Preemptive Detection of Electrical System Anomalies in Particle Accelerators

Timur Guler
School of Data Science
University of Virginia
Charlottesville, VA, USA
tguler8@gmail.com

MacKenzye Leroy
School of Data Science
University of Virginia
Charlottesville, VA, USA
zuf9mc@virginia.edu

Colin O'Brien
School of Data Science
University of Virginia
Charlottesville, VA, USA
tne5bn@virginia.edu

Ryan Pindale
School of Data Science
University of Virginia
Charlottesville, VA, USA
mwp8zy@virginia.edu

Abstract—**Large-scale instruments are vital to the progression of scientific discovery. Instrument downtime often stalls research; by reducing downtime, experimenters can increase research productivity and attain higher returns on investment. Our team focused on instruments of high complexity, where electrical issues in various subcomponents have the potential to cause problems ranging from simple experimental failure to catastrophic system damage. We propose a novel approach for preemptive detection of electrical faults using a variety of machine learning methods on signal data from Oak Ridge Laboratory's Spallation Neutron Source (SNS) particle accelerator. We compared four methods: a prototypical network that uses Symbolic Fourier Approximation for feature engineering and few shot learning for training, a Gaussian Process Classifier, an Approximated Bayesian Neural Network using Monte Carlo Dropout, and an LSTM Autoencoder. We evaluate these methods based on their ROC curves and provide a general commentary on the advantages and disadvantages of each method. Our results demonstrate capacity for identifying the imminence of certain failure states and provide avenues for future enhancement.**

Index Terms—**Feature Engineering, Deep Learning, Physics, Signal Data**

I. Introduction

System downtime has historically been a major impediment to experimental production at the SNS since its inauguration. More specifically, issues with the fifteen High Voltage Converter Modules (HVCM) have hindered progress due to the HVCM's pivotal role in SNS operations. Over time, Oak Ridge has made significant strides in reducing HVCM downtime through mechanical system enhancements, including upgrades to the SCR and modulator tanks and the addition of a smoke alarm system for fire suppression. Despite this optimization of the physical machine, issues persist. [1]

Oak Ridge is currently limited to reactive intervention. While upgrades have reduced downtime, experimenters and engineers can mitigate issues only after they have already occurred, with significant temporal and financial repercussions. The Oak Ridge team believes that if preemptive monitoring and issue detection were implemented, downtime and its consequences could be further reduced. This paper will explore the potential of machine learning for maximizing uptime by preventing HVCM issues.

Several components of the HVCM architecture collect signal data during machine operation. Oak Ridge has long believed that this signal data could contain predictive power for preemptively detecting issues.

This project's two goals are exploratory in nature: to determine the feasibility of using HVCM signal data to predict issues, and to compare the efficacy of several model classes.

An ideal future implementation would constantly monitor signal data from the HVCM and determine the likelihood of impending issues, shutting down the machine when these were imminent. A good model would need to both identify imminent issues and avoid unnecessary machine shutdowns due to Type I "false positives." Additionally, such a model would require flexibility when encountering unseen states, necessitating some mechanism for addressing uncertainty. These business needs drove our decision-making throughout this exploratory analysis. To satisfy these requirements, our team explored four models: Gaussian Process Classification, Prototypical Network with Few Shot Learning (PN-FSL), Approximated Bayesian Deep Learning, and LSTM Autoencoders.

The format of our paper is as follows. First, we will give an overview of our data sources. Next, we will discuss the models we used and our specific implementations, as well as transformations used for feature extraction. From here, we will present and discuss our results. We will conclude with a section on the implications of our analysis and potential areas for further exploration.

II. Data Description

The Oak Ridge team provided us with 208 labeled signal observations – 158 from the "normal" class and 50 from the "fault" class, all coming from one module of the HVCM. For observations where a fault did occur, the metadata also describes the type of fault that occurred. These labels describe the state of the HVCM in the pulse immediately following the observation, in accordance with our goal of preemptive detection as illustrated in Figure 1. These observations consist of 6100 timestamp values for each of the 32 HVCM components, each taken at a rate of 1 sample every 400 ns.

