From MCU to Neuromorphic Chip: A Zero-Gradient Spiking-Compatible Engine for Cross-Domain Predictive Maintenance

Jianwei Lou[1]

[1] *RailMind Systems, Neuss, Germany*
j.lou@railmind.eu

ABSTRACT

Real-time predictive maintenance on resource-constrained edge hardware demands sub-millisecond inference, online adaptation without retraining, and deployment across heterogeneous industrial domains. We present a zero-gradient neural dynamics engine: a population of computational units governed entirely by local plasticity rules, discrete population-level gating, and resource-constrained structural adaptation. The frozen model occupies 50–100 KB with no GPU dependency. We evaluate seven task configurations across four physical datasets (bearing fault diagnosis on CWRU and Paderborn, audio machine monitoring on DCASE, human activity recognition on UCI HAR, and a pooled multi-domain configuration that pools these datasets for in-distribution cross-validation rather than domain transfer), plus one real-world tunnel construction monitoring deployment, using a single unmodified engine configuration. Against a simple raw-feature baseline (the same downstream classifier applied to un-engineered features), the engine improves mean accuracy by +14.2 percentage points; the benefit concentrates where raw features are not pre-engineered, and we do not claim superiority over classical methods that use domain-specific feature engineering. On the four bearing fault tasks, multi-seed consensus voting reaches $\geq$99.9%. End-to-end inference latency is 53 μs on x86, 103 μs on ARM (Raspberry Pi 5), and $\sim$350 μs on a Cortex-M7 MCU at 600 MHz. All engine primitives operate without surrogate gradients or backpropagation, and exact SNN mapping to a neuromorphic chip has been verified. Engine routing is fully unsupervised: online k-means discovers K regimes from the input stream without consuming labels. A one-time calibration step of 50–200 labeled samples, supplied either by a domain expert or by a cloud-based LLM oracle, then maps these discovered regimes to domain-specific fault categories; thereafter the engine runs autonomously.

1. INTRODUCTION

Predictive maintenance (PdM) in industrial settings sets a tight constraint envelope: sub-millisecond response, online adaptation to non-stationary operating conditions, instant switching between diagnostic tasks, and deployment on power-constrained edge hardware (Lei et al., 2020). Conventional gradient-trained pipelines clear the accuracy bar in the lab. They miss most of the rest. Offline retraining, GPU acceleration, and per-domain feature engineering each limit cross-industry use (Zhao et al., 2019).

Spiking neural networks (SNNs) offer a biologically plausible alternative with demonstrated efficiency on neuromorphic hardware such as Intel's Loihi 2 (Davies et al., 2021). However, current SNN training relies on surrogate gradient methods (Neftci et al., 2019) or backpropagation through time (BPTT). Both methods require non-local learning signals. Such signals are awkward to implement on-chip and they limit online adaptation.

A third paradigm exists in the computational neuroscience literature: population-level neural dynamics governed entirely by local update rules. Complex behavior emerges from the interaction of simple units under resource constraints (Schuman et al., 2022; Lansner, 2009). Such systems need no global loss and no gradient. They still self-organize and adapt, driven by biologically inspired resource management (Turrigiano & Nelson, 2004).

In this paper, we present a zero-gradient neural dynamics engine that implements this paradigm for industrial PdM. The engine comprises a population of N computational units with D-dimensional state vectors, updated exclusively through three local mechanisms: (i) Hebbian-type plasticity for unsupervised feature learning, (ii) discrete population-level gating for context-selective activation, and (iii) resource-constrained structural adaptation for maintaining representational capacity under finite budgets. The frozen model occupies 50–100 KB (depending on population size) and requires no GPU.

Our contributions are:

1. A zero-gradient engine validated across **seven task configurations** on four physical datasets plus one real-world industrial deployment, with a single unmodified configuration (Section 4).
2. Characterization of the engine as a **conditional amplifier**: against a simple raw-feature baseline, its benefit is largest when the classifier is given un-engineered features (up to $+49$ pp on the weakest raw baselines) and is neutral-to-slightly-negative when raw features are already strong. We do not claim superiority over classical methods with domain-specific feature engineering (Section 5).
3. End-to-end deployment on four hardware tiers (x86 at 53 μs, ARM Cortex-A76 at 103 μs, Cortex-M7 600 MHz single-core at $\sim$350 μs, and verified SNN mapping to a neuromorphic chip), demonstrating viability from server to sub-milliwatt edge (Section 4).

