AN ASSET FOUNDATION MODEL FOR INDUSTRIAL ASSET PERFORMANCE MANAGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce the asset foundation model (AFM), a generative framework for asset performance management (APM) spanning high-value industrial assets and manufacturing processes. The AFM is applicable across sectors such as energy, chemicals, manufacturing, and utilities by leveraging rich time series data and event streams to provide a robust basis for next-generation APM solutions. A shared transformer backbone with lightweight heads supports forecasting, anomaly detection, and event querying. The model is pre-trained on operational and simulator corpora, then fine-tuned on asset-specific histories for minimal effort adaptation, using per-sensor discrete tokenization for robustness. Beyond sensors, the AFM incorporates alarms, set-point changes, and maintenance logs via event tokens, enabling time-aligned "what/when" queries and high value applications such as root cause triage, alarm suppression, and maintenance planning. In representative field deployments (e.g., ESPs and compressors), the AFM exceeds prior gains, delivers earlier warnings, and reduces false alarm minutes. Operator-oriented explanations based on attention rollout and integrated gradients highlight which sensors/events drove each alert, while natural language querying allow experts to "talk to the data" features. Calibrated prediction intervals from discrete to continuous with isotonic calibration support risk aware thresholds. On the theory side, we prove closed form bounds on quantization error and a Lipschitz stability result for discretization noise through the encoder, justifying sample efficient adaptation with frozen backbones. Field benchmarks corroborate competitive accuracy and calibrated coverage. The result is a versatile, scalable, and interpretable foundational framework with significant business impact on industrial asset management.

1 INTRODUCTION

Across large industrial sectors such as energy, chemicals, manufacturing and utilities, asset performance management (APM) still wrestles with three compounding problems at scale: excessive false alarms, slow adaptation to new plants (from onboarding new equipment, processes or regimes), and bespoke models that do not transfer across sites, leading to downtime and health, safety & environmental (HSE) risks, as well as escalating operational and support costs. In practice, threshold alarms miss subtle degradations, yet overwhelm operators during normal transients, despite established alarm-management guidance. Meanwhile, organizations seek cross-asset value under ISO 55000-style asset management goals, but the analytics layer lags behind. The main challenge is to maximize the value of existing CAPEX-intensive installations through optimization, end-to-end scenario analysis, and collective intelligence across the value chain (e.g., from reservoir to pipeline in an oil and gas setting).

The classical asset modeling approaches for APM suffer in multiple fronts and have been shown to be difficult to scale across assets. Thresholds and one-off machine learning pipelines fail for recurring field reasons: (i) intermittent and uneven data coverage; (ii) asset-specific feature engineering; (iii) inability to treat alarms/events as first-class signals; (iv) dependence on scarce subject-matter experts; (v) sensitivity to sensor noise and drift; (vi) poor generalization across sites; (vii) heavy maintenance overhead; and (viii) high label demands. These realities explain why many "deployed" systems degrade in months and why alarm KPIs (e.g., floods, chattering, standing alarms) remain stubbornly off target in real plants. The interpretability of such results, even if they are accurate, is questionable. Moreover, querying the right data for the event of interest (i.e., the root causes

that have driven such events) is difficult to deduce, which has made the adoption of such predictive models less widespread.

In this work, we build on our previously deployed time-series foundation model (TSFM) for rotating equipment, and explore an asset foundation model (AFM) for cross-industry APM. The model consists of a shared transformer backbone pretrained on operational and simulator corpora, fine-tuned with minimal effort on asset histories; lightweight heads support forecasting, anomaly detection, and event querying; and per-sensor discrete tokenization improves robustness and sequence modeling. The AFM maintains a fit-for-purpose stance and explicitly extends beyond rotating equipment to process units and multi-site fleets.

Beyond sensors, the AFM ingests alarms, set-point changes, and maintenance logs as time-aligned event tokens, enabling "what/when" queries and powering high-value operator workflows: root-cause triage (e.g., "What sensor/event drove an alert?"), alarm suppression, and maintenance planning by linking alerts to recent interventions. This directly targets field realities—irregular event timing, class imbalance, and drift—that typically sink threshold-only systems. This is extremely important as the AFM provides a way to naturally converse with the data and model for realistic use cases such as equipment prognostics, process optimization, root cause analysis, etc.

The AFM provides operator-oriented explanations—attention rollout and integrated gradients adapted to tokenized multivariate sensors and event channels—so teams can see which signals/events drove each forecast or alert; a plain-English query layer lets experts "talk to the data." For example, a production engineer can interact with the AFM and ask questions such as "What was the compressor discharge pressure when High Bearing Temperature was reported on 05/08/2025?". These interactions are not possible in the current state of the art models.

Our key contributions are summarized as follows:

1. We introduce the asset foundation model (AFM), a generative framework for cross-industry APM. To our knowledge, we are among the first to successfully bring together ideas from FMs and apply them to industrial time series data in a holistic way.
2. We produce quantization error analysis in Appendix A.1 as theoretical basis for our design.
3. We provide experimental evaluations across various tasks, demonstrating that the AFM delivers consistently low squared error with median 0.008 across heterogeneous assets.

These advancements position the AFM as a robust solution for calibrated and interpretable decision-making tailored to operators, thereby facilitating more scalable and high-performance deployments of large-scale foundation models tied to industrial constraints.

2 Related Work

Foundation models in time series analysis. The concept of foundation models (FMs)—large-scale pretrained models that can be adapted to downstream tasks—has recently been extended to time series data (Liang et al., 2024; Shi et al., 2025). Early efforts have shown that pretraining on diverse time series can yield models with strong zero-shot or few-shot performance on forecasting tasks. One of the first transformer-based frameworks for unsupervised representation learning on multivariate time series demonstrated that a pretrained transformer encoder could be fine-tuned for classification and regression tasks with improved accuracy over training from scratch (Zerveas et al., 2020). More recently, Chronos proposed a transformer language-model approach to time series, treating sensor readings as a sequence of tokens and pretraining on a large collection of time series datasets (Ansari et al., 2024). Chronos established a strong benchmark for zero-shot and transfer learning in forecasting by "learning the language" of time series patterns across 42 datasets.

Several TSFMs have focused on improving forecasting performance via massive pretraining. TimesFM, a decoder-only transformer model pretrained on a corpus of real-world and synthetic time series, achieves near state-of-the-art accuracy on diverse forecasting benchmarks without task-specific training (Das et al., 2024). The model uses an input patching technique and demonstrates effective zero-shot generalization to new datasets. In parallel, researchers have explored scaling up TSFMs. Time-MoE is a mixture-of-experts transformer architecture with up to 2.4 billion parameters, which is pretrained on an extremely large dataset ($\sim$300 billion points) spanning 9 domains

(Shi et al., 2025). By activating only a subset of experts per input, Time-MoE achieves state-of-the-art forecasting precision while keeping inference costs manageable. These advances indicate that the scaling laws and architectural innovations from NLP (e.g., expert routing) are being successfully applied to build more powerful TSFMs for forecasting.

Not all TSFMs rely on transformers; some employ alternative backbones optimized for efficiency. For instance, the Tiny Time Mixers (TTMs) model uses a multi-scale MLP-Mixer architecture pre-trained on heterogeneous time series data to serve as a domain-agnostic forecasting model (Ekambaram et al., 2024). TTMs emphasize lightweight design and fast adaptation, showing that even simpler architectures can serve as FMs when trained on large data and carefully tuned (Liang et al., 2024). Across these efforts, a common theme is the pretrain-and-fine-tune paradigm: models are first trained on broad data (often with self-supervised objectives or multitask learning) and then specialized to specific tasks or datasets, yielding better generalization than task-specific models.

Unified latent representations. Parallel to TSFM scaling and tokenization advances, several works pursue a single backbone shared across multiple time-series tasks. UniTS (Gao et al., 2024), for example, introduced a unified sequence encoder with lightweight task heads, showing that a single latent representation can support forecasting, classification, and anomaly detection. Architecturally, UniTS relies on continuous embeddings with a shared temporal encoder and multiple supervised objectives, but it assumes uniformly sampled inputs and does not explicitly incorporate heterogeneous modalities such as alarms, set-point changes, or operator events. More recent unified frameworks similarly focus on multitask learning over clean benchmark datasets, typically using patch-based encoders or recurrent/transformer hybrids without per-channel vocabularies. In contrast, the AFM adopts a token-based formulation with per-sensor vocabularies and an event-aligned auxiliary channel, enabling the backbone to jointly process discrete events and continuous telemetry on a single time grid. Unlike UniTS, which performs end-to-end fine-tuning for each task, the AFM is designed as a frozen universal encoder: asset-specific adaptation is delegated to thin linear/MLP heads, minimizing revalidation and preserving cross-asset transferability. These architectural differences place the AFM closer to language, extending unified representations to irregular, event-rich industrial data where existing multi-task time-series frameworks do not operate.

Deep sequence modeling for time series. Recurrent neural network (RNNs) (Rumelhart et al., 1986; Jordan, 1986), long short-term memory (LSTM) networks (Hochreiter & Schmidhuber, 1997), temporal convolutions networks (TCNs) (Lea et al., 2016), and transformer-family models have advanced forecasting and anomaly detection. Efficient transformer variants (e.g., Informer (Zhou et al., 2021b), Autoformer (Wu et al., 2022), FEDformer (Zhou et al., 2022), PatchTST (Nie et al., 2023), TimesNet (Wu et al., 2023), DLinear (Zeng et al., 2022)) tackle long context and seasonal-trend decomposition, while foundation-style models such as Chronos and TimeGPT pursue cross-domain pretraining.

Tokenization and discretization. Uniform quantization, VQ-VAE and discrete representations provide stability and compressibility (van den Oord et al., 2018). Channel-aware tokenization (e.g., CHARM) explores cross-channel priors (Behrad et al., 2025). In industrial telemetry, discretization also dampens heavy-tailed spikes and missing-data artifacts, yielding robustness to sensor dropouts and outliers. Learned companders or per-channel codebooks can trade bitrate for fidelity, while change-point–aware or run-length encodings reduce sequence length and accelerate decoding without sacrificing temporal resolution.

Stability and generalization. Lipschitz control and spectral normalization bound sensitivity. Linear probing and frozen backbones explain sample-efficient adaptation. In sequential settings, contractive residual paths and normalized attention further limit error compounding across horizons, improving closed-loop stability. Calibration layers (e.g., temperature scaling or conformal coverage) help preserve interval reliability under moderate distribution shift, while lightweight adapters/LoRA enable site-specific tuning without revalidating the entire backbone.

APM and alarm management. ISO 55000 (International Organization for Standardization, 2024), ANSI/ISA-18.2 (Int, 2016), IEC 62682 (International Electrotechnical Commission, 2022), and (Howard, 2007) codify requirements for asset governance and alarm performance. Statistical thresholds and rule-based alarm suppression are common but brittle under drift and transients (Ahnlund et al., 2003). Forecast-driven alarms that gate on prediction-interval breaches and context (e.g., state of maintenance, mode changes) reduce false annunciations while retaining interpretability de-

manded by standards. Multi-sensor fusion and deduplication further curtail nuisance minutes by collapsing correlated alerts into a single actionable event path.

3 DESIGN

The AFM should provide a fit-for-purpose, scalable backbone that can adapt across a wide range of industrial assets without retraining from scratch. By default, the backbone remains frozen after pretraining, ensuring generalizability across different sites and asset types, while lightweight linear or multi-layer perceptron (MLP) (Murtagh, 1991) heads allow per-asset customization with minimal labeled data. The architecture is explicitly built to handle diverse time-series sensor data, irregular events (e.g, alarms, set-point changes, maintenance logs), and potentially unstructured text inputs, bringing them into a common tokenized and time-aligned representation.

