REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
08/05/2019	Final	01-09-2017 - 08-05-2019

4. TITLE AND SUBTITLE
Feasibility of Diagnostics, Prognostics and Hybrid Prognostics across Multiple Platforms

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00014-17-1-2726

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Thurston, Michael, G; McConky, Sean, P; Valant, Chris, J; Nenadic, Nenad, G

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ROCHESTER INSTITUTE OF TECHNOLOGY
1 LOMB MEMORIAL DRIVE
ROCHESTER, NY 14623-5603

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
875 North Randolph Street
Arlington, VA 22203-1995

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; distribution is Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Condition Based Maintenance Plus (CBM+) is mandated for all vehicles across the Department of Defense. This program focused on the ability to apply Prognostics Health Management (PHM), a subset of CBM+ capability, across platforms in different branches of the military. RIT evaluated Health and Usage Monitoring System (HUMS) data and maintenance data to identify immediate opportunities for PHM. Autoencoders, Neural Networks, and system-level models were evaluated for their ability to enable anomaly detection and identify condition indicators for prognostics. RIT also researched the current state of PHM in multiple fields to identify the best practices and trends that need to be considered for the future of CBM+.

15. SUBJECT TERMS
CBM, CBM+, Prognostics Health Management, PHM, Health and Usage Monitoring, HUMS, Autoencoder, Convolutional Neural Network, 1D CNN, System-level models, Condition Indicators, Anomaly Detection, Diagnostics, Prognostics, Digital Twin

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Michael G. Thurston
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	SAR	188	585-475-6550

RIT

Rochester Institute of Technology
Golisano Institute for Sustainability
190 Lomb Memorial Drive
Rochester, NY 14623
585-475-4696
http://www.rit.edu/gis/

May 8, 2019

Billy Short
ONR Code 301
ONR Asymmetrical Warfare Research Division
875 North Randolph Street
Arlington, VA 22203-1995

RE: N00014-17-1-2726 Final Report

Dear Mr. Short:

The Golisano Institute for Sustainability at Rochester Institute of Technology is pleased to submit the Final Technical Report with SF298 for the "Feasibility of Diagnostics, Prognostics and Hybrid Prognostics across Multiple Platforms" project under award number N00014-17-1-2726.

If you have any technical questions regarding this report, the Principle Investigator, Dr. Michael Thurston, can be reached at (585) 475-6550 or via email at mgtasp@rit.edu. For administrative or contracting inquiries, I can be reached at (585) 475-4325 or kxkasp@rit.edu.

Sincerely,

Kathleen Kosciolek

Kathleen Kosciolek
Assistant Director for Program Administration

cc: Administrative Office - ONR REG BOSTON (transmittal letter only)
Defense Technical Information Center
Naval Research Laboratory
File

Enclosures: RIT N00014-17-1-2726 Final Technical Report with SF298

RIT | Golisano Institute for Sustainability

Feasibility of Diagnostics, Prognostics, and Hybrid Prognostics across Multiple Platforms

Final Report

This research was conducted under Office of Naval Research Grant N00014-17-1-2726.

Report Generated by:

Golisano Institute for Sustainability
Rochester Institute of Technology
190 Lomb Memorial Drive
Rochester, NY 14623
(585) 475-5101

Table of Contents

1 Introduction

Maintenance practices within the military have historically relied on two main practices, failure replacements, and schedule or usage based preventative maintenance replacements. However, the Department of Defense (DoD) mandated Condition Based Maintenance (CBM) with DoD Instruction 4151.22[1]. These instructions indicate that CBM+ shall be used as the principal consideration for selection of proper maintenance concepts. The implementation of CBM measures in accordance with the policy is required of the program managers through requirements of the Secretaries of the Military Departments and the Directors of the Defense Agencies. The implementation of the program should be in accordance with the Condition Based Maintenance Plus DoD Guidebook [2].

The guidebook breaks down maintenance into reactive and proactive. Reactive maintenance is performed on items that are run to failure. Proactive maintenance is broken down further into preventative/scheduled maintenance or predictive maintenance. Preventative maintenance is either based on a schedule or based on a trigger that may lead to failure (e.g. a visual oil leak). Predictive maintenance is either diagnostic or prognostic. Diagnostic identifies impending failure and prognostic adds a prediction of remaining useful life. The DoD intends to implement CBM+ to reduce maintenance costs and improve the readiness of their assets. Figure 1 provides a visual representation of the planned DoD transition.

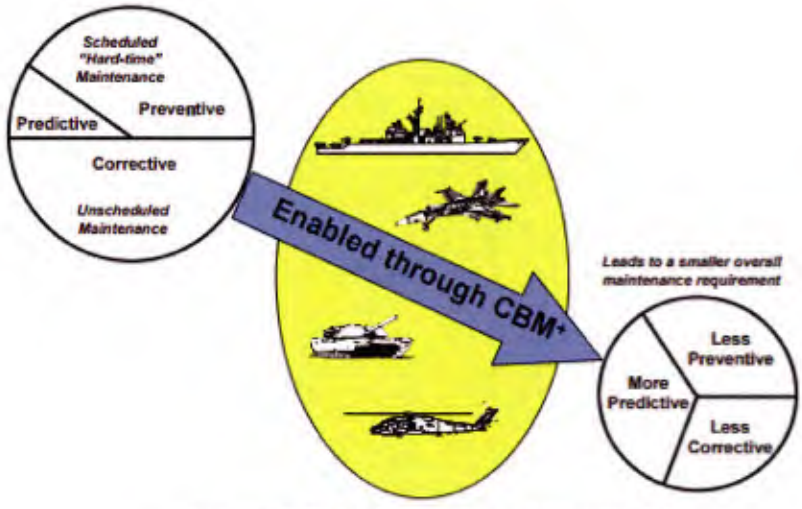


Figure 1 - DoD Maintenance Strategy Transition[2]

A complete CBM system is comprised of seven different layers of system functionality, including: sensing, signal processing, condition monitoring, health assessment, prognostics, decision support, and presentation [3]. The DoD guidebook breaks CBM down into eight infrastructure areas consisting of sensors, data management, condition monitoring, health assessment, analytics, decision support, human interfaces, and communications[2]. The analytics component, typically referred to as prognostics health management (PHM), comprises threshold alarms and alerts, diagnostic assessments, predictive assessments, trend analysis and prognostics assessment, which were the focus of this project.

From the viewpoint of need, the selection of systems for PHM is determined by considering frequency of failure (f) and severity (s), as depicted in Figure 2. According to this diagram, the cost of PHM development can be justified for systems with failures that exhibit high severity (criticality) with low frequency. From the viewpoint of feasibility, the selection must consider the nature of failures. For example, failures that

exhibit exponential failure distributions are not good candidates for PHM and are not considered viable for forecasting [4].

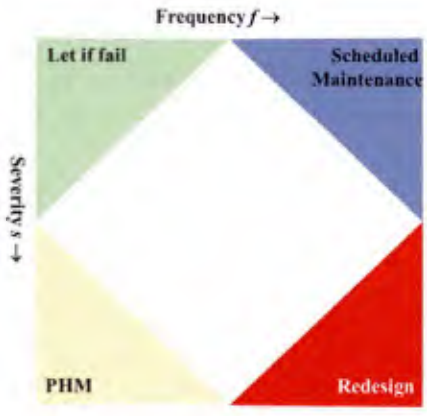


Figure 2: Maintenance strategies based on frequency and severity of failures used in automotive industry.

PHM can be broken down into three types: physics-based, data-driven, and hybrid (which integrates elements of data-driven and physics based models) [5]. Physics-based models typically require extensive data collection and the complexity of the models increases significantly with system complexity and the number of system interfaces. Data-driven PHM is not an all-or-nothing proposition, it can start with basic capabilities and gradually, over time, become more sophisticated [6, 7]. Due to the cost of up-front physics-based model development, data-driven approaches can provide a more cost effective initial implementation option as long as there is opportunity to add more sophisticated models as the system matures. Hybrid models utilize physics-based models and may augment results with data-driven models. Additionally, data-driven approaches may be used where physics-based approaches are too expensive or difficult to implement.

The first layer of capability within a PHM system is anomaly detection, which may not require much of domain-specific knowledge at the outset [8][1]. The system can advance to the next layer of capability, diagnostics, by learning to classify the observed anomalies into different failures modes. In parallel, PHM can learn to forecast remaining useful life (RUL), the prognostic layer of capability, by learning from captured time-domain data from the point of time when the anomaly was first detected to the point of the final failure. In their seminal papers, Engel et al. [9] examined issues associated with RUL, while Saxena et al. [10] discussed metrics for evaluating PHM performance. Uckun et al. [11] compared the methods employed in PHM to those used in medicine and proposed next steps toward maturation of PHM as a field of research.

A more recent concept, the digital twin, has been gaining interest in the PHM community. Glaessgen and Stargel provide an overview of the vision for the digital twin from the perspective of NASA and the U.S. Air Force[12]. This vision includes the use of complete as-built data on assets (such as material properties and extensive component geometric data), sophisticated multi-scale models of system and material performance, and extensive histories of operational data (including usage data, performance and health indicators, and maintenance events) to build a highly detailed and accurate digital model that can predict the performance and the reliability of an individual asset based on its own service history. They group the capabilities of the digital twin model into four categories: high fidelity modeling and simulation, design and certification, situational awareness, and life prediction and extension. A major goal of the NASA and Air Force version of the digital twin is the ability to maintain very high operational reliability of assets, while carefully reducing design margins and heuristic design constraints (such as factors of safety) that can accumulate and contribute to over-designed and over-weight air systems.

Condition Based Maintenance is pushing the need for advancements in prognostics, ranging from simple data driven models to the much more complex digital twins. Many CBM programs are focused on a single system, platform, or vehicle. The ability to apply common monitoring and prognostics approaches on

[1] Specifically, refer to Axiom III , which states that "Identifying the existence and location of damage can be done in an unsupervised learning mode, but identifying the type of damage present and the damage severity can generally only be done in a supervised learning mode"

different types of assets used by different military services will enable a reduction in development and implementation costs.

2 Project Objectives

Under this research effort, three platforms were selected (one each from the Army, Navy, and Marine Corps) for evaluation of cross-platforms PHM opportunities. Through the analysis of these platforms and additional background research, these major objectives were addressed:

1. Identify immediate opportunities across multiple platforms for extracting higher-level information content, including PHM capabilities
2. Evaluate the data quality and completeness across multiple platforms. Data quality includes assessment of missing values, temporal and value resolution, noise, consistency among different sources, as well as identifying missing data elements, such certain contextual information on environmental conditions.
3. Identify the best practices and emerging trends employed in prognostics, across various application spaces, which may be applied to assets of differing requirements, architecture, and environments.
4. Make specific recommendations for a prognostic development roadmap across the different platforms of study. Include specific recommendations for technical approaches including math models (Bayesian, deep learning, neural nets, etc.) and modeling and simulation approaches (potential applicability of digital twin).

3 Summary of Project and Report Overview

For this project, RIT evaluated the current CBM+ capabilities provided by existing systems on platforms across the Army, Marine Corps (USMC) and Navy. This evaluation included an analysis of what health and usage monitoring system (HUMS) data was collected and how it was stored, and an analysis of the maintenance data to identify areas of opportunity. Findings from these analyses were used as the basis for a review of immediate opportunities. The remainder of the project focused on best practices and emerging trends in the area of prognostics health management (PHM) and providing guidance for the future of CBM+, presented as a Roadmap. A summary of the research conducted and high-level results for each of these project components is briefly explained below, with additional detail provided in subsequent sections of the report.

The platforms evaluated included the Army Family of Medium Tactical Vehicles (FMTV), the Marine Corps Medium Tactical Vehicle Replacement (MTVR), and the Navy Landing Platform Dock (LPD).

Cross-platform HUMS and data storage analysis

The evaluation of HUMS identified that the FMTV and MTVR both utilized data acquisition systems that collected available databus data, which is defined by the Society of Automotive Engineers J1939 and J1708 data specifications. For the Army FMTV CBM system, data was collected at a 1Hz frequency. On the MTVR, the data was stored as J1939 captures at their reporting rate. The Navy LPD utilized a completely different system with a more extensive signal list, but the data rate varied from around once per second up to 15 or more minutes.

Additionally, the data storage formats were different. The FMTV data was stored in the Army Bulk CBM Data (ABCD) format. The MTVR data was J1939 packet captures saved into files. These files were also parsed into Matlab files for easy processing. The LPD data was stored a comma separated value files.

The other key difference between the FMTV and MTVR data and the LPD was that the FMTV and MTVR contain Diagnostic Trouble Codes (DTCs), which can point to vehicle faults.

The major takeaway from the HUMS and data storage analysis is the difference in data frequency and signal counts, specifically relating to the engine subsystem, across the three platforms. These differences were further explored in various analyses and are documented in the *Immediate Opportunities* section, below, wherein, the impacts of data frequency relative to autoencoders and the impacts of available number of signals plays into the development of systems level models.

Maintenance Analysis

Maintenance data, consisting of fault descriptions and component replacements, was provided to RIT for the Army FMTV vehicles. The data was analyzed to identify components that contribute significantly to replacement costs. Part cost, labor cost, and frequency of replacement were all considered. The goal was to identify a number of components that may benefit from the implementation of monitoring and prognostic development.

The Navy did not provide maintenance data, but provided HUMS datasets that were associated with select faults on the LPD. The dataset consisted of HUMS data up to the occurrence of an event (fault), thus providing some ground truth of when the fault occurred. There were three different types of faults across the data provided. The maintenance analysis performed on this data was simply to gain an understanding of the dataset; the deep dive into the application of prognostics occurred in the Immediate Opportunities analysis, as described below.

From the maintenance analysis, FMTV work orders for engine replacements, transmission control assembly replacement, and fuel injectors were identified as faults to be investigated in the Immediate Opportunities analysis. Additionally, numerous components were identified as contributing significantly to repair costs and providing some indication of where additional monitoring should be considered. For the LPD, the HUMS data on some platforms was limited, so specific faults that corresponded with larger HUMS data sets were identified as having the most opportunity for further investigation in this research project.

Immediate Opportunities

The goal of this component of the research was to evaluate approaches that could be immediately applied to the faults identified in the Maintenance Analysis. As the first step toward prognostics is anomaly detection, significant effort was applied toward the application of autoencoders, both multi-layer perceptron and convolutional neural networks (NN), to multiple faults. As the data frequency varied between the FMTV and the LPD, the impact of frequency on autoencoders could be assessed.

Autoencoders proved to be valid anomaly detectors for the transmission control assembly fault that was evaluated. Unfortunately, the engine replacements did not have enough data surrounding the fault to develop a model, and the autoencoder for the fuel injector faults was not sensitive enough to identify the actual failures with the data elements in the data set.

Two data modeling approaches were applied to the LPD data, an autoencoder model approach was used to develop condition indicators for specific faults and a systems modelling approach was used for the engine subsystem. The provided datasets contained overlapping faults, which impeded autoencoder development, reinforcing the need for ground truth in the data prior to model development. Despite overlapping data, an autoencoder topology was developed that identified promising condition indicators at the innermost layer through both visual analysis and utilization of a discriminability metric. A system level model was developed based on the many available signals and RIT's engineering knowledge of diesel engines, which provided relationships between the signals collected, allowing for the left turbo outlet pressure to be evaluated utilizing support vector regression. Based on the models, an anomalous region in the data was successfully identified.

Best practices/Emerging Trends

The goal of the best practices and emerging trends effort was to perform a literature review of current approaches and trends in PHM. The review provided an overview of the layers of PHM capability from anomaly detection to diagnostics and finally prognostics. Additionally, practices applied to specific related subsystems such as engines, gears, and electronics were explored. An additional review of related fields, such as medicine, was performed to identify any approaches that may be translatable to vehicles.

From the research, three major developing trends were identified: 1) high performance computing is driving the next generation of PHM, 2) software flexibility is crucial as the libraries and environments utilized in PHM are rapidly changing, and 3) open datasets are providing significant opportunity for PHM development.

Roadmap

A "roadmap" was developed to provide future guidance for PHM, including current industry trends that may have significant impact. The roadmap is included here as Section 7, and begins by covering the PHM development ecosystem, including existing tools and environments that are helping to greatly advance PHM. The Python scientific computational ecosystem is the most commonly utilized and advanced ecosystem for PHM development. Similarly, with an open environment, the ability to link Python with other simulation environments is crucial.

The concept of digital twins was reviewed to show how it relates to traditional PHM implementations. The digital twin concept has received significant attention over the last 5 to 10 years. However, finding a concrete definition of digital twin is difficult, so an attempt was made to show the various aspects that make up a digital twin and how PHM is a critical component (see Section 7.3).

The Robotic Operating System (ROS) is provided as an example of a tangentially related field that may have impacts on future PHM systems due to advance concepts being considered as a part of the ROS ecosystem. In particular, with the expansion of networked sensors, edge devices, etc., ROS 2.0 provides inherent cybersecurity measures, which allowed the Army to select it for their next autonomous vehicle.

Finally, a review of the current state of data storage as it relates to CBM and PHM is presented. Particularly, the split between business analytics and PHM development that impacts how data is stored and used. Apache Spark is a distributed data and processing platforms for Big Data. However, as noted, the Python scientific computational ecosystem is the preferred computational ecosystem for PHM. Therefore, although data may be stored in Spark in significantly large quantities, subsets of the data will likely be pulled into the HDF5 files for repeat processing during PHM development.

4 HUMS and Maintenance Data Evaluation

One of the major objectives of this research was to perform a cross-platform, cross-branch analysis of the application of CBM+. Each branch has implemented some form of CBM+, each with differing characteristics and approaches. Therefore, each branch was asked to provide data for one platform for analysis. For the Marine Corps, the Medium Tactical Vehicle Replacement (MTVR) was selected as it had previously been equipped with monitoring equipment on approximately 24 vehicles, 23 of which had valid data as seen in section 4.1.3. Marine Corps data was provided by the Office of Naval Research. The Army selected the Family of Medium Tactical Vehicles (FMTV) as it was part of the Army CBM+ program implementation and contained the data of 597 vehicles. FMTV data was provided by the U.S. Department of Army's Tank & Automotive Research Development and Engineering Command (TARDEC), which was recently rebranded the U.S. Army Combat Capabilities Development Command (CDCC), Ground Vehicle Systems Center. The Navy selected six ships that consisted of guided-missile destroyers, guided-missile cruisers, and the landing platform dock (LPD). The Navy data was supplied by the Office of Naval Research.

As the goal was to evaluate cross-platform potential, diesel engines were identified as common system across the data sets from all three military branches. The majority of the analysis was focused on the engines. However, this reduced the Navy ship data to the two LPD platforms as they utilize diesel engines for propulsion, as opposed to the gas turbines used in the guided-missile destroyers and cruisers. The three data sets covered differing number of platforms and differing durations of data collection as shown in Table 1.

Table 1 - Summary of the data extent for each dataset provided

Dataset	Number of Platforms	Collection Start Date	Collection End Date
Army	597	19 Dec 2012	14 May 2014
Marine Corps	24 (23 valid)	28 April 2015	3 December 2015
Navy	6	7 September 2016	26 July 2017

4.1 Analysis of HUMS Data

4.1.1 Description of the Army HUMS Data

The Army CBM+ program implementation on HUMS on the FMTV has direct impacts on the capabilities of diagnostics and prognostics on the vehicles. To better understand some of the features of the HUMS data collection, data format, data extent, handling of special information (Diagnostic Trouble Codes), and summary data (STATS files) were evaluated.

4.1.1.1 Data Format

Army data is stored as a set of compact files in Common Data Format (CDF). It is important to emphasize that CDF is more than a data format -- it is a conceptual data abstraction for storing, manipulating, and accessing multidimensional data sets [13]. CDF is very similar to Hierarchical Data Format, version 5 (HDF5) [14], but HDF5 is considerably more widely-used. The contents of a CDF file are "variables" and "attributes." Variables are represented as n-dimensional arrays. A zero-dimension array is a scalar, or single value, whereas a one-dimensional array resembles a vector, and a two-dimensional array resembles a table. Higher dimensions are possible as well.

The other abstraction is the "attribute," which is a form of metadata. Individual variables can have attributes, as can the file itself. Attributes can help describe the variable (for example, the source of the

data or its units of measure). Attributes can also describe the entire file (these are known as "global" attributes) to indicate information like when the file was recorded or provide other information about the overall contents of the container.

The CDF format also allows for data compression. The purpose of compression is to make the file smaller when storing it. Files may be compressed in one of two ways: either by compressing the entire file, or by compressing individual variables therein. The compression method chosen (if any) impacts both size and performance (specifically access speed and temporary file requirements).

CDF performance is comparable to other binary formats. The main performance benefit of using CDF is realized when accessing a single variable within the file. The official library implementation, from NASA, uses indexes to seek to a single variable without the need to read the entire file. For very large files, this eliminates some of the input/output (I/O) bottleneck. Consider, for example, reading a many-gigabyte file over a 6 Gb/s SATA interface; with CDF you can avoid reading the entire file - the interface only transfers the portions of interest.

4.1.1.2 The ABCD Layer

The Army Bulk CBM Data (ABCD) format is a specification that defines how platform health data is stored in CDF files. It includes standard attribute names (including some that are mandatory) that help users interpret the contents of the file. It also defines semantics for handling missing values.

Army CBM vehicle health data is tabular, but the ABCD format requires data be stored as single-dimension arrays. In order to use these variables as if they were tables, the variables are grouped into "organizational channels", as shown in Table 2 .

Table 2: Organizational Channels in the ABCD Format

Channel	Description
1HzData	Data samples (for example, sensor readings). Values are recorded at constant intervals (usually once per second), and all variables are recorded at the same time.
FaultData	Diagnostic Trouble Codes. The sample time is ad-hoc, meaning new entries are recorded as needed. Just like sensor readings, every record includes values for every value.
StartupData	This channel contains basic startup/shutdown information, similar to what AHM calls the "mission table", along with certain cumulative data, similar to what AHM calls "cumulations"

This is accomplished by adding a string attribute to each variable, with the attribute name "LVL90.GEN.OrgChannelName" and a value matching one of the three channel names.

The benefits of the ABCD format are:

1. The activity of an entire day is grouped into a single file. This means that a single file captures the entire activity of one vehicle on one day.
2. File level compression makes these files relatively compact.

To illustrate the implementation of the ABCD format, Table 3 lists the attributes of $EngSpeed$ signal.

Table 3: List of attributes of EngSpeed signal

Attribute	
LVL00.SENS.Meas_Location.meas_loc_site	000004260000002E
LVL90.GEN.FileId	[4200]
LVL90.GEN.FileSubId	[3]
LVL90.GEN.ModeNum	[-1]
LVL90.GEN.RunNum	[-1]
LVL90.GEN.DataModeName	
LVL90.GEN.OrgChannelName	1HzData
LVL90.MC.NumberOfValidPointsLastRecord	[314]
LVL90.MC.Description	NotSet
LVL90.MC.ID	NotSet
LVL90.MC.ChanNumber	[42]
LVL90.MC.OrgChanNumber	[1]
LVL90.MC.ColType	[2]
LVL90.MC.ColTitle	EngSpeed
LVL90.MC.ColUnits	Revolutions Per Minute (RPM)
LVL90.MC.ColKeyword	EngSpeed
LVL90.MC.CDFColType	CDF_FLOAT
LVL90.MC.AreStatsValid	[1]
LVL90.MC.Max	[715.75]
LVL90.MC.Min	[688.375]
LVL90.MC.Mean	[700.0509554140127]
LVL90.MC.RMS	[700.0574196621621]
LVL90.MC.SD	[3.0132314515778784]
LVL90.MC.PointMax	[230]
LVL90.MC.PointMin	[244]
LVL90.MC.PointAbsMax	[230]
LVL90.MC.ColMax	[nan]
LVL90.MC.ColMin	[4.1720134881979434e-308]
LVL90.MC.ColNumBins	[1447821312]
LVL90.MC.ColVarBins	

Attribute	
LVL00.SENS.Meas_Loc_Type.ml_db_id	[1]
LVL00.SENS.Meas_Loc_Type.ml_db_site	000003F700000001
LVL00.SENS.Meas_Loc_Type.ml_type_code	[1]
LVL00.SENS.Meas_Location.meas_loc_id	[256]
LVL02.ACQ.TS.offset_array_name	UTC_1Hz
LVL04.FILE.Multicolumn.version	[1]
LVL04.SENS.Meas_Loc_Assoc.related_mloc_id	[876]
LVL04.SENS.Meas_Loc_Assoc.related_mloc_site	000004260000002E
LVL10.SENS.Missing_value.value	2147483648
LVL90.SENS.Eng_Unit_Type_Name	Revolutions Per Minute (RPM)
LVL90.SENS.Meas_Loc_Chr_Data.Frequency	1
LVL90.SENS.Meas_Loc_Chr_Data.Frequency_Units	Hz
LVL90.SENS.Meas_Loc_Type.Name	Speed, Rotational
LVL90.SENS.Meas_Location.name	J1939 - Engine Speed (Engine)
LVL90.SENS.zVar	EngSpeed

4.1.1.3 Initial Analysis

The CDF format has an application programming interface (API) for most major languages (including C, and Java) and allows fast access to individual signals, obviating the need to load the entire file contents into the memory. Thus, an analyst can load only specific signals of interest, which greatly improves the performance. For example, in the Python scientific ecosystem, using *cdflib* library, a signal with 10,000 points loads into the computational environment of a local machine from a file stored on a server in less than 100 ms.

RIT developed a simple class, *armyCDF* (see the listing in Figure 3), with a few utility methods, to facilitate the data analysis, and a visualization tool (Figure 4), enabling a quick browse through the signals across the vehicles and over time.

```python
import os
import datetime
import time
import numpy as np
import cdflib

class armyCDF():
    def __init__(self, veh= '  Vehicle serial #      :', file_index = 0 ):
        self.topFolder = r'...\Data\cdf\FMTV\FMTV-A1'
        self.vehFolder =  self.topFolder +"\\"+veh
        self.files  = self.getFiles()
        self.file_name = self.files[file_index]
        self.D = cdflib.CDF(self.file_name)
        self.sigs = self.D.cdf_info()['zVariables']
        self. dts = np.diff(cdflib.cdfepoch.unixtime(self.D.varget('UTC_1Hz')).ravel())
    def updateFolder(self, veh):
        self.vehFolder = self.vehFolder.split("FMTV-A1")[0]+"FMTV-A1"+"\\"+veh
        self.files = self.getFiles()
    def getVehs(self):
        return os.listdir(self.topFolder)
    def getFiles(self):
        subFolders = os.listdir(self.vehFolder)
        files = []
        for subFolder in os.listdir(self.vehFolder):
            for file_ in os.listdir(self.vehFolder+"\\"+subFolder):
                files.append(self.vehFolder+"\\"+subFolder+'\\'+file_)
        return(files)
    def getStartTime(self):
        t0 = cdflib.cdfepoch.breakdown(self.D.varget('UTC_1Hz').ravel()[0])
        return(f"{t0[0]}-{t0[1]:02d}-{t0[2]:02d} {t0[3]:02d}:{t0[4]:02d}:{t0[6]:02d}")

    def getNoPts(self):
        return(len(self.D.varget('UTC_1Hz').ravel()))
    def getDuration(self):
        tU = cdflib.cdfepoch.unixtime(self.D.varget('UTC_1Hz').ravel())
        durSec = tU[-1]-tU[0]
        hr = int(durSec//3600)
        min_ = int((durSec-3600*hr)//60)
        sec_ = int(durSec - 3600*hr - 60*min_)
        return(f'{hr:02d}:{min_:02d}:{sec_:02d}')
    def getSignal(self, sigName, excludeOutlier = True):
            s = self.D.varget(sigName).ravel()
            tLabel = self.D.varattsget(sigName)["LVL02.ACQ.TS.offset_array_name"]
            sQuant = self.D.varattsget(sigName)['LVL90.SENS.Meas_Loc_Type.Name']
            sUnit = self.D.varattsget(sigName)['LVL90.MC.ColUnits']
            t =  np.array([datetime.datetime(y,m,d,H,M,S)
                           for (y,m,d,H,M,S,_,_,_) in cdflib.cdfepoch.breakdown(self.D.varget(tLabel).ravel())])
            if excludeOutlier:
                try:
                    t = t[s<1e9]
                    s = s[s<1e5]
                except:
                    t = np.array([])
                    s = np.array([])

            return(t, s, sQuant + ",  ("+sUnit+")")
    def update_data(self, file_name):
        self.D.file.close()
        self.D = cdflib.CDF(file_name)
        self.file_name = file_name
        self.sigs= self.D.cdf_info()['zVariables']
#         try:
#             self.dts= np.array([dt_i.seconds for dt_i in np.diff(self.D['UTC_1Hz'][0])])

    def distance_from_cum(self):
        return  self.D["TotVehDist"][0][-1] - self.D["TotVehDist"][0][0]
    def distance_from_speed (self):
        v = self.D["VehSpeedEng"][0]
        if np.any(v>1e5):
            v[v>1e5] = 0
        return np.dot(v[:-1],self.dts)/3600.
    def engine_hrs(self, rpm_min = 10, fixed_dt = False):
        ind = (self.D["EngSpeed"][0][:-1]>rpm_min).nonzero()[0]
        if fixed_dt:
            return len(ind)/3600
        else:
            return (self.dts[ind].sum()/3600.)
```

Figure 3: A class with a few utility methods for analysis of CDF data.

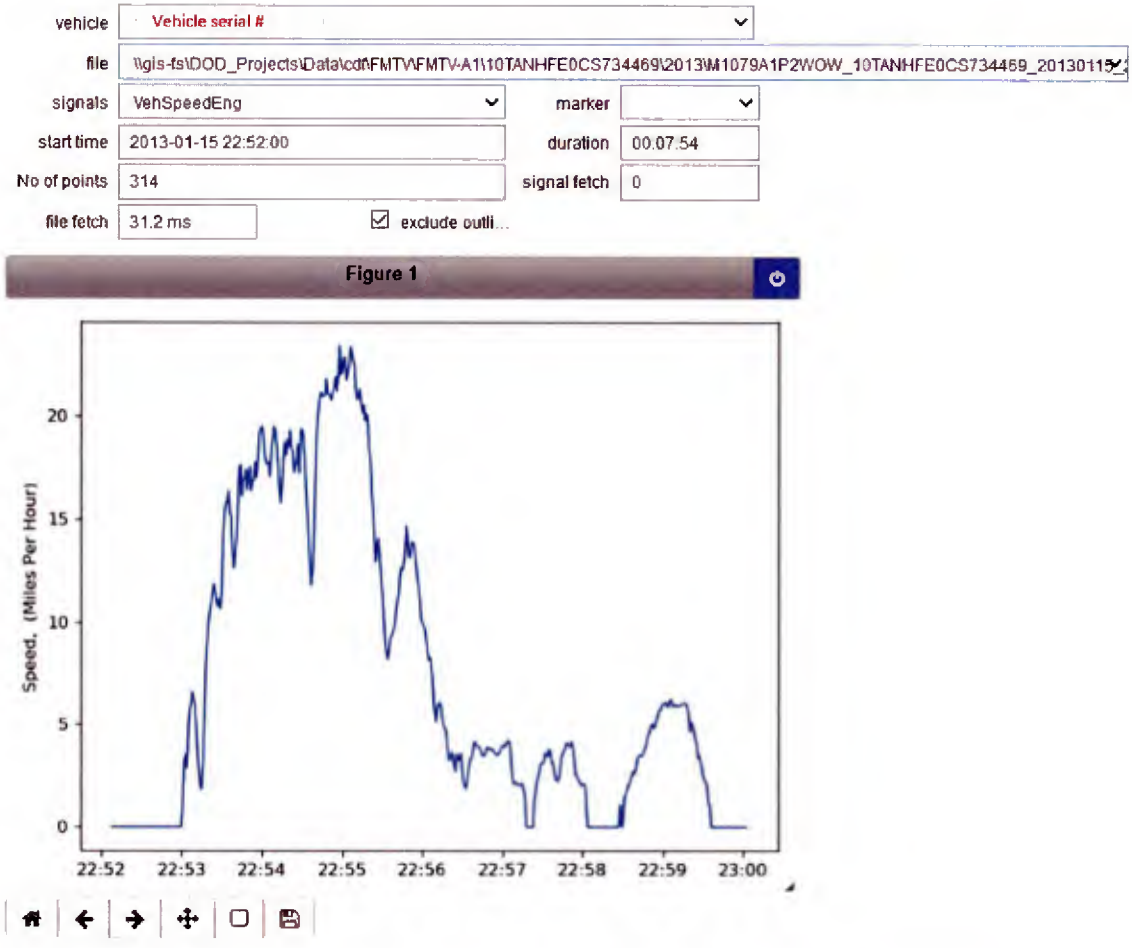


Figure 4: A visual interface for browsing through the signals across fleet vehicles and time.

4.1.1.4 Data Extent

Data extent was examined along multiple dimensions: number of vehicles, number of signals per vehicle, data collection span, temporal resolutions of the data, and vehicle usage activity during the collection. Table 4 summarizes the data extent at a high level, with the overall HUMS fleet data span, number of participating vehicles, number of signals on a vehicle, and the temporal resolution of the signals.

Table 4: High-level summary of data extent

Data range	Number of Vehicles	Number of Signals*	Sampling Rate [Hz]
19 Dec 2012 – 14 May 2014	597	115-162	1

*Number of signals per vehicle can vary as shown in Figure 14 The range here is based on the number of signals at the end of the data collection.

