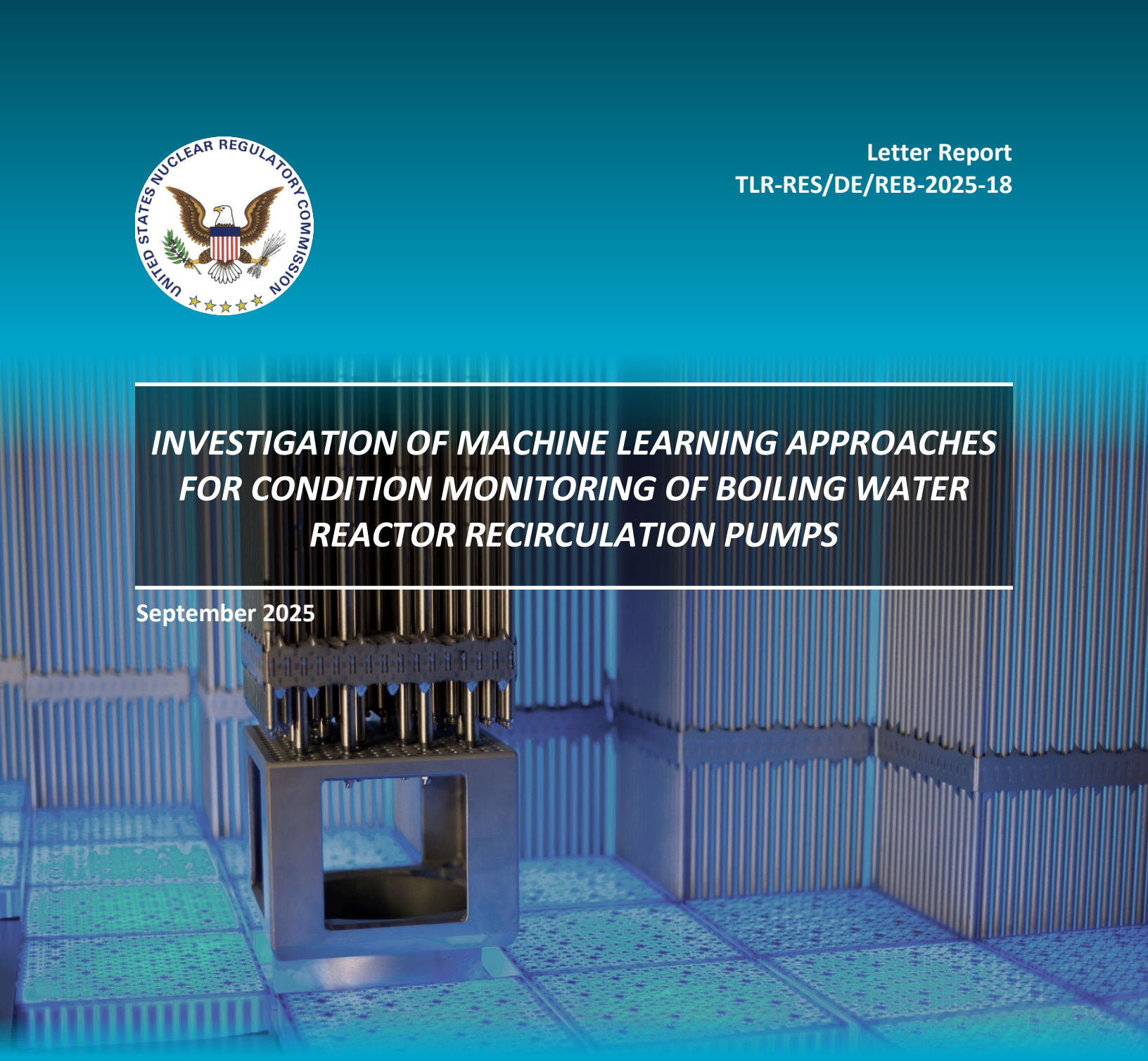

Letter Report
TLR-RES/DE/REB-2025-18

INVESTIGATION OF MACHINE LEARNING APPROACHES FOR CONDITION MONITORING OF BOILING WATER REACTOR RECIRCULATION PUMPS

September 2025

T. Villarreal, J. Matrachisia,
T. Hathaway, R. Iyengar
U.S. Nuclear Regulatory Commission

Division of Engineering
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555–0001

Prepared as part of the Research Assistance Request NRR-2023-023, "Condition Monitoring of Mechanical Systems"

This report does not contain or imply legally binding requirements. Nor does this report establish or modify any regulatory guidance or positions of the U.S. Nuclear Regulatory Commission and is not binding on the Commission.

EXECUTIVE SUMMARY

The nuclear industry has begun considering the use of artificial intelligence and machine learning (AI/ML) to support operation and maintenance activities at nuclear power plants. For example, ML can be used to support condition monitoring for inservice testing (IST) programs by training on sensor readings collected during normal operation and alerting staff if there is an anomaly in performance. Because ML can be used in numerous areas to support nuclear reactor performance, it is critical to understand the capabilities and limitations of these technologies. This technical letter report documents the development and evaluation of a machine learning (ML)-based anomaly detector and classifier, as an example of the types of AI/ML technologies that could be used for condition monitoring of nuclear systems. A hypothetical use case was developed where the anomaly detector and classifier would be used to monitor the condition of recirculation pumps at a simulated boiling water reactor (BWR) nuclear power plant. A full-scope BWR simulator was used to generate multivariate time-series data for both nominal steady-state full-power operating conditions, and six malfunction scenarios related to recirculation pumps. These included recirculation pump runaways and seal failures.

Two ML classifiers were developed, one with the purpose of detecting anomalies, given plant state data, and the other for classifying the type of anomaly present. These classifiers were trained using preprocessed simulated BWR time-series data, initialized from nominal full-power steady-state operation. The data are binned into time windows to perform time- and feature-dependent predictions.

The anomaly detector is a long short-term memory (LSTM)-autoencoder which was trained on a simulated 4-hour nominal full-power operation dataset. The goal of the anomaly detector was to identify deviations from expected full-power behavior. Results demonstrated the feasibility of LSTM-autoencoders as a ML-based method to detect anomalies. The anomaly classifier is an LSTM-Softmax neural network trained on simulated malfunctions affecting the BWR recirculation pumps. The BWR simulator was used to create labeled data by performing 60-second transients with induced malfunctions of varying magnitude, labeling the transient with the type of induced malfunction. The LSTM-Softmax classifier model was trained and optimized using various time window sizes and tested against multiple transient scenarios with varying malfunction timings and severities. Performance testing was conducted to evaluate the capabilities and limitations of the combined models, where the first model identified an anomaly, and the second model determined the malfunction type. The models performed well in identifying the onset of an anomaly and classifying the cause of that anomaly.

While the models performed as expected within the original testing scope, an additional evaluation was added that involved testing on a new malfunction, a heat exchanger tube leak. This test revealed the limitations of both models as it was determined that the selected input features were insufficient to characterize the heat exchanger tube leak malfunction, leading to reduced model performance. This pointed out the importance of understanding possible malfunctions monitored equipment can encounter and ensuring the collected data reflect the impact of possible malfunctions.

