Early Fault Detection in Particle Accelerator Power Electronics Using Ensemble Learning

Majdi I. Radaideh*[1,2], Chris Pappas[1], Mark Wezensky[1], Pradeep Ramuhalli[1], and Sarah Cousineau[1]

[1] *Spallation Neutron Source, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, United States*

[2] *Department of Nuclear Engineering and Radiological Science, University of Michigan, Ann Arbor, MI 48109, United States*

* *Corresponding Author:* radaideh@umich.edu

Abstract

Early fault detection and fault prognosis are crucial to ensure efficient and safe operations of complex engineering systems such as the Spallation Neutron Source (SNS) and its power electronics (high voltage converter modulators). Following an advanced experimental facility setup that mimics SNS operating conditions, the authors successfully conducted 21 early fault detection experiments, where fault precursors are introduced in the system to a degree enough to cause degradation in the waveform signals, but not enough to reach a real fault. Nine different machine learning techniques based on ensemble trees, convolutional neural networks, support vector machines, and hierarchical voting ensembles are proposed to detect the fault precursors. Although all 9 models have shown a perfect and identical performance during the training and testing phase, the performance of most models has decreased in the next test phase once they got exposed to real-world data from the 21 experiments. The hierarchical voting ensemble, which features multiple layers of diverse models, maintains a distinguished performance in early detection of the fault precursors with 95% success rate (20/21 tests), followed by adaboost and extremely randomized trees with 52% and 48% success rates, respectively. The support vector machine models were the worst with only 24% success rate (5/21 tests). The study concluded that a successful implementation of machine learning in the SNS or particle accelerator power systems would require a major upgrade in the controller and the data acquisition system to facilitate streaming and handling big data for the machine learning models. In addition, this study shows that the best performing models were diverse and based on the ensemble concept to reduce the bias and hyperparameter sensitivity of individual models.

1. Introduction

Early fault detection and fault prognosis are crucial to ensure efficient and safe operations of complex engineering systems. Fault prognosis can detect and predict faults early in time before the faults cause system damage, which can be done by examining fault symptoms as early as possible (Vachtsevanos & Vachtsevanos, 2006). From now on, we use "fault precursors" to refer to fault symptoms or fault indicator signals. Advancing prognosis approaches and early fault detection is vital to the success of predictive maintenance, which unlike reactive or preventive maintenance, relies on early fault detection (W. Zhang, Yang, & Wang, 2019). Predictive maintenance has several advantages over preventive maintenance, which include (1) improving machine availability, (2) extension of the machine operation life, (3) prevention of catastrophic failures, and (4) optimizing the resources for preventive maintenance (Fernandes, Corchado, & Marreiros, 2022). All these reasons can lead to improved productivity by the machine.

Fault detection and predictive maintenance with machine learning techniques have already illustrated a promising potential. The study by (Arunthavanathan, Khan, Ahmed, & Imtiaz, 2021) used convolutional neural networks (CNN) and long short-term memory (LSTM) to forecast the system parameters and an unsupervised one-class support vector machine for fault precursor detection. The approach was assessed using the Tennessee Eastman process fault data. Similarly, a deep learning approach (a combination of sparse autoencoder and fully-connected layers) was proposed by (Luo, Wang, Liu, Li, & Peng, 2018) for early fault detection by automatically selecting the impulse responses from vibration

https://doi.org/10.36001/IJPHM.2023.v14i1.3419

signals in a time-varying system. In another study (Shao, Jiang, Wang, & Zhao, 2017), a deep autoencoder was developed for rotating machinery fault diagnosis, which consists of denoising autoencoder and contractive autoencoder layers to enhance the feature extraction ability. The authors of (L. Wang, Zhang, Long, Xu, & Liu, 2016) have developed a deep neural network framework for monitoring the health of wind turbine gearboxes based on the lubricant pressure data, which shows that the deep learning model is more capable than other classical machine learning methods (e.g. k-nearest neighbors, support vector machine). A predictive maintenance model based on LSTM and generative adversarial networks (GAN) was proposed by (Liu, Tang, Zhu, & Nie, 2021) to determine the state of the machine and the fault in advance. Ensemble or tree-based machine learning methods also showed promise in fault detection applications such as Adaboost (Kozjek, Butala, et al., 2017), isolation forests (Kolokas, Vafeiadis, Ioannidis, & Tzovaras, 2020), random forests (Syafrudin, Alfian, Fitriyani, & Rhee, 2018), random survival forests (Bukkapatnam, Afrin, Dave, & Kumara, 2019), and gradient boosting trees (Y. Zhang, Beudaert, Argandoña, Ratchev, & Munoa, 2020). Comprehensive surveys of machine learning methods for fault detection and prognosis were conducted by (L. Zhang et al., 2019; Fernandes et al., 2022).

In fault detection and classification, machine learning and neural networks have also progressed as described in this comprehensive survey (Mohd Amiruddin, Zabiri, Taqvi, & Tufa, 2020). Fault detection applications of neural networks have been demonstrated in different energy and electronics fields such as integrated energy systems (P. Wang, Poovendran, & Manokaran, 2021), large-scale power systems with LSTM (Belagoune, Bali, Bakdi, Baadji, & Atif, 2021), fusion energy devices (Tokamaks) (Mohapatra, Subudhi, & Daniel, 2020), photovoltaic systems (Hajji et al., 2021), building energy consumption (Bode, Thul, Baranski, & Müller, 2020), and similar others. Similar fault detection efforts with non-neural network methods were demonstrated by (Taqvi, Tufa, Zabiri, Maulud, & Uddin, 2020) using nonlinear autoregressive with exogenous input and by (Agasthian, Pamula, & Kumaraswamidhas, 2019) using support vector machine optimized by the Cuckoo search algorithm.

Particle accelerators, such as the spallation neutron source (SNS) (Henderson et al., 2014) and CERN, are complex engineering systems that use electromagnetic fields to propel charged particles to very high speeds and energies to use them for fundamental research applications. The interest in machine learning for control applications in particle accelerators can be seen in these studies (Nguyen, Lee, Sass, & Shoaee, 1991; Edelen et al., 2016). Uncertainty-aware anomaly detection framework of the errant beam pulses was developed by (Blokland et al., 2021) using Siamese neural networks with ResNet blocks. For a beam-based study with real measured data, the authors of (Rescic, Seviour, & Blokland, 2020) employed different machine learning binary classifiers (e.g. logistic regression, gradient boosting, random forests) to predict system failure. The results demonstrated a promising performance with failure prediction accuracy up to 92% after fine tuning the classifier hyperparameters. The fault detection effort was then improved in a subsequent study by the team for pre-emptive detection of machine trips in the SNS (Reščič, Seviour, & Blokland, 2022). Lastly, a recent study by (Felsberger et al., 2020) investigates a deep learning model based on CNN for fault prognosis in a particle accelerator system (CERN). Despite promising performance, the authors found difficulties in predicting certain failures due to the lack of data, given the authors have relied on historical data. A similar effort for using adaptive neural networks for time-varying beam control was demonstrated in (Scheinker, 2021).

In this work, we explore machine learning methods for fault detection in the power systems of the SNS, called high voltage converter modulators (HVCM). HVCMs are used in the SNS to power the klystrons that accelerate the charged particles to about 90% of the speed of light. HVCMs continue to be problematic systems for the SNS, that experience a wide range of failures from mild to catastrophic, causing reliability issues and lost beam time for the SNS. HVCMs are ranked among the top sources of downtime in the SNS (Radaideh, Pappas, Walden, et al., 2022). Compared to our previous study (Radaideh, Pappas, Walden, et al., 2022), which focused on instantaneous anomaly detection in HVCM signals using recurrent neural networks, this work extends the infrastructure and the methodology to allow early fault detection capabilities in the HVCM, which enable operator intervention and predictive maintenance to be performed in most of the fault scenarios. The previous study (Radaideh, Pappas, Walden, et al., 2022) was limited to a very short time scale that allows a graceful shutdown of the facility, due to limitations in the controller and the data acquisition system, all of which are resolved in this work. The major goal of this work is to develop and demonstrate a test facility that shows early fault detection capabilities that warn the HVCM control room of impending failures or long term degradation of components. The authors relied on three major components that highlight the main accomplishments of this work: (1) improved data acquisition system, (2) fast and continuous data streaming, and (3) machine learning fault detection models. To accomplish these goals, an advanced experimental setup that simulates SNS operating conditions and a collection of fault test scenarios are prepared and used to test our proposed fault detection methods based on machine learning.

The remaining sections of this work are organized as follows: Section 2 describes the experimental setup which involves a radio-frequency test facility established for data streaming, model development, and model testing. Section 3 highlights the methodology implemented in this work, which includes data preparation, machine learning models, and performance metrics. The results of this work are presented and discussed in section 4, followed by the conclusions of this work in section 5.

2. EXPERIMENTAL SETUP

The Spallation Neutron Source (SNS) at Oak Ridge National Laboratory (ORNL) accelerates protons to high speeds, which are used to produce neutron beams for neutron scattering and materials research (Henderson et al., 2014). The beam is accelerated in a linear accelerator consisting of a Radio Frequency Quadrupole (RFQ) section, a Drift Tube Linac (DTL) section, a Couple Cavity Linac (CCL) section, and a Superconducting Linac (SCL) section. The accelerating cavities in each of these sections are fed by high power microwave amplifiers or klystrons. The klystrons are powered by High Voltage Converter Modulators (HVCM). The HVCMs can drive as many as 10 klystrons, depending on the klystron type and which section of the linac the klystron is located. There are a total of 15 HVCMs in the SNS, driving a total of 92 klystrons, where the HVCM powering the RFQ section (3 klystrons) was the subject of the analysis in our previous effort (Radaideh, Pappas, Walden, et al., 2022). We recently shared the normal and fault data collected from the 15 HVCMs of the SNS (Radaideh, Pappas, & Cousineau, 2022), collected over 2 years with the data being sparse in time (recorded signals can be separated by hours and even days). Given their time sparsity, that data (Radaideh, Pappas, & Cousineau, 2022) or models can be used for fault classification and identification but not for early fault detection (or prognosis), which is the motivation behind this work.

Given the complexity of the SNS structure, it is very difficult to apply the proposed methods of this work directly on the SNS HVCMs since they would cause interruptions to the daily operation of the facility, especially that our experiments in this work involve fault induction tests, which can cause serious problems to the SNS. Alternatively, the radio-frequency test facility (RFTF) at the SNS provides a robust option to test new methods, components, or materials in an environment similar to the SNS environment. It is a known practice in the particle accelerator community to test new concepts in a test environment before applying it to the main accelerator, given that changing something in the SNS directly requires many approvals and careful investigation. *Therefore, this section and the rest of this work focus on the RFTF setup.*

2.1. RFTF HVCM Description

The authors have used the RFTF facility at the SNS in this work to develop and test machine learning models for early fault detection. Figure 1 shows some of the major components of the RFTF facility. Figure 1(a) shows the H-bridge switch plate of the HVCM while (b) shows the high voltage enclosure and insulation tank, which houses the HVCM assembly. Figure 1(c) shows the linear-beam vacuum tube or the klystron, which is being powered by the HVCM. Lastly, Figure 1(d) demonstrates a section of the beamline at which the particles can be accelerated (which can be powered by similar sources as the RFTF). The reader should notice here that the beamline is not directly utilized in this work and is only shown for the completeness of the facility description.