4.1.1.5 *Recorded Vehicle Usage Activity*

The span of the data collection does not provide complete information. It is important to understand how much activity was captured during this time interval. Three metrics are used to provide an overview of the fleet activity: evidence of HUMS collection for a given day, hours of engine activity, and driving distances.

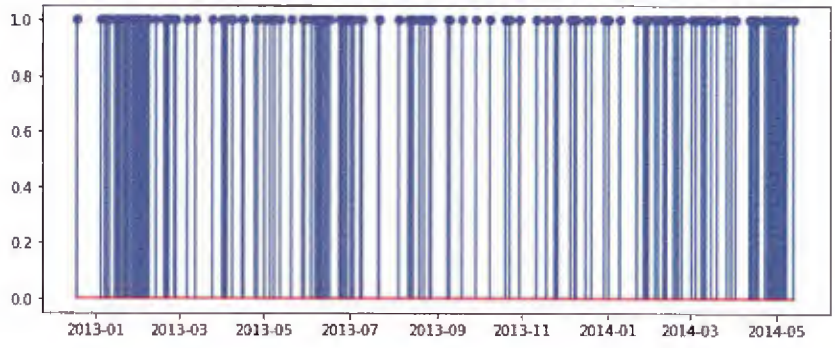


Figure 5: An example vehicle activity

The initial assessment started by evaluating individual vehicles. Figure 5 shows the days of activity recorded by the HUMS system of a typical vehicle, with each elevated line indicating a day when data was collected by the HUMS system.

Figure 6 expands on Table 4 and shows how many vehicles were active on a given day during the time span of the HUMS data collection. During the majority of the time, activity varied widely between as few as 10 vehicles and up to more than 300 on a single day. The subplot on the right of the graph shows a corresponding histogram of the occurrence of the number active vehicles

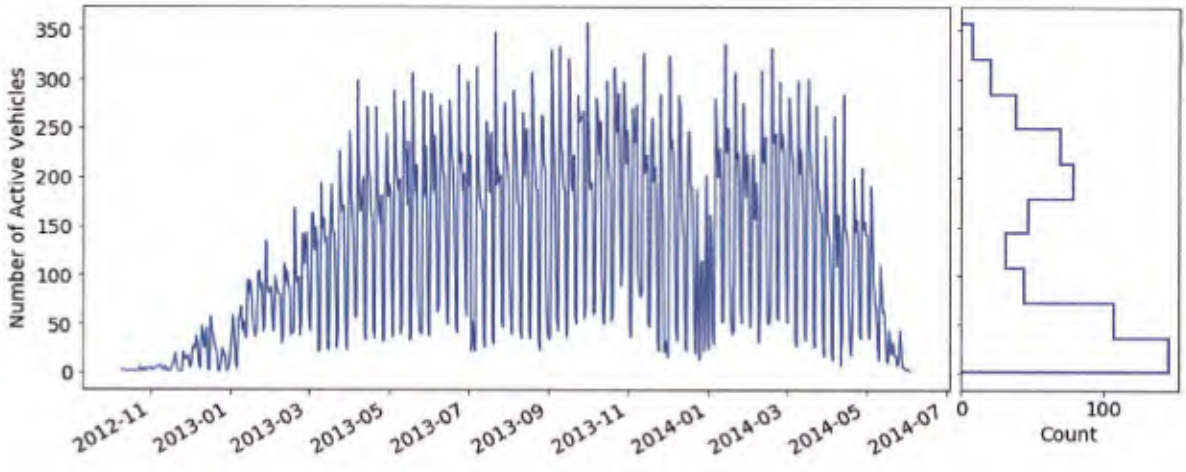


Figure 6: Fleet activity during HUMS data collection, includes 73,590 active days.

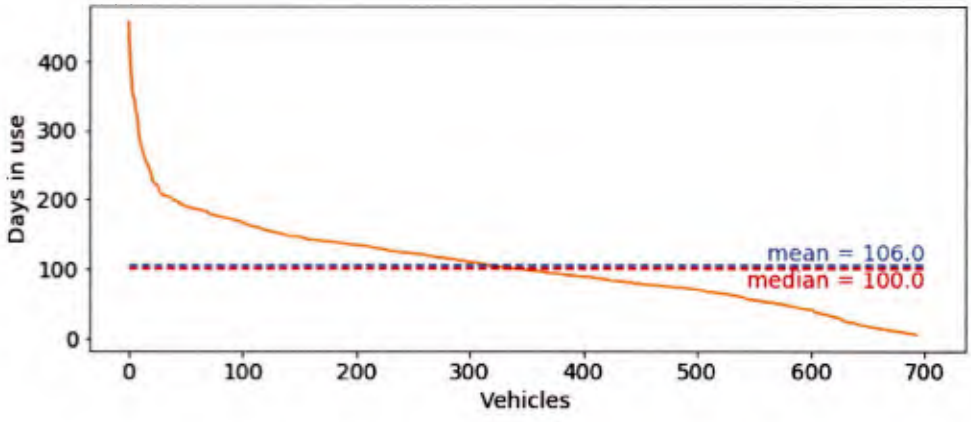


Figure 7: Number of days in use for a given vehicle

Figure 7 shows daily activity of individual vehicles during the data collection span. Very few vehicles collected more than 200 days of activity, with the mean and median activity being 106 and 100 days, respectively.

4.1.1.6 Engine Activity

A good way to describe utilization of a vehicle is a cumulative sum of the engine usage, as shown in Figure 8. Most of the data is collected during short intervals of time, as indicated by the steps in the cumulative engine usage.

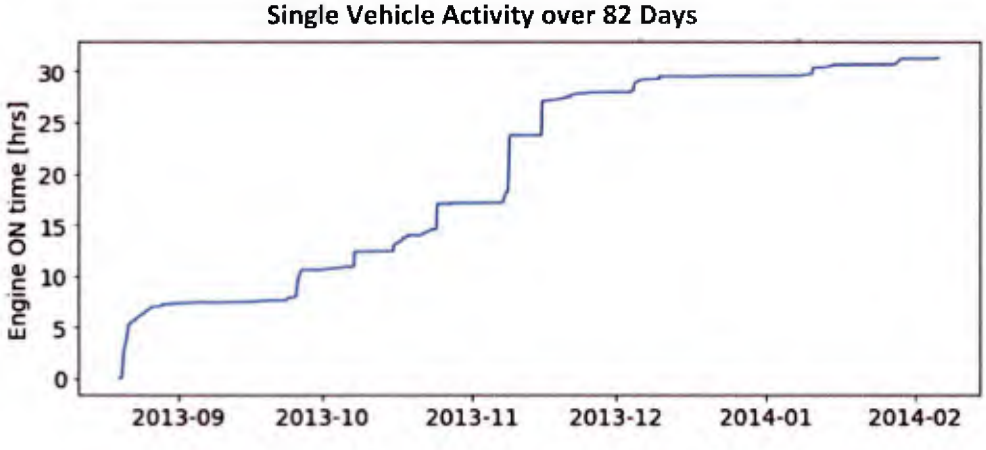


Figure 8: Cumulative engine usage of a single vehicle during the HUMS data collection

Figure 9 illustrates the engine usage of three different vehicles, each exhibiting different operating patterns.

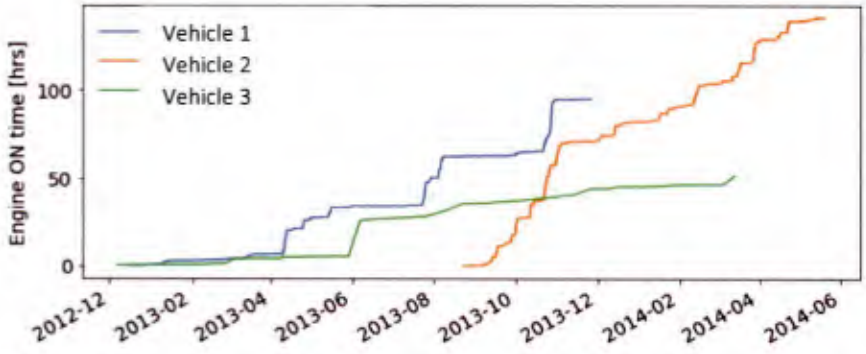


Figure 9: Cumulative engine usage of three trucks during the HUMS data collection

The fleet summary of the usage is provided in Figure 10, with the title indicating the number of the processed vehicles (694), the subset of vehicles with the requisite summary (467 *vehicles with EngTotHrs*), and total hours of engine on. The left subplot shows individual cumulative histories of the vehicles and the right subplot shows the histogram of the total engine hours, with a mean of 117 hours and a median of 86 hours.

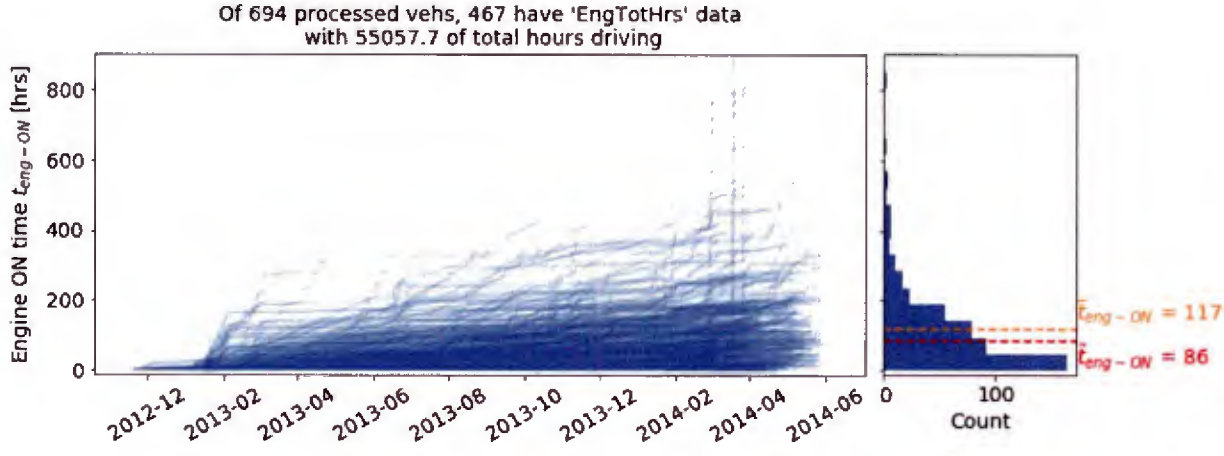


Figure 10: Fleet summary of engine usage by truck

4.1.1.7 Driving Activity

Figure 11 shows a cumulative driving distance for a single vehicle, similar to Figure 8. There are a few short time intervals of significant driving, separated by much longer periods with virtually no driving.

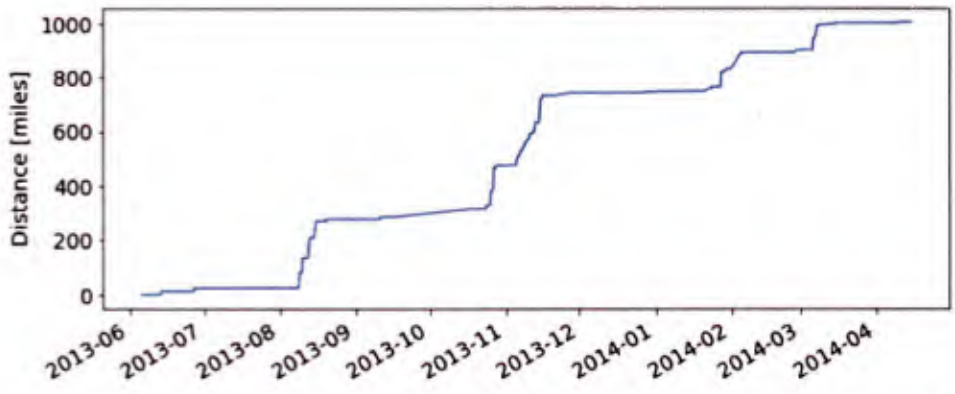


Figure 11: Cumulative driving distribution of a single vehicle during the HUMS data collection

Once again, there is considerable variation from vehicle to vehicle, as depicted in Figure 12, which shows cumulative distance travelled of four different trucks. All vehicles show the same pattern of periods of driving separated by longer intervals of little to no driving.

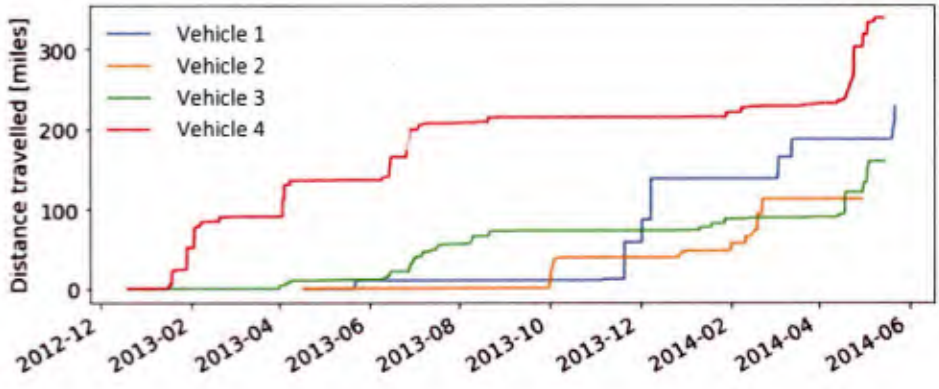


Figure 12: Cumulative driving of four vehicles during the HUMS data collection

Figure 13 summarizes the driving distance across the entire fleet. The fleet contained 447 vehicles with driving distance collected and a total of 285,929 miles of driving. The left subplot shows traces of cumulative driving by individual vehicles and the right subplot contains the distribution of total driving, with the mean and median being 639 and 440 miles, respectively.

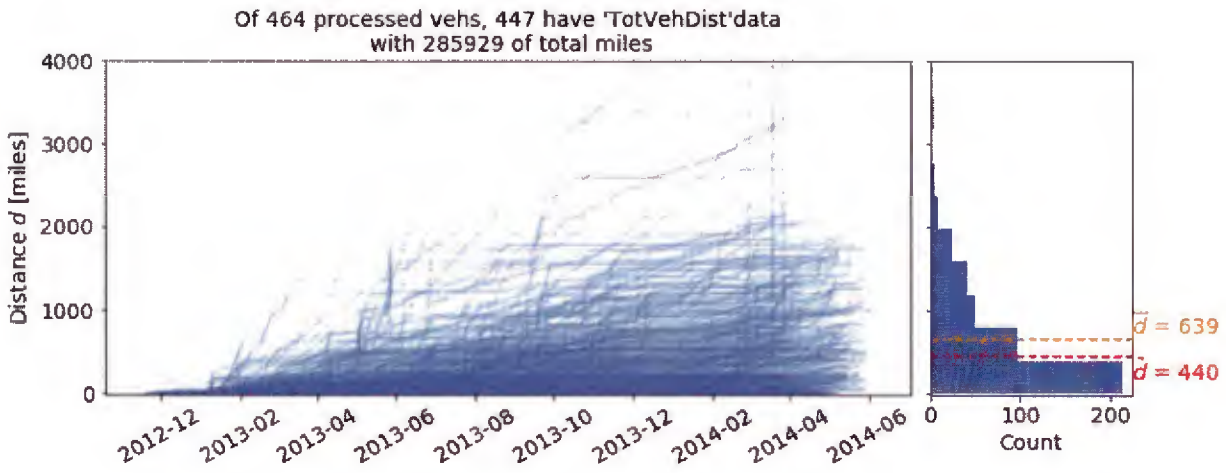


Figure 13: Fleet summary of driving by truck

4.1.1.8 Number of Signals

The HUMS signals are typical signals that can be found on the J1939 or J1708 databus. Within the Army data collection system, the number of signals collected on a vehicle can vary over time, as illustrated for a single vehicle in Figure 14. For simplicity and to avoid the effect of days when the vehicle is not utilized, the number of signals is plotted against the record of utilization (sequential file number), instead of the time. The black trace, associated with the left y-axis, shows the number of signals collected across the number of files, and the red trace, associated with the right axis, shows the change in number of signals.

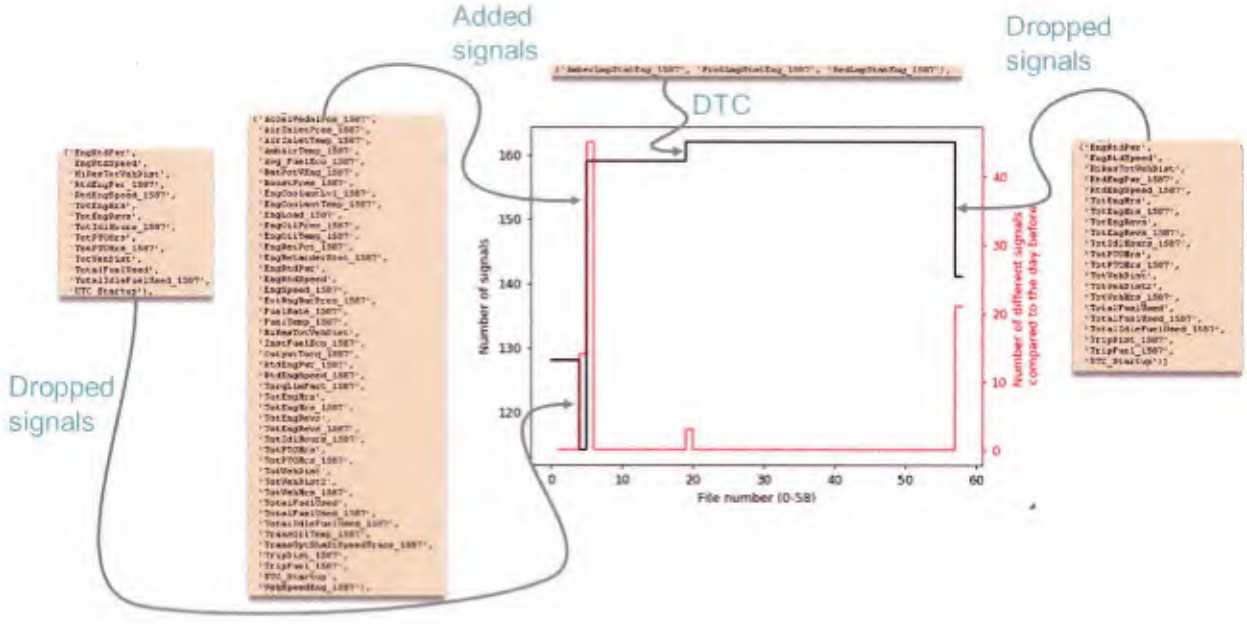


Figure 14: Signals on a single vehicle over time.

Three of the four changes (the first, the second, and the fourth) in the number of signals suggest some changes in the data collection system, as the number of added/dropped signals is relatively large. The

third change appears to be caused by a DTC event. The changing number of signals may present a problem in modeling should the system stop collecting one of the signals being used for a model. It is more likely that the data collection system was being updated, as the availability of signals was identified and any future updates would only see an increase in signals. However, due diligence should be taken when updating the system to identify any potential impacts on existing algorithms or models.

The naming convention for each signal identifies whether the data was collected from the J1587 databus, if it was not, then it is assumed the data was collected from the J1939 databus. The name of the signal can also be correlated directly to the databus signal specifications for J1939 and J1708. Although the data reporting rates for the signals is based on the specification, and for many signals is reported as many as 10 times per second, every data signal is only stored once per second in the CDF files. For a snapshot of the full signal list of collected data on the FMTV, see Appendix A.

4.1.1.9 DTC Analysis

Table 5 shows a Pareto list (top 20) of DTCs across all of the vehicles during the entire data collection period. Each row represents a DTC on a specific vehicle, highlighting the most recurring DTCs. In addition to vehicle identification number, the table shows the associated Suspect Parameter Number (SPN), fault description, and the count. Figure 15 provides a graphical view of the distribution of DTCs from the data in the table.

From Table 5, the DSC related entries, highlighted in blue, indicate a problem with the data collection system. The majority of the remaining repeated faults indicate likely sensor problems, i.e. shorted high or shorted low. The gray highlighted items appear to be related, with the sensor sometimes reporting high but valid values, and at other times appearing to change to rapidly. The red highlighted items represent data that may be valid, but are not typical of operational data.

Table 5: Fault counts. Top 20 recurring DTCs on a vehicle occurrence

Vehicle	SPN	Fault Description	Count
SN:Vehicle1	102	SAE - Boost Pressure - Voltage above normal or shorted to high source	1911
SN: Vehicle2	520196	DSC J1939 no data for 3 seconds	1403
SN:Vehicle3	105	SAE - Intake Manifold Temperature - Voltage above normal or shorted to high source	1343
SN:Vehicle4	520223	DSC Part number invalid	1030
SN: Vehicle5	102	SAE - Boost Pressure - Voltage above normal or shorted to high source	888
SN: Vehicle6	1	SAE - Wheel Sensor ABS Axle 1 Left - Abnormal rate of change	699
SN: Vehicle6	4	SAE - Wheel Sensor ABS Axle 2 Right - Abnormal rate of change	690
SN: Vehicle7	231	SAE - J1939 Data Link - Abnormal update rate	606
SN: Vehicle8	520197	DSC J1708 no data for 3 seconds	596
SN: Vehicle9	102	SAE - Boost Pressure - Voltage below normal or shorted to low source	582
SN: Vehicle10	15	SAE - Relay Diagonal 2 - Voltage below normal or shorted to low source	576
SN: Vehicle6	4	SAE - Wheel Sensor ABS Axle 2 Right - Data valid but below normal operational range - Most severe level	547
SN: Vehicle11	108	SAE - Barometric Pressure - Voltage above normal or shorted to high source	540
SN: Vehicle12	168	SAE - Battery Potential (Voltage) - Data valid but below normal operational range - Most severe level	532
SN: Vehicle13	520223	DSC Part number invalid	510
SN: Vehicle14	100	SAE - Engine Oil Pressure - Data valid but below normal operational range - Most severe level	503
SN: Vehicle15	108	SAE - Barometric Pressure - Voltage above normal or shorted to high source	495
SN: Vehicle16	520197	DSC J1708 no data for 3 seconds	488
SN: Vehicle17	4	SAE - Wheel Sensor ABS Axle 2 Right - Data valid but below normal operational range - Most severe level	481
SN: Vehicle18	2	SAE - Pneumatic Control Unit (PCU) - Voltage below normal or shorted to low source	477

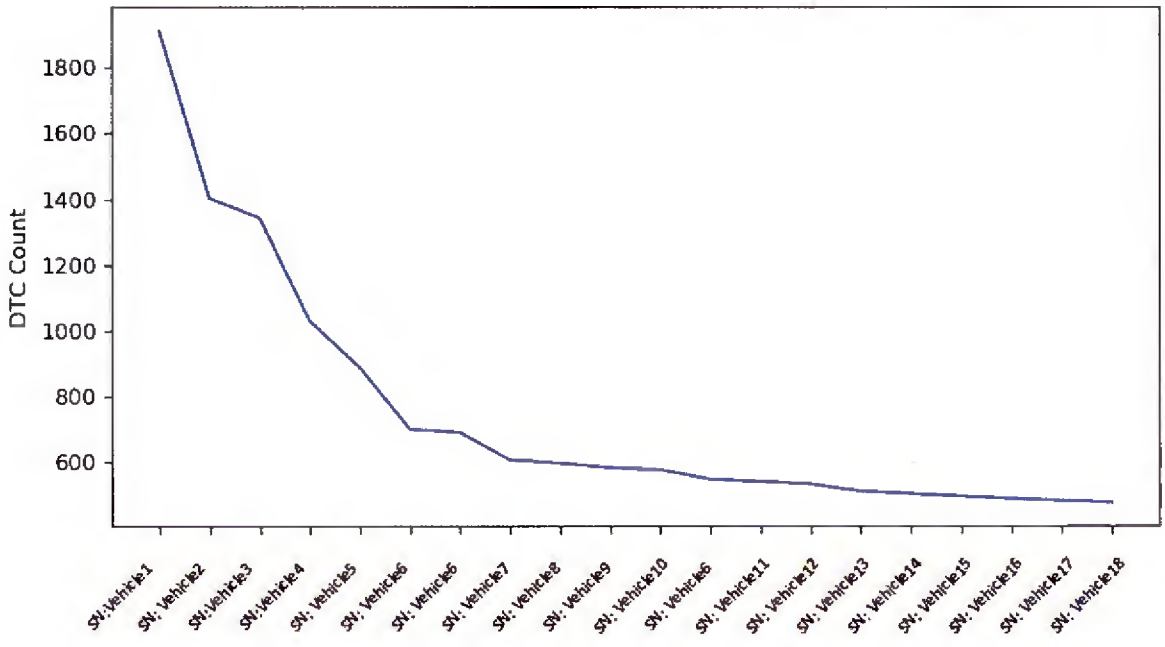


Figure 15: Vehicles sort by DTC counts (top 20).

DTCs can be very useful for development and enhancement of PHM capabilities, because they contain the ground truth information for impending failures. In fact, they represent a level of condition monitoring that is already implemented by electronic control units (ECUs), employing traditionally conservative thresholds to reduce false alarm problems. Furthermore, the four lamps (Malfunction Indicator Lamp (MIL)[2], Red Stop Lamp (RSL), Amber Warning Lamp (AWL) and Protect Lamp (PL)) also contain information on severity of the problem.

To get a better sense of this ground truth data, observed MIL values were examined across the fleet. Table 6 lists the values (in both decimal and Boolean format), together with the number of instances.

Table 6: MIL with state values and number of instances across the fleet

State (Decimal)	State (Binary)	Number of instances
0	0000 0000	10,916
3	0000 0011	395,776
255	1111 1111	218,847

However, the recorded values do not map directly to the J1939 the MIL specification, which defines MIL as having three states as described in Table 7.

[2] Also known as "check engine light".

Table 7: Fault counts. Top 20 based on occurrence

Range Name	Transmitted Value (Boolean)
Disabled (off)	00
Enabled (on)	01
Error Indicator	10
Not available or not installed	11

Amber Warning Lamp (AWL) shows similar results as MIL (see Table 8); however, a value of 0 is never seen; instead, 1, 3 and 255 are seen. A likely interpretation is that 255 value is HUMS's higher-level indication for an unknown value. However, interpretation of the other values is difficult because they do not follow the standard specification.

Table 8: AWL with state values and number of instances across the fleet

State (Decimal)	State (Boolean)	Number of instances
1	0000 0001	11,811
3	0000 0011	394,881
255	1111 1111	218,847

Because DTC were considered as a valuable ground truth for the health of the vehicle, as well as an alternative mechanism (in addition to maintenance data) to trigger data-driven model development, it was important to correctly identify and interpret their content. Although information on the DTCs is collected, it is often associated with an AWL or MIL. The AWL and MIL are the typical feedback systems (check engine light) to a driver in a vehicle. However, the AWL and MIL states recorded by the HUMS system do not directly correlate with the specification, making their interpretation more difficult with regard to faults in the system. It is recommended to keep the states the same in the HUMS system as they are in the specification (00, 01, 10, and 11).

4.1.1.10 Stats File Data

In addition to the daily summary file analyzed above, RIT briefly examined WSTATS files, which contain more succinct daily summaries of the vehicle. Interpretation of the stats signals is shown in Figure 16.

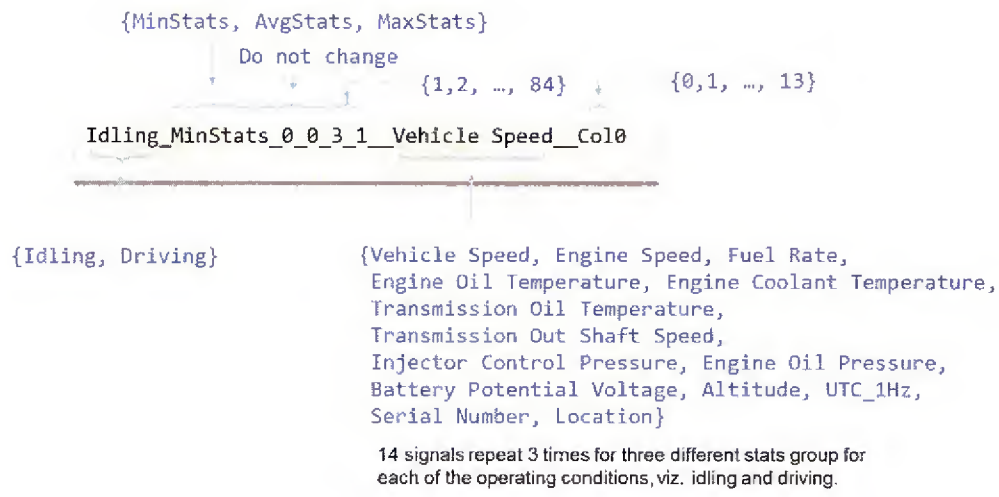


Figure 16: Interpretation of vehicle stat signals.

A stat first specifies the context (whether it relates to idling or driving), then the nature of statistical computation (e.g. minimum, average, or maximum), and the signal it applies to (e.g. vehicles speed, engine speed, etc.). Fourteen signals repeat three times for three different statistical computations and two different contexts (driving and idling).

```
'Idling_MinStats_0_0_3_1__Vehicle Speed__Col0', <zAttrList:
LVL00.SENS.Meas_Location.meas_loc_id: 1 [CDF_INT4]
LVL00.SENS.Meas_Location.meas_loc_site: 0000041D00000002 [CDF_CHAR]
LVL00.SENS.Meas_Location.ml_db_id: 1 [CDF_INT4]
LVL00.SENS.Meas_Location.ml_db_site: 0000041D00000001 [CDF_CHAR]
LVL00.SENS.Meas_Location.ml_type_code: 0 [CDF_INT4]
LVL90.GEN.DataModeName:   [CDF_CHAR]
LVL90.GEN.FileId: 4200 [CDF_INT4]
LVL90.GEN.FileSubId: 3 [CDF_INT4]
LVL90.GEN.ModeNum: -1 [CDF_INT4]
LVL90.GEN.OrgChannelName: Idling_MinStats [CDF_CHAR]
LVL90.GEN.RunNum: -1 [CDF_INT4]
LVL90.MC.AreStatsValid: 1 [CDF_INT4]
LVL90.MC.CDFColType: -1 [CDF_INT4]
LVL90.MC.ChanNumber: 1 [CDF_INT4]
LVL90.MC.ColKeyword: VEHSPEEDENG [CDF_CHAR]
LVL90.MC.ColMax: -1.0 [CDF_DOUBLE]
LVL90.MC.ColMin: -1.0 [CDF_DOUBLE]
LVL90.MC.ColNumBins: -1 [CDF_INT4]
LVL90.MC.ColTitle: Vehicle Speed [CDF_CHAR]
LVL90.MC.ColType: 2 [CDF_INT4]
LVL90.MC.ColUnits: Miles Per Hour [CDF_CHAR]
LVL90.MC.ColVarBins:   [CDF_CHAR]
LVL90.MC.Description: NotSet [CDF_CHAR]
LVL90.MC.ID: NotSet [CDF_CHAR]
LVL90.MC.Max: -999.0 [CDF_DOUBLE]
LVL90.MC.Mean: -999.0 [CDF_DOUBLE]
LVL90.MC.Min: -999.0 [CDF_DOUBLE]
LVL90.MC.NumberOfValidPointsLastRecord: 1 [CDF_INT4]
LVL90.MC.OrgChanNumber: 100 [CDF_INT4]
LVL90.MC.PointAbsMax: 1 [CDF_UINT4]
LVL90.MC.PointMax: 1 [CDF_UINT4]
LVL90.MC.PointMin: 1 [CDF_UINT4]
LVL90.MC.RMS: 999.0 [CDF_DOUBLE]
LVL90.MC.SD: 0.0 [CDF_DOUBLE]
```

```
LVL00.SENS.Meas_Location.meas_loc_id: 1 [CDF_INT4]
LVL00.SENS.Meas_Location.meas_loc_site: 0000041D00000002 [CDF_CHAR]
LVL00.SENS.Meas_Location.ml_db_id: 1 [CDF_INT4]
LVL00.SENS.Meas_Location.ml_db_site: 0000041D00000001 [CDF_CHAR]
LVL00.SENS.Meas_Location.ml_type_code: 0 [CDF_INT4]
LVL90.GEN.DataModeName:   [CDF_CHAR]
LVL90.GEN.FileId: 4200 [CDF_INT4]
LVL90.GEN.FileSubId: 3 [CDF_INT4]
LVL90.GEN.ModeNum: -1 [CDF_INT4]
LVL90.GEN.OrgChannelName: Idling_MinStats [CDF_CHAR]
LVL90.GEN.RunNum: -1 [CDF_INT4]
LVL90.MC.AreStatsValid: 1 [CDF_INT4]
LVL90.MC.CDFColType: -1 [CDF_INT4]
LVL90.MC.ChanNumber: 2 [CDF_INT4]
LVL90.MC.ColKeyword: ENGSPEED [CDF_CHAR]
LVL90.MC.ColMax: -1.0 [CDF_DOUBLE]
LVL90.MC.ColMin: -1.0 [CDF_DOUBLE]
LVL90.MC.ColNumBins: -1 [CDF_INT4]
LVL90.MC.ColTitle: Engine Speed [CDF_CHAR]
LVL90.MC.ColType: 2 [CDF_INT4]
LVL90.MC.ColUnits: Revolutions Per Minute (RPM) [CDF_CHAR]
LVL90.MC.ColVarBins:   [CDF_CHAR]
LVL90.MC.Description: NotSet [CDF_CHAR]
LVL90.MC.ID: NotSet [CDF_CHAR]
LVL90.MC.Max: 683.875 [CDF_DOUBLE]
LVL90.MC.Mean: 675.9583333333334 [CDF_DOUBLE]
LVL90.MC.Min: 659.375 [CDF_DOUBLE]
LVL90.MC.NumberOfValidPointsLastRecord: 6 [CDF_INT4]
LVL90.MC.OrgChanNumber: 100 [CDF_INT4]
LVL90.MC.PointAbsMax: 6 [CDF_UINT4]
LVL90.MC.PointMax: 6 [CDF_UINT4]
LVL90.MC.PointMin: 2 [CDF_UINT4]
LVL90.MC.RMS: 676.0096076117064 [CDF_DOUBLE]
LVL90.MC.SD: 9.128683963916237 [CDF_DOUBLE]
```

(a) *(b)*

Figure 17: Attributes of WSTATS signals.