Overall, this work contributes to the understanding of AI/ML technologies and provides a foundation for assessing their potential role in future condition monitoring applications. The results underscore the importance of data acquisition, data preprocessing, and model

development as well as the capabilities and limitations of ML-based condition monitoring for application at nuclear power plants.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AI	Artificial Intelligence
ASME	American Society of Mechanical Engineers
BPV	Boiler and Pressure Vessel
BWR	Boiling Water Reactor
CE	Cross-Entropy
CDF	Cumulative Distribution Function
ISI	Inservice Inspection
IST	Inservice Testing
LSTM	Long Short-Term Memory
LWR	Light Water Reactor
ML	Machine Learning
MSE	Mean Squared Error
OM	Operation and Maintenance
RNN	Recurrent Neural Networks

1 INTRODUCTION

The growing interest in artificial intelligence and machine learning (AI/ML) applications presents significant opportunities to enhance the operation and maintenance of mechanical systems in both existing and future nuclear reactors. In light water reactors (LWRs), documents such as the American Society of Mechanical Engineers (ASME) *Boiler and Pressure Vessel Code* (BPV Code) [1] and the ASME *Operation and Maintenance of Nuclear Power Plants*, Division 1, OM Code: Section IST (OM Code) [2] as incorporated by reference in Title 10 of the *Code of Federal Regulations* (10 CFR), Section 50.55a, "Codes and standards," [3] establish requirements for the design, construction, operation, inservice inspection (ISI), and inservice testing (IST) of certain components. Additionally, ASME Code Cases [4,5] approved by the U.S. Nuclear Regulatory Commission (NRC) provide alternatives to the ASME OM Code on a case-by-case basis. These Codes and code cases may allow for new technologies to support meeting the ISI and IST requirements. For new and advanced reactors, ASME issued the BPV Code, Section XI, "Rules for Inservice Inspection of Nuclear Power Plant Components," Division 2, "Requirements for Reliability and Integrity Management (RIM) Programs for Nuclear Power Plants," [6] and ASME OM-2-2024 Code, *Component Testing Requirements at Nuclear Facilities,* [7] that could encourage ISI and IST condition monitoring applications. In this context, AI/ML could be leveraged to support current and advanced nuclear reactors meeting ISI and IST requirements.

As the nuclear industry advances, it is anticipated that AI/ML may be introduced to augment insights into the condition and performance monitoring of mechanical systems, providing enhanced prognostics and diagnostics of nuclear power plant operation and performance. Ensuring the reliability of these AI/ML-enhanced technologies is crucial, as both safety-related and non-safety-related mechanical components must be effectively monitored throughout the life of the plant. This drives the need for increased knowledge of the capabilities and limitations of AI/ML methods to understand the impact of AI/ML models applied to condition monitoring of mechanical systems.

To explore this, a use case was developed to demonstrate and evaluate a data-driven ML-based anomaly detector and classifier with the intended purpose of evaluating the condition of a recirculation pump in a simulated boiling water reactor (BWR) nuclear power plant. A full-scope BWR simulator was used to generate synthetic data. The simulator replicates a wide range of initial conditions, operational scenarios, and malfunctions. The intended use of the BWR simulator is for training purposes and performing research, but it was used as a surrogate model for a physical BWR and allowed for the collection of data to train and test machine learning models. The goal was to examine if ML models could be capable of detecting and classifying malfunctions from simulated multivariate, time-series reflecting operating nuclear power plant data.

2 SCOPE AND OBJECTIVES

This technical letter report documents the development and evaluation of ML-based methods for condition monitoring of mechanical systems, with a focus on BWR recirculation pumps. The scope of this work is limited to operational scenarios involving nominal steady-state full-power operating condition and six distinct malfunction conditions relevant to recirculation pump performance. These malfunctions include scenarios such as pump runaway and upper and lower seal failures in each of the simulated BWR recirculation pumps.

Two types of ML models were developed and evaluated:

- **Anomaly Detection Model**—An unsupervised learning approach using a long short-term memory (LSTM) autoencoder neural network to identify deviations from nominal plant behavior based on time-series data.
- **Anomaly Classification Model**—A supervised LSTM-based neural-network model with a Softmax output layer, designed to assign malfunction labels.

The primary objectives of this study are to:

1. Advance understanding of AI/ML applications.

   This work aims to improve NRC staff familiarity with the capabilities, limitations, and practical considerations associated with AI/ML-based tools for condition monitoring of mechanical systems.

2. Demonstrate the applications of AI/ML for anomaly detection and classification.

   This work presents the development, training, and evaluation of two ML models tailored to recirculation pump behavior, providing insights on the performance of ML-based condition monitoring.

3 METHODOLOGY

This methodology describes the tools and techniques used to develop anomaly detection and anomaly classification models. It is organized into three main sections: Data Acquisition, Data Preprocessing, and Model Development.

3.1 Data Acquisition

Synthetic data were generated to create training and testing datasets using a full-scope BWR training simulator. The intended use for this BWR simulator is for training purposes, but it was used in this study as a surrogate model for a physical BWR and employed for the creation of nominal full-power operation data as well as transient data from induced malfunctions.

The simulator was initialized and steady-state operational data were recorded at nominal power for 4 hours (Figure 1), where data were recorded once every second at a uniform time-step. This dataset served as the baseline dataset defining nominal full-power operation. Figure 1 illustrates the variability of the simulated full reactor power under steady-state conditions, showcasing the mean and standard deviation. The average power during the 4-hour simulated full-power operation was 98.8 ± 0.5 percent power.

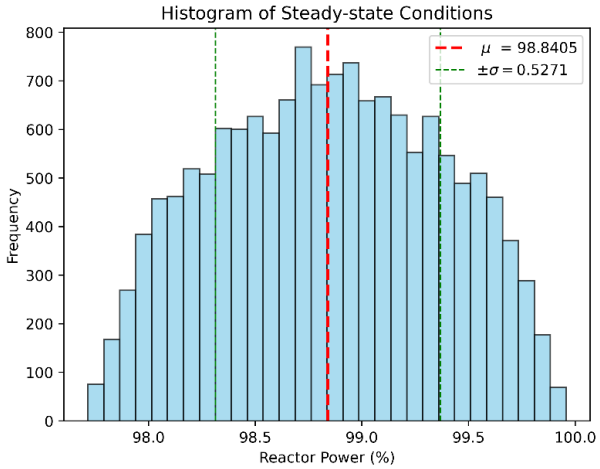

Figure 1. Histogram of the reactor power at nominal full-power steady-state conditions over 4 hours from the start of BWR simulator simulation.

Additional 1-hour runs were performed where malfunctions were introduced to the BWR recirculation pumps during operation. These runs, an example of which can be seen in Figure 2, provided insights into the response of the simulated full-power operation to induced recirculation pump malfunctions. These insights were later used to inform the selection of influential operating parameters which impacted the nominal steady-state operation, as well as transient behavior caused by induced malfunctions. For example, the reactor power trended with the recirculation pump flow rates and upper/lower seal pressures, as seen in Figure 2.