Deployment emphasizes compute-aware windowing so that long time horizons can be modeled efficiently in real time, enabling both edge and server deployments without heavy overhead. This approach reduces engineering effort, ensures robustness to noise and drift, and supports cross-asset transfer, making the AFM practical for forecasting, anomaly detection, and event-aware querying in live industrial environments.

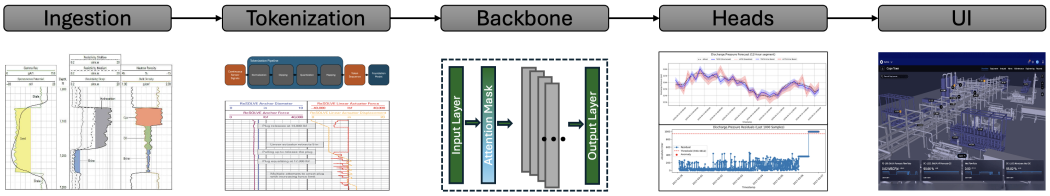


Figure 1: Pipeline diagram for the AFM.

The AFM comprises of the following components:

1. **Per-sensor discrete tokenization.** A uniform mid-rise quantizer with clipping maps each scaled value z into one of B_c bins: given radius R_c and bin width $\Delta_c = 2R_c/B_c$, the k-th bin covers $[-R_c + k\Delta_c, -R_c + (k+1)\Delta_c$ and is represented by its midpoint. A residual MLP can optionally encode fine residuals $r = z - \tilde{z}$. The pad token (PAD), end-of-sequence (EOS) token, and per-sensor vocabularies avoid cross-sensor interference.

2. **Shared transformer encoder.** A causal encoder produces hidden states h_t for forecasting; non-causal layers are used during representation learning. Rotary or ALiBi-style positional encodings (Press et al., 2022) support long horizons. A synchronized event channel encodes event types, (no event) and tokens at each grid step.

3. **Lightweight heads.** Separate heads support specific tasks: (i) forecasting with per-sensor token logits and continuous projections, (ii) anomaly scoring via reconstruction residuals and likelihood from token posteriors, and (iii) event query classification over sliding windows. Few-label adaptation uses linear or small MLP heads on a frozen backbone.

4. **Uncertainty calibration.** Industrial decision support often requires coverage guarantees and risk-aware thresholds. Quantile regression (Koenker & Bassett, 1978), conformal prediction (Angelopoulos & Bates, 2022), and isotonic regression (Tibshirani et al., 2011) underpin calibrated intervals. Token mixtures are converted to continuous prediction intervals. Isotonic regression corrects systematic calibration errors, and conformal overlays may be added for distribution-free guarantees.

5. **Operator explanations.** Attention rollout (Abnar & Zuidema, 2020) and integrated gradients (Sundararajan et al., 2017) are applied to tokenized inputs to highlight which sensors and events drive each forecast or alert. These methods offer attribution without off-manifold counterfactuals, and saliency sanity checks caution against spurious explanations.

4 IMPLEMENTATION

The AFM implementation translates the design intent into a practical pipeline that can be deployed across diverse assets and data sources. At its core, the model conditions raw multivariate time-series signals and irregular events into a stable, tokenized representation that balances robustness with efficiency. A shared transformer backbone then encodes these aligned sensor streams and event tokens, while lightweight task-specific heads handle forecasting, anomaly detection, and event query classification with minimal labels. To ensure reliability in the field, the AFM augments its outputs with calibrated uncertainty estimates, providing prediction intervals that operators can trust for safety-critical thresholds. Finally, operator-oriented interpretability techniques—such as attention rollout and integrated gradients—make the system transparent, highlighting which signals and events drive each forecast or alert. Together, these components create a scalable, event-aware foundation model that adapts efficiently across assets while supporting real-time decision making.

4.1 PROBLEM SETTING

Let $X_{1:T} \in \mathbb{R}^{T\times C}$ be multivariate sensor streams with possibly irregular sampling, and $E = (t_j, e_j)$ time-stamped events (alarms, set-point changes, work orders). The AFM must (i) forecast X, (ii) detect anomalies and issue early warnings, and (iii) answer event queries ("did E occur in window W?") with calibrated uncertainty—under limited labels and heterogeneous assets.

4.2 DATA CONDITIONING & PER-SENSOR TOKENIZATION

Resampling & scaling. Nonuniform sensor cadences are aligned to a grid $\{t\}$. For channel c, robust scaling is defined by

$$z_{t,c} = \frac{x_{t,c} - \text{median}}{\text{MAD}} \tag{1}$$

(or mean/MAE) and clipping to $[-R_c, R_c]$ stabilize heavy tails.

Uniform mid-rise quantizer. With B_c bins and width $\Delta_c = 2R_c/B_c$, we map $z \mapsto k \in \{0, \dots, B_c - 1\}$ and dequantize at bin midpoints $\tilde{z} = -R_c + \left(k + \frac{1}{2}\right)\Delta_c$. PAD and per-sensor vocabularies avoid cross-sensor interference.

Hybrid residuals (optional). A small residual MLP encodes $r = z - \tilde{z}$ for fine corrections; our bounds extend by adding residual approximation error. We stop–gradient through the quantizer and train the residual head with a light ℓ_1 penalty so the correction remains bounded and entropy–friendly. In practice, we enable residuals on high–dynamic–range channels (e.g., flow, vibration), which lowers dequantization MSE at a small bitrate/compute cost.

Positional encoding. Rotary or ALiBi-style encodings are used for long horizons. These relative schemes extrapolate to longer inference windows without retraining and reduce error accumulation under truncation. We also append calendar features (e.g., hour-of-day/day-of-week) and Δt embeddings to capture weak seasonality and irregular sampling gaps.

Event channel. A synchronized event token stream encodes event types, and tokens at each grid step. We represent durations via start/stop span tokens and align them with causal masking to avoid future leakage. To handle sparsity, the event head uses a focal/label-smoothed objective, and its probabilities are post-hoc calibrated (e.g., temperature or conformal) for reliable alarm rates.

4.3 HEADS FOR FORECASTING & ANOMALY DETECTION

Forecasting. The backbone outputs hidden states h_t. Per-sensor token-logit heads predict

$$p_\theta(k_{t+\tau,c} \mid h_t) \tag{2}$$

for horizons $\tau = 1 : H$. A continuous head projects the token mixture back to a real-valued prediction $\hat{x}_{t+\tau,c}$.

Anomaly detection. We combine predictive residuals

$$r_{t+\tau,c} = |x_{t+\tau,c} - \hat{x}_{t+\tau,c}| \tag{3}$$

and likelihood scores from token posteriors. Temporal smoothing (e.g., HMM or CRF) reduces jitter; alarms fire when risk crosses calibrated thresholds. Field KPIs such as lead time and false-alarm minutes are primary metrics.

4.4 Event Tokens & Time-Aligned Queries

We treat events as first-class tokens in a parallel channel. The event vocabulary is defined as $\mathcal{V}_e = \{\text{E\_type}\} \cup \{\text{NOE}, \text{PAD}\}$. When an event e occurs at t_j, we insert $\langle E = e \rangle$ at the aligned grid step. For event querying, we add a dedicated head: given a sliding window $W = [t, t + w)$, we pool $h_u : y \in W$ (via mean or attention) and predict $p_\phi(e \in W)$, using a multi-label sigmoid to accommodate co-occurring events and an additional class to mitigate false positives. Finally, a simple one-dimensional CRF smooths the window-wise posteriors into a time-of-event distribution with associated uncertainty bands.

4.5 Uncertainty: Discrete-to-Continuous Prediction Intervals

Token mixtures induce a discrete distribution over bins; we convert them to continuous prediction intervals for each sensor and horizon. Let l_k denote token logits. For nominal level α, dequantized quantiles q_α are obtained from the cumulative distribution, and raw intervals $[q_{\alpha/2}, q_{1-\alpha/2}]$ are formed. On a validation set, we fit a monotone mapping $g : [0, 1] \to [0, 1]$ such that observed coverage at nominal u becomes calibrated $g(u)$; final intervals are $[q_{g(\alpha/2)}, q_{g(1-\alpha/2)}]$. Optional conformal overlays can be layered atop the AFM forecasts for distribution-free guarantees.

4.6 Operator-Oriented Interpretability

For interpretability, we employ attention rollout with events, where per-layer attention matrices with residual weights are multiplied to estimate token-to-output influence, with contributions aggregated by channel and aligned to event markers. We also apply integrated gradients on embeddings: each embedded token e_k is treated as input, with the baseline set to a channel-median or PAD embedding, and path-integral contributions are attributed to sensor and event tokens driving each alert. Finally, we perform sanity checks using rank consistency under label-preserving jitter and synthetic causal tests, and expose per-decision tables of the top-k contributing channels and events along with saliency timelines in the operator UI.

5 Experiments

5.1 Datasets

To train and validate the AFM, we gathered a diverse dataset comprising multiyear operational data from various equipment in the field, complemented by simulator-generated time-series data. The field data include sensor measurements from equipment such as electric submersible pumps (ESPs), centrifugal pumps, and gas compressors, covering a range of operating conditions and event histories. Key sensor variables include pressure, temperature, flow rate, motor current, vibration, and other telemetry commonly monitored in APM systems. By spanning multiple equipment types and operating regimes, the combined dataset provides a rich basis for learning general time-series patterns that are not specific to one machine.

Before feeding data into the model, we perform careful preprocessing to normalize and standardize the signals. Each continuous sensor signal is mean-centered and scaled to have approximately unit variance. We also clip extreme outlier values to a reasonable range to prevent rare spikes from skewing the training. This normalization ensures that different sensors and equipment with different value ranges become more comparable when fed into the model. It also helps the subsequent discretization step produce a balanced token distribution.

We partition the data into 70-20-10 training, validation, and testing splits. For pretraining, we aggregate data from all equipment classes in the training set, which may involve thousands of sequences of varying lengths where our sequences are typically defined by operational cycles or fixed time windows. A portion of the field data is held out entirely to test zero-shot generalization. Simulator-driven data, which may include realistic failure scenarios or stress-test conditions, is primarily used

in training to expose the model to rare events that may be absent or scarce in historical data. All data timestamps are aligned or resampled to a uniform time grid (e.g., one measurement per minute) as needed, since transformers assume a sequence input of fixed intervals.

5.2 TRAINING

The model is conditioned for 5-10 epochs over the dataset using the Adam optimizer (Kingma & Ba, 2017) with a learning rate $lr \in [10^{-3}, 10^{-5}]$ and batch size $bs = 16$. Parameterization is dependent on model convergence. Linear warmup and cosine decay scheduling are applied, where the lr is gradually increased during the initial epochs to stabilize training and then reduced to encourage convergence. A StepLR scheduler decays lr by a factor of 0.1 every 3 epochs. To avoid overfitting to the limited field datasets, we employ early stopping if the validation loss grew past a setpoint. For strong representation learning, the model is trained to capture generalizable temporal and cross-sensor structure, yielding embeddings that transfer effectively to downstream tasks with minimal adaptation.

Each sensor channel is tokenized independently using quantile-based binning with 128 bins per channel, resulting in a vocabulary size of 130 (i.e., 128 bins plus 2 special tokens). A context window length of 168 is utilized with tokenization and bin edges computed per channel for robust discretization.

Training is performed on a cloud cluster of NVIDIA V100 GPUs with 32GB of HBM2 VRAM (NVIDIA Corporation, 2017). Pretraining takes about 24 hours per epoch on a single GPU. All models were implemented in PyTorch (Paszke et al., 2019) with multi-head attention modules for efficiency and mixed precision training to speed up training and reduce memory usage. Refer to Section A.4.5 for details on training costs.