The attributes for each signal are also provided, as illustrated in Figure 17. In Figure 17a, the dashed box within the figure shows that some of the values are not computed, or are not reliably computed, indicated by -999.0 value; in Figure 17b, the dashed box shows the case of reliably computed values.

Examples of WSTAT values are depicted in Figure 18. As shown, some WSTATS do not have meaningful content (Figure 18a), while others do (Figure 18b). Of 49 analyzed WSTATS files, 30 have Idling_MinStats_0_0_3_2__Engine Speed__Col1 = [-999]. For the remaining 19 files, the values were plausible, as depicted in Figure 19. These types of summary data elements are natural for tracking vehicle usage for the purpose of preventative or predictive maintenance. Due to issues with data quality, these signals were not used in the subsequent analyses as they could be more accurately recreated directly from the HUMS data. In practice, it is important to accurately calculate these within a HUMS or PHM system.

```
Idling_MinStats_0_0_3_1__Vehicle Speed__Col0 [-999.]
Idling_MinStats_0_0_3_2__Engine Speed__Col1 [-999.]
Idling_MinStats_0_0_3_3__Fuel Rate__Col2 [-999.]
Idling_MinStats_0_0_3_4__Engine Oil Temperature__Col3 [-999.]
Idling_MinStats_0_0_3_5__Engine Coolant Temperature__Col4 [-999.]
Idling_MinStats_0_0_3_6__Transmission Oil Temperature__Col5 [-999.]
Idling_MinStats_0_0_3_7__Transmission Out Shaft Speed__Col6 [-999.]
Idling_MinStats_0_0_3_8__Injector Control Pressure__Col7 [-999.]
Idling_MinStats_0_0_3_9__Engine Oil Pressure__Col8 [-999.]
Idling_MinStats_0_0_3_10__Battery Potential Voltage__Col9 [-999.]
Idling_MinStats_0_0_3_11__Altitude__Col10 [-999.]
Idling_MinStats_0_0_3_12__UTC_1Hz__Col11 [1392056457]
Idling_MinStats_0_0_3_13__Serial Number__Col12 ['10TANHFF1DS751419  ']
Idling_MinStats_0_0_3_14__Location__Col13 [27761.]
```

(a)

```
Idling_MinStats_0_0_3_1__Vehicle Speed__Col0 [0. 0. 0. 0. 0. 0.]
Idling_MinStats_0_0_3_2__Engine Speed__Col1 [672.125 659.375 682.875 679. 678.5 683.875]
Idling_MinStats_0_0_3_3__Fuel Rate__Col2 [0.4623011 0.43588388 0.43588388 0.43588388 0.43588388 0.43588388]
Idling_MinStats_0_0_3_4__Engine Oil Temperature__Col3 [-999. -999. -999. -999. -999. -999.]
Idling_MinStats_0_0_3_5__Engine Coolant Temperature__Col4 [ 82.4 195.8 195.8 199.4 194. 197.6]
Idling_MinStats_0_0_3_6__Transmission Oil Temperature__Col5 [ 59.7875 189.44376 192.14375 192.59375 193.04375 191.80624]
Idling_MinStats_0_0_3_7__Transmission Out Shaft Speed__Col6 [0. 0. 0. 0. 0. 0.]
Idling_MinStats_0_0_3_8__Injector Control Pressure__Col7 [1417.83 1347.646 1411.604 1417.83 1417.83 1411.604]
Idling_MinStats_0_0_3_9__Engine Oil Pressure__Col8 [16.2442 16.2442 17.98465 17.4045 17.98465 17.98465]
Idling_MinStats_0_0_3_10__Battery Potential Voltage__Col9 [25.35 27.5 27.5 27.5 27.45 27.5 ]
Idling_MinStats_0_0_3_11__Altitude__Col10 [1917.6509 1916.3385 2387.7952 2387.4673 2423.8845 2423.8845]
Idling_MinStats_0_0_3_12__UTC_1Hz__Col11 [1394979204 1394989015 1394990923 1394991523 1394992374 1394993517]
```

(b)

Figure 18: Examples of WSTAT values (a) no valid computations (b) valid computations.

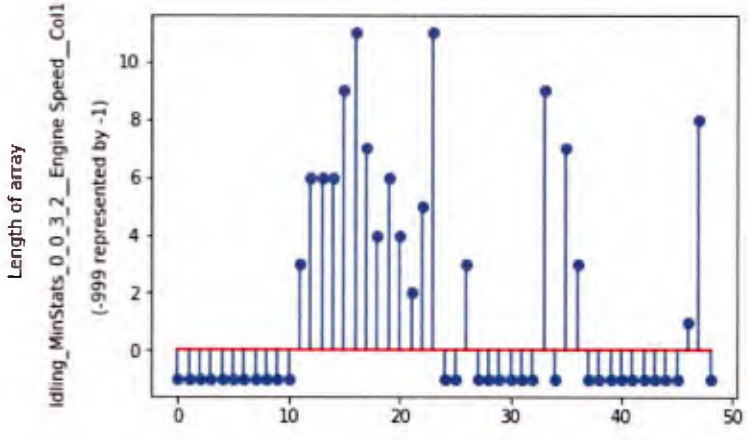


Figure 19: Interpretation of vehicle stat signals.

4.1.2 Marine Corps

4.1.2.1 Data storage

The Marine Corps HUMS data, provided to RIT by Pennsylvania State University (Penn State) at the request of ONR, consisted of 23 folders containing vehicle data, and one excel file containing meta-data about the vehicle folders. The excel file provided information such as vehicle serial number, the data volume in GB, whether the raw databus data was parsed, etc., and is shown below in Figure 20. One vehicle was removed from data collection due to vehicle problems (indicated in red in Figure 20), and did not have a vehicle folder in the dataset. Additionally, for further analysis, the five vehicles in yellow were excluded due to unknown errors experienced while attempting to load the .MAT files.

FLW Vehicles for FE-MTVR Testing

#	Vehicle	Variant	Serial #	Assigned Unit	Install Date	Uninstall Date	Data Volume (GB)	Parsed
1	MTVR1	AMK23 - Armadillo	MTVR1	FLW	28-Apr-15	2-Dec-15	6.11	Yes
2	MTVR2	MK23 - Armadillo (Up with Up Armored Cab...)	MTVR2	FLW	28-Apr-15	2-Dec-15	6.65	Yes
3	MTVR3	AMK23 - Armadillo	MTVR3	FLW	28-Apr-15	2-Dec-15	5.12	Yes
4	MTVR4	AMK25A1	MTVR4	FLW	28-Apr-15	2-Dec-15	11.00	Yes
5	MTVR5	AMK25A1	MTVR5	FLW	28-Apr-15	2-Dec-15	5.07	Yes
6	MTVR6	AMK25A1	MTVR6	FLW	28-Apr-15	2-Dec-15	8.77	Yes
7	MTVR7	AMK25A1	MTVR7	FLW	28-Apr-15	2-Dec-15	6.17	Yes
8	MTVR8	AMK25A1	MTVR8	FLW	28-Apr-15	2-Dec-15	14.60	Yes
9	MTVR9	AMK25A1	MTVR9	FLW	28-Apr-15	2-Dec-15	3.00	Yes
10	MTVR10	AMK25A1	MTVR10	FLW	28-Apr-15	2-Dec-15	9.48	Yes
11	MTVR11	AMK25A1	MTVR11	FLW	28-Apr-15	2-Dec-15	6.47	Yes
12	MTVR12	AMK25A1	MTVR12	FLW	28-Apr-15	2-Dec-15	7.46	Yes
13	MTVR13	AMK25A1	MTVR13	FLW	28-Apr-15	2-Dec-15	1.41	Yes
14	MTVR14	AMK25A1	MTVR14	FLW	28-Apr-15	2-Dec-15	6.27	Yes
15	MTVR15	AMK25A1	MTVR15	FLW	28-Apr-15	2-Dec-15	13.90	Yes
16	MTVR16	AMK25A1	MTVR16	FLW	N/A	N/A		
17	MTVR17	AMK25A1	MTVR17	FLW	28-Apr-15	2-Dec-15	9.27	Yes
18	MTVR18	AMK25A1	MTVR18	FLW	28-Apr-15	2-Dec-15	12.30	Yes
19	MTVR19	AMK25A1	MTVR19	FLW	28-Apr-15	3-Dec-15	7.92	Yes
20	MTVR20	AMK25A1	MTVR20	FLW	28-Apr-15	2-Dec-15	7.70	Yes
21	MTVR21	AMK25A1	MTVR21	FLW	28-Apr-15	2-Dec-15	4.52	Yes
22	MTVR22	AMK25A1	MTVR22	FLW	28-Apr-15	3-Dec-15	9.79	Yes
23	MTVR23	AMK25A1	MTVR23	FLW	28-Apr-15	3-Dec-15	14.30	Yes
24	MTVR24	AMK25A1	MTVR24	FLW	28-Apr-15	2-Dec-15	15.30	Yes

Figure 20: Received data.

Every vehicle folder contained two items: a sub-folder, filled with comma separated value (CSV) and MAT files, and a MAT file named 'Merged_Data_######.mat'. The CSV and MAT files in the subfolder were on the order of KBs and MBs. The CSV files contained raw databus data with file names indicating the start date and time of data collection. As indicated in Figure 20, all of the vehicles' databus data were parsed. The parsed data files (the MAT files) contain all of the data in MATLAB structures. Every CSV file had a corresponding MAT file with an identical file name. The 'Merged_Data_######.mat' utilized the serial number of the vehicle in place of the number signs. These file sizes were approximately 4-6 GBs and contained all the information from the separate MAT files within the sub-folder.

4.1.2.2 Data Format

The data in the files consisted of raw data captures from the two vehicle databuses: J1939 and J1708. The raw data was captured from the ECUs and contains a number of data parameters. The ECU sources were organized by subsystem in the vehicle and are detailed in Table 9 and Table 10. The J1939 databus specification defines Parameter Group Numbers (PGNs), which contain one or more Suspect Parameter

Numbers (SPNs), or measured variables. The J1708 databus specification defines Parameter Identifiers (PIDs), which are the measured variables. It should be noted that not all SPNs (signals) within a PGN may have been reporting, as the specification requires the entire PGN must be sent.

Table 9: J1939 Databus Data

Sources	Source Names	PGNs
SRC_135	FMPT *(unknown system, non-standard ID)*	PGN_65310, PGN_65311, PGN_65314
SRC_000	Engine #1	PGN_57599, PGN_60415, PGN_60671, PGN_61183, PGN_61443, PGN_61444, PGN_65247, PGN_65262, PGN_65263, PGN_65265, PGN_65266, PGN_65270
SRC_003	Transmission #1	PGN_61442, PGN_61445, PGN_65272, PGN_0
SRC_011	Brakes - System Controller	PGN_61441, PGN_65215, PGN_15, PGN_16, PGN_41, PGN_0
SRC_015	Retarder - Engine	PGN_61440
SRC_016	Retarder - Driveline	PGN_65275
SRC_051	Tire Pressure Controller	PGN_61441

Table 10: J1708 Databus Data

Sources	Source Names	PID List
MID_128	Engine #1	PID_100, PID_102, PID_105, PID_108, PID_110, PID_121, PID_168, PID_174, PID_175, PID_183, PID_184, PID_185, PID_187, PID_190, PID_194, PID_2, PID_245, PID_70, PID_71, PID_83, PID_84, PID_85, PID_86, PID_89, PID_91, PID_92, PID_128, PID_41, PID_0
MID_130	Transmission	PID_1, PID_162, PID_163, PID_191, PID_194, PID_136, PID_128
MID_136	Brakes, Power Unit	PID_151, PID_168, PID_194, PID_49, PID_84
MID_2		PID_254

4.1.2.3 Databus Activity in Vehicle Files

The source start times were recorded for each MAT file encountered in the unmerged file set. Databus activity was determined by looking for a J1939 or J1708 data structure in the file. Data collection across the vehicle set, and even within a single vehicle, was inconsistent. Half of the vehicles contained files without any databus data, and if the files contained databus data, the presence of both databuses was

not guaranteed. The total databus activity across the 18 parsed vehicles is shown in Figure 21. The overall bar height shows the total number of files collected, where the red bar represents files with J1939 data and the blue bar represents files with J1708 data. If all of the files contained databus data, the red and blue bars would be of even height and there would be no gray areas within the bars. The gray areas represent the files that exist but do not contain databus data.

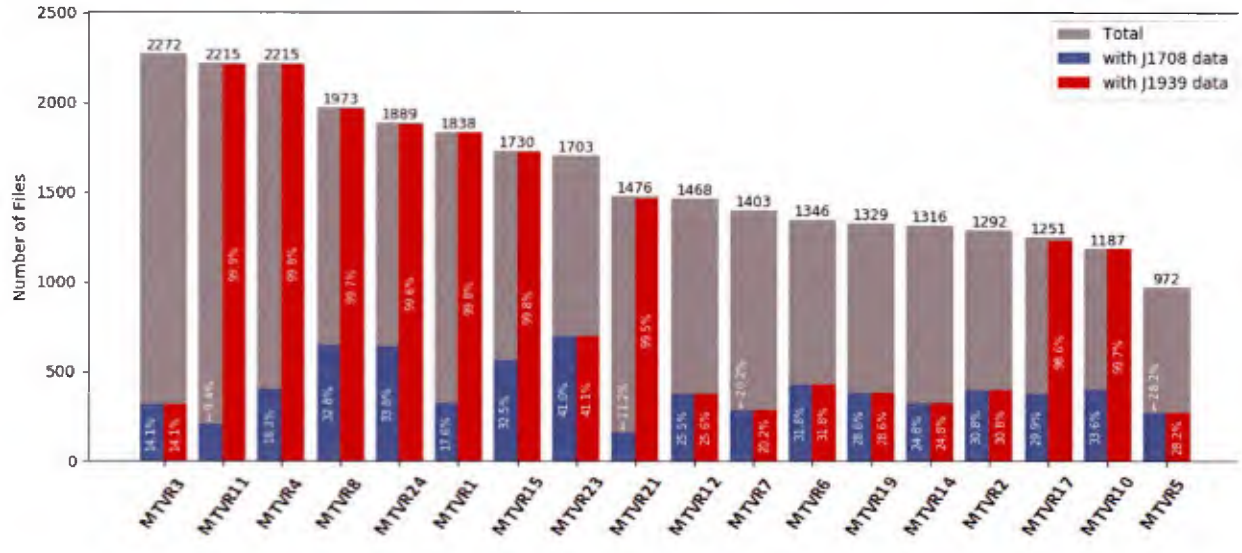


Figure 21: Databus activity on vehicles sorted by number of files.

Collecting a file with no databus activity at all does not necessarily represent a problem. For example, a maintainer may turn on the vehicle briefly to obtain the odometer reading, which may create a file with no data as the vehicle was turned off before data was collected. However, collection of data on only one databus indicates a problem with either the vehicle or the data collection system. Additionally, this absence of data from a databus does not appear to be a problem that occurs over a period of time and is then fixed. Rather it appears to come and go over the course of the data collected. To better illustrate this, the databus activity for two MTVRs is plotted over time in Figure 22 and Figure 23. Within each plot, a vehicle may have both active databuses (top line), only J1708 active (second line from top), only J1939 active (third line from top), or no active databuses (bottom line). Each dot represents a data file and indicates which databuses were active for that file. The first vehicle has activity on either both databuses, only J1939, or neither databus. The second vehicle either has no databus activity at all, or both databuses were active.

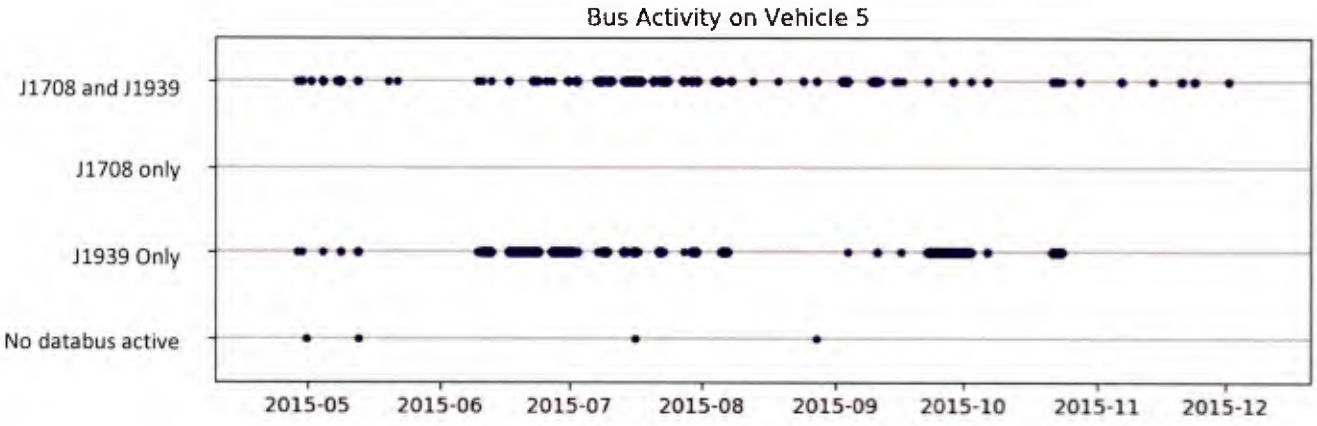


Figure 22: Databus activity on a vehicle where some files only had J1939 data.

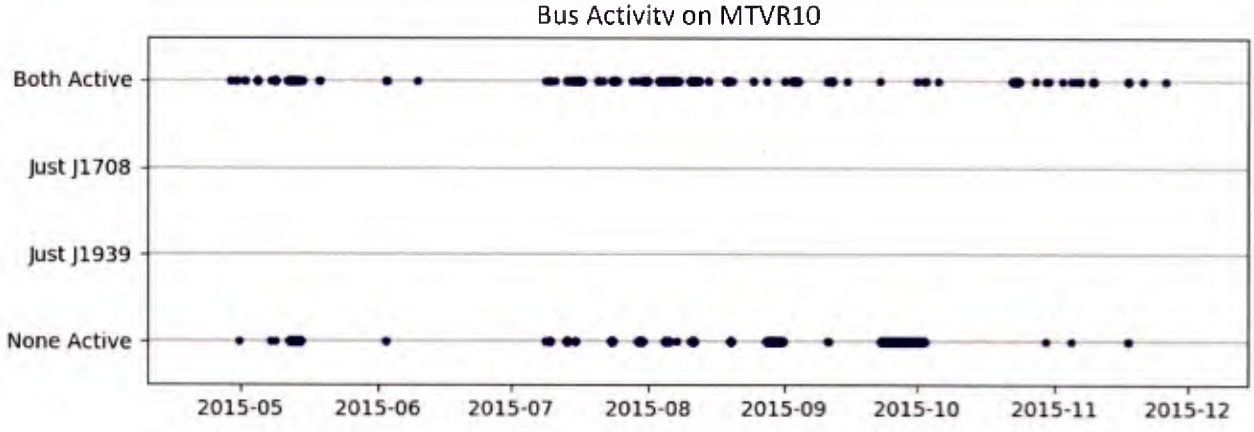


Figure 23: Databus activity on a vehicle with all or no data on a given file.

4.1.2.4 Data Timeline

Start times for each data record were found in the MAT structures source information. End times were considered to be the latest time stamp indicated by any databus parameter data element. Figure 24 provides step plots for each vehicle, where the vehicle is active (on) when the blue data is above the gray line, and inactive (off) when below the line. Data from the 18 vehicles covered a time range of about seven and a half months, between late April 2015 and early December 2015. Figure 25 shows a zoomed-in view of the vehicles' activity for a single day to provide better visibility to the on/off times.

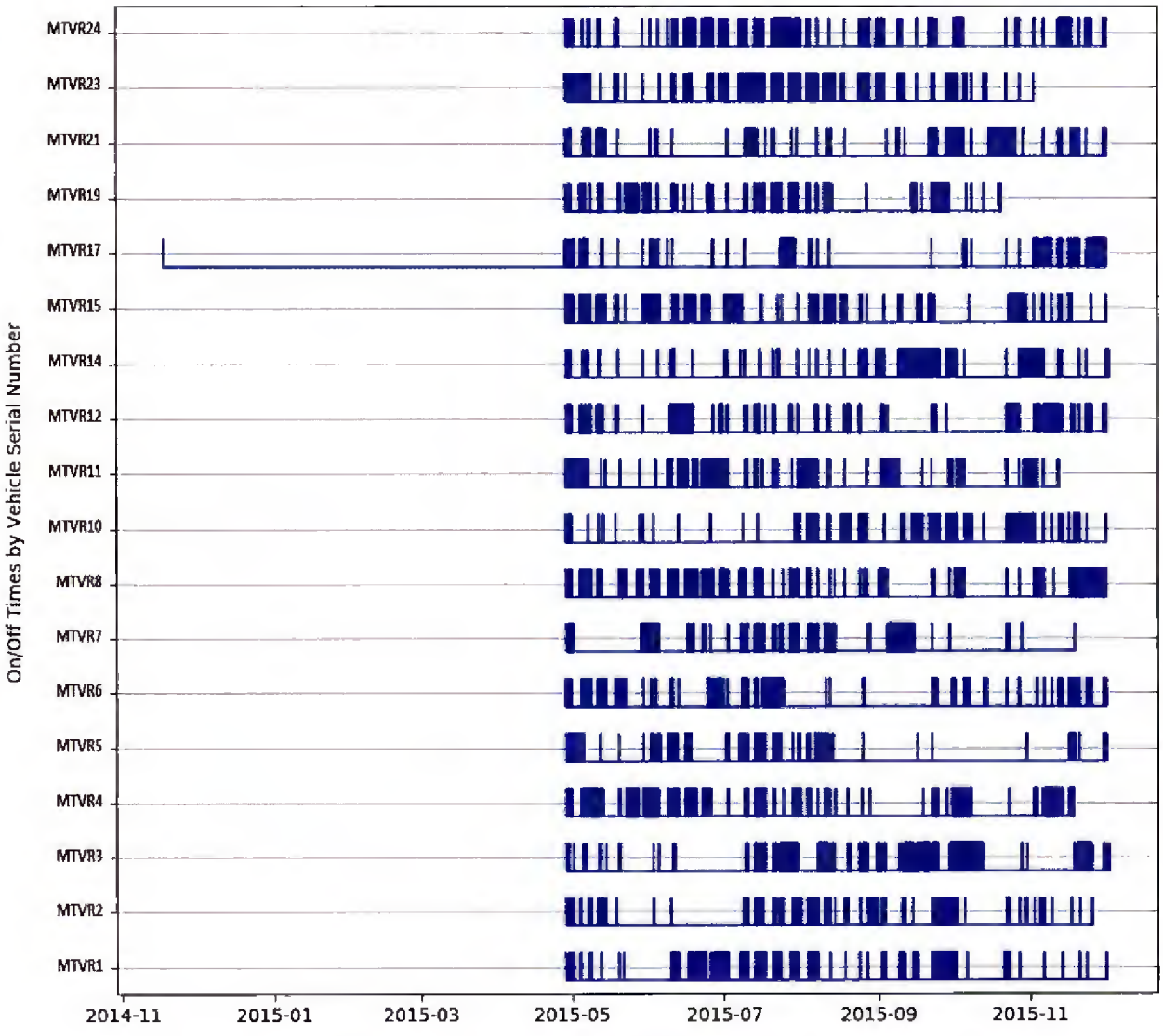


Figure 24: Key-on/off times for all vehicles as indicated in the data files

4.1.2.5 Granularity

As the MTVR data consisted of direct Controller Area Network (CAN) packet captures, the reporting rates of the signals are determined by the specifications for J1939 and J1708. Within these specifications, each signal has a defined reporting rate, e.g. 100ms for engine RPM. Although the data is collected at the highest available frequency, analysis will typically require the data to be aligned to a specific timestamped interval, similar to what is done with the army data at 1 Hz.

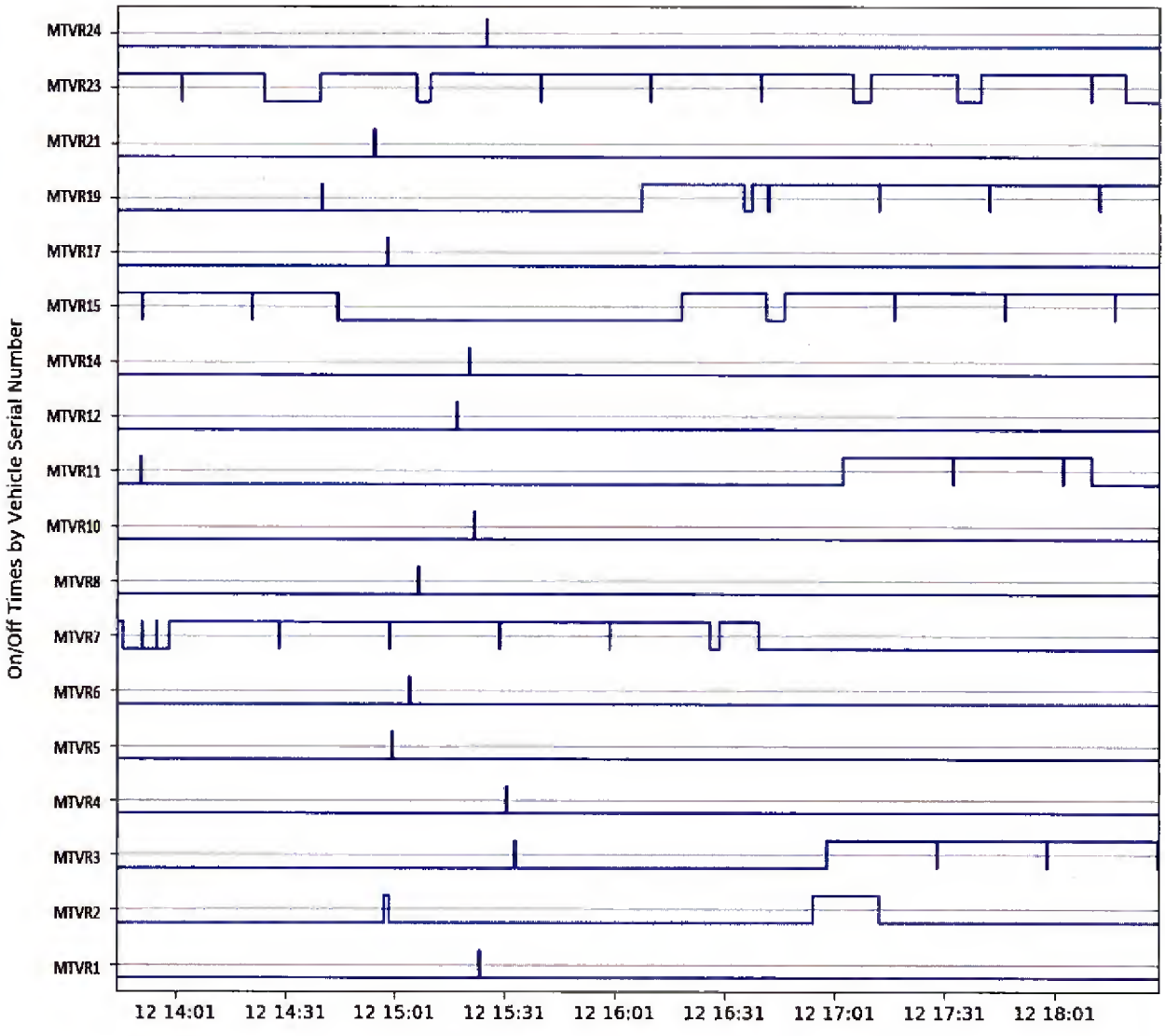


Figure 25: Zoomed in vehicle activity on a single day

4.1.2.6 *Data Availability*

As shown in Figure 21, each MTVR has a different number of overall data files, and contains different quantities of active databus data. Figure 26 provides a visual summarizing how many total hours of available data exists for each vehicle.

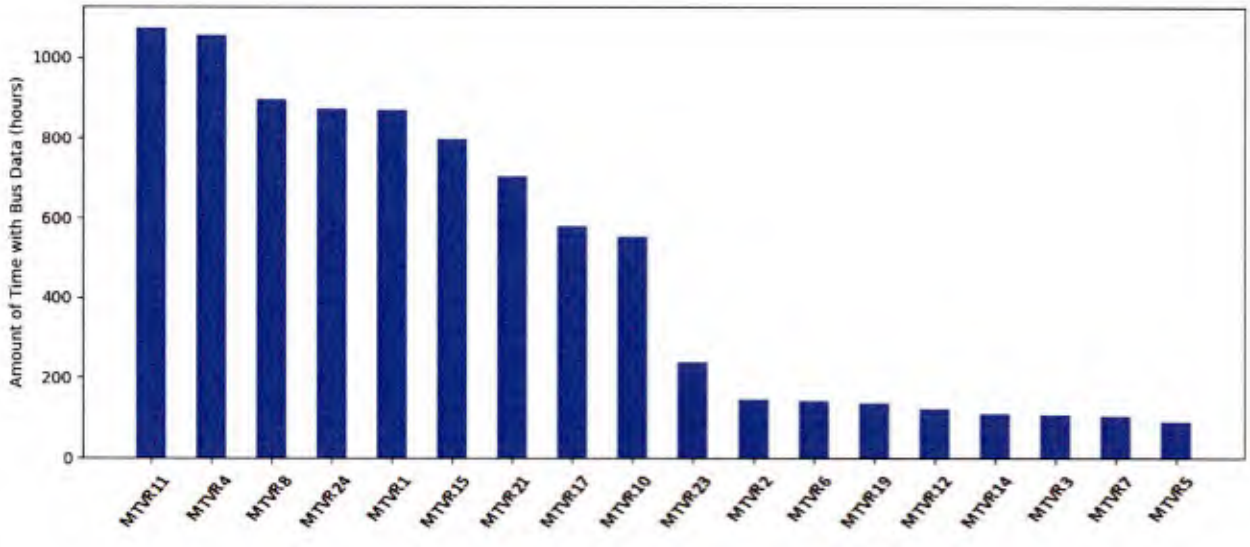


Figure 26: Total time of available data in hours for each vehicle sorted most to least time.

Additionally, each MTVR may contain a different number of signals that is likely dependent on the firmware versions of the ECUs and any additional hardware options. Figure 27 shows the number of signals found on each databus for all vehicles.

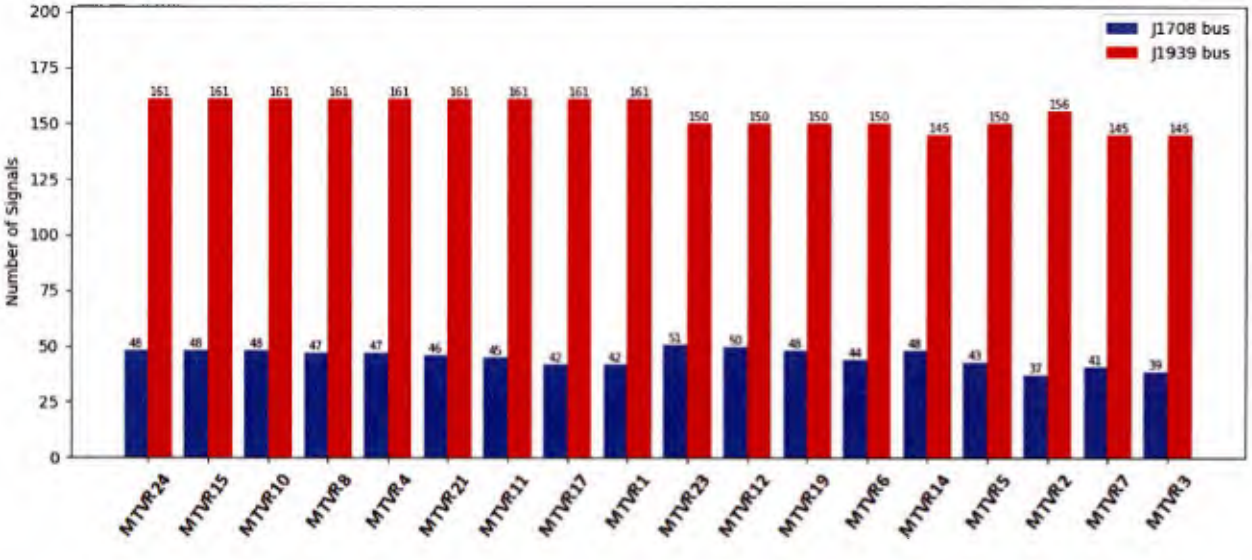


Figure 27: Number of signals on databuses sorted by total number of signals.