III. Methodology

A. Prototypical Network with Few Shot Learning

The first method is an adaptation of the DPSN model developed by Wensi Tang et al, with several modifications. [2]

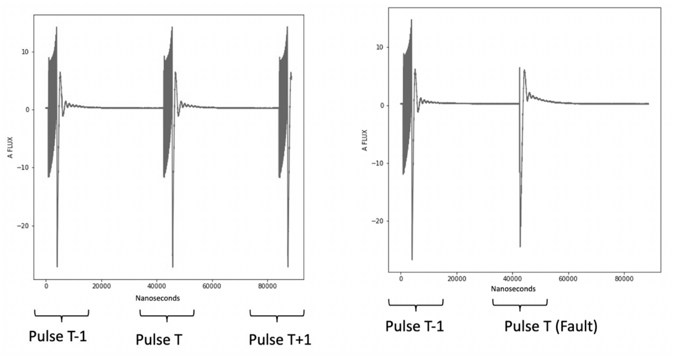

Fig. 1. Left: Example of a Normal Class Observation for a Single Component Right: Example of a Fault Class Observation for the Component

The method consists of three main steps: 1) a Symbolic Fourier Approximation (SFA), 2) a prototypical neural network trained using the few shot learning framework, and 3) a kernel density classification.

1) SFA: The SFA is a computationally efficient way to represent large high-dimensional data. [3] A moving window is applied to time series data, where each frame seen by the window is grouped together in a histogram with similar frames (based on the similarity of their Fourier coefficients). This histogram is converted to a vector for each parameter, which all together create one array for each observation. This array has a (19,8) shape for each observation. It is helpful to think of these arrays as then being placed into two buckets-one consisting of normal observations and another of faults.

2) Prototypical Neural Network: The neural network consists of three layers; a flattened layer, a dense hidden layer with 80 nodes and LeakyRelu activation, and a final output later of three nodes with LeakyRelu activation. The weights of the model are trained as follows: at each iteration, 10 normal observations and 10 fault observations are randomly sampled without replacement from the two buckets generated by the SFA step, called the support set. These are passed through the network, resulting in 3 activations in the final nodes which act as the coordinates of the support set in the learned space of the model. The vector average of the normal and faults observations are taken, resulting in both a normal and fault prototype. 25 new normal and fault observations are then passed through the network, called the query set. For each query set observation, a loss is calculated based on its Euclidean distance from its respective prototype. See Figure 2 for a representation of the learned space, including the prototypes learned from the support set and the loss calculated from the query set. This loss is then used to perform backpropagation on the network and the process repeats for a user-specified number of iterations. The intuition is that over many iterations, the model will learn the weights and biases where the prototypes will converge to a location in the learned space, where normal observations are typically near the normal prototype and fault observations are near the fault prototype.

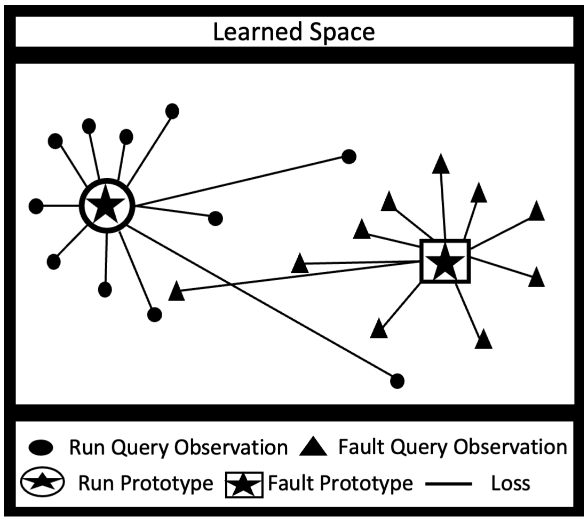

Fig. 2. PN-FSL Learned Space Loss Example

3) Kernel Density Classification: The final step is to perform a kernel density classification in the learned space of the network on hold out data. Kernel density estimation has been demonstrated as an effective non-parametric method to estimate the probability density of a random variable. [4] Hold out data is projected into the learned space, where a kernel density estimation is performed. A threshold learned during cross validation parameter tuning is then used, where any observation with a lower density than the threshold (indicating it is far from the center of the distribution) is classified as a fault.

