Investigating Failure Patterns in Particle Accelerator Infrastructures with Explainable Deep Learning

Thomas Cartier-Michaud, Lukas Felsberger, Andrea Apollonio, Andreas Müller, Benjamin Todd, Dieter Kranzlmüller
Structure

Introduction

Methodology

Experiments and Results

Outlook
Introduction

• Large infrastructures hard to diagnose
• ML approaches scale to very high dimensionality
• Can they be useful to help operators?
**Introduction - Idea**

<table>
<thead>
<tr>
<th>Data</th>
<th>Data Driven Model Prediction</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input (image of animal)</td>
<td>Cock</td>
<td>![Image of animal]</td>
</tr>
<tr>
<td>Label (species)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hammerhead</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hare</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input (past monitoring signals)</th>
<th>Label (leading to alarm in future?)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>![Time ↑ Signals]</td>
<td>No</td>
<td>![Image of past monitoring signals]</td>
</tr>
<tr>
<td>![Time ↑ Signals]</td>
<td>Yes</td>
<td>![Image of past monitoring signals]</td>
</tr>
<tr>
<td>![Time ↑ Signals]</td>
<td>No</td>
<td>![Image of past monitoring signals]</td>
</tr>
</tbody>
</table>
Introduction - Challenges

Image recognition
- Large data sets
- Ground truth usually known
- Explanation easy to interpret

Failure pattern mining
- Small data sets
- Ground truth hard to come by
- Explanation interpretable?
Methodology

Adapting well established machine learning approach:

<table>
<thead>
<tr>
<th>Steps</th>
<th>Data Collection</th>
<th>ML Problem Formulation</th>
<th>Filtering &amp; Subsampling</th>
<th>Algorithm Selection</th>
<th>Training</th>
<th>Model Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>LASER (SQL), CALS (pytimber)</td>
<td>Python</td>
<td>Python</td>
<td>Literature research</td>
<td>Sklearn, keras, tensorflow</td>
<td>Expert discussions</td>
</tr>
<tr>
<td>Duration (indicative)</td>
<td>3 months</td>
<td>1 month</td>
<td>1 week</td>
<td>1 week</td>
<td>2 months</td>
<td>In progress</td>
</tr>
</tbody>
</table>

3/11/2020

lukas.felsberger@cern.ch
Data Collection: LASER

- Centralized service capturing/notifying/storing anomalies for the whole accelerator chain + TI
- Alarms are raised for operators ➔ not an interlock system ➔ need for human (slow) intervention
- 30 fields, of which important ones are:
  - FAULT_FAMILY/_MEMBER_/CODE = pointer to the component and the fault
  - PRIORITY = severity of the fault = 0, 1, 2, 3 ➔ we predict priority 3 alarms, supervised problem
  - SYSTEM_TS = time stamp = events are recorded, no continuous signals
- For PSB (same building / components of the machine): bug investigation for 2018

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># lines</td>
<td>235 M</td>
<td>305 K</td>
<td>473 K</td>
<td>235 K</td>
<td>1 767 K</td>
<td>12 K</td>
<td>1 054 K</td>
<td>504 K</td>
</tr>
</tbody>
</table>

LASER description

CERN ALARMS DATA MANAGEMENT: STATE AND IMPROVEMENTS
Data Collection: extraction from LASER

- Accessing the TN using a Virtual Machine (~1 month)

- Retrieving data from SQL database (~2 months: 1.7TB)
  ➔ automation using bash scripts in parallel sessions
  ➔ process “artisanal” as LASER is not meant for such large requests

- Storing the data using a cernbox account (<50GB once zipped)
Data Collection: extraction from CALS

• Additional signals fetched from CALS based on expert recommendation

• Using pytimmer
  • Maximal data size per request limited
    • Requires splitting of requests and subsequent merging
Formulation of a Supervised Machine Learning Problem

Supervised ML needs:

• Data
  • Input
  • Label/Output
• Model linking in- and outputs
  • Model structure
  • Parameter optimization

We have:

• Alarms in time
  • Alarms in the past
  • Priority 3 alarm in the future
• Model linking past and future alarms
  • Choice of ML models
  • Choice of optimizers
ML Problem Formulation: existence of a model

Failure Modes

<table>
<thead>
<tr>
<th>Observables</th>
<th>non-Observables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{o,F}$</td>
<td>$S_{no,F}$</td>
</tr>
<tr>
<td>$S_{o,N}$</td>
<td>$S_{no,N}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$S_{o,1}$</td>
<td>$S_{no,1}$</td>
</tr>
</tbody>
</table>

