting anomalous conditions, and performing predictive maintenance by leveraging data science and machine learning (ML) techniques. In safety-critical systems, it is essential that real-time data streaming be utilized for monitoring vital system variables and establishing downstream control tasks. One of the most crucial tasks is detecting anomalous conditions that may arise during normal operations, as these can lead to catastrophic failures downstream. This is particularly vital for NPPs, where unexpected equipment degradation and failures can lead to unplanned downtimes or shutdowns, in turn reducing revenue and increasing grid instability. To prevent these issues, plants are investigating automated anomaly detection tools. These tools utilize algorithms to analyze the large volumes of process measurement data sent to plant computers, and to identify subtle indicators of potential problems before those problems escalate. Early detection would grant NPPs additional lead time, allowing for improved planning and problem response.

The advent of deep learning methods and large language models (LLMs) has revolutionized data processing, handling, and inference across various domains, including NPPs. In the nuclear industry, LLMs have been utilized for information retrieval, creation of databases of essential components, knowledge preservation, digitization and structuring of documents, sentiment analyses, and synthetic data generation. As the name suggests, LLMs are specifically trained on text and code so as to perform language-based tasks. In NPPs, data streaming from sensors measuring critical values, such as neutron flux and temperature, is numerical. Infrared monitoring generates videos and images, and procedural documents and log books are textual in nature. Consequently, NPPs handle significant amounts of multimodal data. To fully understand events across various components in an NPP, it is crucial to analyze multimodal data collectively, as different modalities may detect different aspects of an anomaly.

In recent years, deep learning techniques and LLMs have been integrated into foundation models, which can adapt to a wide range of downstream tasks thanks to their training on large datasets. The term “foundation model” was introduced at the Stanford Institute for Human-Centered AI [5]. Foundation models have learned to bridge the gap between various modalities and large-scale training data, enabling contextual reasoning, generalization, and prompt capabilities at test time. These models can be modified through human-provided prompts, without retraining. Examples of this include segmenting a particular object by providing a bounding box when dealing with images and object recognition, engaging in interactive dialogues by asking questions about an image or video scene, and manipulating a robot’s behavior through language instructions. Due to their ability to process multimodal data and adapt to different tasks without retraining or fine-tuning, this work explores the potential of foundation models for detecting anomalies in nuclear components by analyzing textual and time-series data.

Modern deployments increasingly pair retrieval-augmented generation (RAG) with agentic workflows to make models both grounded and actionable. RAG retrieves task-relevant evidence from external knowledge bases and feeds it into the model at inference time, thereby improving factuality and keeping outputs current

by combining parametric knowledge with non-parametric memory. In parallel, agentic approaches treat the LLM as a planner/controller that reasons in natural language, calls tools or application programming interfaces (APIs), inspects observations, and iterates—a pattern exemplified by Reasoning and Acting (ReAct)’s interleaving of “think-then-act” steps to reduce hallucinations and make decisions auditable. Tool-use itself is becoming a first-class capability, with models learning when and how to invoke external functions, and literature documenting rapid progress in autonomous LLM agents. Emerging standards such as the Model Context Protocol (MCP) further stabilize these systems by defining typed tool schemas and connections to enterprise data, thus enabling safer, more reliable orchestration across heterogeneous sources and services.

The primary objective of this research is to survey the feasibility of foundation models for anomaly detection based on multimodal data sources while maintaining the safety and reliability of commercial NPPs, as seen in Figure 1.

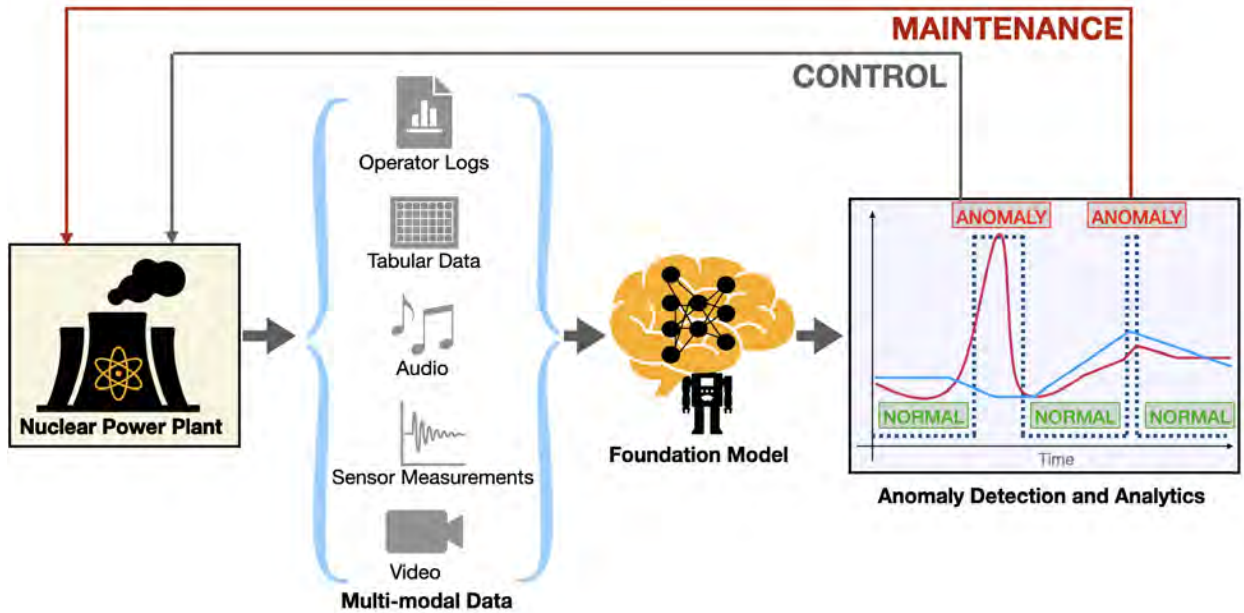


Figure 1: Schematic of multimodal data acquisition in an NPP (sensor streams, operator logs, and maintenance records) and its ingestion by a pre-trained foundation model. The model processes the combined modalities and then outputs anomaly scores either as a classification label or a regression-based severity estimate, perhaps subsequently triggering automated control actions or scheduled component maintenance.

2 OUTLOOK ON FOUNDATION MODELS

The application of multimodal foundation models to anomaly detection represents a convergence of several rapidly evolving research streams. This literature review is structured to provide both the architectural foundations and the domain-specific context necessary to understand the current approaches and identify opportunities for advancement.

We begin with a taxonomy of modern AI systems in order to establish shared terminology and clarify the scope of models under consideration (Section 2.1). We then examine the core architectural paradigms that underpin modern foundation models (Section 2.3). This includes detailed treatments of their constituent mechanisms (e.g., tokenization, attention, and generation strategies). Section 2.4 addresses the specific challenge of multimodal integration, meaning how models fuse information from heterogeneous data sources such as images, text, and time-series and sensor data. We then narrow our focus to examine how these powerful architectures have been adapted and applied to anomaly detection tasks (Section 2.5), and ultimately conclude with a discussion on evaluation methodologies and benchmark datasets (Section 2.6).

2.1 Taxonomy of Modern AI Systems

The rapid evolution of AI has produced a constellation of overlapping terms that describe models, capabilities, and training paradigms. This section establishes a consistent nomenclature for the AI systems relevant to multimodal anomaly detection, organizing them hierarchically from broad architectural classes to specific learning approaches.

2.1.1 Foundation Models and Architectural Families

Foundation models refer to large-scale neural networks that are pre-trained on broad, diverse datasets and can be adapted to a wide range of downstream tasks, with minimal task-specific modifications [5]. Unlike traditional ML models trained for a single purpose, foundation models acquire general-purpose representations during pre-training that transfer effectively across domains and modalities. This paradigm shift has enabled practitioners to leverage massive computational investments in pre-training for specialized applications, including anomaly detection in industrial and scientific contexts.

Foundation models can be categorized by their primary modality and architectural design:

Large Language Models (LLMs) are foundation models specially designed for processing and generating natural language. Examples include GPT-4, Claude, and LLaMA. These models are typically decoder-only transformers trained on vast text corpora by using next-token prediction. While originally designed for text, recent LLMs increasingly incorporate multimodal capabilities.

Vision-Language Models (VLMs) bridge visual and textual modalities, enabling tasks such as image captioning, visual question answering, and text-to-image retrieval. CLIP, ALIGN, and Flamingo exemplify this category. VLMs are particularly relevant for anomaly detection scenarios involving visual inspection combined with textual maintenance logs or sensor annotations.

Multimodal Foundation Models extend beyond vision and language to integrate additional modalities such as time-series data, audio, sensor readings, and structured tabular data. These models are central to this review, as industrial anomaly detection often requires the fusing of heterogeneous data sources—namely, thermal imaging, vibration sensors, operational logs, and maintenance records.

Generative AI describes systems capable of producing novel content—text, images, audio, or synthetic

data—rather than merely classifying or predicting from existing inputs. Generative models such as GPT for text, Stable Diffusion for images, and diffusion models for time-series data can synthesize realistic anomalies for training data augmentation or serve as density estimators for out-of-distribution detection.

In contrast, **discriminative models** focus on classification, regression, or prediction tasks. While encoder-only architectures such as BERT and Vision Transformers are technically discriminative, many modern foundation models blur this distinction by supporting both generative and discriminative fine-tuning.

2.1.2 Capability Tiers and Deployment Classes

Frontier AI refers to the most capable, state-of-the-art models that push the boundaries of performance on benchmark tasks. These models typically emerge from well-resourced research laboratories and demonstrate emergent capabilities absent in their smaller predecessors. For anomaly detection, frontier models offer the potential for few- or zero-shot detection without extensive labeled anomaly datasets.

Flagship models are representative, widely adopted systems from major AI laboratories that serve as reference implementations or benchmark comparisons. Examples include OpenAI’s GPT series, Anthropic’s Claude models, and Google’s Gemini family. While not always the absolute frontier, flagship models are well-documented, accessible, and validated across diverse applications.

Agentic AI encompasses systems that autonomously plan, execute multi-step workflows, and interact with external tools or environments to achieve specified goals. Agentic capabilities enable foundation models to orchestrate complex anomaly investigation workflows—querying databases, analyzing sensor streams, generating diagnostic reports, and recommending corrective actions, without human intervention at each step.

2.1.3 Learning Paradigms and Adaptation Strategies

The practical application of foundation models to anomaly detection leverages several learning paradigms:

Transfer learning describes the process of adapting a pre-trained foundation model to a new task or domain. Rather than training from scratch, transfer learning initializes model parameters from pre-training and refines them on task-specific data. This approach is foundational to the utility of foundation models, as it allows practitioners to benefit from large-scale pre-training even with limited domain-specific data.

Fine-tuning is a specific form of transfer learning in which model parameters are updated via gradient descent on labeled examples from the target task. Fine-tuning can range from updating all parameters (full fine-tuning) to adjusting only a small subset (parameter-efficient fine-tuning via adapters or low-rank adaptation).

Zero-shot learning refers to a model’s ability to perform a task despite not having any task-specific training examples, relying solely on its pre-trained knowledge and a natural language description or specification of the task. For anomaly detection, zero-shot approaches enable deployment without labeled anomaly data, instead using textual descriptions of normal operating conditions or anomaly types.

Few-shot learning extends zero-shot learning by providing a small number (typically 1–10) of labeled examples at inference time. The model conditions its predictions on these examples, without gradient-based parameter updates. Few-shot learning is particularly valuable in anomaly detection, where labeled anomalies are rare and expensive to collect.

In-context learning is the mechanism via which LLMs and some multimodal models learn from examples provided in the input prompt or context window. Unlike fine-tuning, in-context learning does not

modify model parameters; instead, the model leverages its attention mechanism to adapt its behavior based on the provided examples. This capability enables rapid prototyping of anomaly detection systems by simply formatting historical cases as in-context demonstrations.

Self-supervised learning encompasses training objectives that generate supervisory signals from the data itself, without human annotation. Masked language modeling (predicting obscured tokens), contrastive learning (distinguishing similar from dissimilar examples), and denoising (reconstructing corrupted inputs) are common self-supervised objectives used during foundation model pre-training. Self-supervised objectives are particularly relevant for anomaly detection, as reconstruction error and embedding distances can serve as anomaly scores.

2.1.4 Key Technical Concepts

Several technical concepts that recur throughout discussions of foundation models warrant brief definition:

Embedding spaces are high-dimensional vector spaces in which models represent inputs (tokens, images, sensor readings) as continuous vectors. Semantically similar inputs are mapped to nearby points in the embedding space, enabling operations like nearest-neighbor search and interpolation. Anomaly detection methods often leverage embedding spaces to identify outliers as points distant from normal operating regions.

Latent spaces refer to the compressed, learned representations within a model’s hidden layers. Variational autoencoders and latent diffusion models explicitly learn structured latent spaces that capture data-generating factors. Anomalies that violate learned latent structure produce reconstruction errors or low likelihood under the latent distribution.

Pre-training denotes the initial phase of foundation model development, in which the model is trained on large, diverse datasets by using self-supervised objectives. Pre-training imbues the model with general-purpose representations and capabilities that transfer to downstream tasks.

Contextualization describes how modern transformers produce token representations that depend on surrounding context. Unlike static word embeddings, contextualized representations vary based on the sequence in which a token appears, enabling models to disambiguate meaning and capture long-range dependencies.

Having established this taxonomy, we now turn to the architectural families that instantiate these concepts and enable multimodal anomaly detection.

2.2 Typical workflow

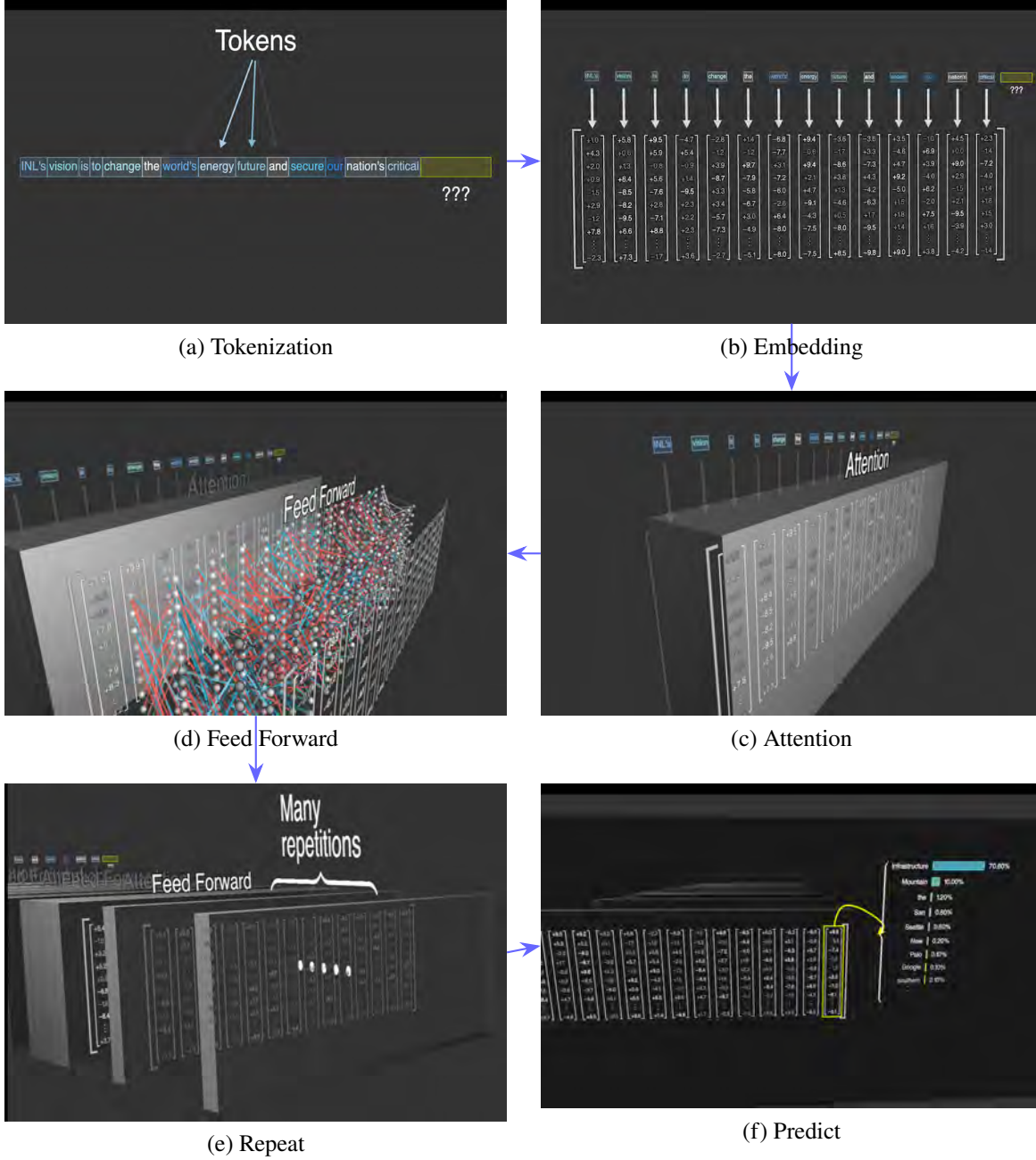


Figure 2: Process flow showing the standard transformer workflow from tokenization to predicting the next word. Phrase: “INL’s vision is to change the world’s energy future and secure our nation’s critical ____.” The predicted next word should be **infrastructure**. (Figure adopted from the 3Blue1Brown YouTube channel and created using the Manim library.)

Figure 2 provides a visual representation of how the transformer architecture predicts subsequent tokens through its autoregressive generation process. In this section, we provide a comprehensive overview and detailed explanations of the key components that typically comprise the pipeline of LLMs, including both agentic and foundation model approaches. We begin with tokenization, where input words are converted

into numerical representations, then proceed to discuss embeddings, which form the core of LLMs. We then explore the attention mechanism, fine-tuning techniques, and conclude with RAG.

To illustrate this workflow, consider the following example prompt: “INL’s vision is to change the world’s energy future and secure our nation’s critical _____.” The transformer’s autoregressive generative procedure [6] initiates with the tokenization phase, which exhibits notable variation across different model architectures. For instance, some tokenization schemes treat [INL] and [’s] as two distinct tokens, while others may combine them into a single unit. This variability stems from differences in the training data, vocabulary size constraints, and the specific subword tokenization algorithms employed, such as Byte-Pair Encoding [7], WordPiece [8, 9], or SentencePiece [10]. Following tokenization, the workflow proceeds to the embedding stage, commonly referred to as word-to-vector or token-to-vector embedding. This transformation is essential because transformer architectures, like all neural networks, operate exclusively on numerical data. During this phase, each discrete token is mapped to a high-dimensional continuous vector representation, typically ranging from several hundred to several thousand dimensions. These dense vector embeddings encode semantic and syntactic information learned during the model’s pre-training phase, positioning semantically similar tokens in proximity within the high-dimensional vector space.

Once the input sequence has been converted into a series of numerical vectors, the transformer’s attention mechanisms are engaged [6]. These sophisticated mechanisms enable the model to dynamically weigh the relevance of different tokens relative to one another, effectively maintaining contextual relationships throughout the sequence. This context-awareness is what allows transformers to understand, for example, that “energy” and “future” are semantically related to “INL’s vision” in our example prompt. The contextualized representations are then passed through feed-forward neural network layers, which apply non-linear transformations to further refine the representations. This process—comprising attention mechanisms followed by feed-forward processing—is repeated numerous times through multiple transformer layers, with each layer progressively building more sophisticated and abstract representations of the input. Finally, the model generates its output by selecting the next token, based on the probability distribution computed from its learned representations. The token with the highest probability from the model’s vocabulary is typically selected, though various sampling strategies can be employed to introduce controlled randomness and improve output diversity.

2.2.1 Tokenization

Tokenization represents the foundational step in the language model’s processing pipeline upon receiving a user prompt. This critical preprocessing phase involves decomposing input text into discrete units called tokens, which can represent whole words, subwords, or even individual characters—depending on the tokenization strategy employed (Figure 3). It is important to note that each flagship language model implements its own proprietary tokenization approach, optimized for its specific architecture and training corpus.

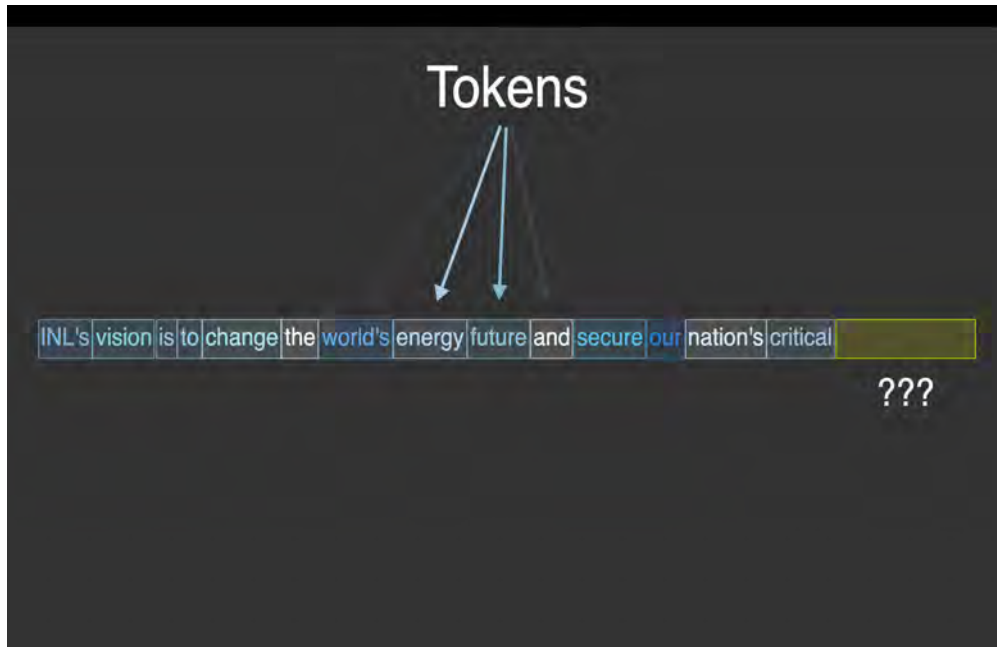


Figure 3: Tokenization process in LLMs, with each word in a sentence being broken down into individual tokens. The LLM processes these tokens through its layers to generate a prediction for the subsequent word, demonstrating the core mechanism of next-word prediction in language modeling.

2.2.2 Embeddings

Embeddings—a fundamental concept in natural language processing (NLP)—are crucial components of LLMs. They serve as numerical representations of words, phrases, or even entire sentences, enabling machines to understand and process human language more effectively. At their core, embeddings transform discrete textual data into continuous vector spaces (Figure 4). Each word or token in a vocabulary is mapped to a unique vector of real numbers. In the following sections, we explain different types of embeddings and offer visual examples.

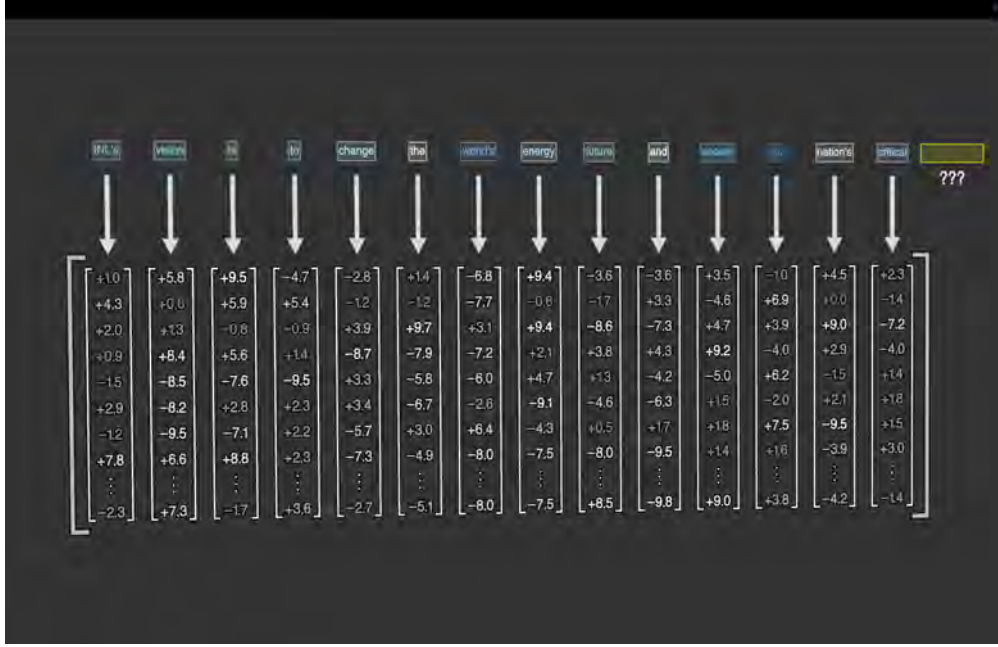


Figure 4: Each word in the vocabulary is represented as a high-dimensional vector in a continuous vector space.

Word Embeddings and Semantic Representations

Embedding models are the mechanisms via which LLMs encode tokens into vectors of numeric values representing the relative semantic similarities of contextualized words. The fundamental principle is that semantically similar words should have embeddings that are proximal in the learned vector space. Formally, if we denote the embedding of a word w as $\mathbf{v}_w \in \mathbb{R}^d$, where d is the embedding dimension, then semantically similar words w_i and w_j should satisfy $\cos(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) \approx 1$, where:

$$\cos(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) = \frac{\mathbf{v}_{w_i} \cdot \mathbf{v}_{w_j}}{\|\mathbf{v}_{w_i}\| \|\mathbf{v}_{w_j}\|}. \quad (1)$$

This geometric interpretation reveals the power of embeddings: the angle between two word vectors encodes semantic similarity. Words with similar meanings (synonyms) have small angular separation, yielding large positive inner products. Conversely, antonyms occupy opposing regions of the embedding space, producing negative or near-zero inner products. This property enables algebraic operations on semantic concepts—a capability that underpins many advanced natural language processing (NLP) applications [11, 12].

Nuclear Reactor Embeddings: A Domain Example Figure 5 illustrates how embeddings capture hierarchical relationships in the nuclear reactor domain. The angle between pressurized-water reactor (PWR) and boiling-water reactor (BWR) embeddings is small because both are light-water reactors that share fundamental design principles: water serves as both the coolant and moderator, operating pressures range from 7 to 15 MPa, and the fuel enrichment levels are similar (3%–5% U-235). In contrast, microreactors occupy a distinct region of the embedding space, thus reflecting their novel characteristics: ultra-compact design, factory fabrication, passive safety systems, and deployment flexibility for remote or military applications.

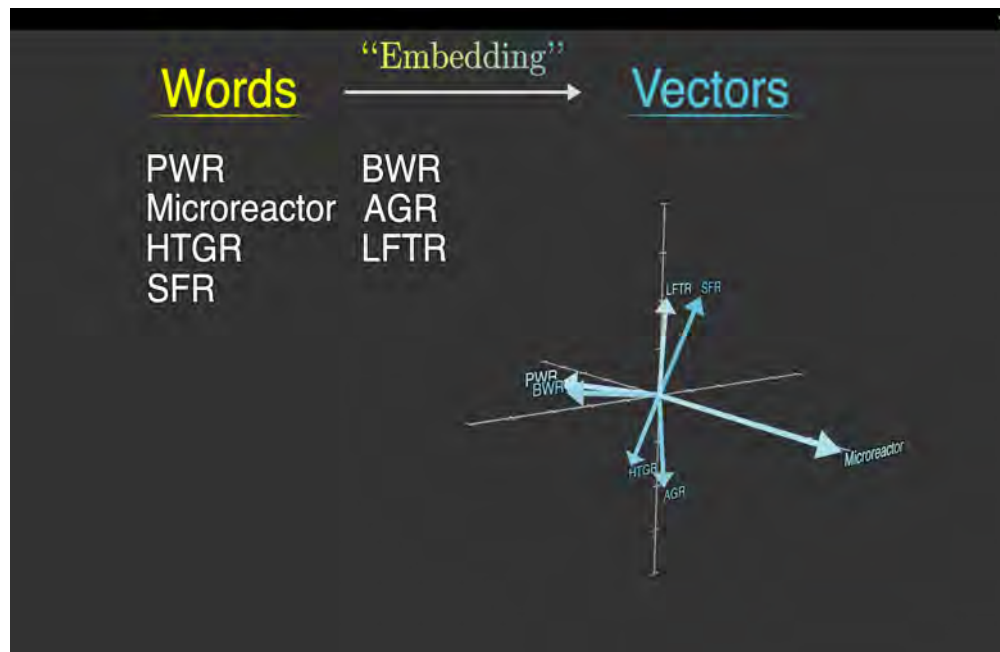


Figure 5: Embedding space visualization of nuclear reactor types. Vector proximity encodes similarity: PWR and BWR cluster tightly (both light-water reactors), high-temperature gas-cooled reactor (HTGR) and Advanced Gas Reactor (AGR) group together (gas-cooled designs), while liquid fluoride thorium reactor (LFTR) and sodium fast reactor (SFR) share commonalities despite belonging to different reactor families (molten-salt reactor vs. fast neutron reactor). The embeddings capture both taxonomic relationships and shared operational characteristics.

Gas-cooled reactors—high-temperature gas-cooled reactor (HTGR) and Advanced Gas Reactor (AGR)—similarly cluster together, united by their use of helium or carbon dioxide coolant, graphite moderation, and high-temperature operation (700°C–950°C). Perhaps most intriguingly, liquid fluoride thorium reactor (LFTR) (a molten-salt reactor) and sodium fast reactor (SFR) (a sodium-cooled fast reactor) exhibit moderate similarity despite belonging to different reactor families. Their proximity reflects shared Generation IV characteristics:

- **High operating temperatures:** Both exceed 500°C, enabling efficient thermodynamic cycles and potential for process heat applications (hydrogen production, desalination, synthetic fuel synthesis).
- **Low-pressure operation:** Unlike pressurized-water reactors, both operate at near-atmospheric pressure, eliminating certain accident scenarios and simplifying the containment design.
- **Advanced fuel cycles:** LFTR implements thorium-uranium fuel cycles with online reprocessing; SFR enables plutonium breeding and transuranic waste burning.

- **Passive safety features:** Natural circulation cooling, negative temperature coefficients of reactivity, and inherent shutdown mechanisms reduce reliance on active engineered systems.
- **Waste management potential:** Both offer pathways to reduce long-lived actinide inventories in comparison to current light-water reactor fleets.

This example illustrates how embeddings transcend simple categorical classification, capturing nuanced multi-dimensional relationships—reactor type, coolant chemistry, neutron spectrum, safety philosophy, and economic deployment context—in a unified vector representation. Similar to textual embeddings, other modality embeddings capture similarity and correlations. Fig. 6 depicts three images of different reactor types, followed by the correlation map between the image embeddings and textual embeddings. The premise is that the highest correlation value reveals the reactor type.

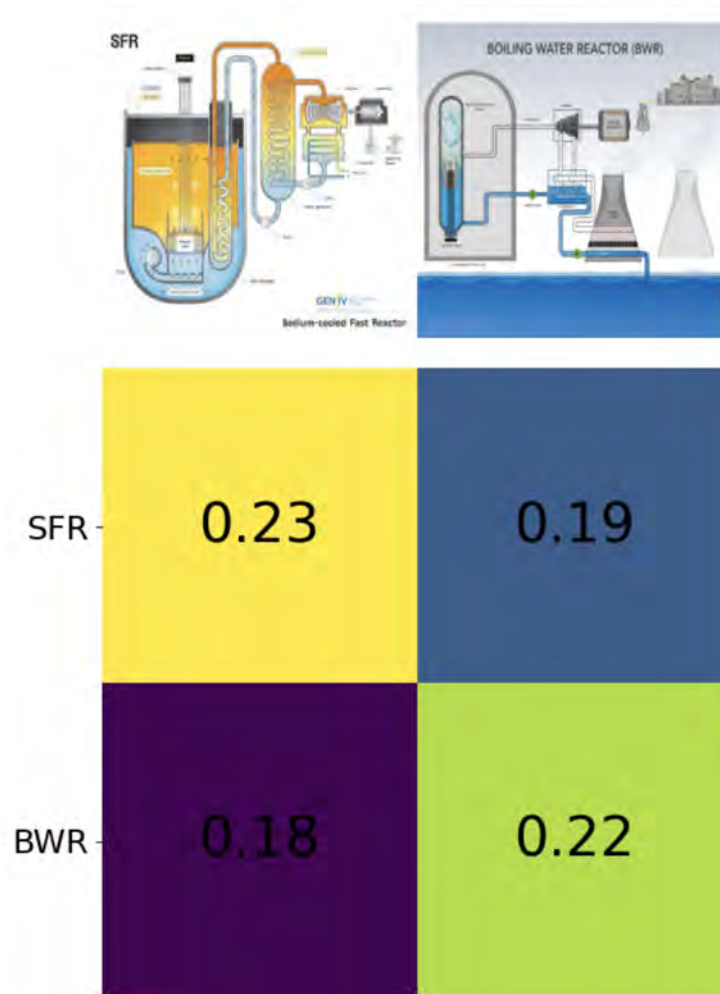


Figure 6: Multimodal embeddings correlations.

In an NLP, embeddings constitute one of the foundational differentiators among flagship AIs. They are critical to model performance, generalization capabilities, and transfer learning effectiveness. Embedding methodologies broadly partition into two families: frequency-based approaches that leverage statistical co-occurrence patterns, and prediction-based techniques that learn representations by optimizing language modeling objectives.

Frequency-Based Word Embeddings

Frequency-based embeddings construct word representations from corpus statistics—specifically, term frequencies and co-occurrence patterns. These methods are conceptually simple, computationally efficient, and interpretable, making them suitable for resource-constrained applications or scenarios requiring explainability.

Term Frequency-Inverse Document Frequency (TF-IDF) term frequency-inverse document frequency (TF-IDF) quantifies word importance by balancing local term frequency against global document frequency [13, 14]. For a term t in document d within corpus D , the TF-IDF weight is:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D), \quad (2)$$

where term frequency measures local prominence:

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \quad (3)$$

with $f_{t,d}$ denoting the count of term t in document d . Inverse document frequency penalizes ubiquitous terms that appear across many documents:

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}, \quad (4)$$

where $|D|$ is the total number of documents and the denominator counts documents containing t .

The logarithmic transform dampens the effect of extreme frequency disparities. A term appearing in every document receives zero IDF weight, correctly reflecting its lack of discriminative power. Conversely, rare but informative terms (e.g., “xenon poisoning” in nuclear engineering texts, “loss-of-coolant accident” in safety analyses) receive high weights, capturing domain-specific semantics.

TF-IDF embeddings represent documents as sparse vectors in $\mathbb{R}^{|V|}$, where $|V|$ is the vocabulary size. While effective for document retrieval and classification, TF-IDF suffers from several limitations: it ignores word order and syntactic structure, treats each term independently (no semantic similarity), and produces high-dimensional sparse representations that scale poorly with vocabulary size [15].

Co-occurrence Matrix and Distributional Semantics The distributional hypothesis—namely, “words appearing in similar contexts have similar meanings” [16, 17]—motivates co-occurrence based embeddings. A co-occurrence matrix $\mathbf{C} \in \mathbb{R}^{|V| \times |V|}$ counts how often word pairs appear together within a fixed context window. Element C_{ij} records the number of times word w_j appears within k positions of word w_i across the corpus.

Formally, for a corpus represented as a sequence of tokens $(w_1, w_2, \dots, w_N)$, the co-occurrence count is:

$$C_{ij} = \sum_{t=1}^N \sum_{\substack{-k \leq \Delta \leq k \\ \Delta \neq 0}} \mathbb{1}[w_t = w_i \wedge w_{t+\Delta} = w_j], \quad (5)$$

where $\mathbb{1}[\cdot]$ is the indicator function and k defines the context window size (typically 5–10 words). Larger windows capture broader topical associations; smaller windows encode tighter syntactic relationships.

The raw co-occurrence matrix is prohibitively high-dimensional ($|V| \times |V|$ can exceed 10^{10} entries for realistic vocabularies) and sparse. **Singular Value Decomposition (SVD)** compresses the matrix into low-dimensional dense embeddings [18]. SVD factorizes $\mathbf{C}$ as:

$$\mathbf{C} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \quad (6)$$

where $\mathbf{U} \in \mathbb{R}^{|V| \times d}$ contains left singular vectors (word embeddings), $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is a diagonal matrix of singular values (capturing the variance explained by each dimension), and $\mathbf{V}^\top \in \mathbb{R}^{d \times |V|}$ provides context embeddings. Truncating to the top d dimensions (typically $d = 100\text{--}300$) yields a compact representation that preserves most variance while filtering noise.

The resulting embeddings encode latent semantic relationships: words frequently co-occurring with similar contexts map to nearby points in $\mathbb{R}^d$. This approach underpins latent semantic analysis [19] and has been successfully applied to information retrieval, document similarity, and semantic search tasks.

However, SVD-based methods face computational challenges: constructing and decomposing large co-occurrence matrices requires $O(|V|^2)$ memory and $O(|V|^3)$ time, making them impractical for web-scale corpora. Moreover, the global matrix factorization does not leverage local context structure as effectively as neural prediction-based methods.

Prediction-Based Word Embeddings

Prediction-based embeddings learn representations by training neural networks to predict words from their contexts (or vice versa). Unlike frequency-based methods that directly encode co-occurrence statistics, prediction-based approaches optimize an objective function that implicitly captures distributional semantics. This paradigm shift—from explicit matrix factorization to implicit representation learning via gradient descent—underpins modern embedding techniques.

Word2Vec: Skip-gram and Continuous Bag-of-Words Word2Vec [11, 20] introduced two architectures that revolutionized word embeddings by demonstrating that simple neural networks trained on massive corpora could learn rich semantic and syntactic relationships.

Skip-gram predicts context words given a target word. For a target word w_t at position t in a sequence, the model maximizes the probability of observing surrounding words within a context window of size k :

$$\mathcal{L}_{\text{skip-gram}} = \frac{1}{N} \sum_{t=1}^N \sum_{\substack{-k \leq j \leq k \\ j \neq 0}} \log p(w_{t+j} \mid w_t), \quad (7)$$

where the conditional probability is parameterized via softmax over the vocabulary:

$$p(w_o \mid w_I) = \frac{\exp(\mathbf{v}_{w_o}^\top \mathbf{v}_{w_I})}{\sum_{w=1}^{|V|} \exp(\mathbf{v}_w^\top \mathbf{v}_{w_I})}. \quad (8)$$

Here, $\mathbf{v}_{w_I}$ is the embedding of the input (target) word and $\mathbf{v}_{w_o}$ is the context word embedding. Each word has two representations (one as target, one as context), though typically only the target embedding is retained after training.

The softmax normalization requires summing over the entire vocabulary, which is computationally prohibitive for large $|V|$. Word2Vec addresses this via **negative sampling** [20]: instead of computing the

full softmax, the model discriminates true context words from k randomly sampled “negative” words:

$$\log \sigma(\mathbf{v}_{w_O}^\top \mathbf{v}_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_{\text{noise}}} [\log \sigma(-\mathbf{v}_{w_i}^\top \mathbf{v}_{w_I})], \quad (9)$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function and P_{noise} is a noise distribution (often the unigram distribution raised to the 3/4 power to oversample rare words). This reduces the complexity from $O(|V|)$ to $O(k)$ per training example, enabling efficient scaling to billion-word corpora.

Continuous Bag-of-Words (CBOW) inverts the skip-gram objective: it predicts the target word from its surrounding context. Given context words $\{w_{t-k}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+k}\}$, CBOW maximizes:

$$\mathcal{L}_{\text{CBOW}} = \frac{1}{N} \sum_{t=1}^N \log p \left(w_t \left| \frac{1}{2k} \sum_{\substack{-k \leq j \leq k \\ j \neq 0}} \mathbf{v}_{w_{t+j}} \right. \right), \quad (10)$$

where context embeddings are averaged (or summed) before predicting the target. CBOW is faster to train than skip-gram because each example involves a single prediction rather than multiple context predictions. However, skip-gram often produces higher-quality embeddings, especially for rare words, because it generates multiple training examples per token occurrence.

Word2Vec embeddings exhibit remarkable semantic and syntactic regularities. The canonical example— $\mathbf{v}_{\text{king}} - \mathbf{v}_{\text{man}} + \mathbf{v}_{\text{woman}} \approx \mathbf{v}_{\text{queen}}$ —demonstrates that linear substructure in embedding space encodes analogical relationships [21]. In the nuclear domain, similar relationships emerge: $\mathbf{v}_{\text{PWR}} - \mathbf{v}_{\text{water}} + \mathbf{v}_{\text{sodium}} \approx \mathbf{v}_{\text{SFR}}$, capturing the conceptual transformation from water-cooled to sodium-cooled reactor designs.

FastText: Subword-Aware Embeddings Word2Vec treats each word as an atomic unit, ignoring internal morphological structure. This limitation becomes problematic for morphologically rich languages (e.g., Turkish and Finnish) and technical domains with extensive compound terminology. **FastText** [22] addresses this by representing words as bags of character n-grams.

For a word w , FastText constructs a set of character n-grams $\mathcal{G}_w$ (typically $n = 3-6$) plus the full word itself. For example, “reactor” with $n = 3$ yields $\mathcal{G}_{\text{reactor}} = \{\langle \text{re}, \text{rea}, \text{eac}, \text{act}, \text{cto}, \text{tor}, \text{or} \rangle, \text{reactor}\}$, where angle brackets denote word boundaries. The word embedding is the sum of its constituent n-gram embeddings:

$$\mathbf{v}_w = \sum_{g \in \mathcal{G}_w} \mathbf{v}_g, \quad (11)$$

where each n-gram g has a learned vector $\mathbf{v}_g \in \mathbb{R}^d$. The skip-gram or CBOW objective is then optimized with these compositional embeddings.

FastText offers several advantages. It handles out-of-vocabulary words by composing n-gram embeddings even for unseen words (e.g., a misspelling like “reaktor” shares n-grams with “reactor”). It captures morphological relationships: “enrich,” “enriched,” and “enrichment” share many n-grams, yielding similar embeddings that reflect their semantic connection. This is particularly valuable in nuclear engineering, where technical terminology exhibits systematic morphological patterns (“fission/fissile/fissionable,” “moderation/moderator/moderated”).

However, n-gram embeddings increase memory requirements (the vocabulary effectively expands to include all observed n-grams) and can introduce noise if uninformative character sequences receive significant weight.

GloVe: Global Vectors for Word Representation global vectors for word representation (Glove) (Global Vectors) [12] bridges frequency- and prediction-based approaches by directly modeling word co-occurrence statistics. The key insight is that the ratio of co-occurrence probabilities encodes semantic relationships more effectively than raw probabilities.

Let X_{ij} denote the number of times word j appears in the context of word i , and define $X_i = \sum_k X_{ik}$ as the total count of words appearing in i 's context. The co-occurrence probability $P_{ij} = X_{ij}/X_i$ measures how likely word j appears near word i . GloVe observes that for two target words i and k and a probe word j , the ratio P_{ij}/P_{kj} is more discriminative than P_{ij} or P_{kj} alone. For example, if $i = \text{“steam”}$ and $k = \text{“turbine,”}$ then $j = \text{“condenser”}$ yields $P_{ij}/P_{kj} \approx 1$ (both are common in steam cycle contexts), whereas $j = \text{“neutron”}$ yields $P_{ij}/P_{kj} \ll 1$ (more associated with reactors than steam).

GloVe captures this intuition by optimizing a weighted least-squares objective:

$$\mathcal{L}_{\text{GloVe}} = \sum_{i,j=1}^{|V|} f(X_{ij}) (\mathbf{v}_i^\top \tilde{\mathbf{v}}_j + b_i + \tilde{b}_j - \log X_{ij})^2, \quad (12)$$

where $\mathbf{v}_i$ and $\tilde{\mathbf{v}}_j$ are word and context embeddings, b_i and $\tilde{b}_j$ are bias terms, and $f(X_{ij})$ is a weighting function that de-emphasizes rare co-occurrences:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max}, \\ 1 & \text{otherwise,} \end{cases} \quad (13)$$

with typical values $x_{\max} = 100$ and $\alpha = 0.75$. This weighting prevents extremely frequent pairs (e.g., “the” and “a”) from dominating the objective, while also avoiding overfitting to rare co-occurrences.

GloVe embeddings combine the statistical efficiency of matrix factorization methods (operating on pre-computed co-occurrence statistics) with the performance of neural prediction models. Empirically, GloVe and Word2Vec achieve comparable quality on semantic similarity and analogy tasks, though GloVe often trains faster and converges more reliably [12].

Contextualized Embeddings and Modern Architectures

Classical embeddings—Word2Vec, FastText, GloVe—assign each word a single fixed vector regardless of context. This is problematic because many words are polysemous: “plant” refers to flora in botanical contexts but to industrial facilities in engineering discourse; “critical” denotes a neutron multiplication factor of exactly 1.0 in reactor physics but severe urgency in safety protocols. Contextualized embeddings address this by generating word representations that depend on the surrounding sentence or document.

ELMo: Embeddings from Language Models ELMo (Embeddings from Language Models) [23] learns contextualized representations via deep bidirectional long short-term memorys (LSTMs). A forward LSTM processes the sentence left-to-right, encoding each token’s left context; a backward LSTM processes right-to-left, capturing right context. The final embedding for token t is a weighted combination of hidden states from all LSTM layers:

$$\mathbf{h}_t^{\text{ELMo}} = \gamma \sum_{j=0}^L s_j \mathbf{h}_{t,j}, \quad (14)$$

where L is the number of LSTM layers, $\mathbf{h}_{t,j}$ is the hidden state at layer j for token t , s_j are learned scalar weights (normalized via softmax), and γ is a scaling parameter. Layer 0 corresponds to character-level

convolutional neural network (CNN) embeddings; higher layers capture increasingly abstract contextual information.

ELMo demonstrated that contextualized embeddings significantly improve performance on tasks like named entity recognition, question answering, and sentiment analysis [23]. However, LSTMs scale poorly to long sequences (quadratic memory in sequence length) and cannot be efficiently parallelized during training.

Transformer-Based Embeddings: BERT, GPT, and Beyond The Transformer architecture [6] replaced LSTMs with self-attention mechanisms that process all tokens in parallel, enabling massive-scale pre-training. **BERT** (Bidirectional Encoder Representations from Transformers) [24] uses masked language modeling: randomly masking 15% of input tokens and training the model to predict them from bidirectional context. For a token t with embedding $\mathbf{h}_t$ from the final Transformer layer, BERT outputs:

$$p(w_t \mid \text{context}) = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}), \quad (15)$$

where $\mathbf{W} \in \mathbb{R}^{|V| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|V|}$ parameterize the output distribution.

BERT embeddings are deeply contextualized: the representation of “plant” in “nuclear power plant” differs markedly from “plant” in “botanical plant,” capturing polysemy naturally. This property makes BERT embeddings particularly effective for domain-specific applications in which terminology is overloaded with multiple technical meanings.

GPT (Generative Pre-trained Transformer) [25] and its successors adopt an autoregressive (left-to-right) language modeling objective, predicting the next token given all the previous tokens. While less suited for tasks requiring bidirectional context, GPT’s unidirectional architecture enables efficient text generation and has scaled to models with hundreds of billions of parameters.

Transformer-based embeddings—BERT, GPT, RoBERTa [26], ELECTRA [27], T5 [28]—have become the default choice for modern NLP. Their contextualized representations, combined with attention mechanisms that model long-range dependencies, enable unprecedented performance on language understanding and generation tasks.

Positional Embeddings

Transformers process input sequences in parallel via self-attention, which is inherently permutation-invariant: reordering tokens does not change attention weights. Yet word order is crucial for meaning: “reactor shutdown” differs fundamentally from “shutdown reactor.” Positional embeddings encode token positions, restoring sequence awareness.

Sinusoidal Positional Encoding The original Transformer [6] uses fixed sinusoidal functions to encode positions. For position t and dimension i (where i ranges from 0 to $d - 1$ for embedding dimension d), the positional encoding is:

$$\text{PE}(t, 2i) = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \text{PE}(t, 2i + 1) = \cos\left(\frac{t}{10000^{2i/d}}\right). \quad (16)$$

Even dimensions use sine; odd dimensions use cosine. The wavelength of the sinusoid increases exponentially with dimension, from 2π (dimension 0) to $10000 \cdot 2\pi$ (final dimension). This allows the model to learn relative positions via linear combinations: $\text{PE}(t + k)$ can be expressed as a linear transformation of $\text{PE}(t)$ for any offset k .

Sinusoidal encodings generalize to unseen sequence lengths (longer than those in the training set) because the functional form extends indefinitely. However, they do not adapt to task-specific positional patterns and may be suboptimal for highly structured inputs (e.g., source code with indentation semantics, tabular data with column order).

Learned Positional Embeddings An alternative is to treat positions as learnable parameters: each position t (up to some maximum length T) receives an embedding $\mathbf{p}_t \in \mathbb{R}^d$ initialized randomly and optimized via backpropagation [24]. The final token representation at position t combines word and positional embeddings:

$$\mathbf{e}_t = \mathbf{v}_{w_t} + \mathbf{p}_t, \quad (17)$$

where $\mathbf{v}_{w_t}$ is the word embedding. Learned positional embeddings can capture task-specific positional biases (e.g., subject-verb-object ordering in English) but do not extrapolate beyond T ; sequences longer than the training maximum require positional interpolation or extension strategies.

Relative Positional Encodings Recent architectures (Transformer-XL [29], DeBERTa [30]) encode relative rather than absolute positions. Instead of assigning position t a fixed embedding, these models represent the relationship between positions t_i and t_j as a learnable function $R(t_i - t_j)$ or bias term added to attention scores:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^\top + \mathbf{B}}{\sqrt{d_k}} \right) V, \quad (18)$$

where $\mathbf{B}_{ij}$ encodes the relative position $i - j$. Relative encodings improve generalization to longer sequences and capture local dependencies more effectively than absolute positions.

Rotary Positional Embeddings (RoPE) Rotary Positional Embeddings (RoPE) [31] apply rotation matrices to query and key vectors as a function of position, encoding relative positions geometrically. For position t , RoPE multiplies the embedding by a rotation matrix $\mathbf{R}_t$:

$$\mathbf{q}_t = \mathbf{R}_t \mathbf{q}, \quad \mathbf{k}_t = \mathbf{R}_t \mathbf{k}, \quad (19)$$

where $\mathbf{R}_t$ rotates different embedding dimensions at position-dependent frequencies (analogous to sinusoidal encodings but applied via rotation). The dot product $\mathbf{q}_t^\top \mathbf{k}_{t'}$ then depends only on the relative position $t - t'$, preserving permutation equivariance while encoding order.

RoPE is used in recent large-scale models (LLaMA [32], PaLM [33]), due to its efficiency, strong performance on long-context tasks, and favorable extrapolation properties.

Multimodal Embeddings

Industrial anomaly detection and nuclear engineering applications increasingly require integrating text with other modalities: thermal images from reactor inspections, vibration sensor time-series data, tabular operational parameters, and structured knowledge graphs. Multimodal embeddings project heterogeneous data types into a shared semantic space, enabling cross-modal retrieval, reasoning, and generation.

Vision-Language Embeddings: CLIP and ALIGN CLIP (Contrastive Language-Image Pre-training) [34] learns joint embeddings for images and text via contrastive learning (see Section 2.4.3). An image encoder (Vision Transformer or ResNet) maps images to $\mathbf{z}_{\text{img}} \in \mathbb{R}^d$; a text encoder (Transformer) maps

captions to $\mathbf{z}_{\text{text}} \in \mathbb{R}^d$. Training maximizes cosine similarity for matched (image, caption) pairs and minimizes it for mismatched pairs. The resulting embeddings enable zero-shot visual classification by encoding class names as text and computing similarities with image embeddings.

For nuclear applications, CLIP can match thermal inspection images with textual descriptions (“hotspot in primary coolant loop near steam generator”), enabling retrieval of similar historical incidents or automated tagging of inspection reports.

Audio and Time-Series Embeddings Acoustic monitoring of rotating machinery (pumps, turbines, generators) produces high-dimensional time-series data that benefit from learned embeddings. **Wav2Vec 2.0** [35] applies self-supervised learning to raw audio waveforms, masking portions of the signal and training a Transformer to reconstruct them. The resulting embeddings capture acoustic patterns—frequency modulation, spectral envelope, transient events—useful for anomaly detection (e.g., bearing wear, cavitation, flow-induced vibration).

Time-series foundation models like **Chronos** [36] and **TimeGPT** [37] tokenize continuous sensor readings via vector quantization (analogous to VQ-VAE for images; see Section 2.4.4) and learn embeddings optimized for forecasting and anomaly detection. These embeddings encode temporal patterns—trends, seasonality, regime shifts—that correlate with equipment degradation or operational transients.

Graph and Structured Data Embeddings Nuclear reactor systems are naturally represented as graphs: components (nodes) connected by coolant flow, electrical power, control signals (edges). **Graph Neural Networks (GNNs)** [38, 39] learn node embeddings that aggregate information from neighbors. For a node v with features $\mathbf{x}_v$ and neighbors $\mathcal{N}(v)$, a GNN layer computes:

$$\mathbf{h}_v^{(l+1)} = \sigma \left(\mathbf{W}^{(l)} \mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}(v)} \frac{1}{|\mathcal{N}(v)|} \mathbf{h}_u^{(l)} \right), \quad (20)$$

where $\mathbf{h}_v^{(l)}$ is the embedding at layer l , $\mathbf{W}^{(l)}$ is a learnable weight matrix, and σ is a nonlinear activation. Stacking multiple layers enables embeddings to capture multi-hop dependencies (e.g., a valve failure propagating through the coolant system to affect heat exchanger temperatures).