Digging deeper into the HVCM structure, the HVCM circuit is shown in Figure 2, which can be summarised by the following events that occur in the system in a frequency of 60 Hz with a high voltage pulse width of 1.3 ms:

1. An input of 13.8 kVAC three-phase line power is converted to $\pm$1300 VDC by the transformer T1 (see Figure 2) and a six-pulse controlled rectifier circuit. Capacitors C1 and C2 in Figure 2 filter this voltage and store sufficient charge to produce 1.3 ms pulses without excessive droop.
2. The DC voltage is supplied to three-phase insulated-gate bipolar transistor (IGBT) H-bridge circuits, see Figure 1(a), operating at a nominal switching frequency of 20 kHz. The IGBT switches are represented by Qa1-Qa4, Qb1-Qb4, and Qc1-Qc4 in Figure 2. The pulse transformers are used to step up the high power pulses to high voltage signals.
3. The leakage inductance of the pulse transformers (XA, XB, XC) form a resonant circuit with the resonant capacitors (Ca, Cb, Cc) in Figure 2, giving the circuit a frequency dependent gain.
4. The high voltage bipolar pulses from the resonant capacitors are recombined and rectified by the diodes Da1 to Dc2.
5. The output pulses from the diodes, which have an apparent switching frequency of 120 kHz, are filtered by C3, C4 and L1, and applied to the cathode of the klystron in Figure 1(c).

2.2. Data Streaming

The HVCM in the RFTF uses PXI-based controller to (1) control the IGBT gating timing, (2) to ensure signal values of the pulse transformers remain in a safe range, (3) to set warning and trip levels for a variety of signals, and (4) to communicate with the control room and other auxiliary systems (e.g. personnel protection systems). More importantly, the controller helps digitize and save waveforms, which are the main source of data in this study.

We performed multiple changes in the RFTF to be able to stream and save data continuously for machine learning efforts, compared to the sparse data collected in our previous study (Radaideh, Pappas, Walden, et al., 2022). First, the

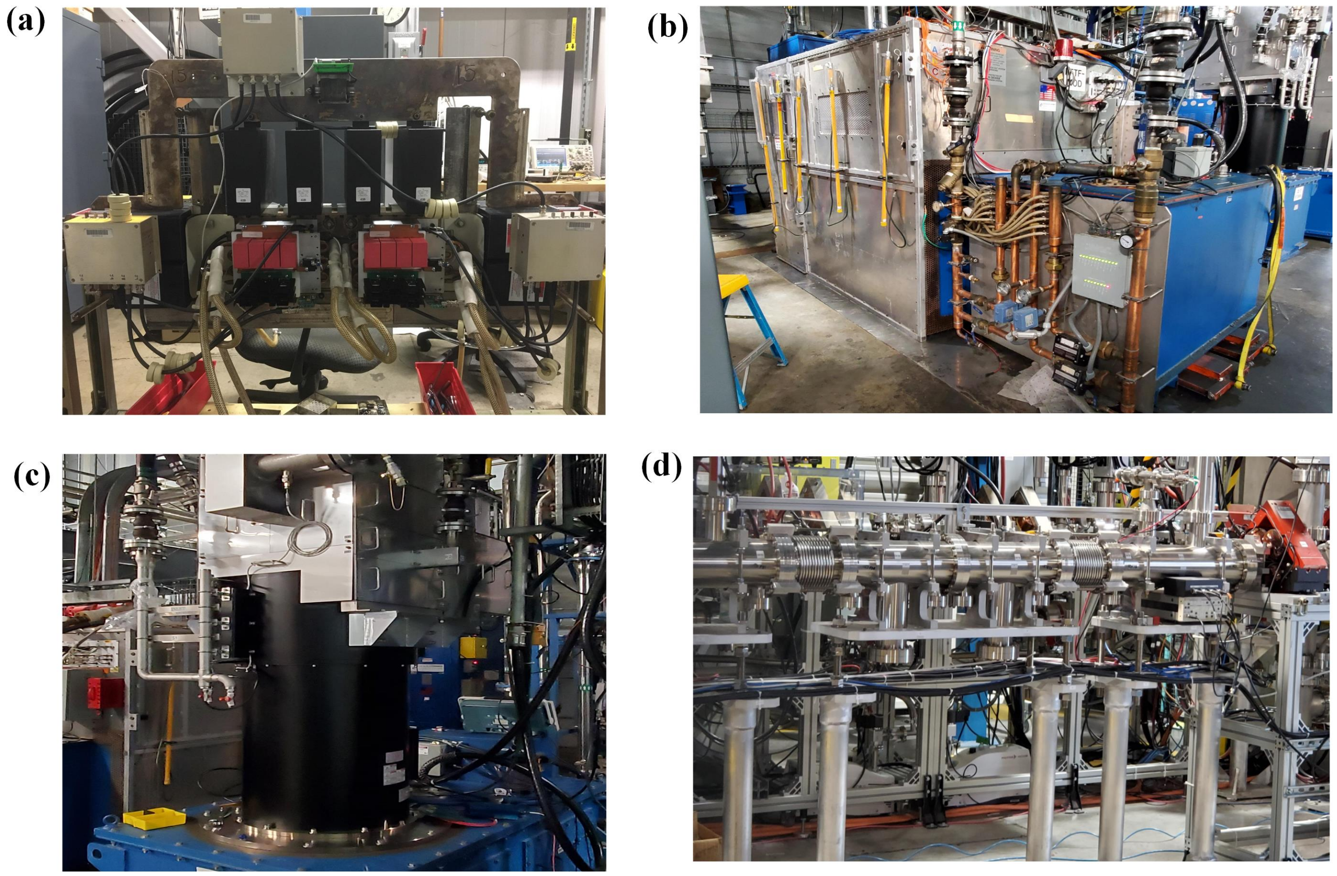

Figure 1. Components from the RFTF experimental setup: (a) HVCM H-bridge circuit, (b) HVCM insulation tank, (c) klystron (linear-beam vacuum tube), (d) section of the beam line

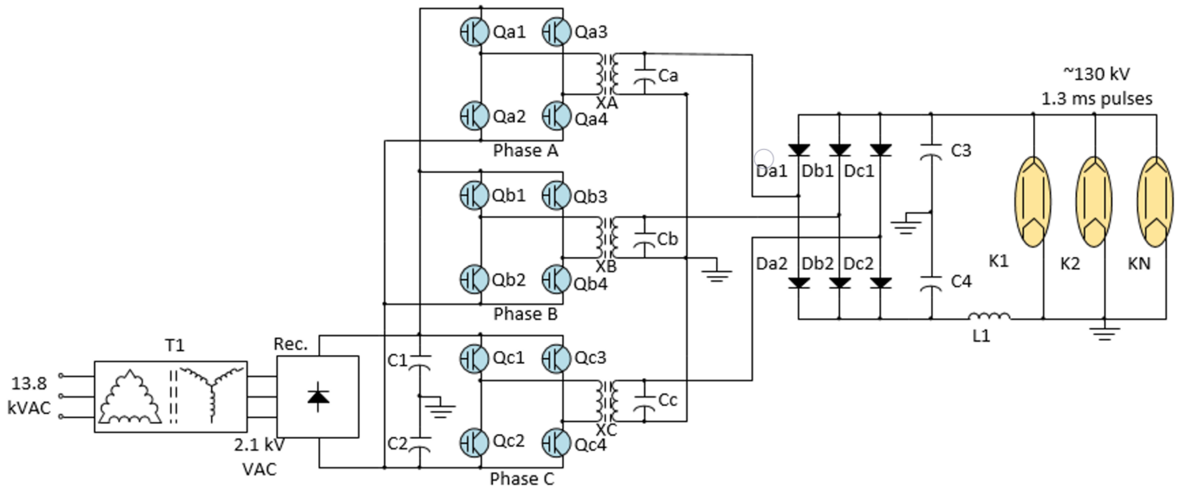

Figure 2. Simplified schematic of the HVCM circuit

normal/fault files archived for the SNS main HVCMs require storage of approximately 30 MB of data when decimated to 2.5 MS/s. However, not all of this data is useful. Therefore, to reduce the massive amount of disk space required to store streamed data, the number of streamed waveform channels was reduced from 32 to 12 in the RFTF. Also, the record length was reduced from 3.6 ms to 1.5 ms with a sampling rate of 400 ns. These changes have reduced the size of each waveform file from 30 MB to approximately 540 kB, which facilitate data pre-processing for the machine learning models and significantly reduce data size. The data are saved to an external hard drive for the authors to use. The 12 waveforms recorded by the controller are (by referring to Figure 2):

1. A+IGBT-I: The current passing through the IGBT switches (Qa1, Qa4) of phase A+ (unit: Ampere).
2. A+*IGBT-I: The current passing through the IGBT switches (Qa2, Qa3) of phase A+* (unit: Ampere).
3. B+IGBT-I: The current passing through the IGBT switches (Qb1, Qb4) of phase B+ (unit: Ampere).
4. B+*IGBT-I: The current passing through the IGBT switches (Qb2, Qb3) of phase B+* (unit: Ampere).
5. C+IGBT-I: The current passing through the IGBT switches (Qc1, Qc4) of phase C+ (unit: Ampere).
6. C+*IGBT-I: The current passing through the IGBT switches (Qc2, Qc3) of phase C+* (unit: Ampere).
7. Mod-I: Modulator current (unit: Ampere).
8. A-Flux: Magnetic flux density for phase A transformer (unit: scaled).
9. B-Flux: Magnetic flux density for phase B transformer (unit: scaled).
10. C-Flux: Magnetic flux density for phase C transformer (unit: scaled).
11. Mod-V: Modulator voltage (unit: kV).
12. CB-V: Cap bank voltage (unit: V).

The second improvement includes the ability to track waveform files when the HVCM is in the tuning mode. Tuning the HVCMs is normally done after maintenance on a particular HVCM. Tuning is done manually by experienced technicians and involves setting start and stop frequencies for IGBT gating to minimize droop, varying the start timing of the initial gate signals to minimize the likelihood of saturating the magnetic transformers. Tuning involves making incremental changes at reduced power while monitoring multiple signals such as klystron voltage, IGBT currents, and core magnetic flux to ensure they meet pulse requirements and remain within predetermined safe values. The data acquisition system records most of the waveforms during the tuning phase since the system would experience many changes.

The third improvement involves the file saving rate during operation, which is no longer fixed by the controller, but determined by the user. For example, the user can record waveforms at a rate of a waveform file every 3 seconds when a large demand for data acquisition is present. The rate can be decreased to a file every 10 minutes when the streamed data are not needed.

Figures 3-4 show live screenshots of the controller screen, which show the setting knobs that are used to tune the HVCM during startup. Also, plots of different waveforms are shown in the screen for the operator, which include a timing diagram of the IGBT gate pulses in Figure 3. In the next section, we will describe how we utilize this data acquisition system and the setting knobs to model fault scenarios.

3. METHODOLOGY

The methodology section involves description of data preparation, performing early fault detection test scenarios, the proposed models, and performance metrics used to evaluate the models. Our methodology workflow is summarized in Figure 5. The novelty of this work is that it lies in the intersection of robust experimental setup, quality data streaming, and machine learning applications; each one of these parts is described next.