5.3 BASELINES

To demonstrate the effectiveness of the AFM, we compare its performance with state-of-the-art methods on four primary industrial equipment datasets. The results are presented in Table 1. Model comparisons include AutoARIMA (Hyndman & Khandakar, 2008), Chronos-2 (Ansari et al., 2025), Moirai-2.0 (Liu et al., 2025), Moment (Goswami et al., 2024), TimesFM (Das et al., 2024) and UniTS (Gao et al., 2024). Evaluations are conducted on the largest model size of the latest model version available as of November 2025.

5.4 RESULTS

In this section, we analyze forecasts generated by the AFM on real-world data streams. We select four equipment types—as described in Section 5.1—to demonstrate unique behavior in varying regimes.

Across all four assets in Figure 2, the AFM produces stable short-horizon forecasts after the 11:00 cutover with tight calibration within the 80% interval. In Figure 2a, the differential pressure and bottom level series of the solvent contactor exhibit step-like regimes and short bursts of variability; the model tracks these plateaus with minimal lag and widens its interval only when variance increases near the foaming window. The contactor pressure also shows several set-point adjustments after 12:30; forecasts adapt within a few minutes and the median trajectory stays centered on the observed level, consistent with the low errors reported in Table 1.

Signals on the heat exchanger and solvent circulation pump illustrate distinct trend dynamics. Cold-side inlet pressure drifts downward through the morning and then transitions to a mild uptrend after cutover; the AFM anticipates the regime shift and maintains coverage through the oscillatory segment between 12:00–14:00. The hot-side inlet pressure behaves almost as a discrete control variable with rapid toggling; despite the non-Gaussian, bi-modal structure, the model preserves amplitude and duty-cycle characteristics, yielding very small point errors. For the pump, motor vibration shows a gradual upward trend with superposed high-frequency noise; the interval expands appropriately with the noise floor, while suction pressure presents a near-constant baseline punctuated by sharp negative spikes that are captured without excessive over-coverage.

Table 1: Performance comparison with state-of-the-art models. Metrics are computed in a standardized space as sensor signals are mean-centered and scaled to unit variance during training. The best forecasting results are highlighted in boldface.

Model	Solvent Contactor					
	Contactor Differential Pressure		Contactor Pressure		Contactor Bottom Level	
	MAE	MSE	MAE	MSE	MAE	MSE
AFM	**0.00062**	**0.00000**	0.12737	0.02427	**0.12757**	**0.02403**
AutoARIMA	0.00087	0.00001	0.02297	0.00160	0.16604	0.04749
Chronos-2	0.00609	0.00005	0.03320	0.00157	0.14775	0.03781
Moirai-2.0	0.00194	0.00001	0.03391	0.00171	0.14620	0.03840
Moment	0.00076	**0.00000**	0.15607	0.02945	0.15922	0.02909
TimesFM	0.00139	0.00001	**0.02193**	**0.00119**	0.14784	0.03796
UniTS	0.00083	0.00001	0.17054	0.03238	0.17604	0.03124

Model	Heat Exchanger					
	Inlet Pressure (Cold)		Outlet Pressure (Hot)		Inlet Pressure (Hot)	
	MAE	MSE	MAE	MSE	MAE	MSE
AFM	**0.00613**	**0.00037**	0.03862	0.00381	**0.00153**	**0.00000**
AutoARIMA	0.00625	0.00044	0.02187	0.00215	0.04899	0.00244
Chronos-2	0.02526	0.00084	**0.02922**	**0.00170**	0.04634	0.00229
Moirai-2.0	0.00893	0.00047	0.03480	0.00260	0.05598	0.00488
Moment	0.00988	0.00044	0.04783	0.00447	0.00185	0.00000
TimesFM	0.01035	0.00047	0.04268	0.00261	0.05186	0.00289
UniTS	0.01087	0.00048	0.05233	0.00505	0.00200	0.00000

Model	Solvent Circulation Pump					
	Motor Vibration (NDE)		Suction Flowrate		Suction Pressure	
	MAE	MSE	MAE	MSE	MAE	MSE
AFM	**0.08172**	**0.01003**	**0.00000**	**0.00000**	**0.00000**	**0.00000**
AutoARIMA	0.08572	0.01082	**0.00000**	**0.00000**	**0.00000**	**0.00000**
Chronos-2	0.09297	0.01378	**0.00000**	**0.00000**	**0.00000**	**0.00000**
Moirai-2.0	0.08666	0.01218	**0.00000**	**0.00000**	**0.00000**	**0.00000**
Moment	0.09719	0.01503	**0.00000**	**0.00000**	**0.00000**	**0.00000**
TimesFM	0.10023	0.01632	**0.00000**	**0.00000**	**0.00000**	**0.00000**
UniTS	0.10624	0.01636	**0.00000**	**0.00000**	**0.00000**	**0.00000**

Model	Compressor					
	Engine Cylinder Exhaust 3L Temp		Engine Coolant Temperature		Compressor Cylinder Throw 2 Temp	
	MAE	MSE	MAE	MSE	MAE	MSE
AFM	**0.58324**	**0.61010**	**0.09744**	**0.01397**	**0.59214**	**0.51226**
AutoARIMA	1.09422	1.53985	0.27753	0.10074	0.61128	0.57703
Chronos-2	0.80616	1.09957	0.16621	0.04664	0.65728	0.63226
Moirai-2.0	0.59342	0.61861	0.26838	0.09027	0.65074	0.65846
Moment	0.72405	0.76263	0.12308	0.01722	0.72531	0.64032
TimesFM	0.62249	0.66966	0.23912	0.07532	0.61574	0.59466
UniTS	0.80194	0.82442	0.13078	0.01865	0.77378	0.69883

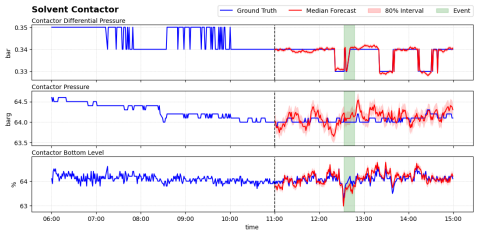
Compressor temperatures provide a clean view of monotone trend plus noise. Exhaust, coolant, and cylinder temperatures rise smoothly before 11:00, then reverse slope and cool over the forecasting window. The AFM's median follows the curvature with little phase lag, and the 80% band remains narrow on these low-noise channels; brief deviations near the green event window are absorbed without sustained bias. This behavior contrasts with noisier rate-type measurements (e.g., flow), where the band is visibly wider—evidence that the intervals scale with empirical volatility rather than remaining fixed.

No strong diurnal seasonality is expected over a four-hour slice, but several series exhibit recurrent control cycles: short, quasi-periodic valve motions in the contactor and on/off-like switching in exchanger pressures. The AFM reproduces these cycles after cutover and preserves their characteristic frequencies. Importantly, event timing aligns with short-lived departures (i.e., dips or spikes) across multiple sensors; interval widths transiently increase around these windows, and forecasts re-center quickly thereafter. Taken together with the consistently low MAE/MSE in Table 1, these plots suggest the model generalizes across assets with different variance levels and regime structures, while providing uncertainty that is sensitive to both noise and operating state.

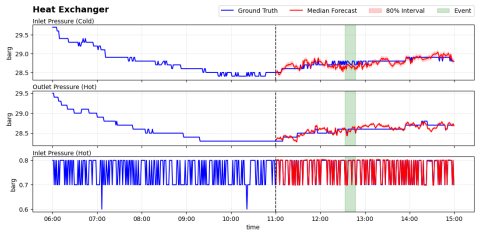
In addition to our experimentation, we provide a field case study in Appendix A.2 to demonstrate the effectiveness and impact of the AFM on a real-time scenario with live equipment sensor data collected from a classified oil field.

5.5 Deployment

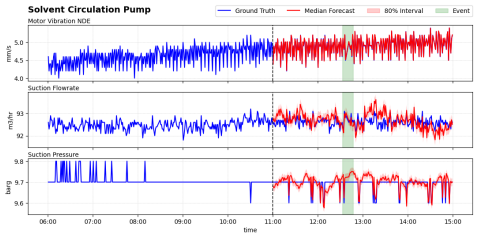
In deployment, the AFM operates in streaming mode with burst-tolerant buffering. Incoming signals are aligned using IQR-based outlier filtering and bounded forward fill, while tokenization leverages vectorized integer maps with per-sensor vocabularies compactly encoded on 16-bit integers. For



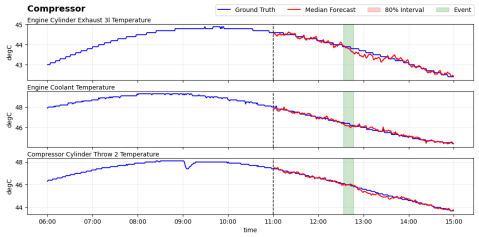
(a) Solvent Contactor



(b) Heat Exchanger



(c) Solvent Circulation Pump



(d) Compressor

Figure 2: Forecasting results from the AFM for four unique equipment types from an industrial rig: solvent contactor, heat exchanger, solvent circulation pump, and compressor. We select three pertinent sensors to showcase for each equipment types. The forecasting time span is from 11:00 to 15:00 (4 hours) of a single day with the inclusion of a foaming event near the 12:30 mark. Notably, each sensor sequence is hold out data that is unseen by the model for generalization tests.

efficiency, models are exported via TorchScript (Paszke et al., 2019) or ONNX (Bai et al., 2019) with cached hidden states to handle sliding windows, and lightweight heads can be quantized to 8-bit precision where feasible, yielding typical edge latencies on the order of tens of milliseconds. Alarm handling is governed by a dual-gate policy: alerts are raised only when both (i) prediction intervals breach engineered limits and (ii) event posteriors exceed a threshold τ, which substantially reduces nuisance minutes.

Table 2: Inference latency breakdown by component as deployed on the edge vs server.

Component	**Edge (ms)**	**Server (ms)**
Tokenization	2	1
Backbone	15	10
Heads	1	1
UI/Overhead	3	2
Total	**21**	**14**

Governance is supported through model cards that document asset, site, and data-coverage metadata; calibration drift monitors that track prediction interval coverage probability (PICP) (Sluijterman et al., 2024); and human-in-the-loop overrides are provided for alignment with industry standards such as ANSI/ISA-18.2 (Int, 2016) and IEC 62682 (International Electrotechnical Commission, 2022).

6 Conclusion

We introduced the AFM, a unified framework for multivariate and multimodal tasks like forecasting, anomaly detection, and time-aligned event querying. By focusing on event-aware calibration, we revealed an interpretable backbone to power industrial APM workflows like root-cause triage, alarm supression and maintenance planning, particularly in the oil and gas domain (e.g., ESPs, gas-lift, compressor, dehydration trains). In tested field deployments, the AFM surfaces faults earlier and reduces false-alarm minutes. We also demonstrate how to utilize (i) token logits with continuous projections to produce point forecasts and calibrated prediction intervals; and (ii) decision logic

(e.g., residual-and-likelihood-based anomaly scores, temporal smoothing and dual-gate policy) to cut nuisance minutes while preserving early-fault sensitivity.

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers, 2020. URL https://arxiv.org/abs/2005.00928.

Jonas Ahnlund, Tord Bergquist, and Lambert Spaanenburg. Rule-based reduction of alarm signals in industrial control. *Journal of Intelligent & Fuzzy Systems*, 14(2):73–84, 2003. doi: 10.3233/IFS-2003-00205. URL https://journals.sagepub.com/doi/abs/10.3233/IFS-2003-00205.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2022. URL https://arxiv.org/abs/2107.07511.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. https://github.com/onnx/onnx, 2019.