1.1. Related Work

Gradient-based PdM. Deep learning has become the dominant approach to fault diagnosis and prognosis. Convolutional and recurrent architectures applied to vibration signals (Zhao et al., 2019; Khan & Yairi, 2018) report 95–99% accuracy on bearing benchmarks, while deep autoencoders (Lei et al., 2020) and attention-based models dominate anomaly detection on satellite telemetry (Hundman et al., 2018). These methods still depend on offline GPU training, per-dataset hyperparameter tuning, and large labeled corpora. None of these conditions hold for cross-domain edge deployment.

Spiking neural networks for PdM. Two training paradigms dominate SNN deployment on neuromorphic hardware. Surrogate-gradient methods (Neftci et al., 2019) replace the non-differentiable spike with a smooth proxy. This restores backpropagation. The cost is a non-local credit-assignment signal that runs across the whole network. BPTT extends the same idea to temporal sequences and inherits the same non-locality. Recent industrial machine-learning surveys (Yamazaki et al., 2022) summarize early SNN PdM deployments. The reported task scope is narrow.

Local-learning models. A separate line of research replaces global gradients with locally computed updates. Spike-Timing-Dependent Plasticity (STDP) models (Bi & Poo, 1998; Diehl & Cook, 2015) update synaptic weights from pre- and post-synaptic spike timing alone; reward-modulated variants (Mozafari et al., 2019) add a global scalar to make STDP supervised. Equilibrium Propagation (Scellier & Bengio, 2017) uses two-phase relaxation dynamics to compute updates equivalent to gradient descent without backward passes. Predictive coding networks (Whittington & Bogacz, 2017) approximate backprop using only local prediction-error signals. Bayesian Confidence Propagation Neural Networks (BCPNN) (Lansner, 2009; Tully et al., 2014) use Hebbian-style log-probability updates without any gradient. These approaches share our zero-gradient principle but have largely been validated on individual benchmark tasks (typically MNIST or one bearing dataset) rather than cross-domain industrial use.

Reservoir and liquid computing. Reservoir computing (Tanaka et al., 2019; Verstraeten et al., 2007) and liquid state machines (Maass et al., 2002) use a fixed random recurrent substrate with only the readout layer trained, often by ridge regression. They share our "no internal gradient" design but typically retain a supervised, gradient-trained linear readout, whereas our calibration step is a static lookup between engine expert groups and class labels, populated from 50–200 oracle-supplied labels (human expert or LLM).

Homeostatic and competitive regularization. Keeping representational diversity without an explicit loss term has been studied in computational neuroscience for decades. Synaptic scaling (Turrigiano, 2008) adjusts unit gain so that the long-term firing rate stays close to a target value; k-winner-take-all networks (Maass, 2000; Rumelhart & Zipser, 1985) produce sparsity through lateral inhibition. Our resource-constrained adaptation principle (Section 2) draws directly on these two ingredients.

2. SYSTEM ARCHITECTURE

2.1. Engine Design Principles

The engine is described here at the observable-behavior level, consistent with the proprietary nature of the implementation. Three design principles distinguish it from both gradient-trained networks and conventional SNNs:

Principle 1: Zero global gradients. No parameter in the engine is updated by a gradient derived from a global loss function. All plasticity is strictly local: each unit's state evolves based only on local information. This eliminates the need for backpropagation infrastructure and enables fully decentralized computation.

Definition (zero-gradient). A learning system is *zero-gradient* if no parameter update is computed from a global loss gradient $\nabla_\theta \mathcal{L}$. All updates depend exclusively on locally available signals: the unit's own state and signals from immediately presynaptic units within a fixed spatial neighborhood. This excludes standard backpropagation and surrogate-gradient methods used in SNN training (Neftci et al., 2019). It admits Hebbian plasticity (Lansner, 2009), homeostatic adaptation (Turrigiano & Nelson, 2004), k-winner-take-all competitive gating (Maass, 2000), and resource-bounded structural updates that depend only on local state.

Principle 2: Discrete population-level gating. The engine

employs a mixture-of-experts (MoE) architecture (Jacobs et al., 1991) where a discrete gating mechanism routes input signals to subpopulations of units. Unlike soft attention mechanisms in transformer architectures, the gating is hard: at each inference step, only a subset of the population is active. This provides natural sparsity and enables instantaneous task switching by changing which expert subpopulation is active.

