ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

DEVELOPMENT OF A SMART ALARM SYSTEM FOR THE CEBAF INJECTOR*

Dan Tyler Abell, Jonathan Edelen, RadiaSoft LLC, Boulder, Colorado, USA
Daniel Grady Moser, Brian Freeman, Reza Kazimi, Chris Tennant
Thomas Jefferson National Accelerator Facility, Newport News, Virginia, USA

Abstract

RadiaSoft and Jefferson Laboratory are working together to develop a machine-learning-based smart alarm system for the CEBAF injector. Because of the injector's large number of parameters and possible fault scenarios, it is highly desirable to have an autonomous alarm system that can quickly identify and diagnose unusual machine states. We present our work on artificial neural networks designed to identify such undesirable machine states. Our initial efforts have been focused on model prototyping and data curation. In this paper we present an overview of our initial findings and our efforts to generate a robust dataset for the alarm system. We conclude with a discussion of plans for future work.

INTRODUCTION

A significant aspect of accelerator operations involves identifying the root causes of faulty machine states. For example, the machine trips on a beam loss monitor. But why? What underlying cause trips the machine? Sometimes the reason is obvious, while other times not. Existing alarm systems commonly indicate when specific machine parameters drift outside their normal tolerances. However, operators must still interpret these alarms in the context of many interacting systems and subsystems before they can take the the most appropriate corrective action.

The project described in this paper has at its primary objective the development of a machine-learning-based model with the ability to rapidly identify potential root causes of machine faults (hence the term Smart Alarm). More specifically, given machine readings (defined more precisely later), the system continuously compares the model's predictions of the expected machine settings (also defined later) against actual machine settings. When a discrepancy arises that exceeds some user-defined threshold, the system raises an alarm that directs operators (or subject matter experts) to the "bad" setting (*e.g.*, corrector, solenoid, rf gradient, *etc.*). If this effort succeeds, a more ambitious goal will be to extend the work to monitor the machine for parameter drifts and identify when the machine needs "tweaking" before a fault event occurs.

During our initial efforts at training and validating machine learning (ML) models, we obtained puzzling and disappointing results. The problems at issue we traced back to various difficulties with our data, including some outlier (read nonsensical) vacuum readings. In the process of that investigation, we examined our data more carefully and found other aspects of data collection and selection that required careful consideration. The lesson to learn here is that *understanding one's data*—though time-consuming—*constitutes a critical part of any ML effort.*

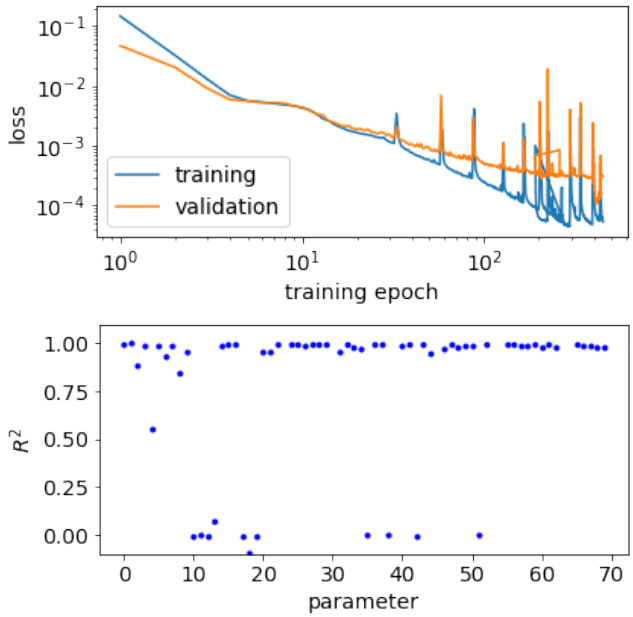

Figure 1: Inverse model trained on operations data from the JLab injector.

In the following sections, we describe briefly some of our early ML efforts and how they led us to make a thorough investigation of our data and the data collection process. We then describe our data, the collection process, and our evolving understanding of how best to curate data that will prove useful for the training and validation of future ML models. We conclude with our plans for future work.