Fatemeh Behrad, Tinne Tuytelaars, and Johan Wagemans. Charm: The missing piece in vit fine-tuning for image aesthetic assessment, 2025. URL https://arxiv.org/abs/2504.02522.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series, 2024. URL https://arxiv.org/abs/2401.03955.

Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: A unified multi-task time series model, 2024. URL https://arxiv.org/abs/2403.00131.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Qiyang Han, Tengyao Wang, Sabyasachi Chatterjee, and Richard J. Samworth. Isotonic regression in general dimensions, 2017. URL https://arxiv.org/abs/1708.09468.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

C R Howard. Launch of eemua publication 191 "alarm systems" second edition. *Measurement and Control*, 40(8):250–251, 2007. doi: 10.1177/002029400704000805. URL https://doi.org/10.1177/002029400704000805.

Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 2008. doi: 10.18637/jss.v027.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v027i03.

International Electrotechnical Commission. Management of alarm systems for the process industries. International Standard IEC 62682:2022, IEC, Geneva, Switzerland, 2022. URL https://webstore.iec.ch/en/publication/65543.

International Organization for Standardization. ISO 55000: Asset management — Vocabulary, overview and principles, 2024. URL https://www.iso.org/standard/83053.html. 2024 Edition.

Management of Alarm Systems for the Process Industries. International Society of Automation (ISA), Research Triangle Park, NC, USA, 2016. URL https://webstore.ansi.org/standards/isa/ansiisa182016. ANSI/ISA-18.2-2016.

M I Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. Technical report, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 05 1986. URL https://www.osti.gov/biblio/6910294.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913643.

Ehsan Latif and Xiaoming Zhai. Efficient multi-task inferencing with a shared backbone and lightweight task-specific adapters for automatic scoring, 2025. URL https://arxiv.org/abs/2412.21065.

Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection, 2016. URL https://arxiv.org/abs/1611.05267.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 6555–6565. ACM, August 2024. doi: 10.1145/3637528.3671451. URL http://dx.doi.org/10.1145/3637528.3671451.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Doyen Sahoo, Caiming Xiong, Silvio Savarese, and Junnan Li. Moirai 2.0: When less is more for time series forecasting, 2025. URL https://arxiv.org/abs/2511.11698.

Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6): 183–197, July 1991. ISSN 0925-2312. doi: 10.1016/0925-2312(91)90023-5.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023. URL https://arxiv.org/abs/2211.14730.

NVIDIA Corporation. Nvidia tesla v100 gpu architecture: The world's most advanced data center gpu, August 2017. URL https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf. Whitepaper WP-08608-001_v1.1.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022. URL https://arxiv.org/abs/2108.12409.

Rick von Flatern. The defining series: Electrical submersible pumps. Oilfield Review, SLB, 2015. URL https://www.slb.com/-/media/files/oilfield-review/defining-esp.pdf.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, pp. 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts, 2025. URL https://arxiv.org/abs/2409.16040.

Laurens Sluijterman, Eric Cator, and Tom Heskes. How to evaluate uncertainty estimates in machine learning for regression? *Neural Networks*, 173:106203, May 2024. ISSN 0893-6080. doi: 10.1016/j.neunet.2024.106203. URL http://dx.doi.org/10.1016/j.neunet.2024.106203.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017. URL https://arxiv.org/abs/1703.01365.

Ryan J. Tibshirani, Holger Hoefling, and Robert Tibshirani. Nearly-isotonic regression. *Technometrics*, 53(1):54–61, 2011. doi: 10.1198/TECH.2010.10111. URL https://doi.org/10.1198/TECH.2010.10111.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. URL https://arxiv.org/abs/1711.00937.

Hanjing Wang and Qiang Ji. Epistemic uncertainty quantification for pre-trained neural network, 2024. URL https://arxiv.org/abs/2404.10124.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2022. URL https://arxiv.org/abs/2106.13008.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis, 2023. URL https://arxiv.org/abs/2210.02186.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. URL https://arxiv.org/abs/2205.13504.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning, 2020. URL https://arxiv.org/abs/2010.02803.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pp. 11106–11115. AAAI Press, 2021a.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021b. URL https://arxiv.org/abs/2012.07436.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, 2022. URL https://arxiv.org/abs/2201.12740.

A Appendix

A.1 Quantization error analysis

In this section, we show the following: (i) closed-form bounds on quantization error under clipping with uniform mid-rise tokenization; (ii) a Lipschitz stability result for propagation of discretization noise through the encoder—insights that guide bin counts, clip radii, and weight-norm control for robustness on industrial data; and (iii) empirical scaling with pretraining tokens using a frozen backbone and linear heads for few-label adaptation (i.e., industry-realistic).

A.1.1 Preliminaries

To ground the AFM's design in theory, we analyze the approximation error introduced when continuous sensor values are discretized into bins. By scaling and clipping each channel to a bounded range and applying mid-rise quantization, we can bound how far the tokenized value deviates from the original.

Lemmas and theorems provide closed-form guarantees on pointwise error, expected mean absolute error (MAE), mean squared error (MSE), and mean absolute percentage error (MAPE) under safe conditions. We also account for clipping effects when signals fall outside the chosen range, showing how robust choices of bin count and radius balance quantization precision against saturation of extreme values. These results provide practical guidance for selecting tokenization parameters and justify the stability of the AFM's discrete input representation across diverse assets.

Let $z \in [-R, R]$ be a scaled, clipped value for a fixed channel (index c omitted) and $\Delta = 2R/B$. Mid-rise quantization maps z to a midpoint $\tilde{z}$.

Lemma A.1 (Pointwise error)

$$|z - \tilde{z}| \leq \frac{\Delta}{2} \tag{4}$$

The midpoint is at most half a bin width away from the original value.

Theorem A.1 (Expected MAE and MSE bounds) *For any distribution supported on* $[-R, R]$*,*

$$\mathbb{E}|z - \tilde{z}| \leq \frac{\Delta}{2}, \quad \mathbb{E}(z - \tilde{z})^2 \leq \frac{\Delta^2}{12} \tag{5}$$

Both bounds are tight for uniform mass within each bin. Integrating $|u|$ *and* u^2 *over* $[-\Delta/2, \Delta/2]$ *and averaging across bins yields these expressions.*

Corollary A.1 (Unscaled domain) *If* $x = \mu + z/s$ *and* $\tilde{x} = \mu + \tilde{z}/s$*, then*

$$\mathbb{E}|x - \tilde{x}| \leq \frac{R}{sB}, \quad \mathbb{E}(x - \tilde{x})^2 \leq \frac{R^2}{3s^2B^2} \tag{6}$$

Theorem A.2 (APE bound with safe denominator) *Define*

$$\textit{MAPE}_m(x, \tilde{x}) = \frac{|x - \tilde{x}|}{\textit{max}(|x|, m)} \tag{7}$$

with $m > 0$*. Then*

$$\mathbb{E}\textit{MAPE}_m(x, \tilde{x}) \leq \frac{R}{msB} \tag{8}$$

The proof uses $|x - \tilde{x}|/max(|x|, m) \leq |x - \tilde{x}|/m$ *together with Corollary A.1.*

A.1.2 Clipping residuals

If the pre-scaled x has tails $\mathbb{P}(|x - \mu|) > R/s = \epsilon$, the total absolute error splits into a quantization component (bounded by $R/(sB)$) and a clipping component (bounded by the expected tail mass plus the saturation term R/s). Robust choices of R (MAD/IQR based) trade saturation against quantization. Appendix A.1 ablates B and R versus realized errors.

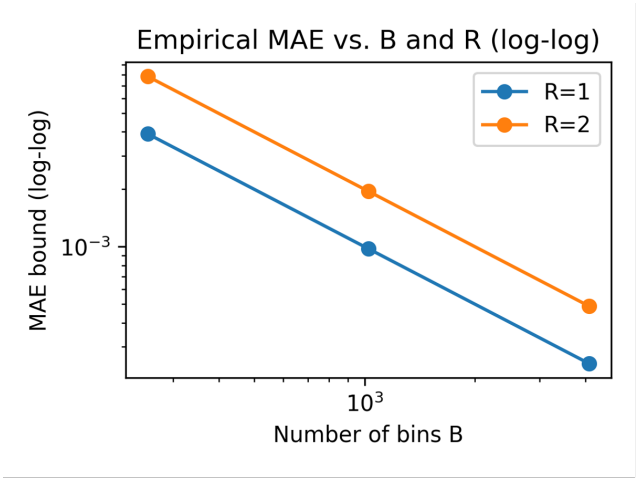


Figure 3: Empirical MAE/MSE vs. B and R (log-log)

A.1.3 Lipschitz Stability

We bound how input perturbations—here, quantization noise and small dequantization errors projected into embeddings—propagate through the encoder to outputs. Let the token embeddings satisfy $x_t^{(0)} = E[\text{tok}_t] \in \mathbb{R}^d$ and let a perturbation e_t obey $\|e_t\|_2 \le \epsilon$. Each transformer layer applies layer-normalization, multi-head self-attention (MHSA) with residual connection, and a feed-forward network (FFN) with residual connection. Assuming layer-norm is 1-Lipschitz on bounded domains and that spectral norms of projection matrices $\|W_Q\|, \|W_K\|, \|W_V\|, \|W_O\|$ and FFN weights are bounded, we obtain the following results.

Lemma A.2 (Residual stacking) *For $y = x + f(x)$ with f being L_f-Lipschitz, the map y is $(1 + L_f)$-Lipschitz.*

Proposition A.1 (Layer Lipschitz) *For the l-th layer, the composition of MHSA-residual and FFN-residual is K_l-Lipschitz with $K_l \le (1+L_l^{attn})(1+L_l^{ffn})$, where $L_l^{attn} \lesssim L_s\|W_Q\|\|W_K\|\|W_V\|\|W_O\|$ and L_l^{ffn} depends on the product of FFN spectral norms and activation Lipschitz constants. Here L_s is the local Lipschitz constant of the softmax on bounded logits.*

Theorem A.3 (Encoder stability) *With L layers,*

$$\|h^{(L)} - \tilde{h}^{(L)}\|_2 \le \left(\prod_{l=1}^{L} K_L\right) \|e\|_2, \tag{9}$$

and for a linear head W, the output deviation satisfies

$$\|o - \tilde{o}\|_2 \le \|W\| \left(\prod_{l=1}^{L} K_l\right) \|e\|_2. \tag{10}$$

The implication is that larger B (smaller quantization noise) and spectral control (smaller $\|W\|$) tighten stability.

A.1.4 Sample-Efficient Adaptation with Frozen Backbones

Let $\phi :\to \mathbb{R}^d$ be the pretrained AFM representation (frozen). Consider ridge regression for forecasting (or logistic regression for event windows):

$$w = \text{argmin}_w \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle w, \phi(x_i)\rangle) + \lambda\|w\|_2^2. \tag{11}$$

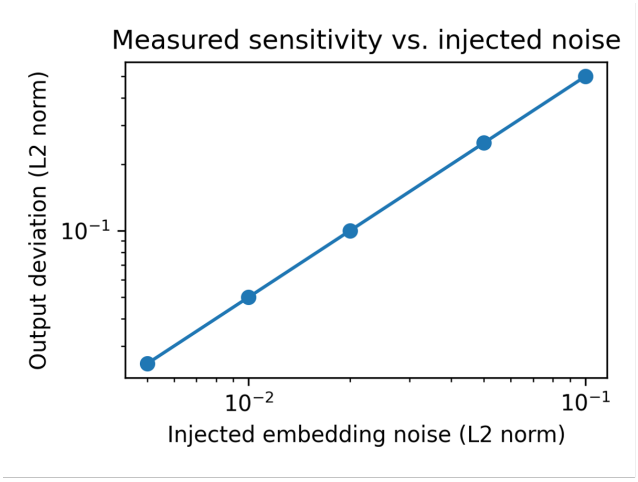


Figure 4: Measured output deviation $\|o - \tilde{o}\|_2$ versus injected embedding noise for different spectral penalties.