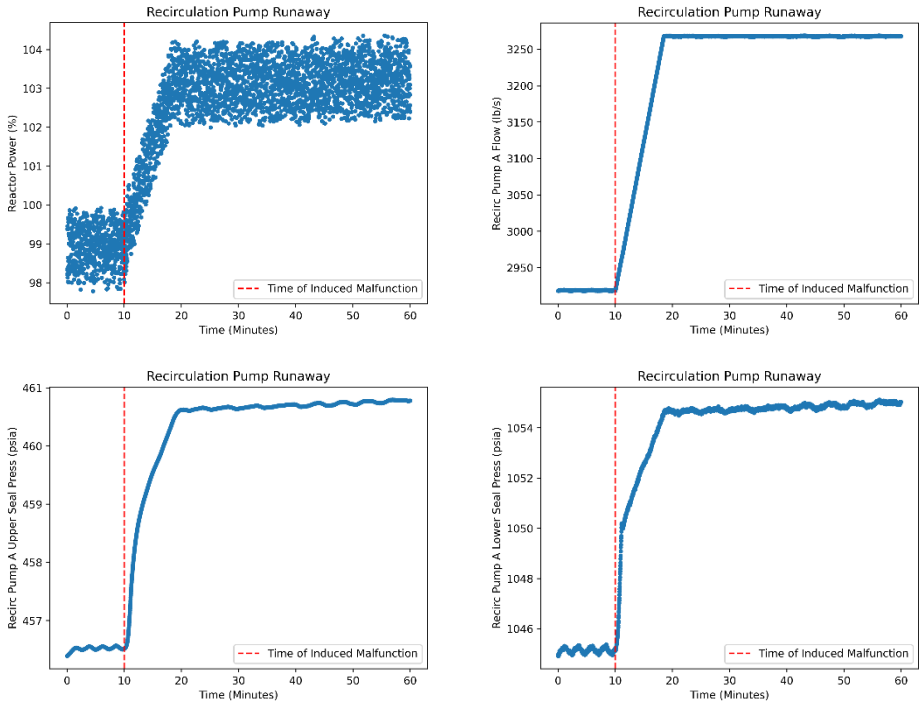

Figure 2. One hour run comparing the reactor power to the recirculation pump flow rates and upper/lower seal pressures for a scenario with an induced recirculation pump malfunction at 10 minutes.

Plant parameters were selected based on expert judgment, which was additionally informed by qualitative observations from plots demonstrated in Figure 2. Table 1 presents the parameters used to monitor the performance of the simulated BWR recirculation pumps, as well as the malfunctions which were induced to create anomalous behavior. The monitoring parameters were used for both the anomaly detector as well as the anomaly classifier.

Table 1. A list of malfunctions and monitored parameters.

Malfunctions (Label)	Monitoring Parameters
Recirculation Pump A: Runaway (RR09C) Recirculation Pump A: Upper Seal failure (RR16A) Recirculation Pump A: Lower Seal failure (RR16B) Recirculation Pump B: Runaway (RR09D) Recirculation Pump B: Upper Seal failure (RR17A) Recirculation Pump B: Lower Seal failure (RR17B)	Reactor power Recirculation Pump A: Mass flow rate Recirculation Pump A: Upper seal pressure Recirculation Pump A: Lower seal pressure Recirculation Pump B: Mass flow rate Recirculation Pump B: Upper seal pressure Recirculation Pump B: Lower seal pressure

3.1.1 Anomaly Detector Data

To generate nominal steady-state full-power operation data, the BWR simulator was initialized and operated for approximately 4 hours. Data were collected for 46 parameters during this steady-state run, consisting of 14,878 time steps taken at 1-second intervals. From this dataset, seven operating parameters, as seen in Table 1, were selected to monitor the full-power recirculation pump performance. These steady-state recordings formed the baseline data used to train the anomaly detection model to recognize deviations from normal operating conditions.

3.1.2 Classifier Data

To develop a dataset for training the anomaly classifier, the simulator started from the same initialization setpoint used to generate data for a nominal full-power run and operated for 60 seconds. A malfunction was introduced after 5 seconds of elapsed time following initialization of the simulation. The same monitoring parameters recorded for the anomaly detector data were collected for the classifier data over the 60-second duration of the transient and the parameters in Table 1. Multiple 60-second transient runs were performed for each of the malfunctions listed in Table 1. The malfunctions introduced increased from 0.1 to 50.0 percent severity, in 0.1 percent increments, resulting in 500 samples for each malfunction. Each malfunction was introduced instantaneously at 5 seconds of elapsed simulation time, that is, it took the same amount of time to introduce a 0.1 percent severity malfunction in the upper seal failure as it took to introduce a 50 percent severity upper seal failure malfunction. The transient data were then labeled with the type of induced malfunction to generate the labeled dataset needed for supervised learning. Figure 3 presents example transients used for training the classifier model.

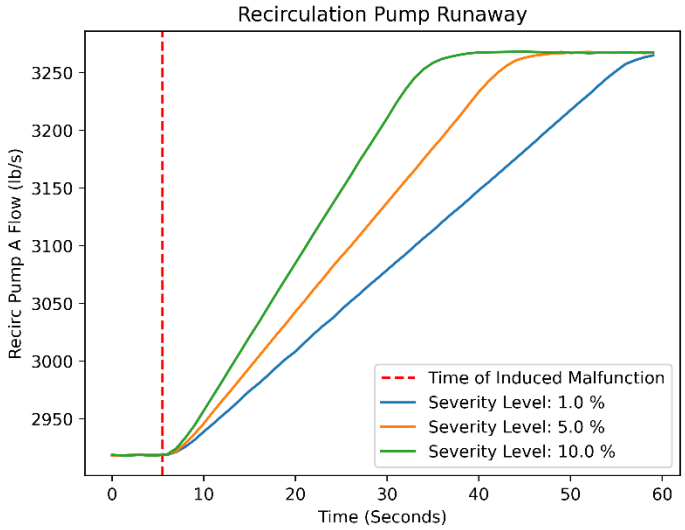

Figure 3. Plot of 60-second runs of recirculation pump mass flow rate at different severities (1.0, 5.0, 10.0 percent).

3.1.3 Transient Data to Test Condition Monitoring Algorithms

Separate testing datasets were created to test the two condition monitoring algorithms together. The datasets were 60 minutes in duration and were initialized from the same setpoint as the other datasets. Malfunctions of varying magnitude were introduced at 10, 15, 25, and 30 minutes after the initial condition. A special transient was created where an anomaly of different magnitudes was introduced, and removed, at different times throughout the 60-minute run. These longer datasets were used to evaluate the performance of both the anomaly detector and classifier. In this case, the anomaly detector was applied to the longer transient to identify when a potential anomaly occurs. Once identified, that portion of the data was provided to the anomaly classifier to attempt to classify the anomaly.

Table 2. Description of 60-minute-long anomaly detector test cases.

Test Case	Malfunction Label	Malfunction Severity (%)	Malfunction Initiation (minutes)
1	RR09C	0.1	15
2	RR09C	10	20
3	RR09C	1	10
4	RR09C	25	25
5[1]	RR17A	0.1,10,15,20	10,15,20,30
6	RR09C	50	30
7	RR09D	0.1	15
8	RR09D	10	20
9	RR09D	1	10
10	RR09D	25	25
11	RR09D	50	30
12	RR16A	0.1	10
13	RR16A	10	20
14	RR16A	1	15
15	RR16A	25	30
16	RR16B	1	10
17	RR16B	10	20
18	RR16B	1	15
19	RR16B	25	30
20	RR17A	0.1	10
21	RR17A	10	20
22	RR17A	1	15
23	RR17A	25	30
24	RR17B	0.1	10
25	RR17B	10	20
26	RR17B	1	15
27	RR17B	25	30

3.2 Data Preprocessing

3.2.1 Anomaly Detection Model

To prepare the nominal steady-state data for machine learning, the technique of data binning was applied to the time-series dataset. The dataset was segmented into overlapping time windows, where each window consisted of a fixed number of sequential time steps. Each new time window is shifted forward by one step. This binning process was applied to each input parameter, creating a structured dataset where each segment captured the short-term trends

[1] Test case 5 is a special test case where the indicated malfunctions of increasing severity were introduced and removed at the indicated times.

and variations in the system's behavior. The final processed dataset was then compiled for training and evaluation. Figure 4 illustrates the data binning process.