Principle 3: Resource-constrained competitive adaptation. The population operates under finite resource budgets. Three biological regularization mechanisms work together to keep representational diversity without an explicit loss function term:

1. *Per-unit firing-rate homeostasis* in the spirit of Turrigiano-style synaptic scaling (Turrigiano, 2008): each unit's effective gain is adjusted to track a target activity level, which prevents any subset of units from monopolizing the population's activation budget.
2. *k-winner-take-all competitive gating* (Maass, 2000; Rumelhart & Zipser, 1985): at every cycle, only the k most strongly activated units in each subpopulation contribute to the output. This produces sparsity and lateral inhibition without weight-based inhibitory connections.
3. *Capacity recycling of chronically silent units*: units whose long-term activity falls below a fixed threshold are re-initialized. This is functionally equivalent to the membrane-reset and homeostatic-threshold mechanisms used in the unsupervised SNN literature (e.g., Diehl and Cook (2015) use adaptive thresholds plus lateral inhibition for the same purpose), and prevents dead units from permanently consuming budget.

The engine uses only these three mechanisms. The specific update equations remain proprietary, but they implement nothing beyond the mechanisms above.

2.2. Two-Phase Deployment Pipeline

For deployment, the engine is embedded in a two-phase pipeline (Figure 1):

Phase 1: One-time semantic calibration. The engine first runs through the calibration stream without consuming labels. Online k-means routing partitions incoming frames into K regimes. Local plasticity adapts each regime's internal representation. The input distribution is partitioned into operating regimes without supervision.

Semantic naming is a separate step. The engine has no way to determine, from the input stream alone, that one regime corresponds to inner-race fault and another to outer-race fault. An external label source is required. We add a one-time labeling step of 50–200 samples drawn from the calibration window, supplied by one of two equivalent oracles:

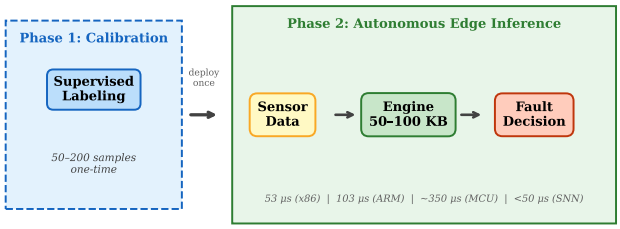

Figure 1. Two-phase deployment pipeline. Phase 1 (left): one-time semantic calibration. The engine first discovers K regimes unsupervised by streaming through the calibration window, then a 50–200 sample labeling step (human expert or LLM oracle) maps the discovered regimes to fault categories. Phase 2 (right): the engine runs autonomously on edge hardware with no further external input.

- *Human expert labeling.* A domain engineer reviews ∼5–20 representative samples per discovered regime ($K \cdot 5 \approx 50$ at $K{=}10$, scaling up to ∼200 for finer-grained partitions) and assigns a fault category to each.

- *LLM-as-oracle labeling.* A cloud-based language model labels the same samples at ∼90% accuracy on bearing-fault data; the LLM is queried once per domain.

The labeled samples populate a static regime-to-category lookup. This completes the engine state-to-fault mapping. No gradient computation runs in either oracle path. The phase executes once per domain.

Phase 2: Autonomous edge inference. The calibrated engine runs independently on edge hardware with no further external input. Feature vectors are projected into the engine's state space, expert gating selects the appropriate subpopulation, and a classification decision is emitted within a single engine cycle ($<$200 μs on ARM, $<$350 μs on MCU).

The frozen engine model occupies 50–100 KB depending on population configuration. Including the calibration mapping and feature projection matrices, the full pipeline fits within on-chip SRAM of modern microcontrollers.

3. EXPERIMENTAL SETUP

3.1. Domains and Datasets

We evaluate the engine across seven task configurations spanning four physical datasets and three task types (classification, audio anomaly detection, and a pooled multi-domain configuration). Table 1 summarizes the configurations. All experiments use identical engine parameters (the same population size, dimensionality, plasticity rates, and gating architecture), with only the input feature projection adapted to match each dataset's dimensionality.

Table 1. Seven task configurations across four physical datasets. K: number of classes. D: feature dimensionality.

Task Configuration	Dataset	K	D	Samples	Task Type
Bearing (10-class)	CWRU (Smith & Randall, 2015)	10	64	∼12K	Classification
Bearing outer race	Paderborn OR (Lessmeier et al., 2016)	12	64	∼8K	Classification
Bearing combined	Paderborn IR+OR (Lessmeier et al., 2016)	17	64	∼16K	Classification
Bearing inner race	Paderborn IR (Lessmeier et al., 2016)	5	64	∼8K	Classification
Pooled multi-domain	Paderborn + CWRU	27	64	∼20K	Classification
Audio monitoring	DCASE Fan (Koizumi et al., 2020)	2	80	∼3.6K	Anomaly det.
Activity recognition	UCI HAR (Anguita et al., 2013)	6	561	∼10K	Classification