Graph embeddings have been applied to root cause analysis in complex systems: given an observed anomaly (e.g., elevated radiation in containment), GNNs propagate information through the reactor graph to identify probable fault origins (e.g., failed fuel cladding, coolant contamination) [40].

Unified Multimodal Embeddings Recent models like **ImageBind** [41] learn a single embedding space spanning six modalities: text, images, audio, depth, thermal, and IMU (inertial measurement) data. By training on naturally paired data (e.g., videos containing synchronized image, audio, and motion), ImageBind learns that semantically related inputs—regardless of modality—cluster together. This enables cross-modal queries: “find thermal scans similar to this vibration pattern” or “retrieve maintenance logs matching this acoustic signature.”

For nuclear operations, unified embeddings could integrate:

- **Visual inspections:** Thermal imaging, radiographic testing, visual surface examinations
- **Sensor telemetry:** Temperature, pressure, flow, neutron flux, radiation levels
- **Operational logs:** Shift reports, alarm histories, control room entries

- **Maintenance records:** Work orders, part replacements, downtime events
- **Regulatory documents:** Technical specifications, safety analyses, incident reports.

By embedding all these modalities into a shared space, LLMs can perform cross-modal reasoning: correlating sensor anomalies with historical maintenance patterns, retrieving relevant regulatory precedents for novel fault scenarios, or generating diagnostic hypotheses that integrate multimodal evidence.

Domain-Specific Embedding Challenges

While general-purpose embeddings (trained on web-scale text and image corpora) capture broad semantic knowledge, domain-specific applications face unique challenges:

Technical Vocabulary and Acronyms Nuclear engineering abounds with specialized terminology and acronyms—PWR, BWR, loss-of-coolant accident (LOCA), emergency core cooling system (ECCS), xenon transient, Doppler broadening, prompt critical—that rarely appear in general corpora. Pre-trained embeddings may represent these terms poorly or conflate them with unrelated general-domain meanings (e.g., “PWR” as an abbreviation for “power” in casual usage). Fine-tuning on domain-specific corpora (technical reports, Nuclear Regulatory Commission (NRC) documentation, reactor vendor manuals) adapts embeddings to encode correct technical semantics [42, 43].

Numerical and Unit-Aware Embeddings Engineering discourse includes extensive numerical data with physical units: “operating pressure of 15.5 MPa,” “enrichment level of 4.2% U-235,” “decay heat removal rate of 2.3 MW.” Standard embeddings treat numbers as arbitrary tokens, missing magnitude relationships (15.5 is close to 15.2 but far from 2.3) and unit semantics (MPa and psi measure the same quantity). Specialized embeddings—**NumBERT** [44], unit-aware transformers—encode numerical values as continuous scalars and units as structured attributes, improving performance on quantitative reasoning tasks.

Temporal and Causal Relationships Reactor transients unfold over time: a coolant leak leads to pressure drop, which triggers safety injection, resulting in reactor shutdown. Capturing these temporal and causal chains requires embeddings that encode event sequences and dependencies. Temporal transformers [45], causal graph embeddings [46], and narrative-aware language models learn to represent causality, enabling better fault diagnosis and root cause analysis.

The development of specialized embeddings for nuclear and industrial domains remains an active research area, with opportunities to integrate physics-informed constraints, safety-critical semantics, and multimodal sensor fusion into learned representations.

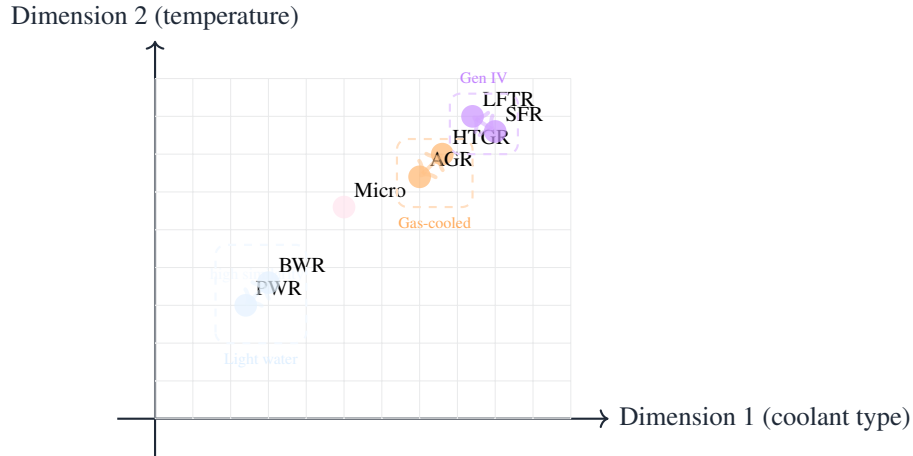


Figure 7: 2D projection of reactor type embeddings. Semantically similar reactor types cluster together: PWR and BWR (both light-water reactors), HTGR and AGR (gas-cooled), and SFR and LFTR (Generation IV designs). Vector proximity encodes multiple dimensions of similarity: coolant type, operating temperature, neutron spectrum, and safety philosophy.

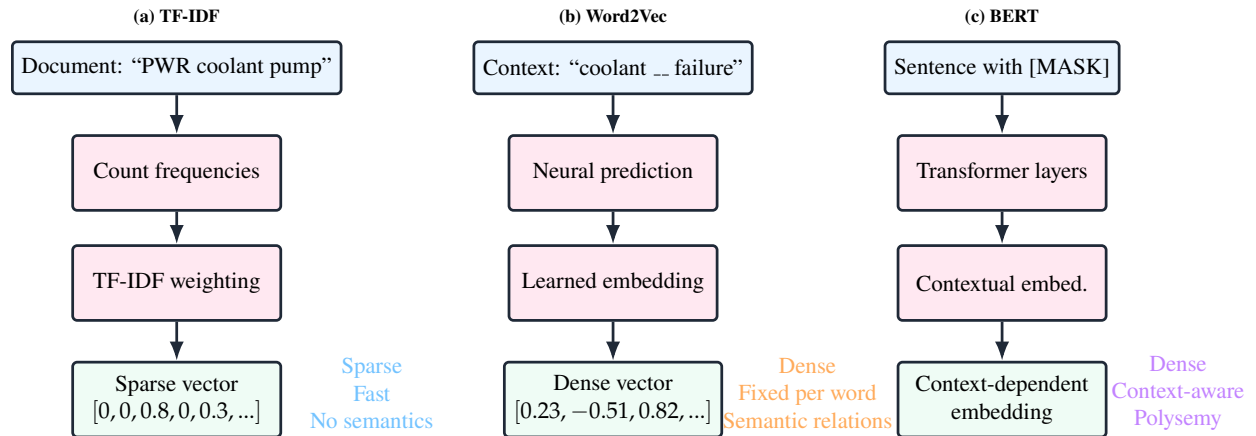


Figure 8: Comparison of embedding methodologies. (a) TF-IDF produces sparse, frequency-based vectors with no semantic awareness. (b) Word2Vec learns dense embeddings via neural prediction, capturing semantic similarity. (c) BERT generates contextualized embeddings that adapt to word usage, handling polysemy (“plant” as flora vs. facility).

2.2.3 Attention Mechanisms

The multilayer perceptron [47], convolutional neural networks (CNNs), and recurrent architecture are primarily responsible for the deep learning boom. For a long time, LSTM networks dominated NLP applications. However, at the present moment, most NLP tasks are based on the Transformer architecture. Attention is a mechanism that forms the backbone of the Transformer models. Bahdanau et al. [48] introduced attention as an improvement to the classic sequence-to-sequence model for language translation. The attention mechanism aims to create an alignment between source and target words, allowing the model to take all source words into account while predicting the target.

1. **Soft Attention:** Soft attention is a differentiable mechanism that allows models to focus on different parts of the input with varying degrees of emphasis. It assigns a weight to each input element, and these weights are learned during training. This allows for smooth and continuous focus on different parts of the

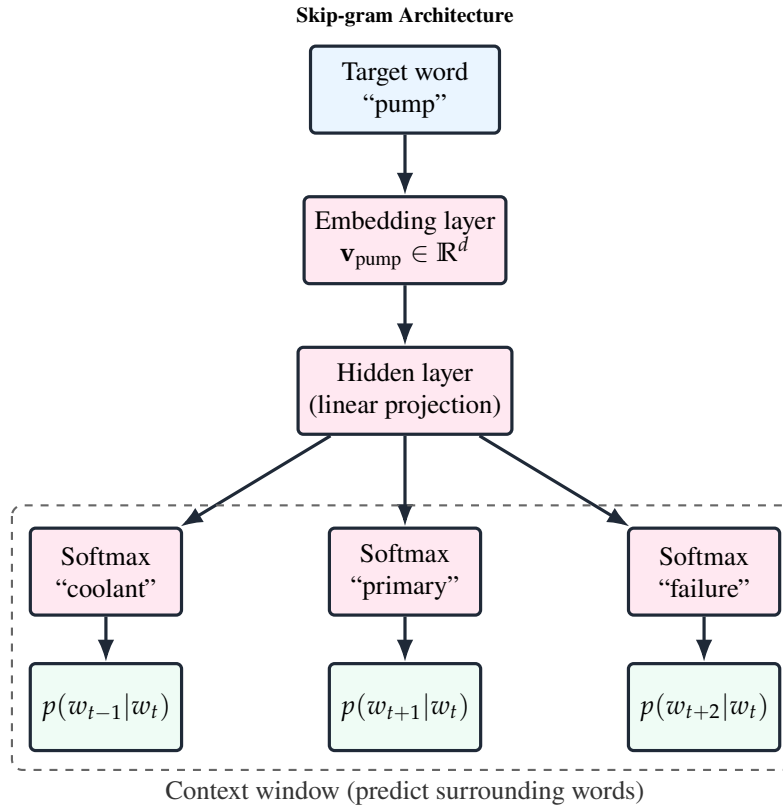


Figure 9: Skip-gram (Word2Vec) architecture. Given a target word (“pump”), the model predicts surrounding context words within a fixed window. The learned embedding $\mathbf{v}_{\text{pump}} \in \mathbb{R}^d$ encodes semantic relationships: words appearing in similar contexts receive similar embeddings.

input, without making hard decisions. Soft attention is widely used in tasks like machine translation and text summarization.

2. **Hard Attention:** Hard attention, unlike soft attention, involves making a hard, discrete decision about which parts of the input to focus on. It selects only one or a few specific elements of the input, rather than a weighted combination. This is generally non-differentiable, making it more challenging to train, as it requires reinforcement learning or other non-standard training methods.

3. **Self-Attention:** Self-attention allows a model to relate different positions within a single sequence to compute a representation of the sequence. It computes a score for each pair of positions in the input sequence, indicating their relevance to each other. This helps the model understand relationships and dependencies between different parts of the input. Self-attention is a core component of transformer models and is crucial for understanding long-range dependencies in text and other sequential data, according to DataCamp. Figure 13 depicts the general anatomy of Scaled Dot-Product Attention.

4. **Multi-Head Attention:** Multi-head attention (MHA) extends self-attention by using multiple sets of learned linear projections of the input. This allows the model to learn different relationships between input elements in different subspaces. It essentially performs self-attention multiple times in parallel, each with its own set of learned parameters. The outputs of these different heads are then combined (often by concatenation and a linear transformation) to produce the final output as shown in Figure 14. This allows the model to capture a richer set of relationships and dependencies within the data.

5. **Spatial Attention:** Spatial attention focuses on identifying important regions within an image. It helps

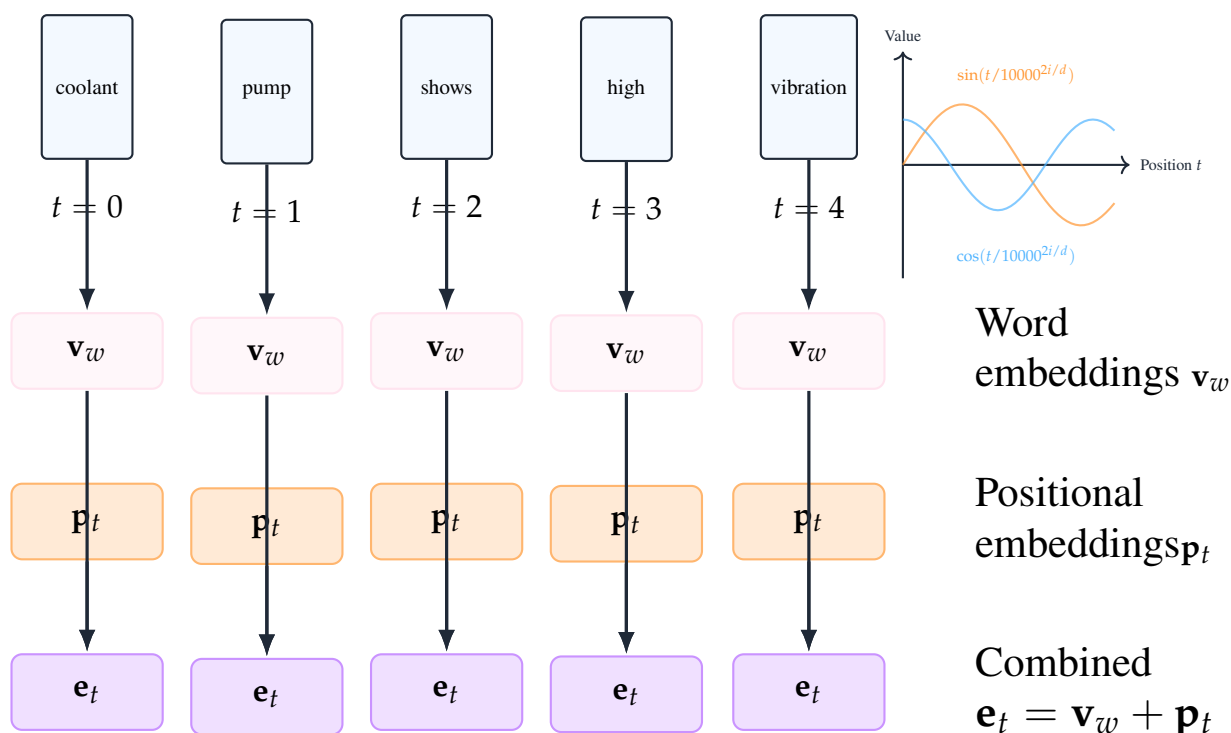


Figure 10: Positional embeddings in Transformers. Each token receives a word embedding $\mathbf{v}_w$ (capturing semantics) and a positional embedding $\mathbf{p}_t$ (encoding sequence position). The final representation $\mathbf{e}_t = \mathbf{v}_w + \mathbf{p}_t$ combines both. Right: sinusoidal positional encodings use different frequencies across embedding dimensions to uniquely encode each position.

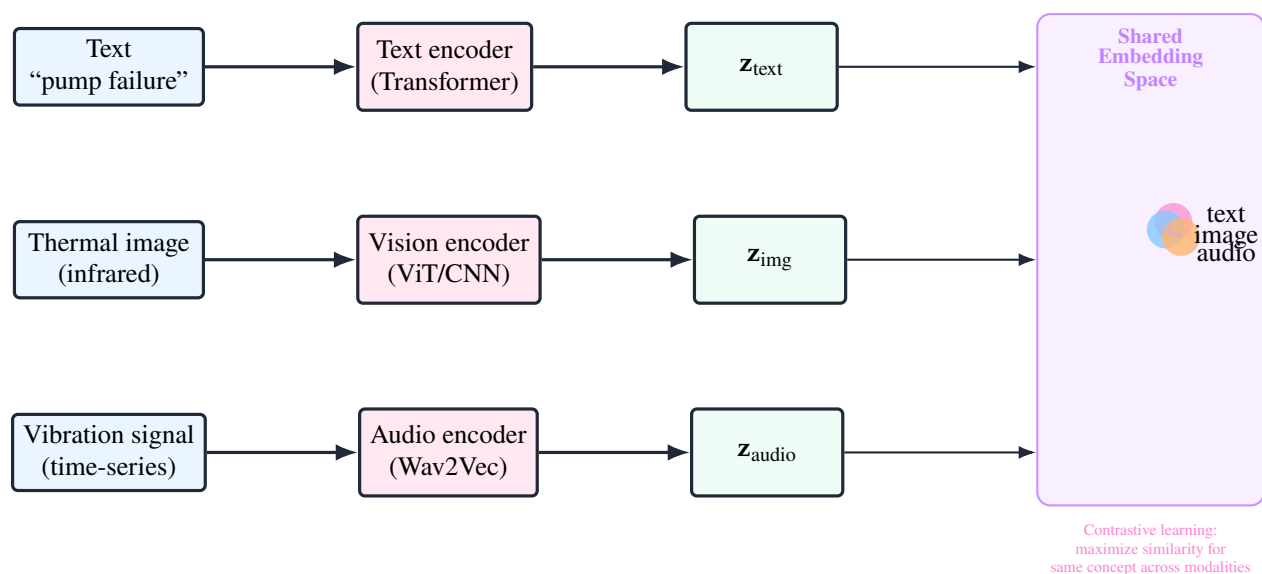


Figure 11: Multimodal embedding architecture. Different modalities (text, thermal images, vibration signals) are encoded by specialized neural networks into a shared embedding space. Contrastive learning ensures that semantically related inputs—regardless of modality—cluster together, enabling cross-modal retrieval (“find thermal images matching this acoustic signature”).

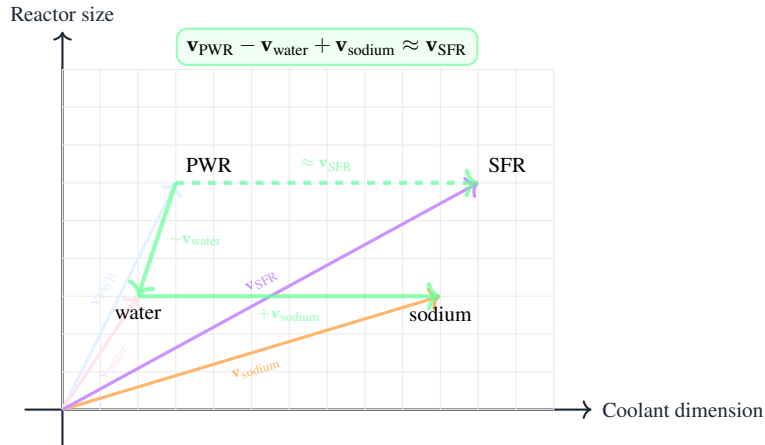


Figure 12: Word embedding analogies capture semantic relationships. The vector arithmetic $\mathbf{v}_{\text{PWR}} - \mathbf{v}_{\text{water}} + \mathbf{v}_{\text{sodium}} \approx \mathbf{v}_{\text{SFR}}$ demonstrates that embeddings encode conceptual transformations: replacing water coolant with sodium transforms a PWR into an SFR. This linear substructure enables analogical reasoning in the embedding space.

Scaled Dot-Product Attention

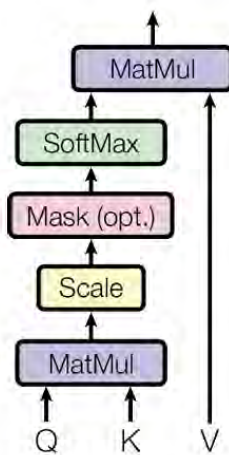


Figure 13: Self (scaled dot-attention) from the original paper: “Attention is All You Need.”[6]

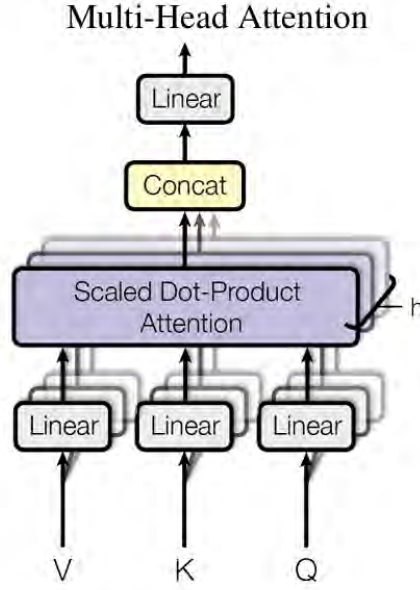


Figure 14: Multi-head attention from the original paper: “Attention is All You Need.”[6]

models attend to those specific locations in an image that are most relevant to the task at hand. For example, in image captioning, spatial attention can help the model focus on the objects being described.

6. **Channel Attention:** Channel attention focuses on identifying the most important feature channels within an image or feature map. It helps the model prioritize different feature channels, giving more weight to those that are most relevant to the task. For example, in object detection, channel attention can help the model focus on those feature channels that are most informative for object recognition.

7. **Temporal Attention:** Temporal attention is used for sequential data such as videos or time-series data. It helps the model focus on the most important time steps within the sequence. For example, in video classification, temporal attention can help the model focus on those time steps that are most relevant to the action being performed.

8. **Global Attention:** Global attention considers the entire input sequence when calculating attention weights. It is used when the entire input sequence is relevant to the task. 9. **Local Attention:** Local attention focuses on a limited neighborhood around each position in the input sequence. It is used when the relationships between nearby positions are most important.

Self-Attention

Definition. Self-attention allows every token in a sequence to attend to all the other tokens (including itself) in order to compute a context-aware representation. Given an input matrix $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$, it computes three projections:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learned parameter matrices.

Computation. Attention scores are obtained as:

$$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}},$$

optionally masked (for causal or padding restrictions), followed by a softmax to yield the normalized weights:

$$\mathbf{A} = \text{softmax}(\mathbf{S}).$$

The final output is the weighted sum of value vectors:

$$\mathbf{Y} = \mathbf{A}\mathbf{V}.$$

Interpretation. Each token dynamically gathers information from all other tokens in the same sequence, enabling modeling of long-range dependencies without recurrence. Self-attention is the foundational operation in both Transformer encoders and decoders. It is illustrated schematically in Figure 15, using light blue to indicate data/structure components and light pink to denote operations.

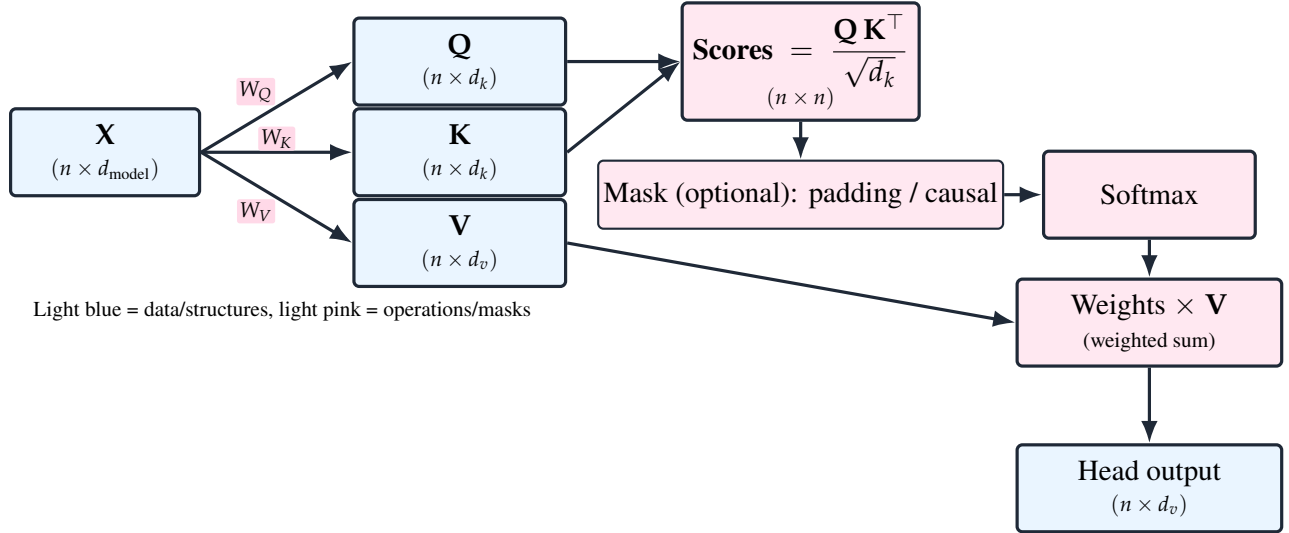


Figure 15: Scaled dot-product self-attention.

Multi-Head Attention

Motivation. A single self-attention mechanism captures only one type of relationship between tokens. To enrich representational capacity, Transformers use MHA, which runs multiple self-attention “heads” in parallel, each with its own learned projections.

Computation. For h heads, each head i performs:

$$\text{head}_i = \text{softmax} \left(\frac{\mathbf{Q}W_Q^{(i)}(\mathbf{K}W_K^{(i)})^\top}{\sqrt{d_k}} \right) \mathbf{V}W_V^{(i)}.$$

All head outputs are concatenated and linearly projected:

$$\mathbf{Y} = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W_O,$$

where W_O is a learned output projection.

Advantages. Different heads can focus on different types of relationships (e.g., syntactic, semantic, positional), enabling richer contextual representations. Variants include:

- *Multi-Query Attention (MQA)*: distinct $\mathbf{Q}$ per head but shared $\mathbf{K}$ and $\mathbf{V}$
- *Grouped-Query Attention (GQA)*: partial sharing of $\mathbf{K}$ and $\mathbf{V}$ within head groups.

Context. MHA forms the core of every Transformer block. Residual connections and layer normalization are applied around it in the standard encoder and decoder architectures. A schematic depiction of MHA is shown in Figure 16.

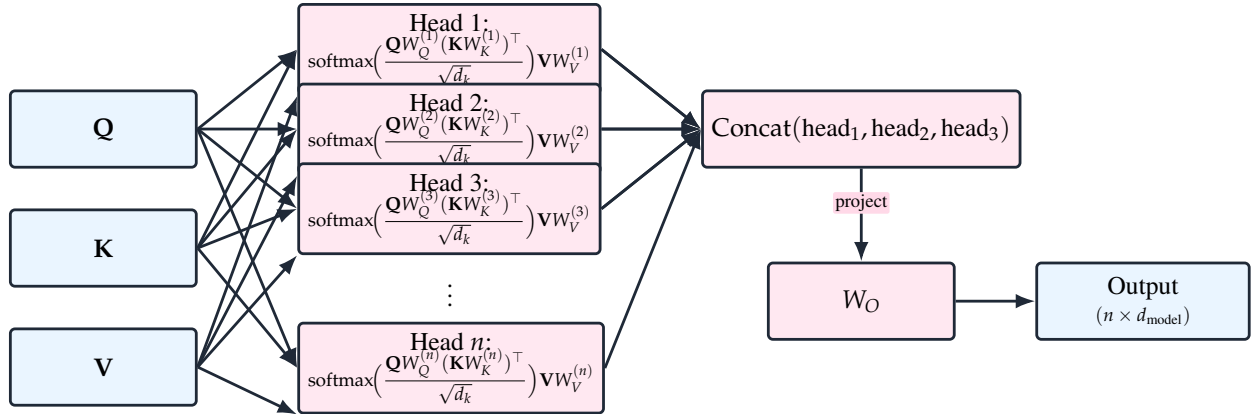


Figure 16: MHA: parallel heads, concatenation, and output projection.

Soft Attention (Scaled Dot-Product)

Setup. Given queries $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, keys $\mathbf{K} \in \mathbb{R}^{n \times d_k}$, and values $\mathbf{V} \in \mathbb{R}^{n \times d_v}$, the *soft (differentiable)* attention weights are obtained by:

$$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}, \quad \mathbf{A} = \text{softmax}(\mathbf{S} + \text{mask}),$$

where the (optional) mask handles padding or causal constraints. The output is a weighted sum:

$$\mathbf{Y} = \mathbf{A} \mathbf{V}.$$

Why it matters. Soft attention is fully differentiable end-to-end, allowing gradients to flow through both the scoring and weighting steps. It captures global token interactions (quadratic in sequence length) and is the core building block of Transformer encoders/decoders. See Figure 17 for a schematic.

Hard Attention (overview)

Idea. Instead of a soft, differentiable weight vector α (where $y = \sum_i \alpha_i V_i$), *hard attention* selects discrete positions (e.g., one index or top- k indices). A one-hot vector $\mathbf{z} \in \{0, 1\}^n$ with $\sum_i z_i = 1$ is sampled from a categorical distribution π over keys:

$$\pi = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right), \quad \mathbf{z} \sim \text{Categorical}(\pi), \quad y = \sum_i z_i V_i (= V_j \text{ if } z_j = 1).$$

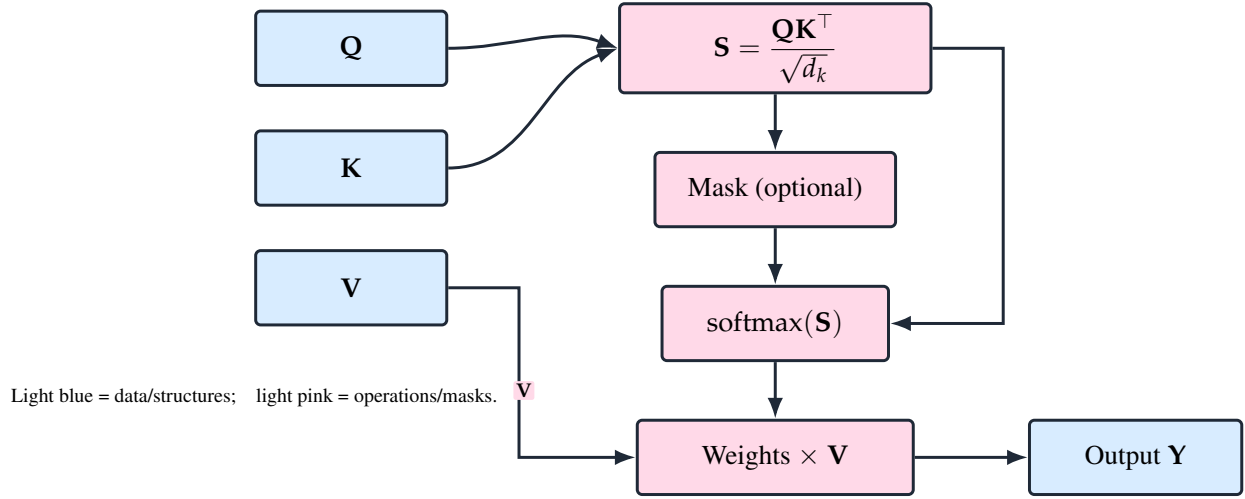


Figure 17: Soft attention (scaled dot-product self-attention).

Why use it. Computation can be *sparse* and *focused* (selecting a few “glimpses”), and the selections are often easier to interpret than dense mixtures.

Training challenge. The discrete sample is non-differentiable. Two common approaches:

- *REINFORCE / policy gradient* with a baseline: $\nabla_{\theta} \mathbb{E}_{\mathbf{z} \sim \pi_{\theta}} [L] = \mathbb{E}[(L - b) \nabla_{\theta} \log \pi_{\theta}(\mathbf{z})]$.
- *Straight-Through (ST) Gumbel-Softmax / Concrete*: Use a continuous relaxation for backprop while keeping a one-hot decision in the forward pass.

Variants. Single-index (“glimpse”), *top-k* hard attention (multi-hot $\mathbf{z}$ with $\sum_i z_i = k$), *monotonic* hard attention for streaming alignment, and pointer-style selection. A schematic depiction of hard attention is shown in Figure 18.

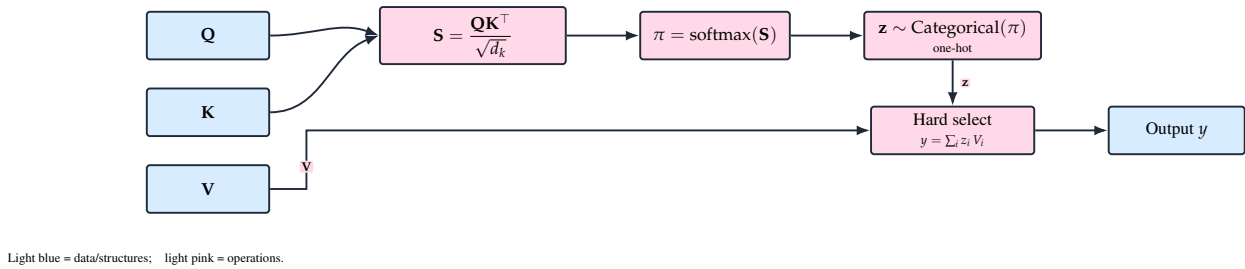


Figure 18: Hard attention

Spatial Attention

Concept. Spatial attention focuses on *where* to look within a feature map. Given $\mathbf{F} \in \mathbb{R}^{H \times W \times C}$, the mechanism highlights those spatial regions that are most informative for a downstream task (e.g., salient pixels in an image). Typically, feature descriptors across channels are aggregated to form spatial queries and keys, followed by attention over the $H \times W$ grid:

$$\mathbf{A}_{\text{spatial}} = \text{softmax}(\mathbf{Q}_s \mathbf{K}_s^{\top}), \quad \mathbf{F}' = \mathbf{A}_{\text{spatial}} \mathbf{V}_s.$$

Spatial attention is widely used in vision transformers and CNN enhancement blocks such as CBAM or BAM. (See Figure 19.)



Figure 19: Spatial attention learns 2D attention maps to emphasize salient spatial locations. The data/structures are shown by light blue, while light pink represents the operations.

Channel Attention

Concept. Channel attention emphasizes *what* to look at—identifying which feature channels are the most informative. It models inter-channel dependencies through global pooling and a small feed-forward network:

$$\mathbf{A}_{\text{ch}} = \sigma(W_2 \delta(W_1 \text{GAP}(\mathbf{F}))), \quad \mathbf{F}' = \mathbf{A}_{\text{ch}} \odot \mathbf{F}.$$

Here GAP is global average pooling, δ is ReLU, and σ is sigmoid. (See Figure 20.)



Figure 20: Channel attention: learns channel-wise weights to emphasize important feature channels.

Temporal Attention

Concept. Temporal attention focuses on *when* to attend, capturing dependencies across time steps in sequential or video data. Given $\mathbf{F} \in \mathbb{R}^{T \times D}$, each step's query interacts with all others:

$$\mathbf{A}_{\text{temp}} = \text{softmax}\left(\frac{\mathbf{Q}_t \mathbf{K}_t^\top}{\sqrt{d_t}}\right), \quad \mathbf{Y} = \mathbf{A}_{\text{temp}} \mathbf{V}_t.$$

Temporal attention allows for flexible, long-range temporal reasoning. (See Figure 21.)



Figure 21: Temporal attention: models dependencies across time steps in sequential data.

Global Attention

Concept. Global attention extends local mechanisms (spatial, channel, temporal) by allowing every element to attend to all others, often with learnable importance weights that balance local and global context. It provides holistic aggregation across the entire feature domain:

$$\mathbf{A}_{\text{global}} = \text{softmax}\left(\frac{\mathbf{Q} \mathbf{K}^\top}{\sqrt{d_k}}\right), \quad \mathbf{Y} = \mathbf{A}_{\text{global}} \mathbf{V}.$$

This structure is the general form underlying transformer-based global reasoning modules. (See Figure 22.)



Figure 22: Global attention: full-range attention across all spatial, channel, and temporal positions.

2.2.4 Finetuning

Pre-trained LLMs like GPT-4o [49], Llama [50], Claude have an impressive capacity to deal with general language tasks. To excel at specific tasks, such as steam generators used in different NPPs, control rod materials for BWRs or current status on microreactor projects, LLMs need to be “fine-tuned” on these domains. Fine-tuning is the process of taking a pre-trained LLM and training it further on smaller, domain- and task-specific datasets to help the model adapt its pre-existing knowledge to subtleties, relationships, and nuances of the target domain or task [51]. It represents a cost-effective solution, especially for smaller research groups and institutions that lack the massive computational resources required to train LLMs. There are multiple methods for fine-tuning LLMs. We discuss a few of them below, stating their advantages and drawbacks.

1. **Feature Extraction:** The approach is based on freezing the first $n - 1$ layers of an n -layered model and training just the final layer on the task-specific dataset. It offers a cost-effective solution that takes advantage of the representations learned by the LLMs $n - 1$ layers [52]. In this method, the LLM is treated as a feature extractor.
2. **Full Fine-Tuning:** This approach fine-tunes the entire model on a task- or domain-specific dataset, updating weights across all layers. It enables the model to learn new feature representations tailored to the target task, potentially delivering superior performance, albeit with substantial computational overhead.
3. **Supervised Fine-Tuning:** This critical process involves training the model on a task-specific dataset with labeled examples. The model is now trained in a fully supervised way, the goal being to minimize the negative log-likelihood (NLL) of the target sequence, conditioned on the input. Supervised fine-tuning of LLMs has shown notable zero-shot performance in various domains such as instruction following, dialogue systems, and data classification and formatting [53], [54].
4. **Instruction Fine-Tuning:** Instruction fine-tuning trains the model on a corpus of instruction-response pairs. This approach excels when the model must obey explicit directives or produce outputs in a prescribed format, making it especially valuable for instruction-following and format-constrained generation tasks [55]. “Flan” [56] models have been instruction fine-tuned on chain-of-thought data and can achieve strong few-shot performance in comparison to larger models (Flan-PaLM 540B vs. PaLM 540B).
5. **Parameter-Efficient Fine-Tuning (PEFT):** Inspired by the sensitivity of LLMs to prompts, introducing trainable prompt embeddings prepended to input text while freezing the original LLM parameters makes fine-tuning computationally efficient. Even when employing PEFT [57], the sheer size of the underlying LLM continues to pose practical challenges. One straightforward mitigation approach is post-training quantization (PTQ), though how PTQ interacts with task-specific parameters remains an active area of research [58]. Another technique, called LoRA (Low-Rank Adaptation), freezes the original model’s weights and adds a small, low-rank matrix to modify the model’s behavior so as to provide computationally efficient solutions for fine-tuning. QLoRA (Quantized Low-Rank Adaptation) [59] quantizes the base model to a lower precision (e.g., 4 bit) while the model is frozen and

LoRA is applied. This reduces the amount of memory needed to store the model, and LLMs can be fine-tuned on less-powerful hardware.

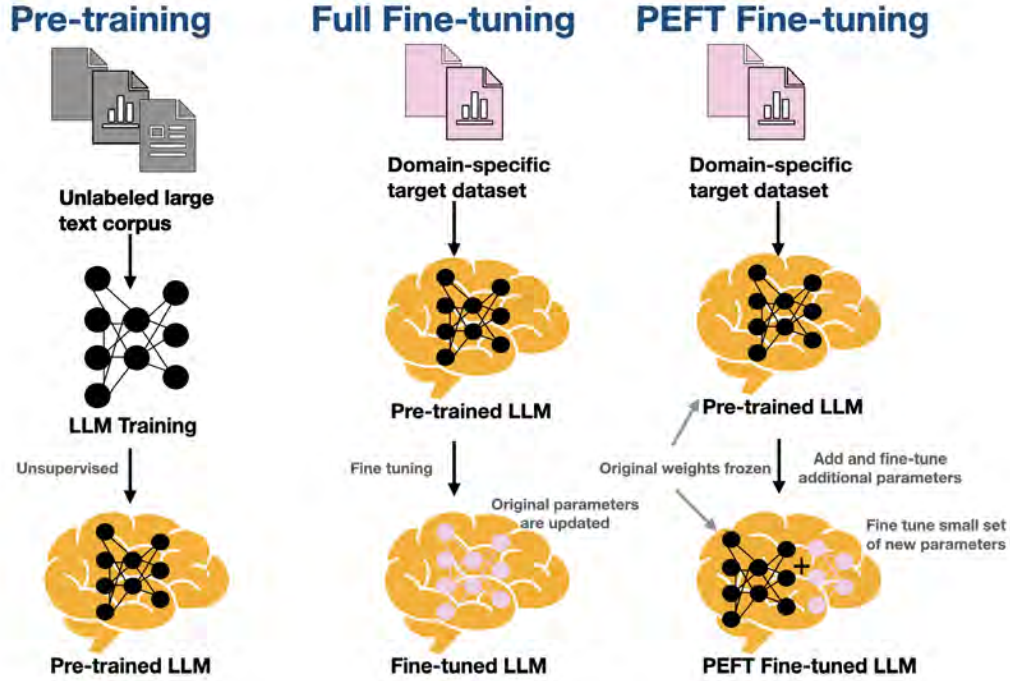


Figure 23: Comparison of model adaptation strategies: (1) Full pre-training of the LLM from scratch, (2) conventional fine-tuning that updates all model weights on a downstream task, and (3) PEFT approach that modifies only a small set of task-specific parameters while keeping the bulk of the pretrained weights frozen.

Fine-tuning an LLM with lightweight, low-rank adapters (or other parameter-efficient modules) captures task-specific knowledge while leaving the bulk of the pretrained weights untouched. This approach reduces memory and computational requirements and enables iteration across downstream tasks. By confining most of the adaptation to a small set of trainable parameters, low-rank adapters preserve the generalization power of the original model, make deployment on resource-constrained hardware feasible, and simplify model management—allowing a single base-LLM to support a diverse suite of applications through a handful of compact, task-specific add-ons, as seen in Figure 23.

In recent literature, multiple studies have performed fine-tuning on LLMs to perform anomaly detection. Jin et al. [60] used supervised fine-tuning on pre-trained LLMs such as BERT [24], GPT [25], and ALBERT [61] to predict anomalies on workflow data from FlowBench [62]. To mitigate the covariate shift between source and target domains for few-shot anomaly detection, Liao et al. [63] employed a contrastive training approach to fine-tune the model on domain data. Detection of anomalies through log data is an important consideration for IT operations. Guo et al. [64] propose a transformer-based framework for log anomaly detection, entailing a two-stage approach of pre-training and then adapter-based tuning for log data pertaining to variant domains. Li et al. [65] performed anomaly detection on tabular data by proposing an end-to-end fine-tuning strategy on large-scale anomaly detection benchmarks (ODDS) [66]. Another approach by Lim et al. [67] performs log anomaly detection (LAD) through PEFTs for adapting LLMs such as RoBERTa [26], GPT-2, and Llama-3 [50].

Thus, a range of models can be fine-tuned via the various aforementioned techniques for anomaly detection tasks. In the next section, we turn our attention to retrieval-augmented generation (RAG), which

supplies the LLM with domain-specific knowledge at inference time, eliminating the need for any additional fine-tuning.

2.2.5 Retrieval-Augmented Generation

This technique combines external data retrieval with LLMs to produce more accurate and relevant responses.

- **Retrieval:** When a user asks a question, the RAG system queries an external knowledge base, often a vector database, to find relevant information. This involves converting the user’s query into an “embedding” and searching for the most similar document embeddings.
- **Augmentation:** The retrieved information is then added to the original prompt, creating a new, more detailed prompt for the LLM.
- **Generation:** The LLM generates a response based on this augmented prompt, using the provided context to produce a more specific and factual answer.

RAG enhances the reliability of LLMs by grounding their responses in specific, retrieved documents, reducing the likelihood of the AI generating inaccurate or fabricated information (hallucinations). Additionally, RAG can access knowledge bases that are continually updated, providing the LLM with fresher, more current data than were available in its initial training set. On the other hand, the accuracy of RAG is heavily dependent on the quality of the external data it accesses. Errors or outdated information in the knowledge base can lead to inaccurate responses. Additionally, when conflicting information is present in the knowledge base, it can be challenging for the system to prioritize the correct data. There are two broad categories of RAG approaches:

1. In-context RAG
2. Integrated RAG

In-context RAG uses frozen LLMs with retrieved passages from documents and prepends them to the prompt at the inference time. The plug-and-play approach requires no model training and works in zero-shot settings. Allowing for rapid deployment with customer-designed databases. The integrated RAG trains or fine-tunes the retriever and generators in a single system, which better aligns the model to optimally retrieve evidence and results in higher accuracy at the cost of additional training. Lewis et al. [68] introduced RAG linking seq2seq models with a neural retriever, improving knowledge-intensive tasks without expanding the model’s parameters. Integrated RAG models like Retrieval-Augmented Language Model Pre-Training [69] pre-trains the retriever jointly with the language model, in an unsupervised manner. Overall, in-context RAG offers simplicity and zero-shot grounding, while integrated RAG provides optimized performance and deeper grounding at the expense of training complexity. For the use cases at commercial NPPs, in-context RAG is more flexible in coupling with proprietary databases and can be easily integrated with legacy systems. Depending on how relevant passages are searched, indexed for efficient lookup, and integrated into the LLM, in-context RAG methods can be further classified into sparse versus dense retrieval, inverted versus vector-based indexing.

Sparse retrieval, including TF-IDF, BM25, and query expansion methods, matches queries to documents through explicit or implicit term representations, while dense retrieval methods rely on deep semantic embeddings to map questions and text passages into the same vector space. TF-IDF uses term frequency and inverse document frequency to score each document based on the occurrence count of query terms scaled down by how common the term is overall. Implementation involves converting documents into high-dimensional sparse vectors of weighted term counts. A query’s TF-IDF vector is then compared via cosine

or dot-product to each document vector to rank results. Since its term-weighting scheme favors documents with many occurrences of rare query terms, it can skew results toward lengthy documents with repetitive but irrelevant content. An improved lexical ranking function uses term frequency saturations and document length normalizations. Each term’s contribution increases with frequency but with diminishing returns, and is thus down-weighted in longer documents, reducing skewed results by TF-IDF. However, both TF-IDF and BM25 rely on exact term matching and cannot retrieve information expressed with different vocabulary, and miss relevant context without direct term overlap [70]. Beyond direct term matching, sparse retrieval can expand the query with additional terms by using pseudo-relevance feedback [71]. The relevance model 3 (RM3) takes the top documents from an initial search and builds a relevance model of terms likely related to the query. These feedback terms are then interpolated with the original query to produce an expanded query. Implementation typically involves estimating a term probability distribution from top-ranked documents and adding the highest-weighted terms back into the query. Compared to sparse retrieval methods like TF-IDF and BM25, query expansion using RM3 significantly improves recall, particularly for ambiguous or underspecified queries. When applied in a RAG setting, RM3 provides the LLM with more complete context, leading to more effective and robust performance than standard retrieval approaches [72]. However, if the initial feedback set is noisy, RM3 may introduce irrelevant terms, potentially misleading the LLM. Therefore, careful parameter tuning—such as selecting the optimal number of expansion terms and setting appropriate interpolation weights—is essential to maximize performance.

Dense retrieval methods, such as dense passage retrieval (DPR) or sentence transformer, use neural embeddings to capture the semantic similarity between queries and documents. DPR is a dual-encoder model, in which a query encoder and a passage encoder are trained to map questions to text passages into the same vector space, such that relevant passages have a high inner-product with the query. At training, DPR trains two separate BERT encoders with a shared architecture but independent weights, optimizing a contrastive QA objective with negative sampling strategies. This makes DPR excel at domain-specific QA retrieval. Sentence transformer fine-tunes a single neural embedding to produce semantically aligned vectors for both query and passage. This makes sentence transformers, including SBERT and BGE, suitable for broad tasks and general retrieval—with better consistency and simple maintenance. At inference, both DPR and the sentence transformer compare a query’s embedding to a corpus of pre-computed passage embeddings by using an approximate nearest neighbor search. Compared to sparse retrieval methods, dense retrieval methods can find semantically relevant information even when words are different. In addition, dense retrieval methods can be further fine-tuned to a target domain by employing appropriate training data, aligning the vector space to that domain’s vocabulary and concepts.

Recursive retrievers act as a hybrid retrieval strategy by iteratively traversing explicit node references to gather both granular and holistic context. In the chunk-reference variant, input documents are nested into overlapping chunks linked to parent nodes, allowing for a fine-grained dense-vector lookup to be followed by pointer-based context walks for broader continuity. The metadata-reference variant instead attaches LLM-generated summaries and Q&A pairs as metadata atop source nodes, enabling recursive surfacing of both raw text and higher-level abstractions. A simpler standard (often referred to as “vanilla”) hybrid approach combines keyword-table (sparse) and vector-store (dense) results via logical “AND” or “OR” set operations to merge result sets with minimal overhead. Similar lexicon-enhanced hybrids can be found in LaPraDoR’s [73] Lexicon-Enhanced Dense Retrieval (LEDRE), which augments dense retrieval with BM25-style lexical matching to boost zero-shot performance on BEIR. Dense X Retrieval further refines this family by indexing at the atomic proposition level, showing that fine-granularity units significantly improve retrieval precision and downstream QA under fixed computational budgets, at the cost of increased preprocessing and ANN search complexity.

In addition to the classical RAG methods discussed above, several emerging RAG techniques, including

graphRAG and multimodal RAG, leverage richer structures and dynamic controls to improve RAG relevance, factuality, and efficiency.

Unlike conventional RAG, where semantic information can be uniformly represented as a 2D grid of image patches or a 1D sequence of textual corpora, one emerging technique, GraphRAG [74], encodes heterogeneous and relational information through its intrinsic “nodes connected by edges” structure. At the retrieval stage, the graph is traversed, or subgraphs are extracted based on the query relevance, and the resulting nodes’ text or corresponding community summaries are passed to the LLM for generation. Compared to conventional RAG, GraphRAG encompasses diverse formats of data, which are stored in heterogeneous sources. Moreover, since GraphRAG stores chunks as interconnected nodes with edges denoting their relations, the retrieval process enables multi-hop traversal to capture other chunks that share a logical connection with existing retrieved chunks. It is worth noting that relations in graph-structured data are domain-specific, and that the underlying data generation process governing the generated graphs can vary significantly across different domains. The graph construction, including entity and relation extraction, and subgraph retrieval can be computationally expensive.

Multimodal RAG [75] extends conventional, text-only RAG by retrieving and fusing evidence from multiple data types—text, images/plots, tables, audio/video, and time series—so the multimodal LLM can ground its outputs in cross-modal context rather than documents alone. In a typical pipeline, modality-specific or joint encoders map queries and candidates into one or more embedding spaces, a hybrid index then performs per-modality retrieval (e.g., text, vision, tabular), and the system fuses the results via late fusion with re-ranking or early fusion through cross-attention over mixed tokens prior to answer synthesis. Compared with conventional RAG, in which both queries and contexts are textual, multimodal RAG supports cross-modal queries such as describing a plot, screenshot, or diagram, and leverages non-text evidence that would otherwise be invisible to the model, thereby improving grounding and reducing hallucinations. In anomaly detection and diagnosis with work orders (text), metrics and traces (often visualized as charts), model analytics results (image), and user reports (natural language), multimodal RAG can retrieve similar past incidents from signatures in plots, log snippets, and analysis results in order to aid in triage and remediation. Recent results [76, 40, 77] show that multimodal LLMs can interpret time-series images effectively. Emerging multi-modal RCA workflows further demonstrate that combining structured retrieval with iterative reasoning can yield more actionable diagnostics than text-only pipelines.

2.3 Foundation model architectures.

Foundation models have revolutionized ML by demonstrating that models pre-trained on broad data at scale can be adapted to diverse downstream tasks with minimal task-specific modification. While these models share common principles—self-attention, large-scale pre-training, and transfer learning—they differ substantially in their architectural designs, each offering distinct advantages for different problem classes.

This section provides a taxonomy of the primary architectural families relevant to multimodal anomaly detection. We focus on four major paradigms: encoder-decoder architectures that excel at sequence-to-sequence transformations, decoder-only architectures optimized for generation, encoder-only architectures designed for representation learning, and diffusion models that offer probabilistic generative capabilities. Understanding these architectural distinctions is essential for selecting and adapting foundation models for anomaly detection, with the choice of architecture fundamentally shaping what types of anomalies can be detected and how.

2.3.1 Encoder-Decoder Overview

The encoder-decoder pattern remains the canonical blueprint for translating between two sequences: the encoder compresses an input $(x_1, \dots, x_n)$ into contextual representations, and the decoder autoregressively emits an output $(y_1, \dots, y_m)$ one token at a time [48, 78]. Transformer-based realizations replace recurrence with multi-head self-attention, allowing the model to simultaneously weigh all positions while preserving a clear gradient path for long-range dependencies.¹ The companion annotated implementation demystifies the architecture by exposing every tensor shape and residual path.² Figure 25 summarizes the data flow through a modern encoder-decoder stack.

