Spiking Neural Networks for Low-Power Vibration-Based Predictive Maintenance

Alexandru Vasilache[1,2], Sven Nitzsche[1,2], Christian Kneidl[3], Mikael Tekneyan[3], Moritz Neher[1,2,4], Juergen Becker[2]

[1] *FZI Research Center for Information Technology, Karlsruhe, Germany*

[2] *Karlsruhe Institute of Technology, Karlsruhe, Germany*

[3] *NETZSCH Pumpen und Systeme GmbH, Waldkraiburg, Germany*

[4] *Infineon Technologies, Dresden, Germany*

vasilache@fzi.de nitzsche@fzi.de

***Abstract*—Advancements in Industrial Internet of Things (IIoT) sensors enable sophisticated Predictive Maintenance (PM) with high temporal resolution. For cost-efficient solutions, vibration-based condition monitoring is especially of interest. However, analyzing high-resolution vibration data via traditional cloud approaches incurs significant energy and communication costs, hindering battery-powered edge deployments. This necessitates shifting intelligence to the sensor edge. Due to their event-driven nature, Spiking Neural Networks (SNNs) offer a promising pathway toward energy-efficient on-device processing. This paper investigates a recurrent SNN for simultaneous regression (flow, pressure, pump speed) and multi-label classification (normal, overpressure, cavitation) for an industrial progressing cavity pump (PCP) using 3-axis vibration data. Furthermore, we provide energy consumption estimates comparing the SNN approach on conventional (x86, ARM) and neuromorphic (Loihi) hardware platforms. Results demonstrate high classification accuracy (>97%) with zero False Negative Rates for critical Overpressure and Cavitation faults. Smoothed regression outputs achieve Mean Relative Percentage Errors below 1% for flow and pump speed, approaching industrial sensor standards, although pressure prediction requires further refinement. Energy estimates indicate significant power savings, with the Loihi consumption (0.0032 J/inf) being up to 3 orders of magnitude less compared to the estimated x86 CPU (11.3 J/inf) and ARM CPU (1.18 J/inf) execution. Our findings underscore the potential of SNNs for multi-task PM directly on resource-constrained edge devices, enabling scalable and energy-efficient industrial monitoring solutions.**

***Index Terms*—Spiking Neural Networks (SNNs), Predictive Maintenance (PM), Neuromorphic Computing, Industry 4.0**

I. Introduction

The integration of sensors within the Industrial internet of things (IIoT) enables advanced Predictive Maintenance (PM) strategies, crucial for optimizing industrial processes and preventing costly downtimes [1]. Condition monitoring often relies on vibration data, particularly for rotating machinery like pumps. However, acquiring high-resolution vibration data generates substantial data volumes, and traditional approaches relying on data transmission to the cloud for analysis incur significant energy costs, limiting the operational lifetime of battery-powered sensors [2], which are desirable for retrofitting machinery without complex wiring. Addressing these limitations necessitates shifting artificial intelligence (AI)-driven analysis directly to the sensor edge.

Spiking Neural Networks (SNNs) offer a compelling solution for low-power edge AI. Their event-driven computation holds the potential for significant energy savings compared to traditional Artificial Neuronal Networks (ANNs), especially on specialized neuromorphic hardware [3]. Research applying SNNs to PM tasks, surveyed in [4], often focuses on fault detection in components like bearings [2], [5]–[8] or gears [9]. Common approaches involve converting vibration data to spikes using time-frequency transformations and various encoding schemes (e.g., Population Coding, TTFS, Current Injection). While shallow feed-forward Leaky Integrate-and-Fire (LIF) networks are prevalent, more complex architectures like Long Short-Term Spiking Neural Networks (LSNNs) [10], [11] or Reservoir SNNs [12], [13] are emerging. Training is mostly supervised, using surrogate gradients or ANN-to-SNN conversion. However, a notable gap exists, as most studies do not report concrete hardware implementations or energy estimates on edge platforms [4], [5], [11].

Among other goals, this paper aims to bridge that gap by presenting energy estimates of a recurrent SNN, developed for a complex, multi-task PM problem on an industrially relevant use case. Using 3-axis vibration data, we target simultaneous regression (flow, pressure, pump speed) and multi-label condition classification (normal, overpressure, cavitation). While traditional Reccurent Neural Networks (RNNs) like GRUs could also be applied, SNNs are investigated here for their theoretical potential for extreme energy efficiency, particularly when deployed on specialized event-driven neuromorphic hardware. This study aims to quantify this potential benefit. The necessary spike encoding step is considered as a prerequisite for leveraging this event-based paradigm, with the potential for future on-chip integration. Section II presents the targeted use case and data acquisition process. In Section III, we present the complete methodology, encompassing data preprocessing, spike encoding, the recurrent SNN architecture, training procedures, hyperparameter optimization, compression techniques (pruning, quantization), and evaluation approach. Finally, Section IV shows performance evaluation and energy estimations comparing conventional (x86 CPU, ARM CPU) and neuromorphic (Loihi) platforms.