3.2. Baseline Methods

To isolate the engine's contribution, we compare engine-augmented features against raw features processed by the same downstream classifier:

- **Classical ML:** Random Forest (100 trees), k-nearest neighbors (k=5), support vector machine (RBF kernel, $C = 1$, γ='scale'), and L2-regularized logistic regression ($C = 1$).
- **Gradient-trained:** Multi-layer perceptron (2 hidden layers, 128 units each, ReLU, Adam at lr= 10^{-3}, 200 epochs with early stopping at patience 20) and a gradient-trained MoE gating network of equivalent capacity.
- **Ablation:** k-means clustering only (no engine dynamics) and raw features without engine processing.

Hyperparameters for the classical baselines are scikit-learn defaults rather than per-task tuned, since the engine vs. raw comparison is meant to reflect typical deployment, where defaults are common. To ensure default parameters did not bias the comparison, we ran a 3-fold cross-validation grid search on a held-out 20% slice of CWRU over $n_{\text{trees}} \in \{50, 100, 200, 500\}$ for RF and $k \in \{3, 5, 7, 10, 15\}$ for kNN. RF accuracy varied within $[0.9977, 0.9985]$ (swing 0.08 pp). kNN varied within $[0.9994, 1.0000]$ (swing 0.06 pp). The raw vs. engine relative ranking did not change. We retain defaults for reproducibility.

For each configuration, the "Raw" column in Table 3 reports the best of the classical methods listed above, applied to the same raw (un-engineered) features the engine receives. The "Engine" column reports the same classifier applied to engine-transformed features. Each cell reports the best of five random seeds; cross-seed variance is low (Table 4, mean pairwise error correlation below 0.03). Table 6 compares against gradient-trained methods on the most challenging configuration.

3.3. Evaluation Protocol

All classification tasks use stratified 5-fold cross-validation; the audio anomaly-detection task (DCASE) uses a temporal train/test split reflecting realistic deployment conditions. We report accuracy for classification and AUC for anomaly detection. Table 2 reports the three-stage data pipeline per task. Stage A (unsupervised engine adaptation) consumes the calibration partition without labels: the engine discovers regimes from the input stream alone. Stage B (semantic calibration) draws 50–200 labeled samples from the same calibration partition and assigns each discovered regime a fault category, using either expert-supplied or LLM-generated labels. Stage C (live inference) operates on the held-out partition, which is never visible to Stage A or Stage B.

Table 2. Three-stage data pipeline per task. Stage A (engine adaptation): online k-means routing discovers K regimes from streaming input; *no labels consumed*. Stage B (semantic calibration): 50–200 labeled samples (human expert or LLM oracle) map discovered regimes to fault categories; samples drawn from the calibration partition only. Stage C (live inference): the calibrated pipeline runs autonomously on the held-out partition; *no labels consumed*. The Stage C partition is never visible to Stage A or Stage B.

Task	Stage A unsup.	Stage B labels	Stage C live	Split policy
CWRU (K=10)	∼9.6K	50–200	∼2.4K	5-fold
Paderborn OR (K=12)	∼6.4K	50–200	∼1.6K	5-fold
Pad. IR+OR (K=17)	∼12.8K	50–200	∼3.2K	5-fold
Paderborn IR (K=5)	∼6.4K	50–200	∼1.6K	5-fold
Pooled multi-dom. (K=27)	∼16K	50–200	∼4K	5-fold
UCI HAR (K=6)	∼8K	50–200	∼2K	5-fold
DCASE Fan	∼2.5K	50–200	∼1.1K	70/30 t.

3.4. Hardware Platforms

Edge deployment is validated across four hardware tiers:

- **x86:** AMD Ryzen workstation (reference platform).
- **ARM:** Raspberry Pi 5 (Cortex-A76 @ 2.4 GHz, 4 GB RAM, $60).
- **MCU:** STM32H7S3 (Cortex-M7 @ 600 MHz single-core, <$10).
- **Neuromorphic (projected):** Innatera T1 Pulsar (∼420 spiking neurons, 22 nm FDSOI, 400–600 μW

Table 3. Single-engine results. Each cell is the best of 5 random seeds; cross-seed variance is low (the consensus-voting analysis, Table 4, gives mean pairwise error correlation below 0.03). "Raw" = best classical method on the same unengineered features. "Engine" = engine-augmented features, same classifier. Δ = Engine $-$ Raw.