Some additional enhancements could include mapping to a higher dimensional space, increasing the depth of the neural network, or replacing the SFA feature engineering step with an embedding using convolutional layers.

B. Gaussian Process Classification

1) Model Design: Gaussian Process Classification (GPC) is a supervised, non-parametric, Bayesian method for classification. [5] GPCs assume that all seen (model fitting) and unseen (model prediction) values of the target variable represent a single observation of an $N + M$ dimensional Gaussian, with N and M here representing the number of observations in the fitting and prediction set, respectively. The covariance matrix of this Gaussian is computed by using a kernel to calculate the similarity between each observation's input space. This structure produces a mean and variance for a posterior Gaussian for each prediction observation conditioned on the seen target observations used during fitting and the known covariance matrix. Our target variable was binary $\{normal, fault\}$, yielding results in the log-odds space, but this methodology could easily be extended to multi-class problems using a one-vs-rest approach.

2) Data Transformation: For the GPC model, our team performed feature extraction on the raw pulse data using a Fourier transformation. Fourier decomposition provides a one-to-one mapping from the time space to the frequency space by transforming a waveform into a summation of sinusoidal functions of various frequencies and intensities. [6] This process is commonly used on signal data in the machine learning field. This transformation yields a vector of frequency intensities for each component, resulting in an order-2 tensor for each observation. This scale was not only too large for a GPC in the practical sense, but also beyond the scope of the GPy package we used to run the model in python. [7] As such, we chose to predict fault likelihood separately for each HVCM component, producing 19 separate fault likelihood distributions which were then aggregated into a single distribution. To attain point estimates necessary for the ROC curve, we used the expected value of the aggregated distribution, which suited our problem better than the traditional MAP approach.

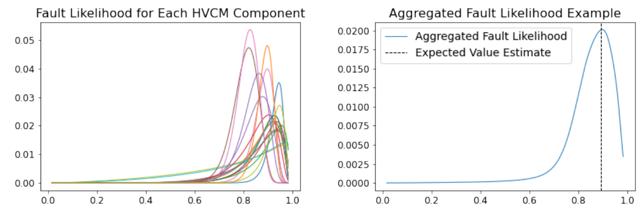

Fig. 3. Left: fault likelihood for individual components. Right: aggregated fault likelihood

3) Hyperparameter Selection: In GPCs, model tuning is done through kernel selection, which has two dimensions. The first is the choice of kernel itself, which determines how output similarity is calculated as a factor of input similarity. We opted for the Matern Kernel based on superior observed performance. [5] Due to the Gaussian assumptions of the model, GPCs contain a built-in method for optimizing the hyperparameters of the kernel, leaving kernel choice as the only manual piece of hyperparameter tuning.

C. Bayesian Neural Network Approximation

Bayesian neural networks are structurally similar to standard neural networks but use distributions for the weights between nodes as opposed to singular deterministic values. By doing so, Bayesian neural networks explicitly account for uncertainty in their predictions. While this uncertainty quantification makes them ideal for applications with small datasets, they can quickly become computationally burdensome as the data size grows. In recent years, Gal and Ghahramani have shown that a more computationally efficient class of models are deterministic models that use Monte Carlo Dropout during both training and inference time [8]. They have shown that these models approximate a traditional Bayesian neural network without the same scaling issues as the more traditional methods.