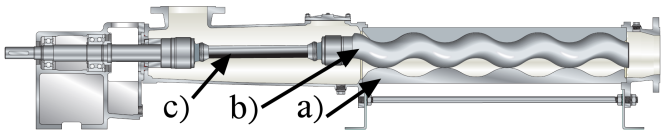

Figure 1: Progressing cavity pump (PCP); Primary Components: a) Stator, b) Rotor, c) Coupling Rod.

II. Use Case

Due to the lack of a standard benchmark dataset in the context of PM or Progressing Cavity Pumps (PCPs), a custom dataset was recorded specifically for this purpose. What follows is the description of the data acquisition process.

A. Pump System

Throughout this study, the NETZSCH NM021BY02S PCP [14] was used, which is a type of rotary positive displacement pump [15]. The pump's primary components include an elastomer stator, a metal screw rotor, and a coupling rod (see Figure 1). Its distinctive feature is the eccentric and rotating motion of the rotor within the stator, which results in pumping chambers sealed through pretension. Predominantly utilized for handling viscous media and in dosing applications, the PCP is versatile across a broad spectrum of uses.

To avoid damage in PCPs, it is necessary to predict deviations from normal operation conditions, considered impermissible, which either cause immediate damage or lead to long-term degradation of pump components. These deviations can be monitored through objective parameters, such as pump speed (rotational speed of the rotor), inlet pressure, outlet pressure, and flow rate. They include, among others, overpressure and cavitation.

The investigated NETZSCH pump is designed around a maximum discharge pressure of 12 bar. Any recordings exceeding this limit, including a small transition zone, were labeled as overpressure.

Any conditions with a suction pressure below 0.75 bar were labeled as cavitation, as the pump setup was known to consistently cavitate below this operation point. In parallel, an experienced operator adjusted the labels by listening with a stethoscope for the specific sound of cavitation.

As such, for each 3-axis vibration recording of 16384 points measured at 6664 Hz, a label was assigned based on the measured regression (flow, outlet pressure, pump speed) and classification (normal, overpressure, cavitation) values.

B. Sensors

To measure the vibration of the pump, the SIEMENS SITRANS MS200 [16] was employed. The vibration sensor was mounted at the hub position on the pump, mechanically attached using M8 threaded stud bolts. It features an integrated Bluetooth Low Energy (BLE) interface, allowing transmission of the collected sensor data to a central gateway. With a sampling rate of 6664 samples per second, the sensor offers high temporal resolution, making it ideal for detecting subtle changes in the vibration behavior of machinery.

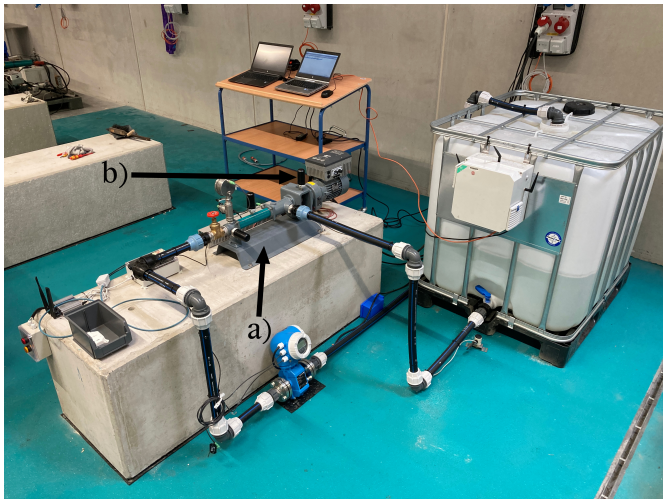

Figure 2: Pump data recording setup: a) PCP, b) Vibration Sensor at the Hub Position.

For pressure measurement, the IFM PG2453 sensor [17] was used, operating in the -1 – 25 bar range and having an error within +/- 0.2% of the measurement range.

The Endress & Hauser Promag W 5W3B25-5TT1/0 [18] was used to measure the flow, detecting values between 0 and 18 m^3/h with an error of +/- 0.5% from the measured value.

To measure pump speed, the Omron E2B-M12KN05-M1-B1 [19] was used. Its range is between 0 and 2000 min^{-1}, and offers an accuracy of +/- 0.1% from the measured value.

C. Data Generation

The dataset was mainly recorded manually, in part due to the operator's necessary expertise regarding the machine's limit states. For this purpose, the operator sets the pump's specified speed on the frequency converter. The pump's flow rate is throttled with a valve until the specified pressure is reached. Speed and pressure are kept constant for 5 minutes to perform the vibration measurement. The data generation setup can be visualized in Figure 2.

Initially, label data was recorded at a frequency of 1 Hz. However, challenges in correlating the vibration data with the labels prompted an increase in the sampling frequency to 50 Hz. The vibration data was captured at 6664 Hz for 16384 data points (approximately 2.5 s), with a 13-second transmission gap due to Bluetooth limitations. The final dataset used for this study amounts to approximately 4000 3-axis recordings of 16384 data points each or approximately 170 minutes.