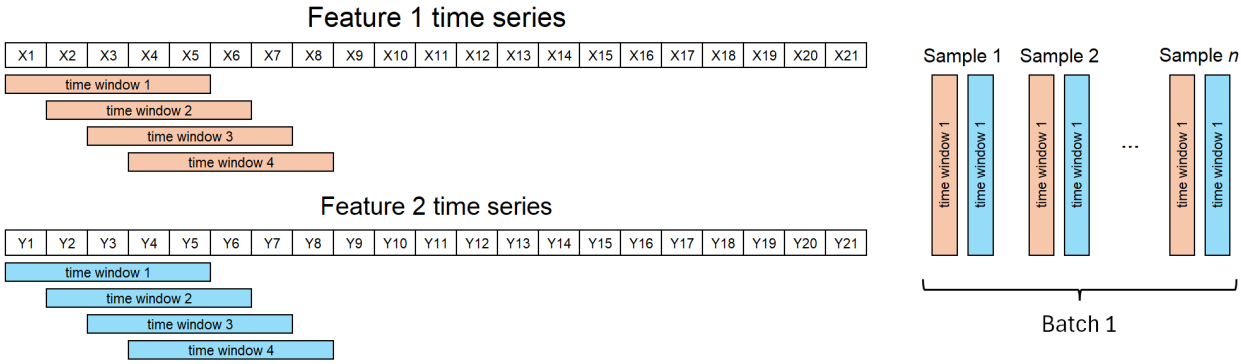

Figure 4. Illustration of data time windows used to preprocess time-series data.

The length of the time window is a hyperparameter which can be fine-tuned during the training process to optimize the short-term trends in data supplied to the neural network. In this case, the short-term trends in the nominal steady-state behavior. Multiple datasets were created using 10, 15, and 20-second time windows to examine the model's performance based on time window length. The resulting datasets provided the input and output for training the unsupervised LSTM-autoencoder model.

3.2.2 Anomaly Classification Model

Additional data preprocessing steps were required to prepare data for the anomaly classifier model. One training hyperparameter was the time window for the transient data provided to the classifier. As mentioned in Section 3.1.2, simulated operating data were recorded for 60-second transients, where a malfunction was introduced at 5 seconds of elapsed time. Rather than performing data binning, the 60-second transient was trimmed to 30 seconds; that is, time-series data beyond 30 seconds of elapsed time were ignored. A similar process was performed considering a 10-second time window. This yielded three datasets with which to train and optimize the classifier: 10 seconds of operating data, 30 seconds of operating data, or 60 seconds of operating data.

There were six malfunctions considered for the anomaly classifier, where a label for the malfunction was assigned to each transient included in the dataset. The categorical data needed to be converted into a unique vector to be used for training the classifier model. This was achieved through the process of one-hot encoding. For one-hot encoding, a list of the category labels is determined, which sets the length of the vector used to encode the labels. For each data point, or transient in this case, a one is assigned to the vector if it reflects the correct label, and a zero is assigned if it does not. An example of this process can be seen in Table 3. It should be noted that Table 3 does not necessarily reflect the one-hot encoding used for training these models but demonstrates an example of one-hot encoding.

Table 3. One-hot encoding example.

Label	One-hot encoding
Recirculation Pump A: Runaway (RR09C)	[1, 0, 0, 0, 0, 0]
Recirculation Pump A: Upper Seal failure (RR16A)	[0, 1, 0, 0, 0, 0]
Recirculation Pump A: Lower Seal failure (RR16B)	[0, 0, 1, 0, 0, 0]
Recirculation Pump B: Runaway (RR09D)	[0, 0, 0, 1, 0, 0]
Recirculation Pump B: Upper Seal failure (RR17A)	[0, 0, 0, 0, 1, 0]
Recirculation Pump B: Lower Seal failure (RR17B)	[0, 0, 0, 0, 0, 1]

Each of the three individual datasets were shuffled, creating a dataset where the order of the transients was randomized. This can assist the model from overfitting by requiring the model to examine a different transient with each training step, rather than training exclusively on one anomaly then transitioning to a different anomaly. Once the dataset was shuffled, the datasets were split into training datasets, validation datasets, and testing datasets. The training dataset is the largest and is used to train the nodes of the model. The validation dataset is used for testing the model's performance. Finally, the testing set is used to evaluate how well the model performs with unseen data. In this case, 80 percent of the data, or 2400 samples, was used for training the model. The remaining data are split evenly, meaning 10 percent of the data, or 300 samples, is used for validation and testing. The anomaly classifier is trained on six malfunctions, but it is not trained to classify normal operation. Therefore, the classifier will assign a malfunction, regardless of whether a malfunction is present. To be successful for condition monitoring, the classifier examined in this work must be paired with the anomaly detector to first identify that an anomaly is present, then the classifier analyses the event. In this instance, the anomaly detector effectively screens out nominal behavior.

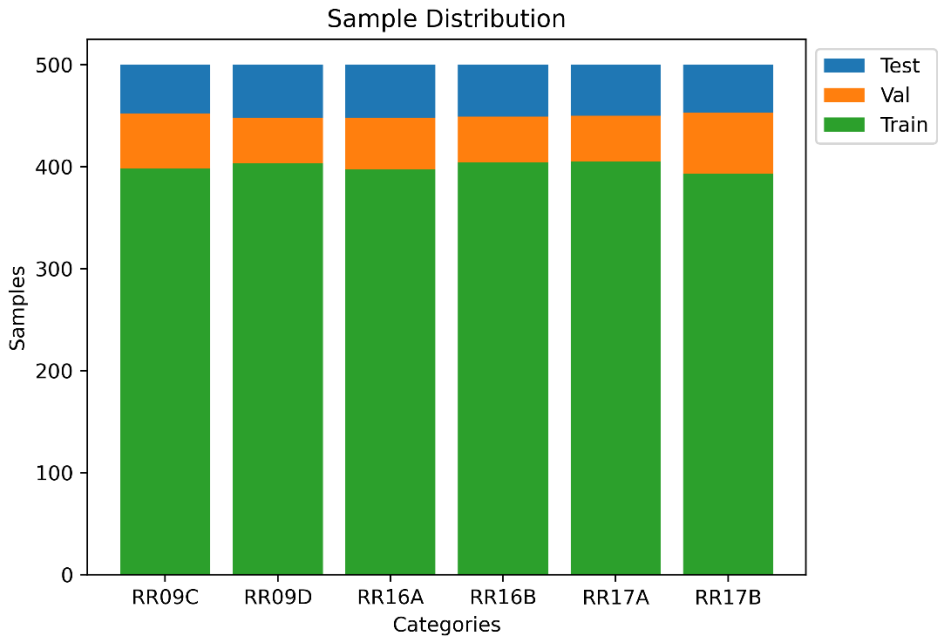

Figure 5. Random distribution of malfunction samples used to train, validate, and test the classifier with an 80 percent for training, 10 percent for validation, and 10 percent for testing split.

3.3 Model Development

The models described below were developed using Python software and trained using Tensorflow [8]. The Tensorflow package has a wide range of hyperparameters and optimizers and the process is detailed in the following subsections.

3.3.1 Anomaly Detection

An anomaly detector was developed using an LSTM-autoencoder. An autoencoder is a symmetrical neural network which consists of three components: an encoder, a decoder, and a latent space. The architecture is structured in such a way that the number of nodes present in the first layer is reduced in the next fully connected layer. The number of nodes in a layer decreases in each successive layer to a minimum. The portion of the architecture with a decreasing number of layers is called the encoder. It encodes features of the input down to a latent space, which is the output of the minimum number of nodes in the architecture. Following the minimum number of nodes, the number of nodes per successive layer begins to increase. This portion of the model is called the decoder. The model is symmetric, meaning the number of layers and the number of nodes per layer is the same on both sides of the latent space. An example of this architecture can be seen in Figure 6. The model architecture compresses input data into a reduced dimensional space (the latent space) and reconstructs the input data as output of the model. The output of the network should closely match the input data [9], therefore the model attempts to reconstruct the input data from the most important features learned about the input data stored in the latent space. The number of layers, and the number of nodes per layer are hyperparameters which can be optimized to improve the performance of the model.