3.1. Training Data Preparation

Following data streaming for about 3 days in normal operating conditions, we collected large amounts of training data at a rate of a waveform file every 5 seconds. For faulty training data, these are a bit more challenging to collect in large amounts. However, we managed to collect about 5000 fault files that come from two main sources: (1) real faults in the RFTF and (2) data collected during HVCM tuning (fault-like data). The second source dominates the faulty training data given that real faults in the HVCM do not occur very frequently (i.e. weeks to few months). As described in the previous section, HVCM tuning is done manually by the operators following a HVCM startup. This process usually involves tweaking the HVCM settings to different values to optimize the waveform shapes. Fortunately, our data streaming is programmed to record pulses at the maximum saving rate (1 pulse per second) during the tuning process, and these tuning waveforms deviate from normal operating ranges. In normal conditions, HVCMs operate at full power, while the data during the tuning phase deviate from these full power conditions, which make them a great source of fault-like data (not real faults). To remove the class imbalance, we only kept 5000 normal waveform files that sufficiently cover normal conditions, to be consistent with the number of faulty files.

After data streaming, the raw data are parsed from the CSV format and saved into a numpy array format with a 3D shape:

$$shape = (N_{pulses} \times N_{times} \times N_{features}) \qquad (1)$$

where $N_{pulses} = 10,000$ is the total number of pulses collected by the RFTF HVCM (both normal and faulty), N_{times}=3753 is the number of time steps for each pulse, and $N_{features} = 12$ is the number of features or waveform types recorded for each pulse. Given the sampling rate is 400 ns, the time length of the pulse is approximately 3753×400 ns $\approx$ 1.5 ms. The 12 waveforms (features) were described in section 2.2.

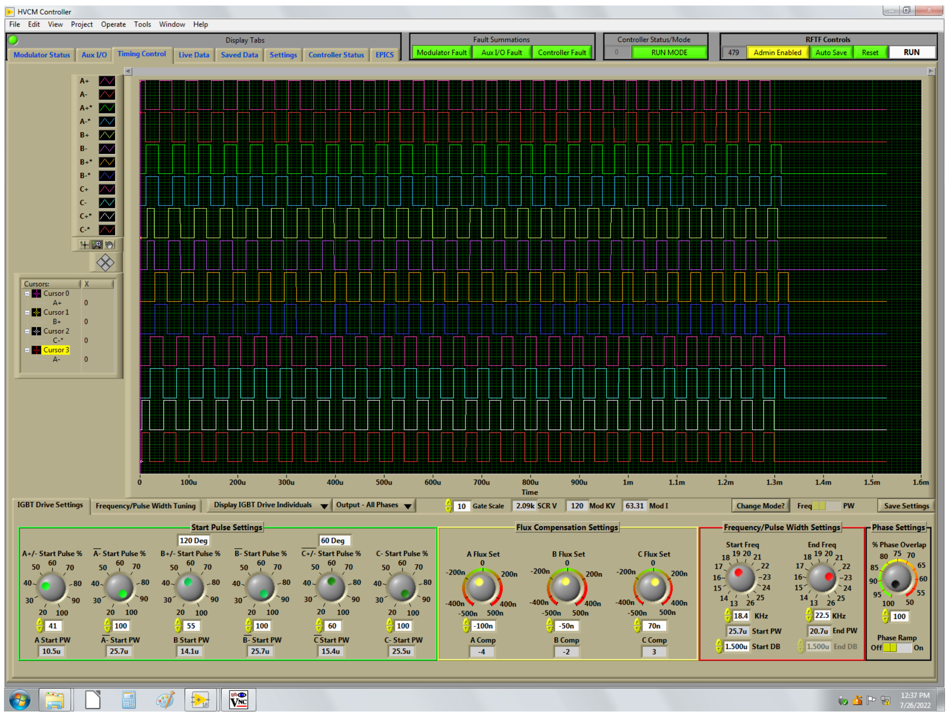
HVCM Controller
File Edit View Project Operate Tools Window Help
Display Tabs
Modulator Status Aux I/O Timing Control Live Data Saved Data Settings Controller Status EPICS
Fault Summations
Modulator Fault Aux I/O Fault Controller Fault
Controller Status/Mode
RUN MODE
RFTF Controls
Admin Enabled Auto Save Reset RUN
A+ A- A+* A-* B+ B- B+* B-* C+ C- C+* C-*
Time
IGBT Drive Settings Frequency/Pulse Width Tuning Display IGBT Drive Individuals Output - All Phases
Gate Scale SCR V Mod KV Mod I Change Mode? Freq PW Save Settings
Start Pulse Settings
Flux Compensation Settings
Frequency/Pulse Width Settings
Phase Settings
A Flux Set B Flux Set C Flux Set
A Comp B Comp C Comp
Start Freq End Freq Start PW End PW Start DB End DB
% Phase Overlap
Phase Ramp Off On

Figure 3. Screenshot of the RFTF HVCM live controller showing the settings and a timing diagram of the IGBT gate pulses

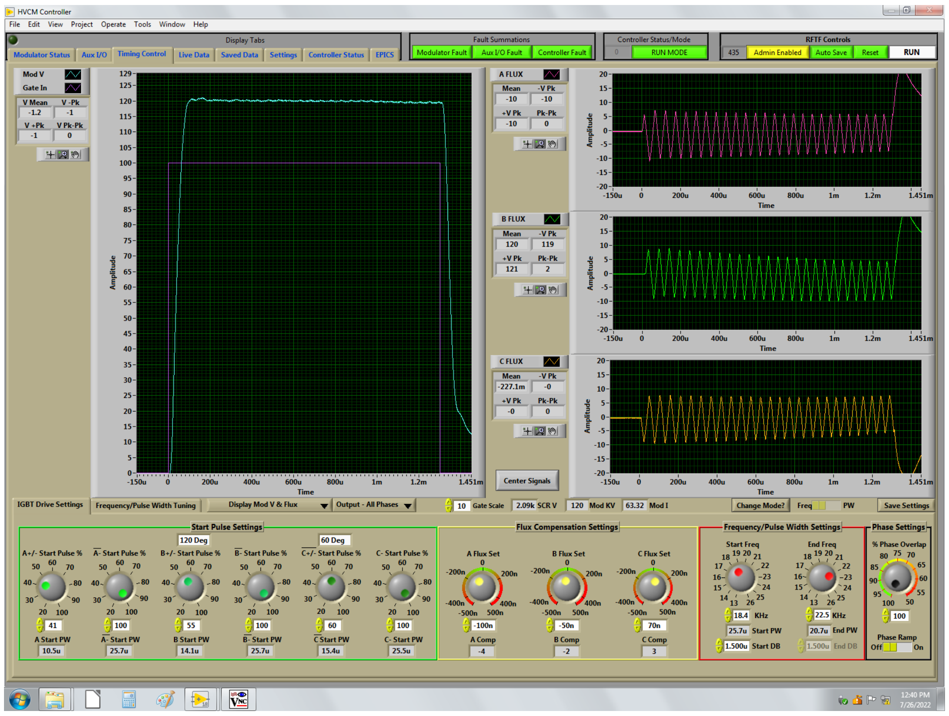
HVCM Controller
Mod V
Gate In
V Mean V -Pk V +Pk V Pk-Pk
Amplitude
Time
A FLUX
B FLUX
C FLUX
Mean -V Pk +V Pk Pk-Pk
Center Signals
IGBT Drive Settings Frequency/Pulse Width Tuning Display Mod V & Flux Output - All Phases
Start Pulse Settings
Flux Compensation Settings
Frequency/Pulse Width Settings
Phase Settings

Figure 4. Screenshot of the RFTF HVCM live controller showing the settings, Mod-V diagram (left), and magnetic flux diagrams (A-Flux, B-Flux, C-Flux) on the right

We plot the pulses/samples for the modulator current (Mod-I) in Figure 6. The reader can easily see that the fault data, which consist primarily of tuning data, deviate from the normal Mod-I pulses that look like identical. It is worth mentioning again that tuning data are not real faults, but can expose machine learning models to conditions that deviate from the

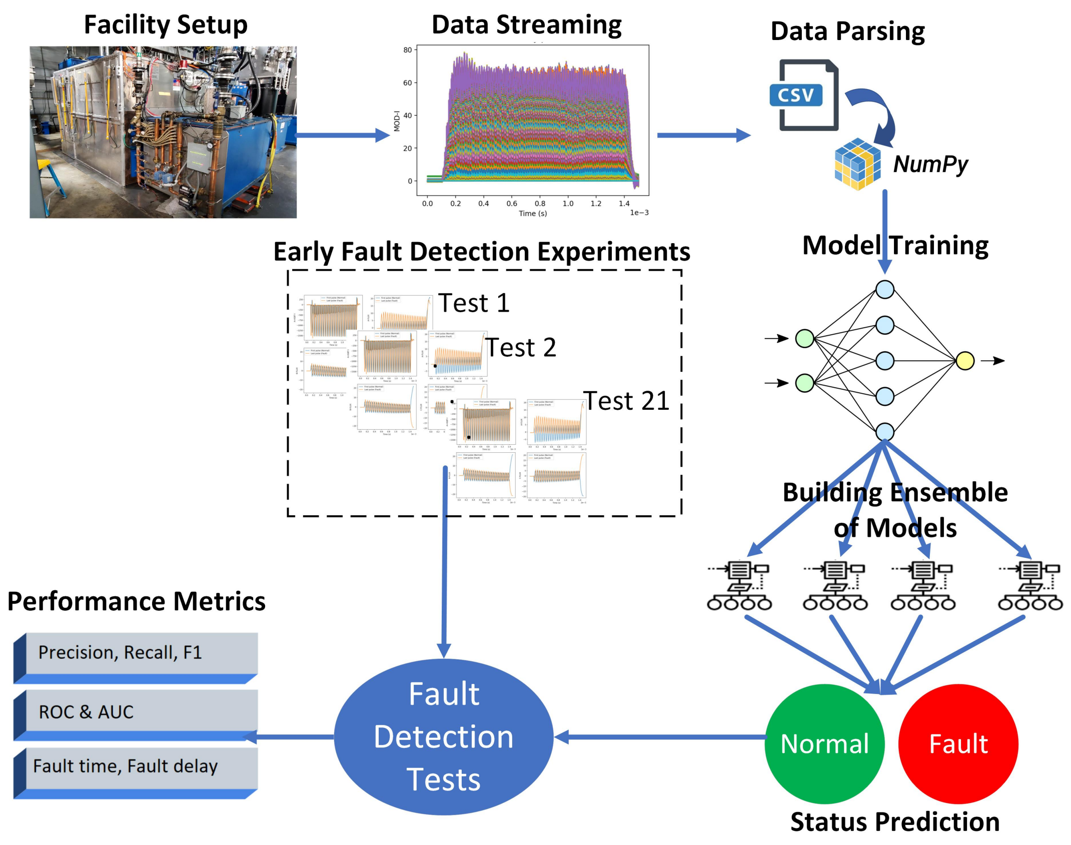

Figure 5. Methodology workflow

normal operating conditions. In both plots of Figure 6, all 5000 normal and faulty pulses are shown, so no legend is provided. Other waveforms (A-Flux, CB-V, A+IGBT-I, etc.) exhibit similar behaviour when comparing normal to faulty data.