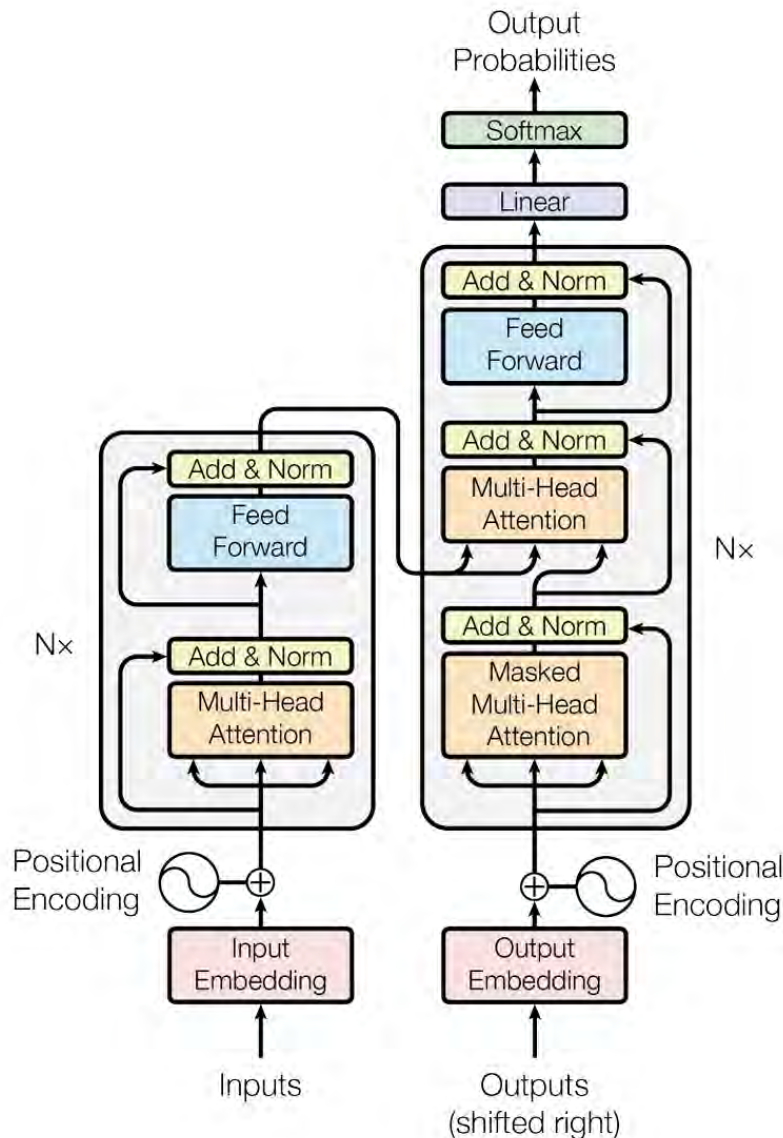


Figure 24: Transformer architecture from the original paper: “Attention is All You Need.”[6]

¹<https://nlp.seas.harvard.edu/annotated-transformer/>

²<https://github.com/harvardnlp/annotated-transformer/tree/master>

2.3.2 Encoder-Decoder Tokenization and Embeddings

Modern translation systems operate on open-vocabulary text by segmenting sentences into subword units with byte-pair encoding or unigram language-model tokenizers [7, 10]. The resulting one-hot matrix $\mathbf{X} \in \{0, 1\}^{n \times |\mathcal{V}|}$ indexes rows of a learned embedding table $\mathbf{W}_{\text{tok}}$; sinusoidal or learned positional codes $\mathbf{P} \in \mathbb{R}^{n \times d_{\text{model}}}$ provide absolute order signals. The encoder consumes the combined representation:

$$\mathbf{E}_x = \mathbf{X}\mathbf{W}_{\text{tok}} + \mathbf{P}. \quad (21)$$

Panel (a) of Figure 26 highlights the tokenizer pipeline, while panel (b) illustrates how token and positional embeddings are fused.

2.3.3 Encoder-Decoder Attention Stack

Each of the $\mathcal{N}$ encoder blocks applies multi-head self-attention followed by a position-wise feed-forward network, with residual connections [79], layer normalization [80], and dropout [81] after every sub-layer. Scaled dot-product attention computes a relevance score between queries Q , keys K , and values V that are linear projections of $\mathbf{E}_x$:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V. \quad (22)$$

Multi-head attention concatenates H these maps, $\text{MHA}(Q, K, V) = \text{Concat}(h_1, \dots, h_H)\mathbf{W}^O$ with $h_h = \text{Attention}(Q\mathbf{W}_h^Q, K\mathbf{W}_h^K, V\mathbf{W}_h^V)$, so different heads specialize to syntactic and semantic relations [6]. Panel (c) of Figure 26 expands this computation.

2.3.4 Encoder-Decoder Decoding and Alignment

The decoder mirrors the encoder with $\mathcal{M}$ blocks, but replaces self-attention with a causal variant that masks future positions ($t' > t$) so as to enforce autoregressive generation. Cross-attention injects the encoder’s memory by treating decoder queries as lookups against the source keys and values, enabling soft word alignments without explicit supervision. Teacher forcing exposes ground-truth prefixes during training, whereas inference alternates cross-attention and softmax sampling or beam search until an end-of-sequence token is produced.

2.3.5 Decoder-Only Overview

Decoder-only transformers such as GPT dispense with a separate encoder and operate on a single stream of prefix tokens. The model consumes an initial prompt, repeatedly predicts the distribution over the next token, and appends sampled outputs back into the context window. Figure 27 illustrates the data path for a single decoding step.

2.3.6 Decoder-Only Tokenizer and Positional Encodings

Byte-level tokenizers normalize prompts into a shared vocabulary that handles punctuation and domain-specific jargon uniformly. Token representations reuse the embedding matrix for both input lookup and output projection, while rotary or ALiBi positional schemes encode relative offsets without explicit sinusoidal tables. Panels (a) and (b) of Figure 28 summarize these stages.

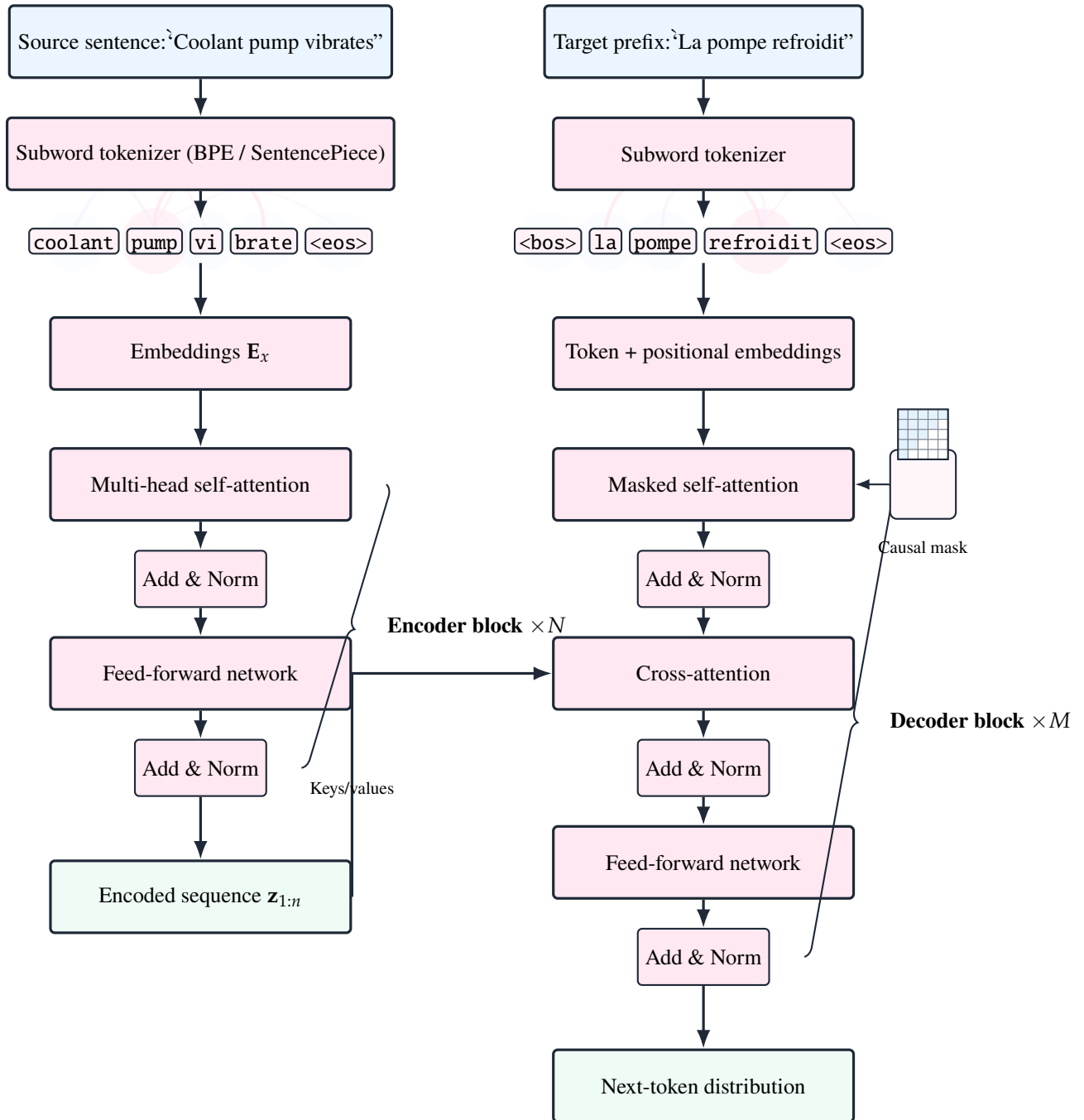


Figure 25: Transformer encoder-decoder translation pipeline. Light-blue nodes denote input representations, light-green nodes capture outputs that feed subsequent modules, and light-pink nodes highlight transformational operations; the flowing attention strands mirror how a query distributes weight across the sequence.

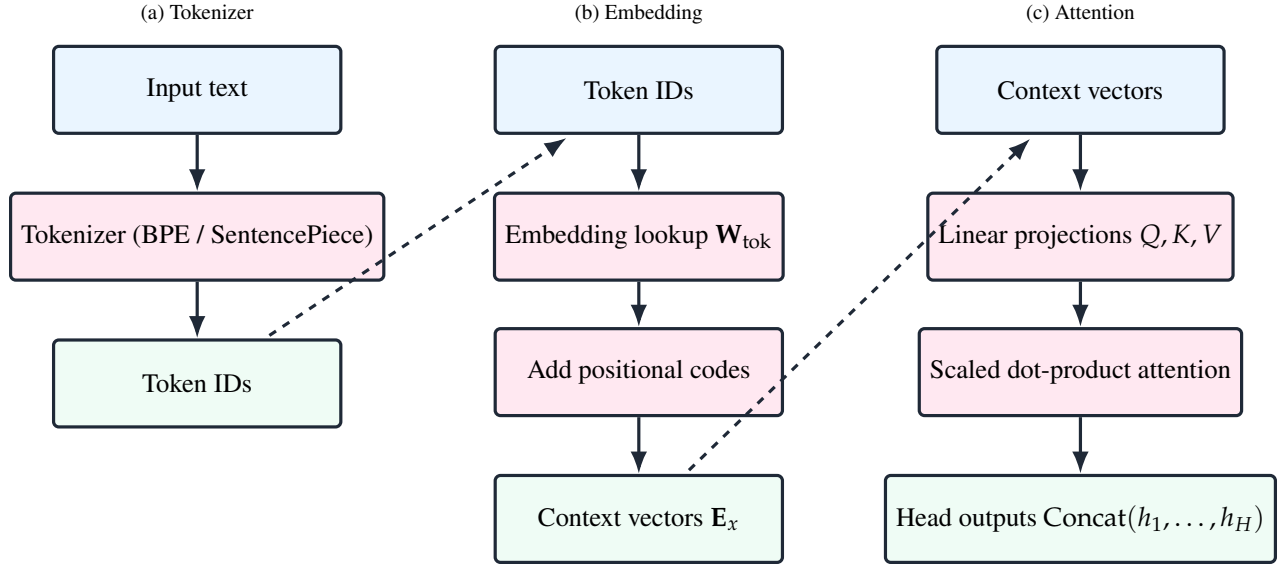


Figure 26: Tokenizer, embedding, and attention components within a transformer encoder-decoder system. Dashed arrows indicate how outputs from one component feed the next stage of the stack.

2.3.7 Decoder-Only Attention and Masking

Masked self-attention introduces a triangular bias matrix $\mathbf{M}$ whose strictly upper-triangular entries are $-\infty$, ensuring that the softmax excludes future tokens. Each block therefore conditions token x_t on $x_{<t}$ only, while residual connections and layer normalization preserve gradient flow. During inference, nucleus or top- k sampling prunes the high-entropy tail of the softmax distribution to trade diversity against factual precision. Panel (c) of Figure 28 highlights how the causal mask modulates attention weights.

2.3.8 Decoder-Only Output Head

The final hidden state feeds a linear projection tied to the embedding matrix and a softmax to produce the next-token distribution. Tokens sampled (or selected via beam search) are appended to the context and cached key-value tensors, enabling long-form generation without recomputing earlier activations [25].

2.3.9 Stable Diffusion Overview

Latent diffusion models reinterpret generative modeling as iteratively denoising a latent variable that has been gradually corrupted with Gaussian noise. The forward process produces a Markov chain,

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\sqrt{1 - \beta_t} \mathbf{z}_{t-1}, \beta_t \mathbf{I}), \quad t = 1, \dots, T, \quad (23)$$

so that $\mathbf{z}_T$ becomes nearly isotropic noise. The neural network ϵ_θ learns to predict the injected noise given a noised sample $\mathbf{z}_t$, the timestep t , and a conditioning vector c (e.g., a text embedding). The simplified objective,

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{t, \mathbf{z}_0, \epsilon} [\|\epsilon - \epsilon_\theta(\mathbf{z}_t, t, c)\|_2^2], \quad (24)$$

encourages the model to reverse the diffusion by sequentially subtracting predicted noise samples [82, 83]. Figure 29 provides an end-to-end view of this process.

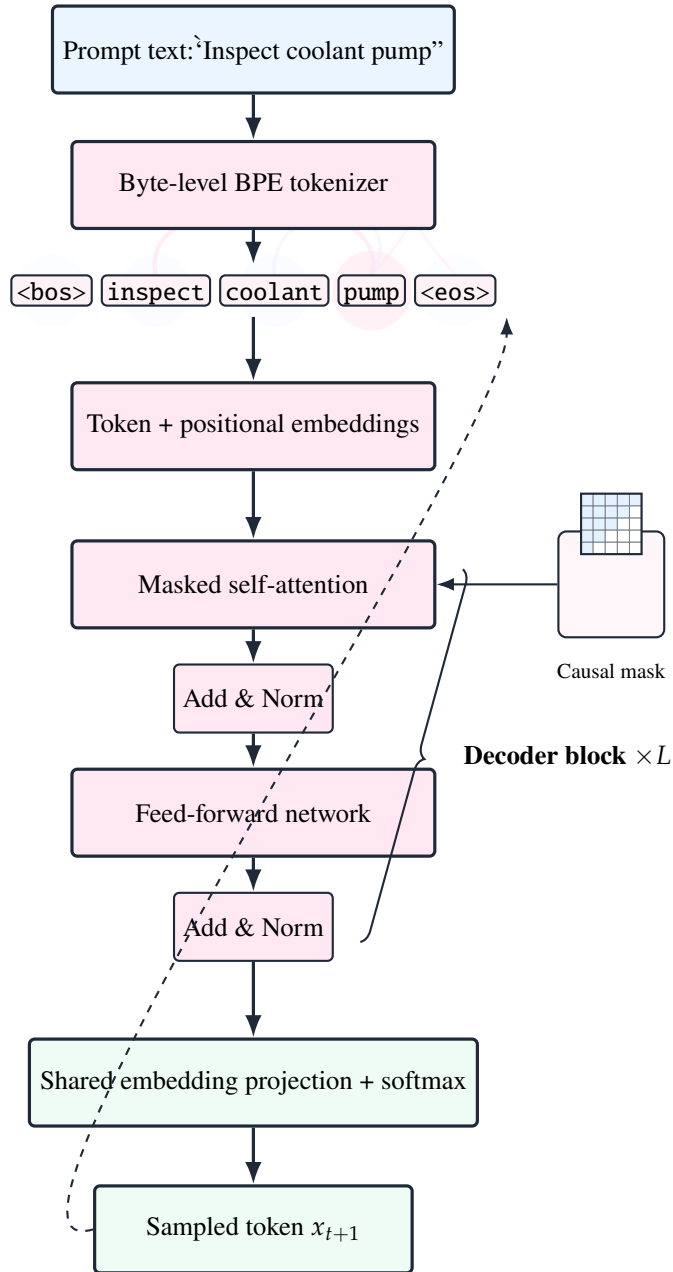


Figure 27: Decoder-only transformer step highlighting the causal attention block and its masked dependency on observed tokens. The curved attention ribbons emphasize the active query focus.

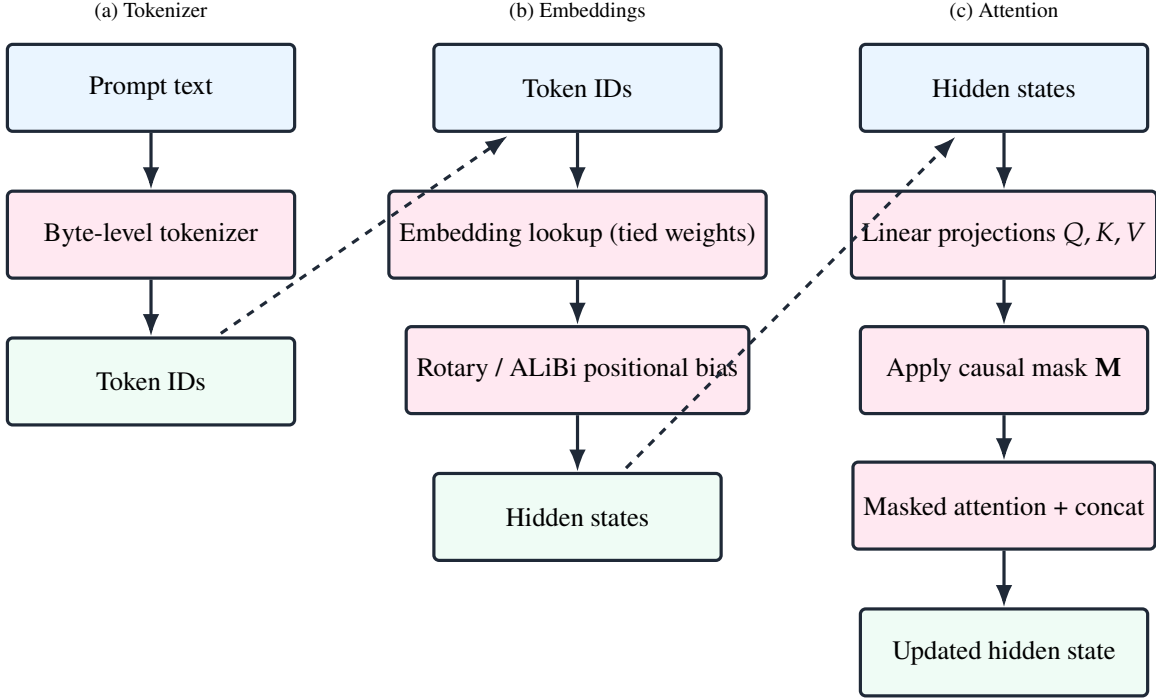


Figure 28: Tokenizer, embedding, and causal attention subcomponents used in decoder-only transformers.

2.3.10 Stable Diffusion Tokenizer and Conditioning

Stable Diffusion relies on the CLIP byte-level tokenizer to project prompts into a shared multimodal vocabulary [34]. After embedding tokens and adding positional codes, a 12-layer transformer produces context vectors $\mathbf{C} \in \mathbb{R}^{L \times d_c}$ that summarize the prompt semantics and are cached for every denoising step. Panels (a) and (b) of Figure 30 summarize this conditioning pathway.

2.3.11 Stable Diffusion Latent Attention

At timestep t , the latent U-Net receives $\mathbf{z}_t$, a sinusoidal timestep embedding, and the text context. Cross-attention modules inserted in the U-Net’s down- and up-sampling blocks compute the following:

$$\text{CrossAttn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V, \quad (25)$$

where Q originates from the latent feature maps and (K, V) are linear projections of $\mathbf{C}$. This mechanism allows spatial locations in the latent to focus on the relevant words in the prompt. Panel (c) of Figure 30 highlights the fusion of latent features and text embeddings.

2.3.12 Stable Diffusion Sampling and Decoding

Classifier-free guidance extrapolates between conditional and unconditional predictions:

$$\hat{\epsilon} = (1 + w) \epsilon_\theta(\mathbf{z}_t, t, c) - w \epsilon_\theta(\mathbf{z}_t, t, \emptyset), \quad (26)$$

where w controls adherence to the prompt [83]. Deterministic samplers such as Denoising Diffusion Implicit Models (DDIM) traverse the reverse process in as few as 20–50 steps while still preserving perceptual quality [84]. The final latent $\mathbf{z}_0$ is decoded by the VAE decoder to produce a high-resolution image.

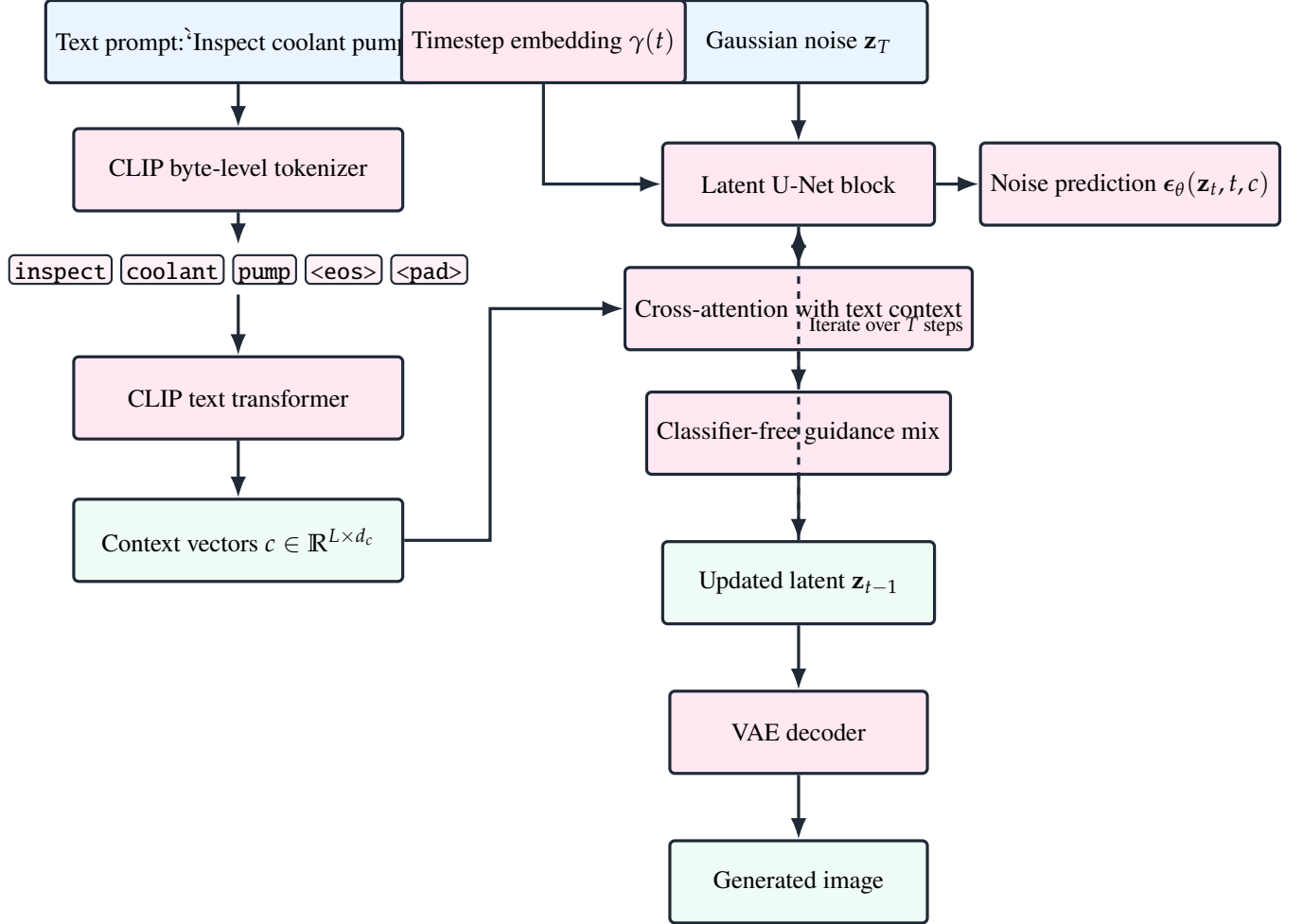


Figure 29: Stable Diffusion pipeline showing text conditioning, latent denoising, and image synthesis. Inputs appear in light blue, latent outputs in light green, and detailed operations in light pink.

2.4 Multimodal Integration Approaches

Having established the fundamental architectural patterns, we now examine how these models integrate information across heterogeneous modalities—a capability essential for anomaly detection in complex systems where text logs, sensor data, thermal imagery, and operational parameters must be jointly analyzed.

The central challenge in multimodal learning is determining when and how to combine information from disparate sources. A thermal camera might reveal hotspots in a reactor coolant system, while vibration sensors detect mechanical stress, and maintenance logs document recent component replacements. Each modality provides complementary evidence, but they differ fundamentally in their representational structure: images are spatial arrays of pixel intensities, time-series are sequential numerical measurements, and text consists of discrete symbolic tokens. Effective multimodal integration must preserve modality-specific patterns while discovering cross-modal correlations that individual modalities cannot capture.

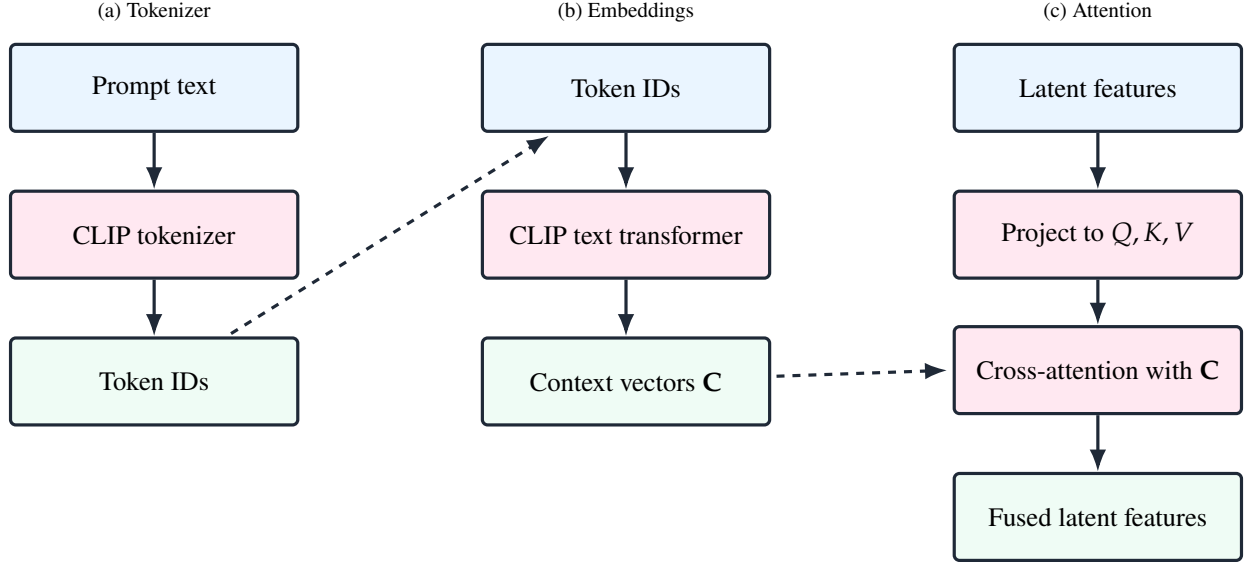


Figure 30: Tokenizer, conditioning embeddings, and cross-attention components within the Stable Diffusion architecture.

This section examines four foundational strategies for multimodal integration: temporal fusion strategies that determine when modalities are combined, cross-modal attention mechanisms that enable selective information exchange, contrastive learning objectives that align modality-specific representations, and unified tokenization schemes that project all modalities into a common representational substrate.

2.4.1 Early Fusion vs. Late Fusion

Multimodal fusion architectures differ fundamentally in terms of when they combine information from different sources. This timing decision shapes what kinds of cross-modal interactions the model can learn and how efficiently it processes heterogeneous inputs.

Early Fusion Architectures Early fusion (also called input-level fusion or feature-level fusion) combines modality-specific representations at the earliest possible stage, feeding the concatenated or merged features into a unified processing backbone [85, 86]. Consider a scenario in which an operator must diagnose pump failure by using both acoustic signatures and thermal imaging. An early fusion system would extract low-level features from each modality—spectrograms from audio, edge maps from thermal images—and concatenate them into a single feature vector that feeds a shared neural network. This approach allows the model to learn joint representations from the outset, discovering correlations between, for instance, specific frequency components in vibration data and temperature gradients in bearings.

Formally, let $\mathbf{x}_A \in \mathbb{R}^{d_A}$ and $\mathbf{x}_B \in \mathbb{R}^{d_B}$ denote feature vectors from modalities A and B . Early fusion constructs a joint representation via concatenation or projection:

$$\mathbf{h}_{\text{joint}} = f_{\theta}([\mathbf{x}_A; \mathbf{x}_B]), \quad (27)$$

where $[\cdot; \cdot]$ denotes concatenation and f_{θ} is a shared encoder with parameters θ . Alternatively, modality-specific projections can map inputs to a common dimensionality before fusion:

$$\mathbf{h}_{\text{joint}} = f_{\theta}(\mathbf{W}_A \mathbf{x}_A + \mathbf{W}_B \mathbf{x}_B), \quad (28)$$

where $\mathbf{W}_A \in \mathbb{R}^{d_{\text{hidden}} \times d_A}$ and $\mathbf{W}_B \in \mathbb{R}^{d_{\text{hidden}} \times d_B}$ are learned projection matrices.

The advantage of early fusion is its capacity to model fine-grained interactions: the shared encoder can learn that a particular thermal pattern is anomalous only when accompanied by a specific vibration signature. However, early fusion requires that all modalities be available at inference time and can struggle when modalities arrive asynchronously or have vastly different intrinsic dimensionalities. Training complexity also increases, as the model must learn joint statistics over the combinatorial space of multimodal features [85].

Late Fusion Architectures Late fusion (decision-level fusion) processes each modality through independent, specialized encoders and combines their outputs only at the final decision stage [87, 88]. Returning to the pump diagnosis example, a late fusion system would train separate neural networks—one analyzing acoustic data, another processing thermal imagery—and combine their anomaly scores or predictions through a learned aggregation function or simple voting mechanism.

Let $f_A(\mathbf{x}_A; \theta_A)$ and $f_B(\mathbf{x}_B; \theta_B)$ denote modality-specific encoders producing outputs $\mathbf{o}_A$ and $\mathbf{o}_B$. Late fusion combines these via:

$$\mathbf{o}_{\text{final}} = g(\mathbf{o}_A, \mathbf{o}_B), \quad (29)$$

where g may be a weighted average, learned MLP, or maximum operation. For anomaly detection, a common formulation computes per-modality anomaly scores $s_A = \|\mathbf{x}_A - \hat{\mathbf{x}}_A\|$ (reconstruction error) and $s_B = \|\mathbf{x}_B - \hat{\mathbf{x}}_B\|$, then fuses them as:

$$s_{\text{anomaly}} = \alpha \cdot s_A + (1 - \alpha) \cdot s_B, \quad (30)$$

with $\alpha \in [0, 1]$ tuned via cross-validation or learned from validation data.

Late fusion offers several practical advantages. Modality-specific encoders can be pre-trained independently, leveraging large unimodal datasets before multimodal integration. The architecture gracefully handles missing modalities: if thermal imagery is unavailable, the system defaults to the acoustic encoder’s output. Computational costs are lower during development, as each encoder can be optimized separately. However, late fusion sacrifices the ability to model low-level cross-modal interactions, potentially missing subtle correlations that manifest only when modalities are jointly analyzed [89].

Intermediate Fusion and Hybrid Strategies Modern architectures increasingly adopt intermediate fusion (mid-level fusion), which exchanges information between modality-specific encoders at one or more intermediate layers [90, 91]. This approach balances the complementary strengths of early and late fusion: modalities are initially processed independently to extract domain-specific features, then fused at a mid-level abstraction where both low-level patterns and high-level semantics can interact.

Intermediate fusion is often implemented via cross-modal attention mechanisms (discussed in Section 2.4.2) or gating functions that modulate one modality’s features based on another’s. For instance, a thermal imaging encoder might adjust its attention weights over spatial locations based on peaks detected in vibration sensor data, focusing on regions where both modalities signal potential anomalies.

Hierarchical fusion strategies employ multiple fusion points: early fusion of tightly coupled modalities (e.g., Red Green Blue (RGB) and depth images from co-located cameras), followed by late fusion with asynchronously collected text logs. The choice of fusion strategy depends on the temporal alignment of modalities, their relative information content, and whether the task benefits more from fine-grained cross-modal feature interactions or robust handling of missing inputs [92].

2.4.2 Cross-Modal Attention

Cross-modal attention mechanisms enable models to dynamically focus on relevant portions of one modality based on queries derived from another, implementing a learnable, context-dependent fusion strategy that adapts to each input instance [93, 94, 95].

Mechanics of Cross-Modal Attention Recall that self-attention (Section 2.3) computes attention weights within a single modality: queries, keys, and values all originate from the same input sequence. Cross-modal attention generalizes this by sourcing queries from one modality and keys/values from another. Consider a vision-language model analyzing an infrared image of a turbine alongside a maintenance log entry. The model must determine which words in the log (“bearing,” “vibration,” “overheating”) correspond to which regions in the image.

Formally, let $\mathbf{H}_A = [\mathbf{h}_1^A, \dots, \mathbf{h}_{n_A}^A] \in \mathbb{R}^{n_A \times d}$ represent encoded features from modality A (e.g., image patch embeddings) and $\mathbf{H}_B = [\mathbf{h}_1^B, \dots, \mathbf{h}_{n_B}^B] \in \mathbb{R}^{n_B \times d}$ from modality B (e.g., text token embeddings). Cross-attention from A to B computes:

$$\mathbf{Q}_A = \mathbf{H}_A \mathbf{W}^Q, \quad \mathbf{K}_B = \mathbf{H}_B \mathbf{W}^K, \quad \mathbf{V}_B = \mathbf{H}_B \mathbf{W}^V, \quad (31)$$

$$\mathbf{A}_{A \rightarrow B} = \text{softmax} \left(\frac{\mathbf{Q}_A \mathbf{K}_B^\top}{\sqrt{d_k}} \right), \quad (32)$$

$$\mathbf{H}_A^{\text{updated}} = \mathbf{A}_{A \rightarrow B} \mathbf{V}_B, \quad (33)$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d_k}$ are learned projection matrices. The attention matrix $\mathbf{A}_{A \rightarrow B} \in \mathbb{R}^{n_A \times n_B}$ assigns a weight to each token in modality B for every element in modality A . Element $\mathbf{A}_{ij}$ indicates how much the i -th image patch should attend to the j -th word in the log.

Importantly, cross-attention is asymmetric: attending from A to B differs from attending from B to A . Bidirectional cross-attention applies both operations:

$$\mathbf{H}_A^{\text{updated}} = \mathbf{H}_A + \text{CrossAttn}(\mathbf{H}_A, \mathbf{H}_B), \quad (34)$$

$$\mathbf{H}_B^{\text{updated}} = \mathbf{H}_B + \text{CrossAttn}(\mathbf{H}_B, \mathbf{H}_A), \quad (35)$$

allowing each modality to condition on the other in a mutually reinforcing manner. This bidirectional exchange is central to models like ViLBERT and LXMERT, which interleave self-attention within each modality with cross-attention between modalities [93, 94].

Application to Anomaly Detection Cross-modal attention excels in scenarios where anomalies manifest differently across modalities but require joint interpretation. For instance, a slight temperature increase in a heat exchanger (visible in thermal imaging) may be benign if the coolant flow rates are stable (recorded in time-series logs), but anomalous if accompanied by a maintenance note mentioning reduced coolant pressure. Cross-attention enables the model to retrieve the relevant text span (“reduced coolant pressure”) when analyzing the thermal anomaly, combining evidence that neither modality provides in isolation.

In practice, cross-modal attention can be visualized to explain model decisions. High attention weights $\mathbf{A}_{ij}$ between a thermal hotspot and the word “overheating” in a log provide interpretable evidence for the anomaly classification. This transparency is particularly valuable in safety-critical applications, where operators must understand and validate AI-driven alerts [96].

Co-Attention and Guided Attention Variants of cross-modal attention include co-attention, which simultaneously computes attention from A to B and B to A , then fuses the results via element-wise summation or gating [97]. Guided attention biases attention weights based on external knowledge: for instance, if domain expertise indicates that vibration sensors are more reliable than temperature probes for a specific fault mode, the attention mechanism can be regularized to prioritize vibration features [98].

Graph-based cross-modal attention extends the mechanism to structured data, where nodes represent entities (sensors, components, subsystems) and edges encode relationships. Attention operates over both intra-modal edges (e.g., spatial proximity of sensors) and inter-modal edges (e.g., a sensor co-located with a camera), enabling the model to reason about complex system topologies [99].

2.4.3 Contrastive Learning

Contrastive learning is a self-supervised paradigm that learns representations by maximizing agreement between related (positive) examples while pushing apart unrelated (negative) examples in the embedding space [100, 101, 102]. Applied to multimodal data, contrastive learning aligns representations from different modalities without requiring fine-grained labels, enabling models to discover cross-modal correspondences from large-scale, weakly-paired data.

The Contrastive Objective The fundamental intuition behind contrastive learning is that semantically similar inputs should have similar representations, while dissimilar inputs should be separated. For a batch of N multimodal pairs $\{(\mathbf{x}_i^A, \mathbf{x}_i^B)\}_{i=1}^N$, where each pair shares semantic content (e.g., an image of a component and its textual description), contrastive learning encodes both modalities into a shared embedding space and defines a loss that pulls paired examples together while pushing unpaired examples apart.

Let f_A and f_B denote encoders for modalities A and B , producing normalized embeddings $\mathbf{z}_i^A = f_A(\mathbf{x}_i^A) / \|f_A(\mathbf{x}_i^A)\|$ and $\mathbf{z}_i^B = f_B(\mathbf{x}_i^B) / \|f_B(\mathbf{x}_i^B)\|$. The InfoNCE loss (used in CLIP and similar models) treats each pair $(\mathbf{z}_i^A, \mathbf{z}_i^B)$ as a positive example and all other embeddings in the batch as negatives:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(\mathbf{z}_i^A \cdot \mathbf{z}_i^B / \tau)}{\sum_{j=1}^N \exp(\mathbf{z}_i^A \cdot \mathbf{z}_j^B / \tau)}, \quad (36)$$

where τ is a temperature parameter controlling the sharpness of the distribution [103, 34]. Minimizing this loss encourages high cosine similarity $\mathbf{z}_i^A \cdot \mathbf{z}_i^B$ for matched pairs and low similarity for mismatched pairs. A symmetric version sums losses in both directions (image-to-text and text-to-image) to ensure bidirectional alignment.

CLIP: Contrastive Language-Image Pre-training CLIP (Contrastive Language-Image Pre-training) exemplifies large-scale contrastive learning for vision-language alignment [34]. Trained on 400 million (image, text) pairs harvested from the internet, CLIP learns a joint embedding space where images and their corresponding textual descriptions are proximal. Critically, CLIP requires no manual annotation beyond naturally co-occurring image-text pairs—captions, alt-text, and metadata provide weak supervision signals.

CLIP’s encoders consist of a Vision Transformer (ViT) for images and a Transformer text encoder. During pre-training, a batch of N image-text pairs produces N^2 potential pairings, of which N are correct. The contrastive loss encourages the model to assign high similarity scores to the N correct pairs and low scores to the $N^2 - N$ incorrect combinations. At inference, CLIP can perform zero-shot classification by encoding candidate class names as text prompts (“a photo of a turbine blade crack,” “a photo of a healthy

turbine blade”) and computing cosine similarities with the image embedding. The highest-scoring text prompt determines the predicted class.

For anomaly detection, CLIP enables zero-shot visual inspection without labeled anomaly images. By encoding descriptions of normal and anomalous states (“nominal pressure gauge reading,” “pressure gauge showing overpressure warning”), practitioners can classify sensor images based on semantic similarity to these descriptions. This approach proved effective in industrial quality control, where defect categories are known a priori but labeled training images are scarce [104].

ALIGN and Scaling Contrastive Learning ALIGN (A Large-scale Image and Noisy text embedding) extends CLIP’s approach to over one billion noisy image-text pairs, demonstrating that contrastive learning scales effectively with data volume despite high label noise [105]. Unlike CLIP, ALIGN does not filter or curate its training data, instead relying on the contrastive objective’s robustness: even if 30% of image-text pairs are misaligned, the majority of in-batch negatives remain valid, and the loss still drives correct pairings toward higher similarity.

ALIGN’s success underscores a key advantage of contrastive learning for multimodal anomaly detection: the ability to leverage abundant, imperfectly labeled data. Maintenance logs may contain incomplete or inconsistent descriptions of equipment failures, and thermal images may be timestamped but not explicitly linked to specific log entries. Contrastive pre-training can still extract useful cross-modal correspondences from such noisy pairings, which fine-tuning on smaller, cleaner datasets can refine.

Hard Negative Mining and Curriculum Learning Naive contrastive learning treats all in-batch negatives equally, but many negatives are trivially dissimilar (e.g., an image of a reactor core paired with a description of a cooling tower). Hard negative mining identifies challenging negative examples—image-text pairs that are semantically similar but not correct matches—and overweights them in the loss [106, 107]. For instance, an image of a “cracked weld” and a description of “weld discoloration” are both weld-related but distinct; focusing on such hard negatives sharpens the model’s discriminative ability.

Curriculum strategies gradually increase contrastive difficulty, starting with easy positive/negative pairs and progressively introducing harder examples as training progresses. This approach stabilizes training and improves final performance, particularly when modalities differ substantially in information content or signal-to-noise ratio [108].

Multimodal Contrastive Learning Beyond Vision-Language While CLIP and ALIGN focus on vision-language alignment, contrastive learning generalizes to arbitrary modality combinations. ImageBind learns a unified embedding space across six modalities—images, text, audio, depth, thermal, and IMU (inertial measurement unit) data—using a web-scale dataset of naturally co-occurring multimodal signals [41]. For anomaly detection, this enables queries like “find thermal images similar to this vibration signature” by projecting both modalities into the shared embedding space and computing nearest neighbors.

Time-series and tabular data can also be incorporated via contrastive objectives. TS2Vec applies contrastive learning to time-series by treating augmented views of the same sequence (e.g., time-shifted or noise-corrupted versions) as positive pairs and other sequences as negatives [109]. The resulting embeddings capture temporal dynamics useful for unsupervised anomaly detection: outliers lie far from the dense cluster of normal operational embeddings.

2.4.4 Unified Tokenization Spaces

The discrete, token-based processing that revolutionized natural language modeling—wherein sentences are segmented into subword units that feed transformer architectures—has been generalized to continuous modalities like images, audio, and time-series data. Unified tokenization projects all input modalities into a shared vocabulary of discrete tokens, enabling a single transformer to process heterogeneous data types through a common representational substrate [110, 111].

Motivation and Conceptual Framework Transformers were originally designed for discrete sequences: text tokens from a finite vocabulary. Processing continuous modalities (images, sensor data) within the same framework requires discretization. Vector quantization and learned codebooks provide this bridge: continuous inputs are encoded into a latent space, then quantized to the nearest entry in a learned discrete codebook. The resulting token indices can be processed by standard transformer architectures, and are treated identically to text tokens [112, 113].

Unified tokenization offers several advantages. First, it enables straightforward transfer learning: a transformer pre-trained on text can be directly applied to images if both are tokenized into a shared vocabulary. Second, it simplifies multimodal modeling by eliminating modality-specific architectural components; a single transformer processes vision, language, and sensor tokens interchangeably. Third, it facilitates generative modeling across modalities: the same autoregressive decoder that generates text can generate image tokens, enabling text-to-image or image-to-text generation within a unified framework.

Vector Quantized Variational Autoencoders (VQ-VAE) Vector Quantized Variational Autoencoders (VQ-VAE) formalize this approach [112]. A continuous input $\mathbf{x}$ (e.g., an image or time-series segment) is encoded into a latent representation $\mathbf{z}_e = f_{\text{enc}}(\mathbf{x})$, then quantized by replacing each latent vector with the nearest entry in a learned codebook $\mathcal{C} = \{\mathbf{e}_1, \dots, \mathbf{e}_K\}$:

$$\mathbf{z}_q = \operatorname{argmin}_{\mathbf{e}_k \in \mathcal{C}} \|\mathbf{z}_e - \mathbf{e}_k\|_2. \quad (37)$$

The quantized representation $\mathbf{z}_q$ is decoded via $\hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{z}_q)$ to reconstruct the input. Training optimizes:

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \|\operatorname{sg}(\mathbf{z}_e) - \mathbf{z}_q\|_2^2 + \beta \|\mathbf{z}_e - \operatorname{sg}(\mathbf{z}_q)\|_2^2, \quad (38)$$

where $\operatorname{sg}(\cdot)$ denotes the stop-gradient (the term inside is treated as constant during backpropagation), β controls the commitment loss, and the three terms correspond to reconstruction error, codebook learning, and encoder commitment to the codebook [112].

Once trained, VQ-VAE represents inputs as sequences of discrete token indices, each indexing a codebook entry. A 256×256 image might be tokenized into a 32×32 grid of discrete codes, producing 1,024 tokens. These tokens can then feed a transformer exactly like text tokens, enabling autoregressive image generation (DALL-E, Parti) or multimodal modeling (CM3, Unified-IO) [110, 114, 111, 115].

Multimodal Tokenization in Practice Unified-IO exemplifies multimodal tokenization applied to diverse tasks [115]. Images are tokenized via VQ-GAN (an improved VQ-VAE variant), audio via a learned discrete codec, and structured data via domain-specific tokenizers. All tokens share a single vocabulary space, differentiated only by learned modality embeddings added to each token. A single encoder-decoder transformer processes arbitrarily interleaved sequences: a prompt might contain text, image tokens, and time-series tokens, with the model generating a response in any modality.

For anomaly detection, unified tokenization enables flexible task formulations. An operator might input: “[IMAGE: thermal scan] [TEXT: vibration sensor data] Diagnose anomaly.” The model processes image tokens, text tokens, and (discretized) sensor tokens jointly, generating a textual explanation or highlighting anomalous image regions via generated attention masks. This flexibility contrasts with modality-specific pipelines that require separate models for vision, language, and time-series analysis.

Challenges and Trade-offs Unified tokenization incurs representational costs. Discretizing continuous data loses information: a codebook with $K = 8192$ entries can only represent $\log_2(8192) = 13$ bits per token, potentially discarding subtle patterns in high-precision sensor measurements. Increasing codebook size mitigates this but expands the vocabulary and slows autoregressive generation. VQ-VAE reconstruction quality depends critically on codebook capacity and encoder/decoder architecture; poor reconstruction introduces anomalies into the discretized representation, confounding downstream anomaly detection.

Another challenge is modality imbalance. Text naturally consists of discrete tokens with rich semantic structure, while images and sensor data are continuous and spatially/temporally redundant. Naively tokenizing all modalities identically may privilege text (which compresses efficiently into a few tokens) over images (which require thousands of tokens). Hierarchical tokenization schemes address this by using coarse tokens for global structure and fine tokens for local details, balancing expressiveness against sequence length [116].

Despite these challenges, unified tokenization represents a powerful paradigm for multimodal anomaly detection. By reducing all inputs to a common token-based representation, it enables seamless transfer learning, flexible task composition, and unified generative modeling across modalities—capabilities that are increasingly essential as anomaly detection systems integrate richer, more heterogeneous data sources.

2.5 Foundation Models for Anomaly Detection

With the architectural foundations and multimodal integration mechanisms established, we now survey how these capabilities have been adapted to anomaly detection across industrial, scientific, and operational domains. The marriage of foundation models and anomaly detection represents a paradigm shift from traditional supervised approaches that require extensive labeled anomaly datasets. Instead, foundation models leverage their pre-trained knowledge, few-shot learning capabilities, and generative modeling to identify deviations from normalcy with minimal task-specific data.

Anomaly detection fundamentally asks: “Does this observation conform to expected patterns?” Traditional ML methods answer this question by learning explicit decision boundaries from labeled normal and anomalous examples [117, 118]. However, in many real-world scenarios—nuclear reactor operations, rare equipment failures, novel cyber intrusions—labeled anomalies are scarce, expensive to collect, or impossible to anticipate comprehensively. Foundation models offer alternative strategies: zero-shot detection using semantic descriptions of anomalies, reconstruction-based methods that identify inputs the model cannot faithfully reproduce, embedding-based approaches that detect outliers in learned representation spaces, and generative models that estimate data likelihoods.

This section organizes anomaly detection approaches by their operational mode and underlying mechanism, examining how foundation model capabilities enable each strategy and where current methods face limitations.

2.5.1 Zero-Shot and Few-Shot Anomaly Detection

Zero-shot and few-shot anomaly detection exploit foundation models’ ability to perform tasks from natural language descriptions or minimal examples, eliminating the need for extensive labeled training data. These approaches are particularly valuable in domains where anomalies are diverse, evolving, or poorly characterized a priori.

Zero-Shot Detection via Vision-Language Models Vision-language models like CLIP enable zero-shot visual anomaly detection by encoding textual descriptions of normal and anomalous states into the same embedding space as images [34]. Consider inspecting manufactured components for defects. Rather than collecting thousands of labeled defect images, an operator provides text prompts: “a photo of a flawless circuit board” vs. “a photo of a circuit board with solder bridges” or “a photo of a circuit board with missing components.” CLIP encodes both the test image and these text descriptions, then assigns the image to the category with highest cosine similarity.

Formally, let $\mathbf{i}$ denote an image embedding and $\{\mathbf{t}_1, \dots, \mathbf{t}_C\}$ denote embeddings of C textual class descriptions (including both normal and various anomaly types). Zero-shot classification selects:

$$\hat{c} = \arg \max_{c \in \{1, \dots, C\}} \frac{\mathbf{i} \cdot \mathbf{t}_c}{\|\mathbf{i}\| \|\mathbf{t}_c\|}. \quad (39)$$

For binary anomaly detection (normal vs. anomalous), this reduces to thresholding the similarity ratio:

$$\text{anomaly} = \mathbb{1} \left[\frac{\mathbf{i} \cdot \mathbf{t}_{\text{anomalous}}}{\mathbf{i} \cdot \mathbf{t}_{\text{normal}}} > \tau \right], \quad (40)$$

where τ is a decision threshold tuned on a small validation set, or set based on operational tolerance for false positives.

WinCLIP extends this framework to localized anomaly detection and segmentation [104]. Rather than encoding entire images, WinCLIP computes CLIP embeddings for sliding windows across the image at multiple scales, comparing each window’s embedding to text descriptions of normal and anomalous regions. Windows with high anomaly scores are aggregated into pixel-level anomaly maps, enabling precise localization of defects without pixel-level training labels. WinCLIP demonstrated state-of-the-art zero-shot performance on industrial defect detection benchmarks (MVTec AD, VisA), identifying scratches, dents, and contamination with a level of accuracy comparable to supervised methods trained on thousands of labeled examples [119, 120].

Prompt Engineering for Anomaly Detection The effectiveness of zero-shot detection depends critically on prompt design. Generic prompts like “an anomaly” perform poorly because they lack specificity; the model has no basis for distinguishing thermal anomalies from mechanical failures from sensor malfunctions. Effective prompts incorporate domain knowledge: “a thermal image showing coolant pump overheating with bright hotspots near bearings” provides actionable semantics.

Prompt ensembling aggregates predictions across multiple phrasings of the same concept [34]. For detecting turbine blade cracks, an ensemble might include:

- “a photo of a turbine blade with a fatigue crack”
- “a turbine blade showing stress fractures”

- “a damaged turbine blade with visible cracks”
- “a turbine blade with material defects and fissures”

The final anomaly score averages similarities across all prompts, reducing sensitivity to specific word choices and improving robustness.

Context calibration adjusts CLIP scores by using a prior distribution over categories estimated from unlabeled background data [121]. In industrial settings where normal operations vastly outnumber anomalies, naive CLIP scores may exhibit baseline shifts. Calibration estimates $P(c)$ for each class c from a reference dataset and computes calibrated scores:

$$\text{score}_{\text{calib}}(c \mid \mathbf{i}) = \frac{\exp(\mathbf{i} \cdot \mathbf{t}_c / \tau)}{\sum_{c'} P(c') \exp(\mathbf{i} \cdot \mathbf{t}_{c'} / \tau)}, \quad (41)$$

down-weighting classes that appear frequently in the background distribution.

Few-Shot Anomaly Detection with In-Context Learning LLMs and multimodal foundation models can perform few-shot anomaly detection via in-context learning, where a handful of labeled examples are provided in the input prompt without updating model parameters [53]. For time-series anomaly detection, a prompt might present:

```
Sensor reading: [98.2, 98.5, 98.1, 98.7] -> Normal
Sensor reading: [98.3, 102.4, 115.2, 89.1] -> Anomalous
Sensor reading: [97.9, 98.0, 98.3, 98.1] -> Normal
Sensor reading: [98.5, 98.8, ?]
```

The model leverages the provided examples to infer the pattern that distinguishes normal from anomalous sequences, and predicts the label for the query.

Recent work has extended in-context learning to multimodal anomaly detection. Xu et al. (2025) evaluated GPT-4V and Gemini on time-series anomaly detection by converting numerical sequences into line plots and prompting with few-shot visual examples [76]. While specialized time-series models generally outperform multimodal LLMs in terms of overall performance, multimodal LLMs demonstrated surprising competence at identifying sudden spikes, drift patterns, and periodicity violations from visual inspection alone—a capability traditional methods lack.

The primary advantage of in-context learning is deployment speed: no fine-tuning or retraining is required when anomaly characteristics change. If a new defect type emerges, operators simply add representative examples to the prompt. However, in-context learning is constrained by context window size (typically thousands of tokens), limiting the number and complexity of examples that can be provided. Performance also degrades when test distributions shift substantially from the few-shot examples.

Limitations and Open Challenges Zero-shot and few-shot methods excel when anomalies are semantically describable or visually distinctive, but struggle with subtle, domain-specific deviations. A CLIP model trained on internet images may fail to recognize that a reactor pressure reading of 15.7 MPa is anomalous if the normal operating pressure is 15.2–15.5 MPa; such numerical deviations lack visual or semantic cues that vision-language models can capture. Similarly, in-context learning with LLMs depends on the model’s pre-training corpus; if the domain (e.g., nuclear engineering sensor patterns) is underrepresented, the model may lack the requisite knowledge to generalize from few-shot examples.