A further breakdown of the typical signals captured across the dataset is provided in Appendix B.

4.1.3 Navy

4.1.3.1 Data Storage

Navy HUMS data was contained in CSV files, with a total of 1,205 files contained in 39 data folders. Each folder contained CSV files with raw data, with some containing additional sub-folders organized by date and ship. All folders had a summary excel file (Figure 28) that contain metadata; information was not provided to interpret the metadata, and therefore it was not used in the analysis.

	Ship #	Scan Group Name	Asset Name	Trend Name	#tags in Trend	Final File Name (this needs to be standardized)
0	SHIP004	MER1_Scan_Groups	SHIP004_GTM_2A	070_GTM_2A_GEP	59	CMAS_SHIP004_GTM2A_2016_10_13
1	SHIP004	MER1_Scan_Groups	SHIP004_GTM_2A	071_GTM_2B_GEP	59	CMAS_SHIP004_GTM2A_2016_10_13
2	SHIP004	MER1_Scan_Groups	SHIP004_GTG_1	072_GTG_1_GEP	44	CMAS_SHIP004_GTG1_2016_10_13
3	SHIP004	MER1_Scan_Groups	SHIP004_MRG_2	073_MRG_2_GEP	36	CMAS_SHIP004_MRG2_2016_10_13
4	SHIP004	MER2_Scan_Groups	SHIP004_GTM_1A	070_GTM_1A_GEP	61	CMAS_SHIP004_GTM_1A_2016_10_13

Figure 28: Summary files in Navy data folders.

The filenames had identifiers separated by underscores, with each file name starting with "CMAS", followed by the ship number, the specific subsystem, an optional page number for some of the subsystems, and a date indicating which of the 39 top folders contained the files. Figure 29 shows the composition of these filenames by the meta-data in their name.

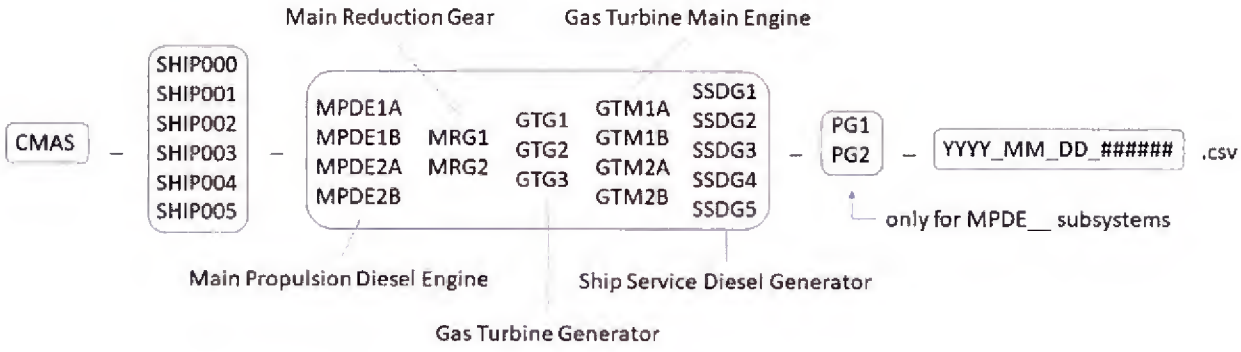


Figure 29: Filename composition by file meta-data.

Each subsystem had multiple subsystem IDs associated with them pertaining to the number of those subsystems in the ship. For instance, ship 0 and ship 5 contained main propulsion diesel engines with four engines; MPDE1A, MPDE2A, MPDE1B, and MPDE2B. Ships 2, 3, and 4 contained gas turbine engines with four engines; GTM1A, GTM2A, GTM1B, and GTM2B. This is shown in Figure 30. Ship 1 had very limited data and was not considered for subsequent analysis.

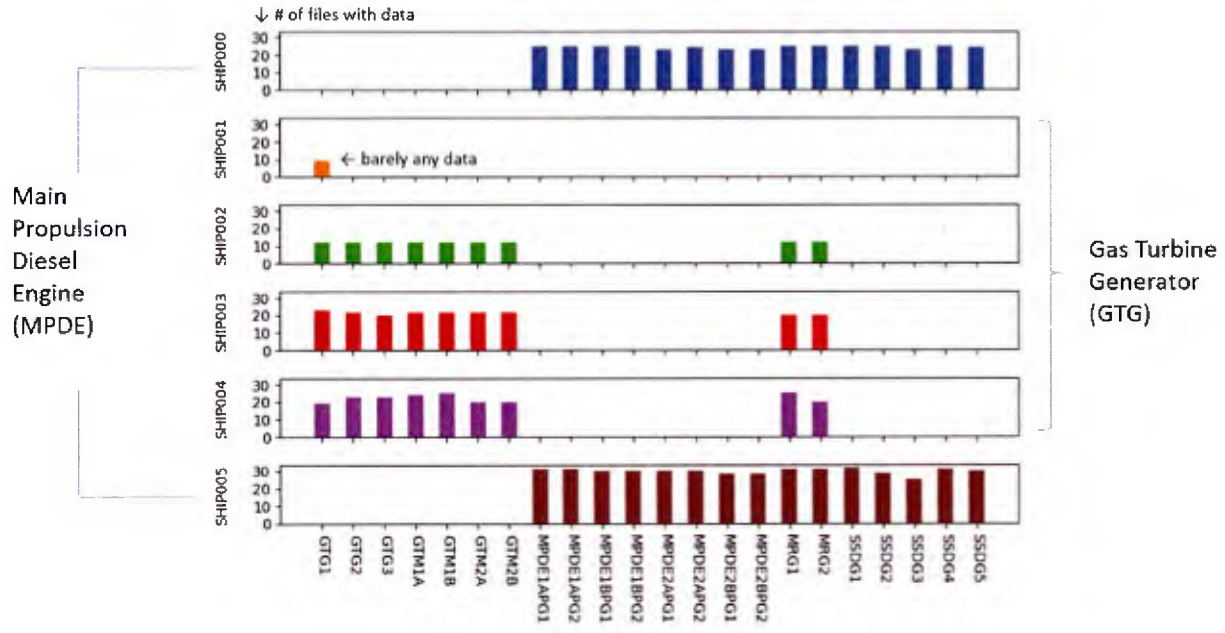


Figure 30: Subsystems of the ships by number of files with data.

The top 39 folders were labeled with dates, as shown in Figure 29, and those dates were used to plot the data folders along a timeline, shown in Figure 31, to understand the data distribution. In Figure 31, the dots represent dates when each ship had data reported. Ship 0 reported data on 25 days, while Ship 1 only reported data on nine days.

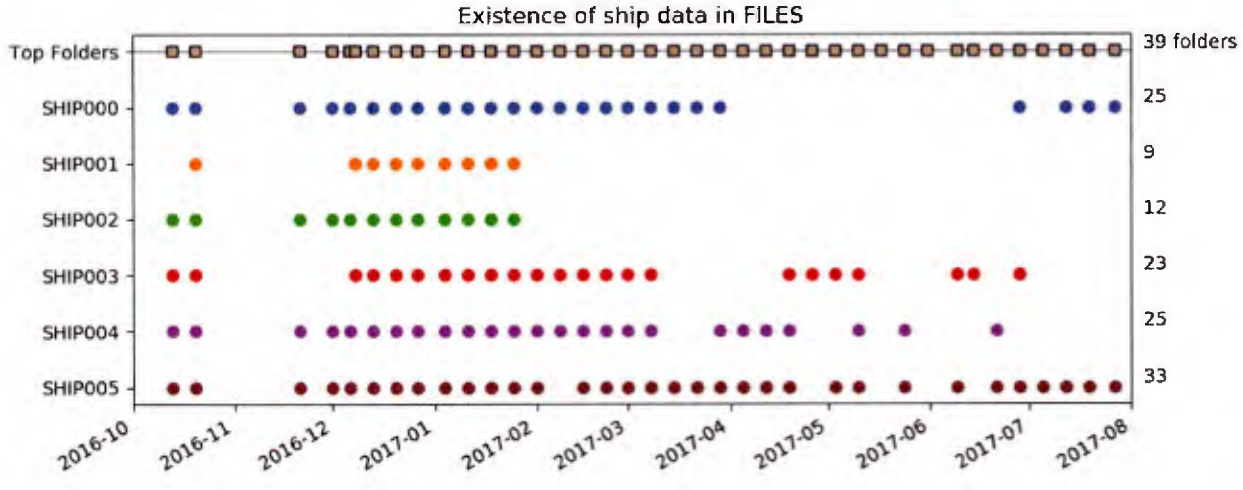


Figure 31: Data files within top folders by ship.

Specific data elements were analyzed for each ship. Figure 32 was built by layering start and end timelines of the raw data for each subsystem CSV file. In this figure, the darker the line, the more the data is duplicated during that timeframe. As shown, almost all of the data at the beginning of the data collection

period contains overlapping data. Further evidence of this overlap can be seen in Figure 33. There was a gap of data missing between April 2017 and mid-June 2017, which is apparent in all signals.

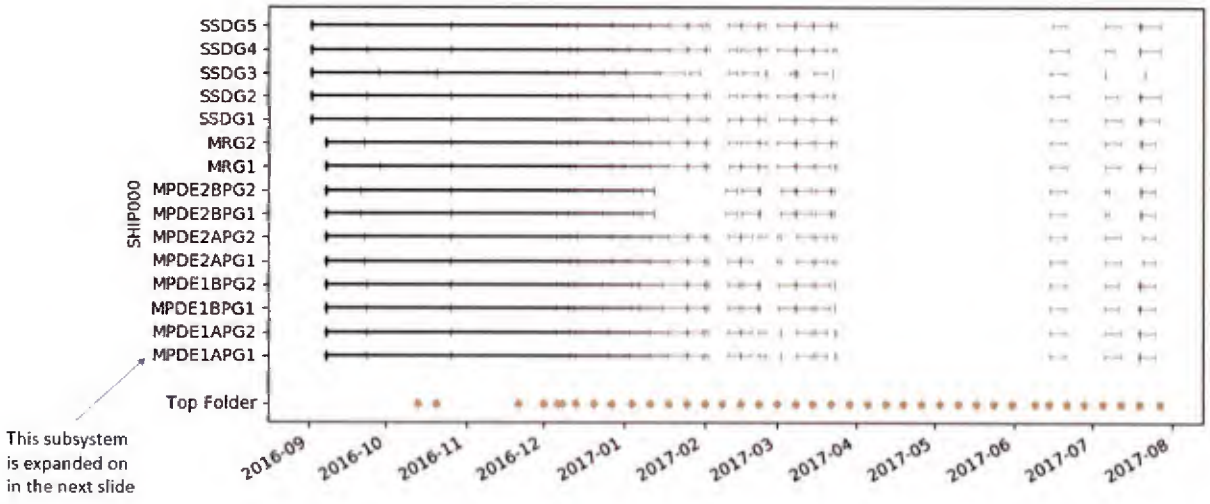


Figure 32: Data files timeline by subsystem of Ship 0.

Figure 33 shows the same plot for a single subsystem of Ship 0, the first page of one of the main propulsion diesel engines data (indicated with PG1). The upper section of this plot (above the title) was taken from Figure 32 and was further broken down to show the overlapping timelines of the CSV files separately, as well as which specific top folder contained the file. It is apparent from this view that the CSV files through January of 2017 contain the data from all previous CSV files up to that point. The overlapping data shown here were observed consistently across the data, i.e., there was typically significant data repetition between different data files.

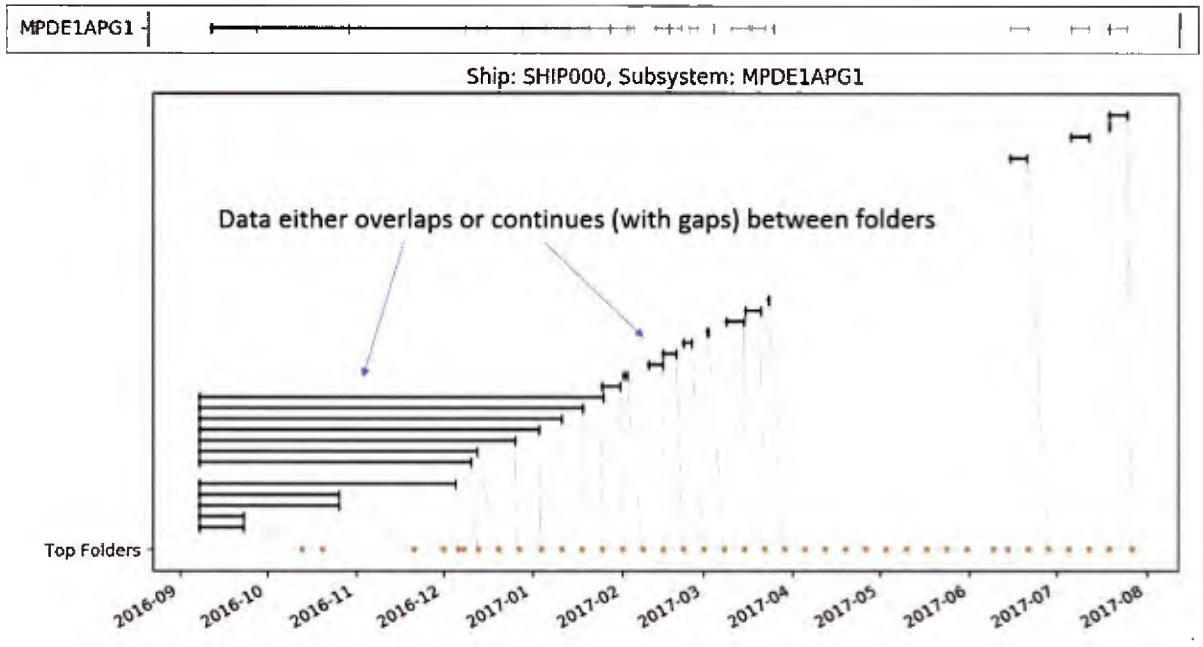


Figure 33: Data files timeline of a specific subsystem of Ship 0.

The repetition shown in Figure 33 (at the subsystem level) also occurs at the signal level as shown by an engine speed signal in Figure 34. For a full list of the PLS signals, see Appendix C.

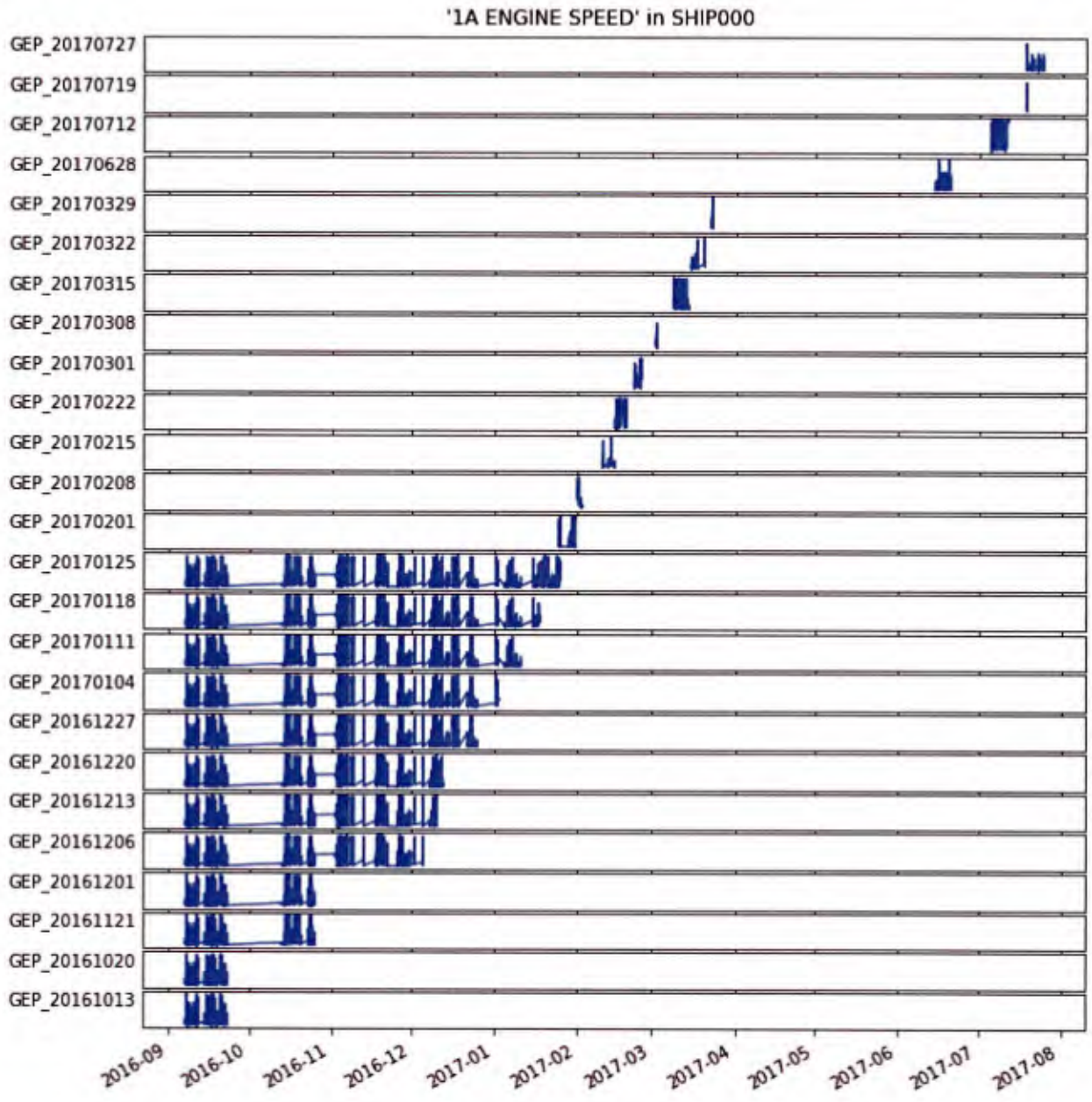


Figure 34: Data uniqueness over the top folders.

In order to facilitate data analysis, Python code was written to consolidate all ship data into a non-overlapping/non-repetitive data set, as shown for engine speed in Figure 35.

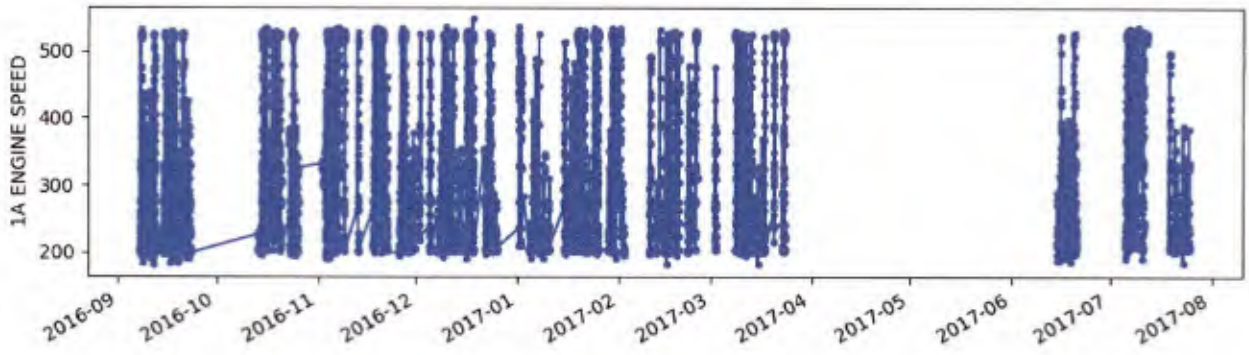


Figure 35: Concatenated unique engine speed data from ship 0.

4.1.3.2 Granularity

The interstitial spacing between data points was not constant. At some points, data were sampled every 1-5 seconds. Other times, there were 10-15 minutes between measurements. Figure 36 shows the time between samples for engine 1A on ship 0. The x-axis is time between measurements, and the y-axis is the number of data points recorded after that interval. About 100 records were observed with 1 second (10^0 on x-axis) between measurements (the leftmost point in the plot). The majority of points, however, were spaced by approximately 10-15 minutes, indicated by the spike at around 10^3 seconds. The highest point on this plot is around 8,000 records (8 x 10^3 on the y-axis) that were spaced by 10 minutes (6 x 10^2 on the x-axis). This distribution of time intervals was very similar for the other engines as well. From analysis of the data, it was determined that higher sampling rates are utilized when data is changing more rapidly in order to capture transient behavior. Sampling is more infrequent when data is steady, i.e. during long steady state (constant speed and load) conditions typical in ships.

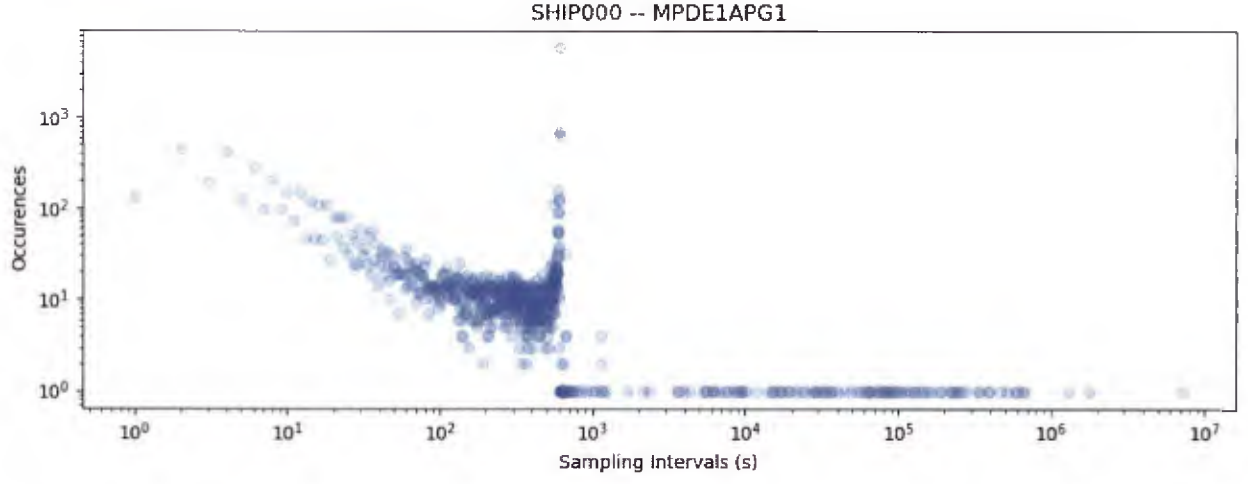


Figure 36: Sampling intervals of one specific subsystem of data.

4.1.3.3 Meta-Data Typos

The file naming convention was previously shown in Figure 29. However, a few files contained typos in the meta-data that were clearly different from the established format. Some filenames had extra or missing underscores in the name, while others were missing a 'PG1' or 'PG2' tag. The number of erroneous

filenames was small enough (47/1,205) that they were changed by hand and recorded in a table to keep record of the corrections. Table 11 shows a list of corrected typos in the filenames before and after the fix by top folder.

Table 11: Corrected filename typos.

Folder	Old Filename	New Filename
GEP_20161013	CMAS_SHIP000_MPDE1A_PG1__2016_10_13_154255.csv	CMAS_SHIP000_MPDE1A_PG1_2016_10_13_154255.csv
GEP_20161013	CMAS_SHIP000_MPDE2A_PG22016_10_13_154422.csv	CMAS_SHIP000_MPDE2A_PG2_2016_10_13_154422.csv
GEP_20161013	CMAS_SHIP000_MPDE2B_PG12016_10_13_154435.csv	CMAS_SHIP000_MPDE2B_PG1_2016_10_13_154435.csv
GEP_20161013	CMAS_SHIP005_MPDE2B_PD22016_10_13_153316.csv	CMAS_SHIP005_MPDE2B_PG2_2016_10_13_153316.csv
GEP_20161020	CMAS_SHIP005_MPDE1B_PD12016_10_20_105715.csv	CMAS_SHIP005_MPDE1B_PG1_2016_10_20_105715.csv
GEP_20161121	CMAS_SHIP000_MPDE_1A_2016_11_22_091712.csv	CMAS_SHIP000_MPDE1A_2016_11_22_091712.csv
GEP_20161121	CMAS_SHIP000_MPDE_1A_2016_11_22_091733.csv	CMAS_SHIP000_MPDE1A_2016_11_22_091733.csv
GEP_20161121	CMAS_SHIP000_MPDE_1B_2016_11_22_091752.csv	CMAS_SHIP000_MPDE1B_2016_11_22_091752.csv
GEP_20161121	CMAS_SHIP000_MPDE_1B_2016_11_22_091805.csv	CMAS_SHIP000_MPDE1B_2016_11_22_091805.csv
GEP_20161121	CMAS_SHIP000_MPDE_2A_2016_11_22_091828.csv	CMAS_SHIP000_MPDE2A_2016_11_22_091828.csv
GEP_20161121	CMAS_SHIP000_MPDE_2A_2016_11_22_091848.csv	CMAS_SHIP000_MPDE2A_2016_11_22_091848.csv
GEP_20161121	CMAS_SHIP000_MPDE_2B_2016_11_22_091909.csv	CMAS_SHIP000_MPDE2B_2016_11_22_091909.csv
GEP_20161121	CMAS_SHIP000_MPDE_2B_2016_11_22_091924.csv	CMAS_SHIP000_MPDE2B_2016_11_22_091924.csv
GEP_20161121	CMAS_SHIP005_MPDE_1A_2016_11_22_085841.csv	CMAS_SHIP005_MPDE1A_2016_11_22_085841.csv
GEP_20161121	CMAS_SHIP005_MPDE_1A_2016_11_22_085915.csv	CMAS_SHIP005_MPDE1A_2016_11_22_085915.csv
GEP_20161121	CMAS_SHIP005_MPDE_1B_2016_11_22_085939.csv	CMAS_SHIP005_MPDE1B_2016_11_22_085939.csv
GEP_20161121	CMAS_SHIP005_MPDE_1B_2016_11_22_090640.csv	CMAS_SHIP005_MPDE1B_2016_11_22_090640.csv
GEP_20161121	CMAS_SHIP005_MPDE_2A_2016_11_22_091044.csv	CMAS_SHIP005_MPDE2A_2016_11_22_091044.csv
GEP_20161121	CMAS_SHIP005_MPDE_2A_2016_11_22_091152.csv	CMAS_SHIP005_MPDE2A_2016_11_22_091152.csv
GEP_20161121	CMAS_SHIP005_MPDE_2B_2016_11_22_091256.csv	CMAS_SHIP005_MPDE2B_2016_11_22_091256.csv
GEP_20161121	CMAS_SHIP005_MPDE_2B_2016_11_22_091315.csv	CMAS_SHIP005_MPDE2B_2016_11_22_091315.csv
GEP_20161121	CMAS_SHIP000_MRG_1_2016_11_22_091942.csv	CMAS_SHIP000_MRG1_2016_11_22_091942.csv
GEP_20161121	CMAS_SHIP000_MRG_2_2016_11_22_092001.csv	CMAS_SHIP000_MRG2_2016_11_22_092001.csv
GEP_20161121	CMAS_SHIP005_MRG_2_2016_11_22_091423.csv	CMAS_SHIP005_MRG2_2016_11_22_091423.csv
GEP_20161121	CMAS_SHIP005_MRG_12016_11_22_091345.csv	CMAS_SHIP005_MRG1_2016_11_22_091345.csv
GEP_20161121	CMAS_SHIP005_SSDG_12016_11_22_091437.csv	CMAS_SHIP005_SSDG1_2016_11_22_091437.csv
GEP_20161121	CMAS_SHIP005_SSDG_22016_11_22_091504.csv	CMAS_SHIP005_SSDG2_2016_11_22_091504.csv
GEP_20161121	CMAS_SHIP005_SSDG_32016_11_22_091521.csv	CMAS_SHIP005_SSDG3_2016_11_22_091521.csv
GEP_20161121	CMAS_SHIP005_SSDG_42016_11_22_091544.csv	CMAS_SHIP005_SSDG4_2016_11_22_091544.csv
GEP_20161121	CMAS_SHIP005_SSDG_52016_11_22_091618.csv	CMAS_SHIP005_SSDG5_2016_11_22_091618.csv
GEP_20161121	CMAS_SHIP000_MPDE1A_2016_11_22_091712.csv	CMAS_SHIP000_MPDE1A_PG1_2016_11_22_091712.csv
GEP_20161121	CMAS_SHIP000_MPDE1A_2016_11_22_091733.csv	CMAS_SHIP000_MPDE1A_PG2_2016_11_22_091733.csv
GEP_20161121	CMAS_SHIP005_MPDE2B_2016_11_22_091256.csv	CMAS_SHIP005_MPDE2B_PG1_2016_11_22_091256.csv
GEP_20161121	CMAS_SHIP005_MPDE2B_2016_11_22_091315.csv	CMAS_SHIP005_MPDE2B_PG2_2016_11_22_091315.csv
GEP_20161121	CMAS_SHIP005_MPDE2A_2016_11_22_091044.csv	CMAS_SHIP005_MPDE2A_PG1_2016_11_22_091044.csv
GEP_20161121	CMAS_SHIP005_MPDE2A_2016_11_22_091152.csv	CMAS_SHIP005_MPDE2A_PG2_2016_11_22_091152.csv
GEP_20161121	CMAS_SHIP005_MPDE1B_2016_11_22_085939.csv	CMAS_SHIP005_MPDE1B_PG1_2016_11_22_085939.csv
GEP_20161121	CMAS_SHIP005_MPDE1B_2016_11_22_090640.csv	CMAS_SHIP005_MPDE1B_PG2_2016_11_22_090640.csv
GEP_20161121	CMAS_SHIP005_MPDE1A_2016_11_22_085841.csv	CMAS_SHIP005_MPDE1A_PG1_2016_11_22_085841.csv
GEP_20161121	CMAS_SHIP005_MPDE1A_2016_11_22_085915.csv	CMAS_SHIP005_MPDE1A_PG2_2016_11_22_085915.csv
GEP_20161121	CMAS_SHIP000_MPDE2B_2016_11_22_091909.csv	CMAS_SHIP000_MPDE2B_PG1_2016_11_22_091909.csv
GEP_20161121	CMAS_SHIP000_MPDE2B_2016_11_22_091924.csv	CMAS_SHIP000_MPDE2B_PG2_2016_11_22_091924.csv
GEP_20161121	CMAS_SHIP000_MPDE2A_2016_11_22_091828.csv	CMAS_SHIP000_MPDE2A_PG1_2016_11_22_091828.csv
GEP_20161121	CMAS_SHIP000_MPDE2A_2016_11_22_091848.csv	CMAS_SHIP000_MPDE2A_PG2_2016_11_22_091848.csv
GEP_20161121	CMAS_SHIP000_MPDE1B_2016_11_22_091752.csv	CMAS_SHIP000_MPDE1B_PG1_2016_11_22_091752.csv
GEP_20161121	CMAS_SHIP000_MPDE1B_2016_11_22_091805.csv	CMAS_SHIP000_MPDE1B_PG2_2016_11_22_091805.csv
GEP_20170329	CMAS_SHIP000_MPDE2B_2017_03_29_093653.csv	CMAS_SHIP000_MPDE2B_PG2_2017_03_29_093653.csv

4.1.4 Comparative Analyses across Different Platforms

A high-level review of the HUMS data sets is shown in Table 12. The Army CDF data format is a well-documented, compressible format that is suitable for collection and transfer of extensive data off the vehicle to a back-end data analysis system. Additional documentation of the interpretation of DTCs within the CDF format is necessary, however, as the format does not appear to follow the databus specification. The CDF files are easily read in a traditional data analysis software, such as Python, through existing libraries.

Although the Navy data has significantly more engine signals, the long duration between samples makes modeling of the system behavior difficult. Conversely, the Army and Marine Corps data consists only of typical J1939 and J1708 databus signals reported by the vehicle Original Equipment Manufacturers (OEMs). This is a limited subset of data, but it is typically reported and stored at higher rates than the Navy data. Obtaining additional proprietary signal data through the OEMs, or adding selected signals to the HUMS system, could enhance the PHM capability of those systems.

From a modeling perspective, the best option would be a high number of signals reported at a very frequent rate. Using the engine as an example system, the smaller number of signals provided by the FMTV and MTVR means that the ability to correlate between signals and make meaningful system level comparisons is reduced. However, the with the naval ship data, there are significantly more signals, but the ability to model some of the higher speed system dynamics is limited by the lower data sampling rates.

Although the data collection and storage mechanisms are significantly different, the one significant piece of data that is enabled through all systems is the tracking of usage histories. The usage of specific components or subsystems (e.g. engine or transmission) and the potential to link that information to life-cycle tracking is beneficial.