3.2. Early Fault Detection Tests

As described before, HVCMs continue to have failures which can range from mild faults that can be resolved by a system restart to catastrophic failures that can lead to IGBT explosion. Figure 7(a)-(b) show a normal IGBT plate in the B phase along with its normal B-Flux waveform. In Figure 7(c)-(d), the IGBT is exploded and the B-Flux waveform immediately before the event was recorded in Figure 7(d), which shows a clear degradation and drooping in the flux signal. This fault event occurred during HVCM operation and was not planned by the authors.

In this study, the authors have performed 21 independent experiments trying to simulate the common faults facing the HVCM like the one in Figure 7 by gradually inducing anomalous changes in the HVCM settings. These setting adjustments are sufficient to cause abnormality in the waveforms but not serious to cause a real fault. The idea is to create a continuous test in time where the machine starts in a normal condition and gradually moves to a fault condition, where the proposed models are assessed in their capabilities in detecting those changes as soon as possible. Each test involves the following steps:

1. The data streaming system in section 2.2 is setup to save a waveform file every 7 seconds to allow the authors to make swift changes.
2. The team starts every test by waiting about 3 minutes to collect normal waveforms using the normal settings. This period collects about 26 waveform files.
3. The team then gradually induces changes in the RFTF settings and waits about a minute to collect waveform data under that change.
4. The settings are changed by adjusting 9 knobs in Figures 3-4 to pre-established values determined by the team. These knobs fall under the categories of “start pulse settings” and “flux compensation settings”. The changes can be in the form of increasing/decreasing start pulse width or increasing/decreasing flux compensation in the three phases. The adjustments can be in a single or multiple forms. Table 1 provides description of the plan and changes involved in each test.

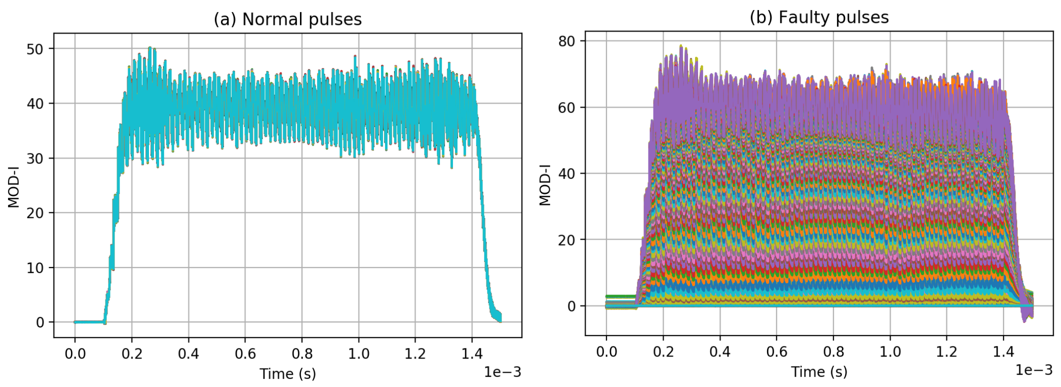

Figure 6. (a) 5000 normal modulator current (Mod-I) pulses, (b) 5000 faulty modulator current (Mod-I) pulses

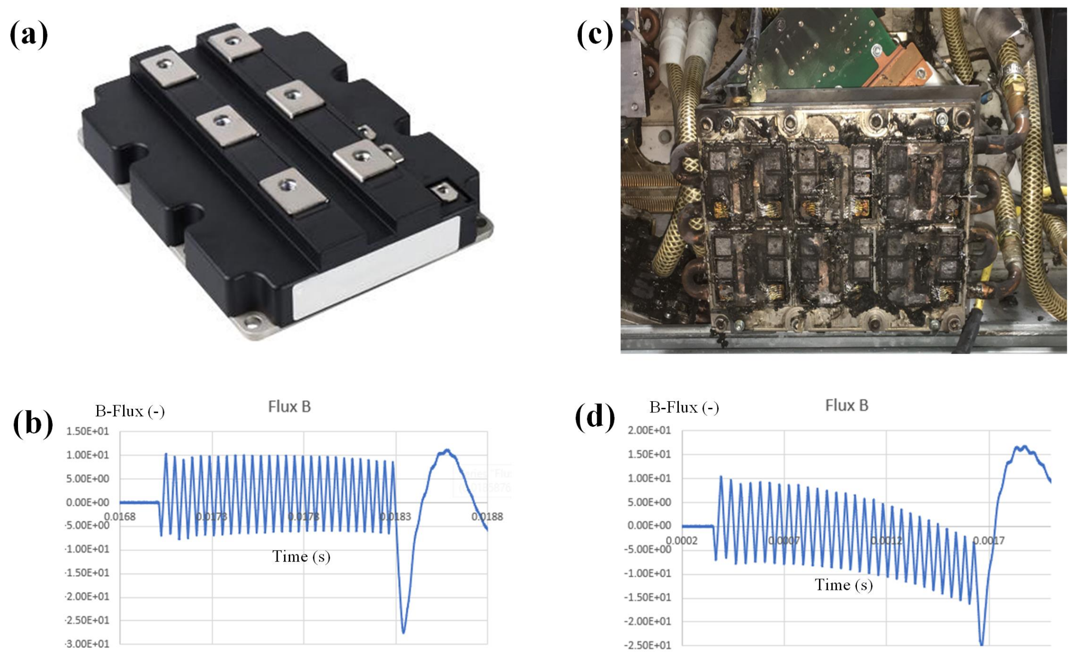

Figure 7. (a)-(b) Normal IGBT plate and its associated normal B-flux waveform. (c)-(d) Exploded IGBT plate and its associated faulty B-flux waveform (the pulse preceding the event)

5. Each test finishes when the allowed (max, min) setting value is reached or when the system is in a serious condition that could lead to immediate failure. Therefore, the reader can expect the tests to have different time lengths as in Table 1.

We provide sample plots of four selected waveforms from Test 1 and Test 11 in Figure 8, which involve increasing the A+ start pulse width and decreasing flux compensation in the B phase, respectively. In each subplot, the first pulse of the test (before any setting adjustment) and the last pulse (after all setting adjustments) are illustrated. These two tests along with the other 19 tests show interesting combinations of fault conditions that could be observed in single or multiple waveforms. For example, the creeping of the fault precursors in Test 1 can be seen to cause an effect in the IGBT current and the fluxes of all phases (A, B, C). However, Test 11 reveals that decreasing the B-Flux compensation only affects the B-Flux waveform, while the fault precursors in other waveforms cannot be detected, which make these fault scenarios a bit tricky to detect.

For completeness, it is interesting mentioning that our original experiment list was planned to include 22 tests. The very last test which involved sudden changes in all 9 knobs/settings together indeed caused a real fault that happened in a fraction of a second after the 3 minutes normal run. Fortunately, the fault was a minor C+ driver fault, which was fixed by a system restart without causing any damage. The authors decided not to repeat that experiment to avoid

Table 1. List of fault detection experiments conducted in the RFTF

Test ID	Description*	Time (s)
1	A+ start pulse width increases by 5%/min	868
2	B+ start pulse width increases by 5%/min	707
3	C+* start pulse width increases by 5%/min	616
4	A+ start pulse width decreases by 5%/min	469
5	B+ start pulse width decreases by 5%/min	581
6	C+* start pulse width decreases by 5%/min	637
7	A-Flux compensation increases by 25ns/min	770
8	B-Flux compensation increases by 25ns/min	924
9	C-Flux compensation increases by 25ns/min	868
10	A-Flux compensation decreases by 25ns/min	812
11	B-Flux compensation decreases by 25ns/min	588
12	C-Flux compensation decreases by 25ns/min	693
13	A+ start pulse width is set to 20%, A-Flux compensation increases by 25ns/min	924
14	A-*/B-*/C- start pulse widths all decrease by 5%/min	763
15	B+ start pulse width is set to 100%, B-Flux compensation decreases by 25ns/min	707
16	A-Flux compensation increases, B-Flux decreases, C-Flux increases by 25ns/min	581
17	A-Flux compensation decreases, B-Flux increases, C-Flux decreases by 25ns/min	630
18	C+* start pulse width is set to 90%, C-Flux compensation increases by 25ns/min	700
19	B-* start pulse width is set to 50%, B+ start pulse width increases by 5%/min	518
20	A+/A-* start pulse widths are set to 20%, A-Flux compensation decreases by 25ns/min	749
21	A-*/B-*/C- start pulse widths are set to 40%, A+/B+/C+* start pulse widths increase by 5%/min	532

causing major trouble to the system, and we were satisfied by having 21/22 tests successful. The question to be answered next is: Can we develop a robust model that can detect fault precursors as soon as possible in most of the 21 tests?

As the authors believe the current dataset can be useful for reproducibility of this study and for other researchers working in machine learning and fault detection areas, we shared all the data collected in this study. See the Data Availability section for more information.

3.3. Machine Learning Modeling

3.3.1. Standalone Models

The methods we select in this work belong to different categories including neural networks, ensemble of decision trees, and other classical methods. The convolutional neural network (CNN) classifier consists of Conv1D, max pooling, and fully-connected layers. The output is a binary prediction of the probability of a sample being normal or faulty pulse. The architecture of the CNN classifier is as follows:

1. Reshape layer to convert 2D data shape compatible with other methods (random forests, bagging classifier, etc.) to 3D shape compatible with Conv1D layers.
2. Conv1D layer with 32 filters, 6x6 kernel, ReLU activation, followed by max pooling of size 2x2.
3. Conv1D layer with 32 filters, 4x4 kernel, ReLU activation, followed by max pooling of size 2x2.
4. Conv1D layer with 16 filters, 3x3 kernel, ReLU activation, followed by max pooling of size 2x2.
5. Conv1D layer with 8 filters, 2x2 kernel, ReLU activation, followed by max pooling of size 2x2.
6. Flatten layer.
7. Dense layer with 32 nodes and ReLU activation.
8. Dense layer with 16 nodes and ReLU activation.
9. Dense layer with 2 nodes and Softmax activation.

The first reshaping layer is important to allow CNN to be trained as part of a large-scale ensemble that consists of other methods that support 2D data shape.