III. Methods

This section details the data preprocessing pipeline, the training procedure, the SNN architecture, the hyperparameter optimization process, the compression techniques employed, and the evaluation approach.

A. Data Preprocessing

To prepare the raw sensor data for the SNN, a multi-stage preprocessing pipeline was implemented, visualized in

Figure 3. This pipeline converts the raw 3-axis (*x, y, z*) accelerometer readings, provided as sequences of `uint16` values within a defined time window (16384 samples), into a multi-channel spike train format suitable for neuromorphic processing. The pipeline consists of the following steps, executed sequentially for each time window:

1) Standardization: Initially, the raw data is normalized using the global mean and standard deviation (of each axis), calculated offline from the *entire* training dataset. Next, the mean and standard deviation are calculated again, but this time *locally* for each axis within the *current* time window in order to capture the specific characteristics of the current segment, which would otherwise be lost after normalization. These local statistics are later converted to spikes and added as extra features to the input signal. The sample is then further normalized using the computed *local* mean and standard deviation. This local normalization is crucial as it ensures the signal has a consistent dynamic range, which is a prerequisite for effective and stable spike generation by the subsequent temporal encoding algorithm (Step-Forward).

2) Spike Encoding: The second and final step involves converting the processed numerical data into binary spike trains using a hybrid encoding approach, implemented in the Spike Encoding Framework proposed in [20]. To capture the dynamic fluctuations within the normalized signal, it is encoded into spikes using the Step-Forward (SF) algorithm [21]. This temporal encoding scheme generates spikes when the signal exceeds an adaptive threshold, producing separate "+" and "-" spike trains for each axis. The thresholds used in this step are optimized offline, based on the training data distribution, using the spike converter optimization tool in [20], which finds the optimal threshold that minimizes the reconstruction error. The scalar local statistics (mean and standard deviation) are also encoded using Poisson rate encoding. First, these scalar values are normalized to a range of [0, 1] using parameters derived from the distribution of these statistics across the training set. The resulting normalized value is then rate-encoded over the duration of the time window (16384 steps), generating two additional spike trains per axis.

The output of this preprocessing pipeline is a multi-channel spike train. For each of the three input axes (*x, y, z*), four spike channels are generated: SF positive-spikes, SF negative-spikes, Poisson-coded local mean, and Poisson-coded local standard deviation. This results in $3 \times 4 = 12$ input channels for the subsequent SNN.

B. Training Procedure

The SNN was trained with PyTorch [22] using a supervised approach targeting both regression and classification tasks simultaneously.

1) Data Preparation and Augmentation: The dataset was split into training (70%), validation (15%), and testing (15%) sets. The long sequences (16384 timesteps) were segmented using a sliding window to augment the training data and reduce the sequence length. Segments of length 1024 timesteps were extracted with a step size of 1. This increased the number of

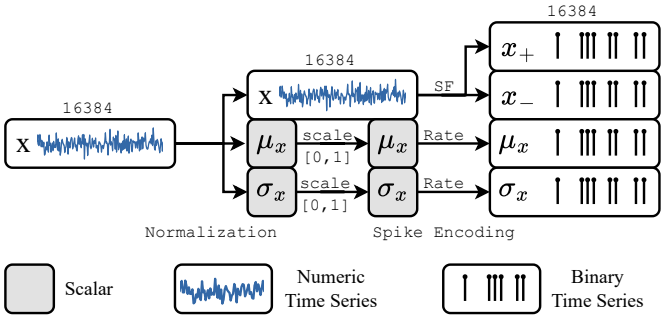

Figure 3: Preprocessing applied to each axis (*x, y, z*). After global standardization, a numeric time series undergoes local normalization, yielding a normalized time series and the window's local mean (μ_x) and standard deviation (σ_x). The normalized time series is encoded into positive (x_+) and negative (x_-) spikes via SF encoding. The local values (μ_x, σ_x) are standardized to [0,1] using global statistics and rate-encoded using a Poisson process. The pipeline outputs four binary time series per axis. Legend indicates data types.

training samples to approximately 40.6 million sequences. The validation and test sets, comprising 606 and 607 sequences, respectively, were kept in their original length of 16384 steps, as the recurrent network can handle variable input lengths during inference.

2) Batching Strategy: Given the large size of the augmented training set, a custom batching strategy was employed. Before training, the augmented training set was first shuffled. Then, 1000 mini-batches containing 9000 sequences (of length 1024) were sequentially drawn and used for training updates. This ensures that the entire training set is seen by the model approximately every five epochs.

3) Target Variable Scaling: The continuous output variables for regression (flow, pressure, pump speed) exhibit different value ranges. To address this, the RobustScaler from the scikit-learn library [23] was used. This scaler standardizes features by removing the median and scaling according to the interquartile range (IQR), making it less sensitive to outliers compared to standard scaling. The scaler parameters were determined using *only* the *training* set and applied to the entire dataset.