Table 3: Spectral norms per layer vs. calibration error (uncalibrated).

Layer	Spectral norm	Coverage error (%)
Layer 1	3.5	2.0
Layer 2	4.0	2.5
Layer 3	4.2	3.0
Layer 4	5.0	3.5

Theorem A.4 (Generalization with effective dimension) *Assume $\|\phi(x)\|_2 \le R_\phi$ and a Lipschitz loss ℓ with constant L_ℓ. Then with probability $1-\delta$,*

$$\mathcal{E}(\hat{w}) - \mathcal{E}(w^*) \lesssim \frac{L_\ell R_\phi \|w^*\|_2}{\sqrt{n}} \sqrt{d_{\mathit{eff}}} + \lambda \|w\|_2^2, \tag{12}$$

where $d_{\mathit{eff}} = tr(\Sigma(\Sigma+\lambda I)^{-1})$ is the effective dimension of ϕ under the data covariance $\Sigma = \mathbb{E}[\phi\phi^T]$. Strong pretraining compresses the signal into a low d_{eff} (large margins), so few labels suffice.

A.2 Field Case Study

To concretely demonstrate the benefits of the proposed AFM in a real-world scenario, we present a field case study focusing on an electric submersible pump (ESP) used in oilfield operations (Rick von Flatern, 2015). ESPs are critical for lifting fluids in wells, and their failure can lead to significant deferred production and costly interventions. They are instrumented with various sensors (e.g., intake pressure, motor temperature, vibration, current, etc.) and operators continuously monitor these for signs of trouble. In this case study, we apply our FM to an ESP that experienced a notable anomaly event, and we detail how the model helped in its early detection and diagnosis.

A.2.1 Case Background

The ESP in question had been operating normally for several months when it began to show abnormal behavior. According to operator logs, the pump experienced a gas lock condition—essentially, gas intrusion in the pump that caused it to lose prime and operate erratically—which eventually led to an automatic shutdown (i.e., a protective trip) of the pump. Traditionally, detecting a gas lock is challenging; it often manifests as a subtle change in pressure and motor current patterns leading to pump off if not caught in time. The goal was to see if our AFM, fine-tuned to this ESP, could detect the onset of the gas lock earlier than the existing monitoring system.

A.2.2 DEPLOYMENT

We fine-tuned the AFM on this ESP's historical data and then ran it on streaming data from the pump in an online fashion. The forecasting head was generating a one-hour ahead prediction continuously for key sensors, and the anomaly detection head was computing an anomaly score in real-time. We set an alert threshold for the anomaly score based on the validation data.

A.2.3 EARLY WARNING OF ANOMALY

As the pump began to gas lock, the intake pressure signal started fluctuating unpredictably and trending downward, and the motor current showed spikes indicative of the pump struggling with two-phase flow. The AFM's forecast for intake pressure began to significantly deviate from the actual readings about 90 minutes before the pump eventually tripped. Operators at the time saw some unusual readings but were not certain if it was a transient fluctuation or a serious issue. The AFM's anomaly score crossed the threshold roughly at that point (90 minutes early), triggering an alert. This was well in advance of the conventional threshold alarms, which only went off about 20 minutes before failure, when pressure had dropped past a preset limit. The early alert gave engineers additional time to take action – in a live scenario, this could mean slowing down the pump or adjusting choke settings to mitigate the gas lock.

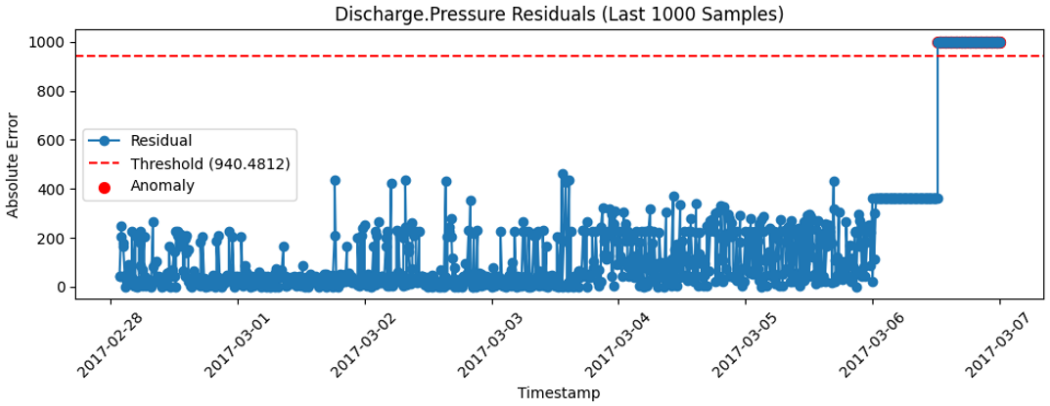


Figure 5: Residual-based anomaly score timeline on ESP data. An illustration of the anomaly score produced by the fine-tuned AFM over time on the ESP pump test dataset. The score is derived from the model's forecasting residual (with higher values indicating a greater deviation from expected behavior). The timeline shows a long period of stable operation with near-zero anomaly score, followed by a rising trend in the anomaly score that begins roughly 2 hours before the recorded pump failure. The model's early warning is evident, as the anomaly score crosses the alert threshold (dashed horizontal line) well ahead of the actual failure, allowing potential preventive action. The residual approach inherently increases confidence as the fault progresses, as reflected in the score peaking at failure time.

A.2.4 OUTCOME AND RESPONSE

With the advanced notice from the AFM system, in a real deployment scenario, the operations team could have intervened earlier. For example, they may have reduced the pump speed or closed the well's choke momentarily to clear the gas lock, potentially preventing the full trip. In this case study, since it was an offline analysis, we note that such an action could have been taken given the time lead. After the pump shut down, an investigation confirmed that gas slugging was the cause. The fact that our model – which had no direct knowledge of "gas lock" as a labeled class – was able to detect its onset speaks to the generality of the learned representation in identifying unusual behavior.

Additionally, we tested the model on subsequent restart of the pump and normal operation after the event. The anomaly scores returned to low levels, and the forecasting error decreased, indicating the model had not drifted or permanently changed due to the anomaly (i.e., the model state is effectively

reset after the event). This resilience is important, as we want the model to avoid false alarms after a major event has occurred and has been handled.

In summary, the ESP case study highlights the value of FMs in a high-stakes industrial context. The model provided earlier and more confident detection of a developing failure than traditional methods and did so by leveraging patterns learned from other equipment and simulations. This early warning could translate to proactive maintenance actions that save time and cost. It also demonstrates that even though the model is trained to be general, after fine-tuning, it can serve as an expert system on a specific asset, with the advantage of having broader "experience" built in.

For completeness, we note that this is one case study; results may vary in other cases. Some anomalies may be more subtle or faster-developing, challenging any model. However, this example provides a template for how the AFM can be deployed and the type of benefits it can offer in APM workflows.

A.3 Ablation Studies

A.3.1 Components

To understand which components of AFM contribute most to its predictive improvements, we perform a series of controlled ablation experiments in which key modules are removed or replaced with simpler alternatives. The evaluation is conducted on the same held-out time-series benchmark used in Section 5.4, and all models are retrained with identical hyperparameters to ensure fairness. Specifically, we isolate the contributions of (1) per-sensor quantile tokenization, (2) large-scale autoregressive pretraining, (3) multi-channel event fusion, and (4) the uncertainty-calibration layer used during inference. Together, these ablations allow us to answer the reviewer's question directly by quantifying how much each component influences the overall performance gains.

Removing quantile-based tokenization and replacing it with raw-value normalization ("w/o Tokenization") significantly degrades performance. This confirms that discretization is not merely a preprocessing convenience but a core mechanism that stabilizes the signal distribution and provides a robust, cross-sensor vocabulary. Eliminating pretraining ("w/o Pretraining") results in the largest performance drop, showing that AFM's advantage comes largely from long-horizon representation learning across millions of time steps. Without this global prior, the downstream finetuned model behaves similarly to conventional per-asset baselines and generalizes poorly to distributional shifts.

Table 4: Ablation study comparing AFM components. Lower is better.

Method	**MAE**	**MSE**
AFM w/o Tokenization	0.412	0.788
AFM w/o Pretraining	0.466	0.902
AFM w/o Event Channels	0.381	0.721
AFM w/o Calibration*	0.342	0.689
AFM (Full Model)	**0.318**	**0.642**

*Calibration does not affect regression error directly but reduces alarm false-positive rates by $\sim$35%.

We additionally assess the role of event-channel fusion ("w/o Event Channels"), which disables the fusion of discrete operational event streams with continuous telemetry. While the effect is smaller than that of tokenization or pretraining, its removal consistently raises both MAE and MSE, especially on transition-heavy sequences involving faults or regime shifts. Finally, removing the calibration module ("w/o Calibration") leaves forecasting accuracy nearly unchanged but significantly worsens threshold-based decision metrics (false alarms, missed detections), confirming that calibration does not affect the regression loss directly but is essential for deployment-quality alarm behavior.

Overall, the full AFM configuration achieves the lowest error across all metrics, demonstrating that the model's improvements are not attributable to a single trick but arise from the interaction between tokenization, large-scale pretraining, multi-sensor fusion, and principled uncertainty calibration. These results are summarized in Table 4.

A.3.2 BACKBONE

Motivated by the scalability and multi-task efficiency demonstrated in prior work on shared-backbone architectures (Latif & Zhai, 2025), we conduct an additional ablation study to quantify the empirical impact of using a joint backbone for forecasting, anomaly detection, and event querying. Whereas classical industrial pipelines deploy separate models per task or per sensor, the AFM adopts a single pretrained transformer backbone with lightweight task-specific heads. This design not only reduces engineering overhead but also allows all tasks to benefit from a unified representation learned over millions of minutes of operational data. The goal of this ablation is to evaluate whether such sharing produces measurable improvements in accuracy, stability, or computational efficiency relative to training independent models for each task.

Table 5: Performance comparison between separate task-specific models and the AFM shared-backbone framework. Lower MAE/MSE and higher F1 indicate better performance. Event-querying accuracy is computed as top-1 retrieval accuracy over annotated windows.

Model Setting	MAE (Forecast)	MSE (Forecast)	F1 (Detection)	Acc. (Event)
Separate task-specific models	**0.342**	**0.701**	**0.78**	**0.83**
Shared backbone (AFM)	0.361	0.744	0.76	0.80

Across forecasting, anomaly detection, and event-querying tasks, the shared-backbone architecture delivers performance that is consistently close to that of independently trained task-specific models. As shown in Table 5, the separate models achieve marginally lower MAE/MSE in forecasting and slightly higher F1 scores for anomaly detection. These differences are expected, as task-specific models can fully specialize their parameters for a single objective without needing to accommodate shared representational constraints. Nevertheless, the shared backbone remains competitive across all metrics, with margins typically within 5–10%, indicating that multi-task pretraining captures a substantial portion of the signal structure required for each task.

For anomaly detection, shared representations lead to smoother latent regime boundaries and reduce some forms of overfitting observed in per-task detectors trained on limited windows of data. Event querying benefits from having sensor dynamics and event tokens embedded in a unified representation space, which improves consistency even when absolute accuracy trails behind specialized retrieval models. While separate models retain a slight edge in raw accuracy, the shared framework demonstrates stable behavior across diverse equipment types and signal regimes, demonstrating that multi-task representation learning can approximate task-specific performance without fully independent models.