Ensemble methods combine several models (e.g. decision trees) to produce better predictive performance than utilizing a single model. Ensemble methods can be broadly classified into bagging and boosting. Bagging builds a stronger model by reducing model "variance" through combining predictions of different models after using bootstrapping to create random subsets of the dataset with replacement, where these subsets are used to train their corresponding models. On the other hand, boosting builds a stronger model by reducing model "bias". In boosting, models are built sequentially where each subsequent model attempts to correct the errors of the previous model. Early models tend to be weak, and as boosting iteration continues, the weights of the models are adjusted to make correct predictions of the difficult samples that prior models failed to predict. In this study, we select four variants of ensemble methods based on bagging and boosting, all are based on randomized decision trees:

1. Bagging Classifiers (BC) (Breiman, 1996): are ensemble estimators that fit base classifiers (e.g. randomized trees), each on random subsets of the original dataset, and then aggregate their predictions. The aggregation can be either by voting or by averaging their probabilistic prediction. Bagging methods are typically used to reduce the variance of the base estimators by introducing randomization into the procedure. In this work, we use BC with

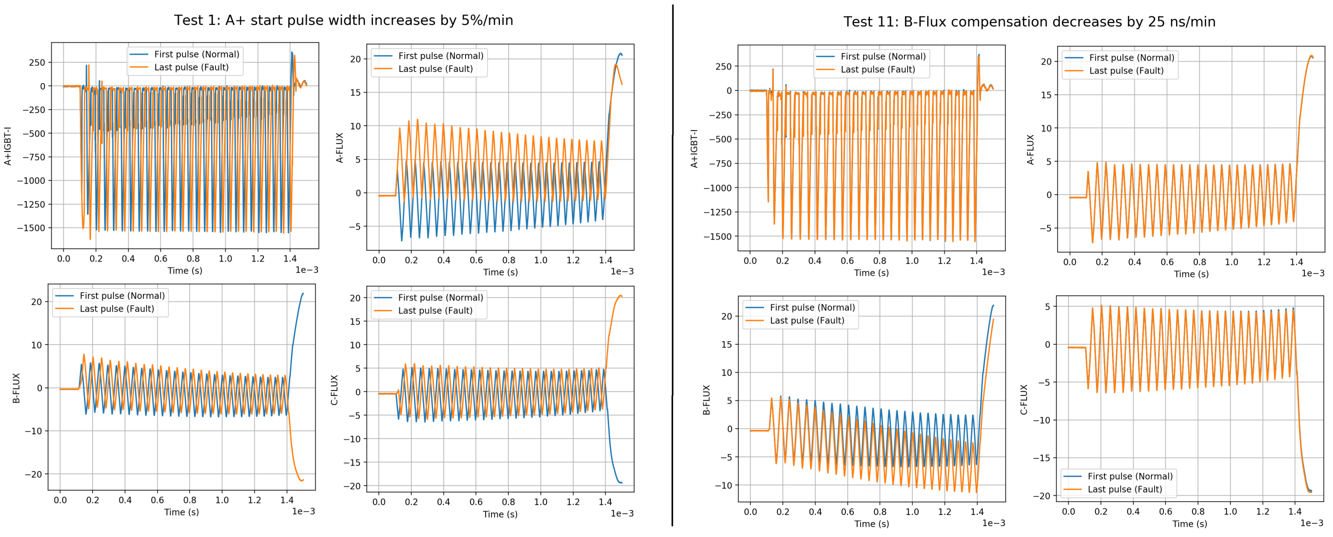

Figure 8. Test 1 (left) and Test 11 (right) results for four selected waveforms. The first pulse (before any change in settings) and the last pulse (after all changes are induced) are plotted

randomized trees and bootstrapping, where the subsets are drawn with replacement.

2. Random forests (RF) (Breiman, 2001): are a special case of BC where the estimators are decision trees, while BC is a framework that can be used with other base estimators (e.g. support vectors, k-nearest neighbours). Nevertheless, RF still have more improvements in how the ensemble trees are built. For RF, when splitting each node during the construction of a tree, the best split is found either from all features or a random subset of the maximum number of features. RF achieve a reduced variance by combining diverse trees and use bootstrapping at the cost of a small increase in bias. We also use Gini impurity metric to measure the quality of a split.
3. Extremely randomized trees (ET) (Geurts, Ernst, & Wehenkel, 2006): are a modification of RF with two fundamental differences: (1) no bootstrapping (meaning ET samples without replacement) and (2) the nodes in ET are split on “random” splits rather than RF that use “best” splits, which could improve the performance.
4. Adaboost (AB) (Freund & Schapire, 1997): The idea of AB is trying to fit a sequence of initial weak trees (i.e. random guessing) on modified versions of the dataset. The predictions from all trees are then combined using a weighted majority vote to produce the final prediction. The data modifications at each boosting iteration consist of applying weights to each of the training samples (starting by equal weights). Then the sample weights are modified and the algorithm is reapplied to the new weights in the next boosting iteration. After a certain number of iterations, the weights continue to adjust, where the algorithm focuses on the harder samples to predict to improve the overall performance.
5. Gradient Boosting (GB) (Friedman, 2001): GB is a generalization of AB to arbitrary differentiable loss functions compared to AB that uses the exponential loss function. For GB, binomial and multinomial deviance used in logistic regression is used as the loss function. GB also uses gradient descent to optimize the loss function.

In all previous methods, the number of trees/estimators is the main tunable parameter.

Support vector machines (SVM) (Noble, 2006; Cervantes, Garcia-Lamont, Rodríguez-Mazahua, & Lopez, 2020) are well-known supervised learning methods used for classification and regression problems. Training is performed by defining separation hyperplanes, such as a line on a 2D surface or a plane in a 3D surface, between training samples to separate the data samples into distinct classes on the sides of the hyperplane. SVM can solve both linear and non-linear dataset problems by employing the kernel trick to transform the data to higher dimensions where the classes can be separable. Therefore, in this work, we use two variants of SVM: (1) linear SVM (LSVM) where the linear kernel is used and (2) non-linear SVM (RBF-SVM) where the radial basis function (RBF) kernel is used.

3.3.2. Hierarchical Voting Ensemble (VE)

Voting ensemble is the expression of “democracy” in the machine learning community and we adopt this approach here to minimize bias. The decision making process in deciding whether the signal is normal or faulty is made by taking the votes of all standalone models in the ensemble, and the decision is made by majority voting. The structure of the voting ensemble is shown in Figure 9, which can be described in the following steps:

1. The training dataset is split into 12 subsets, each one highlights a single waveform data.

2. An ensemble model is trained by training 5 standalone models of RF, ET, GB, AB, and BC (i.e. these standalone models have shown the best performance when included in the ensemble).

3. Each standalone model in the ensemble will vote for the signal type if it is faulty or normal, and the final decision is taken by majority voting.

4. After all 12 waveform models make their vote, the final decision whether the current condition is normal or faulty is decided when **at least one** of the waveform models votes for a fault condition. Otherwise, the decision is normal.

If the final decision is a fault, the operators will see the waveforms that have anomalies to help them diagnosing the fault source.

3.4. Performance Metrics

We use standard metrics to evaluate the performance of our models in this study. Table 2 lists the classification metrics definition and their best and worst values. The four metrics are precision, recall, F1, and area under the curve (AUC), which is the area under the ROC curve (Receiver Operating Characteristic). The ROC curve is a standard diagram that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. An AUC=0.5 indicates a random classifier without any classification capability.

In addition, we report the values for some prognosis metrics according to (Saxena et al., 2008). First, we report the time of detection of fault by the model (t_F), which we compare to the true fault time (t_F^*) in the experiment. The fault time t_F is determined when the model predicts a fault event. If $t_F > t_F^*$, the delay in fault detection is determined by $\Delta t_F = t_F - t_F^* > 0$. Lastly, we report the fraction of the delay from the true total fault time as

$$\rho = \frac{\Delta t_F}{t - t_F^*}\%, \tag{2}$$

where t is the total experiment time from Table 1. The best value for ρ is 0 (no delay) while the worst is 100% (the fault is missed). If $t_F < t_F^*$, which is the case when the model makes a fault prediction earlier than the true time, both Δt_F and ρ can have negative values. Given that the fault detection tests have different time lengths, the delay fraction provides a more accurate representation of the fault delay.

4. RESULTS AND DISCUSSIONS

4.1. Analysis Settings

We performed a grid search to determine the optimal set of hyperparameters for each machine learning method in the study. Notice that the search was not very extensive and focused on the major parameters to avoid overfitting the models to the data, which may impact their ability to generalize. Based on that search, the following hyperparameters are utilized during the training process of all models:

1. AB: Number of estimators is 40 and learning rate is 1.0.
2. BC, RF, ET: Number of estimators is 40.
3. GB: Number of estimators is 40, max depth is 5, and learning rate is 0.1.
4. CNN: Batch size of 64, 5 epochs, and Adam optimizer with 5×10^{-4} learning rate. The loss function is the sparse categorical crossentropy.
5. SVM: Regularization parameter has a value of 0.5 for LSVM and 1.0 for RBF-SVM
6. VE: Inherits same hyperparameters of its members: BC, RF, ET, GB, and AB.

Min-max scaling is applied to all waveforms to facilitate training. Given most methods support 2D data shape (except CNN), the second and third axes in Eq.(1) are flattened into a single axis, so the real number of features becomes $N_{times} \times N_{features} = 3753 \times 12 = 45036$.

We have used Tensorflow/Keras with GPU support using CUDA and CuDNN libraries for the implementation of the CNN model. We also used Scikit-learn for the implementation of other machine learning models including our voting ensemble (VE). All training and analysis were conducted on a GPU cluster with 8 NVIDIA A100 SXM4 40GB GPUs available at the Spallation Neutron Source of the Oak Ridge National Laboratory.

4.2. Training and Testing Results

In the training and testing phase, all models are trained with 8000 pulses/samples and tested with 2000 (i.e. 0.2 test split) using the data in Figure 6. All 12 waveforms are included in the training process. The fault detection tests in Table 1 are not included in any sort during the training and testing phase. The results based on the test set indicate nothing but a superior and perfect performance by all 9 models, see Figure 10 which shows the confusion matrix for CNN. The confusion matrix shows that the CNN model did not make a single mistake in predicting the status of the 2000 test samples by having zero in the false positive and false negative entries. Other models (VE, AB, BC, GB, RF, ET, LSVM, RBF-SVM) show identical confusion matrix, but are not shown for brevity. Accordingly, all 9 models have a perfect 1.0 for precision, recall,

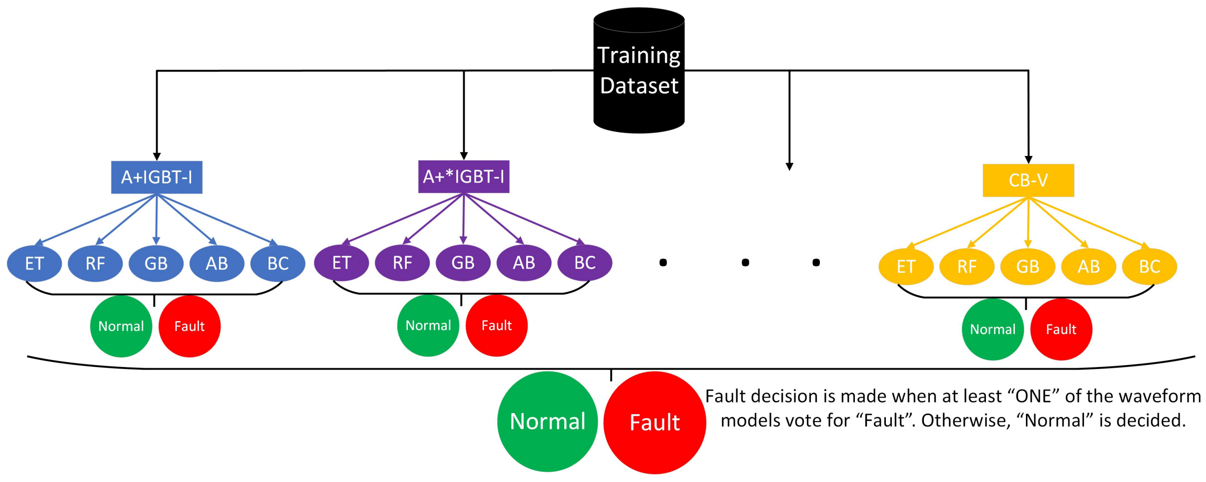
Training Dataset
A+IGBT-I
A+*IGBT-I
CB-V
ET
RF
GB
AB
BC
Normal
Fault
Fault decision is made when at least "ONE" of the waveform models vote for "Fault". Otherwise, "Normal" is decided.