4) Loss Function: A composite loss function was designed to jointly optimize the regression and classification tasks. The Mean Absolute Error (L_1 Loss) was chosen for the regression task (Equation 1). It was calculated for each of the three target variables (flow, pressure, pump speed) and summed to form the total regression loss:

$$\mathcal{L}_{\text{reg}} = \sum_{i=1}^{3} |\hat{y}_i - y_i| \tag{1}$$

For the multi-label classification task (normal, overpressure, cavitation), a weighted Cross-Entropy (CE) Loss was employed (Equation 2). Class weights were calculated offline

based on the inverse frequency of each class in the training dataset to counteract class imbalance.

$$\mathcal{L}_{\text{CE}}^{\text{weighted}} = -\sum_{c=1}^{C} w_c \cdot y_c \cdot \log(\hat{y}_c) \tag{2}$$

where $C = 3$ is the number of classes, w_c are inverse-frequency weights, $y_c \in \{0, 1\}$ is the actual class, and $\hat{y}_c \in [0, 1]$ the predicted probability. The calculated CE loss was additionally rescaled, multiplying it by 10, aiming to adjust the relative contribution of the classification loss compared to the regression loss during optimization. The final loss (Equation 3) was the simple sum of the total regression loss and the rescaled, weighted classification loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{reg}} + 10 \cdot \mathcal{L}_{\text{CE}}^{\text{weighted}} \tag{3}$$

5) Optimization and Regularization: The network was trained using the Adam optimizer [24] with a learning rate of 1.224×10^{-2} and weight decay (regularization) of 1.712×10^{-6}. A `ReduceLROnPlateau` scheduler was employed to adjust the learning rate during training. This scheduler monitored the total loss on the validation set and reduced the learning rate by a factor of 0.3 if no improvement was observed for 3 consecutive epochs. Furthermore, Early Stopping was implemented, with a patience of 10 epochs.

C. Network Architecture

The model employs a recurrent SNN designed to process the 12-channel spike train input described in Section III-A for simultaneous regression and classification.

1) Neuron Model: The neuron model employed is the LIF, implemented using the snnTorch library [25]. The dynamics of each LIF neuron are governed by its membrane potential, which integrates incoming currents (weighted spikes) and decays exponentially over time towards a resting potential (zero in this implementation). When the membrane potential exceeds a predefined threshold, the neuron emits a spike, resetting its potential.

Key parameters defining the neuron dynamics were determined through hyperparameter optimization (see Section III-D) and kept fixed during training. A Membrane Potential Decay $\beta \approx 0.9$ was used, corresponding to the fraction of the membrane potential remaining after one timestep in the absence of input. A threshold value of approximately 0.96 was employed.

Training was enabled by using a surrogate gradient function, specifically the fast sigmoid function with a slope parameter of 5, to approximate the derivative of the spiking mechanism during Backpropagation Through Time (BPTT).

2) Recurrent Architecture: The network follows a recurrent structure without any bias terms, consisting of an input projection of size 12, two hidden recurrent LIF layers of size 160, and an output layer of size 6 (see Figure 4). The weights of all linear layers were initialized using a normal distribution with a mean of approximately -0.048 and a standard deviation of approximately 0.238. The network processes the input sequentially. After the entire sequence (1024 timesteps for training, 16384 for validation/testing) has been processed, the outputs from the final linear layer at each timestep are aggregated by taking their mean. A Softmax activation is applied to the 3 classification outputs. A Dropout rate of 0.07 was applied before the final spiking layer during training to mitigate overfitting.

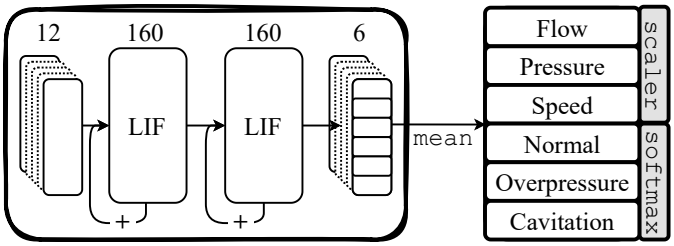

Figure 4: Overview of the recurrent SNN architecture. It processes 12 input spike channels through two recurrent hidden layers, each with 160 LIF neurons. Feed-forward and recurrent connections (indicated by the '+' summation) are present. The final linear layer produces 6 outputs, which are averaged over the sequence timesteps. Regression outputs (flow, pressure, pump speed) are inverse-scaled using the pre-fitted scaler, and classification outputs (normal, overpressure, cavitation) are generated via a Softmax function.

Table I: Optimized Hyperparameters

Hyperparameter	Value
Learning Rate	1.224×10^{-2}
Membrane Decay Factor (β)	0.900
Surrogate Gradient Slope	5
Neuron Threshold	0.959
Weight Init. Mean	-0.048
Weight Init. Std Dev	0.238
Training Window Length	1024
Output Scaler	`robust`

D. Hyperparameter Optimization

To identify the best hyperparameters for the SNN model and training process, we employed the Optuna framework [26] to minimize the total loss on the validation set evaluated after training. The optimization utilized a `MedianPruner` to terminate unpromising trials early, based on intermediate validation loss values compared to the median of intermediate results of previous trials at the same step.