An LSTM-autoencoder is an autoencoder framework that uses LSTM recurrent neural networks (RNNs) as the nodes of an autoencoder. An LSTM neural network is a type of RNN that has special gates which can feed output back to themselves and forget information from the previous state [10]. Due to their nature, LSTM neural networks work well for processing time-series data. With the combination of the LSTM neural network within the structure of an autoencoder, the model can train with a window of multivariate sequential data, such as a time series.

To train the LSTM-autoencoder, input data are provided to the model to predict an output. The predicted output is then compared to the input, and the model weights are adjusted through backpropagation to minimize the loss function, that is, to minimize the difference between the input data and the model's prediction. The minimum loss should represent when the model output closely matches the model input. The mean squared error (MSE) between the prediction and the input data are used as the loss function in this case. The MSE is based on the residuals between the input $\boldsymbol{X}_i$, and the autoencoder's predicted output $\widehat{\boldsymbol{X}}_i$:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i\right)^2 \quad (1)$$

When the model operates on new data which has similar characteristics to the training data, the MSE of the predicted outputs will be low; however, when new datasets quantitatively differ from

the training dataset, the MSE will be noticeably greater, implying there is an anomaly. Figure 6 showcases the general structure of a standard autoencoder.

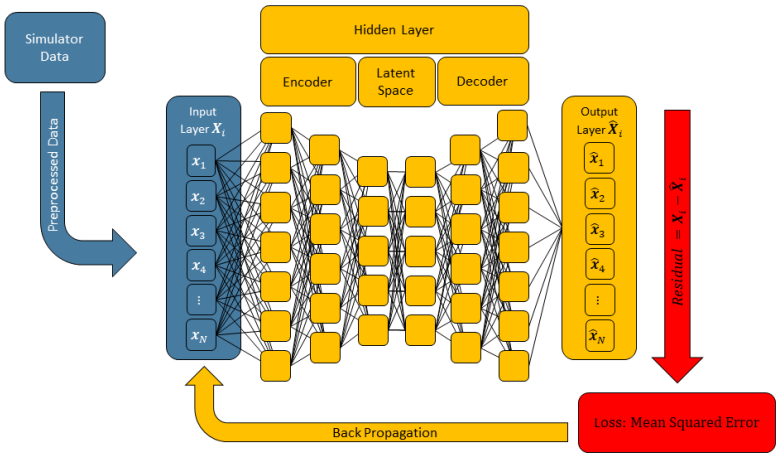

Figure 6. Illustration of a general neural-network autoencoder architecture used to develop the anomaly detector.

The architecture used in this work consists of four LSTM layers, beginning with an input layer of seven features combining with the size of the window. The network then projects to 128 LSTM nodes, encodes to 64 LSTM nodes, and then decodes back through 64 LSTM nodes to, finally, 128 LSTM nodes, before returning to an output layer with seven features and the original input window size. Note, the specifications of the architecture were selected arbitrarily as a baseline to evaluate the feasibility of the autoencoder approach; further refinement of the window sizes were based on the optimization of the model. Multiple autoencoder models were trained using datasets with time windows of 10, 15, and 20 seconds to examine the impact of window size on the performance.

3.3.2 Anomaly Classification

The anomaly classifier developed is an LSTM-Softmax neural network, which is a supervised learning algorithm. The initial hidden layers of the LSTM-Softmax classifier are layers composed of LSTM nodes. This enables the model to look for features in the time-series data. Following the LSTM layers are dense layers with a final Softmax layer, which provide the classification of the input. An example of the structure of an LSTM-Softmax classifier can be seen in Figure 8.

The LSTM-Softmax neural network takes in a window of sequential data and outputs a probability distribution ranging from zero to one using the Softmax function [8] as the activation function at the final layer [11]:

$$\hat{Y}_i \equiv \sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad (2)$$

The maximum value of the distribution is the output is taken as the label of the anomaly.

As stated previously, the labeled categories of the input data are one-hot encoded. Therefore, the category is represented as a vector and the position in that vector that represents the true

category is given a value of one while the remaining vector positions are given a value of zero. That is to say, the one-hot encoding representation of the true category can be interpreted as a probability distribution of all possible categories; the vector position represents the true category and is given a probability of 1 while all other categories have a probability of 0. As the neural network trains, the model's Softmax output is compared to the one-hot encoding of the labels using a loss function, in this case, the categorical cross-entropy (CE) function [11]. The model weights are updated through backpropagation by minimizing the loss function [10]. The CE function is dependent on the predicted probability vector $\hat{Y}_i$ and the correctly labeled vector Y_i:

$$CE = -\sum_{i=1}^{N} Y_i \log(\hat{Y}_i) \quad (3)$$

The correct label Y_i is a binary vector consisting of zeros, with a single element set to one indicating the true category. Upon training, the correct label will be compared to the model prediction. The model's prediction will be optimized so the predicted category will match the true category by altering the model's weighting through backpropagation. The maximum of the Softmax layer is taken as the true category. Figure 7 provides an illustration of the output of a Softmax classifier.

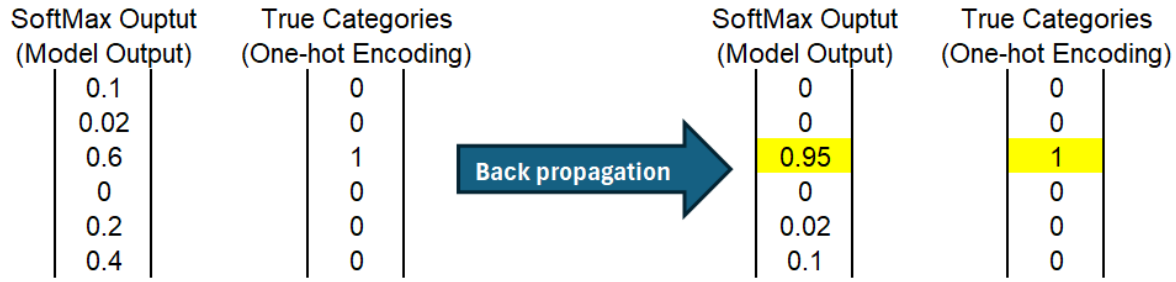

Figure 7. Illustration of training a Softmax classifier using One-hot encoding.

Figure 8 showcases the general method of a LSTM-Softmax neural network. The initial hidden layers comprise LSTM and dense neural nodes, while the Softmax layer takes in the weights from the hidden layers and generates the probability distribution. With the combination of the two, the model can train with a window of multivariate sequential data and output a vector with corresponding labels.

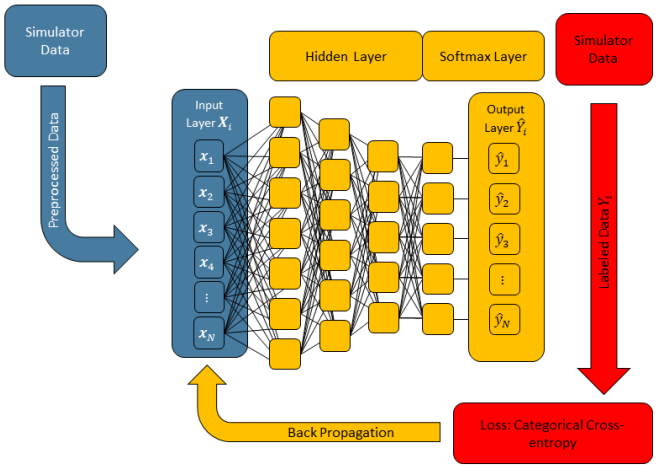

Figure 8. Illustration of a general neural-network architecture used to develop an anomaly classifier.