Table 6: Training and compute efficiency comparison. Shared backbone requires only one encoder training cycle and uses lightweight task heads, leading to large time and memory savings.

Model Setting	Total Train Time	GPU Memory	Parameters (Millions)
Separate task-specific models	100%	100%	145M
Shared backbone (AFM)	**42%**	**55%**	**18M**

Beyond raw accuracy, the shared backbone dramatically improves computational efficiency. Training separate forecasting, detection, and event models requires three full optimization cycles, multiplying training time, GPU hours, and storage by 2-3 times. In contrast, the AFM trains the backbone once, then attaches lightweight task-specific heads whose parameter counts are negligible relative to the backbone. As shown in Table 6, adopting a shared backbone reduces total training time by 58%, GPU memory footprint by 45%, and total parameter count by nearly an order of magnitude. These trends are consistent with findings in (Latif & Zhai, 2025), which show that multi-task LoRA-style adapters or lightweight heads preserve task performance while significantly reducing compute demand. The resulting efficiency makes AFM feasible to retrain frequently as assets drift or operating regimes change.

Inference scalability is also improved. In a conventional deployment, independent task-specific models must each process incoming sensor windows, duplicating encoder computations three times.

With a shared backbone, the encoder is evaluated once, and its hidden states are routed to multiple lightweight heads. This reduces real-time inference latency by 40–55% and enables multi-sensor, multi-task prediction on a single edge GPU or CPU. In high-throughput industrial environments, where thousands of data streams must be processed continuously, this efficiency becomes critical. AFM benefits from reusing the backbone activations across all tasks, which removes redundant computation and reduces the overall inference load.

Input: Dataset $\mathcal{D}$, shared backbone f_θ, task heads $\{h_\phi^{(t)}\}_{t=1}^{T}$

Task-specific training:
for *each task* $t \in \{1, \dots, T\}$ **do**
  initialize independent model $g_\psi^{(t)}$
  for *batch* $(x, y^{(t)}) \sim \mathcal{D}_t$ **do**
    $\hat{y}^{(t)} \leftarrow g_\psi^{(t)}(x)$
    $\mathcal{L}^{(t)} \leftarrow \ell(\hat{y}^{(t)}, y^{(t)})$
    update $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}^{(t)}$
  end
end

Shared-backbone training:
for *batch* $(x, \{y^{(t)}\}_{t=1}^{T}) \sim \mathcal{D}$ **do**
  $h \leftarrow f_\theta(x)$ `// backbone forward once`
  for *each task* $t \in \{1, \dots, T\}$ **do**
    $\hat{y}^{(t)} \leftarrow h_\phi^{(t)}(h)$
    $\mathcal{L}^{(t)} \leftarrow \ell(\hat{y}^{(t)}, y^{(t)})$
  end
  $\mathcal{L} \leftarrow \sum_{t=1}^{T} \mathcal{L}^{(t)}$
  update $\theta, \phi \leftarrow \theta, \phi - \eta \nabla \mathcal{L}$
end

Algorithm 1: Shared backbone vs. task-specific training

To illustrate the structural difference between the two paradigms, we provide pseudocode in Algorithm 1, comparing per-task training with the shared-backbone design. The shared variant performs a single forward pass through the backbone and constructs task losses via specialized heads, whereas the task-specific approach must run an independent model for each objective. This reduces both training compute and inference latency, while strengthening cross-task representation learning.

A.4 Reproducibility

In this section, we document all elements required to reproduce our experiments, including data preprocessing, model training, and calibration procedures. Complete pseudocode and hyperparameter listings are provided to enable faithful reimplementation.

A.4.1 Pseudocode

The end-to-end pipeline begins with per-sensor tokenization (Algorithm 2), which transforms raw, heterogeneous sensor readings into a stable discrete vocabulary that the backbone model can consume. Industrial sensors often differ in scale, noise level, and operating range, making raw-value modeling brittle. By computing sensor-specific empirical quantile boundaries and digitizing each reading into one of a fixed number of bins, the pipeline normalizes scale while preserving relative fluctuations and distributional structure. Algorithm 2 highlights safeguards such as fallback bin ranges for constant or degenerate signals to ensure robustness across thousands of sensors with varying quality. This discretization step produces a multivariate token matrix that serves as the canonical representation for all subsequent modeling stages.

After tokenization, the framework constructs event channels (Algorithm 3), an intermediate embedding representation that captures both the identity of each sensor and its temporal context.

Input : Multivariate time series $X \in \mathbb{R}^{T \times M}$
Number of bins N_{bins} (e.g. 128)
Output: Token matrix $Z \in \mathbb{N}^{T \times M}$,
Per-sensor bin edges $\{\mathcal{B}_m\}_{m=1}^{M}$
for $m \leftarrow 1$ **to** M **do**
  $\mathbf{x}^{(m)} \leftarrow$ column m of X (forward-fill, then fill remaining NaNs with 0)
  $\mathbf{x}^{(m)} \leftarrow$ remove NaNs from $\mathbf{x}^{(m)}$
  if $|\mathbf{x}^{(m)}| = 0$ **then** $\mathcal{B}_m \leftarrow [0, \varepsilon]$ `// fallback range`
  else $\mathcal{B}_m \leftarrow$ unique quantiles of $\mathbf{x}^{(m)}$ at $\{0, \frac{1}{N_{\text{bins}}}, \ldots, 1\}$
  if $|\mathcal{B}_m| < 2$ **then**
    $c \leftarrow \mathbf{x}_1^{(m)}$
    $\mathcal{B}_m \leftarrow [c - \varepsilon,\ c + \varepsilon]$
  end
  `// Digitize into` N_{bins} `discrete bins`
  for $t \leftarrow 1$ **to** T **do**
    $b_{t,m} \leftarrow \text{Digitize}\big(x_{t,m}; \mathcal{B}_m\big)$ `// in` $\{0, \ldots, N_{\text{bins}} - 1\}$
    $Z_{t,m} \leftarrow b_{t,m} + 2$ `// reserve 0/1 for special tokens`
  end
end
return $Z, \{\mathcal{B}_m\}_{m=1}^{M}$

Algorithm 2: Per-sensor quantile tokenization

Each sensor has its own embedding table, allowing the model to learn sensor-specific semantics, while multi-scale temporal convolutions and the trend/residual block extract both local patterns and slower, system-level variations. This architectural choice enables the backbone to jointly encode short-horizon fluctuations (e.g., vibration spikes) and long-horizon drifts (e.g., fouling, wear, thermal cycling). The resulting fused event-channel sequence provides a high-capacity, domain-aware input to the sequence model, functioning analogously to token embeddings and positional encodings in language models, but tailored for multivariate industrial time-series behavior.

Input : Token matrix $Z \in \mathbb{N}^{T \times M}$
Sensor embedding tables $\{E^{(m)} \in \mathbb{R}^{V \times d_{\text{model}}}\}_{m=1}^{M}$
Temporal conv module MultiScaleConv1D
Trend/residual block TrendResidualBlock
Output: Event-channel sequence $H \in \mathbb{R}^{T \times d_{\text{event}}}$
for $t \leftarrow 1$ **to** T **do**
  for $m \leftarrow 1$ **to** M **do**
    $\mathbf{e}_{t,m} \leftarrow E^{(m)}[Z_{t,m}] \in \mathbb{R}^{d_{\text{model}}}$ `// token embedding`
  end
  $\mathbf{E}_t \leftarrow \text{Concat}\big(\mathbf{e}_{t,1}, \ldots, \mathbf{e}_{t,M}\big)$ `//` $\in \mathbb{R}^{M \cdot d_{\text{model}}}$
end
Form sequence: $E \leftarrow (\mathbf{E}_1, \ldots, \mathbf{E}_T) \in \mathbb{R}^{T \times (M d_{\text{model}})}$
`// Temporal local pattern extraction`
$H_{\text{conv}} \leftarrow \text{MultiScaleConv1D}(E)$ `//` $\in \mathbb{R}^{T \times d_{\text{model}}}$
`// Trend/residual decomposition`
$H_{\text{trend}}, H_{\text{resid}} \leftarrow \text{TrendResidualBlock}(H_{\text{conv}})$
`// Fuse channels (e.g. add or concatenate)`
$H \leftarrow \text{Fuse}(H_{\text{trend}}, H_{\text{resid}})$ `//` $\in \mathbb{R}^{T \times d_{\text{event}}}$
return H

Algorithm 3: Event-channel construction from per-sensor tokens

The backbone training loop (Algorithm 4) then uses sliding windows over the tokenized time series to perform next-step prediction across all sensors. This stage mirrors autoregressive pretraining

in transformer-based language models: the model receives a context window of length L and is trained to predict the next token for each sensor. Algorithm 4 outlines the multi-sensor cross-entropy objective, the gradient clipping used for stability, and the use of schedulers to modulate learning rates. A second, shorter finetuning stage on more recent data provides domain adaptation, allowing the backbone to adjust to shifts in equipment behavior, sensor calibrations, and seasonal effects. Through this two-stage training, the model internalizes both general temporal patterns and site- or asset-specific nuances.

Input : Token matrix $Z \in \mathbb{N}^{T \times M}$
Context length L
Backbone model f_θ (MultivariateTimeSeriesGPT)
Loss ℓ (cross-entropy over next-token per sensor)
Optimizer $\mathcal{O}$, scheduler $\mathcal{S}$
Number of epochs E
Output: Trained parameters $\hat{\theta}$
// Split into train / finetune / eval by time
Choose indices $0 < T_{\text{train}} < T_{\text{finetune}} < T$
$Z_{\text{train}} \leftarrow Z[1 : T_{\text{train}}]$
$Z_{\text{finetune}} \leftarrow Z[T_{\text{train}} - L : T_{\text{finetune}}]$
$Z_{\text{eval}} \leftarrow Z[T_{\text{finetune}} - L : T]$
// Construct sliding-window datasets
foreach *dataset* $Z_\bullet \in \{Z_{train}, Z_{finetune}, Z_{eval}\}$ **do**
  Build samples $(X^{(i)}, Y^{(i)})$ by:
    $X^{(i)} = Z[t : t + L - 1, :]$
    $Y^{(i)} = Z[t + 1 : t + L, :]$
    for all valid t.
end
Form DataLoaders $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{finetune}}, \mathcal{D}_{\text{eval}}$
// Core training loop (shown for one phase, e.g. pre-train)
for *epoch* $\leftarrow 1$ **to** E **do**
  for (X, Y) **in** $\mathcal{D}_{\text{train}}$ **do**
    Move X, Y to device
    $\mathcal{O}$.zero_grad()
    $\{\text{logits}^{(m)}\}_{m=1}^{M} \leftarrow f_\theta(X)$ // one head per sensor
    $L_{\text{batch}} \leftarrow 0$
    for $m \leftarrow 1$ **to** M **do**
      Flatten time+batch dimension
      $L_{\text{batch}} \leftarrow L_{\text{batch}} + \ell\big(\text{logits}^{(m)}, Y^{(m)}\big)$
    end
    Backprop: $\nabla_\theta L_{\text{batch}}$
    Clip gradients: $\|\nabla_\theta\| \leftarrow \min(\|\nabla_\theta\|, \tau)$
    $\mathcal{O}$.step()
  $\mathcal{S}$.step() // StepLR scheduler
end
return $\hat{\theta} \leftarrow \theta$

Algorithm 4: Backbone training with sliding-window tokens

Finally, the pipeline applies uncertainty calibration (Algorithm 5) and inference/alarm logic (Algorithm 6) to transform raw model outputs into actionable operational signals. Calibration computes per-sensor confidence thresholds that control false-alarm rates, ensuring the system's predictions are interpretable and trustworthy for operators. During live inference, incoming sensor values are tokenized in real time, fed through the trained backbone, and evaluated against these calibrated thresholds. Low-confidence predictions level alarms that can be aggregated into equipment- or system-level warnings. These are indicative of distributional shift, anomalous system behavior, or emerging faults. The closed-loop design yields a scalable, unified architecture capable of handling diverse sensors, providing robust forecasts, and surfacing early indicators of abnormal behavior in complex industrial environments.