Table 12 - Overview of HUMS Data sets

	Army	Navy	Marine Corps	Comments
Data format	CDF	CSV	CSV/MAT	The CDF format is well documented and provides an easily compressible format. The Marine Corps MAT file are easy to work with as the data is well structures, but the associated file size is too large.
Number of Vehicles Analyze (Fleet size)	447	2	18	For the Army and Marine Corps vehicles, only vehicles with valid data were considered. For the Navy data, only ships with diesel engines were considered.
Total Number of Signals	~141	159	~202	The number of signals varies slightly on the Army and Marine Corps vehicles depending on what the databus is reporting and what the data acquisition system is set-up to record. The Navy list consists of signals specifically for the LPD and does not include the guided missile destroyers or cruisers.

	Army	Navy	Marine Corps	Comments
Time Range of Data	19 Dec 2012 – 14 May 2014	7 Sep 2016 – 26 Jul 2017	28 Apr 2015 – 3 Dec 2015	
Total Engine Hours Across the Fleet	285,929	46,061	8,615	
Number of Engine Signals	~20	~79	~12-15 associated signals	The Naval ships have significantly more signals being collected for the engines (ex. exhaust temperatures for each cylinder). The Army and Marine Corps data are similar, with the difference in signals attributed to the Army data also containing a J1708 databus.
Sampling rate	1 sec	Varying, storage of signals ranges from seconds to minutes depending on the variability of the data itself	~100 ms	The Marine Corps data is direct capture and therefore is the most frequent; the loss of resolution in Army data is minor. The extended sampling times of some of the Navy data will impact the ability to model those systems.
DTCs	Exist	Do not exist	Exist	Diagnostic Trouble Codes exist within the Army and Marine Corps data, but it should be noted that the Army data conversion of the codes is not intuitive. The Marine Corps DTCs may be converted utilizing the databus specification

4.2 Analysis of Maintenance Data

A key component of many prognostic development approaches is ground truth data. Understanding when a vehicle is operating normally and when the vehicle is in need of repair can aid in developing training, testing, and validation data sets. RIT requested maintenance data for all three of the platforms for which HUMS data was supplied. The Army provided maintenance data that included information of the fault, labor hours, and parts replaced by work order. No maintenance data was supplied for the Marine Corps MTVR. The Navy chose to provide HUMS data sets that were associated with specific events. In particular, the naval ship faults were classified as one of three types, and each data set consisted of a ship ID, a fault type, a fault date, and the HUMS data leading up to the date. The subsections below focus mostly on the FMTV, as this vehicle had maintenance work order data. The maintenance data was used to guide some of the follow-on data analysis investigations and identifies which systems and components most warrant PHM coverage.

4.2.1 Army

TARDEC supplied RIT with maintenance data for five families of vehicles: the Family of Medium Tactical Vehicles (FMTV), the Heavy Expanded Mobility Tactical Truck (HEMTT), the Heavy Equipment Transporter (HET), military line haul tractor (LineHaul), and the Palletized Load System (PLS). The maintenance data is a capture of the data collected in Global Combat Support System-Army (GCSS-Army). The data includes, but is not limited to, the serial number of the asset being maintained, the fault date, a description of the

fault, part replacement information, labor hours, and the completion date. The focus of the analysis was the FMTV, since HUMS data also exists for that platform.

This project focused on Condition Based Maintenance, including Prognostics Health Management. Typical strategies for maintenance are based on frequency and severity of the failure, as shown in Figure 37. PHM is typically applied to high severity failures with a relatively low occurrence.

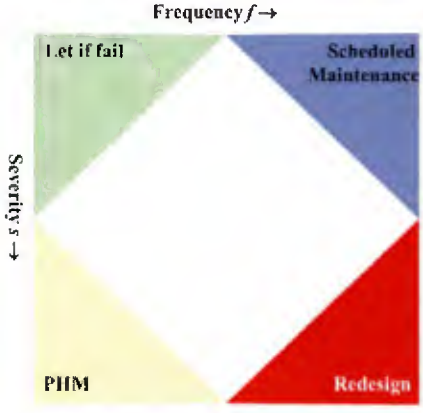


Figure 37 - Maintenance Strategies based on frequency and severity of failures used in the automotive industry.

Severity of failures in military ground assets is typically tied to the deadline status of the asset and how long the asset is out of service, as measured by readiness at a particular point in time and availability over some time interval. A deadline status for the maintenance event does not exist within the supplied TARDEC data. In previous research programs conducted by RIT, deadline information existed in separate systems that track unit readiness. Since the data was not available for this project, in an attempt to understand downtime, an evaluation of the fault date and the work order completion date was performed. Unfortunately, utilizing these dates does not provide an indication of the downtime as it was determined through evaluation of the HUMS data that the vehicles continued to operate within that time period. Therefore, the analysis of the maintenance events focused on cost and frequency of repair.

4.2.1.1 Maintenance Data

The maintenance data was structured into records - essentially rows of information, with a single work order typically consisting of multiple data records. The typical work order has a first record for initial inspection of the problem and a last record for final inspection. The records in between consist of a record per component that was replaced or operation that was performed. A simplified example is shown in Table 13 for a "DRIVER AND PASSAGER SIDE DOOR HANDLES DAMAGED" fault. In addition, each record in a work order has the same fault description, fault date, serial number, and work order number.

Table 13 - Typical rows in a maintenance record for a work order

Correction Narrative	Action Description	Part NSN	Part Nomenclature	Part Qty	Total Man Hours	Task #	Total Cost
REPLACED BOTH DOOR HANDLES	Initial Inspection				0.2	1	
REPLACED BOTH DOOR HANDLES	Replaced	2540013757994	LATCH,DOOR, VEHICULA	1	0.3	2	57.73
REPLACED BOTH DOOR HANDLES	Replaced	2540013757995	LATCH,DOOR, VEHICULA	1	0.3	3	51.89
REPLACED BOTH DOOR HANDLES	Replaced	5342014829230	LINKAGE CLIP	6	0.3	4	47.16
REPLACED BOTH DOOR HANDLES	Replaced	3040013757321	ACTUATOR, MECHANICAL, N	1	0.3	5	43.33
REPLACED BOTH DOOR HANDLES	Replaced	3040013757322	ACTUATOR, MECHANICAL, N	1	0.3	6	41.57
REPLACED BOTH DOOR HANDLES	Replaced	5305014340879	SCREW, MACHINE	6	0.3	7	0.78
REPLACED BOTH DOOR HANDLES	Replaced	3040013757397	CONNECTING LINK, RIGID	2	0.3	8	7.8
REPLACED BOTH DOOR HANDLES	Replaced	3040013756341	CONNECTING LINK, RIG	2	0.3	9	8.32
REPLACED BOTH DOOR HANDLES	Replaced	2540013763998	HANDLE, DOOR, VEHICUL	1	0.3	10	212.33

Correction Narrative	Action Description	Part NSN	Part Nomenclature	Part Qty	Total Man Hours	Task #	Total Cost
REPLACED BOTH DOOR HANDLES	Replaced	2540013763999	HANDLE, DOOR, VEHICUL	1	0.3	11	250.3
REPLACED BOTH DOOR HANDLES	Replaced	5342014800093	CONTROL ROD	2	0.3	12	6.6
REPLACED BOTH DOOR HANDLES	Replaced	5342014800092	CONTROL ROD	2	0.3	13	13.84
REPLACED BOTH DOOR HANDLES	Replaced	3040013757323	ACTUATOR, MECHANICAL, N	1	0.3	14	152.13
REPLACED BOTH DOOR HANDLES	Replaced	3040013776805	ACTUATOR, MECHANICAL, N	1	0.3	15	139.64
REPLACED BOTH DOOR HANDLES	Final Inspection				0.3	16	

An initial analysis was performed on the data to identify the extent of the maintenance data. The focus of this initial analysis was to identify the date range of the data, as well as a number of work orders and replacement part National Stock Numbers (NSNs) ordered. In Table 14, a breakdown of the extent and quantity of data by family type is shown. However, the maintenance contains a number of entries where the work order number was entered as a form of "Not Available". Therefore, the number of distinct work orders includes only a single work order count for the "Not Available (N/A)" entries. In order to identify distinct work orders labeled as "Not Available", the combination of distinct serial numbers and fault dates was utilized. The counts of these work orders and how many required ordering parts is highlighted in orange in the table. For example, the FMTV has 956 distinct work order numbers, but one of those work order numbers is "Not Available". The work orders labeled as "Not Available" may be able to be further broken down into 713 distinct work orders utilizing an analysis of the serial number and fault date of the work order. Out of these work orders, 820 of the 956 work orders required parts and an additional 393 of the N/A work orders required parts. It should be noted that information relating to the faults and vehicles may not be lost without a work order number; however, the analysis is further complicated without this information.

Table 14 - Extent of the TARDEC Supplied Maintenance Data

Vehicle Family	First Fault Date	Last Fault Date	Number of Distinct Vehicle Serial #'s	Number of Distinct Work Orders	Number of Work Orders where parts were required	Number of Work Orders Labeled N/A	Number of N/A Work Orders where parts were required	Number of Distinct NSNs ordered	Total Number of Parts Ordered
FMTV	10/26/2012	3/3/2015	521	956	820	713	393	687	5341
HEMTT	10/22/2012	2/25/2015	674	1695	1476	1267	759	987	8296
HET	11/14/2012	2/12/2015	87	206	184	177	84	231	881
LineHaul	11/28/2012	2/26/2015	107	377	343	254	151	271	2468
PLS	12/12/2012	1/14/2015	132	296	252	182	115	353	1618

Additionally, the labor hours and parts costs were summarized by family type, as shown in Table 15. Once again, the totals include all work orders (including those labeled as "Not Available"). The total maintenance cost is computed as:

$$Total\ Maint.Cost = Total\ Parts\ Cost + (Total\ Labor\ Hours * \frac{\$45}{hr} labor\ rate) \quad (1)$$

Similarly, the totals were computed for all N/A work orders, as well as the percentage of total cost that the N/A work orders represent, to show the amount of costs that are more difficult to trace.

Table 15 - Maintenance Costs by Vehicle Family

Vehicle Family	Total Labor Hours	Total Parts Cost	Total Maint. Cost	N/A WO Labor Hours	N/A WO Parts Cost	N/A WO Maint. Cost	% of Total Cost on N/A WO
FMTV	15279.9	$ 592,597.26	$ 1,280,192.76	3069.6	$ 135,126.86	$ 273,258.86	21.3%
HEMTT	26657.4	$ 2,635,221.77	$ 3,834,804.77	6469.7	$ 917,665.56	$ 1,208,802.06	31.5%
HET	3999.2	$ 425,968.18	$ 605,932.18	1049.2	$ 48,745.20	$ 95,959.20	15.8%
LineHaul	6129.4	$ 242,814.15	$ 518,637.15	454.5	$ 45,483.10	$ 65,935.60	12.7%
PLS	5199.8	$ 298,937.58	$ 532,928.58	590.6	$ 41,426.74	$ 68,003.74	12.8%

Unfortunately, key usage information is missing in the maintenance data, i.e. vehicle mileage and/or engine hours. Without this data, it is difficult to put the costs in cost per mile context for fleet comparisons (across fleets or over time). Instead, the best normalization method available is to evaluate the data based on the number of service years and number of vehicles represented by the work orders. In an integrated data environment with HUMS data, this could easily be calculated.

$$Number\ of\ Service\ Years = \frac{Number\ of\ Days\ between\ First\ and\ Last\ Fault\ Date}{365.25\ Days} \quad (2)$$

Table 16 - Maintenance Costs across Army Vehicles

Vehicle Family	Number of Vehicles	Number of Service Years	WO per Vehicle per Year	Parts per Vehicle per Year	Labor Hours per Vehicle per Year	Total Cost per vehicle per year
FMTV	521	2.35	1.36	4.36	12.49	$ 1,046.05
HEMTT	674	2.34	1.87	5.25	16.88	$ 2,427.79
HET	87	2.24	1.96	4.51	20.48	$ 3,102.37
LineHaul	107	2.24	2.62	10.27	25.52	$ 2,159.08
PLS	132	2.09	1.73	5.87	18.86	$ 1,932.74

Per the analysis in Table 16, the FMTV has the lowest cost per vehicle per year and the HET has the highest, almost three times the FMTV cost; no cause for this difference in apparent maintenance cost was identified.

4.2.1.2 Army Maintenance Data Drill-down

A deeper analysis of the maintenance data was performed to support inferences around high value areas for CBM, in particular to: 1) Identify frequent unscheduled maintenance events and 2) Identify high cost maintenance events. The latter can be further evaluated by total cost, high value component cost, and high labor cost. In order to attain the first goal, RIT needed to identify the work orders that included scheduled maintenance events. From a review of the data, it was determined that the Fault Description could be used to identify scheduled maintenance by filtering for the following search terms: "annual", "ennial", "5000", and "semi". Using these filters, the number of Work Orders containing scheduled

maintenance were determined, as shown in Table 17. In this analysis, scheduled maintenance is not considered toward implementation of a CBM program. However, scheduled maintenance activities should periodically be reviewed to identify areas where schedules may be modified or implantation of condition monitoring may be applied, i.e. oil quality monitoring vs. changes every 5,000 miles.

Table 17 - Work Orders Containing Scheduled Maintenance

Vehicle Family	Number of Work Orders Containing Scheduled Maintenance	Percentage of Total Work Orders Containing Scheduled Maintenance
FMTV	386	40.4%
HEMTT	465	27.4%
HET	61	29.6%
LineHaul	131	34.7%
PLS	142	48.0%

Interestingly, the FMTV has the lowest number of work orders per year, as well as the lowest cost per vehicle per year, followed by the PLS. The PLS and the FMTV have the highest percentage of scheduled work orders, suggesting that the cost of maintenance on the other platforms is being driven by unscheduled maintenance.

For the parts and labor analysis, the full data set, as well as a dataset removing scheduled maintenance work orders, was analyzed. The goal of the separate analysis was to identify and remove parts that are typically replaced during scheduled maintenance.

4.2.1.3 Subsystems Breakdown

As PHM models typically relate to functional importance and failure rate/severity, it is useful to break the maintenance data down by system or subsystem. The data provided did not explicitly include a subsystem breakdown; however, the first two digits of the Functional Work Group appear to indicate the subsystem of the component. Table 18 shows a breakdown of the subsystems from the Functional Workgroup. All other two digit combinations were labeled as "Other" or N/A. Major subsystems were analyzed separately.

Table 18 - Subsystem breakdown

First Two Digits of Functional Work Group	**Subsystem**
01	Engine
03	Fuel
04	Exhaust
05	Cooling
06	Electrical
07	Transmission
08	Drivetrain
09	Drivetrain

First Two Digits of Functional Work Group	Subsystem
10	Axles
11	Axles
12	Compressed Air
13	Wheel Hub/Tires
14	Steering
15	Towing
16	Suspension
18	Troop Compartment
20	Crane/Hoist/Towing
24	Hydraulics
33	Comms
52	A/C

4.2.1.4 FMTV

As RIT only had HUMS data for the FMTV, the drill down maintenance analysis focused on the FMTV. The data was analyzed by subsystem, both with and without scheduled work orders and was only evaluated within the time range of the HUMS data (19 Dec 2012 – 14 May 2014). Each subsystem was analyzed separately to provide a list of top maintenance items. For the purposes of the analysis, the labor cost was assumed to be $45/hr.

A quick analysis of the maintenance data was performed to identify which subsystems have the highest impact to cost and labor. Table 19 shows the subsystems broken down by labor hours, parts cost and total cost, with the systems sorted by total cost. The last column provides visibility into the degree to which the total cost is driven by labor for each subsystem. Further analysis included a separate analysis of each subsystem with a total cost over $50,000, and an additional analysis of the remaining subsystems.

Table 19 - Subsystems order by total cost of replacement

Subsystem	Total Labor Hours	Total Parts Cost	Total Cost of Replacements	% of Total Cost From Labor
Electrical	461.8	$ 188,141.85	$ 208,922.85	9.95%
Other or N/A	787	$ 61,803.52	$ 97,218.52	36.43%
Axles	330.9	$ 50,938.88	$ 65,829.38	22.62%
Engine	233.8	$ 54,531.58	$ 65,052.58	16.17%
Troop Compartment	357.7	$ 40,584.44	$ 56,680.94	28.40%
Transmission	103.6	$ 45,986.03	$ 50,648.03	9.20%
Wheel Hub/Tires	87.6	$ 46,627.98	$ 50,569.98	7.80%
Hydraulics	138.3	$ 24,232.28	$ 30,455.78	20.43%

Subsystem	Total Labor Hours	Total Parts Cost	Total Cost of Replacements	% of Total Cost From Labor
Fuel	204.1	$ 18,620.46	$ 27,804.96	33.03%
Compressed Air	205	$ 13,879.93	$ 23,104.93	39.93%
Comms	66.3	$ 11,823.12	$ 14,806.62	20.15%
Cooling	67.4	$ 7,169.43	$ 10,202.43	29.73%
Suspension	42.6	$ 6,526.92	$ 8,443.92	22.70%
Steering	69.8	$ 4,708.58	$ 7,849.58	40.01%
Crane/Hoist/Towing	45.9	$ 4,753.71	$ 6,819.21	30.29%
Drivetrain	27.6	$ 1,167.78	$ 2,409.78	51.54%
Exhaust	7.5	$ 212.05	$ 549.55	61.41%
A/C	1.2	$ 168.15	$ 222.15	24.31%

4.2.1.4.1 Electrical Subsystem

The electrical subsystem is notable for both a high total parts cost and a high labor cost.

4.2.1.4.1.1 Electrical Subsystem Analysis based on Part Cost:

A total of 13 distinct NSN numbers were identified with a component cost over $1,000. Table 20 contains the five highest cost components, plus one additional component, highlighted in gray, that was included due to the high number of replacements. All of the 13 components over $1,000 were replaced during unscheduled maintenance.

Table 20 – Electrical Subsystem parts replaced with the highest unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
5998014816794	ELECTRONIC COMPONEN	$ 5,269.81	1	1	0.3	$ 5,283.31
5998015267653	ELECTRONIC COMPONENTS	$ 5,176.28	1	1	0.5	$ 5,198.78
5998014842619	ELECTRONIC COMPONENTS	$ 4,899.41	1	1	2.0	$ 4,989.41
6110014983928	PANEL,POWER DISTRIB	$ 2,374.20	1	1	3.0	$ 2,509.20
2920015592715	GENERATOR,ENGINE ACCE	$ 1,952.00	8	8	2.9	$ 16,651.00
6115015040680	GENERATOR,ALTERNATI	$ 1,167.00	12	12	1.7	$ 14,944.50

4.2.1.4.1.2 Electrical Subsystem Analysis based on Total Replacements across the Fleet

The most replaced components in the electrical system are the batteries, followed by headlights and the generator (alternator), see Table 21. The majority of the replacements occurred during unscheduled

maintenance. Adding basic health/condition monitoring for these main components, i.e. batteries and alternator, may provide a better understanding of the failure root-causes, potential failure mitigation opportunities, and opportunities for prognostics. However, for the headlights, the current identification practices (manual) should suffice due to the lower operational severity associated with failure.

Table 21 – Electrical Subsystem parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
6140014851472	BATTERY,STORAGE	$ 379.03	0.4	59	214	57	206
6140014469506	BATTERY,STORAGE	$ 106.97	0.3	25	93	21	77
6220015027312	HEADLIGHT	$ 28.31	0.6	26	30	26	30
6115015040680	GENERATOR,ALTERNATI	$ 1,167.00	1.7	12	12	12	12
6220015021852	HEADLIGHT	$ 69.29	0.6	9	11	9	11

4.2.1.4.1.3 Electrical Subsystem Analysis based on Highest Average Labor Cost

The electrical system contained 15 different NSN numbers with average labor of two hours or more. The seven NSNs with three hours or more of labor are provided in Table 22. The two items highlighted in gray were performed while the vehicle was in for scheduled maintenance; however, these do not appear to be typical scheduled maintenance items.

Table 22 - Electrical Subsystem parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
2530015597462	CABLE AND CONDUIT A	$ 751.67	15.0	1	1
6150015675977	WIRING HARNESS,BRAN	$ 1,153.67	15.0	1	1
2540015222431	CONTROL BOX,WINDSHIEL	$ 314.38	5.5	1	1
2920014604019	STARTER,ENGINE,ELECTR	$ 426.00	5.2	3	3
5935014791602	CONNECTOR,PLUG,ELECTR	$ 2.24	5.0	1	1
6130015022579	BATTERY POWER SUPPL	$ 645.04	3.5	4	4
6110014983928	PANEL,POWER DISTRIB	$ 2,374.20	3.0	1	1

4.2.1.4.1.4 Electrical Subsystem Analysis based on Highest Total Cost (including labor)

As previously shown in Table 19, electrical system repairs have a significant cost ($208,922.85) relative to other systems. Although the majority of these costs are associated with batteries, many other components have high total replacement costs as shown in Table 23. There are an additional 13 components with total replacement costs over $1,000 that are not shown in Table 23.

Table 23 - Electrical Subsystem parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Unscheduled Work Orders	Total Work Orders	Unscheduled WO Cost	Total WO Cost
6140014851472	BATTERY,STORAGE	$ 379.03	57	59	$ 82,287.68	$ 85,427.92
2920015592715	GENERATOR,ENGINE ACCE	$ 1,952.00	8	8	$ 16,651.00	$ 16,651.00
6115015040680	GENERATOR,ALTERNATI	$ 1,167.00	12	12	$ 14,944.50	$ 14,944.50
6140014469506	BATTERY,STORAGE	$ 106.97	21	25	$ 9,393.19	$ 11,397.21
5945015018715	RELAY ASSEMBLY	$ 579.70	9	9	$ 5,896.80	$ 5,896.80
5998014816794	ELECTRONIC COMPONEN	$ 5,269.81	1	1	$ 5,283.31	$ 5,283.31
5998015267653	ELECTRONIC COMPONENTS	$ 5,176.28	1	1	$ 5,198.78	$ 5,198.78
5998014842619	ELECTRONIC COMPONENTS	$ 4,899.41	1	1	$ 4,989.41	$ 4,989.41
2920014873587	STARTER,ENGINE,ELECTR	$ 632.19	6	6	$ 4,477.14	$ 4,477.14
6130015022579	BATTERY POWER SUPPL	$ 645.04	4	5	$ 3,201.16	$ 3,985.70

4.2.1.4.1.5 Conclusions of the Electrical System Maintenance Analysis

Electrical system costs are dominated by battery replacements (NSNs 6140014851472 and 6140014469506). At a total cost of $100,372.42 over the period of study, the cost of batteries is more than the cost of repairs to any other subsystem in Table 19. RIT did not receive information regarding how battery charge is managed on this fleet of vehicles, but previous fleet studies by RIT have shown that charge management plans are typically ineffective on lightly used military vehicles, and that enhancing charge management can significantly improve lead-acid battery life. Additionally, the alternator, starter, and relay assembly (6115015040680, 2920014873587, and 5945015018715) are systems worthy of further health monitoring and predictive maintenance consideration due to the number of replacements over the study period.

4.2.1.4.2 Axles

4.2.1.4.2.1 Axles Subsystem Analysis based on Part Cost:

Only three axle components were identified as high cost items, as shown in Table 25. However, all three components were replaced during unscheduled maintenance. Although these items are high cost and potentially have high operational impact, further understanding of the failure root-cause, current maintenance procedures, and operational impacts is needed to understand whether condition monitoring might be warranted.

Table 24 – Axle subsystem parts replaced with the highest unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
2520015192773	AXLE ASSEMBLY,AUTOMOT	$ 15,119.00	1	1	6.8	$ 15,425.00
2520015048169	AXLE ASSEMBLY,AUTOMOT	$ 3,658.00	2	2	5.0	$ 7,766.00
3040013638153	SHAFT,STRAIGHT	$ 749.50	1	1	4.6	$ 956.50

4.2.1.4.2.2 Axles Subsystem Analysis based on Total Replacements across the Fleet

The majority of the high replacement components for the axles occur during scheduled maintenance, as shown in Table 25, with the unscheduled replacements being significantly lower. The maintenance data suggests that a scheduled replacement strategy is used with these components. Failure causes and operational impacts associated with unscheduled replacements would need to be better understood before considering condition monitoring.

Table 25 - Axle subsystem parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
5331013921637	O-RING	$ 2.36	0.1	66	1202	2	16
5330013605252	SEAL,PLAIN ENCASED	$ 41.85	0.3	88	428	5	10
5330013607753	SEAL,PLAIN ENCASED	$ 28.06	0.2	23	164	6	22
5330013624993	SEAL,PLAIN	$ 3.29	0.1	21	153	4	17

4.2.1.4.2.3 Axles Subsystem Analysis based on Highest Average Labor Cost

Table 26 provides the results of the axels subsystem data analysis based on average labor cost. When analyzing the highest labor cost items, the first three items were identical to the high unit cost items. The fourth item was the only other item with over three hours of labor. It should be noted that all of these repairs occurred during unscheduled maintenance.

Table 26 - Axle subsystem parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
2520015192773	AXLE ASSEMBLY,AUTOMOT	$ 15,119.00	6.8	1	1
2520015048169	AXLE ASSEMBLY,AUTOMOT	$ 3,658.00	5.0	2	2
3040013638153	SHAFT,STRAIGHT	$ 749.50	4.6	1	1
5330013623392	SEAL,NONMETALLIC ROUN	$ 5.32	3.4	2	3

4.2.1.4.2.4 Axles Subsystem Analysis based on Highest Total Cost (including labor)

The five highest total cost items in the axle subsystem are shown in Table 27. The next highest cost item was significantly less expensive at $1,488. The majority of the items replaced were done so during routine maintenance, with the two axle assemblies (NSNs 2520015192773 and 2520015048169) being the exceptions. These two axle assemblies were also the most expensive unscheduled maintenance repairs.

Table 27 - Axle subsystem parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Unscheduled Work Orders	Total Work Orders	Unscheduled WO Cost	Total WO Cost
5330013605252	SEAL,PLAIN ENCASED	$ 41.85	5	88	$ 1,320.19	$ 23,235.30
2520015192773	AXLE ASSEMBLY,AUTOMOT	$ 15,119.00	1	1	$ 15,425.00	$ 15,425.00
2520015048169	AXLE ASSEMBLY,AUTOMOT	$ 3,658.00	2	2	$ 7,766.00	$ 7,766.00
5331013921637	O-RING	$ 2.36	2	66	$ 194.10	$ 6,405.22
5330013607753	SEAL,PLAIN ENCASED	$ 28.06	6	23	$ 1,628.96	$ 6,244.34

4.2.1.4.2.5 Conclusions of the Axles Maintenance Analysis

The axles have significant scheduled maintenance costs, generally associated with seals and O-rings. However, the axle assemblies and the straight shaft have the highest component and labor cost and are replaced as unscheduled maintenance. Thus, these components may be candidates to consider for monitoring, especially if the associated failures could lead to the vehicle being inoperative in the field.

4.2.1.4.3 Engine

4.2.1.4.3.1 Engine Subsystem Analysis based on Part Cost:

An evaluation of the replacements for the engine based on part cost identified two components with a replacement value of over $1,000, consisting of two different diesel engine NSNs (see Table 28).

Table 28 – Engine subsystem parts replaced with the highest unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
2815015265263	ENGINE BLOCK,DIESEL	$ 32,467.00	1	1	74.0	$ 35,797.00
2815014672373	ENGINE,DIESEL	$ 17,961.00	1	1	0.5	$ 17,983.50

The second engine is listed as only having a half hour of labor time, which would be considerably low for an engine swap out. A review of the work order does appear that an engine was ordered as a replacement, not merely listed as the item being worked on. Although only two engines were replaced, engine tuning may have significant labor costs, which could be an additional driver for CBM. Additionally, in modern vehicles like the FMTV, ECUs can be a maintenance driver. These did not show up clearly in the maintenance cost and frequency analysis.

4.2.1.4.3.2 Engine Subsystem Analysis based on Total Replacements across the Fleet

With a snapshot of maintenance that covers approximately 1.5 years, an analysis of parts replacements to identify frequently replaced components may identify recurring problems. To keep the list of frequent items manageable, items that were replaced more than five times across the whole fleet were identified. Table 29 contains a list of engine system components that were replaced five or more times across the entire fleet of vehicles. The unscheduled work orders and replacements are a subset of the total number of work orders and replacements.

Table 29 - Engine subsystem parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
2910015193768	FILTER ELEMENT,FLUID	$ 13.24	1.1	62	62	21	21
5342013825021	MOUNT,RESILIENT,WEA	$ 37.73	2.0	17	24	12	17
6680015689447	GAGE ROD,LIQUID LEV	$ 76.39	0.5	10	10	6	6

4.2.1.4.3.3 Engine Subsystem Analysis based on Highest Average Labor Cost

The highest unit labor items for the engine are summarized in Table 30. The highlighted item was the only item at the top of the labor hour list that was typically done during scheduled maintenance events. It should also be noted that only the mount (NSN# 5342013825021) had more than two unscheduled replacements over the 1.5 year period.

Table 30 - Engine subsystem parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
2815015265263	ENGINE BLOCK,DIESEL	$ 32,467.00	74.0	1	1
2520015053162	ADAPTER,ASSEMBLY FL	$ 632.90	11.0	1	1
5330010695128	GASKET	$ 384.71	4.3	2	2
5305013601993	SCREW,CAP,HEXAGON H	$ 0.46	4.0	1	1
5330015138233	GASKET	$ 60.66	2.8	1	1
5342013825021	MOUNT,RESILIENT,WEA	$ 37.73	2.1	12	17

4.2.1.4.3.4 Engine Subsystem Analysis based on Highest Total Cost (including labor)

The final analysis on the engine sub-system was a breakdown of the highest total cost (parts cost + labor cost) across the fleet. Table 31 shows items with a total fleet cost of greater than $1,000.

Table 31 - Engine subsystem parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Unscheduled Work Orders	Total Work Orders	Unscheduled WO Cost	Total WO Cost
2815015265263	ENGINE BLOCK,DIESEL	$ 32,467.00	1	1	$ 35,797.00	$ 35,797.00
2815014672373	ENGINE,DIESEL	$ 17,961.00	1	1	$ 17,983.50	$ 17,983.50
2910015193768	FILTER ELEMENT,FLUID	$ 13.24	21	62	$ 1,011.54	$ 3,781.88
5342013825021	MOUNT,RESILIENT,WEA	$ 37.73	12	17	$ 2,261.41	$ 3,043.02
5330010695128	GASKET	$ 384.71	0	2	$ 0	$ 1,156.42
2520015053162	ADAPTER,ASSEMBLY FL	$ 632.90	1	1	$ 1,127.90	$ 1,127.90

4.2.1.4.3.5 Conclusions of the Engine Maintenance Analysis

The unscheduled part replacements associated with the engine subsystem have a significant impact on overall engine maintenance costs. Filter replacements (NSN# 2910015193768), though less costly, occur relatively frequently and may be able to be monitored for performance degradation – this might allow for more/less frequent replacement as-needed, allowing for reduced overall maintenance cost and overall better engine performance. Similarly, the mounts (NSN# 5342013825021) appear to be replaced relatively frequently, but the operational impacts of the mount failure is not known from this analysis. If excessive vibration is an observed impact, the application of vibration monitoring may be capable of detecting the fault.

4.2.1.4.4 Troop Compartment

4.2.1.4.4.1 Troop Compartment Subsystem Analysis based on Part Cost:

An evaluation of troop compartment replacements based on part cost (shown in Table 32) identified five components with a replacement value of over $1,000, all of which were replaced during scheduled maintenance.

Table 32 – Troop compartment subsystem parts replaced with the highest unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
2510015680184	DOOR,VEHICULAR	$ 3,455.55	1	1	12.5	$ 4,018.05
5340015680723	HINGE,ACCESS DOOR	$ 2,115.49	1	1	1.0	$ 2,160.49
2510015678841	HINGE,HATCH,VEHICULAR	$ 1,956.26	1	1	1.0	$ 2,001.26
2510013743120	DOOR,VEHICULAR	$ 1,755.97	2	2	1.0	$ 3,601.94
2510015678724	SUPPORT,CAB ASSEMBLY	$ 1,569.70	1	1	1.0	$ 1,614.70

4.2.1.4.4.2 Troop Compartment Subsystem Analysis based on Total Replacements across the Fleet

Analysis of total troop compartment replacements by component identified nine components that were replaced more than 10 times. However, six of the components were low cost (parts and labor) replacements. The remaining four items are shown in Table 33.

Table 33 - Troop compartment subsystem parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
2540015295660	COVER,SEAT,VEHICULAR	$ 82.61	0.6	14	18	0	0
5342013717258	MOUNT,RESILIENT,WEA	$ 11.22	1.1	8	17	8	17
2540013741764	LATCH,DOOR,VEHICULAR	$ 208.87	1.8	14	14	12	12
2510013657152	WINDOW,VEHICULAR	$ 151.11	2.0	10	11	9	10

4.2.1.4.4.3 Troop Compartment Subsystem Analysis based on Highest Average Labor Cost

Table 34 contains the troop compartment components with the highest average labor costs. The item highlighted in gray (NSN #30400015727391) was the only item replaced during scheduled maintenance.

Table 34 - Troop compartment subsystem parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
2510015680184	DOOR,VEHICULAR	$ 3,455.55	12.5	1	1
3040015727391	CYLINDER ASSEMBLY,A	$ 352.46	11.0	1	1
3040013776805	ACTUATOR,MECHANICAL,N	$ 139.64	9.7	1	1
3040013757323	ACTUATOR,MECHANICAL,N	$ 152.13	5.1	2	2
2590015409228	BRACKET,VEHICULAR C	$ 190.60	3.0	1	1
5340015398880	HINGE,BUTT	$ 92.73	3.0	1	1

4.2.1.4.4.4 Troop Compartment Subsystem Analysis based on Highest Total Cost (including labor)

The troop compartment components with the highest total labor costs are shown in Table 35. There were an additional five components with a total cost of over $1,000 that are not shown.