INITIAL ML EFFORTS

Among our first efforts was training an inverse model on data taken from the JLab injector, with the goal of using measurements (readings) to predict machine settings. Figure 1 shows the loss functions and the somewhat disappointing R^2 fit results for one training run with this data. Modifications to the network architecture, number of epochs, choice of optimizer, *etc.*, made relatively little improvement on the overall results.

This circumstance led us to undertake a critical examination of our data, which at the time comprised that taken during (i) regular *operations* of the injector and (ii) a dedicated *study* of the machine. The various parameters (called

* Work supported, in part, by the US Department of Energy, Office of Science, Office of Nuclear Physics, including grant No. DE-SC0019682.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

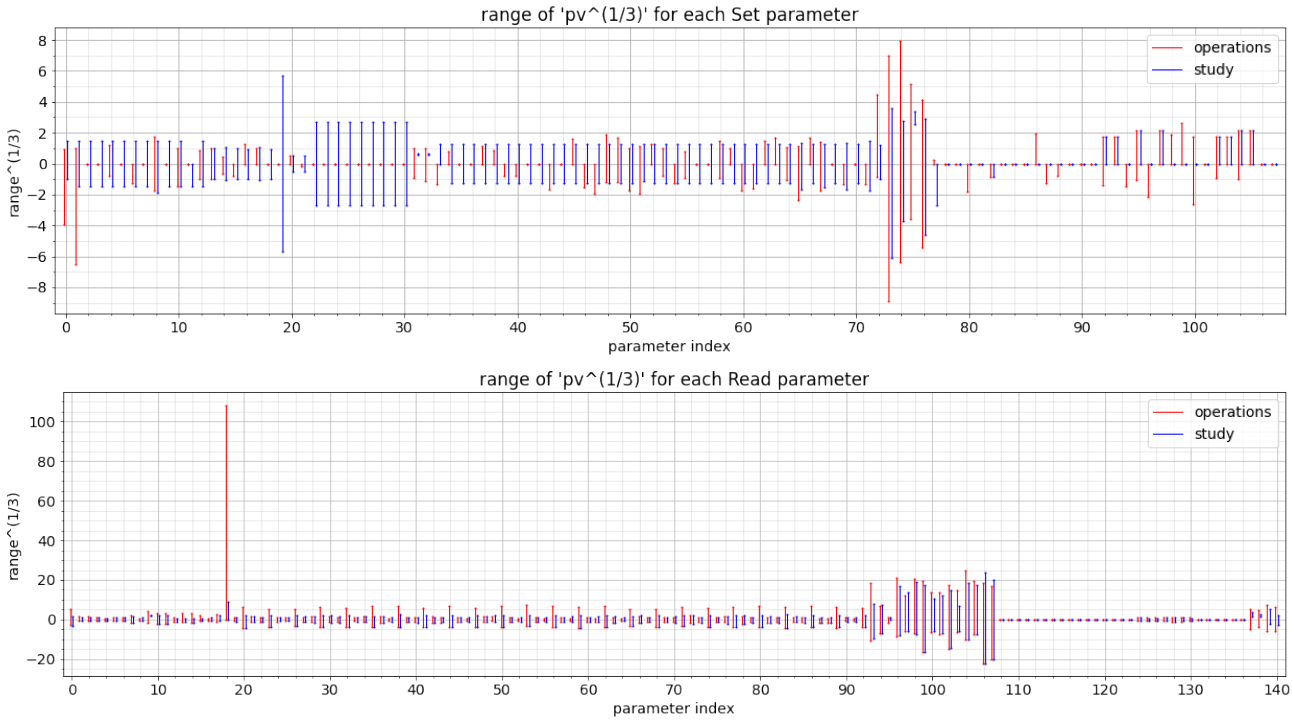

Figure 2: Summary of all Set (upper) and Read (lower) PV data. For each index, the red and blue vertical bars show the range of values spanned respectively by injector operations data and by injector study data. The vertical scales have been compressed by taking a cube-root. The medians of the operations data have been subtracted out.