Given the time series nature of our data and the binary nature of our outcomes, we implemented a one-dimensional convolutional neural network with a Binary Cross Entropy loss function. Our model had four blocks of convolution and pooling that fed into a 30-node dense layer featuring 10% Monte Carlo Dropout. Each convolution and pooling block consisted of two layers of one-dimensional convolution with

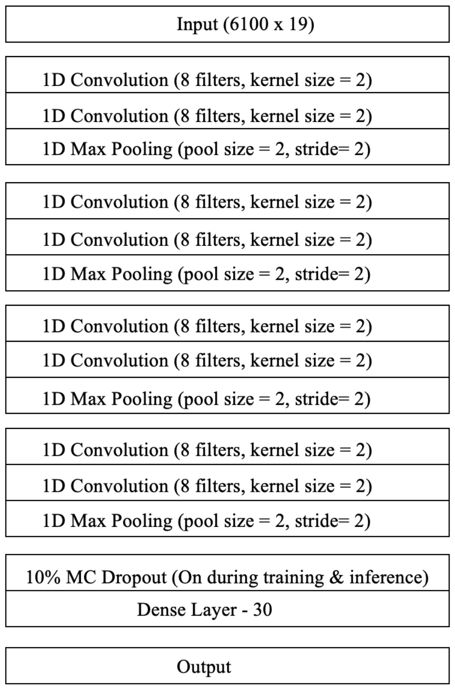
Input (6100 x 19)
1D Convolution (8 filters, kernel size = 2)
1D Convolution (8 filters, kernel size = 2)
1D Max Pooling (pool size = 2, stride= 2)
1D Convolution (8 filters, kernel size = 2)
1D Convolution (8 filters, kernel size = 2)
1D Max Pooling (pool size = 2, stride= 2)
1D Convolution (8 filters, kernel size = 2)
1D Convolution (8 filters, kernel size = 2)
1D Max Pooling (pool size = 2, stride= 2)
1D Convolution (8 filters, kernel size = 2)
1D Convolution (8 filters, kernel size = 2)
1D Max Pooling (pool size = 2, stride= 2)
10% MC Dropout (On during training & inference)
Dense Layer - 30
Output

Fig. 4. Approximate BNN Architecture

8 filters each, a kernel size of 2 and a Relu activation. Each block then contained a one-dimensional max pooling layer with a pool size of 2 and stride of 2. This architecture is illustrated in Figure 4.

D. LSTM Autoencoder

Autoencoders are a type of neural network that is made up of two sequential networks: an encoder and a decoder. Inputs are compressed through the encoding stage and reconstructed through the decoding stage such that the difference metric between the original input and the reconstruction is minimized. As An, Jinwon, and Sungzoon Cho discuss [9], autoencoders can be used for rare case classification in anomaly detection problems by training the model only on normal data and observing the differences between the reconstruction error of different input types. This framework allows a model to be trained and evaluated using the entirety of the dataset despite class imbalance. For the evaluation of this method, we compare the reconstruction mean square error (MSE) between the reconstructed signal and original input signal.

Given that signals are measured in the time domain, autoencoders are constructed with LSTM layers. The full architecture of the LSTM autoencoder is displayed below in Figure 5. Typical autoencoders are designed to reduce the size of the encoded output as much as possible. However, the performance of the autoencoder for anomaly detection depends on the reconstruction rather than the compressed embedding. Therefore, more layers of higher dimensionality were selected in the architecture. To prevent overfitting, two dropout layers were introduced in the encoding stage.

For each of the 19 selected features, a reconstruction MSE was computed from the autoencoder. These 19 reconstruction MSEs were used in a simple logistic regression model to output a probability of normal/fault, which was used for methodology evaluation.