Configuration	Raw	Engine	Δ
CWRU K=10	98.1%	99.8%	+1.7
Pad. OR K=12	41.3%	90.1%	+48.8
Pad. IR+OR K=17	53.2%	90.2%	+37.0
Pad. IR K=5	85.4%	89.8%	+4.4
Pad.+CWRU K=27	68.4%	80.1%	+11.7
DCASE Fan	92.2%	92.3%	+0.1
UCI HAR	97.8%	93.2%	-4.6
Mean (7 configs)	**76.6%**	**90.8%**	**+14.2**

rated (Innatera Nanosystems, 2025)). The engine's frozen forward pass has been exactly mapped to a spiking neural network using the T1 neuron budget (MSE $< 10^{-30}$, 14% neuron utilization), with int8 quantized spike parity of 99.5%.

Latency is measured as wall-clock time for a single inference cycle, averaged over 10,000 steps after 1,000-step warm-up.

4. RESULTS

4.1. Cross-Domain Accuracy

Table 3 presents single-engine results across all seven configurations. Against a simple raw-feature baseline, the engine improves mean accuracy by +14.2 pp.

Three regimes are visible:

Weak baselines (<70%): Paderborn OR (K=12) improves by +48.8 pp, Paderborn IR+OR (K=17) by +37.0 pp, and the pooled multi-domain configuration by +11.7 pp, all measured against a simple raw-feature baseline. The raw features here are not domain-engineered; the transformation supplies separability that hand-crafted bearing features (e.g. envelope spectra) would also provide. We therefore read these gains as evidence that the engine automates feature engineering, not as superiority over feature-engineered classical methods.

Moderate baselines (70–90%): Paderborn IR (K=5) improves by +4.4 pp. Moderate gains where baseline features carry partial information.

Strong baselines (>90%): CWRU (K=10) shows +1.7 pp, DCASE Fan +0.1 pp (effectively neutral), and UCI HAR -4.6 pp. On UCI HAR, the dimensionality reduction required for engine ingestion (561D input compressed to engine state space) loses discriminative information that the raw RF exploits. The worst-case degradation across all nine configurations is -4.6 pp.

Table 4. Multi-seed consensus voting (5 seeds, majority vote) on bearing fault configurations.

Configuration	Engine	+Vote	Δ_V
CWRU K=10	99.8%	**100.0%**	+0.2
Pad. OR K=12	90.1%	$\sim$**100%**	+9.9
Pad. IR+OR K=17	90.2%	**99.9%**	+9.7
Pad. IR K=5	89.8%	$\sim$**100%**	+10.2

Table 5. Edge deployment across four hardware tiers.

Platform	Latency	Power	Mem.	Cost
x86 (Ryzen)	53 μs	n/a	50-100 KB	n/a
ARM (RPi5)[a]	103 μs	2.73 W[b]	50-100 KB	\$60
MCU (STM32H7)[c]	$\sim$350 μs	$\sim$1 W	50-100 KB	<\$10
T1 Pulsar (SNN)[d]	<50 μs	<1 mW[e]	4.6 KB	TBD

[a]Cortex-A76 @ 2.4 GHz, 4 GB. [b]System-level incl. SoC, RAM, peripherals. [c]Cortex-M7 @ 600 MHz single-core. [d]Projected from verified SNN mapping; not yet measured on-chip. [e]Innatera Pulsar rated: 400–600 μW (Innatera Nanosystems, 2025).

4.2. Multi-Seed Consensus Voting

On the four bearing fault configurations where sub-millisecond engine latency permits multiple inference passes, we apply 5-seed majority voting (Table 4). Each seed uses a different random initialization; the final prediction is the majority vote across five independently run engines.

Voting provides +9.7 to +10.2 pp improvement on the three hardest bearing tasks (K=5, 12, 17), pushing all to $\geq$99.9%. This multiplicative synergy arises because engine features produce near-independent errors across seeds (mean pairwise error correlation <0.03), enabling rapid convergence of majority vote.

4.3. Edge Deployment Latency

Table 5 presents inference latency across four hardware tiers.

The MCU result ($\sim$350 μs at 600 MHz single-core) falls within the sub-millisecond envelope on hardware costing under \$10. The engine's frozen forward pass has been exactly mapped to a spiking neural network compatible with the Innatera T1 Pulsar neuromorphic chip, requiring only 136 spiking neurons (14% of the chip's budget) with int8 spike parity of 99.5% and model size of 4.6 KB. At the T1's rated power of 400–600 μW for typical workloads (Innatera Nanosystems, 2025) (actual engine inference power not yet measured on-chip), this would represent an estimated >1000$\times$ power reduction compared to the MCU ($\sim$1 W).