Process Variables, or PVs) are divided into *Settings* and *Readings*. Figure 2 shows the ranges spanned by the data taken for each of the PVs, and in particular the overlap of the ranges between operations and study data. The hope was to use, say, operations data (of which we had much more) for training and validation, and the study data for testing. For that to work well, however, one would need to see that, say, the range of each *Study* PV lies within the range of its corresponding *Operations* PV—or vice versa. But, as shown by the graphics in Fig. 2, neither case consistently holds.

Another aspect of the data, not illustrated here, is that many of the Set PVs take on just a very few discrete values. This minimal variation suggests that it will, as we have found so far, prove difficult for ML networks trained on this data to achieve a useful characterization of the JLab injector. The solution, of course, is more data and, more importantly, better data.

A newer dataset from the JLab injector comprises some 409 500 instances averaged over one-second intervals for some seven weeks last summer. (See the next section for more details.) This data has no natural divide along which one might assign the different records into separate train and validation datasets; hence we attempted such a split randomly. In this case, too, we found some PVs for which the training range covers the validation range, and other PVs for which the reverse holds. This presents a challenge for training because there are ranges where the model would be trying to extrapolate to a new domain. This, a fundamental challenge for machine learning, and one of the challenges of model-based anomaly detection, means that a full understanding of the data is critical to success.

CURATING THE DATA

Five datasets were collected using several different methods. The variety of approaches reflects how our understanding of capturing machine data has evolved (and how it continues to do so). The results of this work will likely inform the modification of current—or the development of new—tools to capture and/or mine data more efficiently.

Collected Datasets

The datasets collected exist in tabular form, with each column representing an EPICS process variable (PV), and each row a particular snapshot in time. For example, collecting data for the three PVs `Date`, `R047GSET`, and `R047PSET` every hour, on the hour, for two days of operation would yield a dataset of size of 48 × 3 [= (2 days × 24 readings/day) × (3 PVs)] (as shown in Fig. 3). We also give the range of dates (to the nearest day) covered by each dataset below (listed in the order collected).

- **Dataset A** Data collected from dedicated beam studies in which select injector beamline components were systematically varied and downstream responses recorded. Date: 2020-09-08 (swing shift) and 2020-09-14 (swing shift). Size: 654 × 223.
- **Dataset B** First attempt at mining the operational archiver for "good beam". Date: 2020-08-30 to 2020-09-18. Size: 24 × 302.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