To minimize training time for the hyperparameter optimization, no sliding window augmentation was applied to the training set, resulting in only 2829 samples. A total of 1000 optimization trials were conducted, searching learning rate, LIF neuron parameters, surrogate gradient slope, weight initialization parameters, the sequence length used during training, and the type of output scaler. An analysis of importance suggested that weight initialization parameters, surrogate gradient slope, and output scaling choice had the highest impact on performance. The optimization process converged to the hyperparameters in Table I.

Table II: Energy per Operation from Literature (J).

Device	Synop Energy	Neuron Energy	Source
CPU x86 (i7-4960X)	8.60×10^{-9}	8.60×10^{-9}	[27]
CPU ARM (Cortex-A5)	9.00×10^{-10}	9.00×10^{-10}	[27]
Loihi	2.71×10^{-11}	8.10×10^{-11}	[28]

E. Compression Methods

To reduce the memory footprint and computational requirements for deployment on resource-constrained hardware, pruning and quantization were applied to the trained SNN.

1) Pruning: After training, magnitude-based weight pruning was employed by iteratively removing the smallest absolute weights in increments of 5%, with a validation loss tolerance threshold of 0.1, resulting in a weight sparsity of approximately 25%.

2) Quantization: Following pruning, fixed-point quantization was applied to further reduce the model size and prepare it for hardware deployment. Weights were quantized to a signed 18-bit format, with 3 bits allocated for the integer part, based on the maximum absolute weight observed after training. While specific neuromorphic platforms have more stringent constraints such as 8-bit weights, the choice of 18-bit quantization allows for a general initial analysis. The weight regularization employed during training (Section III-B) ensured minimal bits were allocated for the integer part by penalizing high absolute weights. Thresholds and Decay Factors (β) were also quantized to an unsigned 16-bit format, with all 16 bits for the fractional part. This quantization scheme was applied to the pruned model parameters before evaluating hardware performance metrics. The resulting model has approximately $80,000$ parameters, a memory footprint of $175\,\text{KB}$, and a weight sparsity of 32.82% (increased due to quantization).

F. Energy Evaluation

To estimate the energy usage of our model on various platforms, we implemented a methodology inspired by the Nengo framework [29], estimating energy per inference based on synaptic operations and neuron updates:

$$E = N_{\text{synops}} \times E_{\text{synop}} + N_{\text{neuron updates}} \times E_{\text{neuron}} \tag{4}$$

where E_{synop} is the energy per synaptic operation and E_{neuron} is the energy per neuron update, specific to the target hardware, summarized in Table II, with values taken from established benchmarks in the literature [27], [28]. The process involves a forward pass of the trained model on the test data (606 samples with 16384 timesteps each), with values averaged across the number of samples. The number of neuron updates ($N_{\text{neuron updates}}$) is the number of neurons multiplied by the number of timesteps per inference, which assumes that every neuron is updated at every step. For the synaptic operations, the calculation differs between spiking and non-spiking hardware:

1) Spiking: Because the computation is event-based, the number of synaptic operations is calculated by summing, over all timesteps t, layers l, and neurons n (in layer l) the product of the neuron's spike state $s_{t,l,n}$ (1 if spiking, 0 otherwise) and its number of non-zero outgoing connections $c_{l,n}$. This assumes the spiking platform can leverage connection sparsity:

$$N_{\text{synops, spiking}} = \sum_{t,l,n} s_{t,l,n} \times \mathrm{c}_{l,n} \tag{5}$$

where t is timesteps, l layers, and n neurons in layer l.

2) Non-spiking: It is assumed that all synaptic connections are computed every timestep. Total synops are the sum of all connections multiplied by timesteps per inference.

This methodology has several limitations, as data transfer costs to/from the device are not included, and precision (bit-width after quantization) is not considered. Furthermore, we underestimate the energy cost for the non-spiking devices (x86, ARM) since we assume that each synop and neuron update is one Multiply-Accumulate Operation (MAC), even though in practice, a neuron update might need more, depending on the implementation. Despite this, it provides a comparative estimate of the energy costs across different hardware platforms. Energy estimates for the evaluated model are presented in Section IV and summarized in Table V.

IV. Results

The performance of the trained and compressed SNN model was evaluated on the test set (15% of original recordings, 607 samples). Results for both regression and classification tasks, along with energy estimations, are presented.

A. Regression Performance

For context, we compare our model's performance to typical industry accuracy specifications for dedicated sensors (not ML-based systems). The Full Scale Error (FSE) and Measured Value Error (MVE) represent standard benchmarks for standalone flow, pressure, and speed sensors [30]–[34].