Figure 9. Hierarchical structure of the voting ensemble (VE) built in this work for early fault detection

Table 2. Classification performance metrics used to evaluate the proposed models

Metric	Formula*	Notes
Precision	$Precision = TP/(TP + FP)$	Best=1.0, Worst=0.0
Recall	$Recall = TP/(TP + FN)$	Best=1.0, Worst=0.0
F1	$F1 = 2TP/(2TP + FP + FN)$	Best=1.0, Worst=0.0
AUC	$AUC = \int_0^1$ TPR (FPR) dFPR	Best=1.0, Worst=0.5

*TP: True Positive, FP: False Positive, FN: False Negative, FPR: False Positive Rate, TPR: True Positive Rate.

F1, and AUC, given the testing results. This perfect performance by all models reveals two interesting conclusions:

1. The amount of training data (see Figure 6) provided to all models is more than enough which makes all models perform very well at this stage, thanks to our facility setup and excellent data acquisition.
2. The values of the hyperparameters selected in section 4.1 for each model have no impact on the performance and are not causing any bias toward a certain model.

However, despite this amazing performance, the goal of this study is to predict the fault well ahead of time. Therefore, the question is, are all these perfect models going to generalize well for the 21 early fault detection scenarios?

4.3. Early fault Detection Analysis

To keep the paper concise, in this section, we focus the analysis on the VE model by presenting its corresponding results. The comparison with other models is provided in the second subsection.

4.3.1. Early Fault Detection Results of VE

The trained VE model is used to predict the fault timing of the 21 test scenarios using the hierarchy voting concept introduced in section 3.3.2. First, the classification metrics (precision, recall, F1, AUC) in distinguishing the normal from the faulty pulses in each test are listed in Table 3. The results show that the VE model still performs at a very high level by generalizing to the fault detection scenarios, obtaining almost perfect metrics for most of the test scenarios. For exam-

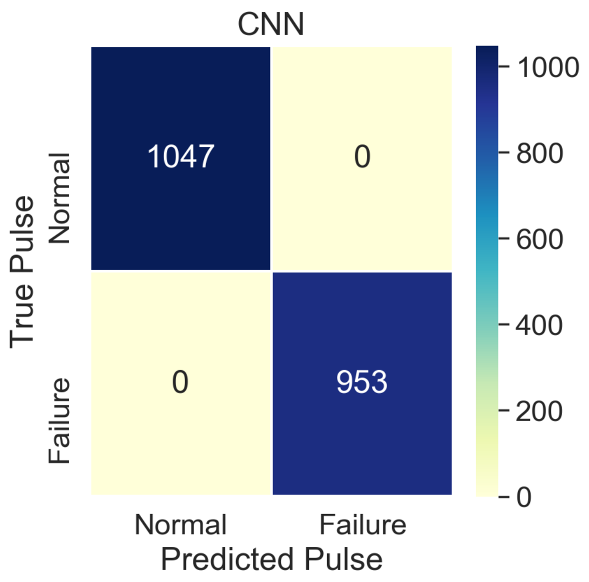

Figure 10. Confusion matrix based on the test set (2000 samples) for CNN. Other models (VE, LSVM, AB, BC, GB, RF, ET, RBF-SVM) show identical confusion matrix

ple, Tests 1-9, 13-14, 17-18, 20-21 show that the VE model is able to perfectly separate the pulses of the normal period (first 3 min) from the faulty pulses. Other tests (10, 12, 15, 16, 19) also have excellent metrics. Test 11 is the only test with poor results as it seems VE is not able to classify the pulses of this test properly.

Similarly, the fault detection metrics for the VE model are listed in Table 4, which shows the detection time of the fault precursors (t_F), the true time of the fault precursors (t_F^*), time delay of the detection (Δt_F), and the fraction of the time delay from the true total fault precursor time (ρ). For better visualisation, the delay and delay fraction are plotted in Figure 11 for all 21 tests. In agreement with the results of Table 3, Figure 11 illustrates that VE is generalizing very well by detecting the fault precursors as soon as they actually appear in the system by achieving zero delay and delay fraction for Tests 1-9, 13-14, 17-18, 20-21. In addition, Tests 15, 16 show a very small delay fraction of 1%, while Tests 10 and 12 show a larger but yet acceptable delay fractions of 10% and 23%, respectively. In Figure 11, Tests 11 and 19 illustrate some distinct differences compared to the rest of the group. Test 11 is completely missed by the VE model as its delay fraction is the maximum 100%, implying that the VE model was not predicting any fault signal after the first 3 min run. Test 11 is one of those tricky tests (see Figure 8), where all but a single waveform remain identical to the normal period. For Test 11, all waveforms except the B-Flux remain almost identical the whole test time. And although the changes in the B-Flux waveform are visible, these were not enough for the VE model to detect them. Two main points should be discussed for Test 11:

Table 3. Classification metrics when using the voting ensemble (VE) for the fault detection tests

Test ID	Precision	Recall	F1	AUC
1	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00
7	1.00	1.00	1.00	1.00
8	1.00	1.00	1.00	1.00
9	1.00	1.00	1.00	1.00
10	0.94	0.92	0.93	0.95
11	0.09	0.31	0.14	0.50
12	0.89	0.81	0.82	0.87
13	1.00	1.00	1.00	1.00
14	1.00	1.00	1.00	1.00
15	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.99
17	1.00	1.00	1.00	1.00
18	1.00	1.00	1.00	1.00
19	0.91	0.89	0.89	0.85
20	1.00	1.00	1.00	1.00
21	1.00	1.00	1.00	1.00

1. The total time of the test is about 588s (see Table 1), which is on the shorter side of the time scale compared to the other tests. Also, the nature of the precursor being introduced (only the B-Flux compensation is adjusted) could imply that running Test 11 for longer times with more adjustments may improve the performance.
2. As mentioned before, the authors avoided hyper-tuning of the proposed models to fit certain scenario(s), so the model can generalize well in the real world when it matters. Therefore, missing a single fault scenario and detecting the other 20 with a flexible model parameter set is already a major accomplishment.

The second test with interesting results is Test 19, which shows negative time delay and delay fraction, implying that the model started to detect the fault precursors before they appear in the system. In this case, 56 seconds earlier, which corresponds to -16.7% delay fraction. This is an interesting observation, and after we explored the raw data of Test 19, we found that indeed the system was having a certain amount of noise in the waveform signals during the normal period, and that noise was significant enough for the VE model to detect them.

Table 4. Fault detection metrics when using the voting ensemble (VE) for the fault detection tests

Test ID	t_F^* (s)	t_F (s)	Δt_F (s)	$\rho(\%)$
1	182	182	0	0
2	182	182	0	0
3	182	182	0	0
4	182	182	0	0
5	182	182	0	0
6	182	182	0	0
7	182	182	0	0
8	182	182	0	0
9	182	182	0	0
10	182	245	63	10
11	182	588	406	100
12	182	301	119	23.3
13	182	182	0	0
14	182	182	0	0
15	182	189	7	1.3
16	182	189	7	1.8
17	182	182	0	0
18	182	182	0	0
19	182	126	-56	-16.7
20	182	182	0	0
21	182	182	0	0

To clarify more, Figure 12 shows a plot of the B-Flux and Mod-I waveform pulses of Test 19. Only the pulses during the **second** and **third** minutes of the test are shown, which belong to the normal period when the VE model indicated fault precursors despite no precursors being introduced yet. The reader can clearly see in Figure 12 that these normal pulses are not identical as desired, and some of them show clear deviation which can be seen more clearly in B-Flux and Mod-I waveforms. For example, the yellow and grey pulses obviously deviate from the rest of the group as indicated in

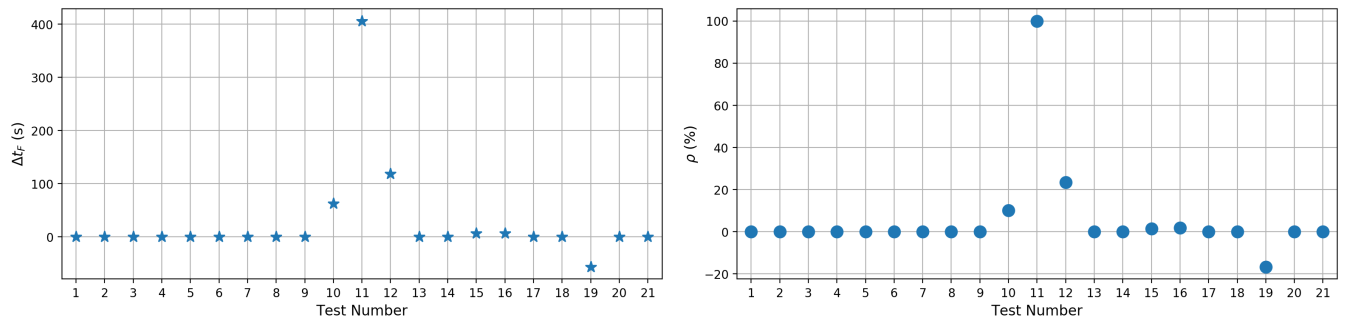

Figure 11. Fault detection performance of the voting ensemble (VE) with fault delay time (left) and delay fraction (right). Test 11 is the only test missed by the model

the zoomed version of Figure 12. This observation reveals excellent proof that this model actually works and was able to detect those abnormal signals earlier in time even though they are not caused by a real fault precursor, but coming from system noise. While the authors are not certainly sure of the reason why those abnormal signals show up in Test 19 in particular, the reason may be attributed to how these tests have been conducted. The authors have conducted these tests in a scheduled time frame where we usually leave 3-5 minutes before starting the next test. One possible reason could be that the machine was not able to restore its normal status following the several changes of Test 18, so small abnormalities remained and were exacerbated after we started Test 19. As the reader can tell, these abnormalities are very difficult to detect by humans unless a clear benchmark to normal signals is provided as in Figure 12. Test 19 results indeed show how these machine learning models can be really valuable in detecting such a subtle change in waveforms. Now, after the normal period ends and real fault precursors start to appear, the VE model continues the excellent performance as can be easily told from the excellent classification metrics for Test 19 in Table 3.

4.3.2. Performance Comparison

All 9 trained models are now applied to the 21 tests to evaluate their performance compared to the VE model, and the results are plotted in a bar chart in Figure 13. We only provided the delay fraction (ρ) and F1 score as representative metrics. Due to the voluminous size of the results for all models and to maintain a concise article, we only reported these two metrics in this paper.