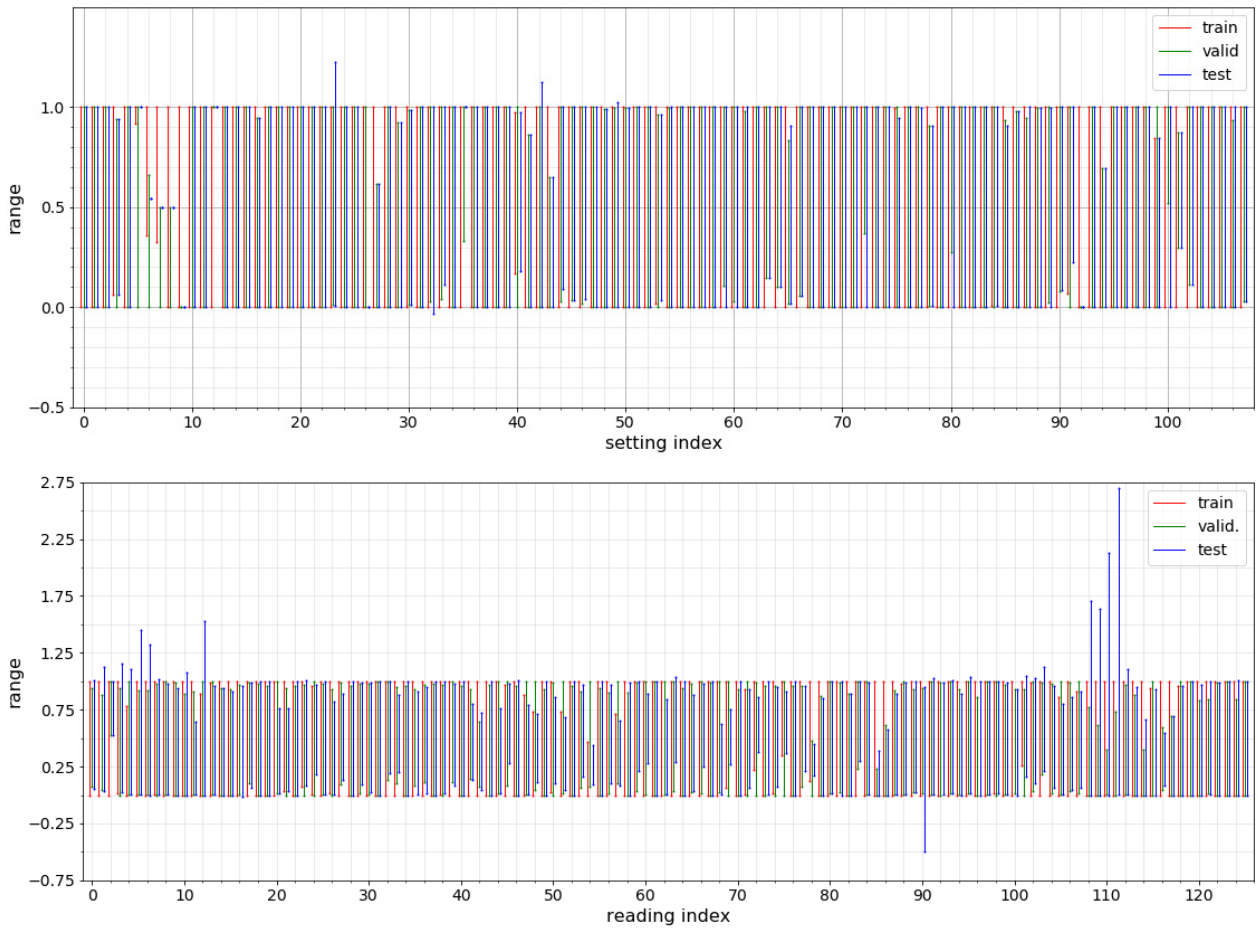

Figure 3: Ranges of the Scaled setting (upper) and Reading (lower) PVs, color coded by data split.

- **Dataset C** Mining the operational archiver for "good beam" (using less stringent constraints). Date: 2020-08-01 to 2020-09-21. Size: 5467 × 303.
- **Dataset D** Mining the operational archiver without imposing "good beam" constraints. Data was collected and averaged over each second of the Fall 2020 operational run. Date: 2020-08-01 to 2020-09-20. Size: 409 549 × 318. (This represents an initial filtering for beam-on conditions.)
- **Dataset E** Mining the historical archiver without imposing "good beam" constraints. Data was collected hourly for the last two years. Whereas the operational archiver stores machine data from the last several months, the historical archiver contains data from the previous few years. Date: 2019-02-01 to 2021-01-30. Size: 17 520 × 262. (The period from 2020-09-20 to 2021-01-30 represents a scheduled accelerator down period and contains no useful data.)

Several observations can be made. First, many of the datasets contain overlapping dates. Second, each dataset records a different number of PVs. Finally, there is a clear evolution in data collection strategy: It started with conventional beam studies efforts where knobs were varied and responses recorded (Dataset A). While this provided several hundred examples, both the complexity of the problem we are trying to solve and the means with which to solve it (deep learning) necessitates *much* more data. We then transitioned to mining the archiver for PV values. Initial efforts placed the burden on carefully designing constraints so as to ensure that we collected only PVs corresponding to "good beam". This resulted in several thousand examples (Dataset C). We then transitioned to a mode of a collecting data from the archiver at regular intervals (either hourly or every second) with no filtering. This shifted the burden of extracting the appropriate data for training models from the data collection effort to the data pre-processing step.