Figure 5 illustrates the model's regression performance for flow (m^3/h), pressure (bar), and pump speed (min^{-1}) prediction. The plots show the target values, the raw SNN output, and the output smoothed using a moving median filter (window size 10).

To quantify performance, we utilize Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Mean Relative Percentage Error (MRPE). MAPE is defined as:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{6}$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and n is the number of samples. MRPE, also interpreted as range-normalized error, is defined as:

$$\text{MRPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{\max(y) - \min(y)} \right| \tag{7}$$

where $\max(y)$ and $\min(y)$ represent the maximum and minimum values of the target variable in the test set.

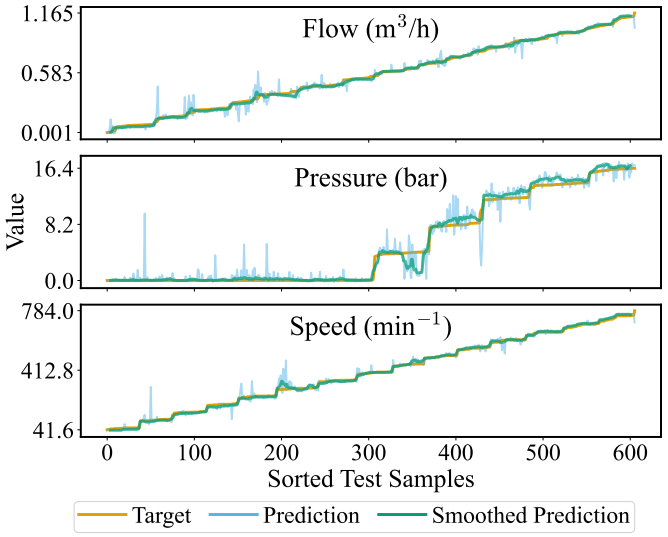

Figure 5: Regression performance on the sorted test set. Targets (orange), raw predictions (light blue), and smoothed predictions (moving median filter with a window size of 10, green) are shown.

Table III: Model Regression Performance on Test Set.

Metric	Flow (m^3/h)	Pressure (bar)	Speed (min^{-1})
MAE Raw	0.021	0.63	10.50
MAE Smooth	0.011	0.40	6.90
MAPE Raw (%)	15.37	9237.21	4.69
MAPE Smooth (%)	4.97	2447.47	3.12
Industry MVE (%)	–	–	0.1 - 2.0 [31]–[33]
MRPE Raw (%)	1.78	3.87	1.41
MRPE Smooth (%)	0.93	2.46	0.93
Industry FSE (%)	0.1 - 5.0 [30]	0.15 - 1.0 [34]	–

Table III summarizes the quantitative regression results. Smoothing improves all metrics. Comparing smoothed performance to benchmarks: flow achieves a MRPE of 0.93%, within typical industry FSE values [30]; the pump speed's MAPE of 3.12% exceeds typical industry MVE [31]–[33] by 1.12%; and pressure's MRPE of 2.46% also exceeds its typical FSE range [34] by 1.46%. The high MAPE values for pressure are mainly influenced by target values near zero.

B. Classification Performance

Figure 6 shows the classification performance for normal, overpressure, and cavitation conditions. Table IV details the classification metrics. The overall accuracy is high (>97%). Critically, the False Negative Rate (FNR) for overpressure and cavitation is 0.0%, as the model correctly identified all of these faults in the test set. The False Positive Rate (FPR) for overpressure is 3.9%, meaning some normal samples were incorrectly flagged as overpressure.

C. Energy Evaluation

Based on the methodology described in Section III-F, energy consumption estimates are presented in Table V. The

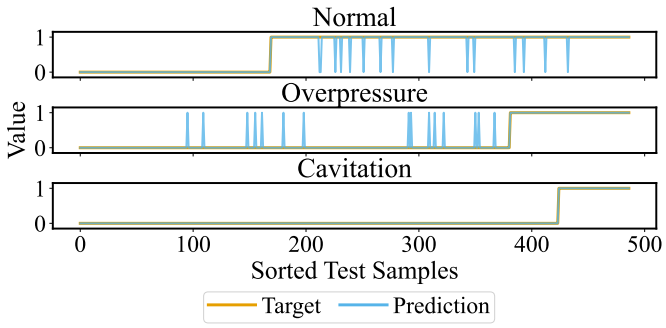

Figure 6: Classification performance on the sorted test set. Targets (orange) and predictions (light blue) are shown.

Table IV: Model Classification Performance on Test Set.

Metric	Normal	Overpressure	Cavitation	Overall
Accuracy	0.9692	0.9692	1.0000	0.9795
F1 Score	0.9758	0.9339	1.0000	0.9699
AUC	0.9891	0.9904	1.0000	0.9932
FPR	0.0000	0.0394	0.0000	0.0131
FNR	0.0472	0.0000	0.0000	0.0157

results compare estimates for conventional platforms (x86, ARM) and a neuromorphic platform (Loihi).