Another challenge is class imbalance and calibration. In real-world deployments, normal operations vastly outnumber anomalies, often by a factor of 1000:1 or more. Zero-shot methods must set decision thresholds that balance false positive rates (flagging normal operations as anomalous, causing alert fatigue) against false negatives (missing genuine anomalies, risking safety incidents). Threshold selection remains an open problem, particularly when the cost of false positives and false negatives differs dramatically from one context to the next.

2.5.2 Reconstruction-Based Approaches

Reconstruction-based anomaly detection rests on the principle that models trained on normal data will faithfully reconstruct normal inputs but struggle to reconstruct anomalies. Deviations between the input and its reconstruction signal potential anomalies. Foundation models—particularly autoencoders, variational autoencoders (VAEs), and diffusion models—excel at learning compressed representations of high-dimensional data, making them natural candidates for reconstruction-based anomaly detection.

Autoencoder-Based Anomaly Detection An autoencoder consists of an encoder f_{enc} that maps inputs $\mathbf{x}$ to a low-dimensional latent code $\mathbf{z} = f_{\text{enc}}(\mathbf{x})$, and a decoder f_{dec} that reconstructs the input from the latent code: $\hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{z})$. Training minimizes reconstruction error on normal data:

$$\mathcal{L}_{\text{recon}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{normal}}} [\|\mathbf{x} - f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}))\|^2]. \quad (42)$$

At test time, anomaly scores are computed as reconstruction errors:

$$s_{\text{anomaly}}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2. \quad (43)$$

Inputs with high reconstruction error—those the autoencoder cannot faithfully reproduce—are flagged as anomalous.

We can intuit that the autoencoder learns a compressed representation that captures only the regularities present in normal training data. Anomalies, which violate these regularities, cannot be encoded efficiently, thus yielding poor reconstructions. For instance, an autoencoder trained on thermal images of normally operating heat exchangers learns to represent typical temperature distributions. A thermal image showing an unexpected hotspot (indicative of coolant blockage) produces a high reconstruction error because the autoencoder has not learned to represent such patterns.

Foundation model architectures enhance classical autoencoders in several ways. Vision transformers (ViTs) used as encoders can capture long-range spatial dependencies in images, outperforming convolutional autoencoders on complex scenes [122]. Multimodal autoencoders jointly reconstruct images, text, and time-series data, enabling anomaly detection when only one modality is corrupted (e.g., vibration sensors report anomalies while thermal imaging appears normal) [123].

Variational Autoencoders for Probabilistic Anomaly Scores VAEs extend autoencoders by learning a probabilistic latent representation [124]. Rather than deterministically encoding $\mathbf{x}$ to $\mathbf{z}$, a VAE learns an encoder distribution $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ (typically a Gaussian with mean and variance predicted by neural networks) and a decoder distribution $p_{\theta}(\mathbf{x} \mid \mathbf{z})$. Training maximizes the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_{\phi}(\mathbf{z} \mid \mathbf{x})} [\log p_{\theta}(\mathbf{x} \mid \mathbf{z})] - D_{\text{KL}}(q_{\phi}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})), \quad (44)$$

where the first term encourages accurate reconstruction and the second term (Kullback-Leibler divergence) regularizes the latent distribution toward a prior $p(\mathbf{z})$, typically $\mathcal{N}(0, \mathbf{I})$.

VAEs provide principled anomaly scores via the negative ELBO (equivalently, the negative log-likelihood). An anomaly has low likelihood under the learned data distribution, yielding a large negative ELBO. This probabilistic framing allows for threshold selection based on statistical significance: flag inputs in the bottom α -quantile of the likelihood distribution.

However, VAEs can suffer from the “reconstruction paradox” in that they sometimes reconstruct anomalies poorly if the decoder is overly flexible, assigning probability mass to regions of input space far from the training distribution [125]. To mitigate this, ensemble VAEs train multiple models with different latent dimensionalities or regularization strengths and flag inputs in cases when ensemble members disagree [126].

Diffusion Models for Anomaly Detection Diffusion models, including Denoising Diffusion Probabilistic Models (DDPMs) and score-based generative models, have emerged as powerful density estimators for high-dimensional data [82, 127]. These models learn to reverse a gradual noising process, enabling likelihood estimation and anomaly detection.

A diffusion model defines a forward noising process that progressively corrupts data $\mathbf{x}_0$ into noise $\mathbf{x}_T$ over T steps:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}), \quad (45)$$

where $\{\beta_t\}$ is a noise schedule. The model learns the reverse process $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$, enabling generation by sampling $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ and iteratively denoising to $\mathbf{x}_0$.

For anomaly detection, the reconstruction error after a full forward-reverse cycle (noising $\mathbf{x}_0$ to $\mathbf{x}_T$ and denoising back to $\hat{\mathbf{x}}_0$) serves as an anomaly score [128]:

$$s_{\text{diff}}(\mathbf{x}_0) = \|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2. \quad (46)$$

Anomalies, which lie outside the model’s learned data manifold, exhibit larger reconstruction errors than do normal samples.

An alternative approach uses the diffusion model’s learned score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ as an anomaly indicator. Normal data have high score magnitude (steep likelihood gradients guide samples toward high-density regions), while anomalies in low-density regions have small scores [127]. This score-based detection avoids the computational cost of full reverse diffusion sampling.

Diffusion models have shown promise for industrial anomaly detection. Graham et al. (2023) applied latent diffusion models to semiconductor wafer inspection, achieving state-of-the-art defect detection by modeling the distribution of normal wafer images and flagging deviations [129]. The primary limitation is computational cost: diffusion sampling requires hundreds of forward passes through the neural network, making real-time inference challenging.

Masked Autoencoders and Self-Supervised Pre-Training Masked Autoencoders (MAEs) represent a recent innovation combining autoencoding with self-supervised pre-training [130]. An MAE randomly masks a large fraction (e.g., 75%) of input patches and trains the model to reconstruct the masked regions from the visible context. This objective encourages the model to learn holistic representations capturing spatial and semantic regularities.

For anomaly detection, MAEs pre-trained on large-scale normal data (e.g., ImageNet or domain-specific industrial image repositories) are fine-tuned on target tasks with minimal labeled anomalies. The pre-trained encoder already captures generic visual patterns (edges, textures, object parts); fine-tuning specializes these representations to detect domain-specific deviations. MAEs have achieved strong performance on

unsupervised anomaly detection benchmarks, often matching or exceeding supervised methods when normal training data are abundant but labeled anomalies are scarce [131].

Multimodal MAEs extend this framework to jointly mask and reconstruct across modalities. For instance, masking time-series sensor readings and thermal image regions simultaneously forces the model to learn cross-modal dependencies. If vibration and temperature normally correlate, a test sample in which the vibration is normal but the temperature is anomalous (or vice versa) yields a high reconstruction error, signaling a potential fault.

2.5.3 Embedding-Based Approaches

Embedding-based anomaly detection maps inputs into a learned representation space where normal data cluster tightly and anomalies appear as outliers. Foundation models produce high-quality embeddings that capture semantic content, enabling effective anomaly detection via distance metrics, density estimation, or one-class classification.

Outlier Detection in Embedding Spaces The core idea is straightforward: encode all inputs (both normal training data and test samples) into a shared embedding space by using a pre-trained foundation model, then apply classical outlier detection algorithms. Normal data form a dense cluster; anomalies lie in sparse regions far from the cluster center or boundary.

Common outlier detection methods include:

- **k -nearest neighbors (k-NN):** Compute the distance from a test embedding to its k nearest neighbors in the normal training set. Large distances indicate anomalies [132]. Formally, for embedding $\mathbf{z}_{\text{test}}$ and normal embeddings $\{\mathbf{z}_i\}_{i=1}^N$:

$$s_{\text{kNN}}(\mathbf{z}_{\text{test}}) = \frac{1}{k} \sum_{i=1}^k \|\mathbf{z}_{\text{test}} - \mathbf{z}_{(i)}\|, \quad (47)$$

where $\mathbf{z}_{(i)}$ denotes the i -th nearest neighbor.

- **Local Outlier Factor (LOF):** Measures local density around $\mathbf{z}_{\text{test}}$ relative to its neighbors [133]. Points in low-density regions (relative to their neighborhood) are flagged as anomalies.
- **Isolation Forest:** Constructs random decision trees that isolate anomalies with fewer splits than normal points, exploiting the intuition that outliers are easier to separate [134].
- **One-class Support Vector Machine:** Learns a decision boundary enclosing normal embeddings in a high-dimensional kernel space; points outside the boundary are anomalies [135].

The choice of algorithm depends on the embedding space structure and computational constraints. k -NN is simple and effective for well-separated clusters but scales poorly to high-dimensional spaces (curse of dimensionality). Isolation Forest handles high dimensions gracefully but assumes that anomalies are uniformly rare. One-class Support Vector Machine provides smooth decision boundaries but requires careful kernel selection.

Foundation Model Embeddings for Anomaly Detection Pre-trained foundation models provide embeddings that generalize across domains without task-specific fine-tuning. CLIP embeddings, for instance, capture high-level visual and semantic concepts, enabling anomaly detection on images the model never saw during training. An industrial inspection system might:

1. Encode a corpus of normal component images by using CLIP’s vision encoder.
2. Compute the centroid μ_{normal} and covariance Σ_{normal} of the normal embedding distribution.
3. For a test image with embedding $\mathbf{z}_{\text{test}}$, compute the Mahalanobis distance:

$$s_{\text{Mahal}}(\mathbf{z}_{\text{test}}) = \sqrt{(\mathbf{z}_{\text{test}} - \mu_{\text{normal}})^\top \Sigma_{\text{normal}}^{-1} (\mathbf{z}_{\text{test}} - \mu_{\text{normal}})}. \quad (48)$$

4. Flag images exceeding the specified threshold distance as being anomalous.

This approach requires no labeled anomalies—only normal data—and leverages CLIP’s rich pre-trained representations. It has proven effective on industrial defect datasets in which the pre-training data (general object images) differ substantially from the target domain (manufactured components) [136].

Time-series foundation models like Chronos and TimeGPT provide embeddings for temporal data [36, 37]. These models, pre-trained on diverse time-series datasets, encode sequences into fixed-dimensional vectors capturing the temporal dynamics, trends, and periodicities. Embedding-based anomaly detection identifies time windows whose encodings deviate from historical norms, signaling potential equipment degradation, sensor malfunctions, or operational anomalies.

Deep SVDD and Neural One-Class Classification Deep Support Vector Data Description (Deep SVDD) combines deep learning with one-class classification by training a neural network to map normal data onto a hypersphere with minimal radius [137]. The model minimizes:

$$\mathcal{L}_{\text{SVDD}} = \frac{1}{N} \sum_{i=1}^N \|f_\theta(\mathbf{x}_i) - \mathbf{c}\|^2 + \lambda \|\theta\|^2, \quad (49)$$

where f_θ is a neural encoder, $\mathbf{c}$ is the hypersphere center (often fixed at the origin), and λ regularizes model complexity. At test time, anomaly scores are distances from the hypersphere center:

$$s_{\text{SVDD}}(\mathbf{x}) = \|f_\theta(\mathbf{x}) - \mathbf{c}\|^2. \quad (50)$$

Pre-training f_θ on related tasks (e.g., self-supervised learning on ImageNet) before fine-tuning with Deep SVDD improves performance, especially when normal training data are limited [138]. This transfer learning strategy mirrors foundation model usage: leverage large-scale pre-training to initialize representations, then specialize to the anomaly detection task with minimal domain-specific data.

Prototypical Networks and Metric Learning Prototypical networks learn embeddings that cluster normal data around class-specific prototypes. For anomaly detection, a single prototype μ_{normal} represents normal data. The model computes:

$$s_{\text{proto}}(\mathbf{x}) = \|f_\theta(\mathbf{x}) - \mu_{\text{normal}}\|^2, \quad (51)$$

flagging samples far from the prototype as being anomalous. Prototypical networks can incorporate few-shot learning: given a handful of labeled anomaly examples, additional prototypes $\{\mu_{\text{anomaly}_i}\}$ are computed, and classification selects the nearest prototype.

Metric learning objectives like triplet loss and contrastive loss further refine embeddings [106, 100]. Triplet loss encourages the network to place normal samples closer together than to any anomaly:

$$\mathcal{L}_{\text{triplet}} = \max(0, \|\mathbf{z}_{\text{normal}} - \mathbf{z}_{\text{normal}}^+\|^2 - \|\mathbf{z}_{\text{normal}} - \mathbf{z}_{\text{anomaly}}\|^2 + \alpha), \quad (52)$$

where $\mathbf{z}_{\text{normal}}^+$ is another normal sample (positive) and α is a margin. Training with metric learning on a small labeled set of anomalies produces embeddings optimized for separability, improving downstream outlier detection.

2.5.4 Generative Approaches

Generative models estimate the probability distribution of normal data, $p(\mathbf{x})$, enabling anomaly detection via likelihood thresholding: inputs with $p(\mathbf{x}) < \tau$ are flagged as anomalous. Foundation models—particularly autoregressive transformers, GANs, and diffusion models—provide powerful generative frameworks that scale to high-dimensional, multimodal data.

Likelihood-Based Anomaly Detection The theoretical appeal of likelihood-based detection is strong: if a generative model accurately captures $p_{\text{normal}}(\mathbf{x})$, then anomalies, which violate the learned distribution, will have low likelihood. For a test sample $\mathbf{x}$:

$$\text{anomaly} = \mathbb{1}[p_{\theta}(\mathbf{x}) < \tau], \quad (53)$$

where p_{θ} is the learned density and τ is a threshold.

Autoregressive models like GPT decompose $p(\mathbf{x})$ into conditional factors:

$$p_{\theta}(\mathbf{x}) = \prod_{t=1}^T p_{\theta}(x_t | x_{<t}), \quad (54)$$

enabling exact likelihood computation via the chain rule. For time-series anomaly detection, an autoregressive transformer trained on normal operational data assigns low probability to sequences containing abrupt spikes, missing values, or out-of-distribution trends [139].

Normalizing flows provide another exact likelihood approach by learning invertible transformations $f_{\theta} : \mathbf{x} \rightarrow \mathbf{z}$ that map data to a simple base distribution (e.g., Gaussian) [140]. The change-of-variables formula gives:

$$\log p_{\theta}(\mathbf{x}) = \log p_{\mathbf{z}}(f_{\theta}(\mathbf{x})) + \log \left| \det \frac{\partial f_{\theta}}{\partial \mathbf{x}} \right|, \quad (55)$$

enabling efficient likelihood evaluation. Normalizing flows have been applied to industrial sensor data, where the learned transformations capture complex dependencies among multivariate signals [141].

Generative Adversarial Networks for Anomaly Detection Generative Adversarial Networks (GANs) learn data distributions through adversarial training: a generator G maps random noise $\mathbf{z}$ to synthetic data $G(\mathbf{z})$, while a discriminator D distinguishes real data from generated samples [142]. While GANs do not provide explicit likelihoods, they can be adapted for anomaly detection via reconstruction or discrimination scores.

AnoGAN trains a GAN on normal data, then detects anomalies by solving an optimization problem: given a test sample $\mathbf{x}$, find the latent code $\mathbf{z}^*$ that best reconstructs $\mathbf{x}$ via the generator [143]:

$$\mathbf{z}^* = \arg \min_{\mathbf{z}} [\|\mathbf{x} - G(\mathbf{z})\|_2^2 + \lambda \|f_D(\mathbf{x}) - f_D(G(\mathbf{z}))\|_2^2], \quad (56)$$

where f_D extracts intermediate features from the discriminator. The anomaly score combines the reconstruction error and discriminator feature mismatch:

$$s_{\text{GAN}}(\mathbf{x}) = \|\mathbf{x} - G(\mathbf{z}^*)\|_2^2 + \lambda \|f_D(\mathbf{x}) - f_D(G(\mathbf{z}^*))\|_2^2. \quad (57)$$

Anomalies yield high scores because the generator, trained only on normal data, cannot produce convincing reconstructions.

Later variants like f-AnoGAN improve efficiency by training an encoder to directly map $\mathbf{x} \rightarrow \mathbf{z}$, eliminating the costly optimization at test time [144]. GANs have been applied to medical imaging anomaly detection (identifying tumors, lesions) and industrial quality control (detecting manufacturing defects) with success, though training stability remains a challenge.

Diffusion Models as Generative Anomaly Detectors As discussed in Section 2.5.2, diffusion models can be used for reconstruction-based anomaly detection. They also enable likelihood-based detection via learned score functions. Score-based generative models estimate $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ (the gradient of the log-density) without computing $p(\mathbf{x})$ explicitly [127]. For anomaly detection, the score magnitude serves as a normality indicator:

$$s_{\text{score}}(\mathbf{x}) = -\|\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})\|^2. \quad (58)$$

Normal data have large score magnitudes (steep likelihood gradients direct samples toward high-density modes), while anomalies in flat, low-density regions have small scores.

Alternatively, the ELBO of a diffusion model lower bounds the log-likelihood, providing an approximate density estimate:

$$\log p(\mathbf{x}) \geq \mathbb{E}_q[\log p_{\theta}(\mathbf{x}_0 \mid \mathbf{x}_{1:T})] - D_{\text{KL}}(q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{1:T})), \quad (59)$$

where q is the forward diffusion process. Samples with a low ELBO are flagged as anomalies.

Diffusion models’ primary advantage is their capacity to model complex, multimodal distributions without mode collapse (a persistent GAN problem). However, likelihood estimates can be unreliable when test data lie outside the training distribution manifold—a scenario common in anomaly detection. Recent work has combined diffusion models with ensemble methods and calibration techniques to improve robustness [129].

Synthetic Anomaly Generation for Data Augmentation Generative foundation models can synthesize realistic anomalies for training supervised detectors, addressing the scarcity of labeled anomaly data. For instance, diffusion models conditioned on text prompts (“add a crack to this turbine blade,” “introduce thermal hotspots in this heat exchanger”) can generate paired normal/anomalous images for supervised training [145]. Similarly, LLMs can generate synthetic sensor fault scenarios (“simulate a coolant pump bearing failure”) that inform rule-based or ML-based anomaly detectors.

This generative data augmentation approach has proven effective in medical imaging, where rare pathologies (cancers, congenital defects) can be synthesized to balance training datasets [146]. In industrial settings, synthetic anomalies must be physically plausible—generated defects should resemble real failure modes rather than arbitrary corruptions. Domain expertise guides prompt engineering and generative model fine-tuning to ensure realism.

Limitations and Failure Modes Generative approaches face several challenges. Likelihood-based detection assumes that the generative model accurately captures $p_{\text{normal}}(\mathbf{x})$, but models can assign high likelihood

to out-of-distribution inputs (the “likelihood paradox”) [125]. For example, a generative model trained on CIFAR-10 images may assign higher likelihood to SVHN images than to CIFAR-10 test images, violating the assumption that in-distribution data have higher likelihood than out-of-distribution anomalies.

Computational cost is another limitation. Sampling from diffusion models or GANs, optimizing latent codes in AnoGAN, or evaluating normalizing flow Jacobians can be expensive, hindering real-time deployment. Furthermore, generative models trained on normal data may “learn to generate” certain anomalies if the training set inadvertently contains unlabeled anomalies or if the model overgeneralizes.

Despite these limitations, generative foundation models remain a frontier in anomaly detection research, offering principled probabilistic frameworks and the capacity to handle high-dimensional, multimodal data at scale.

2.6 Evaluation Frameworks and Benchmarks

Rigorous evaluation of anomaly detection systems requires carefully designed metrics, representative benchmark datasets, and protocols that reflect real-world operational constraints. Unlike balanced classification tasks where accuracy suffices, anomaly detection confronts severe class imbalance (anomalies are rare), variable misclassification costs (missing a critical failure differs fundamentally from a false alarm), and domain-specific requirements (real-time processing, interpretability, graceful degradation). This section surveys the evaluation frameworks and benchmark datasets that enable meaningful comparison of foundation model approaches to anomaly detection.

The central challenge in anomaly detection evaluation is defining the ground truth. In supervised settings, labeled anomalies provide clear benchmarks, but labeling is expensive and often incomplete—real-world datasets may contain unlabeled anomalies in the “normal” training set, biasing evaluation. Unsupervised evaluation must rely on proxy metrics (reconstruction error, embedding distance) whose correlation with true anomaly status is imperfect. Furthermore, anomaly types evolve: a model evaluated on historical defects may encounter novel failure modes in deployment, rendering static benchmarks insufficient.

We organize this discussion around four themes: quantitative evaluation metrics that capture detection performance under class imbalance, benchmark datasets spanning industrial, scientific, and operational domains, evaluation protocols that test generalization and robustness, and domain-specific considerations for safety-critical applications.

Accuracy: Classification accuracy measures the overall correctness of the predictions. It is calculated by dividing the total number of correct predictions by the total number of predictions. The formula for accuracy is given below, where TP is true positives, TN is true negatives, FP is false positives and FN is false negatives:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Standard classification accuracy is misleading for anomaly detection, due to extreme class imbalance. If anomalies constitute 0.1% of samples, a trivial classifier that labels all inputs as normal achieves 99.9% accuracy while detecting zero anomalies. Effective metrics must account for imbalance, misclassification costs, and operating point flexibility.

Precision, Recall, and F-Score Precision and recall form the foundation of anomaly detection evaluation. Let TP (true positives) denote correctly detected anomalies, FP (false positives) denote normal samples incorrectly flagged as anomalous, TN (true negatives) denote correctly identified normal samples, and FN

(false negatives) denote missed anomalies. Precision measures the fraction of flagged samples that are genuinely anomalous:

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (60)$$

High precision minimizes false alarms, reducing alert fatigue and operational disruption. Recall (also called sensitivity or true positive rate) measures the fraction of anomalies successfully detected:

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (61)$$

High recall ensures that critical anomalies are not missed, as is essential in safety-critical domains.

Precision and recall trade off: lowering the anomaly detection threshold increases recall (more anomalies caught) but decreases precision (more false alarms). The F-score harmonizes these metrics:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}, \quad (62)$$

where β weights recall relative to precision. $\beta = 1$ (the F1-score) treats them equally, $\beta > 1$ prioritizes recall (appropriate when missing anomalies is costly), and $\beta < 1$ prioritizes precision (appropriate when false alarms are expensive). For nuclear safety applications, $\beta = 2$ or higher is common, reflecting the imperative to detect all potential failures even at the cost of investigating some false positives [147].

F1 Score: There is often a tradeoff between Precision and Recall, so the F1-score is used to combine them into a single metric —namely the harmonic mean of precision and recall, providing a single metric to balance both. The formula for the F1-score is:

$$\text{F1-Score} = \frac{2TP}{2TP + FP + FN} = 2 \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Receiver Operating Characteristic and Area Under Curve The Receiver Operating Characteristic (ROC) curve plots the true positive rate (TPR = Recall) against the false positive rate (FPR) across all possible decision thresholds:

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}. \quad (63)$$

A perfect detector achieves TPR = 1, FPR = 0 (upper-left corner); random guessing yields the diagonal TPR = FPR. The area under the ROC curve (AUC-ROC) summarizes performance across all thresholds:

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(\text{FPR}) d(\text{FPR}). \quad (64)$$

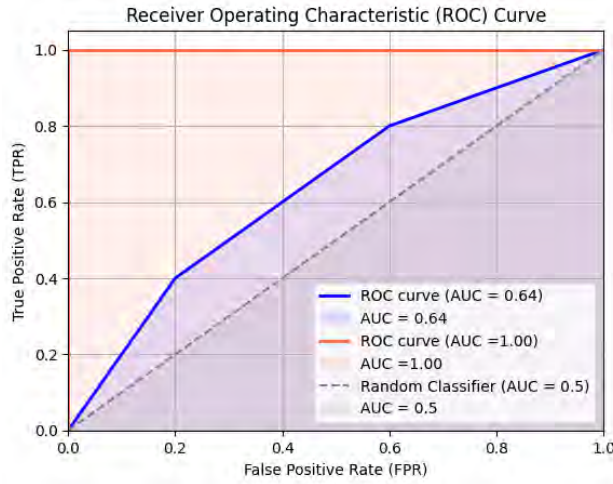


Figure 31: The ROC curve's shape determines the AUC, which defines the true and false positive rates.

AUC-ROC = 1 indicates perfect discrimination, and AUC-ROC = 0.5 corresponds to random performance. AUC-ROC is threshold-independent and robust to class imbalance in one sense (it measures rank-ordering quality), but can be overly optimistic when anomalies are extremely rare. This is because the denominator in FPR (total negatives) is very large, making small absolute increases in false positives appear negligible, as shown in Figure 31 [148].

Precision-Recall Curve and AUC-PR The Precision-Recall (PR) curve plots precision against recall across thresholds, providing a more informative view under severe class imbalance. Unlike ROC curves, which can appear deceptively good when negatives vastly outnumber positives, PR curves emphasize performance on the minority (anomalous) class. The area under the PR curve (AUC-PR) quantifies overall performance:

$$\text{AUC-PR} = \int_0^1 \text{Precision}(\text{Recall}) d(\text{Recall}). \quad (65)$$

AUC-PR is more sensitive to class imbalance than AUC-ROC and better reflects performance in operational scenarios where the positive (anomalous) class is rare [147, 148]. For industrial anomaly detection benchmarks with 1% anomaly rates, AUC-PR often reveals performance differences invisible in AUC-ROC.

Cost-Sensitive Metrics In many applications, misclassification costs are asymmetric. Missing a critical equipment failure (false negative) may cost millions in downtime, environmental damage, or safety incidents, while a false alarm (false positive) costs operator time to investigate. Cost-sensitive evaluation assigns explicit costs to each error type:

$$\text{Cost} = C_{\text{FP}} \cdot \text{FP} + C_{\text{FN}} \cdot \text{FN}, \quad (66)$$

where C_{FP} and C_{FN} are domain-specific costs. The optimal operating point minimizes total cost rather than maximizing F-score or AUC [149].

For nuclear reactor safety, C_{FN} might include the costs of undetected coolant leaks (reactor shutdown, repairs, regulatory penalties, public safety risks), while C_{FP} reflects the cost of unnecessary inspections. Explicitly modeling these costs allows operators to select detection thresholds that align with risk tolerance and operational priorities.

Time-to-Detection and Localization Metrics Static detection metrics ignore temporal dynamics. In time-series anomaly detection, early detection is critical: identifying equipment degradation days before failure enables preventive maintenance, while detection only at catastrophic failure is operationally useless. Time-to-detection measures the lag between anomaly onset and alert:

$$\text{TTD} = t_{\text{detection}} - t_{\text{onset}}, \quad (67)$$

where smaller values indicate a faster response. Evaluation protocols for time-series benchmarks often report both detection accuracy (F1-score, AUC-PR) and average time-to-detection, balancing sensitivity against latency [150].

For spatial anomaly detection (e.g., defect localization in images), pixel-level or region-level metrics assess localization precision. Intersection over Union (IoU) measures overlap between predicted and ground-truth anomaly regions:

$$\text{IoU} = \frac{\text{Area}(P \cap G)}{\text{Area}(P \cup G)}, \quad (68)$$

where P is the predicted anomaly mask and G is the ground truth. MVTec AD and other industrial defect benchmarks report both image-level detection (is this image anomalous?) and pixel-level localization (where is the defect?) using AUC-ROC and AUC-PR for both tasks [119].

2.6.1 Benchmark Datasets and Evaluation Suites

Standardized benchmarks enable reproducible comparisons across methods, driving algorithmic progress. We survey representative datasets spanning visual inspection, time-series monitoring, multimodal data, and domain-specific applications.

Visual Anomaly Detection Benchmarks **MVTec AD (Anomaly Detection)** is the most widely adopted benchmark for industrial visual inspection [119]. It comprises 5,354 high-resolution images across 15 object categories (bottles, cables, capsules, metal nuts, etc.) and texture categories (carpet, leather, wood). Each category includes normal training images and test images with labeled defects (scratches, dents, contamination). Defect types are diverse and realistic, collected from actual manufacturing scenarios. MVTec AD evaluates both image-level anomaly detection (binary: normal vs. defective) and pixel-level anomaly segmentation, reporting AUC-ROC for both tasks. State-of-the-art methods achieve >95% image-level AUC-ROC, but pixel-level performance varies widely (60%–90% AUC-ROC) depending on defect subtlety [151].

VisA (Visual Anomaly) extends MVTec AD with 12 additional object categories and more challenging defect types, including subtle color variations and texture irregularities [120]. VisA emphasizes fine-grained anomalies that require high-resolution analysis and contextual understanding.

BTAD (BeanTech Anomaly Detection) provides real-world industrial images from electronics manufacturing, featuring PCB (printed circuit board) defects and component assembly errors [152]. BTAD includes both structural anomalies (missing components, misalignments) and appearance anomalies (soldering defects, surface damage).

These benchmarks reveal foundation model strengths and weaknesses. Vision-language models like CLIP excel at coarse-grained defect detection (“is there a scratch?”) but struggle with precise localization (“where exactly is the scratch?”). Reconstruction-based methods (autoencoders, diffusion models) achieve strong localization but require careful threshold tuning. Embedding-based methods (k-NN, Deep SVDD) balance detection and localization but depend on feature quality.

Time-Series Anomaly Detection Benchmarks **Skoltech Anomaly Benchmark (SKAB)** contains multivariate time-series data from a water circulation testbed with seven sensors (flow, pressure, temperature) and induced fault scenarios (valve blockages, pump failures) [153]. SKAB provides labeled anomaly intervals, enabling evaluation of detection accuracy and time-to-detection. The dataset includes both abrupt faults (sudden valve closure) and gradual degradation (slow pump wear), testing the methods’ sensitivity to different anomaly dynamics.

Numenta Anomaly Benchmark (NAB) aggregates 58 labeled time-series datasets from diverse domains (server metrics, taxi demand, ambient temperature) with real-world anomalies [150]. NAB introduces a novel scoring function that rewards early detection and penalizes delays, aligning evaluation with operational requirements. Methods are ranked by a composite score that combines accuracy and timeliness.

Yahoo Webscope S5 provides large-scale time-series data from Yahoo’s production systems, including web traffic, server load, and application metrics with labeled anomalies (traffic spikes, service outages) [154]. The dataset tests scalability and robustness to noisy, high-dimensional sensor streams.

UCR Time Series Archive, while primarily a classification benchmark, includes anomaly detection subsets with labeled outlier sequences in domains ranging from ECG signals to industrial sensor data [155]. Foundation models pre-trained on diverse time-series data (Chronos, TimeGPT) are often evaluated on UCR to assess transfer learning across domains.

Multimodal and Domain-Specific Benchmarks **AIOPS Benchmarks** focus on IT operations anomaly detection, combining time-series metrics (CPU, memory, network) with log files and incident reports. The AIOPS Challenge datasets [156] include real-world traces from Alibaba, Baidu, and other large-scale systems, testing multimodal integration of numerical and textual data. Evaluation emphasizes root cause localization: not only detecting anomalies but also identifying which component or service caused the incident.

Prognostics and Health Management (PHM) Benchmarks, such as the NASA Turbofan Engine Degradation dataset [157], provide multivariate sensor readings from aircraft engines operated to failure under various conditions. The task is to predict remaining useful life (RUL) and detect anomalies that indicate impending failure. PHM benchmarks evaluate both detection (“is degradation occurring?”) and regression (“how much time until failure?”).

Medical Imaging Anomaly Benchmarks include datasets for tumor detection (brain MRI, mammography), retinal disease identification (OCT scans), and anatomical abnormality localization [158]. These benchmarks often have limited labeled anomalies (rare diseases, novel pathologies), making them ideal testbeds for few- and zero-shot foundation model approaches.

Cybersecurity Intrusion Detection Benchmarks such as KDD Cup 99, NSL-KDD, and CICIDS provide network traffic data with labeled attack types (DoS, malware, phishing) [159, 160]. Anomaly detection methods must distinguish malicious traffic from benign activity, often under concept drift as attack strategies evolve.

Benchmark Limitations and Future Directions Existing benchmarks exhibit several limitations. First, they often lack diversity: MVTec AD focuses on European manufacturing contexts; SKAB uses a single testbed. Real-world deployments encounter broader variability (lighting conditions, sensor calibration drift, novel defect types). Second, labeling is incomplete: “normal” training sets may contain subtle, unlabeled anomalies, introducing noise. Third, static benchmarks cannot capture temporal evolution— anomaly distributions shift as equipment ages or adversaries adapt.

Future benchmarks should emphasize continual learning (detecting new anomaly types without catas-

trophic forgetting), distribution shift robustness (performance when test conditions differ from training), and cost-sensitive evaluation (explicit misclassification cost modeling). The community also needs multimodal benchmarks that integrate vision, language, time-series, and structured data, reflecting the heterogeneity of real-world monitoring systems [161].

2.6.2 Evaluation Protocols and Best Practices

Beyond metrics and datasets, evaluation protocols—how data are split, how hyperparameters are tuned, how robustness is assessed—critically influence reported performance and generalization to deployment.

Train-Test Splitting and Cross-Validation Standard practice trains anomaly detectors on unlabeled or normal-only data and evaluates on a held-out test set containing labeled anomalies. However, data splitting must account for temporal dependencies and operational contexts. For time-series, random shuffling violates temporal order; instead, chronological splits preserve causality (train on past data, test on future) and assess forecasting ability [162]. Similarly, in industrial settings where sensors are deployed across multiple machines, leave-one-machine-out cross-validation tests generalization to unseen equipment.

For few- and zero-shot evaluation, protocols must carefully separate meta-training (pre-training the foundation model), support sets (few-shot examples provided at test time), and query sets (final evaluation). Leakage between these splits inflates performance estimates. The meta-learning community’s standardized protocols (e.g., Omniglot, miniImageNet few-shot splits) provide templates for anomaly detection benchmarks [163].

Hyperparameter Tuning and Threshold Selection Anomaly detection methods often have sensitive hyperparameters: autoencoder latent dimensions, k-NN values, diffusion sampling steps, detection thresholds. Tuning these on test data—even implicitly by iterative experimentation—leaks information and overstates performance. Best practice reserves a validation set (distinct from both training and test) for hyperparameter search, reporting test performance only after final model selection.

Threshold selection is particularly challenging. In practice, thresholds are chosen to achieve a target false positive rate (e.g., $<1\%$ of normal samples flagged) or to minimize operational cost. However, academic benchmarks often lack the cost information needed for principled threshold setting, defaulting to arbitrary choices (e.g., “flag the top 1% of anomaly scores”). Future evaluation protocols should specify cost functions or operational constraints (“threshold must achieve 95% recall”) to guide threshold selection.

Robustness and Stress Testing Real-world deployments encounter distribution shifts: sensor calibration drift, lighting changes, evolving anomaly types. Robustness evaluation deliberately perturbs test conditions to assess graceful degradation. Common perturbations include:

- **Adversarial examples:** Subtly modified inputs designed to evade detection [164]. For instance, adding imperceptible noise to thermal images to suppress anomaly scores.
- **Covariate shift:** Training on one operating regime (e.g., summer temperatures) and testing on another (winter), without explicit anomaly labels for the new regime.
- **Label noise:** Introducing mislabeled anomalies into the training set to simulate imperfect data collection.

- **Concept drift:** Gradually shifting anomaly distributions to test continual learning and model adaptation.

Stress tests reveal failure modes invisible in standard benchmarks. A method with 98% AUC-ROC on clean data may drop to 70% under modest covariate shift, signaling brittleness. Reporting worst-case performance (minimum accuracy across perturbations) alongside average-case metrics provides a more realistic assessment [165].

Reproducibility and Open-Source Implementations Reproducibility remains a challenge in anomaly detection research. Papers often omit critical implementation details (random seeds, augmentation strategies, early stopping criteria), making results difficult to replicate. The community has responded with standardized evaluation libraries (e.g., ADBench, PyOD) that provide unified interfaces to datasets, metrics, and baseline methods [161, 166]. Best practice is to release code, trained models, and experiment logs alongside publications, enabling independent verification.

Foundation model evaluation introduces additional complexity: models like GPT-4 and Gemini are proprietary APIs whose internal details (training data, architecture, fine-tuning) are undisclosed. Reproducing results requires accounting for API versioning (model updates change behavior), rate limiting (affecting batch processing), and non-determinism (temperature sampling introduces variance). Researchers increasingly report multiple runs with error bars or confidence intervals in order to quantify variability [76].

2.6.3 Domain-Specific Evaluation Considerations

Anomaly detection spans diverse domains—manufacturing, healthcare, cybersecurity, finance—each with unique evaluation requirements shaped by operational constraints, regulatory mandates, and stakeholder priorities.

Industrial and Safety-Critical Systems Nuclear reactors, chemical plants, and aerospace systems demand anomaly detectors with near-zero false negative rates (missed anomalies risk catastrophic failures) but can tolerate moderate false positive rates (human experts investigate alerts). The evaluation emphasizes:

- **Recall at high thresholds:** Achieving 99.9% recall even if precision drops to 10% may be acceptable.
- **Interpretability:** Operators must understand why an alert fired in order to validate the issue and respond appropriately. Black-box foundation models require post-hoc explanation (attention visualization, counterfactual analysis).
- **Certification and compliance:** Regulatory bodies (NRC for nuclear, FAA for aviation) may require provable performance bounds or formal verification, challenging probabilistic deep learning methods.

In these contexts, evaluation often involves domain experts reviewing detected anomalies and assessing whether alerts are actionable. Qualitative metrics—operator trust, response time, false alarm fatigue—complement quantitative AUC scores [167].

Healthcare and Clinical Decision Support Medical anomaly detection (tumor detection, disease diagnosis) balances sensitivity (detecting all pathologies) against specificity (avoiding unnecessary biopsies or treatments). Evaluation protocols must account for:

- **Inter-rater variability:** Ground truth labels often come from radiologists’ annotations, which exhibit 10%–20% disagreement rates. Evaluation should compare model performance to human expert consensus rather than to single-rater labels [168].
- **Subgroup performance:** Models may perform well on average but poorly on underrepresented demographics (pediatric patients, rare diseases). Disaggregated evaluation reflects performance across age, sex, ethnicity, and disease severity strata.
- **Prospective validation:** Retrospective benchmarks (historical data with known outcomes) overestimate real-world performance. Prospective studies deploy models in clinical workflows and evaluate the impact on patient outcomes, physician decision-making, and workflow efficiency.

Cybersecurity and Adversarial Robustness Intrusion detection responds to adversarial attackers who deliberately craft inputs to evade detection. The evaluation must test robustness against adaptive adversaries:

- **Evasion attacks:** Attackers with white-box access (knowing model parameters) or black-box access (querying the model) craft malicious inputs that score below detection thresholds [164].
- **Poisoning attacks:** Attackers inject malicious samples into training data to corrupt the learned model, causing false negatives on future attacks or false positives on benign traffic [169].
- **Concept drift:** Attack strategies evolve faster than the models can be retrained. Evaluation should track performance degradation over time as the threat landscape shifts.

Cybersecurity benchmarks increasingly emphasize adaptive evaluation: after an initial model is published, red teams attempt to evade it, and performance is reassessed against these targeted attacks. Foundation models’ reliance on large, diverse pre-training may improve robustness (by exposing models to wider attack variations), but also introduces new vulnerabilities (adversarial prompts, data poisoning at scale) [170].

Operational Deployment and Human-in-the-Loop Ultimately, anomaly detectors are deployed in sociotechnical systems in which humans interpret, validate, and act on alerts. Evaluation must extend beyond offline metrics to assess:

- **Alert fatigue:** High false positive rates lead operators to ignore alerts, reducing effective detection. Metrics should incorporate operator response rates and time-to-investigation.
- **Explanation quality:** Can operators understand and trust model outputs? User studies assess whether explanations (attention maps, feature importance, counterfactuals) improve decision-making [171].
- **Feedback loops:** In interactive deployments, operator corrections (“this was a false alarm”) can retrain models. Online learning evaluation tracks performance improvement as feedback accumulates.
- **Cost-benefit analysis:** Does the detector reduce downtime, prevent failures, or improve safety? Economic evaluation measures return on investment beyond raw detection accuracy.

As foundation models enable more sophisticated multimodal anomaly detection, evaluation frameworks must evolve to capture these deployment realities—bridging the gap between benchmark performance and operational impact.

2.7 AI Agents And Agentic Orchestration

An LLM-based agent is best understood as a goal-directed system that uses an LLM as a planner or policy inside a closed loop: the agent reasons, plans, and acts by calling tools or APIs, observes their outputs, and then learns from feedback before taking the next step. This multi-step “reason and act”; pattern differs from a one-shot, pure LLM completion and is now a standard way to reduce hallucinations and make behavior auditable. For example, the ReAct [172] framework explicitly interleaves natural-language reasoning with tool calls and uses the observations to revise the plan. Reflexion [173] proposes verbal self-reflection, with agents writing feedback to episodic memory and improving over trials without gradient updates, showing gains across decision-making and coding benchmarks.

2.7.1 Tool Calling

Tool calling, also known as function calling, is the practice of allowing a language model to invoke external capabilities via schema-described interfaces so it can fetch information or perform actions beyond pure text generation. Platform documentation defines function/tool calling as a structured mechanism for models to interface with external systems, where the model emits a tool name plus JSON arguments and an external runtime executes the request [174].

A tool is defined as an externally callable capability the model can use (e.g., hosted web and file search, computer use, or a remote server). A function is a schema-typed way to expose your own code as a tool, where the model emits name plus JSON arguments that the runtime executes. OpenAI’s “Using tools” guide distinguishes built-in tools from custom functions supplied via a tools array, and documents connectors for remote capabilities. Research papers give concrete instances of both notions: WebGPT’s browser module [175] is a retrieval tool for navigating and citing sources during QA, PAL [176] exposes a function-like Python interpreter that executes generated programs and returns results, and Toolformer [177] teaches a model to select and call APIs (e.g., a calculator or search engine) with appropriate arguments. These examples show “tools” as capability endpoints (browsers, APIs, interpreters) and “functions” as the developer’s typed interfaces that wrap such capabilities for the agent.

A tool call is the structured request emitted by the model to invoke a tool/function. The standard lifecycle in OpenAI’s docs has five steps: (1) the app sends a request including available tool schemas with name, description, and JSON parameters; (2) the model decides to call one and returns a structured tool call; (3) the agent and application executes real code or an external API; (4) the agent posts the tool’s result back into the conversation; and (5) the model uses this observation to answer or to issue more calls. This lifecycle is the “act” side of ReAct, where actions interleave with reasoning, and it underlies tool-augmented methods like PAL, which offloads to Python, and Toolformer, which learns when/what to call. Production guides also document optional parallel tool calls and parameter flags that control whether multiple tools may be invoked in one turn [178].

Tool outputs are typically structured data—often JSON—returned by the runtime after executing a call. The agent injects this payload back into the dialogue as a machine-readable message for the model to condition on, enabling grounded follow-ups, retries with revised arguments, or error-aware plans. OpenAI’s help and API docs [179] emphasize validating outputs and surfacing failures/timeouts explicitly. Structured Outputs further constrain model responses to a developer-supplied JSON schema so that the required keys and values are honored, improving reliability when chaining tools. In hosted or MCP settings, the platform can even invoke remote tool endpoints and stream results back to the model, simplifying communications while keeping the tool result in context. Practically, the size of the payload should be bounded to avoid lengthy context, model errors should be surfaced in the function/tool schemas, and each call should be traced

via human-in-the-loop verification and audition.

2.7.2 Agent vs. LLM

There are three major differences between agents and pure LLMs. First, unlike a pure LLM that maps prompts to single-pass completions, an agent acts on the world by deciding when to call tools, which arguments to pass, and how to incorporate observations into subsequent decisions. This capability is central to grounding and error-correction. In the ReAct framework, the model interleaves natural-language reasoning with concrete actions so that each step can be supported or revised by observed evidence [172]. Complementary lines of work show that agents can decide to use tools by themselves, either by learning when and how to invoke APIs [180] or by offloading subproblems to reliable external runtimes [181]. Together, these patterns turn the LLM from a passive generator into a controller that can fetch facts, compute precisely, and verify or refine intermediate results rather than performing simple Q&A. Second, pure LLM prompting is effectively stateless beyond the active context window. However, agents maintain short-term states and long-term memory that persist across steps and sessions, enabling long-horizon tasks, multi-document workflows, and accumulation of experience. Systems like MemGPT [182] introduce virtual context management and tiered memory for fast and slow stores so an agent can recall past interactions or large documents despite window limits. More recent agentic memory designs go beyond naive append-only buffers: they create structured, interlinked memories that evolve over time, improving retrieval and reducing drift as the agent solves related tasks [183]. Surveys of memory mechanisms for LLM agents [184] highlight how the choice of structure (e.g., chunks, triples, and summaries), retrieval strategy, and update policy directly affects an agent’s robustness and sample efficiency. In addition, agents also plan, monitor progress, and revise themselves, rather than relying solely on one-shot in-context learning. Chain-of-thought prompting [185] elicits explicit intermediate steps for complex reasoning, and decoding strategies like self-consistency further improve reliability by sampling multiple solutions and aggregating them [186]. Beyond deliberate planning, Reflexion framework treats the agent’s own feedback as a first-class signal. After each attempt, the agent writes a brief reflection to episodic memory and uses it to adapt future behavior and to achieve improvements without any gradient updates to the model weights [173]. In practice, strong agents combine these ingredients: they reason about top- and sub-objectives, act with tools, inspect observations, and refine both plan and context iteratively.

Despite the existence of fast and cost-effective approaches like supervised learning or in-context learning (ICL) for labeling with templated explanations, these approaches generally lack actuation and typically underperform when fresh evidence must be gathered. Foundation models for time series—such as TimeGPT and Lag-Llama—excel at zero-/few-shot forecast and probabilistic residuals, while agent systems should treat these signals as inputs that trigger richer investigations, not as replacements for root-cause analysis (RCA). Multimodal LLMs read visualizations of series well and are particularly strong on range- and variate-wise anomalies. Meanwhile, agents can route these visual cues into tool-verified checks or alternate hypotheses when confidence is low. In short, ICL/time-series FMs/VLMs are high-value components, while agents provide the coordination, grounding, and decision-making needed for robust diagnosis.

Furthermore, LLM-based agents have the potential to transform RCA from a static, one-shot judgment [187] into an active investigation that plans, executes, and revises multi-step probes. In practice, the agent decomposes a case into hypotheses; queries heterogeneous information such as metrics, logs, traces, analysis results; and iteratively tests and prunes explanations before synthesizing a finding. Empirical studies show that such agents can match or surpass strong CoT/RAG baselines on industrial incident datasets while yielding higher factual accuracy and more usable outputs. For example, a ReAct-style RCA agent [188] demonstrated competitive accuracy and better factual grounding than retrieval-only or CoT-only approaches.

A tool-augmented autonomous framework [189] is proposed to improve root-causes predictions, solutions, evidence, and responsibilities over the ReAct baselines with production integrations being reported in cloud platforms. In addition, since incidents rarely live in a single modality, a credible RCA requires aligning time-stamped metrics, span/trace graphs, error logs, deployment diffs, and analysis results, fuse these views into unified representations for the system states, and then invoke specialized localization and fault-type tools to condition future steps on tool responses. Moreover, unlike task-specific detectors or offline analysis scripts, agents can orchestrate the full incident flow from detection to RCA, explanations and mitigation, with human approvals where required. Production-facing systems report pipelines that collect diagnostics, predict root-cause categories, generate explanatory narratives, and propose targeted remedial actions for cloud incidents [190]. Broader surveys describe tooling patterns that stitch together perception, analysis, and assisted remediation into one controllable workflow for different IT operations [191]. This end-to-end structure is crucial when teams need traceable decisions and safe actuation in live systems.

2.7.3 Beyond the Baseline

Powerful options extending beyond the baseline single-agent patterns are worth reviewing. In practice, two families stand out for hard, cross-organization incidents: multi-agent frameworks, which coordinate specialized agents and people through explicit roles and hand-offs; and evolving agents, which continuously improve their algorithms or operating “playbooks” by using evaluator feedback and curated memory rather than weight updates. These approaches offer orchestration, parallelism, and persistence to a degree that a lone ReAct-style agent would be largely unable to deliver at scale. For anomaly detection and diagnosis, multi-agent orchestration allows us to parallelize evidence gathering across modalities, including metrics, logs, traces, tickets, and mirror organizational reality—each agent or human owns a role, with the Incident Commander coordinating hypotheses and mitigations. Graph-controlled flows like LangGraph and Standard Operating Procedure (SOP)-encoded teamwork like MetaGPT make the process inspectable and enforceable, reducing brittle ad-hoc reasoning and ensuring approvals and runbook compliance. Empirical agent-RCA papers also show that graph-augmented, tool-using workflows improve both accuracy and actionability in comparison to text-only baselines.

AutoGen [192] is an multi-agent framework with an Incident Commander that runs a conversation among specialized agents (e.g., metrics/logs/traces analysts), a knowledge-base/RAG retriever, and a change-diff executor, while keeping a human in the loop for approvals or escalation. Its programming model centers on conversable agents that can call tools, talk to each other, and follow role policies. AutoGen has been used in nuclear applications to create digital twins for thermal energy storage system [193]. LangGraph [194] offers graph/state-machine control over the workflow of nodes like Detection, Scoping, Hypothesis Test, RCA, Mitigation, with retries/guards/branches, and persist state and memory for auditability. This “explicit control-flow” style has become common guidance for production agent stacks because it makes behavior inspectable and recoverable. MetaGPT [195] encodes teamwork as SOP-driven roles (e.g., Project manager, Architect, and Engineer), turning postmortems and runbooks into repeatable “assembly-line” steps such as Triage, Evidence Collection, Hypothesis Vetting, and Fix Problem. This is a pragmatic way to mirror the organization chart in the agent system and ensure that approvals and documentation get enforced. Figure 32 shows the scheme for one of the multi-agent frameworks involving multiple LLM-based agents to decompose and solve a complex problems collaboratively.

Considering that anomaly detection and diagnosis are highly domain-specific, evolving agents can turn every resolved incident into an accumulating advantage. FunSearch/AlphaEvolve-style loops can search for better anomaly scorers, heuristics ranking, or remediation scripts under automated evaluators, while agentic context engineering (ACE) and agentic memory ensure that your playbooks improve continuously with real

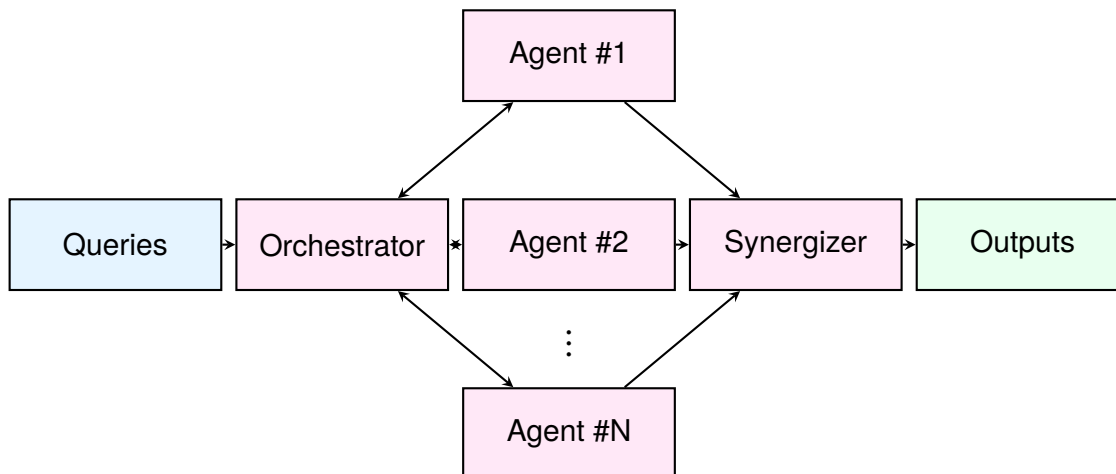


Figure 32: Scheme of multi-agent framework.

execution feedback—without new model checkpoints. The result is a system that not only explains this outage more reliably, but is measurably better at handling the next one.

FunSearch [196] pairs an LLM with an automated evaluator and uses an evolutionary loop to mutate and select code—an approach that avoids free-form hallucination by only keeping variants that pass external tests. DeepMind showed that this can discover new math/CS results and operational heuristics, illustrating how an evaluator-in-the-loop search can produce provably better procedures than static prompts. AlphaEvolve [197] generalizes that idea into a Gemini-powered evolutionary coding agent aimed at algorithm discovery and optimization. It orchestrates LLM-generated edits, multi-criteria evaluation, and selection to steadily improve programs; the public white paper and artifacts describe gains on scientific and engineering problems and outline the autonomous loop that can be adapted to operational algorithms. ACE [198] treats an agent’s context, including the system prompt, tool tips, runbook snippets, validated queries, and known failure modes, as an evolving playbook. Instead of rewrites that lose detail, ACE adds structured “delta” items, tracks helpful/harmful counters, and merges them deterministically, yielding self-improving contexts during runtime and during nightly curation, without fine-tuning. Industry guidance on “context engineering” converges on the same point, with the curation and evolution of what goes into the window being as important as the base model. Agentic memory systems such as MemGPT [182] and A-MEM [183] complement ACE by providing virtual context management and structured, interlinked long-term memories (Zettelkasten-style). These systems page information across tiers, create atomic notes with rich attributes, and maintain links across related cases, as is key for retaining the incident history and reusing it in future investigations. Figure 33 shows the scheme of evolving agents with user defining objective and acceptance criteria, and LLM can orchestrate the entire workflow or just be involved in specific modules.