Cleaning the Data

Because the dates of Dataset D overlap those the three previous datasets (A, B, and C), we now focus only on Datasets D and E. Some initial "cleaning" of the data involved clerical matters—reformatting, renaming columns, combining and the like. The more significant matters included various filters applied to the data:

- Refine filtering from "beam on" to "beam being on" and transported to the end of the injector.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

- Remove those (few) columns that contain a large number of NaNs.
- If a column contains a missing value (*e.g.*, NaN), remove the associated row (across all columns).
- Some PVs take only integer values (*e.g.*, 0 or 1). Because of the averaging the entries occasionally have non-integer values, indicating a change of state. Remove those rows.

After combining the resulting versions of Datasets D and E, removing duplicate rows, and retaining only the common set of PVs (there are 235), we arrive at a dataset of size $406\,397 \times 235$.

Partitioning the Data

Regardless of neural network architecture, one needs a well-defined set of inputs that the model learns to map to a well-defined set of outputs. Our next step here involves partitioning the 234 PVs (we exclude the Date column) into appropriate Setting and Reading datasets. At first, we define a *Setting* as any PV that an operator can adjust during routine beam tuning. These include phase and gradient set-points of radio-frequency (rf) cavities, solenoid strengths, corrector strengths, and the like. A *Reading*, on the other hand, we characterize as read-backs of various diagnostic systems. These include readings from beam loss monitors (BLMs), beam position monitors (BPMs), vacuum signals, beam current monitors (BCMs), and statistical descriptions of the beam as extracted from a synchrotron light monitor (SLM) image. In the end, we identify 108 Setting and 126 Reading PVs.

Executing a partition, however, proves less obvious than it sounds. Consider the beam current monitors: Should they count as Readings, as seems obvious, or as Settings? The BPM wire-sum signals, clearly Readings, act as a proxy for the beam current, which would seem to be a Setting. Depending on the details of a given ML architecture, we might want the model to learn the BCM current readings from BPM wire sums, or learn the wire sums from the BCM readings. In either of these cases, we shall have to reclassify some PVs from Readings to Settings. Or perhaps we ought to add laser power PVs as Settings, so as to capture an alternate proxy for the beam current treated as a Setting.

In the course of this review of PVs, we identified other issues: Several PVs do not appear in the data, including settings for one of the rf cavities and readings from several vacuum PVs. In addition, solenoids and correctors appear in the data, but quadrupoles do not. With regard to excluding he quadrupole PVs, the initial thinking was that quadrupoles do not often change value, and static (or nearly static) training data contains very little not serve to teach a given ML model. Whether to include then in the future requires further consideration.

For future data collection efforts, each injector PV need to be examined and a decision made as to whether or not to include it in the dataset for addressing the problem at hand. Moreover, those PVs should be explicitly defined at the outset, so as to ensure that all subsequent data collection efforts include a common, consistent set of PVs.

FUTURE WORK

Two primary challenges underlie the construction of a Smart Alarm system. The first is feature selection/down-selection, and the second is choosing an appropriate model. We will explore the use of different feature selection tools to develop reduced representations of given datasets. In addition to auto-encoders,we plan to develop both linear and nonlinear classifiers, including the possible use of neural networks and decision trees as nonlinear classifiers. We will compare the performance of different classifiers trained using supervised learning and establish some general heuristics for fault detection on this type of injector. We will begin by identifying a few simple fault scenarios, and then build on this to include a range of different faults, including those caused by coupled errors within the machine. We will also explore the use of unsupervised learning to train classifiers. Algorithms such as DBSCAN, gaussian mixture modeling, and agglomerative clustering are well suited for clustering a variety of distinct types of data. Once trained, those models can track the vicinity of a given machine state to the locations of distinct clusters determined from training.

CONCLUSION

Data provides the fuel for machine learning. In the context of the CEBAF injector, we are still learning how best to collect and curate the appropriate data for training our models. The archiver provides a rich—though largely underutilized—source of data. Using the toolkit of data science, we believe there is potential for a deeper understanding of machine performance from data that has already been collected.

ACKNOWLEDGMENTS

This work is supported, in part, by the US Department of Energy, Office of Science, Office of Nuclear Physics, including grant No. DE-SC0019682.