By looking at the results, it is clear that the perfect models in the training/testing phase (of section 4.2) are no longer that good coming into the early fault detection phase. The models do not seem to provide a satisfactory performance compared to VE in most of the tests. For example, in Tests 5-7, the models other than VE have missed the fault precursors with a large delay (sometimes 100%) and have provided a poor F1 score, while VE shows the opposite. And this is the same story for most of the other tests, however, with some tests showing variability in performance between the models. For example, Test 1 shows ET as the best method after VE, Test 2 shows that BC, RF, GB are as good as VE, Test 3 shows AB and VE to be the best, and so on. The major observations to be discussed from Figure 13:

- All models including VE failed to detect Test 11, obtaining comparable metrics. This could imply that the precursors of Test 11 are hard to detect.
- All models except AB show a negative delay fraction for Test 19, implying that the noise precursors were detectable by most models. This agreement reinforces our observations for VE. AB demonstrated poor performance, missing the whole Test 19.
- Test 15 shows that all models have comparable delay fractions, but by looking at their F1 score, it implies something else. Although all models were able to detect the precursors early enough, some models started to make wrong predictions afterwards, tagging faulty pulses as normal. For example, VE and AB have very good F1 and ρ scores, but ET has a good ρ but a mediocre F1 score of less than 0.6. This shows the value of looking at different metrics to fully assess model performance.

To provide a comprehensive and concise comparison between all models based on these metrics, we counted the number of tests being passed by each model. The passing condition is to achieve a delay fraction $\rho < 25\%$ and $F1 > 0.8$. While these thresholds are arbitrary for evaluation, a 25% delay still provides the operator with enough time to take an action. The summary is provided in Table 5, which shows how these models are ranked. Obviously, the voting classifier (VE) excels by far passing 20 out of 21 tests, yielding an impressive success rate of 95%. The rest of the models are ranked next with comparable performances, with AB and ET coming next with 52% and 48% success rates, respectively. The CNN, BC, RF, GB have similar overall performance, while the two SVM models (LSVM and RBF-SVM) show the worst performance without any advantage of adding the RFB kernel over the linear kernel.

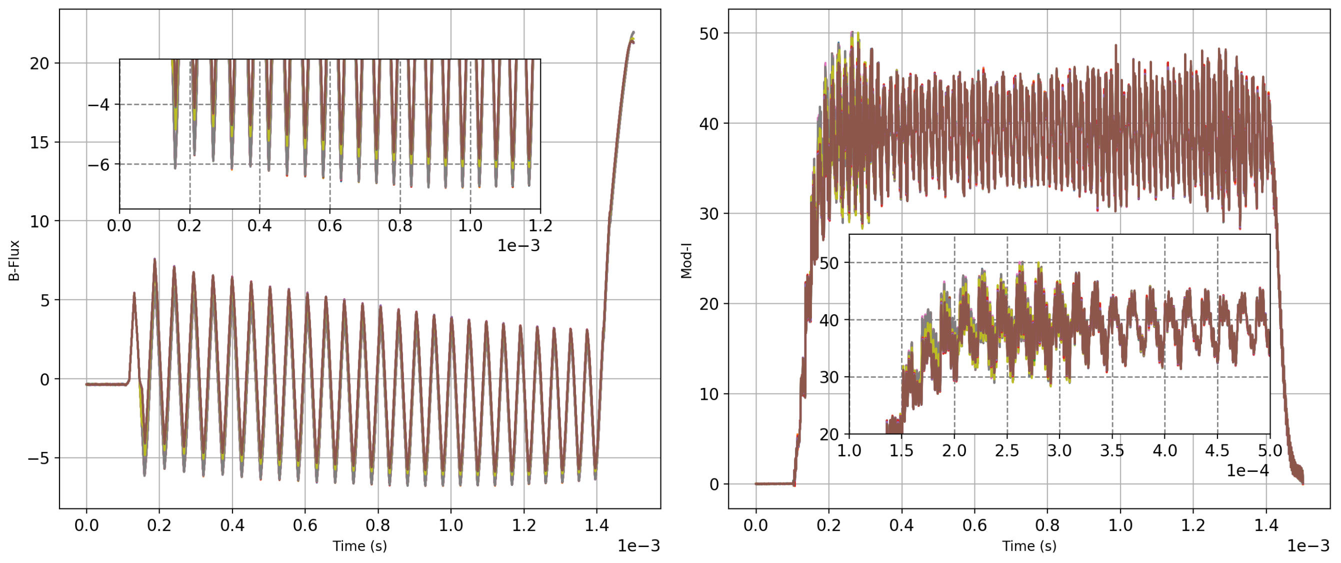

Figure 12. Plot of the B-Flux and Mod-I waveform pulses during the **second** and **third** minutes of Test 19 along with zoomed version

Table 5. Comprehensive summary of the methods passing the early fault detection tests with: $\rho < 25\%$ and $F1 > 0.8$ and their fraction of the total (success rate)

Method	Passed Tests	Success Rate (%)*
VE	20	95.2
CNN	9	42.9
BC	9	42.9
RF	9	42.9
ET	10	47.6
GB	9	42.9
AB	11	52.4
LSVM	5	23.8
RBF-SVM	5	23.8

* Success rate is the fraction of the passed tests from the total number of tests (21).

The proposed models in this work encompass a wide range of machine learning methods from ensembles to neural networks to classical methods like SVM. The VE model, the main hierarchical ensemble based on the voting concept, illustrated to be the best. The VE model is an ensemble of multiple layers, each consists of an ensemble as shown in Figure 9. The first layer features different models, each voting on the status of its corresponding waveform. The second layer involves using five different sub-models within each waveform model to vote on the status of that waveform. In this context, we used models based on random forests, extremely randomized trees, bagging classifiers, adaboost, and gradient boosting. The third layer occurs within each sub-model (RF, BC, etc.), where each sub-model consists of number of estimators (i.e. 40) voting for each sub-model. This advanced hierarchy provides a strong diversity in the decision making process which is the main reason for the excellent performance of VE. The second advantage of this approach is that it significantly reduces the sensitivity of the hyperparameters of each model/estimator given the decision is made by more than 100 models rather than a single model.

The remaining models show that standalone ensembles (GB, BC, etc.) can be as good as neural network models (CNN) for early fault detection, while classical models such as SVM seem to be less powerful. For completeness, we also tested k-nearest neighbors (KNN) and feedforward neural networks, with both showing a poor performance that is not worth to be reported here. This is aside from the fact that KNN was very slow to train and make predictions. Also, we explored including the CNN, LSVM, and RSVM into the VE ensemble without noticing major improvements, albeit that the SVM models made VE even weaker.

Aside from the machine learning part, we should reiterate on the significant value of the experimental setup used in this study, which we believe is as important as the machine learning part. The experimental part involves significant efforts by electronics and control engineers to facilitate data streaming and testing, which were vital to obtain quality and adequate quantity of data to empower machine learning. The RFTF facility can be used in future computational and machine learning studies to explore these techniques in other parts of the accelerator.

The results of this study also reveal that the classical machine learning approach of splitting data into training and test sets is not always guaranteed to confirm the model performance. In this work, 9 models show identical and perfect performance on the test set, but once they are introduced to new scenarios,

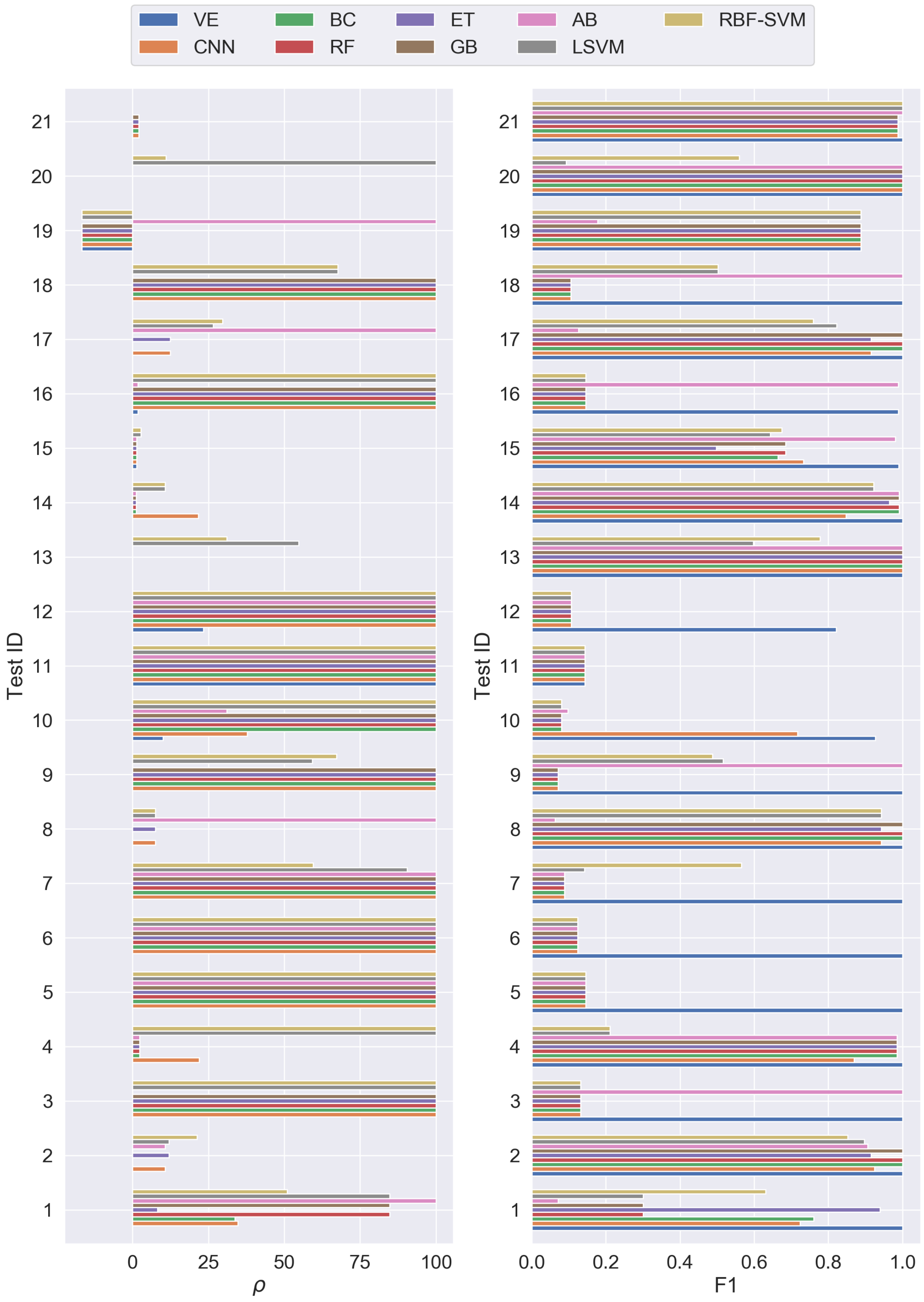

Figure 13. Comparison of the delay fraction (ρ) and F1 score for all methods and all tests

the performance of 8 out of 9 decreased at least by half. This practice reveals the value of implementing the trained models and exposing them to new data to confirm their ability to generalize.

In terms of computing time, the training expenses are quite comparable for most models except for VE, which is more expensive than others as it involves many more models to train. However, given that training is done offline, and the trained model is what is being used on the system, the model prediction time is more important. Similarly, we expect the VE model to be also slower as it counts votes from many submodels to make a prediction. The tests reveal that RBF-SVM is the slowest model among all, taking a prediction time of 28 ms/sample, most likely due to the non-linear transformation. Next is the VE model taking 6 ms/sample, then LSVM with 3 ms/sample, CNN with 2 ms/sample, while the rest of the models (GB, BC, GB, RF, ET) take in average between 1.5-1.8 ms/sample. Given we are streaming waveforms at a rate of 3-5 s, the prediction time of all models is much lower.