Despite these potentials, the agent’s capabilities can be limited by the tool schema, gaps in grounding observations, and memory drift. When deploying an agent that calls external tools, the design of the tool interface—for example, the schemas specifying the tool name, arguments, output types, or error models—directly affects the agent’s reliability. Poorly defined tool contracts can result in mismatches with wrong parameter types and unexpected outputs, perhaps causing the planner to further revise or abort its plan. Meanwhile, agents depend on anchoring their reasoning steps in reliable observations. However, in many deployments, retrieval is usually shallow and based on generic textual search, dependency graphs are absent or weak, and tool outputs may come unverified. These gaps in grounding allow agents to drift and hallucinate plausible-yet-incorrect causes or fixes. One effective mitigation is to adopt a formal protocol (e.g. model context protocol [MCP]) to standardize tool schemas that define tool names, argument types,

Human defines “What”?

Sets evaluation criteria, provides initial solutions and optional background knowledge

Problem definition and acceptance criteria ↓

↑ Improved solution

Evolving agent(s) figures out “How”?

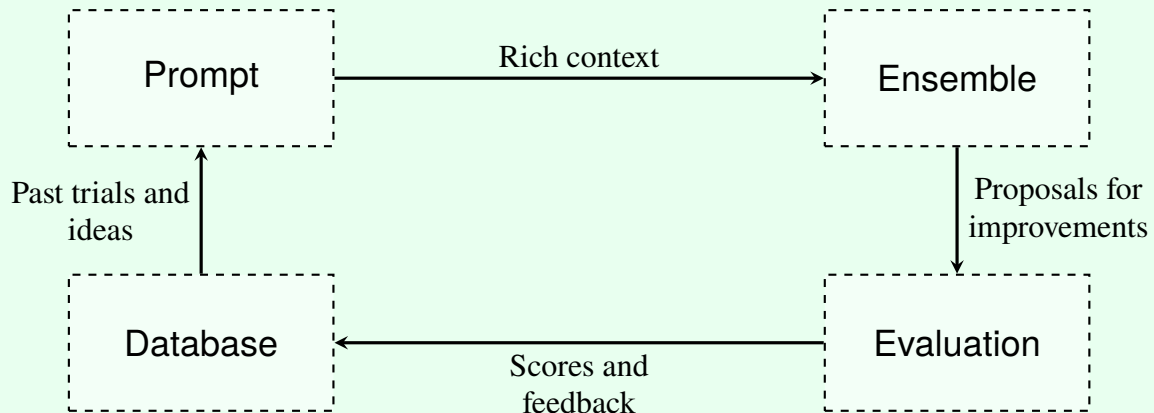


Figure 33: Evolving agent architecture with human-in-the-loop oversight (adopted from [197]).

return schemas, error states, and latency expectations upfront. By registering each tool endpoint with MCP-style metadata, the agent’s planner can reason about capabilities and incorporate retry or fallback paths automatically. Additionally, monitoring and observability become easier with standard protocol, where the agent’s execution loop can compare actual latencies against the schema contract, trigger “abstain” or “fallback to human” when thresholds are exceeded, and log latency deviations for tooling governance.

3 BASE MODEL

The model selected to be the base model for comparison against classical anomaly detectors/classifiers is the Tabular Prior-Data Fitted Network (TabPFN).

TabPFN was not chosen because it is fashionable; it was selected after testing several competitors (FT-Transformer, CatBoost ensembles, Time-Transformers) on the small-data slices that actually exist in plant telemetry. Whenever the train set shrank below a thousand labeled rows, TabPFN remained stable, whereas tree ensembles thrashed between overfitting and underfitting. Moreover, its workflow—encode a handful of support rows, issue a prediction in one forward pass—lines up perfectly with the ReAct agent described later. No other foundation-class model we evaluated combined those two properties (small-data fidelity and agent-friendly inference).

TabPFN is a transformer-trained prior that leverages Bayesian inference for small tabular classification tasks [199]. Instead of gradient-based fine-tuning, the model consumes a few labeled support examples and produces class probabilities for new queries in a single forward pass. Figure 34 depicts the workflow of TabPFN, the premise being that Bayesian inference learns the distribution of the synthetic data and then uses it to predict new instances in a generative sense such that training loss is optimized across millions of datasets, ensuring generalization to new tasks—thus qualifying it to be a powerful foundation model for time-series data. In the Nature-scale follow-on [200], the same workflow is extended to ≤ 10 k-row datasets that mix regression and classification targets, incorporate missing-value flags, and demand calibrated abstentions—exactly the characteristics of industrial telemetry logs and operator narratives used throughout this report.

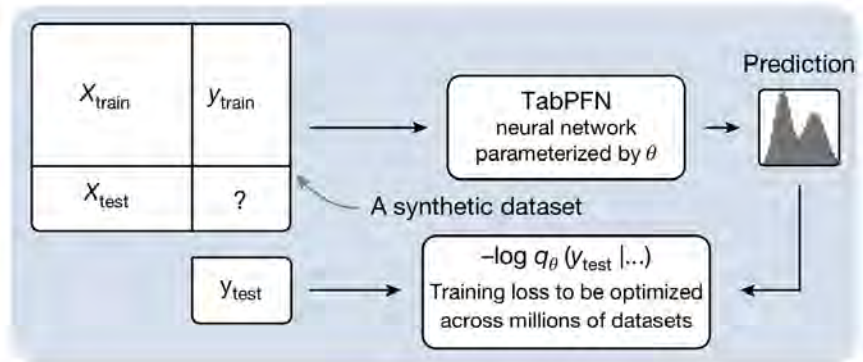


Figure 34: TabPFN workflow (source: “Accurate predictions on small data with a tabular foundation model” [200].)

3.1 Architecture Overview

TabPFN draws support and query tokens from the same transformer sequence as shown in Figure 35. A 12-layer, 8-head decoder-only transformer learns to map the permuted support set and a query token to an approximation of the posterior predictive distribution $p(y^* | x^*, S)$ across tasks drawn from a meta-prior. During inference the model only performs attention-based updates; no gradient steps or external optimizers are required. The Nature variant increases the hidden width to 1 024, adds lightweight adapters for regression heads, and keeps latency below the two-second budget on commodity GPUs by relying on a single forward pass regardless of context size [200]. In practice, this means that deploying TabPFN alongside SKAB or

the Industrial Utility Benchmark telemetry simply involves repackaging rows into tokens; the heavy lifting (Bayesian updating) is entailed in the weights.

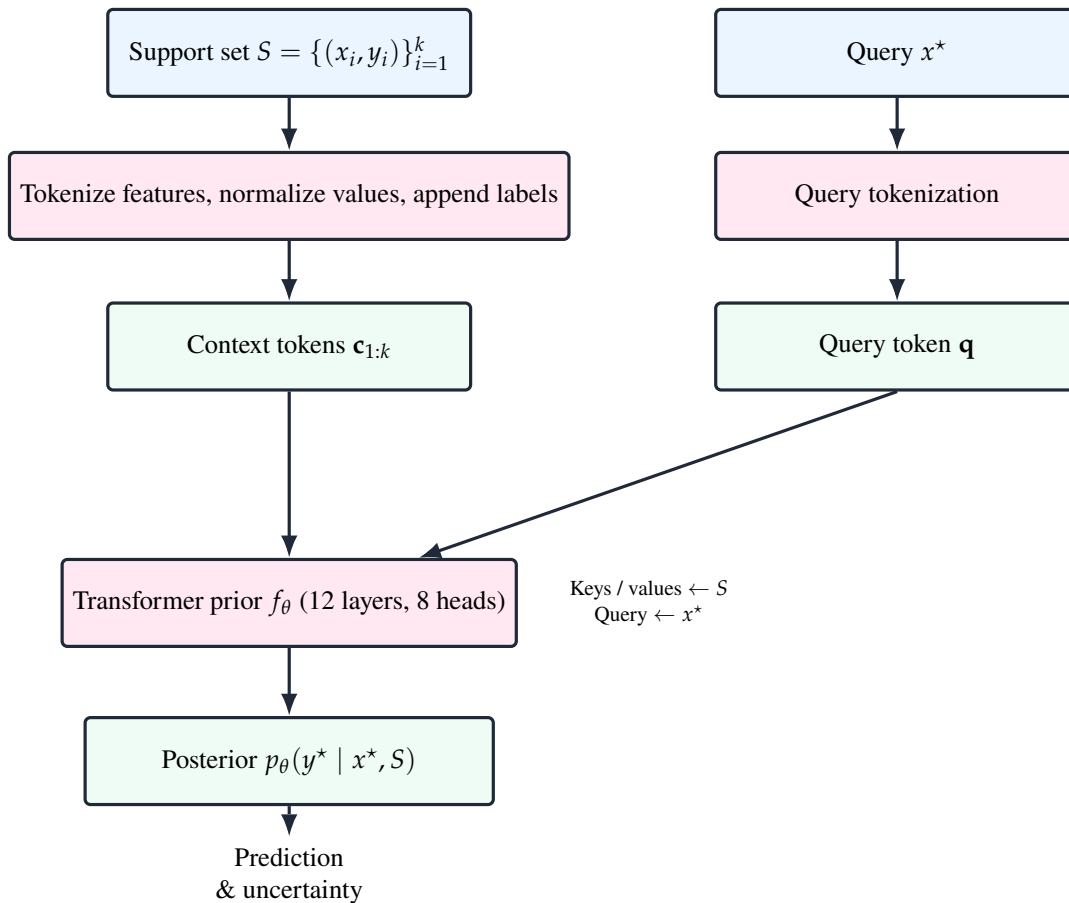


Figure 35: TabPFN inference pipeline. Support examples become context tokens that, together with a query token, are processed by a transformer trained to approximate Bayesian posteriors over synthetic tabular tasks [199].

3.2 Feature and Label Tokenization

Each tabular example is linearized into a sequence of tokens that encode the feature index, a z-scored or rank-normalized value, and (for support points) a class label bit-field [199]. Continuous features are standardized using task-specific statistics, categorical attributes are embedded through learned dictionaries, and missing values attach explicit flags. The tokenizer interleaves support tokens with separator symbols so that attention can respect feature boundaries while remaining permutation invariant. Panel (a) of Figure 36 sketches this encoding. The Nature model further injects modality tags (numeric, categorical, free-text embedding IDs) and missingness indicators so the transformer can learn when to ignore a corrupted sensor column or when to lean on an operator log snippet [200]. Figure 37 illustrates the 2D attention that captures the correlation among samples or time stamps (rows of the tabular data), as well as the correlation among features (columns of the tabular data), thus enabling the model to down-weight all of the feature groups if the support rows disagree with the query.

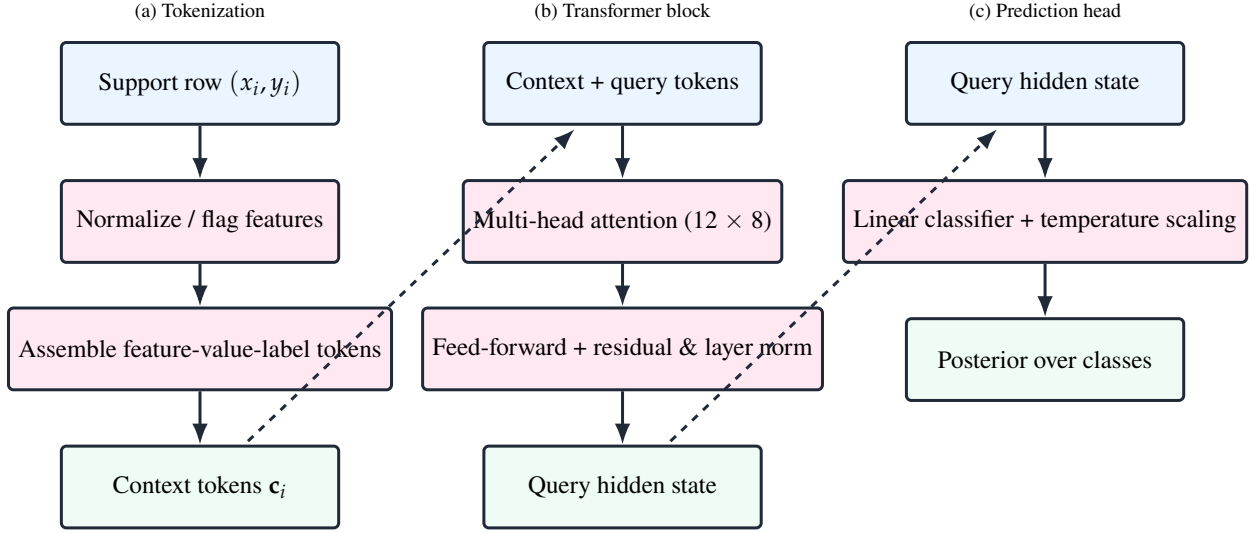


Figure 36: Tokenizer, transformer, and prediction head components within TabPFN. Tokenization enforces permutation invariance while exposing feature semantics, the transformer amortizes Bayesian updates, and the prediction head maps the query representation to class probabilities.

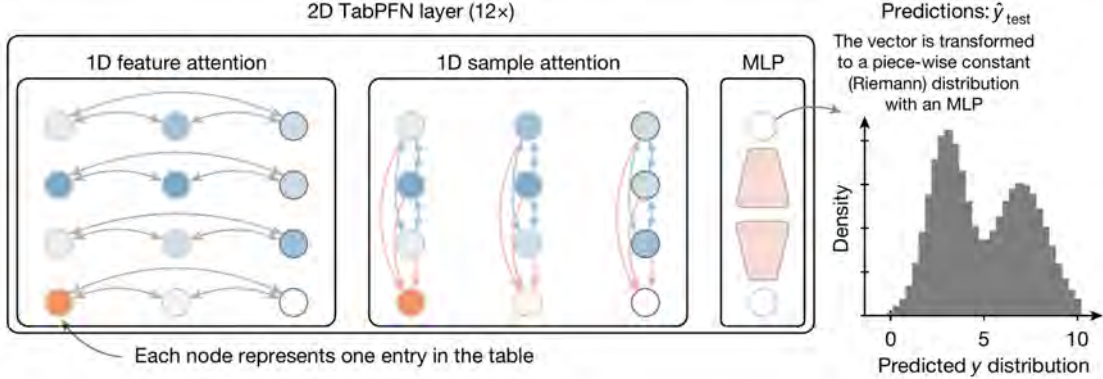


Figure 37: TabPFN 2D attention (source: “Accurate predictions on small data with a tabular foundation model”) [200].

3.3 In-Context Inference

The TabPFN transformer is trained to approximate Bayesian posteriors over tasks drawn from a synthetic meta-prior [199]. The mathematics follow the in-context learning formulation below.

Notation. Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ the label space. A *task* is a joint distribution $\mathcal{T}$ over $(X, Y) \in \mathcal{X} \times \mathcal{Y}$. Let $\mathcal{P}$ denote a *meta-prior* over tasks $\mathcal{T} \sim \mathcal{P}$. A transformer f_θ (parameters θ) maps a *context* of labeled examples to predictive class probabilities for a query x .

Pretraining data $(X, Y)_{\text{pre}}$. We sample many tasks $\mathcal{T} \sim \mathcal{P}$ and, for each, draw a small *support set* $S = \{(x_i, y_i)\}_{i=1}^k$ and one or more *query points* $Q = \{(x_j^*, y_j^*)\}_{j=1}^m$ i.i.d. from $\mathcal{T}$. The model receives the *prompt* $C = \pi(S)$, a random permutation π of the k support pairs, followed by each query input x_j^* . The model outputs $f_\theta(y \mid x_j^*, C)$, a categorical distribution on $\mathcal{Y}$.

Sequence likelihood objective (teacher-forced).

$$\min_{\theta} \mathbb{E}_{\mathcal{T} \sim \mathcal{P}} \mathbb{E}_{(S, Q) \sim \mathcal{T}} \left[- \sum_{j=1}^m \log f_{\theta}(y_j^* | x_j^*, \pi(S)) \right]. \quad (69)$$

The permutation π enforces (approximate) permutation invariance of the support set.

Bayesian/meta-learning view. Each task $\mathcal{T}$ induces a latent parameter ϕ (e.g., class means, margins, noise); the *Bayes-optimal* predictor is the posterior predictive:

$$p(y^* | x^*, S) = \int p(y^* | x^*, \phi) p(\phi | S) d\phi. \quad (70)$$

Pretraining (Eq. 69) trains f_{θ} to *amortize* this posterior predictive:

$$f_{\theta}(y^* | x^*, \pi(S)) \approx p(y^* | x^*, S) \quad \text{for } \mathcal{T} \sim \mathcal{P}. \quad (71)$$

New dataset $(X, Y)_{\text{new}}$. At deployment, we receive a *new* task $\mathcal{T}_{\text{new}}$ with a small labeled set $S_{\text{new}} = \{(x_i, y_i)\}_{i=1}^{k_{\text{new}}}$ and queries $\{x^*\}$ from the same (unknown) data-generating process. *No gradient updates are made to θ* . Predictions are produced purely by conditioning:

$$\hat{p}(y^* | x^*, S_{\text{new}}) = f_{\theta}(y^* | x^*, \pi(S_{\text{new}})). \quad (72)$$

Connection to attention. Inside the transformer, self-attention relates the query token for x^* to the support tokens (x_i, y_i) in the same context window. For standard scaled dot-product attention,

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V, \quad (73)$$

the representations of (x_i, y_i) become the keys/values (K, V) and the query x^* becomes Q ; multi-head attention aggregates multiple *learned* similarity notions in parallel. Through pretraining, f_{θ} learns to implement an *algorithmic update* in-context (e.g., a Bayesian or discriminative rule) that maps S and x^* to a label distribution, without ever changing θ at test time.

Class-conditional likelihood factorization (classification). When the label space is $\{1, \dots, C\}$, one can write $f_{\theta}(\cdot | x^*, S)$ as a softmax over class score:

$$f_{\theta}(y^* = c | x^*, S) = \frac{\exp g_{\theta}(x^*, S, c)}{\sum_{c'=1}^C \exp g_{\theta}(x^*, S, c')}, \quad g_{\theta} : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^k \times \{1..C\} \rightarrow \mathbb{R}, \quad (74)$$

where g_{θ} is realized by the transformer over the concatenated context tokens.

Example: posterior predictive target during pretraining. If the synthetic tasks in $\mathcal{P}$ come from a parametric family with prior $p(\phi)$, and each task produces pairs via $p(y | x, \phi)$ and $p(x)$, then an *ideal* training signal for f_{θ} is the (intractable) posterior predictive

$$p(y^* | x^*, S) = \int p(y^* | x^*, \phi) p(\phi | S) d\phi.$$

TabPFN instead samples from this integral by drawing ϕ and support/query points directly during meta-training, yielding a Monte Carlo estimate of the same target [199].

3.4 Training Regimen and Deployment Considerations

The meta-prior $\mathcal{P}$ combines Gaussian mixtures, decision boundaries with varying margins, and categorical feature generators so that the transformer observes a wide range of inductive biases during training [199]. In the Nature expansion, the prior also samples structural causal models with explicit intervention nodes, draws heavy-tailed noise processes, and injects missingness masks so that the model will rehearse the failure modes it will see in production [200]. Tasks are synthesized with up to a few dozen support examples, matching the low-data regimes targeted at inference time; for longer prompts (≈ 512 rows), the curriculum still enforces that query tokens attend to a sparse, meaningful subset of support rows. Because the model amortizes the posterior, downstream deployment simply requires re-tokenizing the plant telemetry or sensor signals into the same format—no fine-tuning loop or optimizer state must be maintained. In practice, support examples are refreshed as new labeled anomalies arrive, while the fixed TabPFN weights continue to generalize to unseen sensor configurations through attention over feature identifiers.

Differences from generic transformer ICL. TabPFN’s in-context learning differs from language-model ICL in two material ways. First, the tokenization explicitly preserves feature identity and normalized value semantics, enabling the transformer to learn dependencies between rows of a table rather than only sequential tokens. This allows the network to infer task-specific relationships among features in a single forward pass. Second, the training prior $\mathcal{P}$ is explicit: synthetic datasets are generated from a family of structural causal models with varying feature distributions, noise, and class balances, so that the transformer learns an amortized mapping

$$f_\theta : (X, Y)_{\text{pre}} \mapsto p(y^* | x^*, S)$$

for a wide range of tabular distributions. At deployment, the same f_θ is used directly: the new dataset $(X, Y)_{\text{new}}$ is fed as the in-context prompt, and predictions are obtained as

$$\hat{p}(y^* | x^*, (X, Y)_{\text{new}}) = f_\theta(y^* | x^*, (X, Y)_{\text{new}}),$$

without any gradient updates or fine-tuning.

Finally, unlike language-based ICL—where attention operates over sequences of tokens—TABPFN employs row-wise self-attention among training samples, as well as cross-attention from query samples to training samples. This structural modification enables the transformer to perform an implicit Bayesian inference step over the training set, effectively executing a learned meta-inference algorithm in one pass. The resulting behavior can be viewed as a high-dimensional, nonparametric posterior predictor realized through attention rather than gradient descent.

4 PHYSICS-BASED ANOMALY DETECTION

For completeness, this section—prepared by the author from Argonne National Laboratory—offers an overview of physics-based modeling. This approach is generally considered when data-driven anomaly detection does not achieve sufficient accuracy, which is the opposite of our current situation. In cases where data-driven methods struggle to capture the relevant patterns or physical characteristics, physics-based modeling can provide an alternative by explicitly defining these behaviors through physical constraints and governing equations. In some contexts, combining data-driven techniques with physics-based approaches may offer potential benefits. However, a notable limitation is that physics-based modeling can diminish one of the primary advantages of foundation models: their ability to be broadly pre-trained and then adapted to a variety of tasks across different physical domains, without relying on this domain-specific physics during pre-training. This section is included to support readers who may have applications that require intrusive access and explicit incorporation of physical principles, even if doing so may reduce generalization and scalability.

4.1 Introduction

The industrial utility benchmark, relies heavily on water for its thermal efficiency. As a base-load facility utilizing a once-through cooling system, the station was designed to withdraw large volumes of water—with a large design flow per unit to condense turbine exhaust steam. The reliability and efficiency of this Circulating Water System (CWS) are directly linked to the quality and hydraulics of the intake water. However, this is a biologically active and physically dynamic environment, presenting persistent challenges to the mechanical integrity and thermal performance of the plant’s cooling infrastructure.

4.2 The Problem of Biofouling and Debris Accumulation

The primary challenge facing the CWS is fouling caused by the ingress of aquatic life and detritus. Because the station intakes are located on the shoreline of a productive estuary, they are subjected to heavy loads of suspended solids, biological matter, and debris. The operational significance of these challenges was demonstrated in April 1994, when marsh grass floating in the river blocked the traveling screens at the intake structure, forcing operators to reduce reactor power. A series of operational errors during this event led to an automatic reactor shutdown, resulting in a \$500,000 NRC fine.

Similar incidents have recurred with troubling regularity. one of the units was shut down due to circulating water intake blockage caused by debris. Another Unit was tripped following a reduction in circulating water flow from the accumulation of heavy debris blocking the intake. Most dramatically, in another instance, both units experienced flow restrictions following weather events. Both units reduced power due to partial service water intake blockage caused by vegetative matter and large wood pieces flooded into. One of the Units was affected again, when its service water intake was partially blocked with debris—its suction point draws from a marshy area particularly vulnerable to storm-driven debris, while another unit alternate suction point was not appreciably affected. These events underscore that intake fouling is not an isolated occurrence but rather a persistent operational challenge tied to the station’s environmental setting.

This fouling manifests in two distinct categories, both of which degrade system performance:

- **Macrofouling (Debris and Blockage):** Seasonal variations in the river bring surges of marsh grass, silt, and sand. These materials can overwhelm the primary filtration defenses—the trash racks and

traveling screens—located at the Circulating Water Intake Structure (CWIS). While the screens are designed to remove larger debris and fish, smaller particulate matter and fibrous material often pass through, accumulating in the condenser waterboxes and tube sheets. This accumulation restricts flow, increases differential pressure, and can lead to the clogging of the Circulating Water Pump (CWP) motors [201].

- **Microfouling (Biological Growth):** The environment promotes the adhesion and growth of marine organisms on submerged surfaces, a phenomenon known as biofouling. Organisms such as hydroids and mussels can settle within the piping and condenser tubes. This biological layer insulates the condenser tubes, inhibiting heat transfer, and increases fluid frictional resistance, which demands greater pumping power.

4.3 Operational and Economic Consequences

The consequences of biofouling and the subsequent need for flow reductions affect thermodynamic performance, equipment longevity, and regulatory compliance. The primary operational impact is the degradation of the main condenser's heat transfer capability. To maintain turbine backpressure within acceptable limits during fouling events or flow restrictions, the station must often reduce reactor power. Heat balance evaluations indicate that during peak summer demand—when inlet temperatures are highest—a circulating flow restriction of 45% from the permit limit (reducing flow to approximately 555,000 gpm) could necessitate a power reduction of up to 340 MWe when the discharge temperature rise is limited to 21.6° F, or up to 132 MWe when limited to 27.0° F. The permit establishes discharge temperature differential (ΔT) limits of 21.6° F from October through May and 27.0° F from June through September. When these limits are exceeded, mandatory power reductions are required to maintain environmental compliance.

Attempting to manage debris via flow reduction can paradoxically exacerbate microfouling. Reduced flow velocities can increase condenser tube fouling rates. This accelerated accumulation hinders flow further, decreases heat transfer efficiency, and potentially increases corrosion rates within the circulating water piping and condenser tubes. Operating the CWS at reduced capacities forces components to function outside their intended design parameters. Running CWPs at flow rates lower than manufacturer recommendations can lead to increased mechanical fatigue and premature failure. Furthermore, if flow reduction is achieved via throttling, the resulting increase in system pressure poses a risk of failure for circulating water piping.

These physical constraints impose a heavy economic burden. The reduction in electrical output results in significant lost revenue that cannot be recaptured. Additionally, operating in “off-normal” conditions increases the complexity of station operations, requiring intensified monitoring, new seasonal procedures, and analysis of safety margins regarding the condenser's ability to absorb turbine bypass steam during load rejections.

4.4 The Shift to Physics-Based Diagnostics

Given the high costs associated with manual inspection and reactive cleaning, there is a critical need to transition from time-based preventive maintenance to a risk-informed condition-based maintenance strategy. Recent collaborative research between Idaho National Laboratory (INL), and several industrial entities has demonstrated scalable technologies for achieving this transition. This work has established that by leveraging existing plant process data—such as motor currents, intake/discharge temperatures, and vibration signatures—it is possible to construct fault signatures that identify specific degradation mechanisms including waterbox fouling, condenser tube fouling, pump diffuser degradation, and motor misalignment. Advanced

Pattern Recognition techniques and machine learning models, including multi-kernel support vector machines and neural networks with federated-transfer learning, have been developed to distinguish between normal seasonal variations and genuine equipment degradation [202]. Integration of these diagnostic models with three-state Markov chain risk models enables economic assessment of the transition from preventive to predictive maintenance strategies [203].

Building on this foundation, the present work extends the diagnostic capabilities for CWS by employing physics-based diagnostics (PBD) that complement the approaches developed by INL. PBD offers the additional benefit of establishing explicit causal relationships between system behavior and fault mechanisms through first-principles models, enabling operators to understand not only *what* has failed but *why* and *how* it has failed—essential information for making effective corrective decisions.

The PBD approach employed in this work is a *consistency-based* diagnostic framework that uses physics-based models to detect, isolate, and explain faults in thermal-hydraulic systems [204, 205]. The approach constructs normal (fault-free) models of system components using conservation laws and constitutive equations. These models establish constraint relations between observed sensor data—formally defined as Analytical Redundancy Relations (ARRs)—which must be satisfied under normal operation [206]. Violations of these constraints, quantified as non-zero *residuals*, serve as fault symptoms that implicate specific components or sensors through the underlying physics encoded in each ARR. By employing both deterministic and probabilistic reasoning over the residual signatures, the framework produces fault diagnoses that are inherently explainable: each diagnosis can be traced back through the causal chain of physical relationships to the observed symptoms [207].

A key enabling feature of this approach is the use of *virtual sensors*—analytically derived process variables obtained by solving conservation equations for unmeasured quantities [205]. This capability enables diagnostic model construction even when direct measurements are unavailable, significantly expanding coverage of potential fault modes. Physics-based consistency methods have been successfully demonstrated for fault detection and diagnosis in nuclear power plant feedwater systems and sodium test facilities, where they distinguished between component degradation and sensor failures with high sensitivity while maintaining robustness to measurement and modeling uncertainty [205].

For the CWS application, PBD provides a systematic approach to construct fault signatures for the degradation mechanisms identified in subsection 4.2—waterbox fouling, condenser tube fouling, trash rack obstruction, pump degradation, and sensor failures—while simultaneously providing operators with transparent, physically grounded explanations for each diagnosis.

4.5 System Description

The next section provides a brief description of the system used to showcase the PBD approach mentioned in the previous sections.

4.5.1 CWS Role in the PWR Thermal Cycle

Figure 38 illustrates the essential role of the CWS within the broader thermal cycle of a pressurized water reactor. In the primary loop, reactor coolant removes heat from the nuclear fuel and transfers it to the steam generators. The secondary loop steam drives the turbine-generator to produce electricity, after which the low-pressure exhaust steam must be condensed to maintain vacuum and complete the thermodynamic cycle. This condensation duty is fulfilled by the main condenser, which rejects the waste heat to circulating water. The CWIS serves as the critical interface between the natural water source and the engineered

cooling system. As shown in the cutaway view, the CWIS employs trash racks and traveling screens to filter debris and biological matter before water enters the circulating water pumps. These filtration defenses are essential to protecting the condenser tubes and maintaining system hydraulic performance, yet they are also the components most vulnerable to fouling-induced flow restrictions.

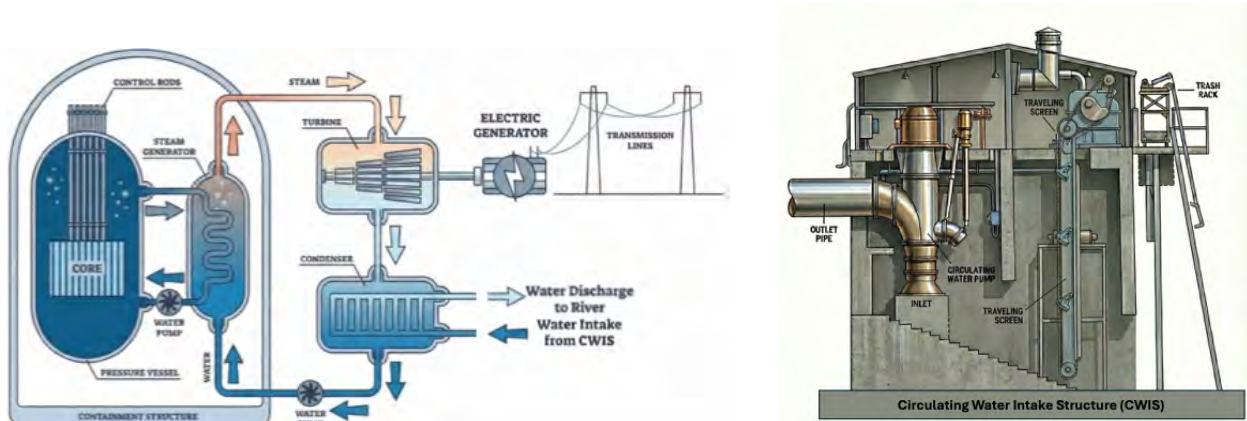


Figure 38: Simplified pressurized water reactor (PWR) cooling circuit (left) showing the role of the CWIS in providing cooling water to the main condenser, and detailed cutaway view of the CWIS (right) illustrating the primary filtration defenses—trash racks and traveling screens—that protect downstream equipment from debris and aquatic life before water enters the circulating water pumps.

4.5.2 System Component Specifications

The CWS represents a large-scale industrial system designed for high reliability under demanding environmental conditions. Understanding the physical characteristics of the major components provides essential context for the simulation model development.

4.5.3 Circulating Water Intake Structure

River water enters the CWS through the CWIS. The CWIS employs a two-stage filtration system to protect downstream equipment from debris and aquatic life:

- **Trash Racks (Bar Racks):** Fixed bar racks constitute the first line of defense, designed to prevent large floating or submerged debris from entering the system. These racks are cleaned periodically with a clam-shell type mechanical rake device to prevent flow-restricting buildup.
- **Traveling Screens:** The second filter barrier, positioned downstream of the trash racks, removes finer materials before water enters the circulating water pumps. These screens are designed to capture and return fish to the river via dedicated troughs and fish counting pools, which enable monitoring of environmental impact. The traveling screens have been modified with improved fish bucket designs to reduce mortality during the screening process.

Each unit's intake structure contains six individual pump bays (wet-pits), with each bay housing one circulating water pump. This compartmentalized design minimizes pump-to-pump hydraulic interactions and allows individual pumps to be isolated and drained for maintenance without affecting other pumps.

4.5.4 Circulating Water Pumps

Each unit is equipped with six vertical turbine-type circulating water pumps that draw water by suction from dedicated wet-pits downstream of the traveling screens. Each pump discharges into an individual line for delivery to one of the six main condenser inlet waterboxes. Nuclear power plants typically employ two to four circulating water pumps per reactor unit; more pumps per unit configuration reflects the high flow requirements of the cooling system.

4.5.5 Main Condenser

The main condenser for each unit consists of three pairs of condensers, yielding six waterboxes total. Each waterbox receives flow from a single dedicated circulating water pump. Upon leaving the respective condenser section, the two outlet waterbox discharge lines combine into a single 120-inch diameter discharge pipe. Each unit has three such discharge pipes that route the heated water approximately 500 feet offshore into the Delaware River. The discharge points are strategically positioned to minimize thermal recirculation to the intake structure and promote rapid mixing with water source.

4.5.6 Sensor Set Adopted

Two sensor configurations are evaluated in this demonstration: one that includes the cooling-water flow-rate measurement and one that excludes it. For both configurations, the sensor set also comprises pump suction pressure, pump discharge pressure, condenser inlet temperature, condenser outlet temperature, and condenser shell pressure.

4.6 Simulation Model Development in Dymola

To enable development and validation of the physics-based diagnostic framework, a high-fidelity thermal-hydraulic simulation model of the CWS was developed using the Dymola modeling environment. This simulation model serves two critical purposes: (1) it provides synthetic sensor data representing both normal operation and fault scenarios for training and testing the diagnostic algorithms, and (2) it establishes the physics-based relationships between observable process variables that form the foundation of the analytical redundancy relations used in PBD. The PBD framework will subsequently be applied to the outputs of this simulation model to demonstrate fault detection, isolation, and diagnosis capabilities.

Figure 39 presents the process flowsheet of the CWS model developed in Dymola. This model represents a single-unit circulating water loop, abstracting the complex infrastructure designed for flows up to 1,110,000 gallons per minute into a component-level thermal-hydraulic representation suitable for physics-based diagnostics.

The flowsheet depicts the primary flow path through the system. Each component serves a specific function in both the physical system and the diagnostic model:

- **Source and sink:** These boundary components represent the Estuary, providing inlet temperature and pressure conditions at the intake and accepting the heated discharge water. They serve as infinite thermal reservoirs in the model, capturing the environmental coupling without modeling the entire estuary hydrodynamics.

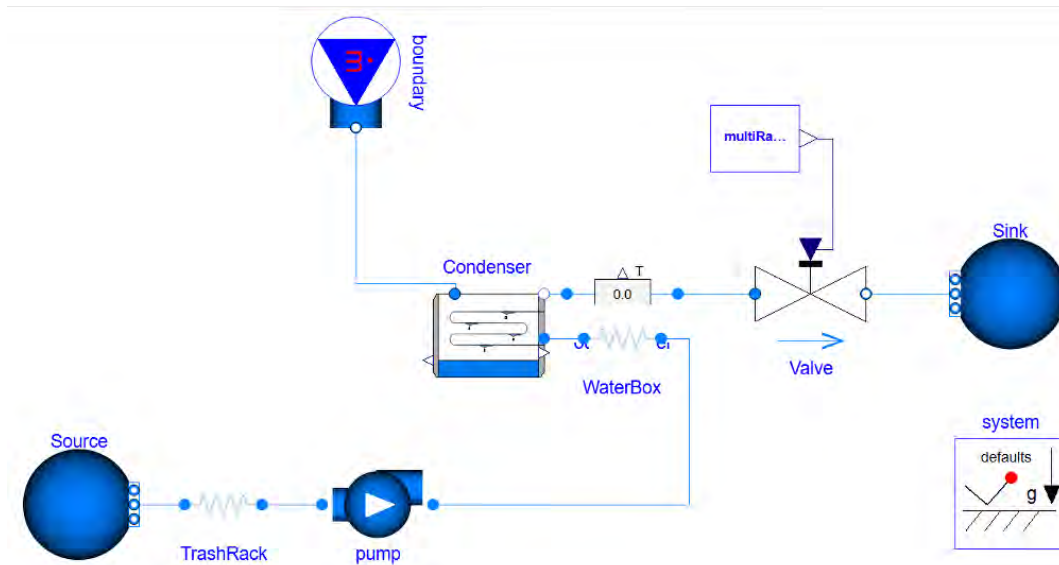


Figure 39: Dymola process flowsheet of the Circulating Water System. The model captures the essential flow path from estuary intake (source) through debris filtration (trash rack), pumping (pump), heat exchange (waterbox and condenser), and discharge (sink). The boundary component provides thermal reservoir conditions representing the Estuary, while the system defaults block establishes gravitational reference.

- **Trash rack:** Represents the primary debris filtration defense at the CWIS. In the actual facility, this includes fixed bar racks and traveling screens designed to exclude fish and remove larger debris such as marsh grass, silt, and sand (detailed in subsection 4.5.3). In the model, this component captures pressure drop as a function of flow rate and allows simulation of fouling-induced flow restriction.
- **Pump:** The CWP is modeled using performance curves derived from manufacturer data, as detailed in subsection 4.6.2. Each unit employs six vertical turbine-type pumps (subsection 4.5.4), and the pump characteristic relates volumetric flow rate, head rise, and power consumption, enabling detection of degradation mechanisms such as diffuser erosion, rotor misalignment, or cavitation through deviations in motor current and differential pressure.
- **Waterbox:** Represents the condenser inlet header or waterbox, a critical location where fine particulate matter and fibrous debris accumulate after passing through the traveling screens. Each unit has several waterboxes, with each waterbox receiving flow from one dedicated pump. Fouling in this component manifests as increased pressure drop and reduced flow distribution to the condenser tubes. The model tracks pressure and temperature to enable early detection of waterbox fouling (WBF), a frequently observed degradation mode.
- **Condenser:** The main heat exchanger where turbine exhaust steam is condensed by transferring thermal energy to the circulating water. The model incorporates heat transfer correlations that account for tube-side fouling, which insulates the heat transfer surface and degrades condenser performance. Temperature rise across the condenser is a primary diagnostic indicator.
- **Multi-ramp source and valve:** These components provide controlled flow delivery to the system through a user-defined valve-opening ramp that represents the startup sequence, gradually increasing the valve position from a low initial value to fully open. The control valve regulates the flow rate and enables the simulation of throttling scenarios used operationally to manage debris loads.

This simplified representation prioritizes the thermal-hydraulic behavior essential for condition-based maintenance while abstracting detailed three-dimensional geometry and internal flow distributions.

4.6.1 Pressure Loss Models

At the nominal operating condition (175,000 gpm), the total head loss through the system is approximately 30 ft. Of this, 27 ft is attributed to the condenser, including the tube bundle, inlet and outlet waterboxes, and associated transition pieces and valves, while the remaining 3 ft accounts for additional losses due to the piping configuration. In the Dymola simulation model, the head losses through the clean trash rack and waterbox at nominal conditions are assumed to be 1 ft and 2 ft, respectively.

The pressure loss through each component is represented by a simplified quadratic function of the mass flow rate:

$$\Delta P = K\dot{m}^2 \quad (75)$$

where K is a pressure loss coefficient estimated at nominal conditions. Faults such as rack clogging or waterbox fouling are modeled by increasing this coefficient.

4.6.2 Circulating Water Pump Model

Each pump is rated at 185,000 gpm at 27 ft of total developed head. An instability region exists below approximately 120,000 gpm, within which operation is not recommended by the manufacturer. The stable portion of the performance curve was reconstructed using a second-order polynomial fit (Figure 40) based on the pump curve provided in public literature. This fitted performance curve was implemented in Dymola to simulate the pump's hydraulic performance.

At the nominal operating condition, the pump delivers approximately 175,000 gpm, slightly below the rated flow, producing around 30 ft of total head.

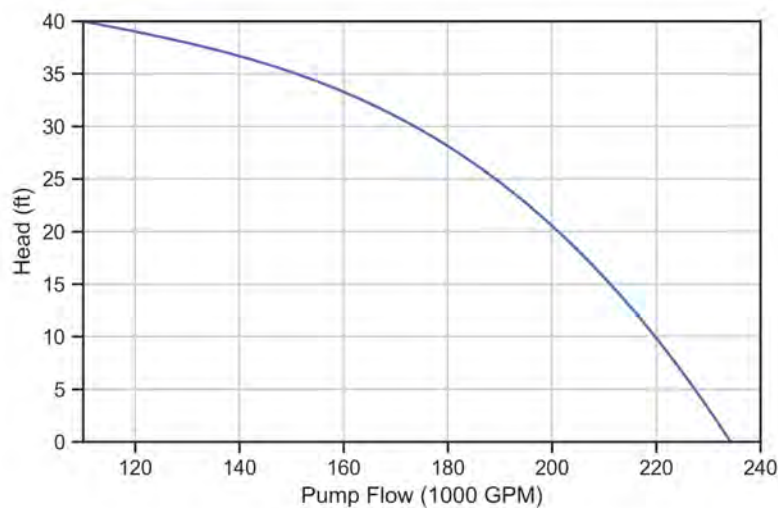


Figure 40: Circulating water pump performance curve in the stable region (reconstructed).

4.6.3 Condenser Model

In the condenser, river water entering through the inlet waterbox is distributed to the individual tubes of the tube bundle to provide cooling for the steam on the shell side, causing it to condense on the tube surfaces. In the simulation model, the flow rates on both sides and the inlet conditions (cooling water temperature and exhaust steam enthalpy) are treated as inputs. The condensate from the condenser shell drain is assumed to be a saturated liquid, with its saturation pressure determined by the condenser shell pressure. Under steady-state operation, the cooling water outlet temperature and shell pressure are dictated by the overall heat balance.

The heat balance equation of the condenser can be expressed as:

$$\dot{Q} = \dot{m}_s(h_{si} - h_{do}) = \dot{m}_w c_{pw}(T_{wo} - T_{wi}) \quad (76)$$

$$h_{do} = h_l(p_{sat}), \quad p_{sat} = p_{condenser} \quad (77)$$

where $\dot{Q}$ is the heat-transfer rate, $\dot{m}_w$ is the mass flow rate of the cooling water, c_{pw} is its specific heat, and T_{wi} and T_{wo} are the inlet and outlet temperatures, respectively. The steam mass flow rate is denoted by $\dot{m}_s$, with h_{si} representing the enthalpy of the incoming steam and h_{do} the enthalpy of the condensate leaving the condenser shell. The condensate enthalpy h_{do} is taken as the saturated liquid enthalpy h_l corresponding to the saturation pressure p_{sat} , which is equal to the condenser shell operating pressure $p_{condenser}$.

To model the heat transfer process in the condenser, we employ the effectiveness-NTU method [208]. The effectiveness is defined through the ratio between the actual heat rate and the maximally achievable heat transfer rate:

$$\dot{Q} = \epsilon \dot{Q}_{\max} = \epsilon \dot{m}_w c_{pw} (T_{\text{sat}} - T_{w,i}) \quad (78)$$

$$\epsilon = 1 - \exp\left(-\frac{UA}{\dot{m}_w c_{pw}}\right) \quad (79)$$

where ϵ is the condenser effectiveness, and $\dot{Q}_{\max}$ the maximum possible heat transfer, corresponding to the case where the cooling water outlet approaches the steam saturation temperature. T_{sat} is the steam saturation temperature, and UA the product of the overall heat-transfer coefficient and the heat-transfer area.

The condenser's effectiveness and overall heat transfer coefficient are estimated at nominal operating conditions, with the heat transfer coefficient then modeled as a function of cooling-water flow rate. Similarly, the total pressure loss on the tube side is modeled as a flow-dependent quantity. At a given operating point, a condenser fouling fault is represented by a simultaneous decrease in the overall heat transfer coefficient and an increase in the pressure loss coefficient.

Having established the simulation model that replicates CWS behavior, the following section describes the diagnostic models that will be calibrated using simulation data and applied for fault detection and isolation.

4.7 PBD Model Development and Implementation

The diagnostic models described in this section serve a different purpose than the simulation models of Section 4.6. While the simulation models replicate the detailed physics of the CWS to generate synthetic sensor data, the diagnostic models establish ARRs that define consistency constraints between measured process variables. These ARRs form the basis for residual generation: deviations from the expected relationships indicate potential faults. The diagnostic models employ more general polynomial forms with calibrated parameters to capture real system behavior, including minor deviations from ideal theoretical models, and to provide the flexibility needed for fault detection through parameter estimation.

4.7.1 Physics-based Diagnostic Models

Pressure Loss Model The pressure loss across components between two pressure measurements is modeled as a quadratic function of the mass flow rate:

$$P_{\text{in}} - P_{\text{out}} = \theta_0 + \theta_1 \dot{m}_w + \theta_2 \dot{m}_w^2. \quad (80)$$

Blockage faults are detected as an increase in the total pressure loss for a given flow rate. Constructing this pressure-loss model requires data from three sensors: inlet pressure, outlet pressure, and the mass flow rate measured at either the inlet or the outlet.

Pump Head Model For a fixed-speed pump, the pump head can be approximated as a quadratic function of the volumetric flow rate. Accordingly, the pressure rise across the pump is modeled as a quadratic function of the mass flow rate:

$$P_{\text{out}} - P_{\text{in}} = \theta_0 + \theta_1 \dot{m}_w + \theta_2 \dot{m}_w^2. \quad (81)$$

This model, which requires sensor readings for the pump suction pressure, discharge pressure, and pump flow rate, represents the pump performance curve calibrated to training data.

Condenser HX Performance Model For the condenser, the overall heat transfer coefficient is modeled as a function of the cooling water mass flow rate:

$$\frac{1}{UA} = \theta_0 + \theta_1 \dot{m}_w^{-0.8}, \quad (82)$$

where the heat transfer resistance on the steam side is assumed to be constant, and the exponent -0.8 is derived from the Dittus–Boelter correlation for turbulent flow. Fouling faults on the condenser tube side are detected as a decrease in the overall heat transfer coefficient. Constructing and calibrating this model requires sensor readings of the condenser shell pressure, the inlet and outlet cooling water temperatures, and the cooling water mass flow rate.

4.7.2 Available Diagnostic Models and Residuals Based on Sensor Configurations

For the sensor configuration that includes the cooling water flow rate sensor, four diagnostic models can be constructed:

- **Suction line pressure loss:** Models the pressure drop between the intake entrance from the river and the pump suction inlet. The residual generated by this model is used to detect fouling (clogging) faults in the trash rack and traveling screen.
- **Pump pressure differential:** Models the pressure rise between the pump suction and discharge points. The residual from this model enables detection of pump performance degradation.
- **Discharge line pressure loss:** Models the pressure drop across the waterbox, condenser, valve, and piping between the pump discharge and the system outlet. The residual from this model allows detection of blockages, such as waterbox fouling.
- **Condenser heat transfer coefficient:** Models the heat transfer performance of the condenser. The residual from this model enables detection of condenser fouling.

For the sensor configuration without the water flow rate sensor, the pump pressure differential model cannot be calibrated. Instead, a pump performance curve is provided as input to the diagnostic framework and is used to estimate the water mass flow rate from the measured inlet and outlet pressures, creating a virtual flow-rate sensor (an analytically derived process variable, as discussed in subsection 4.4). This virtual sensor then allows construction of three of the four diagnostic models listed above (excluding the pump model).

4.7.3 Explainability using LLMs

A key advantage of physics-based diagnostics is the ability to provide causal explanations for fault diagnoses. The ARR in the PBD framework establish explicit dependency structures connecting each residual to the specific sensors, components, and potential faults involved. This causal knowledge enables the diagnostic system to explain not only *which* fault occurred, but also *why* it was identified and *why* other faults were ruled out.

To make this explainability accessible to operators, we integrate a Large Language Model (LLM) agent that interfaces with the diagnostic framework [209]. The agent receives background information about the system (sensors, components, residuals, and their dependency structures) and real-time diagnostic results (active residuals, fault probabilities). Using this context, the LLM can respond to natural language queries from operators about diagnoses.

The agent provides three primary capabilities:

- **Fault identification (`fault_query`):** Identifies the diagnosed fault from the observed residual signature and lists the sensors involved in the active residuals.
- **Diagnostic reasoning (`custom_query`):** Answers arbitrary questions about the diagnosis, such as explaining why certain faults were exonerated based on inactive residuals.
- **Sensor data analysis (`query_sensor_data`):** Analyzes statistical and spectral properties of sensor data to identify anomalies.

4.8 Simulation Results: Startup and Fault Scenarios

The Dymola simulation model described in Section 4.6 serves two purposes: generating synthetic training data for calibrating the diagnostic models and producing fault scenarios to test the diagnostic framework. This section presents the results of these simulations, including a startup sequence for model training and three fault scenarios representing common CWS degradation mechanisms. Noise representing measurement uncertainty was added to the simulation outputs during post-processing.

Throughout this section and Section 4.9, we evaluate the diagnostic framework using two sensor configurations: one that includes a direct flow-rate measurement, and one that relies on a virtual flow-rate sensor derived from pressure measurements using the pump performance curve.

4.8.1 Startup Simulation

A startup sequence of the system was simulated to provide training data for the diagnostic models. During the startup, the discharge control valve opening was controlled using an input ramp function representing

different operating levels of the system. This controlled process enabled the cooling-water flow rate to increase incrementally from an initial value to the full-flow condition. Plots of the main operating variables during this startup sequence are provided in Figure Figure 41.

For the sensor configuration without a flow-rate measurement, a virtual sensor was implemented using the pump performance curve to estimate the cooling-water flow rate from the pump suction and discharge pressure readings.

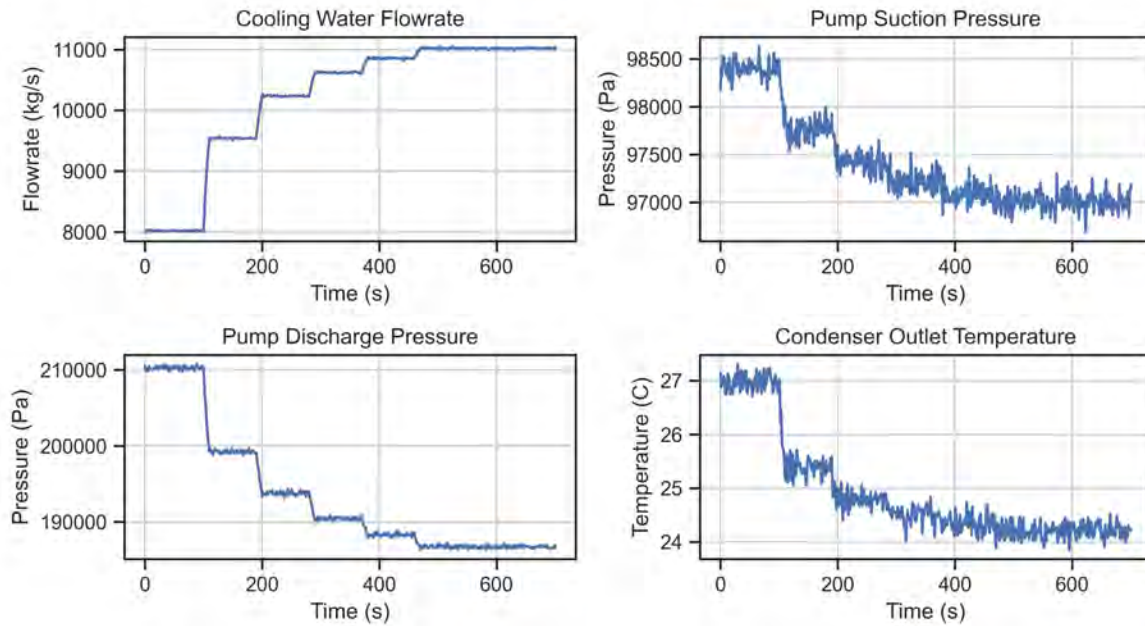


Figure 41: Main operating variables during the startup ramp.

4.8.2 Fault Scenario Simulations

To demonstrate the diagnostic capabilities of the PBD framework, three fault scenarios were simulated: trash rack fouling, waterbox fouling, and a pump discharge pressure sensor fault. These scenarios represent both component degradation (fouling) and sensor failures, enabling evaluation of the framework's ability to distinguish between different fault types and locations.

4.8.3 Trash Rack Fouling

The trash rack fouling fault was simulated by imposing a step increase in the component's pressure-loss coefficient. The results are shown in Figure 42. The increased pressure loss in the suction line reduced both the pump suction pressure and the pump flow rate, which in turn lowered the pump discharge pressure. The condenser outlet water temperature increased as a consequence of the reduced cooling capacity.