Table 35 - Troop compartment subsystem parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Unscheduled Work Orders	Total Work Orders	Unscheduled WO Cost	Total WO Cost
2540013741764	LATCH,DOOR,VEHICULAR	$ 208.87	12	14	$ 3,518.94	$ 4,026.68
2510015680184	DOOR,VEHICULAR	$ 3,455.55	1	1	$ 4,018.05	$ 4,018.05
2510013743120	DOOR,VEHICULAR	$ 1,755.97	2	2	$ 3,601.94	$ 3,601.94
2510013657152	WINDOW,VEHICULAR	$ 151.11	9	10	$ 2,456.10	$ 2,652.21
2540013763999	HANDLE,DOOR,VEHICUL	$ 250.30	8	8	$ 2,348.90	$ 2,348.90
5340015680723	HINGE,ACCESS DOOR	$ 2,115.49	1	1	$ 2,160.49	$ 2,160.49
2510015678841	HINGE,HATCH,VEHICULAR	$ 1,956.26	1	1	$ 2,001.26	$ 2,001.26

4.2.1.4.4.5 Conclusions of the Troop Compartment Maintenance Analysis

The troop compartment was the subsystem with the fifth highest total maintenance costs. The components with the seven highest total costs account for $20,809. The majority of which (97%) are unscheduled maintenance costs. However, only the actuators have potential for monitoring, since the other components are mostly body components.

4.2.1.4.5 Transmission

4.2.1.4.5.1 Transmission Subsystem Analysis based on Part Cost:

For the transmission, there were only four replacements with a part cost over $1,000, all of which were unscheduled maintenance (see Table 36). The first two components, NSNs 2520015484841 and 252001493605 have both high part and high labor costs.

Table 36 – Transmission subsystem parts replacements sorted by unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
2520015484841	TRANSMISSION,MECHANIC	$ 23,303.00	1	1	30.0	$ 24,653.00
2520014936059	TRANSMISSION,HYDRAULI	$ 13,416.00	1	1	31.0	$ 14,811.00
3010014609681	CONTROL ASSEMBLY,TA	$ 2,568.81	1	1	1.5	$ 2,636.31
2520013922330	COOLER,FLUID,TRANSM	$ 1,692.41	1	1	4.0	$ 1,872.41

4.2.1.4.5.2 Transmission Subsystem Analysis based on Total Replacements across the Fleet

After analyzing the transmission for both scheduled and unscheduled maintenance, the items with the most replacements were identified, as shown in Table 37. The listed valve component was only replaced under annual service work orders. The three other components could be considered for health monitoring, as they have either significant parts or labor costs.

Table 37 - Transmission subsystem parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
2910015127970	FILTER ELEMENT,FLUI	$ 34.92	8.3	10	12	4	5
820011513692	VALVE,VENT	$ 1.33	2.3	4	12	0	0
2520015549359	CONTROL ASSEMBLY,TR	$ 582.83	3.8	4	4	4	4
2520015392851	CONTROL ASSEMBLY,TR	$ 330.47	2.7	3	3	3	3

4.2.1.4.5.3 Transmission Subsystem Analysis based on Highest Average Labor Cost

The three highest labor hour components are provided in Table 38, all of which cost over $500 in labor. All of these components were unscheduled maintenance.

Table 38 - Transmission subsystem parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
2520014936059	TRANSMISSION,HYDRAULI	$ 13,416.00	31.0	1	1
2520015484841	TRANSMISSION,MECHANIC	$ 23,303.00	30.0	1	1
4140015069651	FAN ASSEMBLY,CENTRIFU	$ 242.17	12.0	1	1

4.2.1.4.5.4 Transmission Subsystem Analysis based on Highest Total Cost (including labor)

The transmission subsystem consisted of six components whose total replacement costs were over $1,000 during the 1.5 years under study, as shown in Table 39. It should be noted that all of these components were replaced as unscheduled maintenance. The gray shaded components would be good candidates for monitoring, as they are high cost items. The remaining items should be considered for monitoring based on their operational impacts. For example, it is possible that a fault control assembly or a leaky transmission cooler would render the transmission inoperable, therefore potentially warranting monitoring.

Table 39 - Transmission subsystem parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total labor Hours	Total Cost
2520015484841	TRANSMISSION,MECHANIC	$ 23,303.00	1	30	$ 24,653.00
2520014936059	TRANSMISSION,HYDRAULI	$ 13,416.00	1	31	$ 14,811.00
3010014609681	CONTROL ASSEMBLY,TA	$ 2,568.81	1	1.5	$ 2,636.31
2520015549359	CONTROL ASSEMBLY,TR	$ 582.83	4	3.8	$ 2,502.32
2520013922330	COOLER,FLUID,TRANSM	$ 1,692.41	1	4	$ 1,872.41
2520015392851	CONTROL ASSEMBLY,TR	$ 330.47	3	2.7	$ 1,112.91

4.2.1.4.5.5 Conclusions of the Transmission Maintenance Analysis

The two transmission assemblies (NSNs 2520014936059 and 2520015484841) dominate the analysis due to their high component and labor costs. Additionally, the failure of these components would likely lead to operational failure and a deadlining condition. Similarly, control assemblies appear multiple times on the list, but the impact of their failure is unknown, indicating that further analysis of failure root-cause and operational implications is warranted.

4.2.1.4.6 Wheel Hub/Tires

4.2.1.4.6.1 Wheel Hub/Tires Analysis based on Part Cost:

For the wheel hub/tires subsystem, there were only two replacements with a part cost over $1,000 (see Table 40). The second component listed in the table, in gray, had 26 total replacements, of which four were performed on a single maintenance order with scheduled maintenance.

Table 40 – Wheel hub/tire parts replaced with the highest unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
2530015715857	WHEEL, PNEUMATIC TIR	$ 1,816.80	2	2	0.4	$ 3,669.60
2530015004619	WHEEL,PNEUMATIC TIRE	$ 1,300.00	13	26	1.4	$ 35,424.50

4.2.1.4.6.2 Wheel Hub/Tires Analysis based on Total Replacements across the Fleet

Within the wheel/tire subsystem (see Table 41), the highest replacement items consisted of relatively low cost items, with the exception of tires. Additionally, most of the low cost items were replaced under scheduled maintenance work orders.

Table 41 - Wheel hub/tire parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
5310014545553	WASHER,SEAL	$ 6.57	0.1	11	84	N/A	N/a
4460012842344	FILTER ELEMENT,FLUID	$ 3.24	0.2	7	40	N/A	N/A
2530015004619	WHEEL,PNEUMATIC TIRE	$ 1,300.00	1.4	13	26	12	22
5310001410447	NUT,PLAIN,SINGLE BA	$ 1.17	0.0	3	24	3	24
2610013569098	TIRE,PNEUMATIC,VEHICU	$ 656.10	2.0	11	12	11	12
5310004807606	NUT PLAIN SINGLE BALL	$ 1.10	0.1	1	10	N/A	N/A

4.2.1.4.6.3 Wheel Hub/Tires Analysis based on Highest Average Labor Cost

The only item with more than two hours of labor in the wheel/tire subsystem was an O-ring, which was replaced only once (see Table 42).

Table 42 - Wheel hub/tire parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
5331014835079	O-RING	$ 1.02	4	1	1

4.2.1.4.6.4 Wheel Hub/Tires Analysis based on Highest Total Cost (including labor)

The items with the highest total maintenance cost in the wheel/tire subsystem were all tires (see Table 43). No other items had a total cost over $1,000.

Table 43 - Wheel hub/tire parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Unscheduled Work Orders	Total Work Orders	Unscheduled WO Cost	Total WO Cost
2530015004619	WHEEL,PNEUMATIC TIRE	$ 1,300.00	12	13	$ 30,053.50	$ 35,424.50
2610013569098	TIRE,PNEUMATIC,VEHICU	$ 656.10	11	11	$ 8,930.70	$ 8,930.70
2530015715857	WHEEL, PNEUMATIC TIR	$ 1,816.80	2	2	$ 3,669.60	$ 3,669.60

4.2.1.4.6.5 Conclusions of the Wheel Hub/Tire Maintenance Analysis

Tires are a significant cost driver for this subsystem, accounting for 89.6% of the subsystem maintenance costs. Additionally, most tires were replaced on unscheduled work orders. If the vehicle has a central tire inflation system (CTIS), capturing the air pressure from the ECU may provide an indication of slow leaks. However, if the tires are suffering failure due to objects penetrating the tire, monitoring will not be able to provide any assistance.

4.2.1.4.7 Other components

4.2.1.4.7.1 Other Components Analysis Based On Part Cost:

An additional nine components with a part cost of over $1,000 were found across the remaining subsystems (see Table 44). Radiators and "Envelope, power unit" (hydraulic system) were the only relatively high cost items in the "other" category replaced more than two times across the fleet, and all replacement of these items were performed as unscheduled maintenance.

Table 44 – Other subsystem parts replaced with the highest unit cost

Part NSN	Part Nomenclature	Component Cost	Total Work Orders	Total Replacements	Average Man Hours per w/o	Total WO Cost
4820015229935	VALVE,REGULATING,SYST	$ 1,121.46	2	2	0.6	$ 2,296.92
4820013722769	VALVE,REGULATING,SY	$ 1,010.26	2	2	1.4	$ 2,146.52
2530015554681	STEERING GEAR	$ 2,429.30	1	1	2.0	$ 2,429.30
5360013757092	SPRING,LEAF	$ 1,056.22	1	2	2.0	$ 2,292.44
2930015193573	RADIATOR,ENGINE COO	$ 1,490.89	4	4	5.1	$ 6,872.56
4820015210811	VALVE,LINEAR,DIRECTIO	$ 1,602.52	1	1	2.5	$ 1,715.02
4730015278543	MANIFOLD ASSEMBLY,H	$ 1,738.21	1	1	2.0	$ 1,828.21
4730015678670	MANIFOLD ASSEMBLY,H	$ 1,704.14	1	1	10.0	$ 2,154.14
6115015669209	ENVELOPE, POWER UNIT	$ 1,170.82	8	8	4.1	$ 10,851.56

4.2.1.4.7.2 Other Components Analysis based on Total Replacements across the Fleet

There were a significant number of components with at least 10 replacements across the remaining subsystems (see Table 45). However, many of these items were parts kits, wiper blades, or filters requiring typical maintenance. For those items, once scheduled maintenance is removed, there are significantly fewer replacements. The one item that stands out as significant due to quantity and cost is the hydraulic pumping unit (NSN #4320014174876).

Table 45 - Other subsystem parts sorted by Number of Replacements in the Fleet

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Total Number of Work Orders	Total Number Replaced	Unscheduled Work Orders	Unscheduled Replacements
4020014449193	CORD ASSEMBLY,ELASTIC	$ 8.75	0.3	16	47	5	11
5340014990935	BRACKET,MULTIPLE AN	$ 17.64	0.3	19	45	12	28
5965015337661	MICROPHONE,DYNAMIC	$ 65.32	0.4	11	12	N/A	N/A
5330001721919	PACKING,PREFORMED	$ 0.50	0.3	27	66	16	38
2910013606366	FILTER ELEMENT,FLUID	$ 15.44	0.5	155	155	14	14
2910014247315	FILTER ELEMENT,FLUID	$ 17.54	0.5	100	100	7	7
2940015265483	FILTER ELEMENT,INTAKE	$ 135.97	0.5	31	31	3	3
5330004770338	PACKING,PREFORMED	$ 6.68	0.4	30	30	7	7
4320014174876	PUMPING UNIT,HYDRAULI	$ 491.69	2.3	11	12	11	12
2540014822300	BLADE,WINDSHIELD WIPE	$ 3.90	0.3	64	136	55	113
4440015423419	PARTS KIT,AIR DRIER	$ 86.47	0.9	84	84	N/A	N/A
5975010482922	STRAP,TIEDOWN,ELECT	$ 0.13	0.1	29	58	29	58
2540014540415	REFILL BLADE,WIPER	$ 2.01	0.3	15	24	15	24
2590015287507	SERVICE KIT,VEHICLE	$ 573.37	8.9	24	24	2	2
2590015336745	PARTS KIT,SPECIALIZED	$ 418.15	4.5	18	18	12	12
4730015247892	AIR DRIER AND COOLER,	$ 354.65	1.1	16	16	3	3
2590015336748	PARTS KIT,SPECIALIZ	$ 581.22	5.1	15	15	6	6
2910013757624	FILTER ELEMENT,FLUID	$ 4.79	0.5	71	71	N/A	N/A

4.2.1.4.7.3 Other Components Analysis based on Highest Average Labor Cost

Table 46 includes all repairs from the remaining subsystems with an average repair time of more than five hours. Although there are some significant repair times noted in the table, many of the items were repaired only once. The more significant, multiple repair items are highlighted. These highlighted items also excluded repair kits, which are typical of scheduled maintenance. Of the highlighted items, all were replaced under unscheduled maintenance except for one of the three reciprocating compressors. It is possible that the one repair was an unscheduled replacement that was performed while the vehicle was in for scheduled maintenance.

Table 46 - Other subsystem parts replacements sorted by average labor cost

Part NSN	Part Nomenclature	Component Cost	Average Labor Hours	Number of Work Orders	Number of Replacements
2530013613966	CHAMBER,AIR BRAKE	$ 89.49	10.0	1	1
4310013615075	COMPRESSOR,RECIPROC	$ 629.00	7.8	3	3
2930015071419	PUMP,COOLING SYSTEM,E	$ 522.00	12.0	1	1
5330015248348	GASKET	$ 16.48	12.0	1	1
2930015193573	RADIATOR,ENGINE COO	$ 1,490.89	5.1	4	4
2910014706177	INJECTOR ASSEMBLY,FUE	$ 724.49	7.0	1	1
4730015678670	MANIFOLD ASSEMBLY,H	$ 1,704.14	10.0	1	1
3040013705486	CYLINDER ASSEMBLY,ACT	$ 350.26	6.0	3	3
2590015287507	SERVICE KIT,VEHICLE	$ 573.37	9.0	2	2
2590015336748	PARTS KIT,SPECIALIZ	$ 581.22	8.3	6	6
4720010144915	HOSE,NONMETALLIC	$ 0.34	6.0	1	1
2590015336745	PARTS KIT,SPECIALIZED	$ 418.15	5.5	12	12
2590015287239	SERVICE KIT,VEHICLE	$ 455.09	5.4	2	2
4720014219712	TUBING,NONMETALLIC	$ 0.37	5.0	1	1
2540013776607	BUMPER,VEHICULAR	$ 395.14	5.8	1	1

4.2.1.4.7.4 Other Components Analysis based on Highest Total Cost (including labor)

The highest total repair cost items for the remaining subsystems can be found in Table 47. Only items whose total repair cost was greater than $6,000 were included. Once again, some of the parts kits made the list but were mostly repaired under scheduled maintenance. Therefore, the most significant cost items that may also have monitoring potential have been highlighted.

Table 47 - Other subsystem parts with the highest total maintenance cost

Part NSN	Part Nomenclature	Component Cost	Unscheduled Work Orders	Total Work Orders	Unscheduled WO Cost	Total WO Cost
5830015336139	INTERCOMMUNICATION	$ 618.44	5	10	$ 3,362.20	$ 6,557.90
2930015193573	RADIATOR,ENGINE COO	$ 1,490.89	4	4	$ 17,983.50	$ 6,872.56
2910015458117	INJECTOR ASSEMBLY,F	$ 759.62	4	4	$ 1,011.54	$ 7,466.58
6115015669209	ENVELOPE, POWER UNIT	$ 1,170.82	8	8	$ 10,851.56	$ 10,851.56
4320014174876	PUMPING UNIT,HYDRAULI	$ 491.69	11	11	$ 7,151.28	$ 7,151.28
2590015287507	SERVICE KIT,VEHICLE	$ 573.37	2	24	$ 1,956.74	$ 23,386.38
2590015336748	PARTS KIT,SPECIALIZ	$ 581.22	6	15	$ 5,737.32	$ 12,183.30
2590015336745	PARTS KIT,SPECIALIZED	$ 418.15	12	18	$ 7,987.80	$ 11,194.20
4440015423419	PARTS KIT,AIR DRIER	$ 86.47	N/A	84	N/A	$ 10,611.48
2590015287508	SERVICE KIT, VEHICLE	$ 616.78	N/A	10	N/A	$ 9,209.80
2590015287239	SERVICE KIT,VEHICLE	$ 455.09	2	8	$ 1,396.18	$ 6,871.72
4730015247892	AIR DRIER AND COOLER,	$ 354.65	3	16	$ 1,266.45	$ 6,430.40

4.2.1.4.7.5 Conclusions of the Maintenance Analysis of the Other Components

From the "other component" analysis, several components stand out as candidates for health monitoring due to their potential for degradation and sensing. Those items are the radiator (NSN #2930015193573), the injector assemblies (NSN #2910015458117), the "Envelope, power unit" (NSN #6115015669209), the reciprocating compressor (NSN #4310013615075), the hydraulic pumping unit (NSN #4320014174876), and the hydraulic manifolds (NSN #4730015278543 and #4730015678670). Considering the hydraulic system, for example, monitoring of system pressures may also provide condition insight into components, such as filters. Filter type components that are replaced based on "failure" will likely be associated with degraded performance and potentially secondary system damage. Filter type components replacements based on "time" are likely suboptimal from a cost standpoint. These secondary systems should be considered when evaluating the potential return on investment (ROI) of adding health monitoring to other system components.

4.2.1.5 Identification of Opportunities for Further Analysis on the FMTV

Utilizing knowledge of the available HUMS data from the FMTV, the failures identified in the Army Maintenance data were reviewed to identify immediate opportunities for PHM analysis. The HUMS data for the engine, cooling, and fuel system can be considered together as the signals can be related through modeling. Additionally, these systems, as a group, have the most available HUMS data. Reviewing the engine, cooling, and fuel system maintenance data identified two engine replacements, four radiator replacements, and a number of injector replacements as potential opportunities for evaluation.

Considering the available coolant related data, particularly coolant temperature and level, there is limited value in further investigating identification of a radiator failure. The level data may be utilized to identify

a decreasing coolant level, indicating a coolant leak somewhere in the system, but the data is not likely to lead to an indication of remaining life of a particular component, such as the radiator.

On the other hand, the engine failures may be identifiable through an analysis of the larger engine data set. Two work orders identified engine replacements, 2NT520200340 and 4E6BA0419860. However, evaluation of the HUMS data for the associated vehicles identified that there was no relevant HUMS data collected near the maintenance events, as seen in Figure 38, where the gray dashed lines represent the occurrence of the maintenance event and the blue lines represent days with HUMS data.

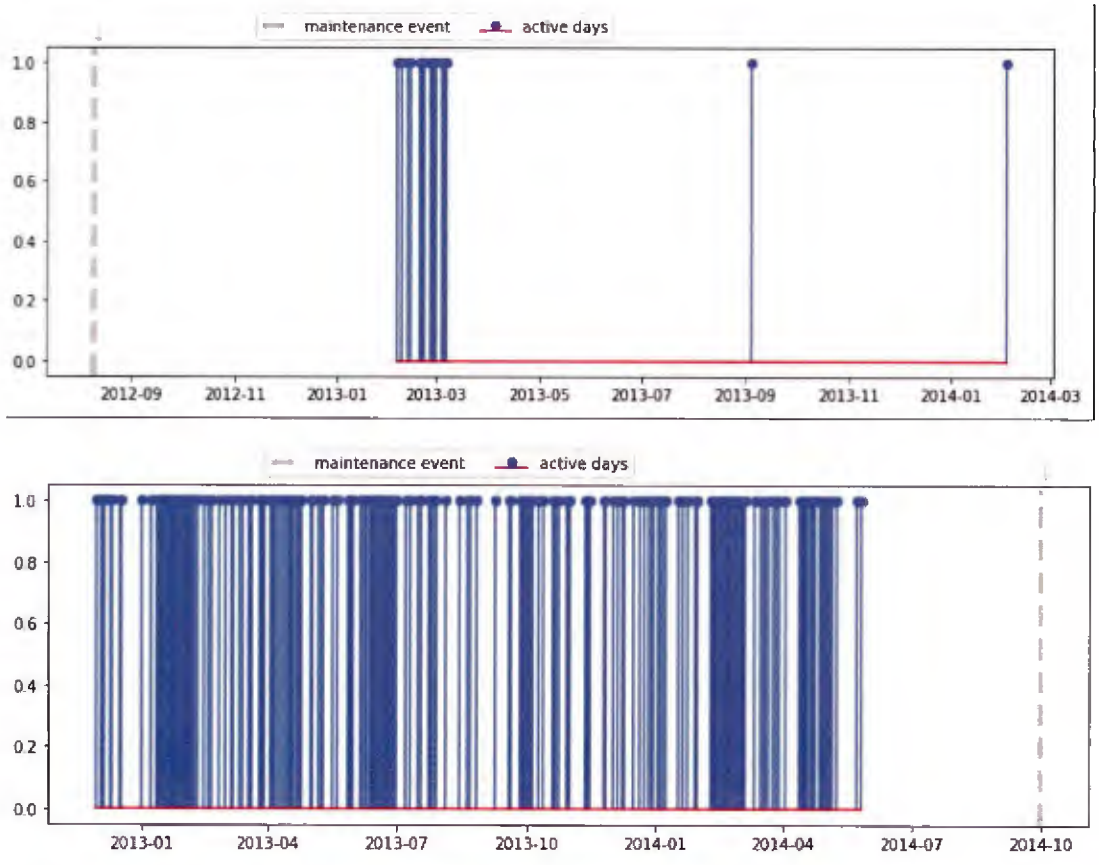


Figure 38 - HUMS data availability near engine replacements on two separate FMTV assets, (a) and (b)

There were five work orders associated with fuel injector replacements that described the engine as running rough or knocking. The information on the work orders and the availability of HUMS data can be found in Table 48 and Figure 39.

Table 48 - Injector Replacements

Vehicle	Work Order	NSN	Number Replaced	Availability of data
Vehicle 1	4E6BA0311476	2910015458117	1	HUMS data exists prior to and after the maintenance event, including DTCs related to injectors.
Vehicle 2	H1MAA1400207	2910015458117	6	The HUMS data set is small, especially after the maintenance event. DTCs did occur for the injectors prior to the event.
Vehicle 3	2NT301300208	2910015458117	1	The HUMS data set is limited after the maintenance event.
Vehicle 4	6HNK20200368	2910015458117	6	This event occurs prior to collection of HUMS data
Vehicle 5	DJAAA0400515	2910015458117	1	The HUMS data set ends just after the maintenance event.

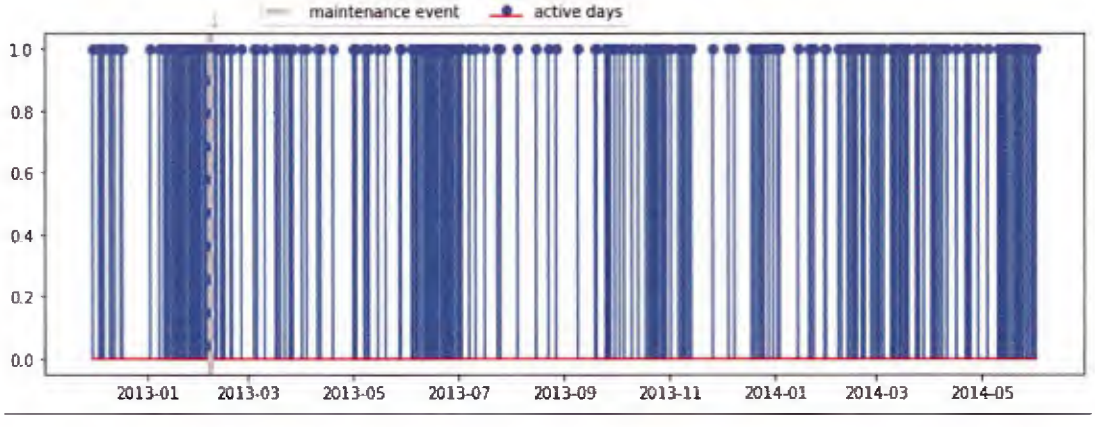


Figure 39 - HUMS Data availability near injector replacement on Vehicle 1.

Based on the alignment of the HUMS and maintenance data, the only clear match-up occurs for the injector failure on vehicle 1, which was further investigated for PHM potential (details are provided in Section 5).

4.2.2 Navy and Marine Corps Maintenance

Explicit maintenance data was not provided for the Navy ships or Marine Corps MTVR. The Navy did provide data for a few significant failure events to guide the investigations and analysis of the HUMS data. Three event types (failures) were indicated in the data, identified as 19 occurrences of EVENT_TYPE=1, 28 occurrences of EVENT_TYPE=2, and eight occurrences of EVENT_TYPE=3. The data consisted of

information indicating the ship, engine, time of the event, type of event, and the HUMS data leading up the event, which was a subset of the previously provided HUMS data. The data was presented as a file for each event consisting of the HUMS data preceding the event to enable data mining. These failures were evaluated for their PHM potential, as described in the next section.

Although maintenance data was not provided for the USMC MTVRs, some of the insights gained from the maintenance analysis of the FMTV may be applicable to the MTVR. One of the major cost drivers, batteries, is likely to be a similar cost driver on the MTVR. Similarly, other components such as injectors, radiators, engines, and ECUs may show similar failures and warrant similar monitoring approaches.

4.2.3 Other HUMS Opportunities

The goal of this analysis was to evaluate the maintenance data and identify areas with corresponding HUMS data to enable further PHM analysis. Although there are limited opportunities outlined above for the data received for this study, this does not mean that a PHM implementation must be narrow. The existing signal set may be utilized to calculate a number of usage parameters, including but not limited to: engine hours, engine idle hours, driving time, miles, time at or above certain temperatures (such as engine oil, transmission or coolant temp), or total braking duration. Additionally, contextual information should be gathered to ensure that operating and environmental conditions were properly captured. Condition indicators, and outputs of machine learning models in general (more on this in Chapter 5), respond to the changes in operating conditions [8].

Additionally, simple monitoring approaches and additional sensors may be opportunistically employed to enhance capabilities. Table 49 provides a list of some potential PHM capabilities by subsystem, including the signals that may enable them. The list is directed toward engine driven vehicles, but includes electrical, cooling and air intake subsystems.

Table 49 - Additional monitoring potential

System/subsystem	Signal	New or existing sensor	Analysis Context	Potential capability
Engine/Cooling	Coolant level	Existing	Monitoring change rate	Alert on low coolant, Alert on potential leaks
Engine/Cooling	Coolant temp	Existing	Monitor temp against engine load	Identify changes in cooling performance over time, Alert on high temperatures
Hydraulic	Hydraulic pressure	New	Monitor against engine speed	Identify changes in pump performance over time, Identify drops in pressure indicative of leaks
Electrical/Battery	Battery monitoring system – Voltage, current, state-of-charge	New	Monitor battery health, especially around vehicle down time	Identify batteries that need to be replaced prior to the batteries completely dying
Engine/Intake	Turbo/boost pressure	Existing	Monitor pressure against RPM	Identify performance changes over time, alert on low boost pressure
Electrical/Alternator	Alt. Out. Voltage Alt. Out. Current Battery Voltage Engine speed Ground Voltage Field Voltage Vibration Temperature	Existing New	High ground potential Output vs. Eng. speed Low/High/erratic voltage Motor Current Signature Analysis Vibration analysis	Identify electrical shorts, Identify regulator problems, Identify bearing faults, Identify low output
Engine	Oil Temperature Coolant Temp. Oil pressure Fuel rate Inst. Fuel economy Engine speed Pedal position Engine load Engine torque Injection Control Pressure Intake Manifold Temperature	Exists on many vehicles Exists on some vehicles	Engine speed vs oil Pressure Engine speed, pedal position, engine load, engine torque, injection control pressure, intake manifold temp. and fuel rate Oil temp, coolant temp, engine speed, and engine load	Lubrication system status System level modeling, performance degradation Cooling system modeling/performance

5 Immediate PHM Opportunities

This section explores opportunities for data-driven PHM development using the data sets described in the preceding sections. Because the development was data-driven, it required ground truth from maintenance data records. The approach was opportunistic: utilizing the significant, repeated repairs identified in Section 4.2 and the understanding of the data collected from the analysis performed in Section 4.1, the diesel engine was identified as the common subsystem across platforms for a deeper dive

into PHM application. The ground truth of the maintenance events combined with the specific relevant signals from the HUMS data enables statistical comparison of data related to normal operation, found well before the occurrence of the maintenance event or after the repair, to that of the onset of failure, found leading up to the repair.

When evaluating the immediate potential for PHM on any of the platforms, the first level of capability is anomaly detection. The low-hanging-fruit opportunities were identified as HUMS data that preceded recurring maintenance events. Anomaly detection could be immediately tested by contrasting the normal operation to the data before the maintenance. This step of PHM development is discussed here in detail, with multiple examples. The path to the higher capability levels, viz. diagnostics and prognostics, are discussed in the conclusion.

The core focus of the analysis was on anomaly detection approaches. The state of the art anomaly detectors, based on deep learning models and specifically on autoencoders, were considered first. These models are described in detail because of their wide-ranging applicability in PHM. Autoencoders based upon 1D convolutional neural networks are introduced here for the first time for PHM applications to the best of our knowledge.

A key feature of autoencoders is that they do not require data associated with faults for anomaly detection. This is invaluable for PHM development, where normal data comes in abundance and the fault- and failure-related data is scarce. However, in addition to anomaly detection, autoencoders can serve as data-driven generators of condition indicators. This application was first examined using Navy data and then further explored using a simulation.

This section concludes with an exploratory analysis of Navy ship data, which employs domain knowledge. Understanding of the physics of the system informs the model about relevant relationships among signals. Thus, the model had less to learn from the data and the classical machine learning was sufficient.

5.1 Autoencoders

Traditionally, the time and effort associated with extracting effective features is the dominant activity in data-driven PHM. The key distinguishing factor in the success of machine learning approaches is feature engineering [15], which represents ≈90% of industrial machine learning [16]. The promise of modern deep learning approaches is that they can develop better representations directly from data, or with minimum feature engineering. Neural networks applications have a rich history in PHM, with an early successful deployment using autoencoder topology [17] preceding the recent deep learning revolution.

This study examined the potential of a straightforward applications of autoencoders for detecting anomalies in the Army HUMS data, collected on FMTV platforms (Section 4). The model was first developed for an engine problem (fuel injector) and later successfully applied to transmission ECU failure. The engine was the subsystem of choice for the PHM development, because it is a common subsystem for the Navy (ships) and Army assets (trucks). Although the engine model did not perform well in the sense that it was not able to find the anomaly, the details of its development are presented here for two reasons. First, the engine subsystem is of critical importance for many assets, and it was used in this work to illustrate model development. Second, the original engine model was modified, and successfully applied to a transmission subsystem, where it was able to detect an anomaly.

5.1.1 Preprocessing and Modeling

The approach consisted of two main steps: data preprocessing and model training. The diagram in Figure 40 shows the activities associated with the two main steps and their interactions.

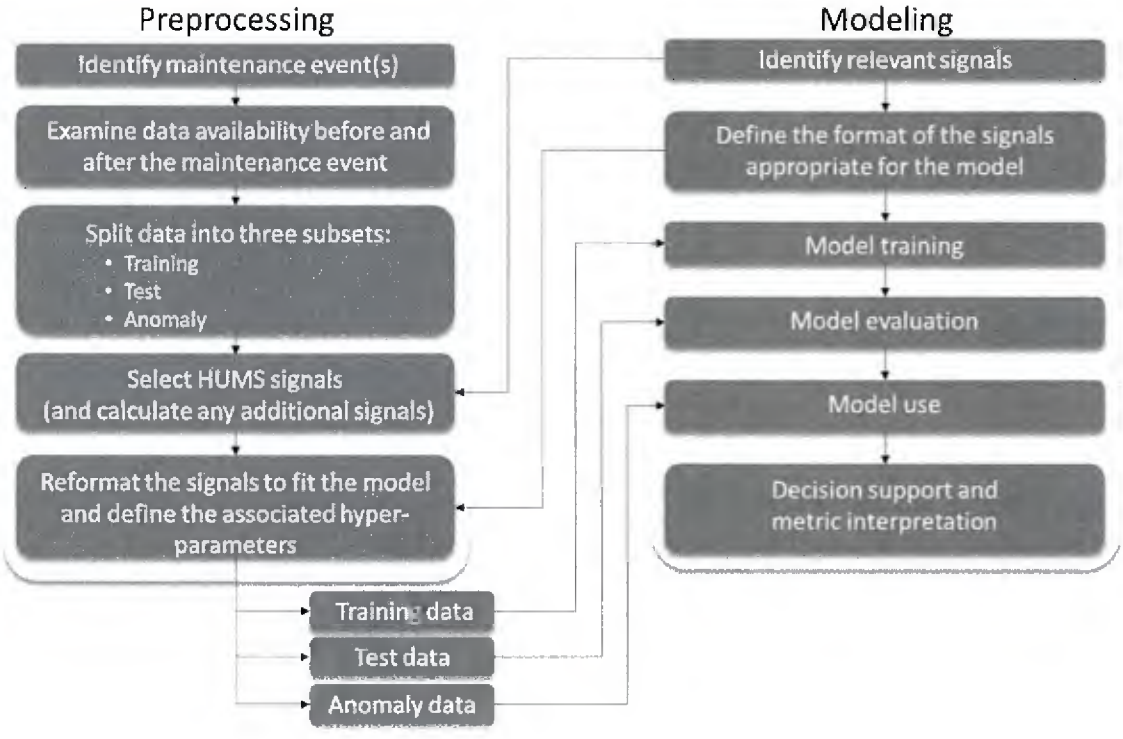


Figure 40: Preprocessing and modeling activities and their interactions.