4.4. Comparison With Gradient-Based Methods

Table 6 compares the engine against gradient-trained alternatives on the most challenging configuration (Paderborn OR, K=12).

Table 6. Engine vs. gradient-based methods on Paderborn OR (K=12).

Method	Accuracy	Gradients?
Raw features + RF	41.3%	No
k-means only	44.7%	No
Gradient MLP	78.2%	Yes
Gradient MoE Gate	82.6%	Yes
Engine (ours)	**90.1%**	**No**

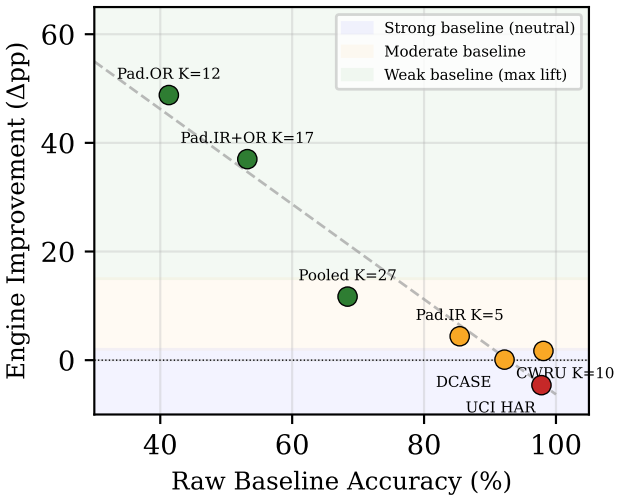

Figure 2. Conditional amplification: engine improvement (Δ) vs. raw baseline accuracy across the seven task configurations. Maximum benefit where baselines are weakest.

All methods in Table 6 receive the same raw features. This comparison therefore does not establish that the engine's advantage persists when the same classifiers are given hand-engineered (FFT/envelope) features, nor does it isolate the contribution of the engine's dynamics from that of its clustering and label-mapping pipeline.

4.5. Context: Published Deep Learning Benchmarks

For domains with established deep learning benchmarks, we note published results for comparison: on CWRU bearing data, 1D-CNN architectures achieve 97–99% accuracy for 10-class problems (Zhao et al., 2019; Lei et al., 2020); on UCI HAR, CNN/LSTM methods report 93–97% (Anguita et al., 2013). The engine's performance (99.8% on CWRU, 93.2% on UCI HAR) is competitive with these published deep learning results while requiring zero gradient computation and no GPU.

Figure 2 illustrates the conditional amplification characteristic.

Table 7. Comparison with neuromorphic, SNN, and local-learning PdM approaches. "Conf." = number of distinct task configurations validated in the cited reference.

Approach	Latency	Power	Train	Conf.
Algorithms (typical / published)				
BPTT-SNN (2019)	∼100μs	GPU	BPTT	1
STDP unsup. (2015)	∼ms	CPU	STDP	1
R-STDP (2019)	∼ms	CPU/GPU	R-STDP	1
Eq. Prop. (2017)	∼ms	GPU	EP	1
Pred. coding (2017)	∼ms	GPU	Local PC	1
BCPNN (2014)	∼ms	CPU	Hebbian	1–2
Reservoir (2019)	∼1ms	∼1W	Ridge	1–3
Hardware-grounded deployments				
Loihi (2021)	∼10μs	∼100mW	Surr. grad.	1–2
Engine on MCU	∼350μs	∼1W	**None**	**7+1**
Engine on T1[a]	<50μs	<1mW[b]	**None**	**7+1**

[a]Projected; SNN mapping verified. [b]Innatera Pulsar: 400–600 μW (Innatera Nanosystems, 2025).

5. DISCUSSION

5.1. The Conditional Amplifier Interpretation

A central observation in these results is the engine's behavior as a *conditional amplifier*: large gains on weak raw baselines (<70%: mean +32.5 pp across three configurations), a modest gain on the moderate baseline (70–90%: +4.4 pp), and near-neutral on strong baselines (>90%: worst case −4.6 pp). This pattern is consistent across the fault-classification and pooled multi-domain configurations.

We hypothesize that the engine performs a self-organizing feature transformation that exposes discriminative structure latent in the input features. When raw features already provide sufficient separability, the transformation adds minimal information. When raw features are ambiguous, the transformation exposes separability that the same classifier does not capture on un-engineered features. We do not isolate this effect to the engine's dissipative dynamics specifically; a clustering and label-mapping transformation on the same features is a plausible contributor and is not separately ablated here.