The classifier was trained using data with time-series of 10, 30, and 60 seconds to examine the impact of the duration of the time-series on the model's performance. Its architecture consisted of a single LSTM layer, a dense layer, and a Softmax output layer. The model's input layer processed seven features combined with the window size, followed by a projection to 64 LSTM nodes, then to 64 nodes with a rectified linear unit activation function, before producing six output labels from the Softmax layer. Like the autoencoder, the specifications of the architecture were initially chosen arbitrarily as a baseline to assess the feasibility of the Softmax approach.

4 RESULTS

4.1 Anomaly Detector Performance

Several test cases were conducted to evaluate the capabilities of the ML models. For the anomaly detector, the first set of test cases was evaluated by assessing discrepancies in the mean squared error between normal and abnormal operating condition data. The optimized neural network, trained with 251,783 parameters, 14,878 samples, 100 epochs, and the 7 input features showed suitable performance. Figure 9 presents the MSE results of the training dataset; that is, the MSE of the autoencoder when operating on simulated nominal full-power operation. It demonstrates an average MSE of 0.6181 with a standard deviation of 0.0605. This figure demonstrates the variability of the error between the model input, or measured operational data, and the predicted input when representing nominal full-power operation.

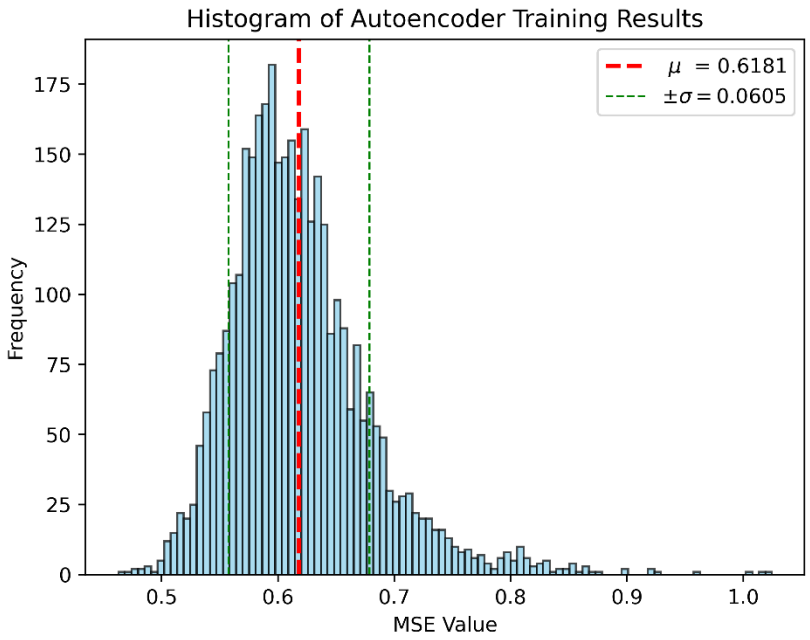

Figure 9. Distribution of mean squared error of the anomaly detector for simulated nominal reactor power.

During the testing phase of the autoencoder, the trained model was applied to 60 minutes of simulated operation, where a malfunction of a defined magnitude was introduced to the recirculation pump at specified time. The MSE results were plotted against 1 hour of steady-state operation data to demonstrate the change in MSE of the model output when the anomalous behavior is introduced, an example of which can be seen in Figure 10. Figure 10 depicts a recirculation pump runaway scenario which led to increased MSE, clearly indicating deviation from normal operation. Similar test cases with different malfunctions and time intervals were evaluated and produced similar behavior within 5 seconds of the induced malfunction.

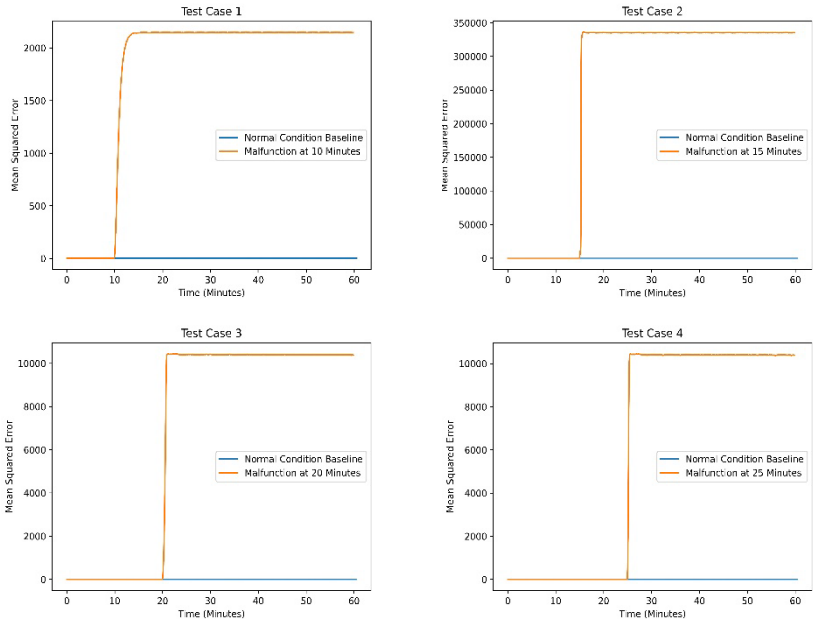

Figure 10. Autoencoder MSE for normal operating conditions vs. abnormal operating conditions at 10, 15, 20, and 25 minutes.

Test Case 5 introduced simulated malfunctions at varying intervals and severities. The simulated malfunctions were introduced then removed, except the final malfunction occurring at 30 minutes. Figure 11 demonstrates that smaller anomalies introduce smaller deviations in the models' MSE. Insights such as this demonstrate that if a model uses a threshold on the MSE for identifying anomalies, smaller anomalies may be harder to identify depending on the MSE threshold used.

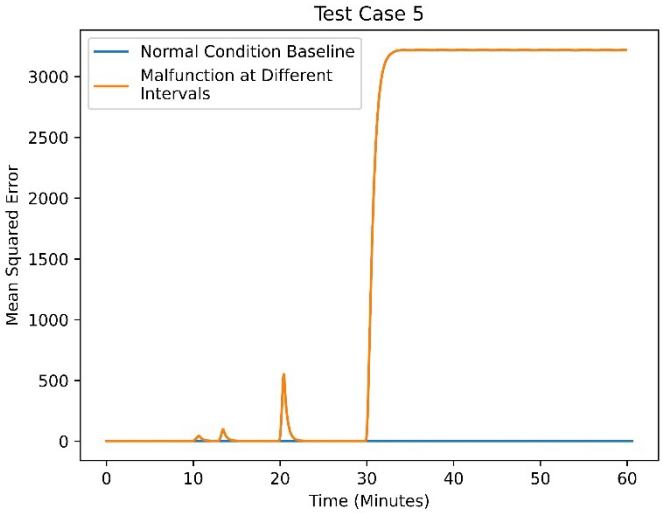

Figure 11. Autoencoder MSE for normal conditions vs. abnormal operating conditions at various times and severities.

As a note, the autoencoder was trained on a limited set of nominal full-power operating conditions and was not tested with other potential operating states. Additional testing could be needed to test the anomaly detector at other operational states, such as operating at a reduced power.