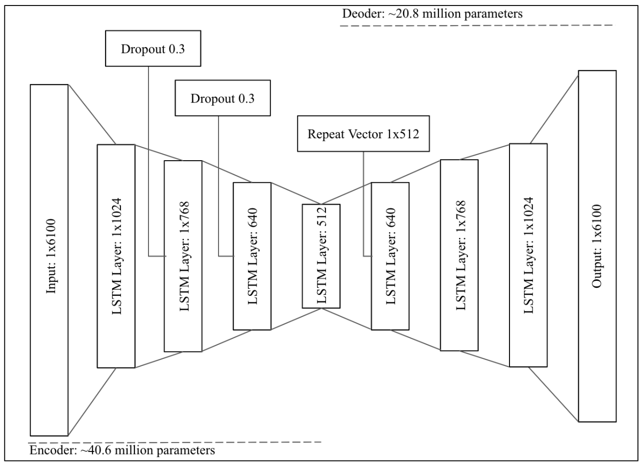

Fig. 5. LSTM Autoencoder Architecture

IV. Results

To evaluate theses methodologies, we performed 5-fold cross validation. Train and test predictions were recorded for each fold, then concatenated to produce a train and test ROC curve. The four ROC curves are displayed below in Figures 6-9. Due to the stochastic nature of predictions in the approximated BNN, 100 inferences were made on both the training and test data for each fold and then averaged.

V. Discussion

Since the overall goal of preemptive fault detection is to reduce downtime of the machine, it is more important to minimize false positive rates than it is to maximize true positive rates. Otherwise, shutting down the machine from false positives could inadvertently increase downtime. As such, we pay particular interest in the portion of the ROC curves where the false positive rate is close to zero. The two strongest performers in this regard are the GPC and the Autoencoder. However, there are several factors that would motivate the use of approximated BNN and PN-FSL methodologies, such as their ability to perform uncertainty estimation (like the GPC) and group similar signals, respectively. We will compare all four methodologies in terms of scalability, computational complexity, and interpretability.

A. Scalability

The number of normal and fault observations in our dataset is small compared to the frequency in which events occur in the SNS. In addition, the number of normal events far exceeds the number of fault events, so the implementation of one or more of these methodologies requires a discussion on how performance is expected to change with more data.