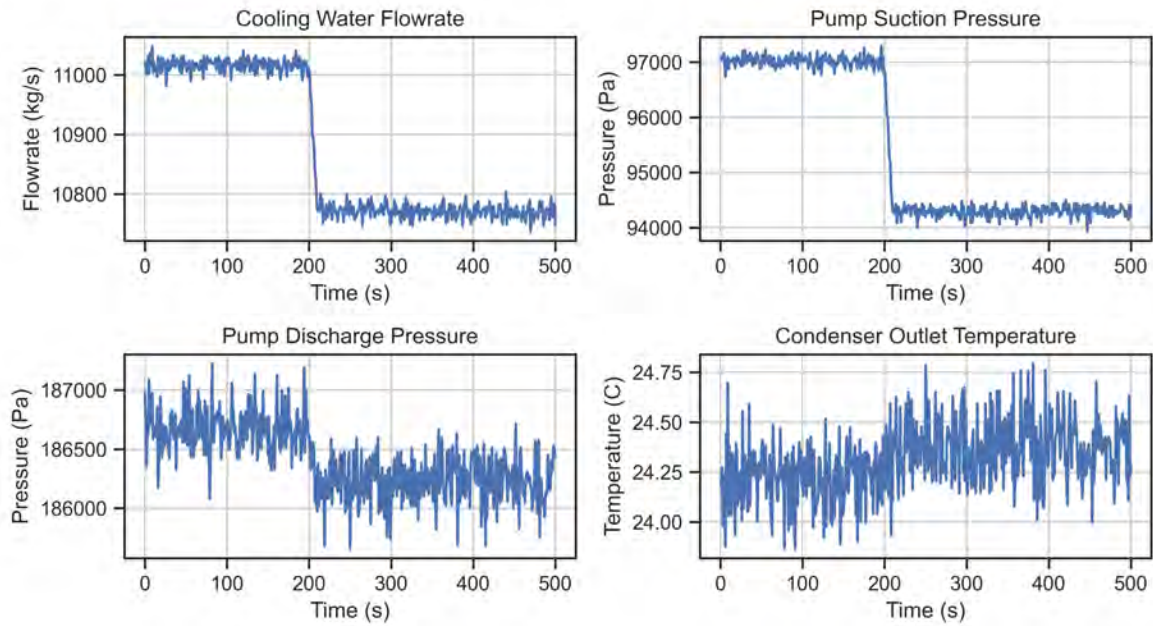


Figure 42: Main operating variables during the simulated trash rack fouling scenario.

4.8.4 Waterbox Fouling

Similarly, results for the waterbox fouling scenario are shown in Figure 43, where the fault was simulated by applying a step increase in the waterbox pressure-loss coefficient. The primary effect of the fault was a reduction in the cooling-water flow rate. Secondary effects included increases in both the pump suction and discharge pressures, as well as an increase in the condenser outlet-water temperature.

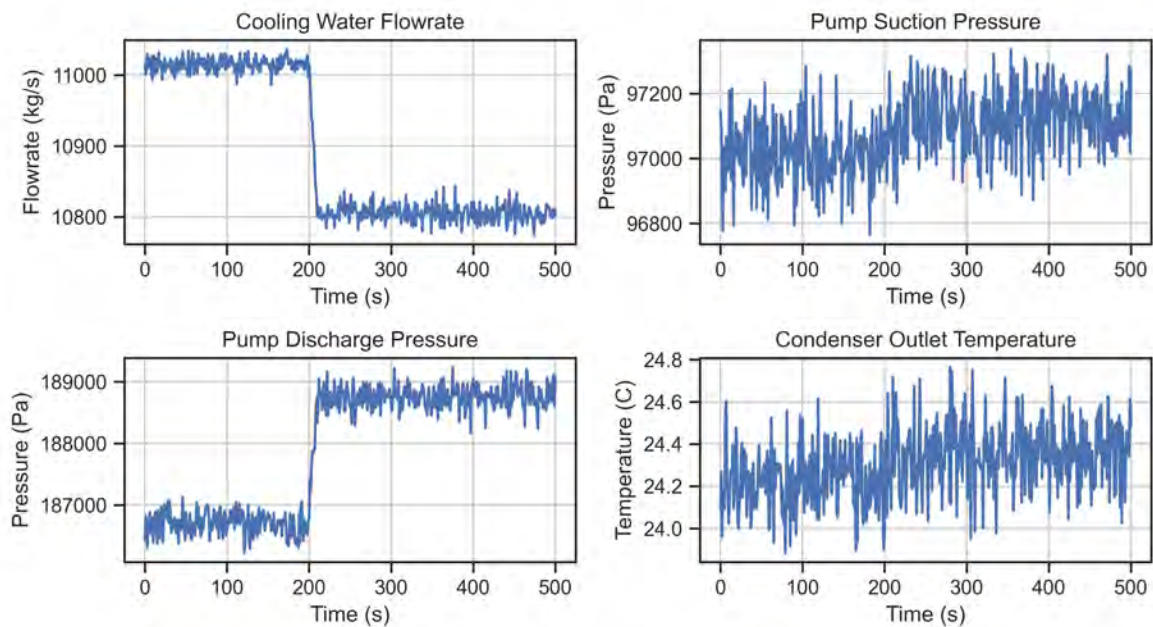


Figure 43: Main operating variables during the simulated waterbox fouling scenario.

4.8.5 CWP Discharge Pressure Sensor Fault

As shown in Figure 44, a sensor fault scenario was simulated by injecting a bias into the pump discharge pressure sensor reading, while all other sensors remained unaffected during full-flow operation. This scenario is used to demonstrate the capability of the diagnostic framework to detect and differentiate between sensor faults and component faults.

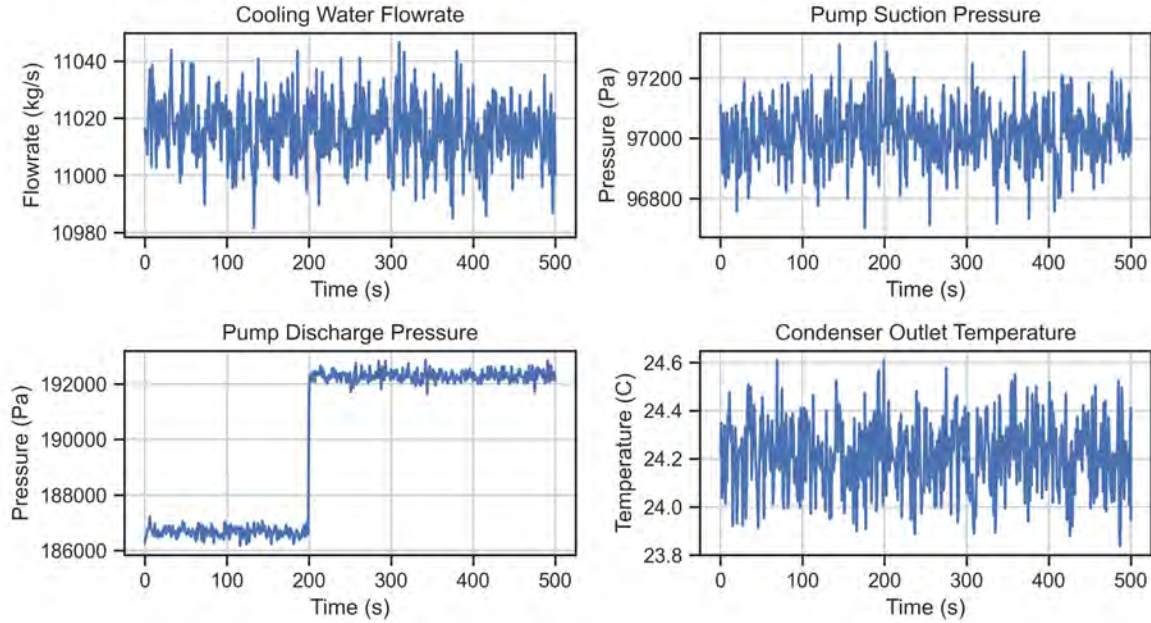


Figure 44: Main operating variables during the simulated discharge pressure sensor fault scenario.

Having established the simulation framework and generated fault scenarios in the preceding section, we now turn to applying the physics-based diagnostic framework to analyze these faults and demonstrate the framework's diagnostic capabilities. The following section presents how the ARR-based residuals respond to each fault, how the Bayesian inference engine ranks fault candidates, and how the LLM-enhanced explainability system communicates diagnostic reasoning to operators.

4.9 Fault Diagnosis Results

This section presents the diagnostic results obtained by applying the PBD framework to the three fault scenarios described in Section 4.8. The diagnostic framework evaluates residuals generated from the ARRs, identifies active residual patterns (fault signatures), and computes posterior fault probabilities using Bayesian inference. For each scenario, we compare the performance of the two sensor configurations and examine the framework's ability to isolate the correct fault from among all possible candidates.

To provide the required inputs for the probabilistic diagnostic framework, prior probabilities of potential faults were estimated from historical CWS maintenance data reported in [210]. For fault types not documented in that study (rack fouling, leakage, and sensor faults), nominal prior probability values of 5.0% and 0.5% were assigned based on engineering judgment.

For each fault scenario below we present the quantitative diagnostic results (posterior fault probabilities). For the trash rack fouling and waterbox fouling scenarios, we include agent outputs demonstrating

Table 1: Estimated Prior Probabilities for Different Fault Types

Fault Type	Prior Probability (%)
Waterbox Fouling	20.3
Rack Fouling	5.0
Sensor Fault	5.0
Pump Misalignment	2.3
Leakage	0.5
Pump Diffuser	0.2

the explainability capabilities through the `fault_query()` and `custom_query()` functions. The agent’s responses illustrate how the physics-based causal relationships enable clear, grounded explanations that help operators understand the diagnostic reasoning.

Before examining the detailed results, we introduce the nomenclature used in the diagnostic outputs. The framework defines four residuals based on the ARRs described in Section 4.7:

- **r1:** Pump performance residual (monitors pump head vs. flow rate)
- **r2:** Condenser heat transfer residual (monitors heat transfer coefficient)
- **r3:** Suction line pressure loss residual (inlet to pump suction)
- **r4:** Discharge line pressure loss residual (pump discharge to outlet)

Each potential fault is assigned a unique identifier (F1–F15) and is associated with a specific set of residuals that would activate if that fault occurred. This mapping between faults and their expected residual signatures forms the basis for both deterministic fault isolation and probabilistic ranking.

4.9.1 Fault Scenario 1: Trash Rack Fouling

In this scenario, using the sensor configuration that included the water flow-rate sensor, the probabilistic framework correctly identified the fault as the sole most likely cause, as listed in Table 2. The fault had the highest posterior probability, clearly distinguishing it from other potential faults.

For the sensor configuration without the flow-rate sensor, the virtual flow-rate sensor remained valid because trash rack fouling does not affect the pump performance curve or its constituent pressure measurements. This configuration achieved the same top-ranked diagnosis, correctly identifying the rack fouling fault without ambiguity.

Table 2: Physics-Based Diagnostic Results for Fault Scenario 1 (Probabilistic Method)

Rank	Fault Name	Probability
1	Trash rack fouling	0.72271
2	CWP inlet pressure sensor fault	0.01366
3	Waterbox fouling	0.00272
4	CWP leakage	0.00084
5	Valve blockage	0.00067
6	Condenser tube side inlet temperature sensor fault	0.00059
7	Condenser tube side outlet temperature sensor fault	0.00059
8	Condenser shell pressure sensor fault	0.00059
9	CWP pump fault (performance degradation)	0.00049
10	Valve leakage	0.00007
11	Condenser shell side leakage	0.00006
12	CWP outlet pressure sensor fault	0.00002
13	Condenser fouling	0.00001
14	CWP outlet flow rate sensor fault	0.00000
15	Condenser tube side leakage	0.00000

To demonstrate the explainability capabilities of the LLM-integrated diagnostic framework, we query the agent to explain the diagnosis. Listing 1 shows the agent’s response to a `fault_query()`, which identifies the fault from the observed residual signature and indicates which sensors are involved.

```

1  [agent.fault_query()]
2  [Output]
3  The fault signature identified is **F12** which corresponds to the
   fault **Trash rack fouling**.
4
5  The residuals identified by this fault are:
6  - **r3**: Residual from pressure loss model between system intake and
   CWP inlet, which relies on the sensors:
7  + System intake pressure
8  + CWP inlet pressure (suction)
9  + Virtual sensor (flow rate from pressure measurements)
10
11 The unique set of sensors involved in these residuals are: CWP inlet
   pressure (suction), System intake pressure

```

Listing 1: Output of a fault query for Scenario 1.

The agent correctly identifies that only residual r3 is active, implicating fault F12 (Trash rack fouling). To understand how the framework ruled out other potential faults, we issue a `custom_query()` asking the agent to explain the exoneration process, shown in Listing 2.

```

1  [agent.custom_query('Explain why the other faults were exonerated.')]
2  [Output]
3  The diagnostic process works by matching the observed pattern of active
   residuals---the fault signature---to the unique pattern that each
   specific fault is expected to produce.
4

```

In this case, the observed signature was that **only residual r3 was active**, while r1, r2, and r4 were inactive.

Why F12 (Trash rack fouling) was identified:

- **Expected Signature for F12:** According to the dependency list r3: [F12, F2, F3, F5], fault **F12** is only associated with residual **r3**. Therefore, if **F12** occurs, it is expected to activate **r3** and nothing else.
- **Observed Signature:** **r3** was active, and all other residuals were inactive.
- **Conclusion:** The observed signature perfectly matched the expected signature for **F12**, leading to its identification.

Why other faults were exonerated:

The other faults were ruled out because the observed signature did not match their expected signatures.

1. Faults associated with r1:

- **Faults:** **F1** (CWP pump fault - performance degradation), **F3** (CWP inlet pressure sensor fault)
- **Reason for Exoneration:** These faults would have caused residual **r1** to become active. Since **r1** was inactive, these faults were ruled out.

2. Faults associated with r2:

- **Faults:** **F7** (Condenser shell side leakage), **F8** (Condenser fouling), **F9** (Condenser tube side inlet temperature sensor fault), **F10** (Condenser tube side outlet temperature sensor fault), **F11** (Condenser shell pressure sensor fault)
- **Reason for Exoneration:** These faults are all linked to residual **r2**. Since **r2** was inactive, these faults were ruled out.

3. Faults associated with r4:

- **Faults:** **F13** (Valve leakage), **F14** (Waterbox fouling), **F15** (Valve blockage)
- **Reason for Exoneration:** These faults would have activated residual **r4**. Since **r4** was inactive, they were ruled out.

4. Faults with more complex signatures:

Some faults are linked to multiple residuals. They were exonerated because their complete signature did not match the observation.

- **F2 (CWP leakage):** Expected to activate **r1** and **r3**. We observed **r3** was active, but **r1** was inactive. This mismatch exonerated **F2**.
- **F4 (CWP outlet pressure sensor fault):** Expected to activate **r1** and **r4**. Both **r1** and **r4** were inactive, ruling out **F4**.
- **F5 (CWP outlet flow rate sensor fault):** This sensor fault would have affected all four residuals (r1, r2, r3, r4). Since only **r3** was active, **F5** was clearly exonerated.

```

36 - **F6 (Condenser tube side leakage):** Expected to activate **r2** and
37   **r4**. Both were inactive, ruling out **F6**.
38
   In summary, every fault except **F12** would have produced a different
   pattern of active residuals. The unique observation that only **r3**
   was active allowed the system to confidently isolate the problem to
   **F12**.

```

Listing 2: Output of a custom query for Scenario 1.

This scenario demonstrates the ideal case for model-based diagnostics: the observed fault signature (r3 active, all others inactive) uniquely identifies the trash rack fouling fault. The LLM agent effectively communicates the diagnostic reasoning by explaining how the physics-based dependency structure enables systematic elimination of all other fault candidates. For an operator, this explanation provides confidence in the diagnosis and actionable guidance: the fault is localized to the trash rack (upstream of the pump suction), and the implicated sensors (suction pressure and ambient pressure) should be verified to rule out sensor failures. This represents a significant advantage over black-box diagnostic methods, which would only provide a fault label without the underlying causal reasoning.

4.9.2 Fault Scenario 2: Waterbox Fouling

In this scenario, the deterministic diagnosis would indicate multiple possible faults, including waterbox fouling, valve blockage, and valve leakage, that could explain the observed symptoms. The probabilistic framework, as shown in Table 3, correctly identified waterbox fouling as the most likely fault, based on its highest posterior probability. This outcome reflects the prior knowledge incorporated into the framework, namely that waterbox fouling is the most common fault in the system.

As in Scenario 1, the virtual flow-rate sensor remained valid since waterbox fouling does not corrupt the pump performance curve or pressure measurements, allowing both sensor configurations to achieve the same top-ranked diagnosis.

Table 3: Diagnostic Results for Fault Scenario 2

Rank	Fault Name	Probability
1	Waterbox fouling	0.79326
2	Valve blockage	0.19538
3	Trash rack fouling	0.02230
4	Valve leakage	0.01954
5	Condenser fouling	0.00244
6	CWP outlet pressure sensor fault	0.00213
7	Condenser tube side inlet temperature sensor fault	0.00072
8	Condenser tube side outlet temperature sensor fault	0.00072
9	Condenser shell pressure sensor fault	0.00072
10	CWP pump fault (performance degradation)	0.00025
11	Condenser tube side leakage	0.00024
12	Condenser shell side leakage	0.00007
13	CWP leakage	0.00003
14	CWP inlet pressure sensor fault	0.00001
15	CWP outlet flow rate sensor fault	0.00000

Unlike Scenario 1, where the fault signature uniquely identified a single fault, this scenario presents a more challenging case where multiple faults share the same residual signature. We query the LLM agent to examine how the framework handles this ambiguity, shown in Listing 3.

```

1  [agent.fault_query()]
2  [Output]
3  The fault signature identified is **[F13, F14, F15]** which corresponds
   to the faults **Valve leakage, Waterbox fouling, Valve blockage**.
4
5  The residuals identified by this fault are:
6  - **r4**: Residual from pressure loss model between CWP outlet and
   system discharge, which relies on the sensors:
7  + CWP outlet pressure (discharge)
8  + System discharge pressure
9  + CWP outlet flow rate
10 + Valve opening
11
12 The unique set of sensors involved in these residuals are: CWP outlet
   pressure (discharge), System discharge pressure, CWP outlet flow
   rate, Valve opening

```

Listing 3: Output of a fault query for Scenario 2.

The agent identifies that only residual r4 is active, which is consistent with three possible faults (F13, F14, F15). While the probabilistic framework ranks waterbox fouling highest based on prior probabilities, it is instructive to see how the agent explains the systematic elimination of other faults. Listing 4 shows the response to a query about the exoneration process.

```

1  [agent.custom_query('Explain why the other faults were exonerated.')]
2  [Output]
3  The diagnostic system operates on a principle of elimination. A fault
   is exonerated if a residual that is designed to detect it remains
   inactive. In this case, the observed residual pattern was:
4  - **r1** = Inactive (0)
5  - **r2** = Inactive (0)
6  - **r3** = Inactive (0)
7  - **r4** = Active (1)
8
9  Let's examine the inactive residuals and the faults they rule out:
10
11 **1. Analysis of Inactive Residual r1:**
12 - **Dependency:** **r1** is linked to faults [F1, F2, F3, F4, F5].
13 - **Explanation:** Residual **r1** monitors the pump head of the CWP.
   Since **r1** is inactive, it indicates that the pump's performance
   and its directly related sensors are functioning as expected.
14 - **Exonerated Faults:** This exonerates:
15 + **F1**: CWP pump fault (performance degradation)
16 + **F2**: CWP leakage
17 + **F3**: CWP inlet pressure sensor fault
18 + **F4**: CWP outlet pressure sensor fault
19 + **F5**: CWP outlet flow rate sensor fault
20
21 **2. Analysis of Inactive Residual r2:**

```

```

22 - Dependency: r2 is linked to faults [F6, F7, F8, F5, F9, F10,
    F11].
23 - Explanation: Residual r2 monitors the heat transfer process
    within the Condenser. Its inactivity suggests that there are no
    significant heat transfer issues or faults with the sensors
    measuring temperature and flow for this process.
24 - Exonerated Faults: This exonerates:
25 + F6: Condenser tube side leakage
26 + F7: Condenser shell side leakage
27 + F8: Condenser fouling
28 + F9: Condenser tube side inlet temperature sensor fault
29 + F10: Condenser tube side outlet temperature sensor fault
30 + F11: Condenser shell pressure sensor fault
31 + (It also provides further evidence against F5)
32
33 3. Analysis of Inactive Residual r3:
34 - Dependency: r3 is linked to faults [F12, F2, F3, F5].
35 - Explanation: Residual r3 monitors the pressure loss between
    the river inlet and the pump inlet, which is where the trash rack is
    located. Because r3 is inactive, it rules out any significant
    blockage or fouling in this section.
36 - Exonerated Faults: This exonerates:
37 + F12: Trash rack fouling
38 + (It also provides further evidence against F2, F3, and F5)
39
40 Conclusion:
41 The only residual that was active was r4. The faults identified
    (F13, F14, F15) are all associated with r4 and are
    not linked to any of the inactive residuals (r1, r2, r3). Therefore,
    the combination of three inactive residuals systematically ruled out
    all other possible faults, leaving only those related to the
    pressure loss downstream of the pump as the confirmed diagnosis.

```

Listing 4: Output of a custom query for Scenario 2.

This scenario illustrates a more realistic diagnostic challenge: when the residual signature does not uniquely identify a single fault, the deterministic framework can only narrow the candidates to those sharing the same signature. The probabilistic extension then leverages prior knowledge about fault frequencies to rank the candidates, correctly identifying waterbox fouling as most likely. The LLM agent’s explanation is particularly valuable here, as it clarifies the distinction between deterministic exoneration (ruling out faults based on inactive residuals) and probabilistic ranking (ordering remaining candidates by prior probabilities). This two-stage reasoning—elimination followed by ranking—provides operators with both diagnostic precision (narrowed to three candidates) and practical guidance (waterbox fouling is most likely based on historical data).

4.10 Fault Scenario 3: Pump Discharge Pressure Sensor Fault

This scenario demonstrates a limitation of virtual sensors. With the sensor configuration that includes the water flow-rate sensor, the probabilistic framework correctly identified the sensor fault as the sole most likely cause, as shown in Table 4. However, when relying on a virtual flow-rate sensor derived from pump pressure measurements, the diagnostic performance degrades significantly, as the fault corrupts the very

measurements used to compute the virtual sensor.

Table 4: Diagnostic Results for Fault Scenario 3

Rank	Fault Name	Probability
1	CWP outlet pressure sensor fault	0.86016
2	Waterbox fouling	0.28565
3	CWP pump fault (performance degradation)	0.10072
4	Valve blockage	0.07036
5	Trash rack fouling	0.02216
6	CWP leakage	0.01072
7	Valve leakage	0.00704
8	CWP inlet pressure sensor fault	0.00277
9	Condenser fouling	0.00089
10	Condenser tube side inlet temperature sensor fault	0.00065
11	Condenser tube side outlet temperature sensor fault	0.00065
12	Condenser shell pressure sensor fault	0.00065
13	CWP outlet flow rate sensor fault	0.00012
14	Condenser tube side leakage	0.00009
15	Condenser shell side leakage	0.00006

For the sensor configuration without the water flow-rate sensor, which relies on the virtual flow-rate sensor derived from pump pressure measurements, the diagnostic performance degrades significantly. Table 5 shows the results for this configuration.

Table 5: Diagnostic Results for Fault Scenario 3 Without Flow-Rate Sensor

Rank	Fault Name	Probability
1	CWP inlet pressure sensor fault	0.47952
2	CWP outlet pressure sensor fault	0.41104
3	Waterbox fouling	0.21032
4	CWP pump fault (performance degradation)	0.12023
5	Trash rack fouling	0.07864
6	Condenser fouling	0.06485
7	Condenser tube side outlet temperature sensor fault	0.05333
8	Condenser tube side inlet temperature sensor fault	0.05333
9	Condenser shell pressure sensor fault	0.05333
10	Valve blockage	0.05180
11	Condenser tube side leakage	0.00648
12	CWP leakage	0.00648
13	Condenser shell side leakage	0.00533
14	Valve leakage	0.00518

The degraded performance without a direct flow-rate sensor is evident: the correct fault (pump discharge pressure sensor, F4) ranks only second with 41% probability, while the pump suction pressure sensor (F3) incorrectly ranks first with 48% probability. This ambiguity arises because the virtual flow-rate sensor is computed from both pressure sensors using the pump performance curve. When the discharge pressure sensor fails, it corrupts the virtual flow-rate calculation, making it impossible to distinguish whether the fault lies in the discharge pressure sensor, the suction pressure sensor, or the pump itself. All three faults would produce

similar distortions in the virtual flow rate, resulting in indistinguishable residual patterns. This fundamental limitation reveals an important constraint on virtual sensor use: while they are valuable for reducing instrumentation costs, virtual sensors cannot reliably diagnose faults in their constituent measurements. Direct physical measurements remain essential for complete diagnostic coverage in safety-critical systems.

4.10.1 Discussion of LLM Explainability

The LLM agent outputs presented for Scenarios 1 and 2 demonstrate several key advantages of integrating natural language explainability with physics-based diagnostics:

- **Causal Reasoning Communication:** The LLM agent effectively translates the underlying physics-based dependency structures into clear explanations accessible to operators. In Scenario 1, the agent explains that trash rack fouling (F12) is “only associated with residual r3” and therefore produces a unique signature when “r3 is active and all other residuals are inactive.” This direct mapping from observed symptoms to fault causes, grounded in the ARR structure, provides operators with confidence that the diagnosis is based on fundamental physical relationships rather than statistical correlations.
- **Systematic Exoneration Logic:** Perhaps the most valuable aspect of the LLM explanations is the systematic presentation of why other faults were ruled out. In both scenarios, the agent methodically explains how inactive residuals eliminate entire classes of faults. For example, in Scenario 2, the agent explains: “Residual r2 monitors the heat transfer process within the Condenser. Its inactivity suggests that there are no significant heat transfer issues,” thereby exonerating six condenser-related faults (F6–F11). This exhaustive reasoning process helps operators understand not just the diagnosis, but the entire logical pathway that led to it, including what was considered and rejected.
- **Sensor and Component Traceability:** The agent outputs explicitly list which sensors are involved in each active residual, providing actionable guidance for operators. For example, in Scenario 1, the agent identifies “CWP inlet pressure (suction), System intake pressure” as the unique set of sensors involved in the trash rack fouling signature. This directs operators to verify these specific sensors to rule out sensor failures, a critical step before committing to maintenance actions based on component fault diagnoses.

4.11 Physics-based Discussion and Conclusion

The persistent challenge of biofouling and debris accumulation in the CWS has imposed significant operational and economic burdens, including unplanned power reductions of up to 340 MWe and costly reactive maintenance interventions. The transition from time-based preventive maintenance to condition-based maintenance demands diagnostic capabilities that not only detect degradation early but also explain the underlying physical causes to support informed operator decision-making. Building on the foundation established by INL’s Advanced Pattern Recognition techniques and machine learning models for CWS fault detection, this work demonstrates that physics-based diagnostics provide a complementary capability: the ability to trace observed symptoms through causal chains of physical relationships to specific component faults or sensor failures, offering a causal explanation for observed anomalies that allows operators to understand the physical mechanisms behind a fault.

The diagnostic framework developed in this work successfully detected and isolated three distinct fault scenarios using physics-based ARRs derived from conservation laws and constitutive equations. For trash rack fouling, the framework achieved unique fault identification with 72% posterior probability based on a

distinctive residual signature. For waterbox fouling, despite residual signature ambiguity shared by three candidates, the probabilistic extension correctly identified the fault with 79% probability by leveraging historical maintenance data. The pump discharge pressure sensor fault scenario revealed an important limitation: virtual sensors derived from constituent measurements cannot reliably diagnose faults in those same measurements, underscoring the continued need for optimal instrumentation in safety-critical applications. The integration of LLMs to generate natural language explanations demonstrated that physics-based causal reasoning can be made accessible to operators, systematically communicating both the positive evidence supporting a diagnosis and the elimination logic ruling out alternative hypotheses.

This work establishes that physics-based diagnostics offer valuable capabilities for condition-based maintenance of nuclear power plant CWS, particularly in providing transparent, explainable fault diagnoses grounded in fundamental thermal-hydraulic principles. The approach extends the diagnostic toolset available to plant operators by complementing pattern recognition methods with causal reasoning, enabling more confident maintenance decisions when multiple fault candidates share similar symptoms. Future work should address several key challenges: validation using operational plant data to assess real-world diagnostic performance under measurement noise and modeling uncertainty; extension to multiple simultaneous faults and cascading failure scenarios; development of uncertainty quantification methods to help LLM explanations appropriately express diagnostic confidence; and integration of these physics-based diagnostic models with the existing INL risk assessment framework to enable economic optimization of maintenance scheduling strategies across the full spectrum of CWS degradation mechanisms.

5 BENCHMARKS

In this section, we elaborate on the two benchmarks used in our study of multimodal anomaly detection using foundation models. These benchmarks were derived from real-world systems and provide robust datasets for evaluating the performance and effectiveness of our models. By examining these datasets, we aim to demonstrate how our approach can be applied to practical scenarios, highlighting its strengths and potential areas for improvement.

5.1 Skoltech Anomaly Benchmark (SKAB)

SKAB [153] is an industrial real-world benchmark for evaluating anomaly detection algorithms on multivariate time series sensor data. Developed by a team of researchers at Skolkovo Institute of Science and Technology (Skoltech), it reflects real engineering problems and realistic varying levels of noisy data—unlike synthetic, human-made datasets and benchmarks. The dataset encompasses several sensor data from a testbed for a water pump system’s simultaneous data streams. The sensor data include:

- Vibration: Accelerometer1RMS, Accelerometer2RMS [$\frac{m}{s^2}$]
- Motor current: Current [A]
- Pressure: Pressure [bar]
- Motor body temperature: Temperature [$^{\circ}\text{C}$]
- Fluid temperature: Thermocouple [$^{\circ}\text{C}$]
- Motor voltage: Voltage [V]
- Flow rate: RateRMS [$\frac{L}{min}$]

Across the more than 30 experiments, Skoltech engineers triggered a diverse set of degradations, including progressive pump cavitation, cooler inefficiency, partial pipeline blockage, and controlled leaks. Exploratory data analysis highlights how different fault modes manifest through distinct temporal signatures: cavitation drives high-frequency vibration and current spikes, while cooler failures induce slower drifts in body and fluid temperatures. The combination of fast-changing and slowly drifting channels makes SKAB suitable for both point-wise detectors and window-based methods that model gradual dynamics.

Labels are provided in two complementary formats. The `labels.csv` file marks the start and end indices of anomalous regimes for each experiment, while the per-sample annotations flag whether a specific timestamp should be treated as anomalous. This enables training strategies that split each experiment into a healthy “baseline” segment for unsupervised learning and a fault segment for validation. There are two types of labeled anomalies:

- **Point Anomalies:** isolated outliers that reflect abrupt spikes or dips at single timestamps
- **Changepoints:** anomalous patterns that extend over subsequences of data

The testbed is a closed-loop water pump, meaning a water-circulation system with monitoring and control, built using Industrial Internet of Things technologies.

The stand consists of the following systems:

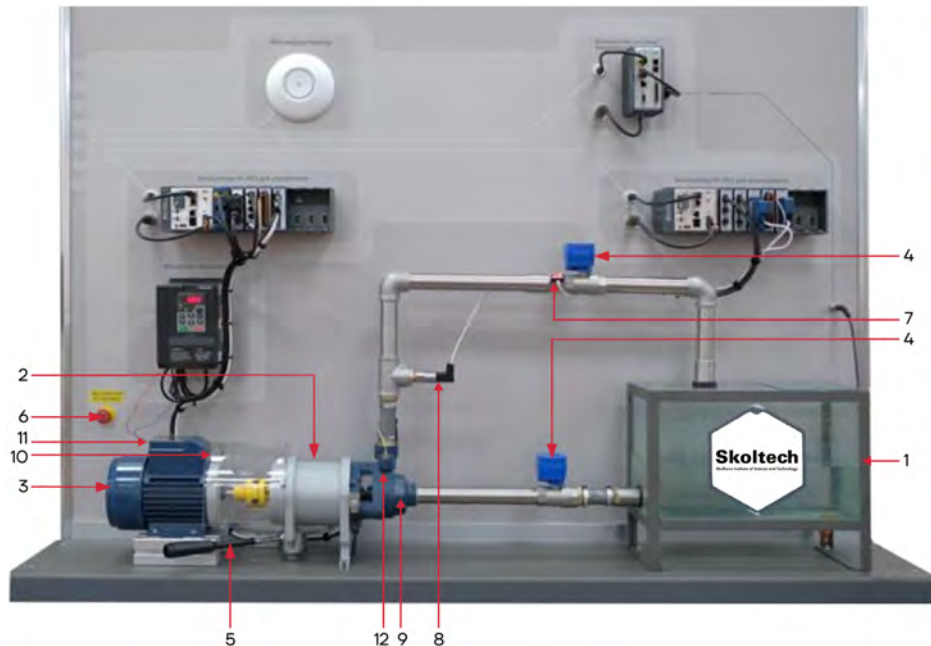


Figure 45: Water-cooling pump testbed used to acquire the SKAB dataset. The rig contains the centrifugal pump, reservoir, flow and pressure sensors, and controlled valves for injecting faults.

1. Water-circulation system
2. Control system for the circulation loop
3. Condition-monitoring system for the circulation loop
4. Time-Sensitive Networking demonstration system
5. Data collection, processing, and visualization system.

The circulation system emulates a water-supply system. A storage tank holds the water, and the pump circulates water through the pipes. The circulation system can simulate the following faults:

- Introducing imbalance on the coupling shaft between the motor and the water pump
- Changing the valve capacity at the pump inlet
- Changing the valve capacity at the pump outlet.

5.2 Industrial Utility Benchmark

The Industrial Utility Benchmark has a circulating water system (CWS), an important non-safety system that impacts the plant site's power generation capability. During operations, the utility offers a large amount of historical plant process data, preventive/corrective maintenance records, failure data that can be utilized for health monitoring, fault and anomaly detection. The CWS is a heat sink for the main steam turbine and associated auxiliaries and is designed to maximize steam power cycle efficiency while minimizing any adverse impacts to the River. The CWS, shown in Figure 46, consists of the following major components:

1. Six vertical, motor-driven circulating pumps (circulators) with components to remove debris and marine life
2. Main condenser
3. Condenser waterbox air removal system
4. Circulating water sampling system
5. Screen wash system
6. Necessary piping, valves, instrumentation, and supports for operation.

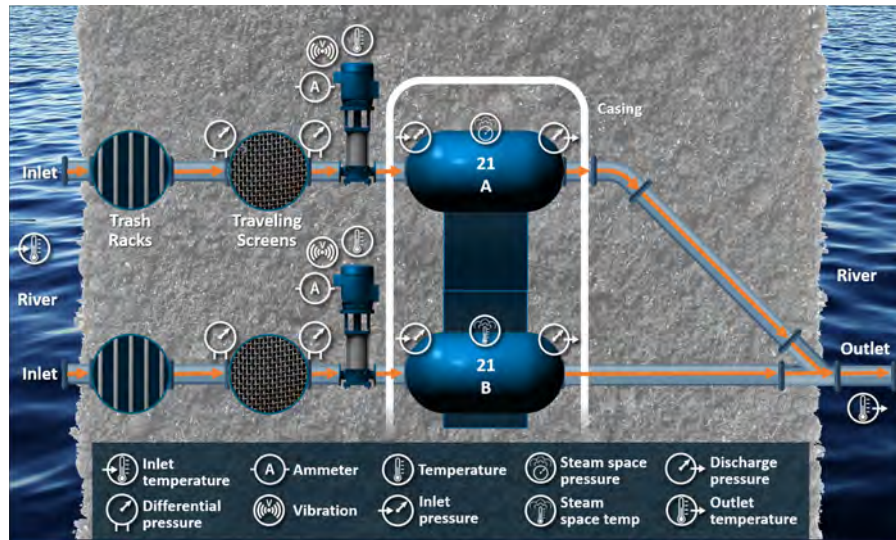


Figure 46: Circulating Water Pump Plant layout.

For condition monitoring, each CWS was instrumented with multiple sensors, as shown in Figure 47. The recorded measurements included the differential temperature, motor current, stator winding temperature, inboard and outboard motor bearing temperatures, and motor axial vibration. These parameters, together with other operational data, were gathered at the utility-operated NPP over the periods of 2009–2020 and 2023–2024 in order to investigate various fault-induced anomalies.

A concise summary of the data is provided here; comprehensive details regarding the CWS and its associated datasets can be found in [211].

Both datasets underwent the same preprocessing and synchronization pipeline, using a time-indexed framework. Signals were aligned according to timestamps, with any duplicates being removed. Additional engineered features—such as the number of circulating water pumps (CWPs) in operation (NCWPs_{ON}) and the calendar week of the year—were introduced to capture operational context and seasonal variability. Data irregularities and noise were mitigated using physically consistent rules: negative temperature or current readings were set to zero, and statistical outliers were filtered using a z-score cutoff of $|z| > 2.5$. For each signal, the z-score was computed as $z = \frac{x - \mu}{\sigma}$, where μ and σ denote the sample mean and standard deviation after eliminating any nonphysical values.

Points with $|z| > 2.5$ were flagged as outliers and substituted with the median value from the non-outlier samples in that column. Assuming near-Gaussian noise, this threshold identifies approximately the top 1% of sensor errors and spurious spikes. Comparable outlier handling was applied by [211] to eliminate

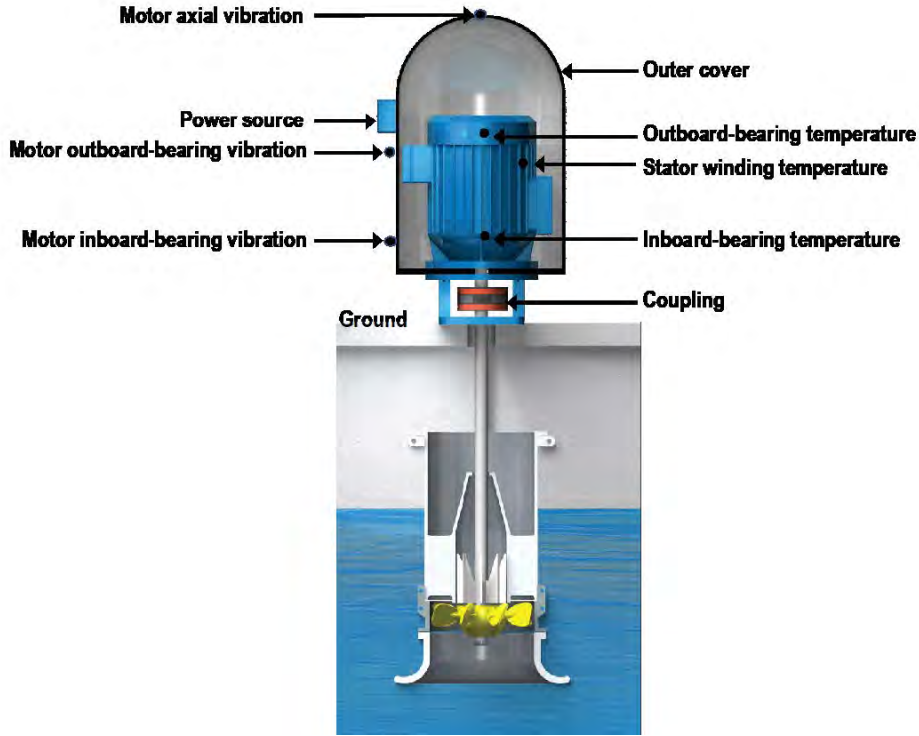


Figure 47: Diagram of a CWP motor, showing sensor placements for vibration and temperature monitoring.

repetitive and non-diagnostic spikes linked to the weekly eSOMS inlet-pressure fluctuations and condenser temperature jumps from CWP transitions. Operational status flags (e.g., ON/START, OFF/TRIP/Shutdown, invalid entries) were converted to numeric representations in the range $[0, 1, NaN]$ for use in model training.

Labels were generated using a 7-day forward window, marking a timestamp as *Healthy* = 0 if the following conditions held:

- $NCWPsON \in 4, 5, 6$
- The specific CWP status equals ON (numeric 1)
- $NCWPsON$ remained non-decreasing over the window.

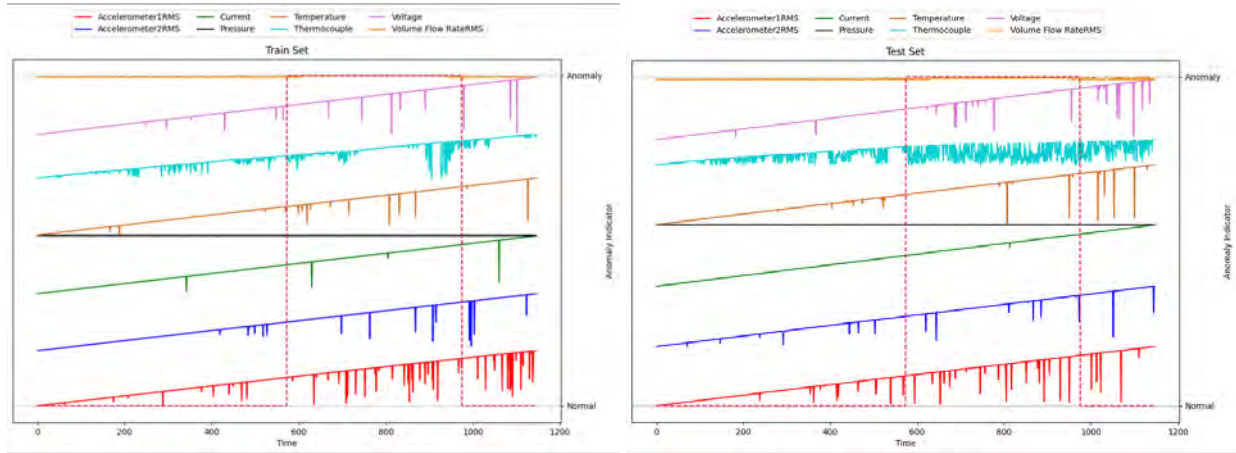
The 7-day window was chosen so as to represent dominant operating cycles while avoiding bias from short-term signal variations. It spans several tidal cycles (approximately 12-hour semi-diurnal oscillations) influencing vibration and process measurements, as well as the weekly eSOMS inlet-pressure adjustments that recur every 7 days. Such fluctuations are considered non-diagnostic, as treating them as faults would introduce false alarms [212, 213]. The restriction to $NCWPsON \in 4, 5, 6$ throughout the labeling window ensures that only steady-state reactor conditions are tagged as healthy. Because each NPP unit operates six CWPs, losing more than two results in power derating or reactor shutdown. Enforcing monotonic $NCWPsON$ avoids labeling data from transitional regimes (e.g., pump dispatch changes), as is consistent with prior findings that abrupt dispatch shifts cause step changes in condenser differential temperature and related variables.

6 RESULTS

In this section, we compare traditional tree-based data-driven techniques such as CatBoost, Random Forest, and XGBoost, which have dominated classification tasks for the last few years, against the Foundation Model TabPFN for anomaly detection. Both single-modality (sensor measurement data) and multimodal (sensor + text data) were used for training and testing. We applied these methods to two benchmarks described in Section 5 so as to classify normal vs. anomalous data for the SKAB water pump system and Industrial Utility Benchmark data. Classification metrics, as defined in Subsection 2.6, were calculated for each dataset so as to compare the different models and analyze how adding a modality affects anomaly detection accuracy.

6.1 SKAB Dataset

This dataset contains multi-variate time series collected from sensors installed on a water pump system testbed. We considered data obtained from experiments by closing the valve at the outlet of the flow from the pump. The anomaly typology was detected through negative offsets over time slots in sensor measurements [214].



(a) Sensor measurements with anomalies (Train Set)

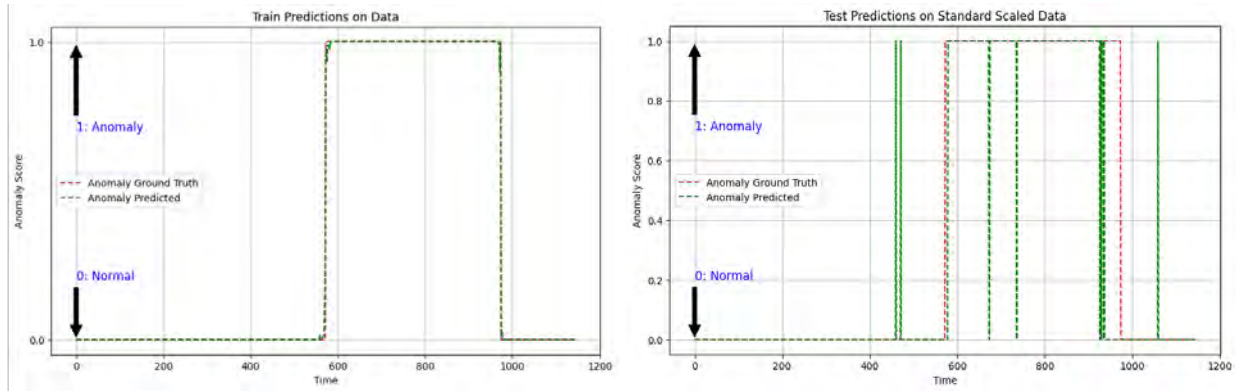
(b) Sensor measurements with anomalies (Test Set)

Figure 48: Acceleration, current, pressure, temperature, voltage, and volume flow rate, as measured for a water pump system in which varying the flow from the outlet valve introduced anomalies in the system’s operations.

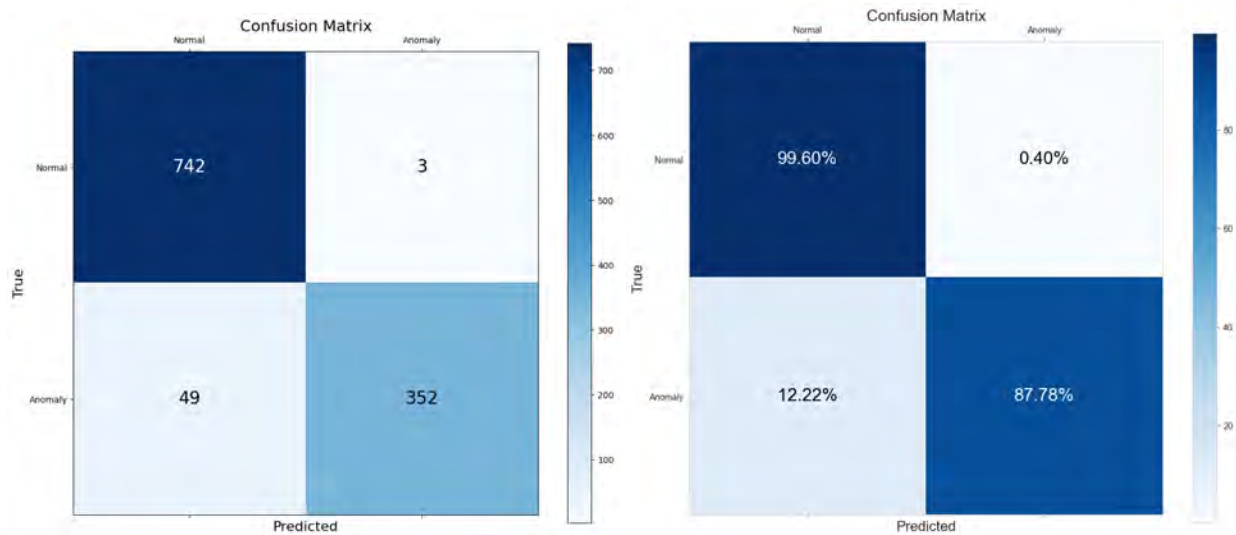
Exploratory data analysis (EDA) showed that standardizing the data improves classification accuracy. The train and test sets were standardized by removing the mean and scaling to unit variance. TabPFN processes tabular data, with each sensor (e.g., Accelerometer1RMS, Current) as a feature in columns, and time steps as rows. The classification target is the Anomaly score, with 0 indicating normal operations and 1 suggesting an anomaly. The TabPFN classifier performs supervised classification and anomaly detection by using training features and targets. As a foundation model, it does not update weights during training. Instead, it uses the training set as prompts or examples, allowing it to directly “learn” from the input sequence, without a new training phase, when presented with new, unseen tabular data.

6.1.1 Single Modality Binary Classification

The TabPFN classifier model was first fit using sensor measurements. Figure 49a displays the classifier's predictions for the training set. It accurately captures both normal and anomalous conditions but exhibits slight ringing or overshoot near sharp transitions from normal to anomalous states. Figure 49b shows an ROC-AUC of 0.9208, indicating a high degree of discrimination ability—meaning that the model is very good at distinguishing between the normal and anomalous classes on the unseen data. At transitions between normal and anomalous conditions, the classifier tends to misclassify a few instances at around time 580. When transitioning from anomalous to normal conditions, it produces false negatives, leading to underestimation of the anomaly durations. It also has a few false positives outside the anomalous region at time 500.



(a) Train data are used for in-context learning in TabPFN. (b) Test predictions results in a ROC-AUC = 0.9208
Figure 49: Single Modality-based anomaly detection results in a high ROC AUC of 0.9208 for unseen data. It has an accuracy of 95.46% but cannot accurately estimate the start of the anomaly.



(a) Off-diagonal terms show 49 false negatives (b) 12.22% of healthy data is misclassified as anomalous
Figure 50: The confusion matrix displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The single-modality TabPFN classifier misclassified 49 (12.22%) anomalous instances as normal and three (0.40%) normal instances as anomalous.

For safety-critical engineering systems, false positives occur when the model detects anomalies despite normal operation, while false negatives happen when actual anomalies go undetected. Genuine threats or defects that can lead to catastrophic failures and accident scenarios are overlooked. Therefore, a key classification metric is Recall, which measures a model’s ability to minimize false negatives. The single-modality TabPFN model achieves a Recall of 95.46%. The lower off-diagonal element of the confusion matrix in Figure 50a indicates 49 (or 12.22%) of the actual anomalous timesteps (Figure 50b) being misclassified as normal, despite indeed being anomalous. In the next section, we analyze classification using multimodal data.

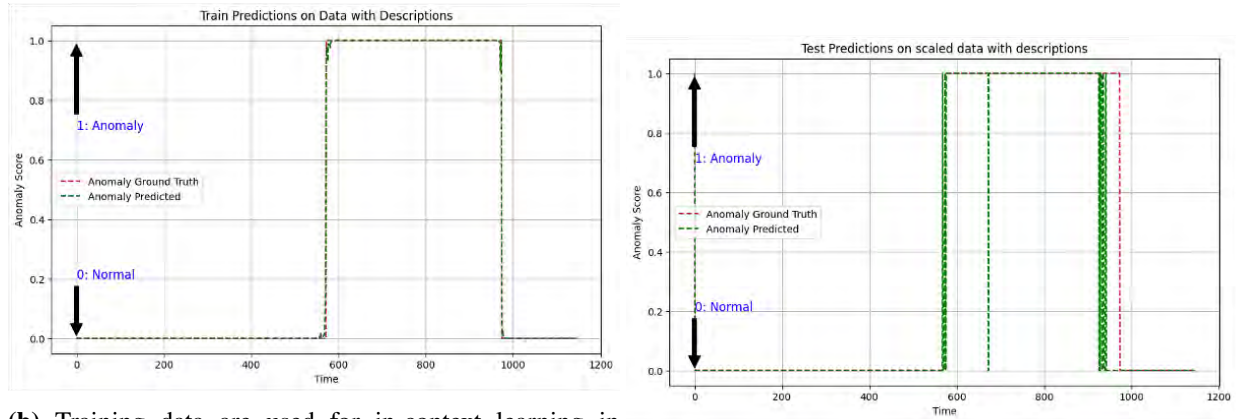
6.1.2 Multimodal Binary Classification

To compare single-modality classification with multimodal classification, we added a column to the tabular data fed into TabPFN, containing textual descriptions of sensor measurements (e.g., pressure fluctuations or process start/end tags), as shown in Figure 51a. These descriptions mimicked operator logbooks without introducing bias, by avoiding terms like “anomaly” or “off-normal conditions.” The training classification results are similar to the single modality results (Figure 51b). The test set predictions have a higher accuracy of 96.42%. This classifier accurately estimates anomaly onset—but underestimates the anomaly end by a smaller duration as compared to the single modality classifier. The Precision score increases to 96.57%, while the Recall is now 96.42%, meaning that the false positives and false negatives are lower in number.