4.2 Anomaly Classifier Performance Test

The performance of the anomaly classifier was evaluated by the accuracy of the predicted label to the actual label. The classifier was trained with 2,700 training samples, 7 input features with a time window of 30 time steps, and 6 malfunction labels. There were three hidden layers, with the first layer consisting of 64 LSTM nodes, which fed into 64 dense nodes, and then lastly a Softmax layer to compute the probability distribution of the potential classes. After training the ML algorithm, the accuracy of the model reached 99.92 percent. Figure 12 presents the results for the training, validation, and testing datasets using confusion matrices. A confusion matrix demonstrates how well the classifier predicts the true label relative to misclassifications. The diagonal of the confusion matrix indicates the number of times the classifier predicted the true class, where counts off the diagonal indicate where a malfunction was misclassified. There is a slight variability in the number of test samples per class because the training data were shuffled and the testing and validation data were taken from that randomized dataset.

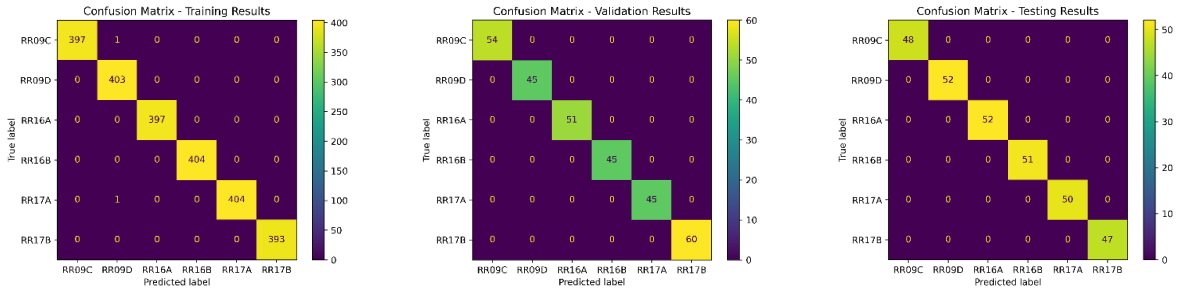

Figure 12. Comparison of the training, and validation results from the anomaly classifier.

4.3 Condition Monitoring Performance Test

A final test was conducted to evaluate the capabilities of the pre-trained models. There were 27 test simulations of different severities and induced malfunction times. These tests are listed in Table 2 of Section 3.1.3. For this test, the anomaly detector first spots the anomaly, then the classifier predicts the malfunction label. Initially, the anomaly detector used the maximum value of the MSE from the previously trained model as the threshold for flagging an anomaly. It was found that this threshold caused the anomaly detector to be too sensitive to random variability in the data. Because of this, nominal operating data could be flagged as anomalous. The threshold was increased to double the maximum of MSE to compensate and decrease sensitivity. The anomaly detector and classifier were successful in identifying and classifying the anomalous behavior for each of the test cases. On average, anomaly detectors spotted the malfunction within 7.4 seconds of the introduction of a malfunction, where the 5th percentile detection time was 2.1 seconds and the 95th percentile was 42.5 seconds.

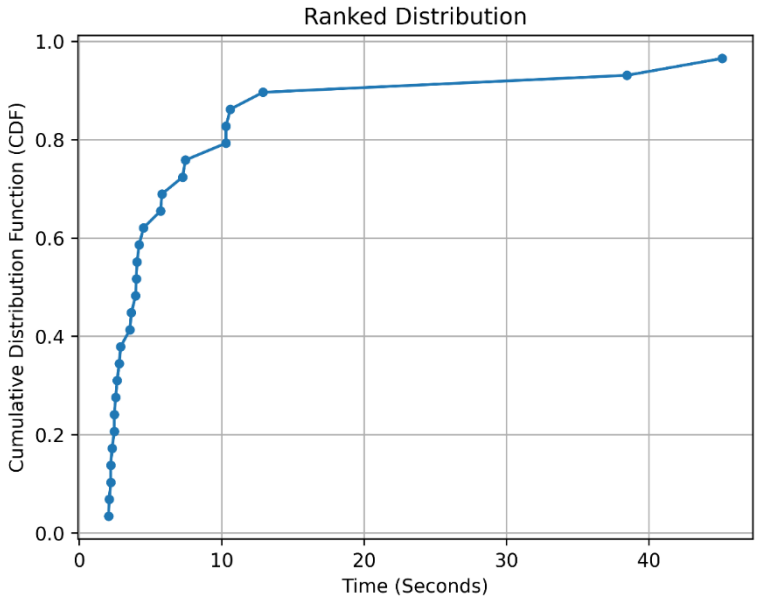

Figure 13. A plot of the cumulative distribution function (CDF) versus time error.

Figure 13 displays the CDF and is calculated [12]:

$$CDF(n) = \frac{1}{N+1} * \left(e_n + CDF(n-1)\right), \quad (4)$$

where N is the total number of samples, n is the sample number, and e is the time error. The time error at sample n is calculated by taking the absolute difference between the time at which the malfunction occurred and the time at which the anomaly was detected from the model:

$$e_n = \left|t^n_{actual} - t^n_{predicted}\right|. \quad (5)$$

The plot shows that 96.55 percent of the samples are less than 45.15 seconds and 78.57 percent of the samples are less than 10.30 seconds.

Lastly, the classifier labeled 27 out of 27 malfunctions correctly. This performance, while satisfactory, was given under ideal data generated from the BWR simulator. Neither the anomaly detector nor the anomaly classifier have been trained or tested on real-world plant data; therefore, it is unknown how the model would perform if fine-tuned with physical plant data and applied to real-world operating conditions.

5 DISCUSSION

In developing practical applications, particularly those impacting the response of safety-related or non-safety-related components, it is essential to understand the capabilities and limitations of AI/ML models. Within this methodology, both the detector and classifier were developed using a neural-network-based architecture. Although these models achieved nearly 100 percent accuracy on the training data, certain nuances and conditions must be carefully considered to

ensure their effectiveness in real-world scenarios, as highlighted during simulated condition monitoring tests.

The low training and testing errors suggest the models successfully learned patterns under controlled conditions. The ability to detect anomalies and classify conditions is integral to comprehensive condition monitoring, allowing the system to not only recognize abnormal behavior, but also to categorize the specific type of issue. However, for the models explored in this effort to perform effectively, specific conditions must be met. For time-series data, the frequency of the input data must be nearly uniform and aligned with the correct timescale, using an appropriate time window. If the model was trained with unform time steps, but tested with non-uniform time steps, the model might not recognize patterns in the data, thus allowing room for a false anomaly detection or misclassification. Also, an anomaly detector must be properly calibrated to establish an appropriate threshold for anomaly detection, enabling accurate distinction between normal and abnormal conditions. The anomaly detector needs to be able to identify anomalous behavior while not giving false positives due to variability in the normal operating condition.

An important aspect of using the anomaly detection methodology is setting the threshold to distinguish between normal variability in the operating data and anomalous behavior. As demonstrated in Section 4.3, a threshold on MSE should be large enough to capture true anomalous events, but not so low that natural variability in the MSE for nominal behavior can falsely indicate an anomaly. The threshold should also not be so high that smaller magnitude anomalous behavior is missed by the detector. Examinations of the distribution of the MSE error, as demonstrated in Figure 9, could be used to determine an appropriate threshold. The distribution of the MSE from normal variability in the nominal operational data can be compared to the distribution of the MSE following the introduction of realistic malfunctions. The detection threshold could then be set to minimize false positives caused by natural variability while maximizing the detection of anomalous behavior of a range of severities.