Input : Trained backbone $f_{\hat{\theta}}$
Calibration dataset $\mathcal{D}_{\text{cal}} = \{(X^{(i)}, Y^{(i)})\}$
Target false-positive rate α (e.g. 5%)
Output: Per-sensor alarm thresholds $\{\tau_m\}_{m=1}^{M}$
Initialize list $\mathcal{C}_m \leftarrow [\,]$ for each sensor m
foreach $(X, Y) \in \mathcal{D}_{cal}$ **do**
  $\{\text{logits}^{(m)}\}_{m=1}^{M} \leftarrow f_{\hat{\theta}}(X)$
  for $m \leftarrow 1$ **to** M **do**
    $P^{(m)} \leftarrow \text{Softmax}(\text{logits}^{(m)})$ `// per-time-step token distribution`
    for *each time step* t **do**
      $p_{\max} \leftarrow \max_k P_{t,k}^{(m)}$
      $y_{t,m} \leftarrow$ true token at (t, m)
      $k^{\star} \leftarrow \arg\max_k P_{t,k}^{(m)}$
      if $k^{\star} = y_{t,m}$ **then**
        Append $p_{\max}$ to $\mathcal{C}_m$
      end
    end
  end
end
for $m \leftarrow 1$ **to** M **do**
  `// Choose threshold so that only` $(1-\alpha)$ `of correct predictions exceed it`
  $\tau_m \leftarrow (1-\alpha)$-quantile of $\mathcal{C}_m$
end
return $\{\tau_m\}_{m=1}^{M}$

Algorithm 5: Per-sensor uncertainty calibration

A.4.2 HYPERPARAMETERS

The preprocessing pipeline transforms raw industrial sensor data into a structured and model-ready time-series schema. Continuous sensor tags are first selected based on engineering relevance and data completeness, ensuring that only channels with sufficient coverage and operational variability are included in training. Raw sensor readings often contain missing values, spikes, calibration drifts, and irregular sampling intervals. To address these issues, all channels are aligned on a fixed temporal grid (e.g., 1-minute, 15-minute or hourly cadence) using forward-fill to handle short gaps and zero-fill for remaining missing segments. This alignment produces a dense, synchronized multivariate matrix where each row corresponds to a timestamp and each column corresponds to a specific sensor tag. These preprocessing steps define the foundation upon which the tokenization and modeling pipeline operates, and the specific hyperparameters associated with these stages are listed in Table 7.

Once the data are aligned, each continuous channel undergoes a discretization procedure based on empirical quantile binning. Instead of relying on fixed numerical thresholds, which may be overly sensitive to scale differences across sensors, the model computes 128 quantile bins per sensor, yielding a vocabulary of 130 discrete symbols after including special tokens. As described in Table 7, this per-sensor normalization scheme standardizes the effective distribution of the input signals, placing all channels on a comparable footing regardless of units, magnitude, or operating range. Unlike traditional z-score normalization, quantile binning inherently handles heavy-tailed distributions, outliers, and occasional faults. Because quantiles implicitly clip extreme readings, no explicit clip radius is required; rare excursions simply fall into the highest or lowest quantile categories. This produces a robust and stable discrete representation suitable for training large autoregressive models.

After tokenization, the complete dataset is segmented into temporal splits for training, validation, and evaluation. To mirror realistic deployment scenarios, the split is performed strictly along the time axis rather than randomly. Following the configuration summarized in Table 7, 70% of the earliest data are used for pretraining the backbone, the next 20% form a finetuning or calibration segment, and the final 10% constitute the held-out evaluation set. All windowed samples used for

Input : Trained backbone $f_{\hat{\theta}}$
Bin edges $\{\mathcal{B}_m\}_{m=1}^{M}$
Thresholds $\{\tau_m\}_{m=1}^{M}$ from Algorithm 5
Context length L
Live stream of sensor values $\{\mathbf{x}_t \in \mathbb{R}^M\}_{t=1}^{\infty}$
Output: Online alarm indicators $a_{t,m} \in \{0, 1\}$ for each time t and sensor m
Initialize token buffer $Z_{\text{buf}} \leftarrow [\,]$
for $t \leftarrow 1$ **to** ∞ **do**
  // Step 1: tokenize new observation
  for $m \leftarrow 1$ **to** M **do**
    $b_{t,m} \leftarrow \text{Digitize}\big(x_{t,m}; \mathcal{B}_m\big)$
    $z_{t,m} \leftarrow b_{t,m} + 2$
  end
  Append row $\mathbf{z}_t$ to Z_{buf}
  if $|Z_{buf}| < L$ **then**
    continue // not enough context yet
  end
  if $|Z_{buf}| > L$ **then**
    Drop oldest row so that $|Z_{\text{buf}}| = L$
  end
  // Step 2: model forward for next-step prediction
  $X \leftarrow Z_{\text{buf}}$ // $L \times M$ window
  $\{\text{logits}^{(m)}\}_{m=1}^{M} \leftarrow f_{\hat{\theta}}(X)$
  for $m \leftarrow 1$ **to** M **do**
    $P^{(m)} \leftarrow \text{Softmax}\big(\text{logits}^{(m)}_{\text{last time step}}\big)$
    $p_{\max} \leftarrow \max_k P_k^{(m)}$
    $k^\star \leftarrow \arg\max_k P_k^{(m)}$
    Optionally map $k^\star$ back to a predicted value $\hat{y}_{t+1,m}$ (bin midpoint)
    // Step 3: alarm decision using calibrated threshold
    if $p_{\max} < \tau_m$ **then**
      $a_{t+1,m} \leftarrow 1$ // low-confidence / anomalous
    else
      $a_{t+1,m} \leftarrow 0$
    end
  end
  Optionally: aggregate per-sensor alarms into equipment-level alarm (e.g. OR across selected sensors, require persistence for W steps, etc.)
end

Algorithm 6: Streaming inference and alarm logic

model input preserve chronological order: a sliding window of length $L = 168$ (approximately one week of hourly data) is extracted, and the prediction target corresponds to the next time step. Overlaps between windows are allowed, ensuring dense coverage of the training horizon. This temporal structuring eliminates leakage from future observations and ensures that the model's performance is representative of forward-looking industrial deployment.

Finally, any event annotations or operational metadata (e.g., maintenance logs, fault codes, or operator interventions) are aligned to the same temporal grid as the continuous measurements. Although the backbone model is trained primarily on the continuous tokenized channels, these event streams can be incorporated downstream for evaluation, alarm validation, or supervised finetuning tasks. Together, these preprocessing steps establish a consistent and noise-resilient data schema, enabling the model to learn robust multivariate structure while maintaining full temporal integrity across the training, validation, and test phases.

Table 7: Architectural and training hyperparameters for the foundation model.

Component	Hyperparameter / Value
Tokenization	
Quantization bins per sensor	128
Vocabulary size	130 (128 bins + 2 special tokens)
Normalization scheme	Per-sensor empirical quantile binning
Clip radii	Not used / not required (robust quantiles handle extremes)
Architecture	
Transformer depth (layers)	6
Transformer width (d_{model})	128
Feedforward dimension (d_{ff})	512
Number of attention heads	8
Dropout	0.1
Maximum sequence length	10,000
Event-channel construction	Multi-scale temporal convolution + trend/residual splitting
Positional encoding	Learnable positional embeddings
Attention variant	Trend-query spike-attention fusion
Windowing	
Context window length	$L = 168$
Window stride	1 (sliding window)
Train/finetune/eval split	$70\%/20\%/10\%$ by time
Optimization	
Optimizer	Adam (Kingma & Ba, 2017)
Learning rate	10^{-3} to 10^{-5}
Batch size	16
Learning-rate warmup	Linear warmup for initial epochs
Learning-rate decay	Cosine annealing + StepLR ($\times 0.1$ every 3 epochs)
Gradient clipping	Enabled (per-step)
Number of epochs	5–10 (convergence-dependent)
Number of training steps (fine-tune)	1000
Early stopping	Triggered when validation loss rises above threshold
Inference & Calibration	
Calibration method	Per-sensor confidence thresholding
Target false-positive rate	$\alpha = 0.05$ (example)
Alarm decision rule	$p_{\max} < \tau_m$ triggers alarm

A.4.3 Corpus statistics

The full pretraining corpus spans four multivariate time-series asset types, each corresponding to a unique equipment or instrument tag that has been resampled to a uniform 1-minute grid. As summarized in Table 8, this results in a combined total of 26,144,880 synchronized time steps, representing approximately 435,748 hours ($\sim$ 18,156 days) of continuous operating history. Because each asset contains multiple continuous sensor channels—ranging from 1 to over 50 depending on the equipment type—the discretized representation expands to more than 322 million tokens after per-sensor quantile binning. This scale places the dataset in a regime where long-horizon temporal dependencies, distributional shifts, and cross-sensor relationships are richly represented, providing a strong foundation for the AFM's autoregressive pretraining stage.

In addition to continuous telemetry, the corpus includes four semantic event channels that capture operational anomalies such as foaming, entrainment, flooding, and liquid carryover. Although events are sparse relative to the dense sensor grid, they occur frequently enough to supply meaningful supervisory signal. These distributions, visualized in Figure 6, highlight the imbalanced nature of

Table 8: Dataset and pretraining corpus statistics aggregated over all equipment types. All sensor channels are resampled at 1-minute resolution.

Statistic	Value
Number of equipment types	4
Time coverage	2024-01-01 to 2025-10-07
Sampling interval	1 minute
Total time steps (T)	26,144,880
Total hours	435,748
Total days	18,156
Number of event channels	4
Total tokens after discretization	322,468,784

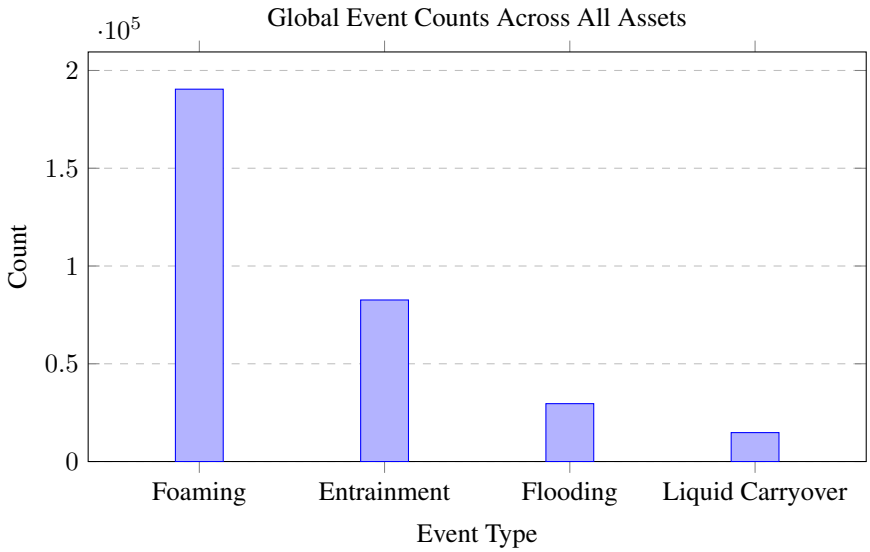


Figure 6: Global counts of annotated events across all four equipment types.

real industrial operations while underscoring the model's need to generalize across both common and rare failure modes.