The interactions between the modeling and preprocessing steps, depicted in Figure 40, suggest that a realistic approach to performing these analyses was to develop an end-to-end workflow and identify hyperparameters – the parameters changed by analysts during the iterative process of model development. The following sections describe these activities in more detail in the context of developing a specific model.

5.1.2 Engine Data Preprocessing

The first step in preprocessing was to select an appropriate maintenance event and relate its location in the context of the span of the HUMS data collection. Table 50 lists the entries of the maintenance records associated with the repair. The fault description was *ENGINE IS KNOCKING*, and the correction narrative was *REPLACED FUEL INJECTOR*.

Table 50: The maintenance events associated with an engine failure of interest

Family	FMTV
Fault Date	2/7/2013
Date Out	2/20/2013
Fault Description	ENGINE IS KNOCKING
Correction Narrative	REPLACED FUEL INJECTOR
Maintenance Level	FIELD
Action Description	Replaced
Part NSN	2910015458117
Part Nomenclature	INJECTOR ASSEMBLY,F
Part Quantity	1
Total Man Hours	1.3
Total Cost	759.62

Figure 41 shows the extent of the HUMS data collection, with data collection indicated by the blue line and the maintenance event indicated by the vertical dashed gray line. The dots on the blue line signify days with vehicle activity that was captured in HUMS. The gray vertical dashed line indicates the maintenance event in February of 2013.

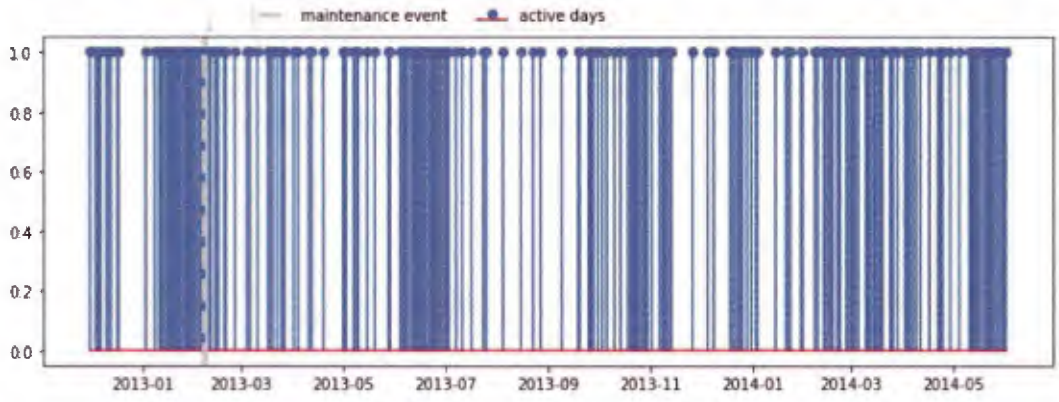
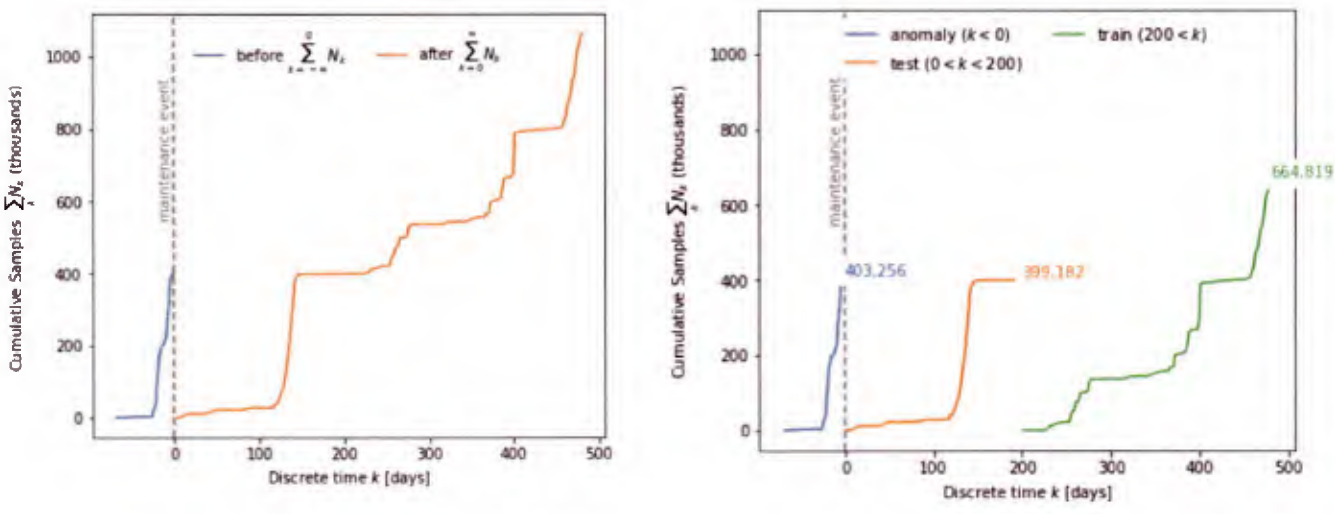


Figure 41: Vehicle daily activities and the maintenance event related to engine failure.

Figure 42a shows cumulative plots of number of rows of "1 Hz data" (the normal HUMS data). The cumulative plots were selected because they made it easier to compare how much data was available across the timeline of HUMS data for the specific vehicle. Note that the data horizon before the maintenance event was short. Because there was a relatively small amount of data before the maintenance event, the autoencoder model training was performed based upon the data collected *after* the engine repair. The model was trained on normal data (no fault present), because autoencoder-based anomaly detectors are trained to compress the input data, extracting the critical features of the system in the process of compression, and then reproduce the inputs in their decoding layers. Thus, models trained on normal data produce outputs that are consistent with normal data. Thus, if the system changes its behavior (due to a fault), the error signal, based on the difference between the estimated output and the measured output, becomes large as the discrepancies between normal behavior and fault increase.



(a) (b)

Figure 42: (a) Cumulative rows of data before and after the maintenance event (b) split data into training, test, and anomaly subsets.

Figure 42b shows how the data was split into the three subsets: training, test, and anomaly. The tacit assumption was that the repair would bring the vehicle back to its *normal* state and that no other fault would exist or occur after the repair that could corrupt the "normal" model. It would be preferred to have a few hundred days of data prior to the maintenance event in order to see if the model could track evolution of the damage. However, this was not possible because the maintenance event took place close to the beginning of the HUMS data record.

Table 51: HUMS signals and their symbols

Symbols	HUMS Signals
$\dot{\varphi}$	FuelRate
ω_e	EngSpeed
p_{eo}	EngOilPres
T_{ec}	EngCoolantTemp
τ_{ep}	EngPctTorq
p_{ic}	InjCtlPres
v	VehSpeedEng

Table 51 shows seven signal candidates that were sub-selected from the twenty-four engine signals, listed in Table 52. The sub-selection process, shown as "Identify relevant signals" in Figure 40, first excluded redundant signals captured on J1587, then selected signals that reasonably relate to the engine power generation process. During modeling iterations, signals such as different torque measures (e.g. EngPctTorq vs. EngDmdPctTorq) were added and removed from the list in Table 51, to investigate their potential to improve the model's ability to identify the fault. Because the performance of the model did not improve when the set of input signals from Table 51 was expanded, the final implementation employed just those seven signals.

Table 52: HUMS signals related to engine

	Engine Signals
1	AccelPedalPos
2	EngPctLdAtCurSpd
3	EngPctTorq
4	EngSpeed
5	EngDmdPctTorq
6	EngDesiredOpSpd
7	EngOilPres
8	EngRemPTOGovernorPreProgSpdCtrlSwtch
9	EngRemPTOGovernorVarSpdCtrlSwtch
10	EngRtdPwr
11	EngRtdSpeed
12	EngRetSwtch_1587
13	EngLoad_1587
14	EngOilPres_1587
15	EngCoolantTemp_1587
16	EngSpeed_1587
17	EngCoolantTemp
18	EngCoolantLvl_1587
19	EngOilTemp_1587
20	EngRetPct_1587
21	EngRetLvlSwtch_1587
22	EngRetarderStat_1587
23	FuelRate
24	VehSpeedEng

A daily CDF file consisted of many discontinuities, or data gaps. It is important to identify data gaps, because they signify different driving segments, or "missions". Otherwise, concatenating the tail data from one driving segment with the engine start data of the subsequent driving segment would produce input segments with discontinuities that could confuse the health assessment analysis. In the preprocessing performed here, a discontinuity was defined by any data gap longer than two seconds, because the analysis described in Section 4 showed that normal, continuous data is sampled at 1 Hz, but sometimes also at 2 Hz. This data gap was considered a hyperparameter Δt_{break}, whose value was set to two in our experiments. Figure 43 illustrates multiple signals with data gaps. It shows a part of daily data (contents of a single CDF file), restricted to the signals of interest. The rightmost column of subplots shows time-domain view of the signals; the sub-plots below the main diagonal are scatter plots between two signals, and the diagonal shows histogram of individual signals. Individual continuous segments are indicated by color. The presence of multiple colors shows that there were multiple time gaps greater than two seconds. This specific file had 2,323 records per signal and 25 data gaps, which produced 26 continuous data segments.

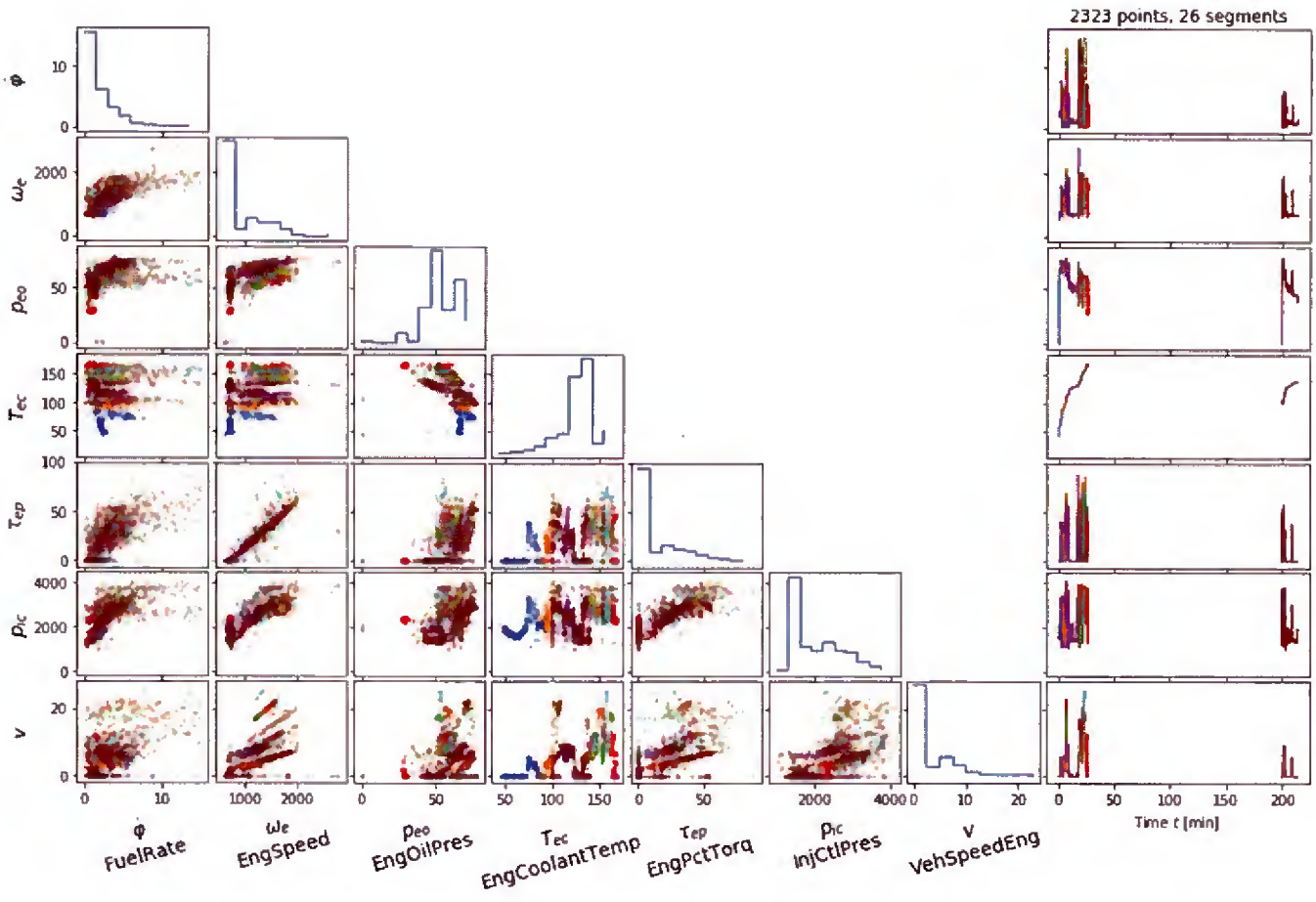


Figure 43: Selected signals and their mutual relationships

The model topology specified some requirements for data preprocessing. Figure 44 (below) shows an autoencoder neural network model that was conceptualized for detecting anomalies. The input vector $\overline{x}$ was defined as $m = 10$ concatenated sub-vectors each of length n as shown in equation 1.

$$\overline{x}^T = \left[\overline{\dot{\varphi}}^T \ \overline{\omega}_e^T \ \overline{p}_{eo}^T \ \overline{T}_{ec}^T \ \overline{\tau}_{ep}^T \ \overline{p}_{ic}^T \ \overline{v}^T \ \overline{\varphi}^T \ \overline{\theta}^T \ \overline{\bar{p}}_{out}^T\right]^T, \tag{3}$$

Based on considerations of the "minimum system dynamics" that needed to be captured in a sample, 30 seconds was selected as the data sample length. Because the data sampling rate was 1s, and because computations associated with deep learning are more efficient when the processor operated on vectors whose length is a power of 2 [18], $n = 32$ was used.

The first seven signals correspond to HUMS signals listed in Table 51 and additional three were calculated sub-vectors. The first calculated signal was fuel consumed over the 32 point data range (φ)

$$\varphi = \int_0^t \dot{\varphi}(t')dt', \tag{4}$$

the second was engine rotation angle (θ)

$$\theta = \int_0^t \omega_e(t')dt', \tag{5}$$

and the third is a measure of engine power output ($\tilde{p}_{out}$)

$$\tilde{p}_{out} = \omega_e \tau_{ep}. \tag{6}$$

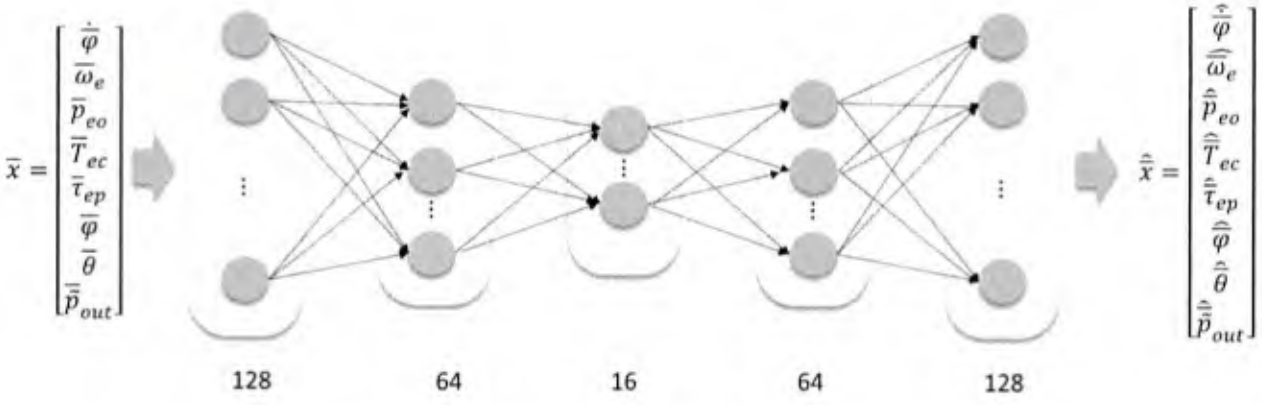


Figure 44: Model configuration

Figure 45 and Figure 46 illustrate the process of mapping the contents of the CDF files into input vectors for the machine learning model of Figure 44. Figure 45 shows the process of identifying continuous data segments (the left part of the illustration) and mapping it into a list of input segments for the model. Figure 46 shows the process that transposed the sub-vectors and concatenated them as prescribed by Eq. (3).

As shown in Figure 40, preprocessing produced training, test, and anomaly datasets. In this implementation, all three datasets, together with associated metadata, were saved in a single HDF5 file. The sizes of the input matrices were:

- Training size$\left(\bar{\bar{X}}_{train}\right)$ = 12465×320
- Test: size$\left(\bar{\bar{X}}_{test}\right)$ = 9243×320
- Anomaly size$\left(\bar{\bar{X}}_{anomaly}\right)$ = 1193×320

The first dimension denotes the number of training samples, whereas the second dimension is the length of the concatenated vector (10 sub-vectors of length $n = 32$).

Figure 45: Transforming the CDF data into a file for training of the autoencoder. Step 1 of 2: identifying discontinuity in the data sets and breaking the continuous segments of the desired length.

Figure 46: Transforming the CDF data into a file for training of the autoencoder. Step 2 of 2: transposing the data.

The signals had to be normalized before they were used for the model. To avoid data snooping, offset and scale were computed from the training data. Refer to [19] for the discussion about data snooping and the problems that can arise if all data is used for normalization. Two standard types of normalization were considered: min-max, and Gaussian.

Because there are m different sub-vectors that correspond to m HUMS and computed signals ($m = 10$ in this example), m offsets and scale parameters were needed; otherwise, the selected units of individual signals can make a single signal dominate the others. In Gaussian normalization, offsets were

$$\mu_k = \frac{1}{nL}\sum_{i=1}^{L}\sum_{j=kn}^{(k+1)n-1} x_{i,j}\ , \quad k \in \{0,1,\dots,m-1\}, \tag{7}$$

and the scales were

$$\sigma_k = \sqrt{\frac{1}{nL-1}\sum_{i=1}^{L}\sum_{j=kn}^{(k+1)n-1}\left(x_{i,j}-\mu_k\right)^2}\ , \quad k \in \{0,1,\dots,m-1\}. \tag{8}$$

The offsets and scale normalized the inputs using the standard equation

$$\hat{x}_{i,j} = \frac{x_{i,j}-\mu_k}{\sigma_k}, \quad k \in \{0,1,\dots,m-1\}. \tag{9}$$

To illustrate the number of unknown parameters that are optimized during training, refer to Figure 44 and consider just the first layer, where 320 inputs were mapped into 256 outputs via an affine transformation

$$\overline{\overline{W}}_{256\times320}\overline{x}_{320\times1} + \overline{b}_{256\times1} \tag{10}$$

before a nonlinear activation function was applied. The affine transformation needed the weight matrix $\overline{\overline{W}}_{256\times320}$ and the offset vector $\overline{b}_{256\times1}$, which together had 256×320+256 = 321×256=82,176 parameters. Thus, the number of parameters associated with the first layer of the network was 82,176. Extending this approach to the entire network (see Figure 44 for the number of neurons in layers) yielded

$$\underbrace{321\times128}_{1^{st}\,layer} + \underbrace{129\times64}_{2^{nd}\,layer} + \underbrace{65\times16}_{3^{3r}\,layer} + \underbrace{17\times64}_{4^{th}\,layer} + \underbrace{65\times128}_{5^{th}\,layer} + \underbrace{129\times320}_{6^{th}\,layer} = 101{,}072 \tag{11}$$

This network, although not particularly deep, required nearly a quarter million training parameters.

5.1.3 Engine model implementation

Keras (TensorFlow) implementation was straightforward. Figure 48 lists the Keras code and the model summary. The model summary shows the layers and the parameters per layer (see also Figure 44 and Eq. (11)). Dropout was used for regularization [20]. The activation function was the commonly used rectified linear unit ReLU [21], defined as

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{12}$$

Selection of the loss function and the type of optimizer affects the model performance. Mean Squared Error (MSE) was used for the loss and Mean Average Error (MAE) for the metric (to compare it later to the loss). The selected optimizer was RMSprop [22].

```
auto_enc = models.Sequential()
auto_enc.add(layers.Dense(128,input_shape = (m*n,)
                          ,activation = "relu"))
auto_enc.add(layers.Dropout(.1))
auto_enc.add(layers.Dense(64,activation = "relu"))
auto_enc.add(layers.Dropout(.1))
auto_enc.add(layers.Dense(16,activation = "relu"))
auto_enc.add(layers.Dropout(.1))
auto_enc.add(layers.Dense(64,activation = "relu"))
auto_enc.add(layers.Dropout(.1))
auto_enc.add(layers.Dense(128,activation = "relu"))
auto_enc.add(layers.Dropout(.2))
auto_enc.add(layers.Dense(m*n,activation = "linear"))
auto_enc.compile(optimizer = optimizers.RMSprop(lr=5e-
                 loss = losses.MSE,
                 metrics = [metrics.MAE,])
auto_enc.summary()
```

Layer (type)	Output Shape	Param #
dense_42 (Dense)	(None, 128)	41088
dropout_35 (Dropout)	(None, 128)	0
dense_43 (Dense)	(None, 64)	8256
dropout_36 (Dropout)	(None, 64)	0
dense_44 (Dense)	(None, 16)	1040
dropout_37 (Dropout)	(None, 16)	0
dense_45 (Dense)	(None, 64)	1088
dropout_38 (Dropout)	(None, 64)	0
dense_46 (Dense)	(None, 128)	8320
dropout_39 (Dropout)	(None, 128)	0
dense_47 (Dense)	(None, 320)	41280

Total params: 101,072
Trainable params: 101,072
Non-trainable params: 0

Figure 47: Keras implementation of the model summary.

5.1.3.1 Training

The training is illustrated in Figure 48. The plot on the left shows the loss (MSE) during training and test vs. the number of epochs (passes of the training dataset through RMSprop optimizer). The plot on the right shows the metric (MAE) vs. the number of epochs. The test error improvement seems to level off significantly after about 10 epochs. It may seem surprising that the training error is higher than the validation error, but this is due to the fact that, using normal TensorFlow/Keras settings, dropout regularization was employed during training but not during validation.

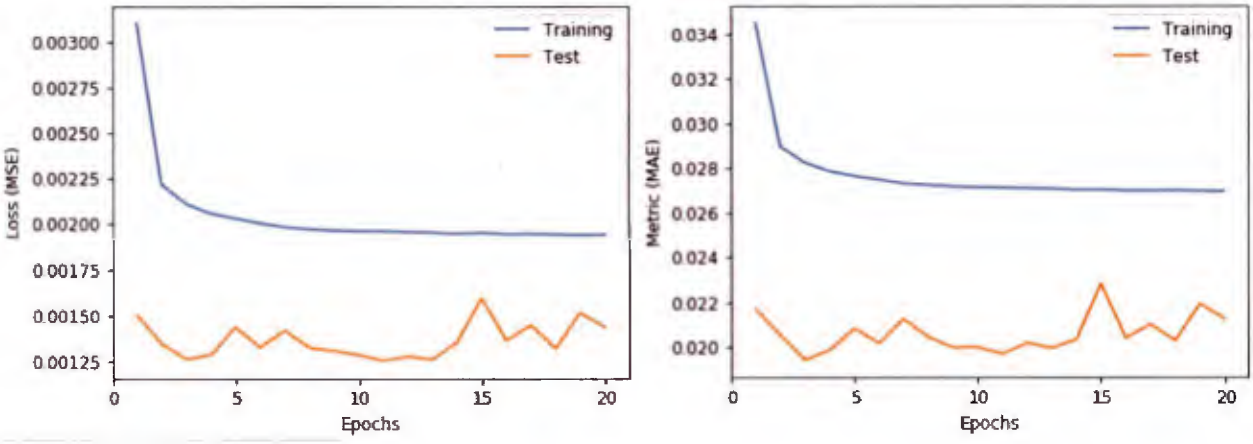


Figure 48: Training and test error during model development.

5.1.3.2 Performance Evaluation

The model represented the data well, as illustrated in Figure 49, which shows the training data (blue traces) and estimated outputs (orange traces). The model output generally looked as a smoothed out version of the input signal, which strongly suggested that overfitting was avoided. The estimation error for rapidly changing signals such as fuel rate $\dot{\varphi}$ or engine speed ω_e was higher than the error corresponding to slower signals like engine coolant temperature T_{ec}.

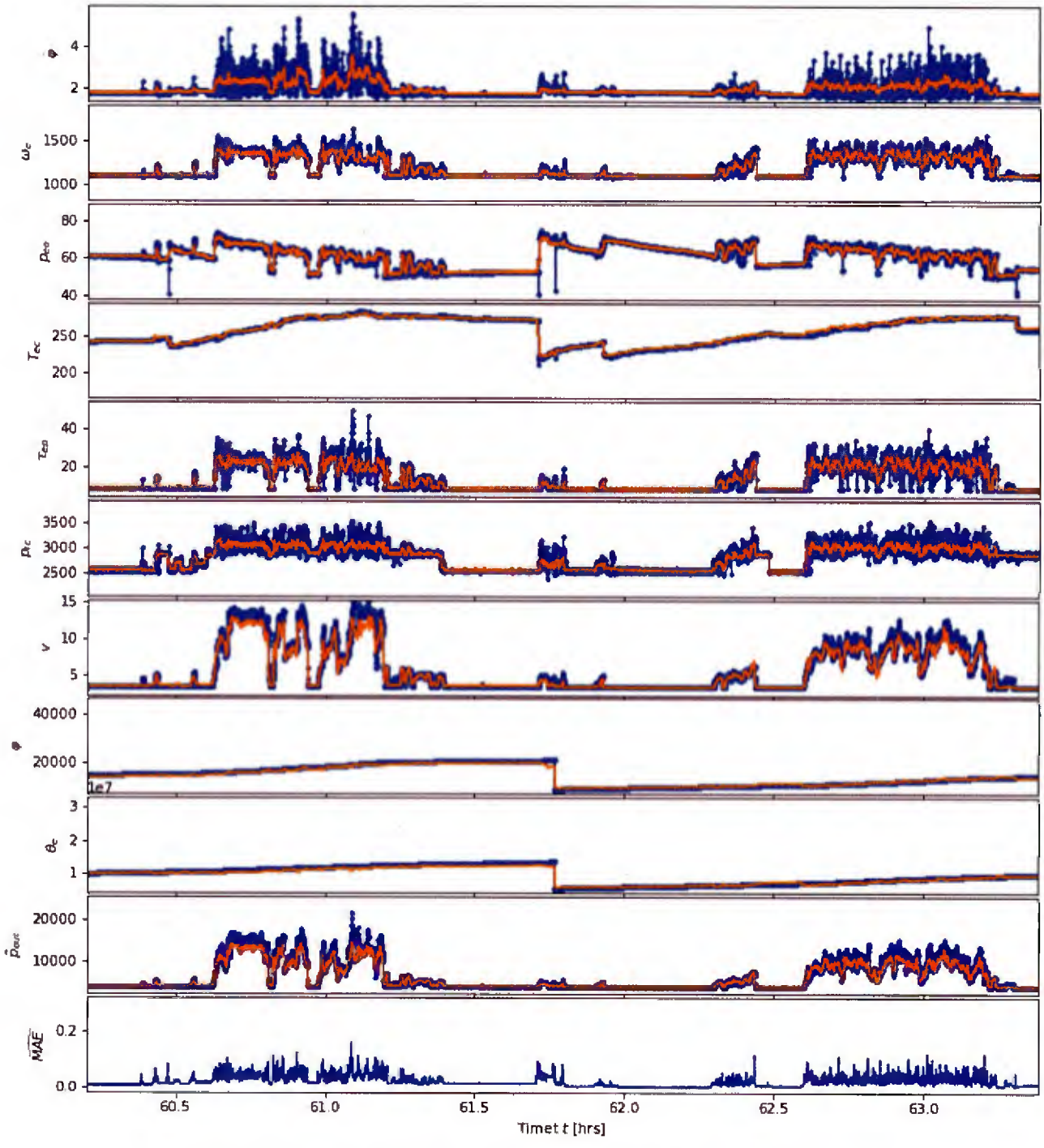


Figure 49: Engine model fidelity illustrated on the subset of signals and the total MAE.

It was more convenient to look at the estimated metric $\widehat{MAE}$ in the form of a histogram than in the time domain, as shown in Figure 50. The histogram form also allowed the error comparison for the train, test, and anomaly data. Table 53 lists the statistical summaries of the three error distributions.

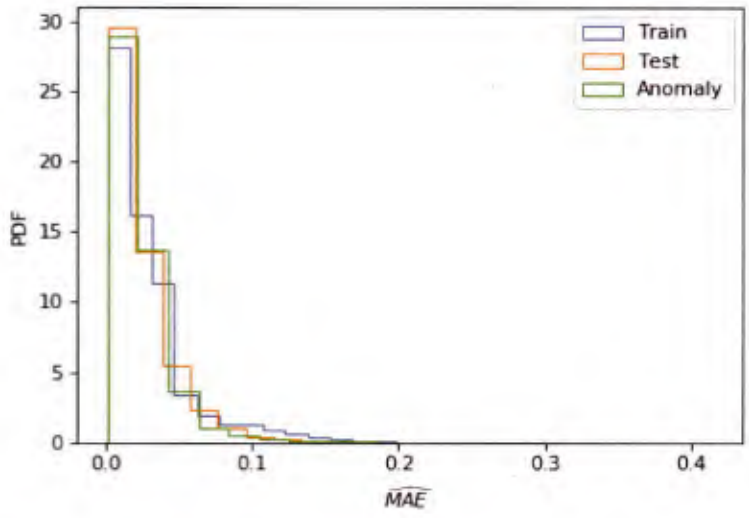


Figure 50: Engine model fidelity illustrated on the subset of signals and the total MAE.

Table 53: Statistical summaries of the deep learning engine model error associated with test, train, and anomaly data.

	μ	σ	max()	data size
ϵ_{train}	0.028817	0.029273	0.277263	381984.0
ϵ_{test}	0.024010	0.022408	0.380039	295776.0
$\epsilon_{anomaly}$	0.022310	0.021393	0.414573	381984.0

The model error distribution associated with the anomaly data was not distinguishable from the error distribution associated with the training and test data, which indicated that the model was not able to detect the anomaly. Several model variations of were tried, but the results remained consistent. It may be possible that the degradation and failure information was not contained in the selected signals; however, all signals thought to be relevant were selected. Therefore, the conclusion is that this content is probably not in the HUMS data set. Since engine torque is not directly measured, the torque and power signals are based on a torque value calculated by the ECU, which may not be sensitive to an injector defect. As stated above, the diesel engine was initially selected for this analysis, as it is a common asset type across the Army, USMC, and Navy platforms. It still may be possible, with additional work, to develop a functional autoencoder based anomaly detector for injector faults; however, additional data signals would be required.

5.1.4 Transmission Model

Since the autoencoder analysis was not successful on the engine data, transmission was considered as an alternative system to use for the autoencoder investigation. As with the engine model development, the process started with analysis of maintenance data, which uncovered an event of interest, listed in Table 54. The problem was related to the transmission ECU, with the maintenance event occurring in December of 2013. The data availability is shown in Figure 51 (compare this plot to that of Figure 41).