5.2. Comparison With Neuromorphic and Local-Learning Approaches

Table 7 positions the engine relative to two adjacent families. The first is surrogate-gradient and BPTT-trained SNNs typical of neuromorphic deployment. The second is local-learning models that share our zero-gradient approach: STDP, R-STDP, Equilibrium Propagation, predictive coding, BCPNN, and reservoir computing.

Cross-domain evaluation. The local-learning models in Table 7 are mostly evaluated on a single task family. Diehl and Cook's STDP network (2015) and Mozafari et al.'s R-STDP variant (2019) are MNIST-only. Scellier and Bengio's Equilibrium Propagation (2017) is also demonstrated on MNIST. Whittington and Bogacz's predictive coding network (2017) is evaluated on small image datasets. BCPNN-

based PdM applications (Tully et al., 2014) cover one or two bearing or machine-fault datasets. Whether a single zero-gradient configuration applies across heterogeneous PdM domains is therefore unsettled in the cited works. Our 7+1 task evaluation under one unmodified configuration covers bearing diagnosis, audio monitoring, activity recognition, a pooled multi-domain configuration, and one industrial deployment. The engine is evaluated on each domain separately; the pooled multi-domain configuration is in-distribution cross-validation over combined datasets, not train-on-one-domain/test-on-another transfer.

Training-pipeline requirements. The cited local-learning methods place specific structural demands on the training pipeline. STDP and R-STDP need spike-encoded inputs and clean episode separation. Equilibrium Propagation needs a multi-phase oscillation between free and clamped phases at every step. Predictive coding needs iterative settling to fixed points. Our engine takes dense feature vectors as input. It runs strictly forward in a single cycle. It adapts without any phase separation. The same forward pass serves both the one-time calibration and lifelong inference.

Power and latency context. On the projected T1 Pulsar deployment, the engine reaches sub-50 μs latency at the chip's rated 400–600 μW (Innatera Nanosystems, 2025) (on-chip engine inference power not yet measured), an estimated $>1000\times$ power reduction versus the MCU baseline.

5.3. Industrial Case Study: Tunnel SHM

To validate beyond public benchmarks, we deployed the engine on a real tunnel construction monitoring project in China (55 monitoring points, two bores, 182 days of daily deformation measurements; client details withheld for confidentiality).

The engine was configured with K=3 expert groups to discover deformation regimes:

- **Regime alignment**: The engine's unsupervised regimes aligned with the project's operator-defined deformation categories (stable, slow convergence, accelerated convergence). Because those categories are themselves defined by deformation-rate thresholds, this alignment is a consistency check on the deployment pipeline rather than independent evidence of detection capability; we therefore do not report it as an accuracy benchmark.
- **No dimensionality expansion**: The raw 8-dimensional deformation features were used without preprocessing.
- **Deployment**: Sub-100 KB engine on ARM (Cortex-A76), suitable for a construction-site office with no cloud dependency after calibration.

5.4. Limitations

Several limitations should be noted. First, the engine cannot improve already-strong baselines; on UCI HAR (raw 97.8%), the engine shows -4.6 pp degradation due to information loss in the dimensionality reduction step required for engine ingestion.

Second, unsupervised regime discovery is distinct from semantic naming. The engine partitions the input distribution into K regimes on its own. Assigning each regime a human-readable label ("inner-race fault," "outer-race fault," "healthy operation") still requires an external source. We bridge this gap with a 50–200 sample labeling step. The labels are supplied either by a human expert or by an LLM oracle. The LLM path is economical at current commodity pricing (under $1 per domain calibration in our setup). Operator labeling remains a viable substitute when cloud LLM access is unavailable.

Third, we initially included anomaly-detection results on the SMAP (Hundman et al., 2018) and Z24 (Maeck & De Roeck, 2003) benchmarks and have withdrawn both after a controlled re-analysis. Our own ablation showed the dimensionality-expansion step contributes only ± 0.02 AUC, and the reported gains are not attributable to the engine's dynamics: they arise from the clustering and label-mapping pipeline. The Z24 result is additionally confounded by environmental temperature variation, a known characteristic of the Z24 dataset, where the reported separation is reproducible from the temperature distribution alone and does not reflect damage detection. We therefore report neither anomaly-detection benchmark in this version; this does not affect the classification results or the deployment and footprint claims.

Fourth, the T1 Pulsar neuromorphic results are projections from verified SNN mapping, not on-chip measurements. Hardware validation is ongoing.

Fifth, the present evaluation focuses on classification and anomaly detection. Extension to remaining useful life (RUL) prediction remains ongoing work (see the companion PHME 2026 Data Challenge submission).