Adding textual descriptions as a feature eliminates false positives at around time 500 in the single-modality case, reducing them from 3 to 2 as is seen in Figure 52a. The multimodal model also reduces false negatives from 49 to 39 (or 9.73%) (Figure 52b), correctly identifying 10 previously misclassified anomalous instances that required operator intervention or system shutdown. Multimodal data offer a more comprehensive and robust understanding of a system’s normal behavior, enabling detection of subtle anomalies that single-source models might miss. By integrating data sources like operator logs and sensor measurements, TabPFN’s attention mechanism captures patterns and correlations, enhancing anomaly detection. In the next section, we compare TabPFN-based anomaly detection against premier ML models for tabular data classification.

	datetime	Accelerometer1RMS	Accelerometer2RMS	Current	Pressure	Temperature	Thermocouple	Voltage	Volume Flow RateRMS	anomaly	change point	description
0	3/9/20 10:34	0.027080	0.039615	0.871339	0.054711	75.4955	25.8338	244.091	32.0000	0	0	Process starts
1	3/9/20 10:34	0.026995	0.038759	1.301280	0.054711	75.5445	25.8408	224.170	32.0000	0	0	Sensor readings increasing
2	3/9/20 10:34	0.026807	0.039521	1.284500	0.382638	75.6607	25.8227	234.157	32.9986	0	0	Sensor readings increasing
3	3/9/20 10:34	0.026817	0.038630	1.364950	0.054711	75.5750	25.8262	229.902	32.9986	0	0	Pressure and Volume Flow RateRMS fluctuations
4	3/9/20 10:34	0.026393	0.038770	0.791839	0.054711	75.4351	25.8382	251.697	32.0015	0	0	Pressure and Volume Flow RateRMS fluctuations

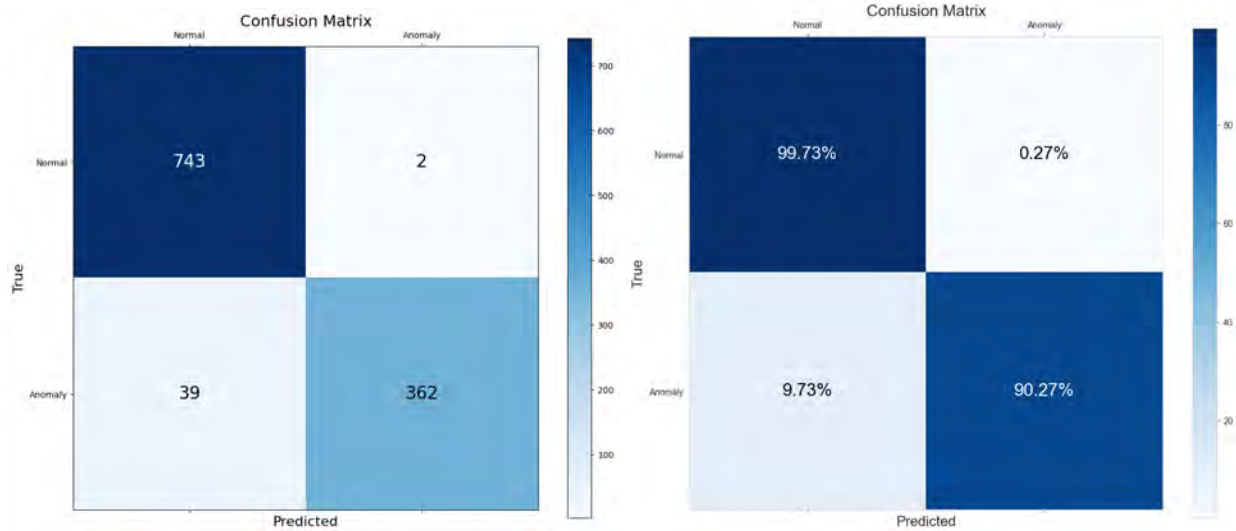
(a) Adding a modality in the form of textual descriptions of sensor readings—without mentioning anomalies or off-normal conditions, to avoid bias.



(b) Training data are used for in-context learning in TabPFN.

(c) Test predictions result in a ROC-AUC of 0.9865.

Figure 51: Adding a modality such as text descriptions along with sensor measurements increases the ROC-AUC from 0.9208 to 0.9865 on test data, with a classifier accuracy of 96.42%. This indicates a lower rate of false positives and shows that it can capture the start of the anomaly despite underestimating the end.



(a) Off-diagonal terms show 39 false negatives

(b) 9.73% of healthy data are misclassified as anomalous.

Figure 52: The confusion matrix displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The Multimodal TabPFN classifier misclassified 39 (9.73%) anomalous instances as normal, and two (0.27%) normal instances as anomalous.

6.1.3 Comparison of Classification Models

CatBoost [215], Random Forest [216], and XGBoost [217] are powerful ML algorithms widely used for classification tasks on tabular data. Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (for classification) of the individual trees. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It builds trees sequentially, with each new tree correcting the errors of the previous ones. CatBoost is a gradient boosting library specifically designed to handle categorical features effectively. Most features fall into two categories: Numerical and Categorical. Numerical data represent a quantity or measurement such as number of items, height, or weight. Categorical data represent a quality or category such as gender, color, or names. CatBoost automatically handles categorical features by converting them into numerical representations without requiring manual preprocessing. However, unlike TabPFN, other models lack an attention mechanism for processing real text or language. To incorporate operator logs or textual descriptions, they require a special encoding step, whereas TabPFN allows for direct addition to a column.

Table 6 presents classification metric scores for TabPFN with single-modality data, multimodal TabPFN, and hyperparameter-tuned Random Forest, CatBoost, and XGBoost classifiers trained and tested on the SKAB dataset. Multimodal TabPFN leads with the highest accuracy, recall, precision, and F1-score. For single-modality data, all models performed comparably, with Random Forest being just 0.26% more accurate than TabPFN. Next, we enhanced CatBoost, XGBoost, and Random Forest by adding textual modality, encoded into numerical representations. An estimate of classification metrics on a single train-test split may be biased due to data peculiarities and can tend to over- or underestimate model performance. Figure 53 illustrates a three-fold cross-validation in which the data are divided into three equal parts and the models are trained three times on different folds, using the remaining two folds for validation each time. The final performance was averaged across all three folds to provide a more reliable estimate of model accuracy than that afforded by a single train-test split. The results show an increase in ROC-AUC for each model when textual modality is added, as indicated by the green bars. For CatBoost, the ROC-AUC rises from 0.9967 to 0.9991, while for Random Forest the ROC-AUC increased from 0.9953 to 0.9987. Similarly, for XGBoost the ROC-AUC increased from 0.9947 to 0.9982, and TabPFN showed an increase from 0.9953 to 0.9995.

This comparison of classification metrics across models highlights the superior performance of multimodal approaches, particularly those leveraging attention mechanisms and foundation models. These models not only achieve higher accuracy, recall, precision, and F1-scores, but also require no retraining, making them robust and efficient for real-world applications.

Table 6: Comparison of classification metrics across different models.

Metric	TabPFN	Multimodal TabPFN	Random Forest	CatBoost	XGBoost
Precision	0.9568	0.9657	0.9591	0.9543	0.9591
Recall	0.9546	0.9642	0.9572	0.9528	0.9563
F1 Score	0.9539	0.9638	0.9566	0.9522	0.9556
Accuracy	0.9546	0.9642	0.9572	0.9528	0.9564
ROC-AUC	0.9208	0.9865	0.9387	0.9367	0.9376

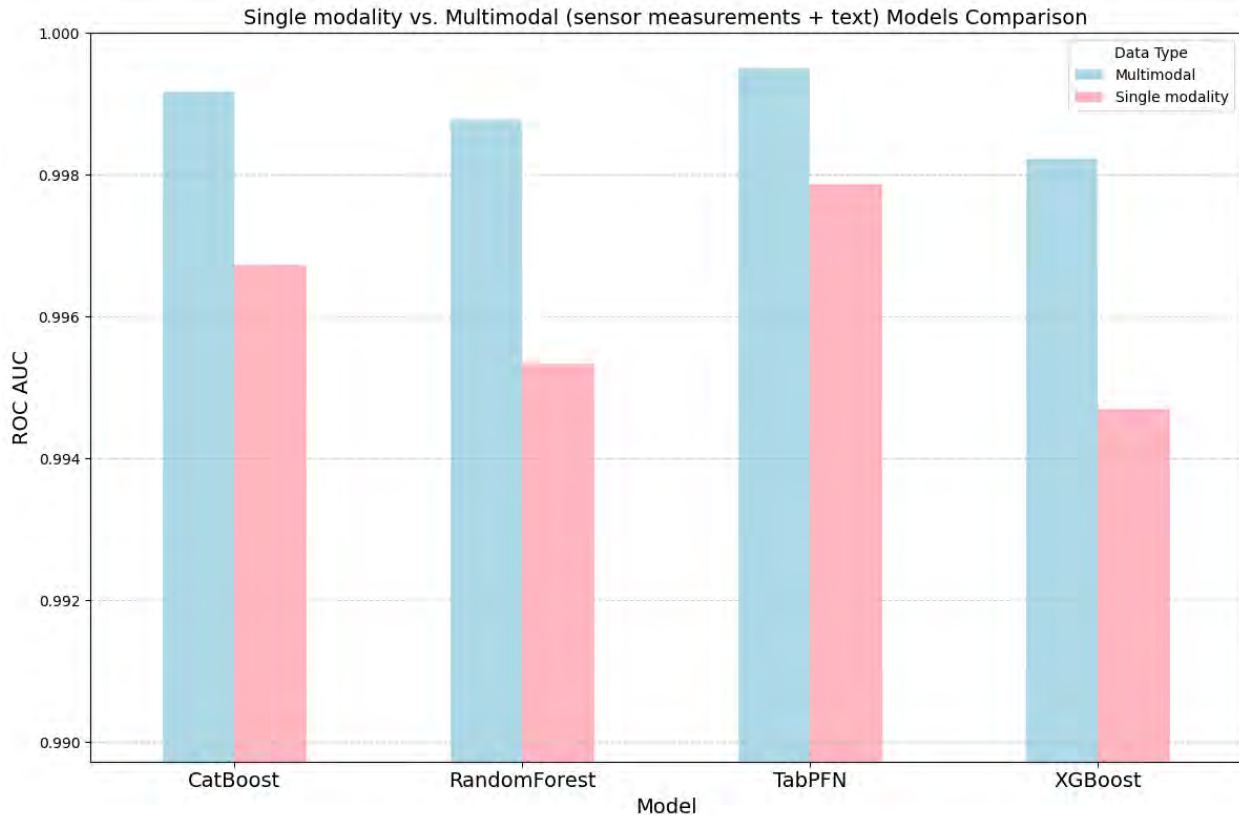


Figure 53: Three-fold cross-validation comparing four models for classifying anomalous vs. normal data. TabPFN achieves the highest ROC-AUC with both single and multimodal data. The additional step of encoding string data is required for CatBoost, Random Forest, and XGBoost. For all four models, adding a modality improves the ROC-AUC for classification.

6.1.4 RAG-Based Additional Features

This section presents a case study on the SKAB valve data so as to evaluate integration of RAG with a LLM into a TabPFN-based anomaly detection pipeline. The objective is to keep the classifier fixed (TabPFN) and compare two feature configurations: a minimal window-based baseline (Baseline A) that uses only raw sensor values, and an extended configuration (Baseline B) that augments engineered sliding-window statistics with additional features derived from domain documentation via a RAG + LLM workflow.

In both configurations, the valve time series is first segmented into overlapping sliding windows of fixed length. In Baseline A, the feature representation is intentionally simple and closely aligned with the original CSV structure. Each window is represented by the raw numeric sensor values at its central timestamp, together with basic window metadata (file identifier and window indices) and the corresponding binary label (normal vs. anomaly), which are obtained by aggregating the SKAB annotations over the samples within the window. No explicit sliding-window statistics, event detection, or external knowledge are incorporated. This configuration serves as a lightweight reference that reflects what can be achieved with minimal preprocessing of the valve time series.

Baseline B uses the same sliding-window segmentation and window-level labeling, but replaces the raw midpoint encoding with a richer set of features. First, for each window and each numeric sensor channel, a collection of summary statistics was computed, including the mean, standard deviation, minimum, maximum,

range, a linear slope estimate across the window, and separate variance estimates for the first and second halves of the window. In addition, a numeric step-change detector was applied to derive window-level event features such as a binary indicator for the presence of a step and a normalized position of the detected step within the window. On top of these numerical features, Baseline B incorporates RAG + LLM-derived attributes that inject information from valve-related documentation. A retrieval index was constructed over textual resources, describing expected operating regimes and failure modes. For each window, a short textual description of the observed sensor behavior was generated and then used as a query to retrieve relevant documentation snippets. The retrieved passages, together with basic window context, were passed to an LLM via a structured prompt that requested a JSON object with a fixed schema. The JSON contained: (i) a categorical hypothesis about the valve state (e.g., throttling, opening, closing, unknown), encoded as one-hot features; (ii) discrete trend indicators for key signals such as flow, pressure, and motor current, also encoded as one-hot features; (iii) event-related descriptors such as the inferred presence and approximate position of a step; and (iv) a scalar consistency score that reflects how well the explanation aligns with the retrieved documentation. These JSON fields were parsed and concatenated with the sliding-window statistics to produce a RAG-enhanced feature matrix that remains directly consumable by TabPFN. Table 7 summarizes the main feature groups used in each baseline configuration.

Table 7: High-level comparison of feature groups used in Baseline A (raw midpoint window features) and Baseline B (engineered sliding-window statistics + RAG/LLM features).

Feature group	Example features (not exhaustive)	Baseline A	Baseline B
Window-level raw midpoint sensor values	Raw numeric channels at the central timestamp of each window, taken directly from the original CSV	✓	–
Sliding-window summary statistics per sensor channel	channel_mean, channel_std, channel_min, channel_max, channel_range, channel_slope, channel_var_left, channel_var_right	–	✓
Window-level step event features (numeric detector)	events_step_exists, events_step_pos	–	✓
LLM valve state hypothesis (one-hot)	hyp_throttling, hyp_opening, hyp_closing, hyp_unknown	–	✓
LLM-derived signal trends (one-hot)	signals.flow_trend_{down, flat, up}, signals.pressure_trend_{down, flat, up}, signals.motor_current_trend_{down, flat, up}	–	✓
LLM-derived event descriptors	has_step, step_pos (from the LLM explanation, distinct from the numeric detector)	–	✓
LLM explanation quality / consistency	consistency_score	–	✓

For both baselines, TabPFN was used as the downstream classifier, and grouped cross-validation was performed with the file identifier as the grouping variable so as to prevent windows from the same valve run from leaking into both training and validation folds. Performance was evaluated using ROC-AUC, accuracy, precision, recall, and F1-score, and the aggregated validation results are summarized in Table 8. Baseline A (raw midpoint window features) achieved an ROC-AUC of 0.9927, accuracy of 0.9174, precision of 0.8987, recall of 0.9006, and F1-score of 0.8913. Baseline B (engineered sliding-window statistics + RAG/LLM features) improved over all these metrics, reaching an ROC-AUC of 0.9979, accuracy of 0.9750, precision of 0.9564, recall of 0.9808, and F1-score of 0.9672.

Table 8: Validation performance of the two TabPFN baselines on the SKAB valve subset. Baseline A uses raw midpoint window features, while Baseline B uses engineered sliding-window statistics augmented with RAG/LLM-derived features.

Baseline	ROC-AUC	Accuracy	Precision	Recall	F1
A	0.9927	0.9174	0.8987	0.9006	0.8913
B	0.9979	0.9750	0.9564	0.9808	0.9672

The confusion matrices in Figure 54 and Figure 55 further illustrate this improvement: compared to Baseline A, Baseline B reduces both the number of normal windows incorrectly flagged as anomalous (from 8 to 6) and the number of missed anomalies (from 2 to 1). This reduction in false negatives is particularly important in an anomaly detection setting and indicates that the combination of engineered statistics and RAG/LLM-derived features provides a more discriminative representation of the valve behavior than do raw midpoint features alone.

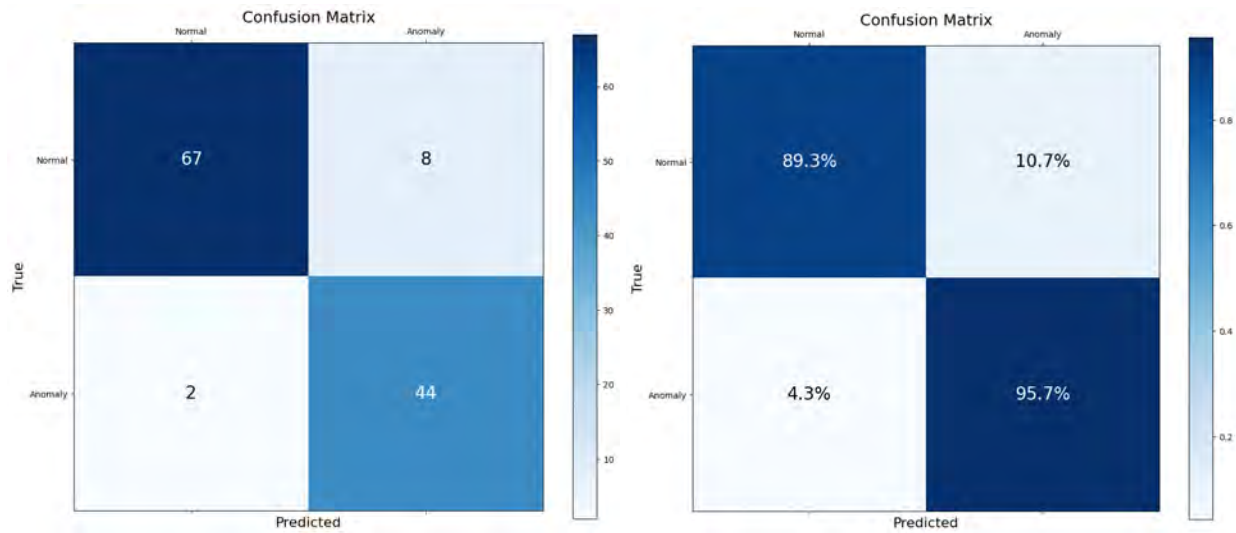
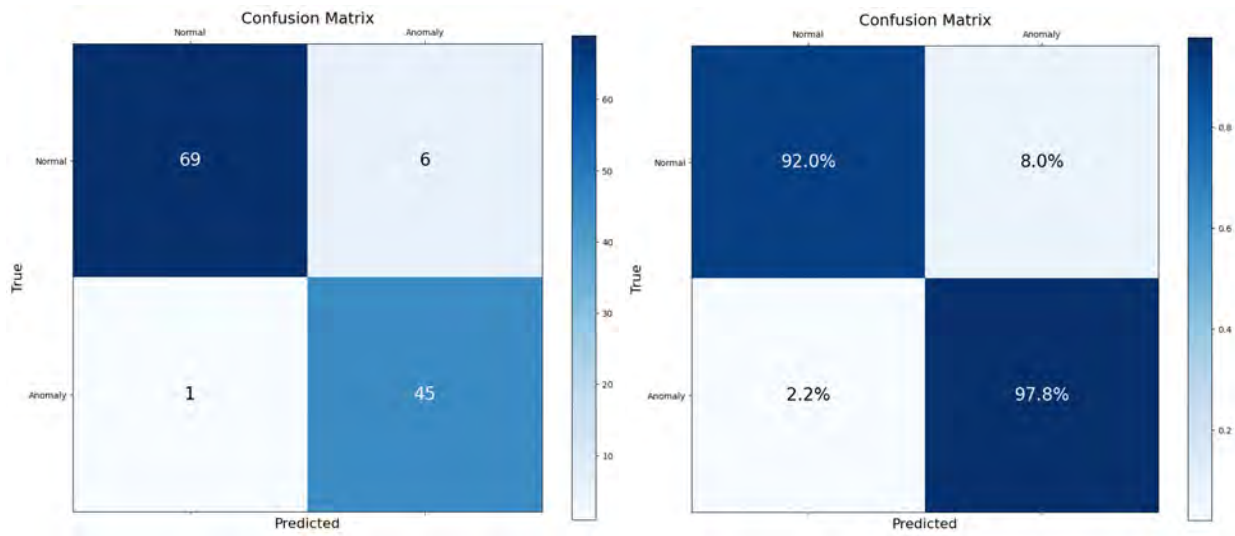


Figure 54: The confusion matrix for Baseline A displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The TabPFN classifier misclassified two (4.3%) anomalous instances as normal and eight (10.7%) normal instances as anomalous.



(a) Off-diagonal terms show one false negative (b) 2.2% of the total anomalies are misclassified as normal
Figure 55: The confusion matrix for Baseline B displays correct predictions (true positives and true negatives) on the diagonal, and errors in the off-diagonal cells. The TabPFN classifier misclassified one (2.2%) anomalous instances as normal and six (8.0%) normal instances as anomalous.

Despite these promising results, this case study was limited in several regards. First, it was conducted primarily for demonstration purposes and focused on a limited subset of SKAB valves with a relatively small number of windows. As a result, the observed improvements of Baseline B over Baseline A may not generalize to more complex equipment, operating conditions, or anomaly types.

Second, the RAG+LLM features were derived from a specific set of documents, using a particular preprocessing pipeline—including text cleaning, structuring, chunking, and indexing—along with a fixed prompt design. This makes the approach sensitive to the quality, coverage, and preparation of the source documents, as well as to prompt engineering and LLM behavior.

Third, no statistical significance testing or ablation studies were performed to isolate the individual contributions of the engineered sliding-window statistics vs. the RAG/LLM-derived features.

Nonetheless, this study provides a useful proof of concept. In future work, we plan to apply the same RAG-enhanced pipeline to long-term historical plant data, including time-aligned sensor measurements, maintenance records, work orders, and other operational documentation. This will allow us to evaluate its effectiveness in more realistic predictive maintenance and anomaly detection scenarios.

6.1.5 Agentic Workflow

In this section, we showcase the results of incorporating agentic workflows with multimodal foundation models for more dynamic and context-aware anomaly detection.

Single Modality Binary Classification: To demonstrate agentic workflow for anomaly detection, a simple AI agent that coordinates several anomaly detection tools and blends their answers into one decision was developed. The agent works via a transparent reason-then-act loop, in which the LLM decides which tool to run, runs it, reads the result, and uses that evidence to choose the next step—an approach popularized by the ReAct pattern for agentic systems. Figure Figure 56 shows the scheme of this workflow: three

classifiers—Random Forest, XGBoost, and TabPFN—are applied to the SKAB industrial sensor dataset (a water-circulation testbed), along with a lightweight comparator that reviews their outputs to issue a final label. As part of setup, all models are “fitted” to the training data. For Random Forest and XGBoost, new models are trained from scratch on the labeled examples. For TabPFN, “fitting” means reformatting the training set into TabPFN’s expected input so that those examples can be carried alongside each new case and used as context during predictions, thus leveraging the model’s pretraining and in-context learning design [199]. This case study will demonstrate whether, without extra context or hand-tuning, simply comparing and aggregating the three model outputs by using majority voting will deliver higher F1-scores and fewer false alarms than any single model when evaluated on held-out SKAB runs. This work used GPT-4o-mini [49] as the LLM engine and the LlamaIndex ReAct [218] module for the formal protocols of tool calling, executions, and feedback.

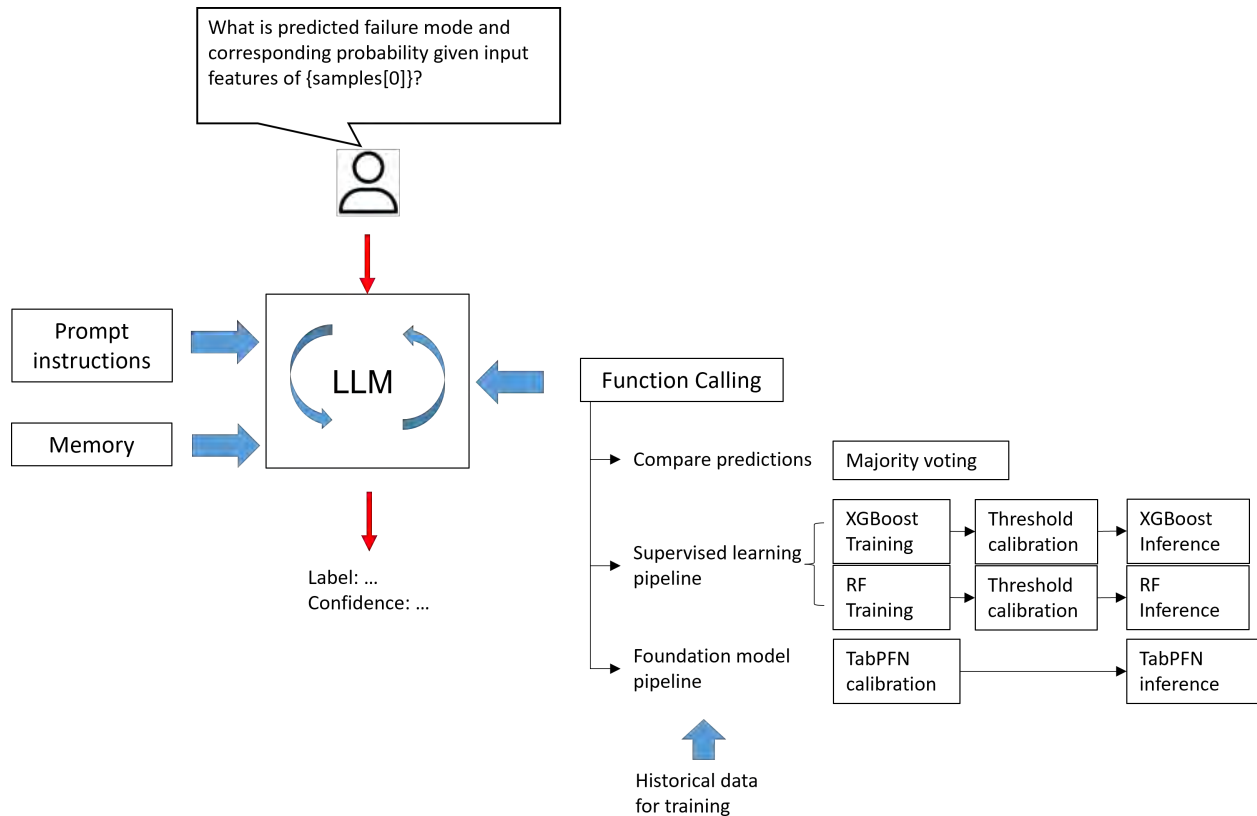


Figure 56: Simple agent workflow with tool access for single modality and binary classification.

To prepare the benchmarks, two time-series sources with different feature profiles were used as the training and testing sources to create reproducible mini-experiments. For each run, a random training subset was created with 150 data points, due to limited computing power, and a 10-instance testing subset was created to include five positives and five negatives. These generated run folders were then replayed by the single-model pipeline and by the agentic ensemble workflow to compute precision/recall/F1 and false-alarm rates for the final comparison. It is worth noting that the prediction results and model performance should **not** be compared against those in previous sections, where the fact that 10x more training data were used resulted in better performance for TabPFN and XGBoost. The goal is to demonstrate the agent’s capabilities in automating anomaly detection through the ReAct workflow, and compare its performance against single supervised learning and TabPFN models in order to identify potential values and shortfalls.

The LLM is prompted with a tightly-scoped ReAct instruction that includes all tools, including Random Forest, XGBoost, TabPFN, and the comparator. The instruction also enforces one call per predictor, with a strictly JSON-typed payload that contains the path to the csv file for training data, and key-value pairs for `x_instance`. After the prediction results are collected, the LLM then passes a single call to the comparator, followed by a final Answer that reports the ensemble label and each model’s label/confidence.

Table 9 demonstrates the agent’s responses when it was asked to describe the available tools and the instructed tasks. The instructions were provided as system headers so the agent could recognize them throughout the workflow.

Table 9: Agent setup with available tools and operating procedure.

Question	Answer (concise rendering)
What tools are available?	The system provides four specialized tools for predictions on the SKAB dataset and for comparing those predictions: 1. <code>predict_with_random_forest</code> — trains a Random Forest and returns a label for a supplied instance. 2. <code>predict_with_xgboost</code> — trains an XGBoost model and returns a label for a supplied instance. 3. <code>predict_with_tabpfn</code> — prepares TabPFN inputs and returns a label for a supplied instance. 4. <code>compare_model_predictions</code> — aggregates the three model outputs (e.g., by majority or confidence) to produce a final label.
What are you instructed to do and what is your thought process?	Act as an evidence-driven anomaly analyst with the following workflow: 1. Plan — determine needed information and select which tools to invoke. 2. Call Tools — execute tools with well-formed JSON payloads; record observations (labels, confidences). 3. Synthesize — combine evidence (e.g., majority vote, confidence weighting) to form a decision. 4. Report — present the final answer succinctly, stopping tool calls once sufficient evidence is gathered. The reasoning follows a transparent loop of <i>Thought</i> → <i>Action</i> → <i>Observation</i> until enough evidence is collected, then a final <i>Thought</i> produces the answer without further tool use.

A redacted agent trace is shown as follows. The agent first queries the three predictors with the provided instance features, captures each tool’s label and confidence, and then invokes the comparator to aggregate these outcomes into a single decision. For each iteration in the loop, the agent will start with reasoning tokens (“Thought”), tool invocations (“Action” with JSON input), and then reading results (“Observation”) until it can emit the specified Answer without using any tools.

The trace also shows basic self-repair: the agent makes an error when first calling `predict_with_tabpfn`

without a payload, immediately detects the exception, and autocorrects by retrying with the proper `{train_csv, x_instance}` payload before proceeding to the final ensemble decision. Meanwhile, to ensure ensemble labels can be parsed correctly, LLM is instructed to generate final messages that adhere to the required schema—`Answer:{"ensemble_label":..., "model_predictions":{...}}`—providing transparent, machine-checkable outputs for each instance.

Agent ReAct Trace: Thought → Action → Observation Loop

```

1 {
2   "prompt": "Evaluate anomaly predictions using three tools: Random Forest, XGBoost, TabPFN.
3     Call each tool exactly once with JSON keys 'train_csv' and 'x_instance'.
4     After collecting all predictions, call 'compare_model_predictions' with the
5     summarized results, then return a final Answer JSON.",
6   "steps": [
7     {
8       "thought": "Use the Random Forest tool for the given instance.",
9       "action": "predict_with_random_forest",
10      "action_input": {
11        "payload": {
12          "train_csv": "<REDACTED_TRAIN_CSV>",
13          "x_instance": "<REDACTED_FEATURES>"
14        }
15      },
16      "observation": {
17        "rf_label": <int>, "rf_confidence": <float>, "rf_proba": {...}, "rf_classes": [...]
18      }
19    },
20    {
21      "thought": "Use the XGBoost tool for the same instance.",
22      "action": "predict_with_xgboost",
23      "action_input": {
24        "payload": {
25          "train_csv": "<REDACTED_TRAIN_CSV>",
26          "x_instance": "<REDACTED_FEATURES>"
27        }
28      },
29      "observation": {
30        "xgb_label": <int>, "xgb_confidence": <float>, "xgb_proba": {...}, "xgb_classes": [...]
31      }
32    },
33    {
34      "thought": "Use the TabPFN tool for the same instance.",
35      "action": "predict_with_tabpfn",
36      "action_input": {},
37      "observation": "ERROR: missing required positional argument: 'payload'"
38    },
39    {
40      "thought": "Auto-correct by providing the proper payload to TabPFN.",
41      "action": "predict_with_tabpfn",
42      "action_input": {
43        "payload": {
44          "train_csv": "<REDACTED_TRAIN_CSV>",
45          "x_instance": "<REDACTED_FEATURES>"
46        }
47      },
48      "observation": {

```

```

48     "tabpfm_label": <int>, "tabpfm_confidence": <float>, "tabpfm_proba": {...},
49     "tabpfm_classes": [...]
50 }
51 },
52 {
53     "thought": "Compare model predictions to derive an ensemble decision.",
54     "action": "compare_model_predictions",
55     "action_input": {
56         "payload": {
57             "prediction_results": {
58                 "rf_label": <int>, "rf_confidence": <float>,
59                 "xgb_label": <int>, "xgb_confidence": <float>,
60                 "tabpfm_label": <int>, "tabpfm_confidence": <float>
61             }
62         }
63     },
64     "observation": {
65         "highest_confidence_prediction": {...},
66         "label_vote_counts": {...},
67         "most_voted_labels": [...],
68         "models_evaluated": [...],
69         "ensemble_label": <int>,
70         "avg_prob_anomaly": <float>,
71         "which_models_disagree": [...],
72         "agreement_flag": <bool>,
73         "predictions": [...]
74     }
75 },
76 ],
77 "answer": {
78     "ensemble_label": <0_or_1>,
79     "model_predictions": {
80         "RandomForest": {"label": <int>, "confidence": <float>},
81         "XGBoost":      {"label": <int>, "confidence": <float>},
82         "TabPFN":       {"label": <int>, "confidence": <float>}
83     }
84 }
85 }

```

Although minimal (single-step planning per instance and a lightweight comparator), this run demonstrates a successful end-to-end agentic workflow for anomaly detection: the agent was correctly prompted, then it selected and executed tools in a reason-act-observe loop, recovered from minor calling errors, and produced auditable ensemble decisions for the full test batch—exactly the behavior ReAct agents are intended to provide in tool-augmented classification settings.

Across the combined summary of four saved runs ($n = 40$ evaluations per model), the simplified ensemble that aggregates Random Forest, XGBoost, and TabPFN predictions did not outperform the strongest single models on this benchmark. Its F1 fell to 0.629, whereas each individual model remained near 0.66; its false-alarm rate (FAR) rose to 0.20, as compared with 0.15 for Random Forest; and its recall dropped to 0.55, as opposed to 0.60 for XGBoost and TabPFN. Taken together, these results show that a context-free, simple aggregation of model outputs does not yield a higher F1 and lower FAR than any single model on held-out SKAB data. See Table 10 for the summary.

Two major patterns explain the shortfall: (i) the comparator counts labels rather than using calibrated confidences or model-specific reliability, and so weak, low-confidence votes can outweigh a highly confident,

historically reliable minority, thus hurting both recall and precision; and (ii) many mistakes are cases in which all three base learners make the same wrong call (e.g., all predict “anomaly” when the truth is “normal”). Without any model context, an react agent simply preserves the error and offers no path to improvement.

Table 10: Combined summary of single-model and ensemble performance on held-out SKAB runs.

Model	n	Precision	Recall	F1	FAR	TN	FP	FN	TP
Ensemble	40	0.73	0.55	0.63	0.20	16	4	9	11
RandomForest	40	0.79	0.55	0.65	0.15	17	3	9	11
TabPFN	40	0.75	0.60	0.67	0.20	16	4	8	12
XGBoost	40	0.75	0.60	0.67	0.20	16	4	8	12

To address the issue of a simple comparator, the majority voting is first replaced by a confidence-weighted soft vote that respects how sure each base learner is. Each model’s output is mapped to a class-consistent probability by using three decision rules for the LLM agent to follow: (i) final ensemble results will predict “1” if the average confidence ≥ 0.5 ; (ii) final ensemble results will predict “1” only if the average confidence ≥ 0.5 and at least two models are confidently positive with confidence ≥ 0.60 ; and (iii) final ensemble results allow a single, very confident model with confidence ≥ 0.85 to overrule two weak, low-confidence peers if their confidences are ≤ 0.55). This shifts the aggregator from counting labels to aggregating probabilistic evidence but directly mitigates weak-vote swamping of a reliable minority and creates a mechanism to exploit residual diversity among the base learners.

Applied to the same benchmark, these weighted rules reduce false alarms in which the simple vote over-reacts and recovers missed anomalies in disagreement cases where a single model provided compelling evidence. As a result, the guardrailed soft vote lowered the FAR to 0.15 (as compared to 0.2 with majority voting) without sacrificing precision. It also reduced the F1 to 0.61 and the recall to 0.5. These effects are precisely what more sophisticated ensembles are designed to deliver thanks to confidence-aware combinations to counteract correlated errors and prevent the majority from entrenching a mistake when one model has strong, consistent evidence.

Multi-Modality Binary Classification: To inject context awareness into the ReAct loop, the agent’s final decision is augmented with a metadata retrieval step in which textual and model-specific metadata are generated and called by the agent for evidence-weighted reasoning. After calling the three predictors, the agent calls a file reading tool, which retrieves per-instance training context, in textual format, for Random Forest and XGBoost models. The context includes out-of-fold (OOF) reliability, such as the F1, recall, and Brier scores, and out-of-distribution (OOD) indicators derived from Mahalanobis distance and featuring a corresponding χ^2 p -value.

OOF reliability is estimated by separating the labeled data into stratified folds, repeatedly fitting on the in-fold data, and producing probabilistic predictions for the held-out fold so that every example receives a strictly held-out score. These raw scores are then passed through an isotonic calibration step that fits only on the corresponding validation portion so as to reduce probability miscalibration without any parametric links. The resulting OOF probabilities for the positive class are summarized via a reliability diagram by binding the scores uniformly in $[0,1]$. Within each bin, the average predicted probability is compared against the empirical event rate. At the end, the miscalibration is condensed into an expected calibration error computed as the sample-weighted absolute gap between these two quantities across bins:

$$\text{ECE} = \sum_b \frac{n_b}{N} |\widehat{\text{freq}}_b - \bar{p}_b|.$$

At the same time, sharpness and calibration together are reflected by the Brier score, the mean squared error between predicted probabilities, and binary outcomes:

$$\text{Brier} = \frac{1}{N} \sum_{i=1}^N (p_i - y_i)^2.$$

The OOD signal is computed by measuring how far a feature vector x lies from the training distribution under a covariance-aware distance, then converting that distance into a statistical surprise score. First, estimate the training mean μ and a well-behaved covariance Σ : when the sample is large and potentially contaminated by outliers, a high-breakdown robust estimator is preferred so as to resist leverage points. When the sample is thin relative to dimensionality, use a shrinkage-stabilized covariance so that the estimate remains well-conditioned. These choices provide a numerically stable and outlier-resistant view of location and scatter, as is critical for OOD detection. With μ and Σ in hand, compute the squared Mahalanobis distance

$$d^2 = (x - \mu)^\top \Sigma^{-1} (x - \mu),$$

a scale- and correlation-aware measure of how atypical x is under the training cloud. Under the common Gaussian reference model, d^2 follows a χ^2 distribution with degrees of freedom equal to the feature dimension, so a tail probability

$$p = 1 - F_{\chi_p^2}(d^2)$$

quantifies extremeness; a small p indicates likely OOD behavior, and a threshold such as $p < \alpha$ (e.g., $\alpha = 0.01$) yields a clear-cut OOD flag. This pipeline—robust or shrinkage covariance for stability, Mahalanobis distance for geometry, and a χ^2 tail probability for calibration—provides a principled, interpretable OOD score grounded in classical multivariate statistics.

Together, these additions give the agent a rational path to avoid preserving a shared mistake, especially when all base learners agree but on dubious evidence, and make the ensemble decision context-aware rather than a context-free vote—within the same reason-act-observe loop popularized by ReAct. Model context is created during training and persisted per instance so the agent can justify its final answer with an evidence table. This design aligns with established practice on ReAct-style tool use, probability calibration for reliable weighting, and distance-based OOD detection.

Table 11 summarizes the basic setup of the agent used in our study, including the tool suite available to the agent and the instructions that govern its ReAct-style workflow.

With the context-aware agent workflow, even when two models oppose a third, the agent can select the minority label when the context warrants it. The JSON snippet below illustrates a case in which the ensemble outputs a positive label after weighing prediction confidences together with reliability and OOD signals. Despite OOD flags on the tree-based models, the agent’s rationale cites the reliability evidence in favor of the chosen minority prediction, showing how the decision is supported by explicit, auditable signals rather than a blind majority:

Context-Aware Agent Decision with Metadata

```

1 {
2   "ensemble_label": 1,
3   "model_predictions": {
4     "RandomForest": {"label": 1, "confidence": 0.6014991181657849},
5     "XGBoost":      {"label": 0, "confidence": 0.6362694549758757},
6     "TabPFN":        {"label": 0, "confidence": 0.6229264736175537}
7   },

```

```

8  "metadata_summary": {
9    "RandomForest": {"mahalanobis_distance": 23.5783258197667, "f1": 1.0},
10   "XGBoost":      {"mahalanobis_distance": 23.5783258197667, "f1": 1.0}
11 },
12 "rationale":
13 "Despite both Random Forest and XGBoost being flagged as OOD, the Random Forest model's
14 perfect metrics make it the most reliable choice."
15 }

```

Table 12 compares the anomaly detection results from context-aware vs. context-free ensemble agent for the same 40 runs. Context-aware ensemble drastically outperforms the context-free runner in every metric. This is because the metadata-enhanced prompt shown above lets the agent weigh reliability and OOD cues,

Table 11: Context-aware agent setup with available tools and operating procedure.

Question	Answer (concise rendering)
What tools are available?	The system provides four specialized tools: 1. <code>predict_with_random_forest</code> — trains a Random Forest on SKAB and returns a predicted label (and related metrics) for a given instance. 2. <code>predict_with_xgboost</code> — trains an XGBoost model on SKAB and returns a predicted label (and related metrics) for a given instance. 3. <code>predict_with_tabPFN</code> — prepares TabPFN inputs on SKAB and returns a predicted label (and related metrics) for a given instance. 4. <code>load_model_metadata</code> — reads stored metadata (e.g., out-of-fold performance, calibration error, χ^2 test results) for the tree models and returns the key metrics for the current instance.
What are you instructed to do and what is your thought process?	Act as an <i>evidence-driven anomaly analyst</i> and follow this workflow: 1. Plan — determine needed information. 2. Call Tools — invoke prediction and metadata tools (RandomForest/XGBoost/TabPFN; OOD/calibration metadata) with well-formed JSON payloads. 3. Synthesize — combine evidence per rubric: favor models with higher OOF recall/F1 and better calibration; de-emphasize models flagged as OOD. 4. Report — provide a concise counter-evidence statement and a JSON-compatible payload; avoid unnecessary loops or empty actions; present the evidence table and verdict before concluding. Reasoning proceeds in a transparent loop of <i>Thought</i> $\rightarrow$ <i>Action</i> $\rightarrow$ <i>Observation</i> until sufficient evidence is gathered, then a final <i>Thought</i> produces the answer without further tool use.

whereas the context-free agent relies on raw votes alone.

Advanced Agent Workflow: A natural next step is a quantitative study of where agentic orchestration truly pays off in comparison to strong single-model baselines. Beyond accuracy metrics (precision/recall/F1) and FAR, the operational cost, including latency, tool-call budget, and token usage, and the conditions under which the ReAct loop adds value (e.g., degree of model disagreement, training-set size, class imbalance, and distribution shift) should be systematically analyzed. Such a quantitative study would also help map the boundary of applicability for agent-based anomaly detection.

Because the agent operates in a reason-act-observe loop, future work should also study the explanatory side explicitly: do reasoning traces correlate with calibrated confidence and with final correctness, and do they maintain accuracy comparable to non-agentic baselines? User studies can test whether ReAct traces are longer and semantically richer, and whether that richness improves behavioral trust and post-hoc diagnosis quality without inducing over-reliance. Prior human-machine interaction results on explanation effects provide measurement templates that can be adapted to the anomaly-detection context.

To evolve into a fully autonomous, learning-in-the-loop anomaly-detection framework, a memory-augmented, multi-agent architecture is proposed that not only classifies but continually improves its own orchestration and decisions over time. A Coordinator (ReAct planner) sets goals and delegates; a Tool Agent discovers or synthesizes new tools for prediction and metadata extraction, including novel feature generators, domain checks, calibration/OOD probes, and exposes them through a standard interface like MCP for runtime registration and versioning; a Runner equipped with customized virtual environment and compilers executes tools; and an Evaluator tracks accuracy, FARs, latency, token/tool cost, and risk–coverage trade-offs to trigger targeted recalibration experiments and dynamic ensemble selection when performance regresses in regimes like class imbalance, small training sets, or drift. A complementary operational role can ultimately close the loop by recommending concrete improvements for adding evidence or engineered features, instrumenting additional sensors where coverage is weak, or even proposing maintenance strategies—for example, requesting human verification because of low confidence in predictions, or requesting condition-based inspections when signals indicate incipient degradation or high uncertainty. Throughout the autonomous framework, agents log prompts, tool calls, and outcomes; tools are typed, validated, and sandboxed, and memory is curated to avoid stale or biased adaptations. With memory to learn from experience, multi-agent roles to dynamically create and execute tools, and operational hooks to recommend sensor and maintenance actions, the system moves from context-free vote tallies to an adaptive, auditable, decision-centric workflow that knows when it is able to help, when to request human verification, and when the right action is more than a label.

Table 12: Comparison of SKAB anomaly-detection workflows from the context-aware agent vs. context-free ensemble and individual base models.

Method	N	Precision	Recall	F1	False Alarm	TN	FP	FN	TP
Ensemble (context aware)	40	1.0000	0.9000	0.9474	0.0000	20	0	2	18
Ensemble (context free)	40	0.6250	0.5000	0.5556	0.3000	14	6	10	10
RandomForest	40	0.7857	0.5500	0.6471	0.1500	17	3	9	11
TabPFN	40	0.7500	0.6000	0.6667	0.2000	16	4	8	12
XGBoost	40	0.7500	0.6000	0.6667	0.2000	16	4	8	12

6.2 Industrial Utility Benchmark Dataset

The Industrial Utility Benchmark dataset, as described in section 5, contains multi-variate time series collected from sensors installed on the circulating water pump system. Historical data pertaining to sensor readings (Figure 57, Figure 58) from the years 2008–2015 were used as features to train both a binary and multi-class classifier to predict anomalies and normal conditions. Each anomaly had a distinct signature usable to label the timeseries data through analysis of the CWS process and vibration data. Along with sensor measurements, the collected data also included metadata related to the plant processes, maintenance logs, operator logs, and work-order documents. The operator logs were added as an additional column for multimodal anomaly detection analysis.

The central hypothesis of this study is that operator logs’ text narratives documenting observations, actions, and contextual cues during plant operations contain complementary information that can reveal anomalies missed by raw sensor measurements alone. While sensor streams capture quantitative changes in pressure, temperature, flow, and other process variables, they may not reflect subtle or transient events such as early-stage fouling, manual interventions, or equipment wear—events only noted by operators in their daily reports. In the next section, by integrating these textual logs as an additional modality in a multimodal classification framework, we aim to assess whether the combined data enhance detection of water-box fouling and CWP diffuser faults, especially in low-data regimes where sensor-only traditional ML models struggle. The experiments therefore test whether the semantic content of operator entries provides discriminative signals that enhance anomaly-detection performance beyond what is achievable with sensor data alone.

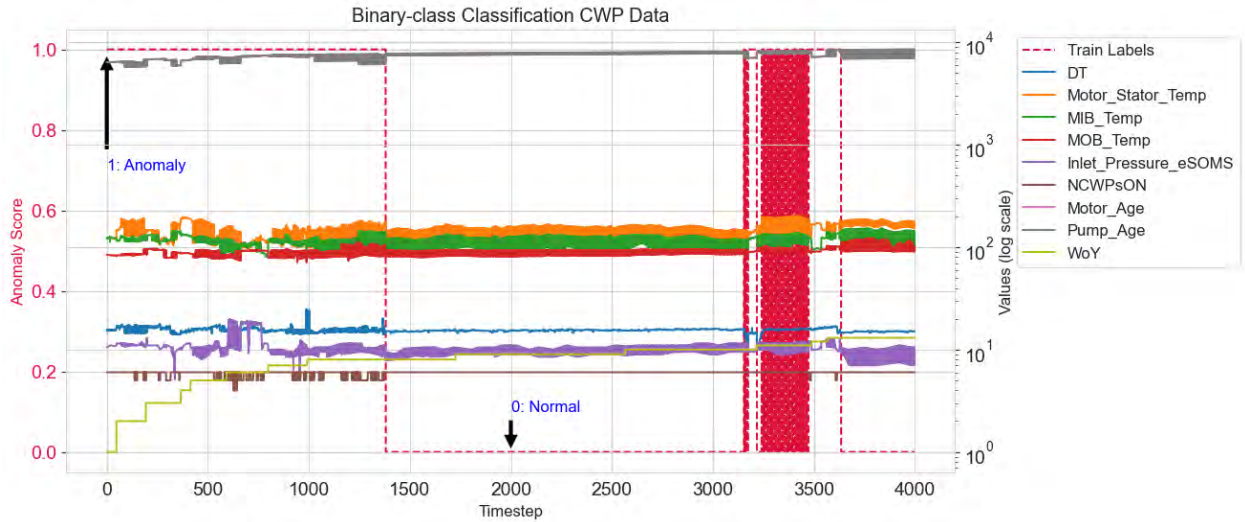


Figure 57: Sensor data, including motor stator temperature, inlet/outlet temperature, inlet pressure, motor and pump age, and number of active CWPs, are used to predict classification labels indicating anomalous conditions such as water fouling and normal healthy states for binary classification.

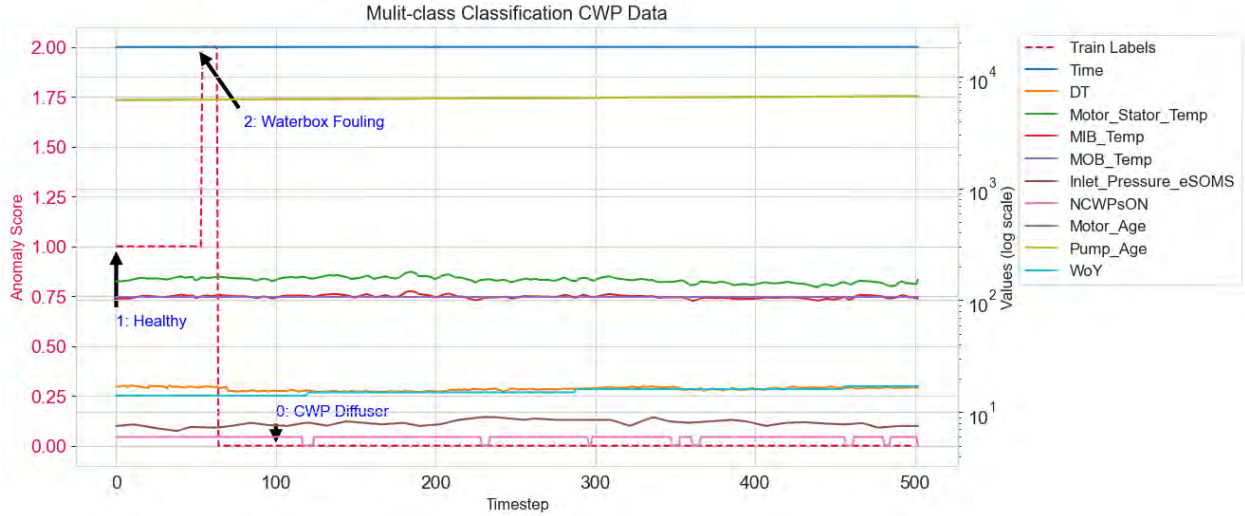
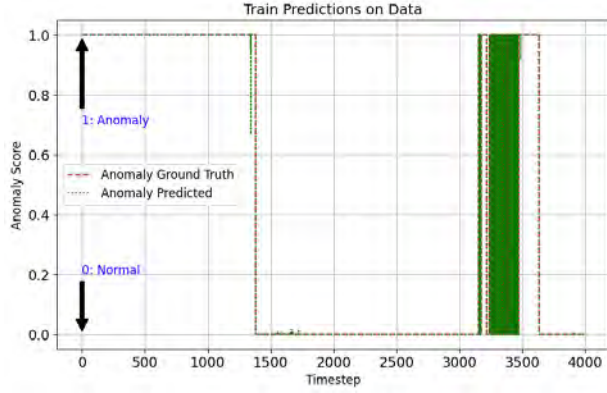


Figure 58: Sensor measurements used for multi-class classification, illustrating the two anomaly types: water-box fouling and CWP diffuser fault.

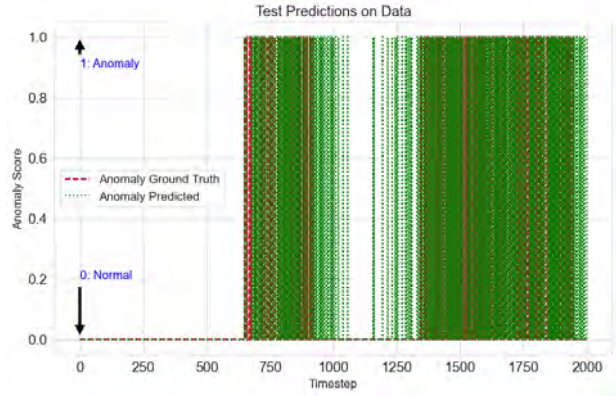
6.2.1 Single-Modality Binary Classification

The pre-trained foundation model TabPFN was first evaluated by using only the sensor-measurement vectors to determine whether it could discriminate anomalous from healthy operating states via in-context learning. The training set was used to provide examples of features and target labels, while no training or weight update took place (Figure 59a). The model was tested on a new set of features, the goal being to determine whether the target corresponded to anomalous or normal conditions. Under this regime, the model achieved a ROC-AUC of 0.9860, as seen in Figure 59b, indicating excellent separability between the two classes. The corresponding confusion matrix revealed that only four anomalous instances (Figure 59c) were mistakenly labeled as healthy, for a false-negative rate of roughly 2.11% (Figure 59d). This result demonstrates that a frozen, pre-trained foundation model can already extract highly discriminative patterns from raw sensor data, without any task-specific training.

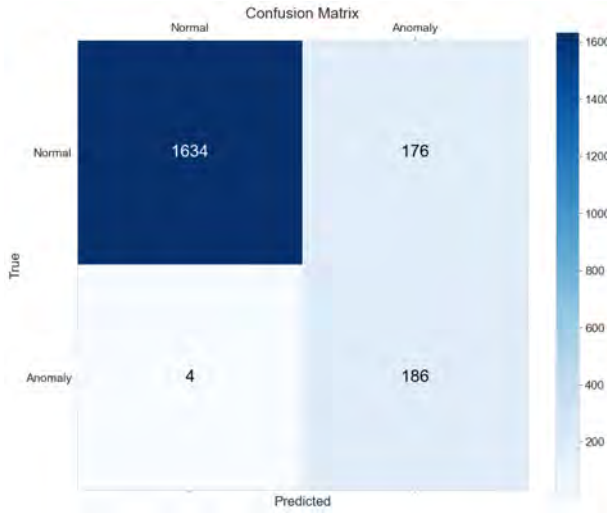
The model produced approximately 176 false-positive detections, a non-trivial value that will showcase a very high FAR in safety-critical systems. One objective of this study was to pinpoint the exact temporal boundaries of anomalies within a continuous sensor stream. As illustrated in Figure 59b, the inferred anomaly interval was systematically elongated around timestep 800, causing the algorithm to overestimate the true fault duration. A second cluster of spurious alerts appears near time step 1100, at which point the model prematurely flags an anomaly, thereby undershooting the actual onset of the second fault in the test set. This pattern highlights the need for finer temporal resolution and more precise boundary detection in the postprocessing stage. In the following section, we evaluate this multimodal configuration, testing whether the semantic information contained in the logs mitigates false-positive alerts and yields a more accurate anomaly window.