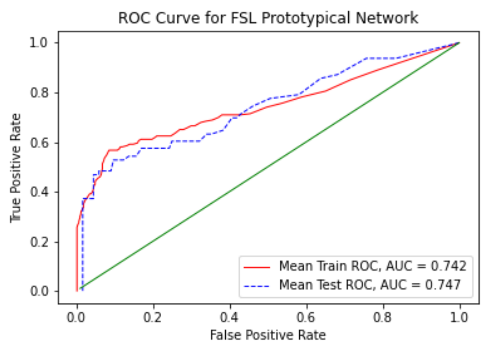

Fig. 6. PN-FSL ROC

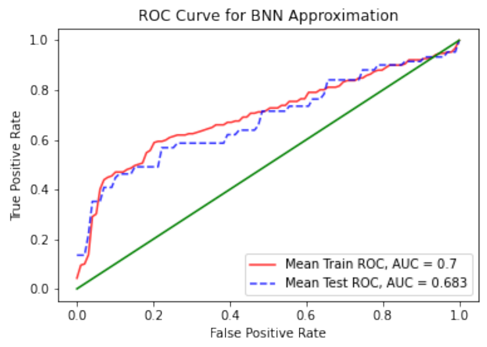

Fig. 7. Approximated BNN ROC

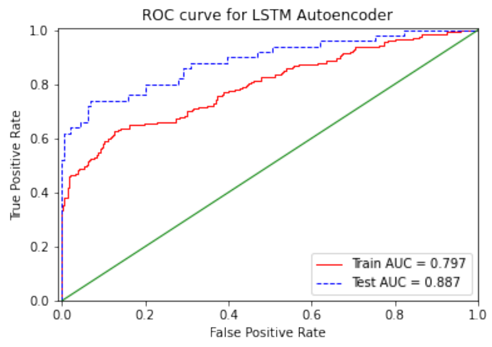

Fig. 8. Autoencoder ROC

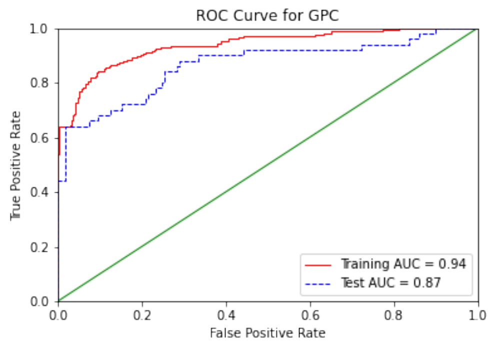

Fig. 9. GPC ROC

The Autoencoder and approximated BNN methodologies are particularly positioned to see a boost in performance with more data. In the case of the Autoencoder, more normal events should increase the robustness of the reconstructions, allowing for greater confidence in identifying highly anomalous events with low false positive rates in an imbalanced dataset. However, variation between normal waveforms over time from things like electrical component degradation or replacement in the HVCM could decrease robustness. In this case, periodic retraining on the most recent data is recommended. In a similar fashion, the approximated BNN is expected to increase in robustness with more data but could be at less risk of unwanted bias from component degradation because of its uncertainty consideration.

While the PN-FSL method is designed to perform well with small datasets, the framework could improve with more observations, which may allow clustering methods such as Gaussian mixture models (GMM) to be used. have been shown to be effective in other anomaly detection problems. [10] An ideal number of clusters could be learned from a parameter tuning process and GMM would provide a measurement of uncertainty. The implementation of GMM struggles to find meaningful clusters with our current dataset but could perform better with more data. Unlike the other methodologies, GPC is constrained by poor scalability, and requires a relatively small data set for effective fitting. Still, creating a balanced subset with more examples of rare fault types has potential to balance accuracy and performance.

B. Computational Complexity

Since time between events in the SNS is measured in microseconds, fast computation time is required in any implementation. More complex methods like the PN-FSL or the approximated BNN are expected to have a higher time to prediction than the Autoencoder. In the PN-FSL method, the SFA feature engineering step drives the high time to prediction. For the approximated BNN, the culprit is the multiple inferences required in the uncertainty estimation. GPC is at a risk of high time to prediction with a time complexity of $O(N^3)$, but appropriate sub-setting may lower prediction time while maintaining high accuracy. The lack of preprocessing and uncertainty estimation positions the Autoencoder to potentially have the lowest time to prediction in implementation.

C. Interpretability

An obvious extension to the binary classification problem is a multi-class classification for different types of faults. Furthermore, understanding how various groups of faults compare in relation to each other could be useful in problem diagnostics. This unsupervised approach to understand fault type clusters can be achieved with the PN-FSL method described above. Given that the output dimension space can be visualized if kept under four dimensions, there is a potential for clustering. Higher output dimensions can also be used and visualised with techniques such as t-SNE. [11] As discussed above, additional data and the introduction of GMM is a suitable candidate to identify clusters. Figure X shows an example of a learned space with four clusters (2 prominently normal clusters and 2 clusters where all the faults are of the same mechanical failure).

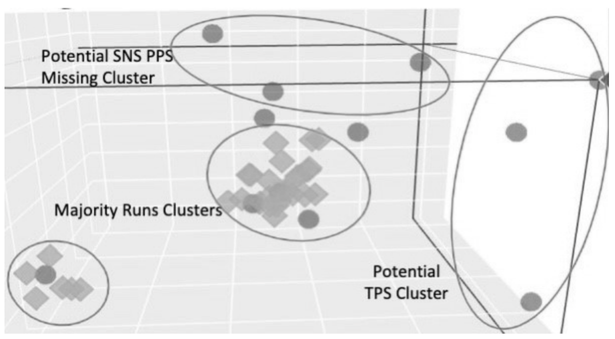

Fig. 10. PN-FSL Learned Space

The Autoencoder can also be expanded to include an output clustering using reconstruction MSEs to identify similarities between faults. Current SNS data retrieval allows for labeled fault types, which makes supervised learning possible, specifically in the approximated BNN and GPC. The approximated BNN can output a probability distribution with multiple inferences. The GPC methodology outputs normal/fault probability distributions for each prediction, which makes visual inspection of differences in distribution possible. A one-versus-rest model would be a strong extension to the GPC to address multiclass classification of fault types.