Loihi exhibits the lowest estimated energy consumption per inference (3.16×10^{-3} J), significantly lower than the x86 CPU (1.13×10^{1} J) and the ARM CPU (1.18 J) estimates. It is worth noting that for Loihi most of the energy consumption (86.3%) is due to the synaptic operations energy, which is dependent on the number of spikes within the network (Equation 5).

During inference on the test set (606 sequences of 16,384 timesteps), the network generated on average $530{,}317 \pm 73{,}104$ spikes per sample (Table VI). This corresponds to approximately 32.4 ± 4.5 spikes per timestep across all neurons, or a per-neuron spiking probability of 0.0975 ± 0.0134. Thus, each LIF neuron spikes in $9.75\% \pm 1.34\%$ of timesteps on average, yielding an activation sparsity of $90.25\% \pm 1.34\%$. This sparsity was achieved without explicit spike regularization in the loss function, relying on the inherent dynamics of the LIF neuron. It is also evident that by reducing this number, or the number of non-zero-weights, the energy consumption would also be reduced, which could be investigated in future work.

Given that each inference processes 2.46 seconds of data, the estimated average power consumption for the Loihi platform is approximately 1.28 mW (3.16 mJ / 2.46 s), compared to 4.59 W for the x86 CPU and 0.48W for the ARM CPU.

Table V: Estimated Energy per Inference (mJ) (mean ± std).

Device	Energy (mJ) Total	Synaptic	Neuron
x86	11273.04	11227.10	45.94
ARM	1179.74	1174.93	4.81
Loihi	$\mathbf{3.16 \pm 0.33}$	$\mathbf{2.73 \pm 0.33}$	**0.43**

Table VI: Spike generation by network layer per Inference (mean $\pm$ std over 606 test samples).

Layer	Neurons	Spikes	Spike Rate
input	12	59121 ± 6825	0.3007 ± 0.0347
lif0	160	195677 ± 33864	0.0746 ± 0.0129
lif1	160	275519 ± 66654	0.1051 ± 0.0254
TOTAL	332	530317 ± 73104	0.0975 ± 0.0134

V. Conclusion and Outlook

This paper demonstrated the application of a recurrent SNN for low-power, vibration-based Predictive Maintenance (PM) of a Progressing Cavity Pump (PCP). The model successfully performed simultaneous regression of key operating parameters (flow, pressure, pump speed) and classification of pump conditions (normal, overpressure, cavitation).

Key findings include high classification accuracy, notably achieving a zero False Negative Rate (FNR) for overpressure and cavitation faults. Regression performance achieved or approached typical industrial sensor standards, supporting the possibility of replacing multiple costly industrial sensors (flow, pressure, pump speed) with a single device that integrates vibration monitoring and local processing with SNNs.

Energy evaluation estimates indicated up to 3 orders of magnitude less energy per inference for neuromorphic hardware compared to conventional x86/ARM platforms, highlighting the potential for considerable energy savings in battery-powered edge devices. It further supports the feasibility of migrating complex data analysis tasks from the cloud directly to the sensor, thus reducing communication overhead and extending operational lifetimes.

Future work should validate these energy and performance estimates on physical neuromorphic hardware and extend the comparison to include other common edge AI accelerators to provide a broader performance context. More aggressive pruning strategies or the integration of spiking frequency minimization within the objective loss could lead to further energy improvements on neuromorphic hardware. Further refinement of the model architecture and frequency-domain preprocessing could improve regression accuracy. Long-term deployment studies are also needed to assess the robustness and reliability of the SNN-based PM solution in operational environments.

ACKNOWLEDGMENTS

This research is funded by the German Federal Ministry of Education and Research as part of the project "ThinKIsense", funding no. 16ME0564.

References

[1] M. . Company, "Establishing the right analytics-based maintenance strategy," McKinsey & Company, Tech. Rep., 2021. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/establishing-the-right-analytics-based-maintenance-strategy

[2] S. Nitzsche, M. Neher, S. von Dosky, and J. Becker, "Ultra-low power machinery fault detection using deep neural networks," in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, ser. Communications in Computer and Information Science. Springer International Publishing, 2021, pp. 390–396.

[3] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," 2019. [Online]. Available: https://arxiv.org/abs/1812.01739

[4] A. Vasilache, S. Nitzsche, D. Floegel, T. Schuermann, S. von Dosky, T. Bierweiler, M. Mußler, F. Kälber, S. Hohmann, and J. Becker, "Low-power vibration-based predictive maintenance for industry 4.0 using neural networks: A survey," *arXiv preprint arXiv:2408.00516*, 2024.