$S_{o,F}[t,t+dt] \approx \Gamma(S_{o,(-\infty,t]})$

$S_{o,F}[t,t+dt] = \Gamma(S_{o,(-\infty,t]}, S_{no,(-\infty,t]}$
ML Problem Formulation: Discretization

Failure Modes

\[ S_{0,F}[t+pt, t+pt+2\cdot\delta t] \approx \Phi([S_{0,t-n\cdot\delta t}, \ldots, S_{0,t-\delta t}]) \]

\[ S_{0,F}[t+pt, t+pt+no\cdot\delta t] \]

\( \delta t \): bin width

\( n_i \): size of input window

\( no \): size of output window

\( pt \): prediction time

Transformation similar to L. Serio et al. allowing to compare with their approach of “Association Rules Mining”

see presentation
Filtering and Subsampling

- Focus on EPC of PSB ➔ reducing the number inputs
- Failures are rare (< 30) ➔
  - Filter out signals with too low or without activity
  - balancing class 0 (no failure) and class 1 (failure) elements by subsampling class 0
  - forcing contrast
Algorithm Selection

• Goal was to test explainable deep learning
  • But will be compared against “traditional” ML algorithms

Traditional ML
(based on popular choice for universal learners):
• SVM with linear kernel
• Random Forest
• K Nearest Neighbour

Deep Learning
(based on latest review papers for time series problems):
• Fully Convolutional Network
• Fully Convolutional Networks with Dropout Regularization
• time-Convolutional Neural Network

References in paper
Training: Implementation

- python3 + 2 main libraries:
  - Learning: [https://github.com/hfawaz/dl-4-tsc](https://github.com/hfawaz/dl-4-tsc)
  - Explaining: [https://github.com/albermax/innvestigate](https://github.com/albermax/innvestigate)
  - Implemented in keras and tensorflow
Training: Computation

• Not computationally intensive but many trainings (~100 000) for different meta parameters
  ➔ No use of GPU as reading / transforming / writing would have been the bottleneck
• Generation of thousands of jobs using 1 core each and executed in parallel on CERN Cluster
  ➔ 220 000 cores in the cluster, usage of up to 1000 cores at once
• Limit of 2GB/core the cluster
  ➔ Necessity to rewrite the transformation algorithm
• Limit of 100GB in AFS/work
  ➔ Easily reached with more than 100 000 combinations of hyper parameters * 1MB

We underestimated the input/output/postprocessing importance for large scans
Model Selection

For every choice of training algorithm + parameterization:

- Learning data
- Validation fold 1
- Train fold 1
- Validation fold 1
- Test error
- Calculate test error
- Calculate mean and standard deviation of validation error
- Testing data
- Validation fold 2
- Train fold 2
- Validation fold 2
- :
- Validation fold K
- Train fold K
- Validation fold K
Model Selection – (Hyper) Parameters

Steps
- Data Collection
- ML Problem Formulation
- Filtering & Subsampling
- Algorithm Selection
- Training
- Model Selection

# Hyper-Parameters
- 1
- 4
- 4
- 2
- 3

# ML Parameters
- <10^6

Parameter Cardinality
- 20
- 4^4
- 2
- 10
- Infinite
- 4
Experiments and Results

Goal: Use framework to predict and explain accelerator failures.

Using new framework on new problems requires iterative approach:

1. Verify and test new framework using synthetic data with known ground truth
2. Attack new problem (predict+explain faults in PSB) with verified framework
Verify and test new framework using synthetic data with known ground truth:
Synthetic Data Experiment

F1 = 2 * (Precision * Sensitivity) / (Precision + Sensitivity)
Sensitivity (Recall) = TP / (FN + TP)
Precision = TP / (TP + FP)

Accuracy = (TP + TN) / All Predictions

Synthetic Data Experiment - Discussion

✓ Learned predictive models from less than 10 training examples for up to 64 noise channels

✓ Framework works in principle
Real Data Experiments

- Test new problem (predict+explain faults in PSB) with validated framework

- Data
  - LASER alarms for power converters in PSB
  - CALS logging
    - External condition signals
    - PSB beam destination signal

lukas.felsberger@cern.ch
Mixing Synthetic and Real Data

- Preliminary step: Redo previous experiment and replace noise by real data
- Idea: if there was a well defined pattern in PSB data, would we detect it?
Mixing Synthetic and Real Data

- Learned predictive models from less than 10 training examples and for 43 PSB signals (as noise)
- Discovers correct synthetic pattern:

![Graph showing comparisons between FCN and SVM models](image)

- **Input**
  - Time [days]: 4, 3, 2, 1, 0
  - Other alarms
  - Relevant precursor

- **FCN**
  - Time [days]: 4, 3, 2, 1, 0
  - Alarm type

- **SVM**
  - Time [days]: 4, 3, 2, 1, 0
  - Alarm type
Mixing Synthetic and Real Data - Discussion

✓ Should find patterns in PSB data if there are

✓ Learned predictive model from less than 10 training examples for 43 PSB signals (as noise)

✓ Finds correct failure precursors
Real data

• Predict high priority alarms in LASER

• Example
  • Converter: BR3.DVT13L4 (ACAPULCO)
  • Fault code: 20
  • PC Permit not present

• Trained on data from 2015-09-01 to 2016-09-01; tested from 2016-09-12 to 2017-05-31

• Question:
  • Does it predict?
  • Does it explain?
Real data

Does it predict?

• Yes, when there are patterns
• But: there don’t seem to be many patterns
• And: We are at small data limit of machine learning → performance estimations uncertain

<table>
<thead>
<tr>
<th>score</th>
<th># alarms Test</th>
<th># alarms Train</th>
<th>fcn</th>
<th>fcn_3d</th>
<th>fcn_2d</th>
<th>kNN</th>
<th>random_forest</th>
<th>svm</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc_test</td>
<td>3</td>
<td>7</td>
<td>0.96</td>
<td>0.96</td>
<td>1</td>
<td>0.92</td>
<td>0.84</td>
<td>0.88</td>
</tr>
<tr>
<td>acc_val</td>
<td>2.4</td>
<td>4.6</td>
<td>0.84</td>
<td>0.90</td>
<td>0.93</td>
<td>0.82</td>
<td>0.68</td>
<td>0.73</td>
</tr>
<tr>
<td>F1_test</td>
<td>3</td>
<td>7</td>
<td>0.8</td>
<td>0.8</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F1_val</td>
<td>2.4</td>
<td>4.6</td>
<td>0.27</td>
<td>0.4</td>
<td>0.68</td>
<td>0</td>
<td>0.073</td>
<td>0.29</td>
</tr>
</tbody>
</table>
Real data

Does it explain?

- It gives hints
- Complex to interpret and ambiguous
Real data - Discussion

- Achieves good predictive performance when patterns exist
- Learned patterns are hard to interpret
- Interpretation when combined with logbook data easier
- Too little data to draw conclusions with certainty
  - Approaching limits for machine learning
  - Improve data selection/pre-processing/problem formulation
Conclusions and Outlook
Conclusions - Computation

- Underestimation of
  - Effort required to download the data
  - Execution time apart from training

- Benefitted from
  - Usage of well known libraries
  - Computation using CERN cluster

- Useful tools
  - Swan notebooks to prototype and share scripts (but terrible for debugging)
  - CERN cluster to do massively parallel computation
  - Mattermost to communicate (2 of us in Meyrin + 1 in Prévessin)
  - cernbox to share data
  - GitLab to share code: https://gitlab.cern.ch/tcartier/mlcern
Conclusions - Results

- Framework predicted and explained failure patterns correctly for generated test cases
  - Detected patterns from fewer than 10 training examples within up to $10^2$ signals
- For PSB data experiments
  - Predictions were accurate if well defined patterns existed
  - Explanations were hard to interpret for studied cases
    - Could be improved by different choice of input/output data
- Deep learning approaches outperformed traditional ML in presented cases
  - Random Forest outperformed everything else in traffic jam prediction problem (check paper)
Outlook

- Will implement a more data effective representation for learning
- Synthetic data: More complex patterns to better study failure mechanism explanation (e.g. Boolean logic between signals)
- Real data: Reformulate prediction problem and find less complex application scenarios with the goal of
  - Having more examples to learn from
  - Learning models of more controlled systems
- Framework is modular and re-usable
  - Clone our code and investigate how air traffic explains spread of Corona
Thank you for your attention!

Further details in:

• Publication

• Code
  • https://github.com/lfelsber/alarmsMining (public)
  • https://gitlab.cern.ch/tcartier/mlcern (CERN)
Training

- Concept of machine learning: Train on observed data, apply to unseen data
- Have to find best predictive model by systematic search over
  - Input data selection
  - Algorithm selection
  - Problem parameter selection
- \(\rightarrow\) requires (more than a bit of) computation