The anomaly detector and anomaly classifier developed in this work applied synthetic data using a BWR simulator to represent the performance of a physical system. When generating synthetic data using physics-based simulators, it is important to understand how data are generated and how that may influence the performance of the anomaly detector. In this work, synthetic data were generated to train the classifier model by instantaneously initiating a malfunction at a fixed time in a transient. It is uncertain how the classifier might respond to more-slowly-developing malfunctions. Additionally, a nuclear reactor may have numerous operating states that may be considered nominal operation though not at full power, such as the period of coastdown prior to refueling. Therefore, operational data, and synthetic data in this case, should be representative of potential operating states that might occur during normal operation and performance during malfunctions.

Another important aspect to the development of an anomaly detector is understanding the anomalous behavior, or malfunctions, that can occur in the monitored system. Without that understanding, it is possible that insufficient data are collected to monitor the underlying system. If this occurs, the anomaly detector might not be responsive to certain anomalous behaviors or predict an incorrect malfunction. In these instances, the condition monitoring system could lead reactor operators to make unneeded repairs or to miss repairs that are needed.

To explore some of these concepts, an additional study was performed where synthetic data were generated for a seventh malfunction, a heat exchanger tube leak. A new anomaly classifier model was created using the same monitoring parameters appearing in Table 1, but it attempted to classify the heat exchanger tube leak malfunction as well. It was found that the anomaly detector did not perform as well as the original model presented in Section 4.2. The reduction in performance could be attributed to monitoring parameters not capturing the response of the heat exchanger tube leak. Moreover, the training of the classifier affected the training of the original six malfunctions. This finding highlights the importance of the underlying data where the selected features may not fully capture the input necessary to distinguish between the target, in this case the heat exchanger tube leak malfunction, and another set of labels.

Along with the classifier, the heat exchanger tube leak malfunction was tested on the anomaly detector. It was found that the anomaly detector could not distinguish between nominal operation and operation with the heat exchanger tube leak. This indicates that the monitored parameters selected, which were the same as the original classifier, were not able to indicate a malfunction. This showcases another example where the importance of the scope of the data and understanding the characteristics of the data are crucial before training these models.

In this work, expert judgment was used to determine which features would be necessary to train the models, but there are potentially many other ways to select the appropriate data. Rather than expert judgment, data analysis could be performed to determine the most appropriate features that capture the behavior of malfunctions of interest. Additionally, models can be used to perform dimensionality reduction to limit the number of features in a dataset while maintaining the amount of information passed to the anomaly detector. Additionally, as more data are introduced to the model, additional preprocessing steps may be required to ensure the model is learning the important relationships between the data. For example, the data may require normalization (i.e., scaling all the data to a range of 0 to 1) to ensure no feature dominates the response of the model because the magnitude of the monitored parameter is larger. A temperature reading may be on the order of hundreds of degrees where a vibration measurement, if available, may be on the order of thousands of hertz (Hz). Normalizing the data removes the magnitude of the parameter from consideration to focus on the response of the monitored parameter to the perturbation.

Finally, it is worth noting that the data are deterministic, which is an artifact of simulated data. It is assumed that real plant data may be noisy and non-uniform. It is important that the anomaly detector and classifier learn the pattern from the features and not learn deviations from training data.

6 CONCLUSION

This use case explored the application of ML models for BWR recirculation pump condition monitoring using synthetic data generated from a full-scope BWR simulator. The scope of this use case was primarily centered around recirculation pump condition monitoring, with a particular emphasis on developing and implementing a neural-network-based anomaly detector and classifier. The anomaly detector used the LSTM-autoencoder architecture, and the anomaly classifier used the LSTM-Softmax neural network as its structure.

A methodology was developed and used to demonstrate the capabilities and limitations of ML-based models which include data acquisition, preprocessing, and model development. Performance tests of the anomaly detector and classifier revealed that machine-learning-based condition monitoring is feasible and can achieve relatively low error during training and testing under controlled conditions. Overall, the performance of both the anomaly detector and anomaly classifier combined resulted in the detection of an induced anomaly within an average of 7.4 seconds and identified the correct label for the test cases considered. While initial testing proved successful, further testing indicated challenges related to the development of models. When applied to a new malfunction, the anomaly detector and classifier could not identify the new failure. It is believed that this was due to the data used to train the model not capturing the system behavior caused by the new failure mode.

The results indicate that preprocessed data are essential for training and testing machine learning models. A model developer should understand what the possible failure modes that monitored equipment may encounter and that the monitored system features are carefully selected and properly processed to ensure the model captures the essential patterns extracted from the data. The lack of information from the data could result in poor training of the models and poor overall performance. The BWR simulator provided an ideal environment for generating data using machine learning models for testing and training when true process data were not available, but the simulator potentially provides data which may not be representative of true data in terms of data cleanliness and availability. Therefore, models trained on simulated data may require supplementation with true data to ensure the model functions well when deployed on a real system.

Future research should explore the application of models developed using simulated data to real operational data, as well as fine tuning models developed using simulated data with real operational data, to examine changes in the performance of the models. Research should continue to explore the capabilities and limitations of the models to ensure their applicability and reliability across a variety of plant operating conditions, including steady-state operation at different power levels throughout the operating cycle, operating transients, and equipment malfunctions. Additionally, methods should be explored to examine architecture to overcome challenges with deploying ML-based systems. For example, further research should consider additional data processing methods to use more of the available data, creating ensembles of models using subsets of monitored data to overcome potential data loss during operation, and the effect of noisy data on the response of the model, etc. While the neural-network-based models developed in this use case exhibit significant potential and high accuracy for condition monitoring, their accuracy relied on simulator data which might not reflect the conditions of a real-world nuclear power plant. In this case, the focus on pump condition monitoring has provided valuable insights for future research.

7 REFERENCES

1. ASME. *Boiler and Pressure Vessel Code.* American Society of Mechanical Engineers, New York, NY.
2. ASME. *Operation and Maintenance of Nuclear Power Plants,* Division 1, OM Code: Section IST. American Society of Mechanical Engineers, New York, NY.
3. Codes and Standards, 10 CFR 50.55a.
4. U.S. Nuclear Regulatory Commission. Regulatory Guide 1.192: Operation and Maintenance Code Case Acceptability, ASME OM Code. Rev. 5. Washington, DC. March 2024. ADAMS Accession Number: ML23291A006.
5. ASME. *Code Case OMN-29: Pump Condition Monitoring Program*. ASME Operation and Maintenance of Nuclear Power Plants (OM Code). American Society of Mechanical Engineers, New York, NY.
6. ASME. *Boiler and Pressure Vessel Code, Section XI, Rules for Inservice Inspection of Nuclear Power Plant Components,* Division 2, Requirements for Reliability and Integrity Management (RIM) Programs for Nuclear Reactor Facilities. American Society of Mechanical Engineers, New York, NY.
7. ASME. *Code Case OM-2-2024 Component Testing Requirements at Nuclear Facilities.* ASME Operation and Maintenance of Nuclear Power Plants (OM Code). American Society of Mechanical Engineers, New York, NY.
8. Martín Abadi et. al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
9. Srivastava, N., E. Mansimov, and R, Salakhutdinov, *Unsupervised Learning of Video Representations using LSTMs*. in Proceedings of the 32nd International Conference on Machine Learning, 2015. Lille, France.
10. Hochreiter, S. and J. Schmidhuber*, Long Short-Term Memory.* Neural Computation, 1997. pp. 1735-1780.
11. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. MIT Press. pp. 180–184. ISBN 978-0-26203561-3.
12. U.S. Nuclear Regulatory Commission. Applying Statistics. Rev. 1. NUREG-1475. Washington D.C. March 2011. ADAMS Accession Number: ML11102A076.