VI. Conclusion

Our initial results demonstrate that HVCM signal data contains predictive power for detecting imminent issues. For the narrow goal of preemptive fault detection with this limited dataset, the GPC and the LSTM Autoencoder had the initial best results, with the LSTM Autoencoder having an additional advantage of lower time complexity and the GPC having the advantage of uncertainty measurements. The PN-FSL and the approximated BNN are expected to see some improvement in performance with more observations.

All four methods can be expanded to the fault type classification problem, with the PN-FSL method having the advantage in terms of interpretability. The PN-FSL method can also be expanded to include GMM, which would also provide a measurement of uncertainty.

Acknowledgments

We would like to thank our sponsors, the Data Science team at Jefferson Laboratory, for their guidance and support throughout this project, especially Dr. Malachi Schram, Dr. Thomas Britton, and Dr. Lasitha Vidyaratne. We would also like to acknowledge Dr. Peter Alonzi, who helped us set up and manage our technical environment, Dr. Michael Porter and Dr. Stephen Baek, who gave us theoretical guidance. Finally, we would also like to thank our Capstone advisor, Dr. Alec Gosse, for overall support in every facet of the project.

REFERENCES

[1] Peplov, Vladimir V, Anderson, David E, Cutler, Roy I, Hicks, Jim, Saethre, Robert B, and Wezensky, Mark W. SNS LINAC MODULATOR OPERATIONAL HISTORY AND PERFOMANCE. United States: N. p., 2011. Web.

[2] Tang, Wensi, Lu Liu, and Guodong Long. ”Interpretable time-series classification on few-shot samples.” 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020.

[3] Schäfer, Patrick, and Mikael Högqvist. ”SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets.” Proceedings of the 15th international conference on extending database technology. 2012.

[4] Terrell, George R., and David W. Scott. ”Variable kernel density estimation.” The Annals of Statistics (1992): 1236-1265.

[5] Rasmussen, Carl Edward, et al. Gaussian Processes for Machine Learning. United Kingdom, MIT Press, 2006.

[6] Boashash, Boualem. Time Frequency Signal Analysis and Processing: A Comprehensive Reference. Elsevier, 2005.

[7] “GPy.” GPy by SheffieldML, https://sheffieldml.github.io/GPy/.

[8] Gal, Yarin, and Zoubin Ghahramani. ”Dropout as a bayesian approximation: Representing model uncertainty in deep learning.” international conference on machine learning. PMLR, 2016..

[9] An, Jinwon, and Sungzoon Cho. ”Variational autoencoder based anomaly detection using reconstruction probability.” Special Lecture on IE 2.1 (2015): 1-18.

[10] Zong, Bo, et al. ”Deep autoencoding gaussian mixture model for unsupervised anomaly detection.” International conference on learning representations. 2018.

[11] Van der Maaten, Laurens, and Geoffrey Hinton. ”Visualizing data using t-SNE.” Journal of machine learning research 9.11 (2008).

[12] G. Pappas, D. Lu, M. Schram, D. Vrabie, Machine learning for improved availability of the sns klystron high voltage converter modulators, Tech. rep., Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States); Thomas . . . (2021).

[13] Park, Daehyung, Yuuna Hoshi, and Charles C. Kemp. ”A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencode.” IEEE Robotics and Automation Letters 3.3 (2018): 1544-1551.

[14] Sharma, Balaram, Prabhat Pokharel, and Basanta Joshi. ”User behavior analytics for anomaly detection using LSTM autoencoder-insider threat detection.” Proceedings of the 11th International Conference on Advances in Information Technology. 2020.

[15] Radaideh, Majdi I., et al. ”Time Series Anomaly Detection in Power Electronics Signals with Recurrent and Convlstm Autoencoders.” Available at SSRN 4069225.