In this work, we accomplished two main goals that were limitations of our previous effort (Radaideh, Pappas, Walden, et al., 2022): (1) resolving data limitations by upgrading the controller and the data acquisition system of the HVCM, (2) demonstrating that under near-continuous data streams, machine learning can be effectively used for early fault detection to detect the fault precursors well ahead of time. In the previous paper, (Radaideh, Pappas, Walden, et al., 2022), which highlights the HVCM powering the RFQ section, the paper demonstrates a promising potential for fault detection, but with a very limited time scale (about 1.5 ms before the fault happens), which is a time that only allows for a quick system shutdown. This approach may prevent the fault from happening and reduce the damage to the HVCM (i.e. through preventive maintenance), but does not resolve the downtime issue of the SNS. The results of this work, which are based on the RFTF facility simulating the SNS conditions, extend these methods to allow for early fault detection, prognosis, and predictive maintenance, giving the operators sufficient time to either re-tune the modulator, skipping the warning if it is only a noisy signal, or shutting down the system if the issue is serious.

Broadly speaking, our proposed techniques can be used for other particle accelerators only if their data structure is consistent with our research. To illustrate, our input data is comprised of time series signals, with known labels, hence we anticipate that the data structure of other accelerators would be similar. Furthermore, the size of the data, the number of unique waveforms, and the level of noise in the signals (i.e., signal quality) would all impact the effectiveness of the models and the degree of hyperparameter tuning required.

5. CONCLUSIONS

In this work, a variety of machine learning methods is tested in performing early fault detection in particle accelerator power electronics to reduce their catastrophic failures and improve particle accelerator reliability. The study highlights the spallation neutron source (SNS) and the high voltage converter modulators (HVCM) that power the klystrons and accelerating cavities. An advanced experimental setup featuring a radio-frequency test facility with operating conditions similar to the SNS is used to stream waveform data in much higher rates than what was achieved before. The authors have conducted 21 test experiments mimicking the fault conditions that occurred in the HVCM in the past without causing a real fault, where machine learning models are tested in discovering these fault precursors as soon as they are introduced in the system. A variety of techniques including ensemble trees, convolutional neural networks, support vector machines, and hierarchical voting ensembles are trained and tested in this study. Although all models have shown a perfect and identical performance during the training and testing phase, the performance of most models has decreased in the early fault detection scenarios once they got exposed to real-world data that feature the 21 experiments. Nevertheless, the hierarchical voting ensemble maintains a distinguished performance in early detection of the fault precursors in 20 out of 21 tests, followed by adaboost and extremely randomized trees that detected 11 tests and 10 tests, respectively. The support vector machine models provided the worst performance detecting only 5 tests. The performance of the other models was in between.

Overall, the results of this study lead to several conclusions that for a successful implementation of machine learning in the SNS or particle accelerator power systems (e.g. HVCM), the data acquisition system should be improved to be more advanced and capable for continuous streaming and handling of big data to feed to the machine learning models. The machine learning models are better to be diverse and based on ensemble concepts to reduce bias and hyperparameter sensitivity. Given the authors have shared all experimental data used in this work, future work ideas would focus on improving the models architecture and their generalizing abilities to allow for accurate fault detection, prognosis, and predictive maintenance.

ACKNOWLEDGMENT

The authors are grateful for support from the Neutron Sciences Directorate at ORNL in the investigation of this work. This work was supported by the DOE Office of Science under grant DE-SC0009915 (Office of Basic Energy Sciences, Scientific User Facilities program). This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. The authors would like to thank Willem Blokland from the spallation neutron source for his useful comments and feedback on this work.

DATA AVAILABILITY

The data will be shared in a Mendeley repository following the peer-review of this work to ensure the authors' credit is fully appreciated. The data will include the training data as well as the fault detection tests. Also, a companion data article is prepared to describe how to utilize the dataset.

CREDIT AUTHOR STATEMENT

Majdi I. Radaideh: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Visualisation, Formal analysis, Writing - Original Draft.
Chris Pappas: Conceptualization, Data Curation, Software, Writing – Review and Edit.
Mark Wezensky: Data Curation, Software, Validation, Writing – Review and Edit.
Pradeep Ramuhalli: Conceptualization, Methodology, Supervision, Writing – Review and Edit.
Sarah Cousineau: Conceptualization, Funding acquisition, Project Administration, Writing – Review and Edit.

REFERENCES

Agasthian, A., Pamula, R., & Kumaraswamidhas, L. A. (2019). Fault classification and detection in wind turbine using cuckoo-optimized support vector machine. *Neural Computing and Applications*, *31*(5), 1503–1511.

Arunthavanathan, R., Khan, F., Ahmed, S., & Imtiaz, S. (2021). A deep learning model for process fault prognosis. *Process Safety and Environmental Protection*, *154*, 467–479.

Belagoune, S., Bali, N., Bakdi, A., Baadji, B., & Atif, K. (2021). Deep learning through lstm classification and regression for transmission line fault detection, diagnosis and location in large-scale multi-machine power systems. *Measurement*, *177*, 109330.

Blokland, W., Ramuhalli, P., Peters, C., Yucesan, Y., Zhukov, A., Schram, M., ... Jeske, T. (2021). Uncertainty aware anomaly detection to predict errant beam pulses in the sns accelerator. *arXiv preprint arXiv:2110.12006*.

Bode, G., Thul, S., Baranski, M., & Müller, D. (2020). Real-world application of machine-learning-based fault detection trained with experimental data. *Energy*, *198*, 117323.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Bukkapatnam, S. T., Afrin, K., Dave, D., & Kumara, S. R. (2019). Machine learning and ai for long-term fault prognosis in complex manufacturing systems. *Cirp Annals*, *68*(1), 459–462.

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, *408*, 189–215.

Edelen, A. L., Biedron, S., Chase, B., Edstrom, D., Milton, S., & Stabile, P. (2016). Neural networks for modeling and control of particle accelerators. *IEEE Transactions on Nuclear Science*, *63*(2), 878–897.

Felsberger, L., Apollonio, A., Cartier-Michaud, T., Müller, A., Todd, B., & Kranzlmüller, D. (2020). Explainable deep learning for fault prognostics in complex systems: A particle accelerator use-case. In *International cross-domain conference for machine learning and knowledge extraction* (pp. 139–158).

Fernandes, M., Corchado, J. M., & Marreiros, G. (2022). Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: a systematic literature review. *Applied Intelligence*, 1–35.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3–42.

Hajji, M., Harkat, M.-F., Kouadri, A., Abodayeh, K., Mansouri, M., Nounou, H., & Nounou, M. (2021). Multivariate feature extraction based supervised machine learning for fault detection and diagnosis in photovoltaic systems. *European Journal of Control*, *59*, 313–321.

Henderson, S., Abraham, W., Aleksandrov, A., Allen, C., Alonso, J., Anderson, D., ... others (2014). The spallation neutron source accelerator system design. *Nuclear*

Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, *763*, 610–673.

Kolokas, N., Vafeiadis, T., Ioannidis, D., & Tzovaras, D. (2020). A generic fault prognostics algorithm for manufacturing industries using unsupervised machine learning classifiers. *Simulation Modelling Practice and Theory*, *103*, 102109.

Kozjek, D., Butala, P., et al. (2017). Knowledge elicitation for fault diagnostics in plastic injection moulding: A case for machine-to-machine communication. *CIRP annals*, *66*(1), 433–436.

Liu, C., Tang, D., Zhu, H., & Nie, Q. (2021). A novel predictive maintenance method based on deep adversarial learning in the intelligent manufacturing system. *IEEE Access*, *9*, 49557–49575.

Luo, B., Wang, H., Liu, H., Li, B., & Peng, F. (2018). Early fault detection of machine tools based on deep learning and dynamic identification. *IEEE Transactions on Industrial Electronics*, *66*(1), 509–518.

Mohapatra, D., Subudhi, B., & Daniel, R. (2020). Real-time sensor fault detection in tokamak using different machine learning algorithms. *Fusion Engineering and Design*, *151*, 111401.

Mohd Amiruddin, A. A. A., Zabiri, H., Taqvi, S. A. A., & Tufa, L. D. (2020). Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems. *Neural Computing and Applications*, *32*(2), 447–472.

Nguyen, D., Lee, M., Sass, R., & Shoaee, H. (1991). *Accelerator and feedback control simulation using neural networks* (Tech. Rep.). Stanford Linear Accelerator Center, Menlo Park, CA (USA).

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, *24*(12), 1565–1567.

Radaideh, M. I., Pappas, C., & Cousineau, S. (2022). Real electronic signal data from particle accelerator power systems for machine learning anomaly detection. *Data in Brief*, *43*, 108473.

Radaideh, M. I., Pappas, C., Walden, J., Lu, D., Vidyaratne, L., Britton, T., ... Cousineau, S. (2022). Time series anomaly detection in power electronics signals with recurrent and convlstm autoencoders. *Digital Signal Processing*, *130*, 103704.

Rescic, M., Seviour, R., & Blokland, W. (2020). Predicting particle accelerator failures using binary classifiers. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *955*, 163240.

Reščič, M., Seviour, R., & Blokland, W. (2022). Improvements of pre-emptive identification of particle accelerator failures using binary classifiers and dimensionality reduction. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *1025*, 166064.

Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008). Metrics for evaluating performance of prognostic techniques. In *2008 international conference on prognostics and health management* (pp. 1–17).

Scheinker, A. (2021). Adaptive machine learning for robust diagnostics and control of time-varying particle accelerator components and beams. *Information*, *12*(4), 161.

Shao, H., Jiang, H., Wang, F., & Zhao, H. (2017). An enhancement deep feature fusion method for rotating machinery fault diagnosis. *Knowledge-Based Systems*, *119*, 200–220.

Syafrudin, M., Alfian, G., Fitriyani, N. L., & Rhee, J. (2018). Performance analysis of iot-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors*, *18*(9), 2946.

Taqvi, S. A., Tufa, L. D., Zabiri, H., Maulud, A. S., & Uddin, F. (2020). Fault detection in distillation column using narx neural network. *Neural Computing and Applications*, *32*(8), 3503–3519.

Vachtsevanos, G. J., & Vachtsevanos, G. J. (2006). *Intelligent fault diagnosis and prognosis for engineering systems* (Vol. 456). Wiley Online Library.

Wang, L., Zhang, Z., Long, H., Xu, J., & Liu, R. (2016). Wind turbine gearbox failure identification with deep neural networks. *IEEE Transactions on Industrial Informatics*, *13*(3), 1360–1368.

Wang, P., Poovendran, P., & Manokaran, K. B. (2021). Fault detection and control in integrated energy system using machine learning. *Sustainable Energy Technologies and Assessments*, *47*, 101366.

Zhang, L., Lin, J., Liu, B., Zhang, Z., Yan, X., & Wei, M. (2019). A review on deep learning applications in prognostics and health management. *Ieee Access*, *7*, 162415–162438.

Zhang, W., Yang, D., & Wang, H. (2019). Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, *13*(3), 2213–2227.

Zhang, Y., Beudaert, X., Argandoña, J., Ratchev, S., & Munoa, J. (2020). A cpps based on gbdt for predicting failure events in milling. *The International Journal of Advanced Manufacturing Technology*, *111*(1), 341–357.