[5] N. Dennler, G. Haessig, M. Cartiglia, and G. Indiveri, "Online detection of vibration anomalies using balanced spiking neural networks," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2021, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/9458403/

[6] L. Zuo, L. Zhang, Z.-H. Zhang, X.-L. Luo, and Y. Liu, "A spiking neural network-based approach to bearing fault diagnosis," *Journal of Manufacturing Systems*, vol. 61, pp. 714–724, 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0278612520301138

[7] R. Li and J. Yuan, "Research on fault diagnosis based on spiking neural networks in deep space environment," in *2022 3rd Asia Service Sciences and Software Engineering Conference*. ACM, 2022, pp. 165–170. [Online]. Available: https://dl.acm.org/doi/10.1145/3523181.3523205

[8] L. Zuo, F. Xu, C. Zhang, T. Xiahou, and Y. Liu, "A multi-layer spiking neural network-based approach to bearing fault diagnosis," *Reliability Engineering & System Safety*, vol. 225, p. 108561, 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0951832022002095

[9] Y. H. Ali, F. Y. H. Ahmed, A. M. Abdelrhman, S. M. Ali, A. A. Borhana, and R. Ishak Raja Hamzah, "Novel spiking neural network model for gear fault diagnosis," in *2022 2nd International Conference on Emerging Smart Technologies and Applications (eSmarTA)*. IEEE, 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9935414/

[10] L. Zanatta, F. Barchi, A. Burrello, A. Bartolini, D. Brunelli, and A. Acquaviva, "Damage detection in structural health monitoring with spiking neural networks," in *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*. IEEE, 2021, pp. 105–110. [Online]. Available: https://ieeexplore.ieee.org/document/9488476/

[11] F. Barchi, L. Zanatta, E. Parisi, A. Burrello, D. Brunelli, A. Bartolini, and A. Acquaviva, "Spiking neural network-based near-sensor computing for damage detection in structural health monitoring," *Future Internet*, vol. 13, no. 8, p. 219, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/8/219

[12] S. Dey, D. Banerjee, A. M. George, A. Mukherjee, and A. Pal, "Efficient time series classification using spiking reservoir," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9892728/

[13] V. Kholkin, O. Druzhina, V. Vatnik, M. Kulagin, T. Karimov, and D. Butusov, "Comparing reservoir artificial and spiking neural networks in machine fault detection tasks," *BDCC*, vol. 7, no. 2, p. 110, 2023. [Online]. Available: https://www.mdpi.com/2504-2289/7/2/110

[14] "NEMO® BY Blockpumpe in Industrieausführung." [Online]. Available: https://pumps-systems.netzsch.com/de/produkte-und-zubehoer/nemo-exzenterschneckenpumpen/nemo-by-blockpumpe-in-industrieausfuehrung

[15] J. Moineau, "Pompe," Patent US 1 892 217.

[16] "SITRANS SCM IQ, SITRANS CC220, SITRANS MS200." [Online]. Available: https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/7MP2210-2AB21-2AB1

[17] "PG2453 - Pressure sensor - ifm." [Online]. Available: https://www.ifm.com/de/en/product/PG2453

[18] "Promag W 300, 5W3B1H." [Online]. Available: https://www.de.endress.com/de/messgeraete-fuer-die-prozesstechnik/5W3B1H

[19] "E2B-M12KN05-M1-B1 OMI | OMRON." [Online]. Available: https://industrial.omron.de/de/products/E2B-M12KN05-M1-B1

[20] A. Vasilache, J. Scholz, V. Schilling, S. Nitzsche, F. Kaelber, J. Korsch, and J. Becker, "A pytorch-compatible spike encoding framework for energy-efficient neuromorphic applications," 2025. [Online]. Available: https://arxiv.org/abs/2504.11026

[21] N. Kasabov, N. M. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. G. Doborjeh, N. Murli, R. Hartono *et al.*, "Evolving spatio-temporal data machines based on the neucube neuromorphic framework: Design methodology and selected applications," *Neural Networks*, vol. 78, pp. 1–14, 2016.

[22] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia,

W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. [Online]. Available: https://pytorch.org/assets/pytorch2-2.pdf

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.

[26] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[27] B. Degnan, B. Marr, and J. Hasler, "Assessing trends in performance per watt for signal processing applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 58–66, 2015.

[28] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[29] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, no. 48, pp. 1–13, 2014.

[30] "What is Flow Meter Accuracy?" [Online]. Available: https://koboldusa.com/articles/common-questions/what-is-flow-meter-accuracy/

[31] Z. Xie, J. Dong, Y. Li, L. Gu, B. Song, T. Cheng, and Z. L. Wang, "Triboelectric rotational speed sensor integrated into a bearing: A solid step to industrial application," *Extreme Mechanics Letters*, vol. 34, p. 100595, 2020.

[32] L. Wang, Y. Yan, and K. Reda, "Comparison of single and double electrostatic sensors for rotational speed measurement," *Sensors and Actuators A: Physical*, vol. 266, pp. 46–55, 2017.

[33] L. Wang, Y. Yan, Y. Hu, and X. Qian, "Rotational speed measurement through electrostatic sensing and correlation signal processing," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 5, pp. 1190–1199, 2013.

[34] "MIP Series Heavy Duty Pressure Transducers | Honeywell." [Online]. Available: https://automation.honeywell.com/de/de/products/sensing-solutions/sensors/pressure-sensors/mip-series