6. CONCLUSION

We have presented a zero-gradient neural dynamics engine: a sub-100 KB model that achieves cross-domain predictive maintenance on commercial hardware without any gradient computation, backpropagation, or global loss function. Validated across seven task configurations on four physical datasets plus one real-world tunnel monitoring deployment, the engine acts as a conditional amplifier against a simple raw-feature baseline: a mean +14.2 pp improvement, concentrated where raw features are not pre-engineered and neutral-to-negative where they are already strong. We do not claim superiority over classical methods that use domain-specific

feature engineering.

End-to-end inference of 53 μs (x86), 103 μs (ARM), and $\sim$350 μs (MCU 600 MHz) places the engine within the operating envelope of dedicated neuromorphic hardware on COTS processors. Verified SNN mapping to the Innatera T1 Pulsar (14% neuron utilization, 99.5% spike parity) suggests a pathway toward sub-milliwatt always-on PdM monitoring.

Future work: (1) shrinking the semantic-calibration budget further via active learning that requests labels only on routing-drift triggers, (2) on-chip neuromorphic hardware validation, and (3) extending the conditional amplifier theory to predict which domains will benefit most.

ACKNOWLEDGMENT

The CWRU Bearing Data Center, Paderborn University, NASA Prognostics Center of Excellence, the DCASE community, and the PHM Society (PHME 2026 Data Challenge) are acknowledged for making datasets publicly available.

REFERENCES

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *Proc. European Symposium on ANN*, 437–442.

Bi, G.-Q. & Poo, M.-M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24), 10464–10472.

Bouhadjar, Y., Wouters, D. J., Diesmann, M., & Tetzlaff, T. (2022). Sequence learning, prediction, and replay in networks of spiking neurons. *PLoS Computational Biology*, 18(6), e1010233.

Davies, M., Wild, A., Orber, G., et al. (2021). Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5), 911–934.

Diehl, P. U. & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9, 99.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Soderstrom, T. (2018). Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In *Proc. 24th ACM SIGKDD*, 387–395.

Indiveri, G., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5, 73.

Innatera Nanosystems (2025). Pulsar: The world's first mass-market neuromorphic microcontroller for the sensor edge. Product datasheet. `https://innatera.com/pulsar`

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1), 79–87.

Khan, S. & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107, 241–265.

Koizumi, Y., et al. (2020). Description and discussion on DCASE 2020 Challenge Task 2. In *Proc. DCASE Workshop*, 1–5.

Lansner, A. (2009). Associative memory models: From the cell-assembly theory to biophysically detailed cortex simulations. *Trends in Neurosciences*, 32(3), 178–186.

Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., & Nandi, A. K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing*, 138, 106587.

Lessmeier, C., Kimotho, J. K., Zimmer, D., & Sextro, W. (2016). Condition monitoring of bearing damage in electromechanical drive systems. In *Proc. European Conference of the PHM Society*, Bilbao, Spain.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Computation*, 12(11), 2519–2535.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11), 2531–2560.

Maeck, J. & De Roeck, G. (2003). Description of Z24 benchmark. *Mechanical Systems and Signal Processing*, 17(1), 127–131.

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., & Masquelier, T. (2019). Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, 94, 87–95.

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6), 51–63.

Rumelhart, D. E. & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, 9(1), 75–112.

Scellier, B. & Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuro-*

science, 11, 24.

Schuman, C. D., Kulkarni, S. R., Parsa, M., Mitchell, J. P., Date, P., & Kay, B. (2022). Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2, 10–19.

Smith, W. A. & Randall, R. B. (2015). Rolling element bearing diagnostics using the Case Western Reserve University data. *Mechanical Systems and Signal Processing*, 64–65, 100–131.

Tanaka, G., et al. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, 115, 100–123.

Tully, P. J., Hennig, M. H., & Lansner, A. (2014). Synaptic and nonsynaptic plasticity approximating probabilistic inference. *Frontiers in Synaptic Neuroscience*, 6, 8.

Turrigiano, G. G. (2008). The self-tuning neuron: Synaptic scaling of excitatory synapses. *Cell*, 135(3), 422–435.

Turrigiano, G. G. & Nelson, S. B. (2004). Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, 5, 97–107.

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, 20(3), 391–403.

Whittington, J. C. R. & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, 29(5), 1229–1262.

Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., & Le, N. (2022). Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7), 863.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115, 213–237.

BIOGRAPHIES

Jianwei Lou is the founder of RailMind Systems, based in Neuss, Germany. His research focuses on bio-inspired computing architectures for industrial applications, with emphasis on zero-gradient neural dynamics, edge deployment, and cross-domain generalization for predictive maintenance.