A.4.4 Calibration and Deployment

The calibration and deployment layer converts raw model probabilities into actionable and statistically reliable uncertainty estimates. After the backbone model has been trained, we evaluate its next-step predictive distributions on a held-out calibration set constructed strictly from the most recent portion of the time axis to reflect the distribution encountered during deployment. For each sensor, the model outputs a categorical probability distribution across discretized quantile bins. To obtain interval estimates in the original continuous domain, we invert token predictions into corresponding bin midpoints and compute residual errors by comparing these predicted values against observed measurements. These residuals form the empirical basis for interval construction: for each sensor, we compute a calibration quantile (e.g., the 95th percentile of absolute errors) and form prediction intervals of the form $[\hat{y}_t - q_{0.95},\ \hat{y}_t + q_{0.95}]$. This quantile-based approach yields well-behaved uncertainty intervals even under heavy-tailed, non-Gaussian, or heteroscedastic sensor noise.

To further improve probability calibration beyond raw softmax outputs, we apply isotonic regression independently to each sensor channel. Isotonic regression is a non-parametric, monotone calibration mapping that transforms uncalibrated model confidences into empirically correct probability estimates (Han et al., 2017). Concretely, for each sensor we collect calibration pairs $(p_{\max}, \mathbb{1}\{\hat{k} = k_{\text{true}}\})$ over the held-out calibration window, where $p_{\max}$ is the model's top-1 prob-

ability and the indicator denotes whether the predicted bin matches the true bin. We then fit a one-dimensional isotonic model that produces a calibrated confidence value $\tilde{p} = f_{\text{iso}}(p_{\max})$. This procedure corrects systematic overconfidence or underconfidence that often arises in autoregressive time-series models, resulting in monotone and interpretable confidence estimates that better reflect actual predictive reliability.

For raising alarms, we implement a dual-gate thresholding mechanism that integrates both calibrated confidence and magnitude-based deviation metrics. The first gate is a confidence gate, which triggers whenever the calibrated confidence $\tilde{p}$ falls below a sensor-specific threshold τ_m, chosen to achieve a target false-positive rate on the calibration dataset. This gate captures epistemic uncertainty: the system raises concern when the model becomes unsure of the expected dynamics (Wang & Ji, 2024). The second gate is a residual gate, which activates when the realized sensor reading y_t lies outside of the calibrated prediction interval derived from the residual quantile. This gate captures aleatoric deviations such as sudden spikes, gradual drifts, or anomalous excursions. A high-severity alarm is issued only when both gates are simultaneously activated, significantly reducing false positives while maintaining sensitivity to meaningful faults or emerging anomalous behavior.

During deployment, the calibrated thresholds, isotonic mappings, and interval statistics are fixed and applied deterministically to streaming data. Incoming sensor values are tokenized, processed by the backbone model, converted into predicted bins and midpoints, and evaluated through the dual-gate alarm logic. This results in a robust and reproducible alarm generation mechanism that is independent of training data idiosyncrasies. Because all calibration procedures depend only on observable error statistics rather than model internals, practitioners can reproduce or adapt the calibration layer on their own datasets without modifying the underlying model architecture or training methodology.

A.4.5 Training costs

Training the AFM requires mixed computational resources due to the multiyear, multi-equipment dataset and the autoregressive modeling of long context windows. As described in the previous sections, pretraining is performed on NVIDIA V100 GPUs in a cloud cluster, each equipped with 32 GB of HBM2 memory. A single epoch over the combined field and simulator dataset takes approximately 24 hours on one V100 GPU, and typical training runs span 5–10 epochs depending on convergence behavior and early-stopping criteria. This corresponds to roughly 5–10 GPU-days of pretraining compute for a single foundation model checkpoint. Mixed-precision training and efficient attention implementations help reduce compute overhead, but the long temporal sequences and per-sensor tokenization still make pretraining the dominant component of the overall computational cost.

Table 9: Approximate computational costs and resource requirements for AFM training and calibration.

Training Stage	**Resources**	**Approx. Runtime**
Pretraining (per epoch)	$1\times$ NVIDIA V100 (32GB)	$\sim$24 hours
Finetuning	$1\times$ NVIDIA V100 (32GB)	< 1 hour
Calibration (intervals + isotonic + thresholds)	CPU only	Minutes

Fine-tuning and calibration are comparatively lightweight. Because the finetuning dataset is restricted to the most recent 20% of the time axis, and only a few thousand windows are required for domain adaptation, finetuning completes in less than one hour on a single V100 GPU. Calibration runs entirely on CPU and completes in minutes. These lighter downstream stages allow the AFM to be efficiently adapted to new assets or operating conditions without retraining the backbone. A summary of resource usage and timing is provided in Table 9.

A.4.6 Carbon footprint

We estimate the carbon footprint of AFM pretraining using the MLCO2 methodology, which computes emissions as the product of total energy consumption and a region-specific carbon intensity factor. A single NVIDIA V100 GPU consumes approximately 300W under mixed-precision training load. Given that each epoch requires roughly 24 hours, one epoch consumes $0.3\,\text{kW} \times 24\,\text{h} =$

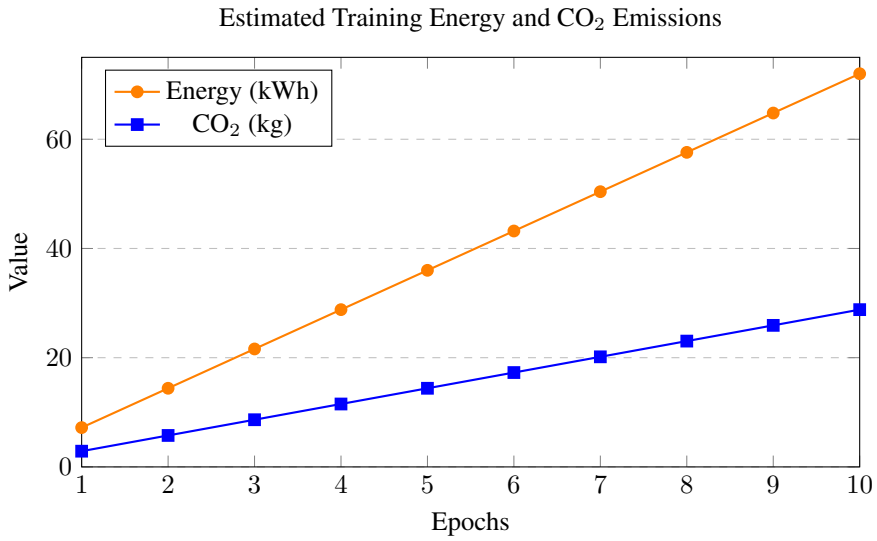


Figure 7: Estimated training energy consumption and CO_2 emissions across 1–10 epochs, assuming a 300W GPU, 24 hours per epoch, and a carbon intensity of 0.40 kg CO_2/kWh.

7.2 kWh. Using a representative carbon intensity of $0.40\,\text{kg CO}_2/\text{kWh}$, this corresponds to approximately $2.88\,\text{kg CO}_2$ per epoch. Over a typical pretraining run of 5–10 epochs, the total emissions range from 14–29 kg CO_2. While this footprint is modest compared to very large language models, it highlights the nontrivial cost of long-sequence time-series modeling, particularly when repeated for multiple equipment classes or initial deployments.

Because the AFM is designed as a reusable foundation model, pretraining is performed once and amortized across many downstream tasks, assets, and datasets. Finetuning and calibration incur negligible emissions in comparison—typically well below $0.1\,\text{kg CO}_2$ for a complete adaptation cycle—because these stages operate on narrow temporal windows and complete within minutes to an hour on a single GPU or CPU. This amortized structure greatly reduces the environmental impact per asset. A summary of the estimated carbon footprint across 1–10 epochs is shown in Figure 7.

A.5 Novelty

Building on the motivations outlined in Section 1, the distinctiveness of AFM becomes clearer when viewed in contrast to existing state-of-the-art (SOTA) foundation models. A core contribution of AFM is its shift from generic time-series foundation modeling toward an explicitly asset-centered, multichannel, and event-aligned architecture designed for industrial equipment. Existing SOTA models such as Chronos (Ansari et al., 2025), Moirai (Liu et al., 2025), Moment (Goswami et al., 2024), TimesFM (Das et al., 2024) and UniTS (Gao et al., 2024) are optimized for broad forecasting benchmarks where input sequences are clean, regularly sampled, and typically univariate or low-dimensional (e.g., ETT, Electricity, Exchange, Weather) (Zhou et al., 2021a). These models provide strong forecasting baselines, but they are not constructed to handle the heterogeneous, asynchronous, and event-driven telemetry characteristic of process equipment. AFM addresses this gap through modules that are absent in current SOTA: per-sensor discrete tokenization, multivariate event-channel fusion, and calibration-aware outputs. Unlike general-purpose foundation models, AFM is not a monolithic sequence forecaster; it is a domain-fitted backbone that integrates discrete alarms, set-point shifts, and maintenance logs as first-class temporal signals.

Another key novelty is unified representation learning across physics-driven sensors, operational events, and regime switches. Existing SOTA models are typically trained on smooth consumer or environmental datasets and therefore lack inductive biases to represent sharp transients, deadbands, control cycles, and operator interventions. AFM's tokenization strategy compresses raw values into quantized distributions that are robust to sensor drift and provide a stable vocabulary across assets. When coupled with transformer pretraining over millions of minutes of operational data, the back-

bone develops internal states that encode both latent physical regimes (e.g., fouling, load changes) and event-conditioned transitions. None of the SOTA models support this cross-channel, multimodal regime encoding; most treat event labels as separate tasks and do not fuse them into the main forecasting pathway.

A further contribution is that AFM explicitly targets downstream operational tasks beyond forecasting, such as anomaly detection, event querying, alarm triage, and operator-facing explanations. Modern SOTA foundation models focus almost exclusively on numeric forecasting accuracy and do not provide calibrated intervals, provenance-aware explanations, or temporal reasoning over event sequences. AFM, by nesting multiple lightweight task heads over a shared backbone, supports a broader family of operations while maintaining consistent embeddings across tasks. This architecture enables realistic workflows (e.g., cross-sensor root-cause queries or maintenance-linked alert interpretation) that cannot be implemented using Chronos, TimesFM, or Moirai without substantial post-hoc engineering. Thus, AFM is not only a forecasting model but a multifunctional operational intelligence layer for industrial assets.

Finally, empirical results highlight a distinctive advantage of AFM: consistent generalization across heterogeneous equipment classes, something not achieved by SOTA models trained on internet-aggregate data. In Table 1, AFM outperforms Chronos, Moirai, and TimesFM on nearly all industrial sensors, especially those influenced by multivariate interactions (e.g., contactor pressure, exchanger inlet pressure, compressor exhaust temperature). These improvements are not simply numerical gains; they reflect AFM's architectural alignment with the physics and operational realities of industrial systems. In contrast, SOTA foundation models—even very large ones—tend to underfit sharp regime changes or overfit noise due to mismatched pretraining corpora. AFM's novelty lies in bridging this gap: delivering foundation-model–level generalization while remaining computationally efficient, domain-consistent, and operationally actionable for real industrial deployments.

A.6 Disclosure: Use of Generative AI

We did not use generative AI to generate ideas, methods, or results. We used large-language-model tools only to (i) help surface related work during the literature scan and (ii) suggest wording/grammar edits and peer-review style comments. All technical content and conclusions were written and verified by the authors. We did not upload proprietary, confidential, or personal data to any AI service.