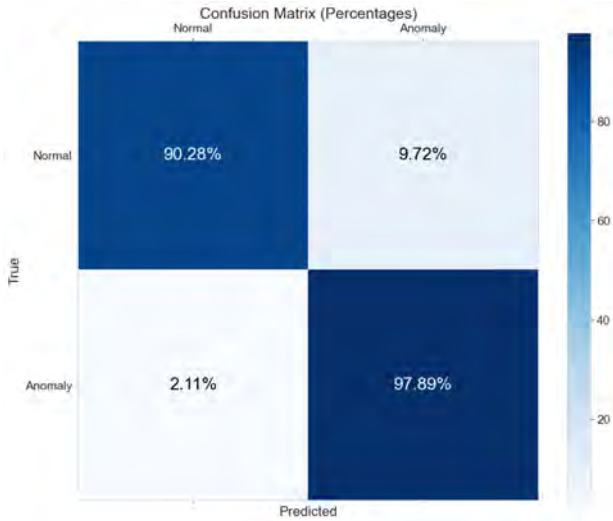
(a) Training data are used for in-context learning in TabPFN



(b) ROC-AUC = 0.9860



(c) The off-diagonal terms show four false negatives

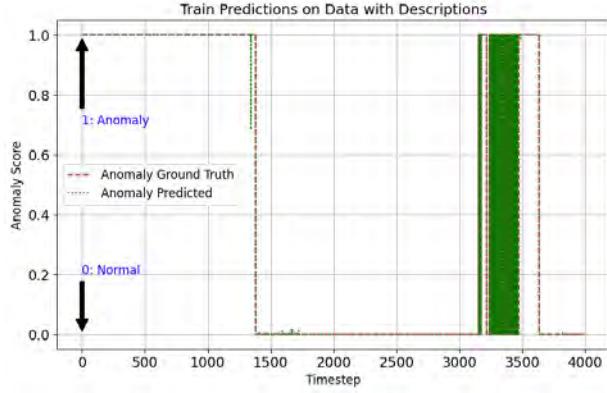


(d) 2.11% of the total anomalies are misclassified as normal

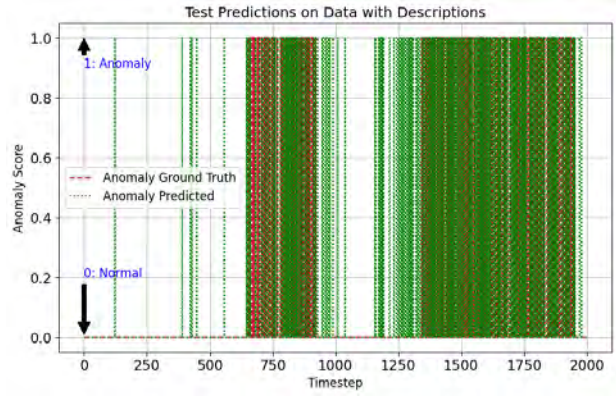
Figure 59: Single modality based anomaly detection results in a high ROC-AUC of 0.9860 for unseen data. Its accuracy is 9.46%, but it cannot accurately estimate the start of the anomaly.

6.2.2 Multimodal Binary Classification

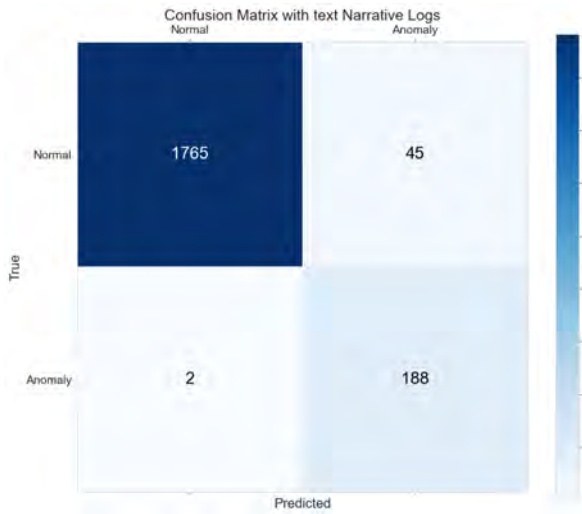
To assess whether a complementary modality can reduce spurious detections and sharpen the temporal delineation of faults, we extended the input to TabPFN by appending the operator-log text associated with each time step. The same in-context learning protocol was used (i.e., the model received only a handful of annotated exemplars and no gradient-based fine-tuning). Incorporating the textual modality reduced the number of anomalous instances that were incorrectly labeled as healthy from four to two (Figure 60c), corresponding in the FAR dropping from 2.11% to 1.05%, as illustrated in Figure 60d. Incorporating the operator-log text as an additional feature for both the training and test sets further enhanced the model's performance, as seen by the ROC-AUC increase from 0.9860 to 0.9961 (Figure 60b). This indicates near-perfect discrimination between healthy and anomalous states, decreasing the number of false-positive detections from 176 to 45 and reducing the overall FAR to 2.49%, a level considerably more acceptable in the monitoring of safety-critical systems.



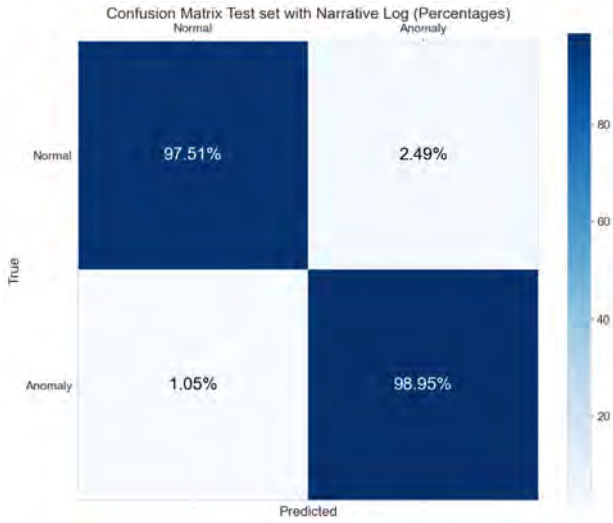
(a) Training data are used for in-context learning in TabPFN



(b) ROC-AUC = 0.9961



(c) The false negatives are reduced to two with log data



(d) 1.05% false negatives

Figure 60: Adding a modality such as operator logs, along with sensor measurements, increases the ROC-AUC from 0.9860 to 0.9961 on the test data, and the classifier has an accuracy of 99.70%. This reduces the number of false positives from 176 to 45, and the start and end of the anomalies are successfully captured.

The inclusion of operator-log text as an auxiliary modality also sharpened the model’s ability to delineate the temporal boundaries of each fault. In the sensor-only configuration, the anomaly interval was systematically overestimated, producing a cluster of false-positive detections between time steps 900 and 1500 after the first fault had already subsided. After augmenting the feature set with the textual descriptions, this overshoot was markedly reduced: the number of spurious alerts in that region dropped dramatically, yielding a cleaner transition back to the healthy state. Likewise, the onset of the second anomaly was captured more precisely; the model now flags the fault at the correct time step rather than lagging behind or truncating its duration. These improvements indicate that the semantic cues embedded in the operator logs provide complementary context that helps the classifier resolve ambiguous sensor patterns and produce a more accurate, time-resolved anomaly segmentation.

The experiments demonstrate that a frozen, pre-trained foundation model can already achieve a high degree of discrimination between healthy and anomalous operating states, and that its performance is markedly enhanced when multimodal information is incorporated as an additional feature. These findings substantiate the hypothesis that pre-trained foundation models, when leveraged through in-context learning

and enriched with multimodal data, can deliver superior accuracy and more reliable alarm characteristics than sensor-only baselines. Having established these gains in a binary-classification setting, the subsequent section will extend the analysis to a multi-class scenario—distinguishing two distinct fault types—and compare TabPFN against conventional tree-based learners (CatBoost, Random Forest, XGBoost) to assess whether the same multimodal advantage persists across more complex classification tasks.

6.2.3 Multi-Class Classification

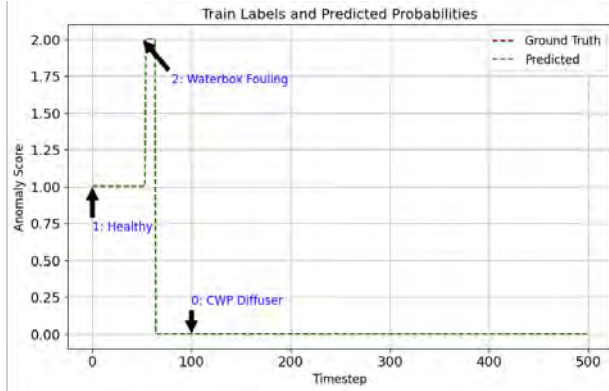
In the historical data from the Industrial Utility Benchmark dataset, one circulating water pump (CWP) had two different types of anomalies: waterbox fouling and CWP diffuser fault. CWP diffuser faults are infrequent and may be caused by blockages in air/water intakes, but failure to diagnose them can result in unexpected downtime, trips, drops in gross load, and additional maintenance costs. The TabPFN classifier model was first fit using sensor measurements, as described in section 5. Figure 61a displays the classifier’s predictions for the training set. From the figure, we see that the CWP initially operates normally, then suffers waterbox fouling, which is then followed by a CWP diffuser fault. The data series is concatenated, and a maintenance interval separates the waterbox fouling event from the diffuser fault. Likewise, in the test set the transition from a diffuser fault back to healthy operation coincides with a maintenance/replacement period. Consequently, the timestamps capture the sequence of events accurately, even though they are not exact. The TabPFN model accurately captures healthy status, waterbox fouling and CWP diffuser faults without any retraining, as it fits data through in-context learning. In case of the test data, the model predicts CWP diffuser faults, healthy regions, and waterbox fouling accurately, but cannot capture the healthy portion at the end of the time series (at around time step 260), as seen in Figure 61b.

The multi-class model accurately predicts all anomalous conditions with no false negatives, as indicated by zero lower off-diagonal terms in the confusion matrix Figure 61c. This makes it a powerful classifier for safety-critical systems, ensuring that no anomalies are missed. However, it misclassifies 10.78% of normal conditions as waterbox fouling anomalies, likely as a result of the shapes of the training and test sets (Figure 61d). Despite limited data on these anomalies, TabPFN outperforms the other models in low-data regimes (Table 13). The multi-class TabPFN model has an accuracy of 92.67% on unseen data, while XGBoost comes in second with an accuracy of 66.67%. This is likely because TabPFN does not require retraining, enabling accurate in-context learning with limited data. In contrast, other models need training and weight updates, perhaps reducing their accuracy in low-data regimes.

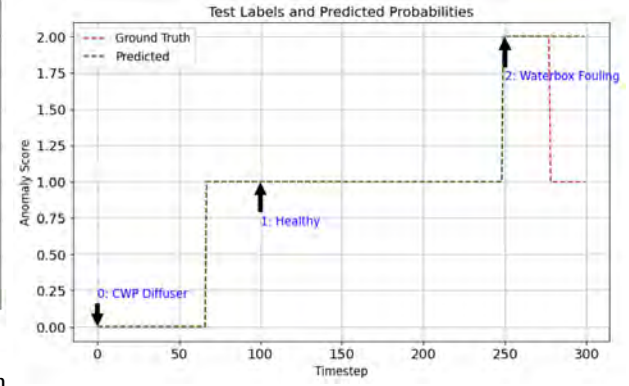
Based on prior experiments, we appended a textual description of the sensor readings as an additional feature column to both the training and test sets, evaluating the impact of multimodal inputs on multiclass classification. The text contains no explicit labels or cues. Three-fold cross-validation was used to evaluate four models on a multi-class anomaly-detection task (anomalous vs. normal). Across both single- and multi-modal inputs, TabPFN attained the highest ROC-AUC, as seen in Figure 62, with the multi-modal version edging out its single-modal counterpart. In contrast to the binary-classification results on the SKAB dataset, incorporating an additional modality did not boost the performance of the tree-based models (CatBoost, Random Forest, and XGBoost) when operating in the low-data regime.

Large pre-trained foundation models such as TabPFN can be leveraged through in-context learning, eliminating the need for the task-specific fine-tuning or extensive feature engineering that traditional ML pipelines require. By simply providing a few annotated examples in the prompt, the model learns to map the combined sensor vector and accompanying log text to the target class. Because these models are trained on massive corpora, they natively understand and process raw text, allowing the operator logs to be fed directly without handcrafted embeddings or separate natural-language pipelines. This capability makes foundation models a natural fit for multimodal anomaly-detection tasks in which textual and numerical data must be

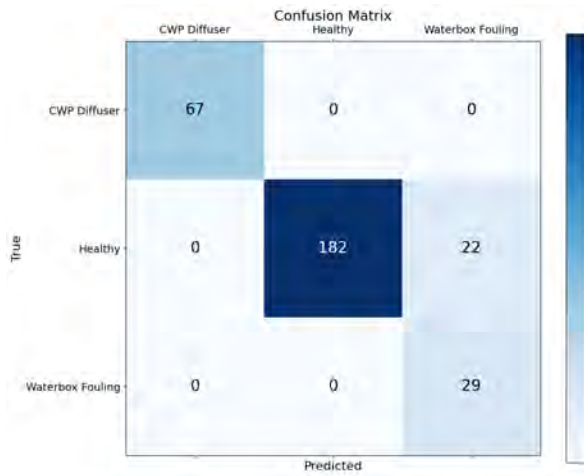
jointly interpreted.



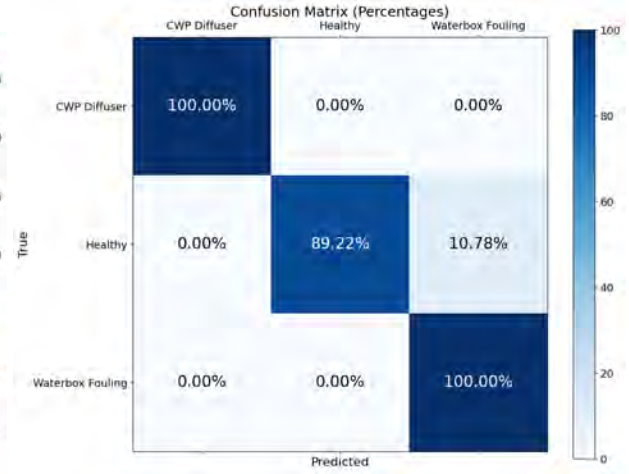
(a) Training data are used for in-context learning in TabPFN



(b) Test data show an accuracy of 92.67%



(c) Off-diagonal terms show 22 false positives



(d) 10.78% of the healthy data are misclassified as anomalous

Figure 61: TabPFN accurately identifies classes in multi-class classification and achieves zero false negatives on unseen data. This makes TabPFN a reliable model for safety-critical systems, as anomalies do not go undetected.

Table 13: Comparison of classification metrics across different multi-class models.

Metric	TabPFN	Random Forest	CatBoost	XGBoost
Precision	0.9583	0.7967	0.6421	0.6613
Recall	0.9267	0.4300	0.6133	0.6667
F1 Score	0.9347	0.3599	0.5900	0.6407
Accuracy	0.9267	0.4300	0.6133	0.6667

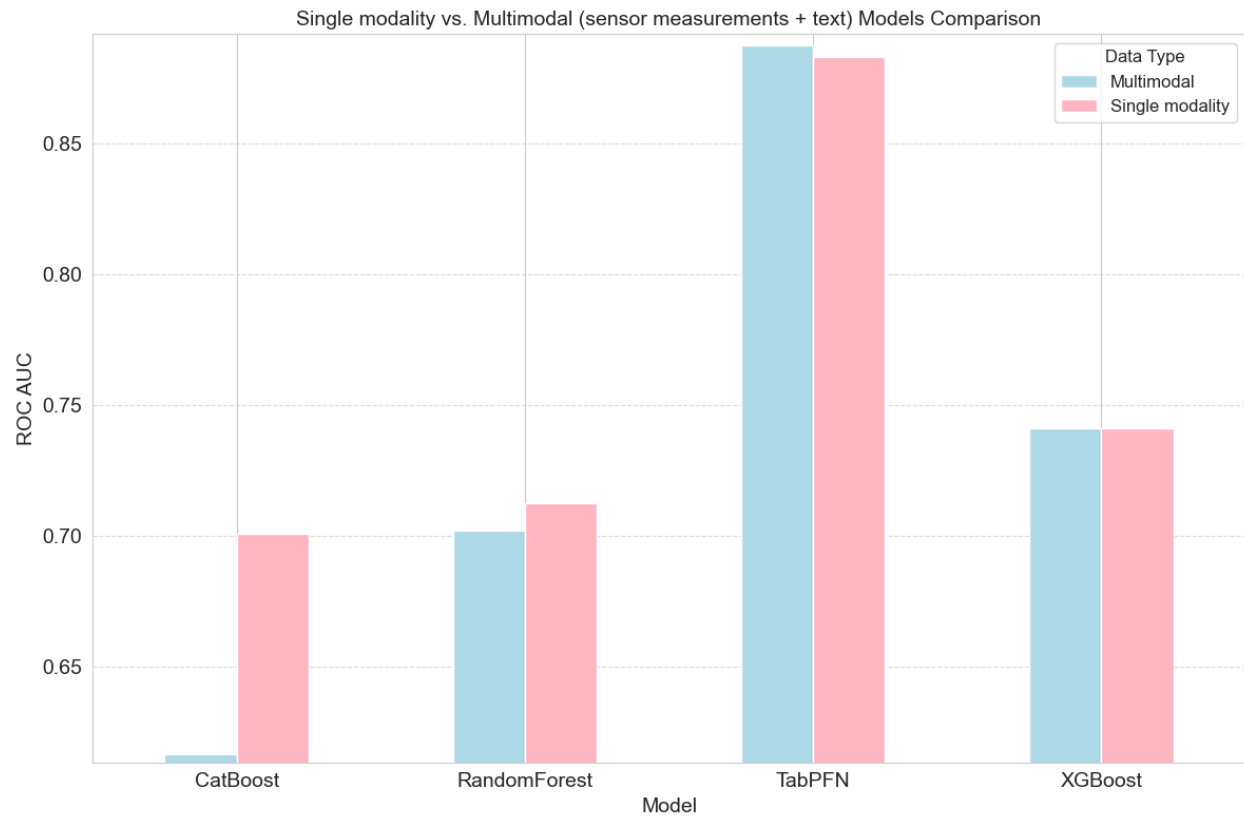


Figure 62: Three-fold cross-validation evaluates four models on a multi-class anomaly-detection task (anomalous vs. normal).

7 CONCLUSIONS

The integration of foundation models such as TabPFN with multimodal data represents a significant advancement in anomaly detection for engineering systems. By leveraging the attention mechanism of TabPFN, we can effectively process and analyze diverse data sources, including textual descriptions and sensor measurements, without the need for complex preprocessing. This approach not only enhances the model’s ability to detect subtle anomalies but also improves overall performance metrics such as accuracy, recall, precision, and F1-score. The superior results achieved with multimodal TabPFN underscore its potential as a robust, reliable solution for monitoring and maintaining critical systems, ensuring both operational efficiency and safety.

Across the SKAB valve-closing benchmark, the contrast between single- and multimodal inference makes this capability concrete. Starting from a single-modality TabPFN configuration, the model already achieved a ROC-AUC of 0.9208 with 95.46% recall, yet it still recorded 49 anomalous timesteps as normal and struggled to localize the onset of faults. Appending concise textual descriptors of the sensor traces raised the ROC-AUC to 0.9865, increased the accuracy to 96.42%, and cut the number of false negatives to 39—evidence that the attention layers use linguistically encoded context to disambiguate borderline readings while keeping the inference path unchanged. The reduction in ringing near regime transitions, coupled with the elimination of spurious alerts at around time step 500, illustrate how a modest multimodal prompt can deliver outsized safety benefits without retraining or bespoke feature engineering.

Contrary to common beliefs about LLMs and deep learning models requiring vast amounts of data, pre-trained foundation models like TabPFN can adapt out-of-the-box to tasks such as anomaly detection in both multi-class and binary scenarios, even in low-data regimes.

Likewise, the Industrial Utility Benchmark circulating-water pump study showed that even when only a few hundred timesteps carry ground-truth labels, the same pre-trained TabPFN prompt can separate healthy and faulty spans: the sensor-only run attained a ROC-AUC of 0.9860 with four false negatives (2.11%), and enriching the prompt with operator-log snippets pushed the ROC-AUC to 0.9961 while halving the false negatives and shrinking the false-alarm count from 176 to 45. Because these gains emerge without weight updates or handcrafted embeddings, they directly validate the claim that foundation models can be fielded “cold start” in low-data regimes yet still exceed the precision and recall of traditionally trained tree ensembles.

Complementing this, this work layered a ReAct-style agentic workflow on top of the multimodal model so the system could reason→plan→act across tools and fuse evidence before issuing a decision. Specifically, the agent orchestrated TabPFN alongside tree-based learners via a customized tool schema, aggregated calibrated per-model confidences rather than raw votes, and consulted context signals, including out-of-distribution scores from Mahalanobis distance and simple reliability summaries—to down-weight dubious inputs or trigger abstention when evidence is weak. This converted the final step from a context-free tally into evidence-weighted decision-making with transparent traces, following the best practice for ReAct agents and probability-aware ensembling.

The results demonstrate that pre-trained foundation models originally designed for language tasks but readily adaptable to tabular inputs such as TabPFN offer a uniquely powerful paradigm for industrial anomaly detection. By exploiting in-context learning, these models can ingest raw sensor measurements alongside free-text operator logs, without any task-specific fine-tuning, yet still achieve near-perfect discrimination and dramatically lower false-negative and false-positive rates. The textual modality supplies contextual cues that resolve ambiguous sensor patterns, enabling more accurate delineation of fault onsets and durations. Consequently, a single, frozen foundation model can replace a cascade of bespoke preprocessing pipelines,

feature-engineering steps, and separate ML algorithms, delivering a unified, scalable solution that is both data-efficient and robust to the low-sample regimes typical of safety-critical systems. This capability positions multimodal foundation models as indispensable tools for next-generation condition-monitoring platforms, in which rapid deployment, high reliability, and the seamless integration of heterogeneous data sources are paramount.

Beyond the foundation model itself, the comparative experiments with CatBoost, Random Forest, and XGBoost highlight the same pattern. Hyperparameter-tuned baselines with engineered encodings narrow the gap under single-modality inputs, yet the moment that textual descriptors are added—even after the overhead of bespoke encoders—the multimodal TabPFN still tops accuracy, precision, recall, F1, and ROC-AUC, including the three-fold cross-validation sweep where its ROC-AUC rises from 0.9953 to 0.9995. These outcomes confirm that attention-equipped foundation models scale gracefully, while other learners incur growing complexity costs as modalities accumulate.

In the benchmark, an agent that simply aggregates labels can propagate shared mistakes when all base learners call a normal point an “anomaly,” offering no route to correction. By contrast, a context-aware agent that integrates calibrated confidences and OOD diagnostics can discount unreliable votes and elevate a well-supported minority. These behaviors match the literature, with calibrated probabilities based on context yielding better evidence for combination than uncalibrated scores, while Mahalanobis-style distance tests provide a lightweight signal for distributional shift that helps prevent overconfident errors. Overall, the agent can act as an interpretable controller that selects tools, justifies choices with explicit thought→action→observation traces, and adapts decision strategies. Because the workflow is modular, it can evolve toward dynamic per-instance model selection and richer context checks while preserving auditability. This makes the agentic framework a practical, safety-aligned complement to multimodal TabPFN in high-consequence monitoring.

The upgraded workflow evaluated on 40 SKAB mini-experiments demonstrates the practical payoff of this context infusion: precision jumps to 1.0000, recall to 0.9000, and the false alarm rate collapses to zero, outstripping both the context-free ensemble (precision 0.6250, false-alarm 0.3000) and any single base learner. These lifts stem directly from the ReAct agent’s ability to weigh calibrated confidences, reliability metadata, and Mahalanobis-based OOD scores before issuing its verdict, proving that orchestration quality, not just model strength, determines field performance.

NPPs operate under the most stringent safety requirements found in any industrial sector. A single undetected fault can cascade into loss-of-coolant accident, core damage, or radiological release, with consequences that far outweigh the cost of occasional false alarms. In this context, false-negative detections are intolerable, because they allow an emerging anomaly to progress unchecked while operators remain unaware of the underlying degradation. The experiments reported herein demonstrate that a foundation model, through in-context learning, can achieve lower false negative rates with the addition of a modality. The key to this improvement is the seamless integration of operator logs as a textual modality alongside the numeric sensor measurements. These free-text entries capture human observations, maintenance actions, and contextual cues that are invisible to raw sensor data alone. By providing the model with this complementary information, the system learns to recognize subtle precursors of failure—such as early fouling or manual interventions—that would otherwise be missed, thereby dramatically reducing the risk of undetected faults.

Within the Industrial Utility Benchmark deployment setting, these enhancements manifest as materially tighter temporal bounds on detected faults: the multimodal TabPFN no longer overestimates the decay of the first anomaly between timesteps 900 and 1500, correctly trims the inferred duration of the second event, and trims the false-alarm volume by more than 4×. Coupled with the multi-class experiments in which TabPFN preserves zero false negatives, even when distinguishing between water-box fouling and diffuser faults, the

evidence indicates that multimodal in-context learning can shoulder the stringent recall requirements that nuclear operators impose, without overwhelming crews or nuisance alarms.

In sum, the ability of foundation models to jointly process sensor columns and unstructured text makes them uniquely suited to the safety-critical demands of NPP monitoring. By markedly lowering false-negative rates while keeping false-positive alarms at an acceptable level, they provide a more reliable early-warning system that can help prevent catastrophic incidents and support the high-integrity operations required in the nuclear domain.

Looking ahead, the same infrastructure can absorb retrieval-augmented evidence, richer operator narratives, and iterative agent feedback so that anomaly judgments are accompanied by sourced explanations and proactive mitigation suggestions. As the memory-augmented, multi-agent orchestration loop matures—with calibrated probes, dynamic tool discovery, and auditable reasoning traces—it will extend the multimodal TabPFN foundation into a continuously learning monitoring tier that not only detects incipient faults, but also prescribes context-aware actions that align with nuclear quality assurance mandates.

REFERENCES

- [1] G. Sanderson, “But what is a neural network? — chapter 1, deep learning.” YouTube, 2017. 3Blue1Brown.
- [2] G. Sanderson, “Visualizing transformers and attention — chapter 6, deep learning.” YouTube, 2024. 3Blue1Brown.
- [3] 3Blue1Brown, “Manim: Mathematical animation engine,” 2020. Python library for mathematical animations.
- [4] The Manim Community Developers, “Manim community edition,” 2021.
- [5] R. Bommasani, “On the opportunities and risks of foundation models,” *arXiv preprint arXiv:2108.07258*, 2021.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [7] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1715–1725, 2016.
- [8] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [9] M. Schuster and K. Nakajima, “Japanese and korean voice search,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, IEEE, 2012.
- [10] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” *arXiv preprint arXiv:1808.06226*, 2018.
- [11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *International Conference on Learning Representations (ICLR)*, 2013.
- [12] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.
- [13] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
- [14] J. Ramos *et al.*, “Using tf-idf to determine word relevance in document queries,” *Proceedings of the First Instructional Conference on Machine Learning*, vol. 242, pp. 29–48, 2003.
- [15] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.
- [16] Z. S. Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.
- [17] J. R. Firth, “A synopsis of linguistic theory, 1930-1955,” in *Studies in Linguistic Analysis*, pp. 1–32, Blackwell, 1957.

- [18] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.
- [19] T. K. Landauer, P. W. Foltz, and D. Laham, “An introduction to latent semantic analysis,” *Discourse Processes*, vol. 25, no. 2-3, pp. 259–284, 1998.
- [20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119, 2013.
- [21] T. Mikolov, W.-t. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, 2013.
- [22] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.
- [23] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 2227–2237, 2018.
- [24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 4171–4186, 2019.
- [25] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” tech. rep., OpenAI, 2019.
- [26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.
- [27] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “Electra: Pre-training text encoders as discriminators rather than generators,” in *International Conference on Learning Representations*, 2020.
- [28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.
- [29] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, 2019.
- [30] P. He, X. Liu, J. Gao, and W. Chen, “Deberta: Decoding-enhanced bert with disentangled attention,” in *International Conference on Learning Representations*, 2021.
- [31] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, “Roformer: Enhanced transformer with rotary position embedding,” *arXiv preprint arXiv:2104.09864*, 2021.

- [32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [33] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*, 2022.
- [34] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763, 2021.
- [35] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 12449–12460, 2020.
- [36] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, *et al.*, “Chronos: Learning the language of time series,” *arXiv preprint arXiv:2403.07815*, 2024.
- [37] A. Garza and M. Mergenthaler-Canseco, “Timegpt-1,” *arXiv preprint arXiv:2310.03589*, 2023.
- [38] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.
- [39] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations*, 2017.
- [40] Y. Tian, Y. Liu, Z. Chong, Z. Huang, and H.-A. Jacobsen, “Gala: Can graph-augmented large language model agentic workflows elevate root cause analysis?,” *arXiv preprint arXiv:2508.12472*, 2025.
- [41] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, “Imagebind: One embedding space to bind them all,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15180–15190, 2023.
- [42] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, “Biobert: a pre-trained biomedical language representation model for biomedical text mining,” *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.
- [43] I. Beltagy, K. Lo, and A. Cohan, “Scibert: A pretrained language model for scientific text,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 3615–3620, 2019.
- [44] A. Thawani, J. Pujara, P. A. Szekely, and F. Ilievski, “Representing numbers in nlp: a survey and a vision,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 644–656, 2021.
- [45] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker, “Time2vec: Learning a vector representation of time,” in *arXiv preprint arXiv:1907.05321*, 2019.

- [46] I. D. Zhu, J. Wang, A. Guez, D. Calandriello, and T. Graepel, “Causal reasoning from meta-reinforcement learning,” in *International Conference on Learning Representations*, 2019.
- [47] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.
- [48] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [49] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, *et al.*, “Gpt-4o system card,” 2024.
- [50] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, “The llama 3 herd of models,” *arXiv e-prints*, pp. arXiv–2407, 2024.
- [51] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid, “The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities,” *arXiv preprint arXiv:2408.13296*, 2024.
- [52] J. Shen, N. Tenenholz, J. B. Hall, D. Alvarez-Melis, and N. Fusi, “Tag-llm: Repurposing general-purpose llms for specialized domains,” *arXiv preprint arXiv:2402.05140*, 2024.
- [53] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [54] G. Dong, H. Yuan, K. Lu, C. Li, M. Xue, D. Liu, W. Wang, Z. Yuan, C. Zhou, and J. Zhou, “How abilities in large language models are affected by supervised fine-tuning data composition,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 177–198, 2024.
- [55] B. Peng, C. Li, P. He, M. Galley, and J. Gao, “Instruction tuning with gpt-4,” *arXiv preprint arXiv:2304.03277*, 2023.
- [56] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, *et al.*, “Scaling instruction-finetuned language models,” *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.
- [57] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” *arXiv preprint arXiv:2101.00190*, 2021.
- [58] J. Kim, J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee, “Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 36187–36207, 2023.
- [59] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *Advances in neural information processing systems*, vol. 36, pp. 10088–10115, 2023.
- [60] H. Jin, G. Papadimitriou, K. Raghavan, P. Zuk, P. Balaprakash, C. Wang, A. Mandal, and E. Deelman, “Large language models for anomaly detection in computational workflows: from supervised fine-tuning to in-context learning,” in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–17, IEEE, 2024.

- [61] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” *arXiv preprint arXiv:1909.11942*, 2019.
- [62] G. Papadimitriou, H. Jin, C. Wang, R. Mayani, K. Raghavan, A. Mandal, P. Balaprakash, and E. Deelman, “Flow-bench: A dataset for computational workflow anomaly detection,” *arXiv preprint arXiv:2306.09930*, 2023.
- [63] J. Liao, X. Xu, M. C. Nguyen, A. Goodge, and C. S. Foo, “Coft-ad: Contrastive fine-tuning for few-shot anomaly detection,” *IEEE Transactions on Image Processing*, vol. 33, pp. 2090–2103, 2024.
- [64] H. Guo, J. Yang, J. Liu, J. Bai, B. Wang, Z. Li, T. Zheng, B. Zhang, J. Peng, and Q. Tian, “Log-former: A pre-train and tuning pipeline for log anomaly detection,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, pp. 135–143, 2024.
- [65] A. Li, Y. Zhao, C. Qiu, M. Kloft, P. Smyth, M. Rudolph, and S. Mandt, “Anomaly detection of tabular data using llms,” *arXiv preprint arXiv:2406.16308*, 2024.
- [66] S. Rayana, “Odds library,” *Stony Brook University, Department of Computer Sciences*, 2016.
- [67] Y. F. Lim, J. Zhu, and G. Pang, “Adapting large language models for parameter-efficient log anomaly detection,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 325–337, Springer, 2025.
- [68] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.
- [69] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “Retrieval augmented language model pre-training,” in *International conference on machine learning*, pp. 3929–3938, PMLR, 2020.
- [70] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, “Dense passage retrieval for open-domain question answering,” in *EMNLP (1)*, pp. 6769–6781, 2020.
- [71] W. B. Croft, S. Cronen-Townsend, and V. Lavrenko, “Relevance feedback and personalization: A language modeling perspective,” in *DELOS*, 2001.
- [72] O. Huly, I. Pogrebinsky, D. Carmel, O. Kurland, and Y. Maarek, “Old ir methods meet rag,” in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2559–2563, 2024.
- [73] C. Xu, D. Guo, N. Duan, and J. McAuley, “Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval,” *arXiv preprint arXiv:2203.06169*, 2022.
- [74] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang, *et al.*, “Retrieval-augmented generation with graphs (graphrag),” *arXiv preprint arXiv:2501.00309*, 2024.
- [75] S. Gupta, R. Ranjan, and S. N. Singh, “A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions,” *arXiv preprint arXiv:2410.12837*, 2024.
- [76] X. Xu, H. Wang, Y. Liang, P. S. Yu, Y. Zhao, and K. Shu, “Can multimodal llms perform time series anomaly detection?,” *arXiv preprint arXiv:2502.17812*, 2025.

- [77] Y. Wang and K. P. Birman, “Diagnosing and resolving cloud platform instability with multi-modal rag llms,” in *Proceedings of the 5th Workshop on Machine Learning and Systems*, pp. 139–147, 2025.
- [78] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.
- [79] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.
- [80] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
- [81] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [82] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.
- [83] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.
- [84] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 24824–24837, 2020.
- [85] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, “Multimodal machine learning: A survey and taxonomy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2018.
- [86] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *Proceedings of the 28th International Conference on Machine Learning*, pp. 689–696, 2011.
- [87] C. G. Snoek, M. Worring, and A. W. Smeulders, “Early versus late fusion in semantic video analysis,” *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pp. 399–402, 2005.
- [88] P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli, “Multimodal fusion for multimedia analysis: A survey,” *Multimedia Systems*, vol. 16, no. 6, pp. 345–379, 2010.
- [89] D. Ramachandram and G. W. Taylor, “Deep multimodal learning: A survey on recent advances and trends,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 96–108, 2017.
- [90] W. Wang, D. Tran, and M. Feiszli, “What makes training multi-modal classification networks hard?,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12695–12705, 2020.
- [91] R. Hu and A. Singh, “Unit: Multimodal multitask learning with a unified transformer,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1439–1449, 2021.
- [92] K. Gadzicki, R. Khamsehashari, and C. Zetsche, “Early vs late fusion in multimodal convolutional neural networks,” in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pp. 1–6, IEEE, 2020.

- [93] J. Lu, D. Batra, D. Parikh, and S. Lee, “Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [94] H. Tan and M. Bansal, “Lxmert: Learning cross-modality encoder representations from transformers,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 5100–5111, 2019.
- [95] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, *et al.*, “Oscar: Object-semantics aligned pre-training for vision-language tasks,” in *European Conference on Computer Vision*, pp. 121–137, Springer, 2020.
- [96] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.
- [97] J. Lu, J. Yang, D. Batra, and D. Parikh, “Hierarchical question-image co-attention for visual question answering,” in *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [98] A. Das, H. Agrawal, L. Zitnick, D. Parikh, and D. Batra, “Human attention in visual question answering: Do humans and deep networks look at the same regions?,” in *Computer Vision and Image Understanding*, vol. 163, pp. 90–100, 2017.
- [99] L. Yao, C. Mao, and Y. Luo, “Graph convolutional networks for text classification,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7370–7377, 2019.
- [100] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1735–1742, 2006.
- [101] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607, 2020.
- [102] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.
- [103] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.
- [104] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer, “Winclip: Zero-/few-shot anomaly classification and segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19606–19616, 2023.
- [105] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, “Scaling up visual and vision-language representation learning with noisy text supervision,” in *Proceedings of the 38th International Conference on Machine Learning*, pp. 4904–4916, 2021.
- [106] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

- [107] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, “Contrastive learning with hard negative samples,” in *International Conference on Learning Representations*, 2020.
- [108] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, “Curriculum learning: A survey,” *International Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.
- [109] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, “Ts2vec: Towards universal representation of time series,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 8980–8987, 2022.
- [110] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” in *Proceedings of the 38th International Conference on Machine Learning*, pp. 8821–8831, 2021.
- [111] A. Aghajanyan, B. Huang, C. Ross, V. Karpukhin, H. Xu, N. Goyal, D. Okhonko, M. Joshi, G. Ghosh, M. Lewis, *et al.*, “Cm3: A causal masked multimodal model of the internet,” *arXiv preprint arXiv:2201.07520*, 2022.
- [112] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [113] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with vq-vae-2,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [114] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, *et al.*, “Scaling autoregressive models for content-rich text-to-image generation,” in *Transactions on Machine Learning Research*, 2022.
- [115] J. Lu, C. Clark, R. Zellers, R. Mottaghi, and A. Kembhavi, “Unified-io: A unified model for vision, language, and multi-modal tasks,” *arXiv preprint arXiv:2206.08916*, 2022.
- [116] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han, “Autoregressive image generation using residual quantization,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022.
- [117] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.
- [118] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, “Deep learning for anomaly detection: A review,” *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–38, 2021.
- [119] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “Mvtec ad—a comprehensive real-world dataset for unsupervised anomaly detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9592–9600, 2019.
- [120] Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer, “Spot-the-difference self-supervised pre-training for anomaly detection and segmentation,” in *European Conference on Computer Vision*, pp. 392–408, Springer, 2022.
- [121] S. R. Menon and C. Vondrick, “Visual classification via contextual calibration,” in *Advances in Neural Information Processing Systems*, 2022. arXiv:2210.00080.

- [122] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [123] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.
- [124] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [125] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, “Do deep generative models know what they don’t know?,” *arXiv preprint arXiv:1810.09136*, 2018.
- [126] M. Salehi, N. Sadjadi, S. Baselizadeh, M. H. Rohban, and H. R. Rabiee, “Multiresolution knowledge distillation for anomaly detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14902–14912, 2021.
- [127] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in *International Conference on Learning Representations*, 2021.
- [128] J. Wyatt, A. Leach, S. M. Schmon, and C. G. Willcocks, “Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 650–656, 2022.
- [129] M. S. Graham, W. H. Pinaya, P. Wright, P.-D. Tudosiu, Y. H. Mah, J. T. Teo, R. Jager, D. Werring, P. Nachev, S. Ourselin, *et al.*, “Denoising diffusion models for out-of-distribution detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2947–2956, 2023.
- [130] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.
- [131] H. Zhou, J. Huang, H. Lu, T. Wang, and Y. Liu, “Anomaly detection via reverse distillation from one-class embedding,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9737–9746, 2022.
- [132] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” in *ACM SIGMOD Record*, vol. 29, pp. 427–438, 2000.
- [133] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: identifying density-based local outliers,” *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.
- [134] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *IEEE International Conference on Data Mining*, pp. 413–422, 2008.
- [135] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [136] T. Defard, A. Setkov, A. Loesch, and R. Audigier, “Padim: a patch distribution modeling framework for anomaly detection and localization,” in *International Conference on Pattern Recognition*, pp. 475–489, 2021.

- [137] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, “Deep one-class classification,” *International Conference on Machine Learning*, pp. 4390–4399, 2018.
- [138] K. Sohn, C. Zhang, C.-L. Li, J. Zhang, H. Lee, and T. Pfister, “Learning and evaluating representations for deep one-class classification,” in *International Conference on Learning Representations*, 2020.
- [139] L. Li, J. Yan, H. Wang, and Y. Jin, “Time series anomaly detection with multiresolution ensemble decoding,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 4232–4239, 2021.
- [140] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International Conference on Machine Learning*, pp. 1530–1538, PMLR, 2015.
- [141] P. Kirichenko, P. Izmailov, and A. G. Wilson, “Why normalizing flows fail to detect out-of-distribution data,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 20578–20589, 2020.
- [142] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [143] P. Seeböck, S. M. Waldstein, S. Klimesch, H. Bogunovic, T. Schlegl, B. S. Gerendas, R. Donner, U. Schmidt-Erfurth, and G. Langs, “Unsupervised identification of disease marker candidates in retinal oct imaging data,” vol. 38, pp. 1037–1047, IEEE, 2018.
- [144] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, “f-anogan: Fast unsupervised anomaly detection with generative adversarial networks,” *Medical Image Analysis*, vol. 54, pp. 30–44, 2019.
- [145] M. Zhu, Y. Liu, Z. Luo, C. Jing, H. Chen, G. Xu, X. Wang, and C. Shen, “Unleashing the potential of the diffusion model in few-shot semantic segmentation,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 42672–42695, 2024.
- [146] P. Chambon, C. Bluethgen, J.-B. Delbrouck, R. Van der Sluijs, M. Polacin, J. M. Z. Chaves, T. M. Abraham, S. Purohit, C. P. Langlotz, and A. Chaudhari, “Roentgen: Vision-language foundation model for chest x-ray generation,” *arXiv preprint arXiv:2211.12737*, 2022.
- [147] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, 2006.
- [148] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” *PloS One*, vol. 10, no. 3, p. e0118432, 2015.
- [149] C. Elkan, “The foundations of cost-sensitive learning,” in *International Joint Conference on Artificial Intelligence*, vol. 17, pp. 973–978, 2001.
- [150] A. Lavin and S. Ahmad, “Evaluating real-time anomaly detection algorithms—the numanta anomaly benchmark,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 38–44, IEEE, 2015.

- [151] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, “Towards total recall in industrial anomaly detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14318–14328, 2022.
- [152] P. Mishra, R. Verk, D. Fornasier, C. Piciarelli, and G. L. Foresti, “Vt-adl: A vision transformer network for image anomaly detection and localization,” in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, pp. 01–06, IEEE, 2021.
- [153] I. D. Katser and V. O. Kozitsin, “Skoltech anomaly benchmark (skab).” <https://www.kaggle.com/dsv/1693952>, 2020.
- [154] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, “Time-series extreme event forecasting with neural networks at uber,” vol. 34, pp. 1–5, 2015.
- [155] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, “The ucr time series archive,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.
- [156] L. Zhang, T. Jia, M. Jia, Y. Wu, A. Liu, Y. Yang, Z. Wu, X. Hu, P. S. Yu, and Y. Li, “A survey of aiops for failure management in the era of large language models,” 2024.
- [157] A. Saxena, K. Goebel, D. Simon, and N. Eklund, “Damage propagation modeling for aircraft engine run-to-failure simulation,” in *2008 International Conference on Prognostics and Health Management*, pp. 1–9, IEEE, 2008.
- [158] B. Charan, D. Jaswanth, E. Hemanth, and M. S. Naidu, “Machine learning and deep learning approaches for healthcare predictive analytics,” pp. 1698–1707, 2024.
- [159] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, 2009.
- [160] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116, 2018.
- [161] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, “Adbench: Anomaly detection benchmark,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 32142–32159, 2022.
- [162] C. Bergmeir, R. J. Hyndman, and B. Koo, “A note on the validity of cross-validation for evaluating autoregressive time series prediction,” *Computational Statistics & Data Analysis*, vol. 120, pp. 70–83, 2018.
- [163] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, “Matching networks for one shot learning,” in *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [164] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, 2017.
- [165] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” 2019.
- [166] Y. Zhao, Z. Nasrullah, and Z. Li, “Pyod: A python toolbox for scalable outlier detection,” *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

- [167] M. Khurram, C. Zhang, S. Muhammad, H. Kishnani, K. An, K. Abeywardena, U. Chadha, and K. Behdinin, “Artificial intelligence in manufacturing industry worker safety: A new paradigm for hazard prevention and mitigation,” *Processes*, vol. 13, no. 5, p. 1312, 2025.
- [168] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, *et al.*, “Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning,” *arXiv preprint arXiv:1711.05225*, 2017.
- [169] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” in *Proceedings of the 29th International Conference on Machine Learning*, pp. 1467–1474, 2012.
- [170] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, “Poisoning web-scale training datasets is practical,” *arXiv preprint arXiv:2302.10149*, 2023.
- [171] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery,” *Queue*, vol. 16, no. 3, pp. 31–57, 2018.
- [172] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafraan, K. R. Narasimhan, and Y. Cao, “React: Synergizing reasoning and acting in language models,” in *The eleventh international conference on learning representations*, 2022.
- [173] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language agents with verbal reinforcement learning,” *arXiv preprint arXiv:2303.11366*, 2023.
- [174] OpenAI, “Function calling — openai api guide,” 2025. Accessed 2025-11-20.
- [175] R. Nakano *et al.*, “Webgpt: Browser-assisted question-answering with human feedback,” *arXiv preprint arXiv:2112.09332*, 2021.
- [176] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, “Pal: Program-aided language models,” *arXiv preprint arXiv:2211.10435*, 2022.
- [177] T. Schick, J. Dwivedi-Yu, J. Weed, *et al.*, “Toolformer: Language models can teach themselves to use tools,” in *NeurIPS 2023 (Findings)*, 2023.
- [178] OpenAI, “Using tools (hosted tools and function tools) — openai agents sdk,” 2025. See also Agents SDK docs: <https://openai.github.io/openai-agents-python/tools/>.
- [179] OpenAI, “Structured model outputs: Ensure responses adhere to a json schema,” 2025. Accessed 2025-11-20.
- [180] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, “Toolformer: Language models can teach themselves to use tools,” *arXiv preprint arXiv:2302.04761*, 2023.
- [181] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, “Program-aided language models,” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP) – Findings*, 2022.
- [182] C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez, “Memgpt: Towards llms as operating systems,” *arXiv preprint arXiv:2310.08560*, 2023.

- [183] W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang, “A-MEM: Agentic memory for LLM agents,” *arXiv preprint arXiv:2502.12110*, 2025.
- [184] Z. Zhang, X. Bo, C. Ma, R. Li, X. Chen, Q. Dai, J. Zhu, and J.-R. Wen, “A survey on the memory mechanism of large language model based agents,” *arXiv preprint arXiv:2404.13501*, 2024.
- [185] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [186] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, “Self-consistency improves chain of thought reasoning in language models,” in *International Conference on Learning Representations (ICLR)*, 2023.
- [187] L. Lin, T. E. Reeves, and V. Agarwal, “Using a large language model for accurate technical language generation in the predictive maintenance of circulating water systems in nuclear power plants,” in *Nuclear Plant Instrumentation and Control & Human-Machine Interface Technology (NPIC&HMIT 2025)*, 2025.
- [188] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, “Exploring llm-based agents for root cause analysis,” *arXiv preprint arXiv:2403.04123*, 2024.
- [189] Z. Wang, Z. Liu, Y. Zhang, A. Zhong, J. Wang, F. Yin, L. Fan, L. Wu, and Q. Wen, “Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models,” *arXiv preprint arXiv:2310.16340*, 2024. Accepted at CIKM 2024.
- [190] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, D. Zhang, and T. Xu, “Automatic root cause analysis via large language models for cloud incidents,” *arXiv preprint arXiv:2305.15778*, 2023.
- [191] L. Zhang *et al.*, “A survey of aiops in the era of large language models,” *arXiv preprint arXiv:2507.12472*, 2025.
- [192] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, *et al.*, “Autogen: Enabling next-gen llm applications via multi-agent conversations,” in *First Conference on Language Modeling*, 2024.
- [193] P. Athe, L. Lin, and N. Dinh, “Multiagent ai collaboration for digital twin development and assessment,” in *Nuclear Plant Instrumentation and Control & Human-Machine Interface Technology (NPIC&HMIT 2025)*, 2025.
- [194] LangChain Inc., “Langgraph - agent orchestration framework.” <https://www.langchain.com/langgraph>, 2025. Accessed: 2025-11-12.
- [195] FoundationAgents, “Metagpt: The multi-agent framework.” <https://github.com/FoundationAgents/MetaGPT/tree/main>, 2025. Accessed: 2025-11-12.
- [196] B. Romera-Paredes, M. Barekatin, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, *et al.*, “Mathematical discoveries from program search with large language models,” *Nature*, vol. 625, no. 7995, pp. 468–475, 2024.
- [197] A. Novikov, N. Vű, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, S. Shirobokov, B. Kozlovskii, F. J. Ruiz, A. Mehrabian, *et al.*, “Alphaevolve: A coding agent for scientific and algorithmic discovery,” *arXiv preprint arXiv:2506.13131*, 2025.

- [198] Q. Zhang, C. Hu, S. Upasani, B. Ma, F. Hong, V. Kamanuru, J. Rainton, C. Wu, M. Ji, H. Li, U. Thakker, J. Zou, and K. Olukotun, “Agentic context engineering: Evolving contexts for self-improving language models,” *arXiv preprint arXiv:2510.04618*, 2025.
- [199] N. D. Hollmann *et al.*, “TabPFN: A transformer that solves small tabular classification problems in a second,” in *Proceedings of the 40th International Conference on Machine Learning*, 2023. arXiv:2207.01848.
- [200] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter, “Accurate predictions on small data with a tabular foundation model,” *Nature*, vol. s41586-024-08328-6, 2025. Open access.
- [201] V. Agarwal, A. V. Gribok, K. Araseethota Manjunatha, J. A. Smith, N. J. Lybeck, V. Yadav, M. Yarlett, B. Diggans, and H. Palas, “Integrated risk-informed condition based maintenance capability and automated platform: Technical report 3,” Tech. Rep. PKM-DOC-21-0007, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2022.
- [202] V. Agarwal, K. A. Manjunatha, A. V. Gribok, T. J. Mortenson, H. Bao, R. D. Reese, T. A. Ulrich, R. L. Boring, and H. Palas, “Scalable technologies achieving risk-informed condition-based predictive maintenance enhancing the economic performance of operating nuclear power plants,” Tech. Rep. INL/EXT-21-64168, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2021.
- [203] N. Goss, B. Diggans, F. Lukaczyk, P. Lahoda, J. Hanson, H. Palas, V. Agarwal, A. V. Gribok, V. Yadav, J. A. Smith, N. J. Lybeck, and K. A. Manjunatha, “Integrated risk-informed condition based maintenance capability and automated platform: Technical report 1,” Tech. Rep. INL/RPT-22-67330, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2021.
- [204] T. N. Nguyen, T. Downar, and R. Vilim, “A probabilistic model-based diagnostic framework for nuclear engineering systems,” *Annals of Nuclear Energy*, vol. 149, p. 107767, 2020.
- [205] T. N. Nguyen, R. Ponciroli, P. Bruck, T. C. Esselman, J. A. Rigatti, and R. B. Vilim, “A digital twin approach to system-level fault detection and diagnosis for improved equipment health monitoring,” *Annals of Nuclear Energy*, vol. 170, p. 109002, 2022.
- [206] J. De Kleer and J. Kurien, “Fundamentals of model-based diagnosis,” *IFAC Proceedings Volumes*, vol. 36, no. 5, pp. 25–36, 2003.
- [207] R. B. Vilim, T. N. Nguyen, and R. Ponciroli, “Explainable and trustworthy diagnostics achievable through process-based automated reasoning,” in *12th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, (Virtual Meeting), 2021.
- [208] R. K. Shah and D. P. Sekulić, *Fundamentals of Heat Exchanger Design*. Hoboken, NJ: John Wiley & Sons, 2nd ed., 2023.
- [209] A. J. Dave, T. N. Nguyen, and R. B. Vilim, “Integrating llms for explainable fault diagnosis in complex systems,” *arXiv preprint arXiv:2402.06695*, 2024.
- [210] C. M. Walker, V. Agarwal, L. Lin, A. C. Hall, R. A. Hill, T. J. Mortenson, N. J. Lybeck, *et al.*, “Explainable artificial intelligence technology for predictive maintenance,” tech. rep., Idaho National Laboratory (INL), Idaho Falls, ID (United States), 2023.

- [211] V. Agarwal, K. Araseethota Manjunatha, J. A. Smith, A. V. Gribok, V. Yadav, H. Palas, M. Yarlett, N. Goss, S. Yurkovich, B. Diggans, *et al.*, “Machine learning and economic models to enable risk-informed condition based maintenance of a nuclear plant asset,” Tech. Rep. INL/EXT-21-61984, Idaho National Laboratory, 2021.
- [212] C. M. Walker, L. Lin, V. Agarwal, N. J. Lybeck, A. C. Hall, R. A. Hill, and R. L. Boring, “Demonstration and evaluation of explainable and trustworthy predictive technology for condition-based maintenance,” tech. rep., Idaho National Laboratory (INL), Idaho Falls, ID (United States), 2024.
- [213] L. Lin, C. Walker, and V. Agarwal, “Explainable machine-learning tools for predictive maintenance of circulating water systems in nuclear power plants,” *Nuclear Engineering and Technology*, vol. 57, no. 9, p. 103588, 2025.
- [214] T. Le Magueresse and S. Le Gall, “Comparison of anomaly detection models in an industrial context: application to non-cyclic datasets,” 2023.
- [215] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “Catboost: unbiased boosting with categorical features,” *Advances in neural information processing systems*, vol. 31, 2018.
- [216] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [217] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.
- [218] LlamaIndex, “Workflow for a react agent.” https://developers.llamaindex.ai/python/examples/workflow/react_agent/, 2025. LlamaIndex Python Documentation. Accessed 2025-11-17.