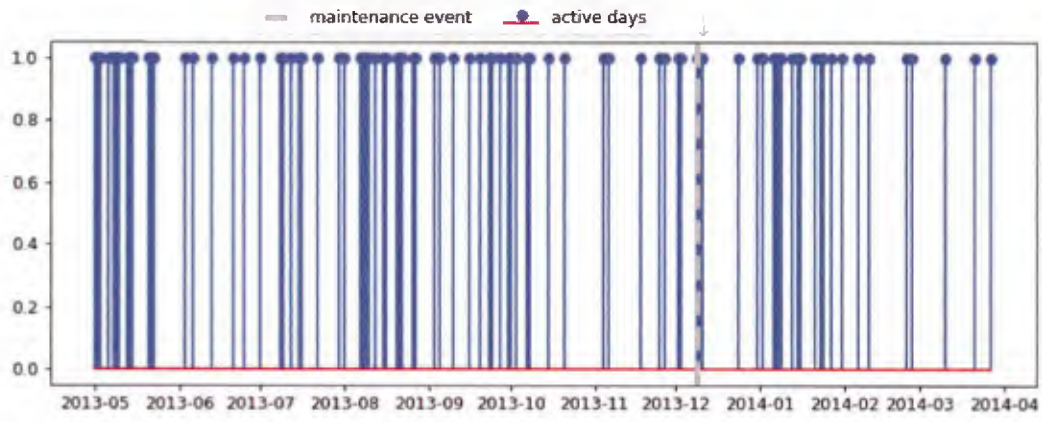
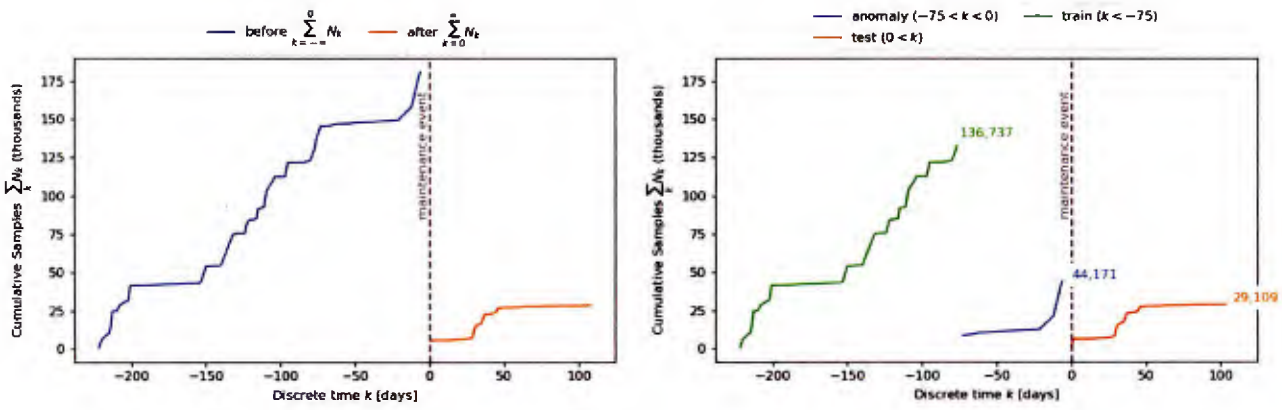


Figure 51: Vehicle daily activities and the maintenance event related to transmission ECU failure.

Table 54: Maintenance events of interest.

Family	FMTV
Fault Date	12/9/2013
Date Out	2/10/2014
Fault Description	TRANSMISSION INOPERATIVE
Correction Narrative	REPLACED TRANSMISSION ELECTRONIC CONTROL UNIT ...
Maintenance Level	FIELD
Action Description	Replaced
Part NSN	3010014609681
Part Nomenclature	CONTROL ASSEMBLY,TA
Part Quantity	1
Total Man Hours	3.1
Total Cost	2568.81

Because there was considerably more data before the maintenance event, the training data was selected from the period before the maintenance event, and test data were selected after the maintenance event. This was a natural data split that could not be made in the previous example because the fault occurred very early in the observation period. The anomaly data set was chosen to be the data collected within 75 days of the maintenance event. Figure 52 shows the data availability and the data split into train (65.1%), test (13.9%), and anomaly subsets (21%).



(a) *(b)*

Figure 52: (a) Cumulative rows of data before and after the transmission maintenance event (b) split data into training, test, and anomaly subsets.

The model was simplified to use just three HUMS signals: engine speed ω_e, vehicle speed v, and transmission oil temperature T_{to}, listed in Table 55. These signals were selected due to the relationship of vehicle and engine speed to transmission gear ratio. Transmission Oil Temp was one of the few signals available that provide insight into transmission condition. The scatter plot of signals for a daily mission is shown in Figure 53.

Table 55: HUMS signals used by the transmission model and their symbols

Symbols	HUMS Signals
ω_e	EngSpeed
v	VehSpeedEng
T_{to}	TransOilTemp

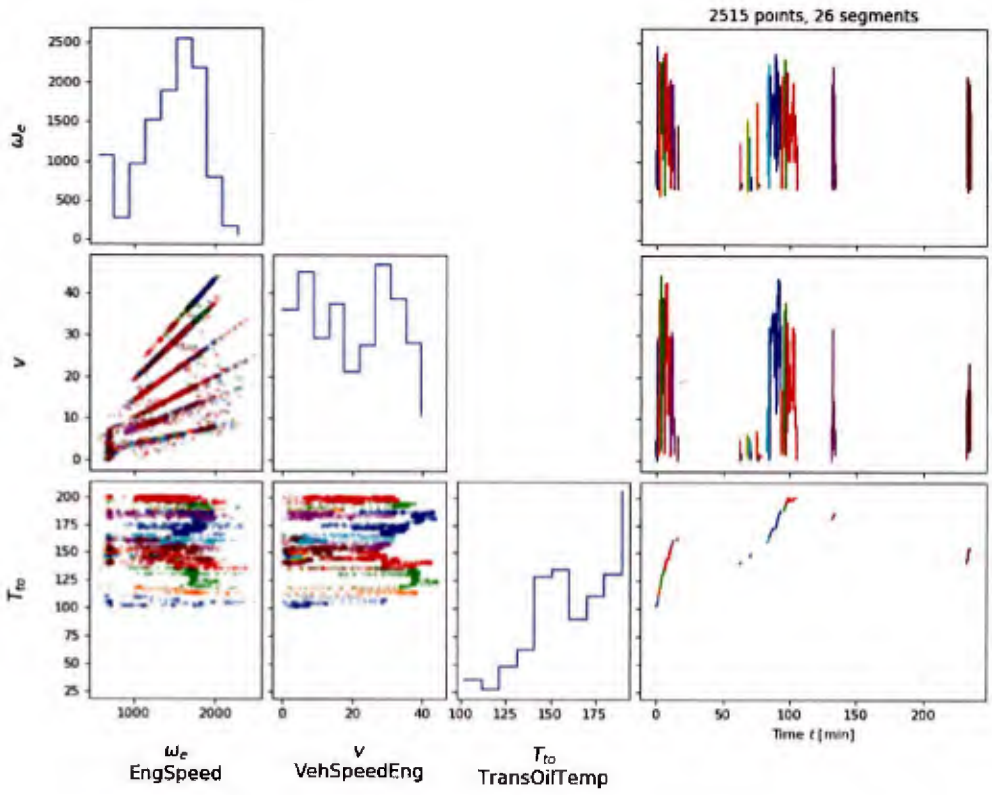
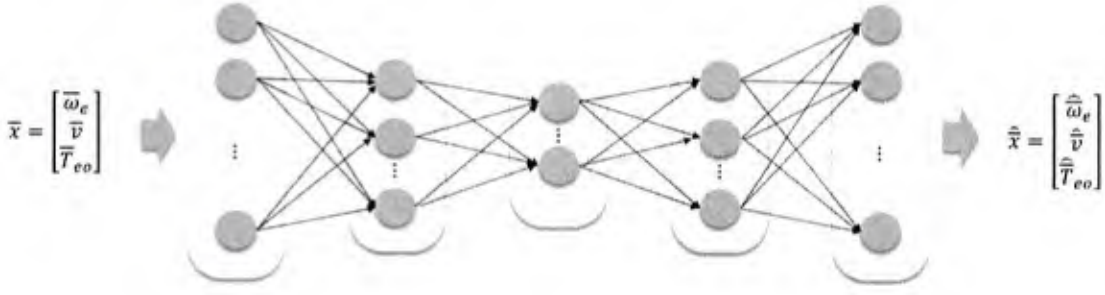


Figure 53: Selected signals and their mutual relationships for the transmission model.

Thus, the input vector was given as

$$\overline{x}^T = \left[\overline{\omega}_e^T \quad \overline{v}^T \quad \overline{T}_{oil}^T\right]^T. \tag{13}$$

with the model topology shown in Figure 54, with the summary from Keras implementation. The total number of parameters was 41,376.



Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 128)	12416
dropout_4 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 64)	8256
dropout_5 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 128)	8320
dropout_6 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 96)	12384

Total params: 41,376
Trainable params: 41,376
Non-trainable params: 0

Figure 54: Transmission dense autoencoder model with the summary from Keras

The model fidelity for the three signals was very good, as is illustrated in Figure 55. Like the engine example, the blue trace signifies the data and the orange trace the model output. The figure demonstrates that the lossy compression of encoding does not suppress relevant information of the signals. This data segment was extracted from the training data, and the error comparison across the training, validation, and test data sets was explored next.

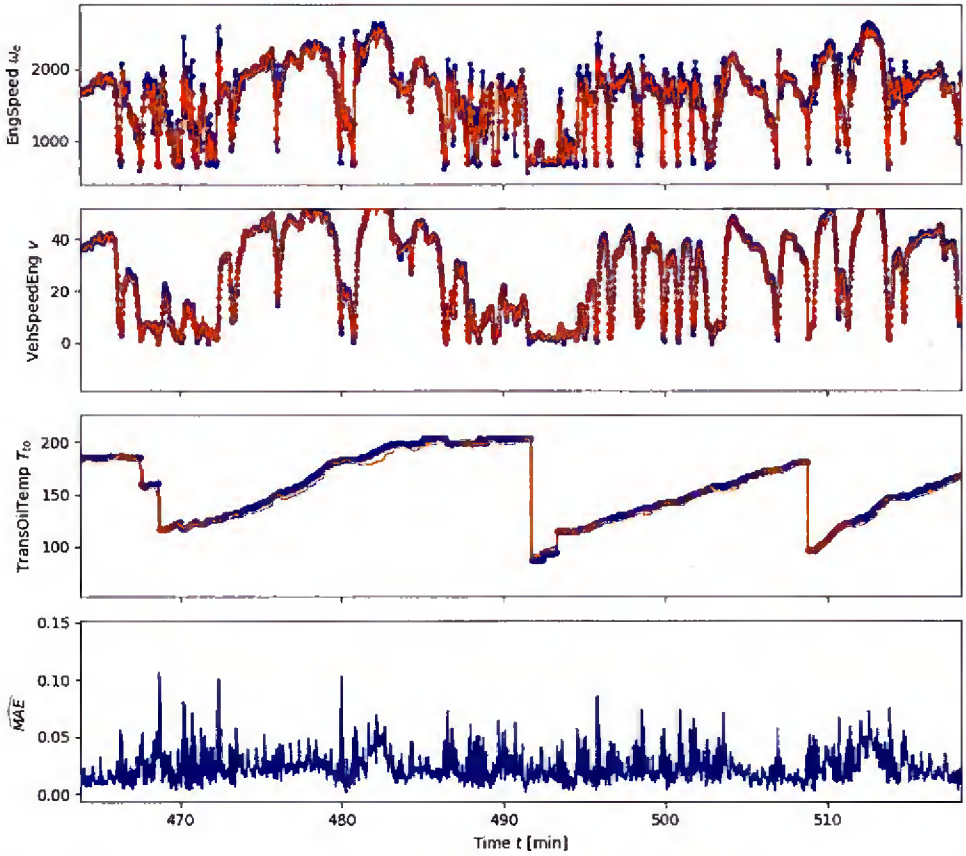


Figure 55: Transmission dense autoencoder model performance shown on a segment of training data.

The error distribution associated with the train, test, and anomaly data clearly showed that the anomaly data exhibited a tail ($\widehat{MAE} > 0.15$) that was inconsistent with the train and the test data (Figure 56).

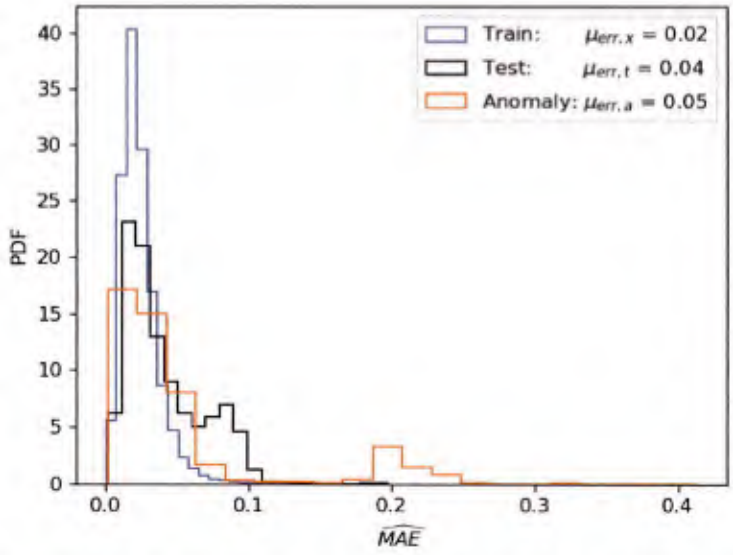


Figure 56: Error distribution for the transmission model and engine ECU failure.

Table 56 lists the statistical summaries of the error distributions shown of Figure 56.

Table 56: Statistical summaries of the deep learning engine model error associated with test, train, and anomaly data.

	μ	σ	max()	data size
ϵ_{train}	0.023655	0.012934	0.144762	63936.0
ϵ_{test}	0.040483	0.026620	0.197038	8320.0
$\epsilon_{anomaly}$	0.052857	0.061362	0.413776	13280.0

Figure 57 shows these errors in the time domain. The error rises significantly before the maintenance event (denoted by the gray, dashed, vertical line). The time was concatenated across all driving missions, and the time gaps associated with no usage were deleted. Although the increase in error is obvious at the time of the failure, there appear to be some elevated error events prior to the fault (time = ~1100). The next section covers how anomaly detection may be used in decision support.

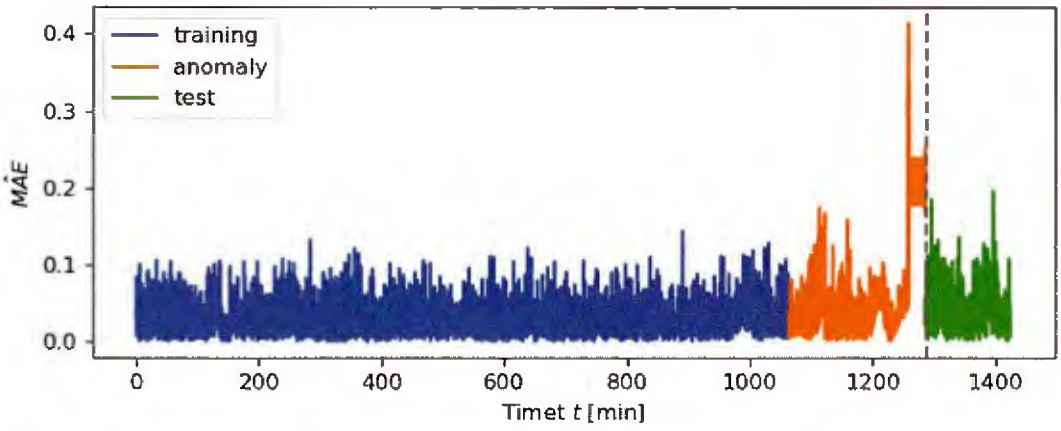


Figure 57: $\widehat{MAE}$ over time

5.1.5 Anomaly Interpretation

An autoencoder anomaly detector can produce a metric that indicates an anomaly, but this metric needs to be interpreted by a decision support tool. To interpret the metric, RIT employed the Sequential Probability Ratio Test (SPRT) attributable to Abraham Wald (WW2) [23, 24], and introduced to PHM in the context of monitoring nuclear power plants [25] and later applied to other PHM applications, such as monitoring of software failures in servers [26].

To apply the SPRT approach, first the log-likelihood ratio of the error metric ϵ

$$\ell = \sum_i \log \mathrm{P}(\epsilon_i | Faulty) - \sum_i \log \mathrm{P}(\epsilon_i | Healthy) \ , \tag{14}$$

where ϵ_i is an $\widehat{MAE}$ error sample and the operator P is probability, was computed. The probability of a healthy state can be readily estimated from the training data (because the training data corresponded to the assumed healthy state of the system). Log-normal distribution was a good fit to the error data, as shown in Figure 58.

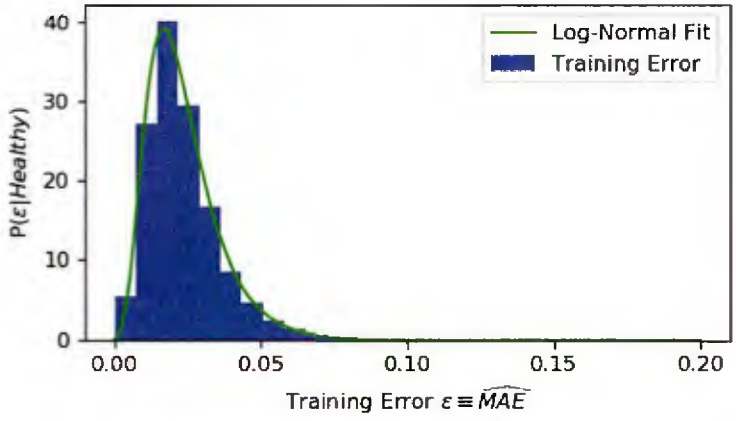


Figure 58: Estimating $P(\epsilon|Healthy)$ from the data

The probability associated with the faulty state was not known in advance. To express the state of ignorance, Laplace's principle of insufficient reason and model $\mathrm{P}(\epsilon_i|Faulty)$ was used as a uniform

distribution, with the upper range set to be twice the range of the input data. Figure 59 shows $\mathrm{P}(\epsilon_i|Faulty)$ in the same graph with $\mathrm{P}(\epsilon_i|Healthy)$ to facilitate comparison.

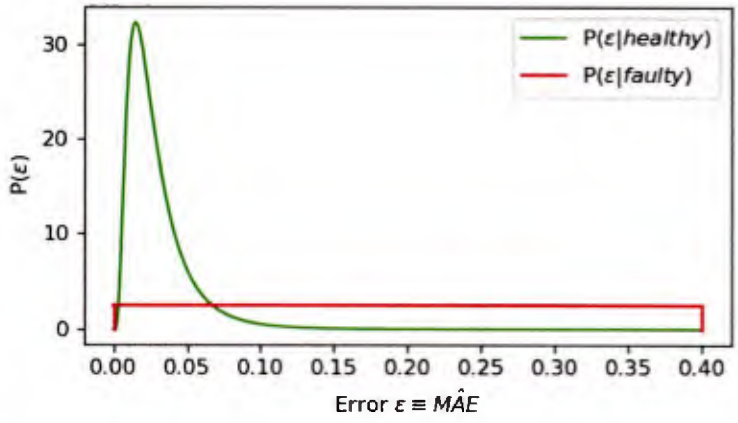


Figure 59: Modeled probability distributions of healthy and faulty data

After the log-likelihood ℓ was defined, it required the lower limit ($\log B$) and the upper limit ($\log A$) to be compared. The decision was then made as follows:

If $\ell < \log B$, decided $Healthy$

If $\log A < \ell$, decide $Faulty$

If $\log B \leq \ell \leq \log A$, continue testing (insufficient data for making the decision)

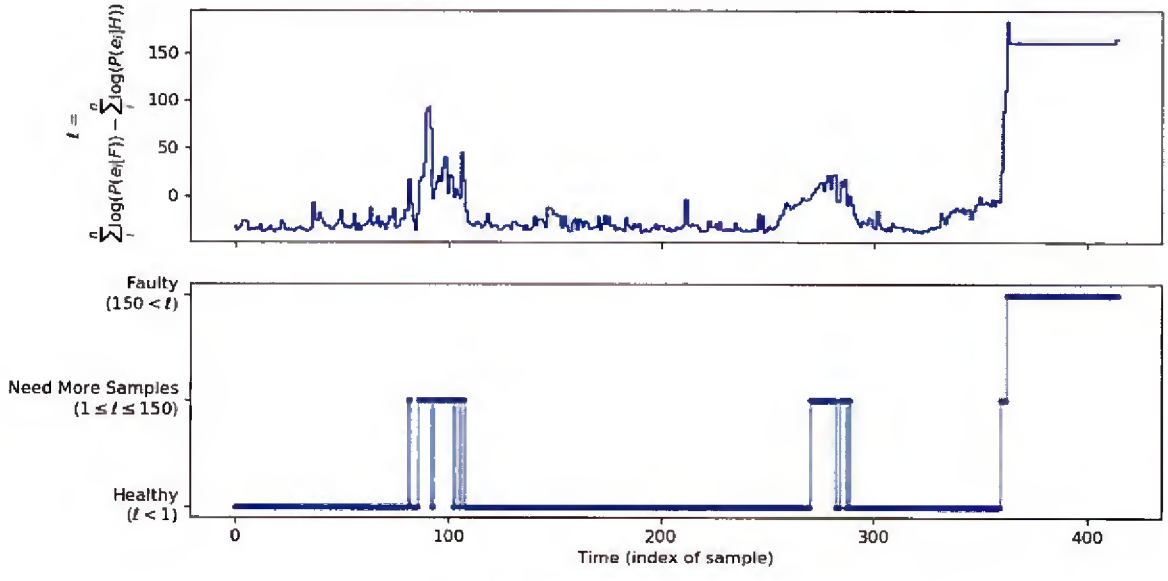


Figure 60: Modeled probability distributions of healthy and faulty data

The decision process is illustrated in Figure 60: the top graph shows the log-likelihood ratio over samples and the bottom shows the comparison to the two heuristically selected limits. This decision process, applied to the original problem, is shown in Figure 61.

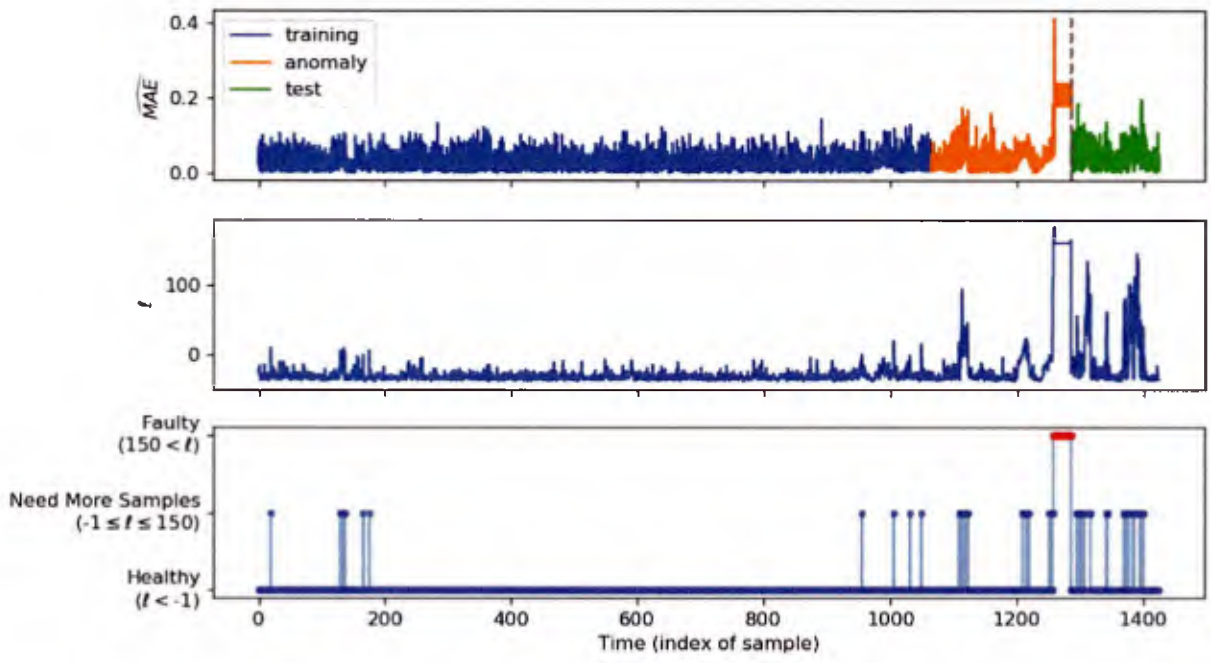


Figure 61: Modeled probability distributions of healthy and faulty data

5.1.6 Model Applicability and Uses

To test the generality of the model, the autoencoder anomaly detector was applied to the data generated by a different vehicle with a similar ECU failure. Figure 62 shows the data and the model outputs. The model was able to reliably identify contradictory data: $\widehat{MAE}$ became large multiple times. Utilizing these anomalies, a human-in-the-loop was able to explain the anomalies: vehicle speed cannot be greater than zero while the engine speed is zero, i.e. the condition $v_{veh} > 0$ and $\omega_{eng} = 0$ is a contradiction in normal operation. The power of representation learning was demonstrated here as this condition had not been defined by an expert rule, but was simply learned from how the signals relate to one another during normal operation. A subject matter expert can easily understand the anomaly but may not have easily noticed a problem without the addition of an anomaly detector.

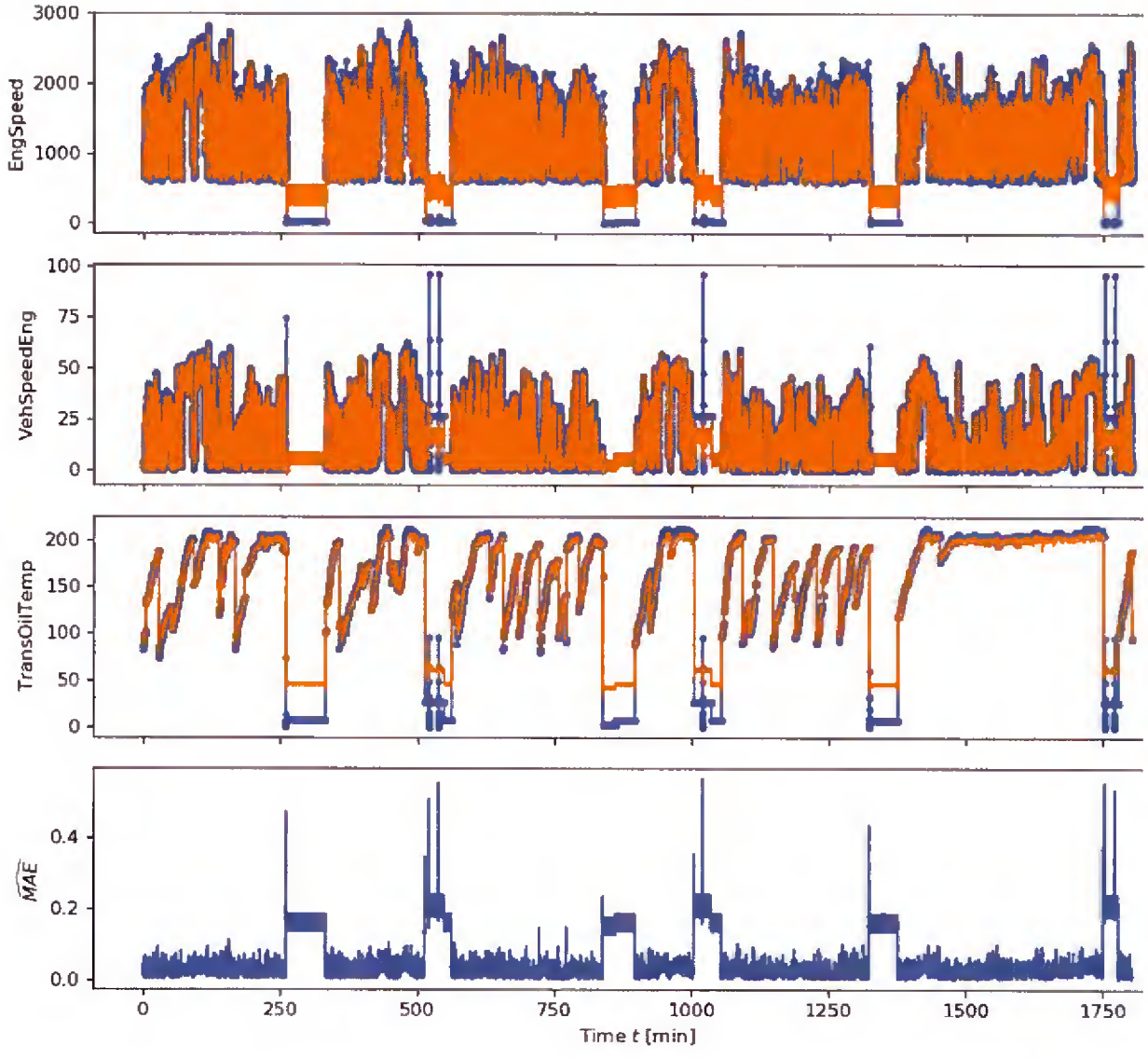


Figure 62: Applying anomaly detection model to a different vehicle with a transmission ECU problem.

Demonstrating the ability of the model to detect an anomaly on an asset that was not used at all during training was encouraging. In this case, the model readily detected an anomaly without any adaptation of a model to a new asset.

More detailed investigation was taken into peaks that appeared to be precursors to failure in the first example, shown in Figure 57, specifically the time $t \sim 1100$ min. The situation there was not as severe (see Figure 63), but patterns of idling engine accompanied with significant speed were detected, suggesting that these were very plausible intermittent precursors to failure. The degradation seemed to be gradual, but not monotonic, which is very common among electronics equipment [27]. This observation may have a potential to be converted into a physics-based condition indicator for this specific failure. Perhaps this approach can be used more often, even converted into a process, where anomaly detection is used to guide physics-based modeling by drawing attention to situations where added scrutiny can yield a compact condition indicator. The tentative process would consists of three steps: 1)

detect an anomaly using a general, autoencoder-based deep learning model; 2) employ domain knowledge to interpret the anomaly and investigate possibility for a direct classification of the anomaly (diagnostics); 3) formulate a physics-based model that paves the path to RUL estimation.

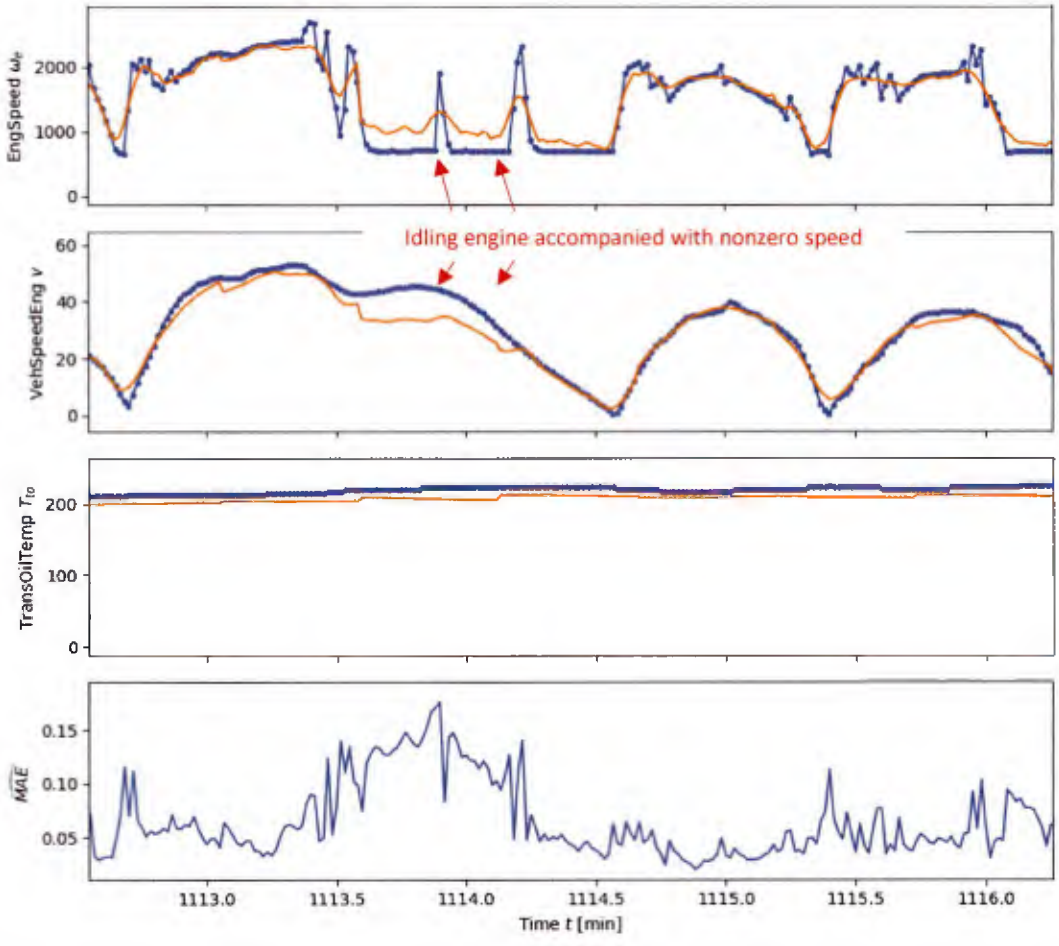


Figure 63: Zoom into precursors to failure shown in Figure 57. The inconsistency in data is annotated in red

5.2 Convolutional Models

The autoencoder described above employed *dense* layers, which feature large number of parameters. This type of neural network is referred to as a Multi-Layer Perceptron (MLP). Convolutional Neural Networks (CNNs) are more compact than MLPs and better exploit computational advantages of Graphical Processing Units (GPU). In addition, the linear part of the layers of these models, as described in more detail in the next subsection, are based upon linear filters, which have a long tradition in modeling linear-time invariant systems. The new aspect of neural networks is that they add a nonlinearity in the form of activation functions after the weighted sum of linear filters, as described in more details below (see Eq. (16) and Figure 65). In the following subsections, the topology of these models is discussed, the models are demonstrated and compared to the transmission example previously modeled and MLP, and CNNs are applied on a new example from the Navy data set.

5.2.1 1D CNN Model

The topology of a 1D CNN is very similar to that of its more popular 2D CNN counterpart. Figure 64 depicts the basic model structure: the model is a series of repeated basic blocks.

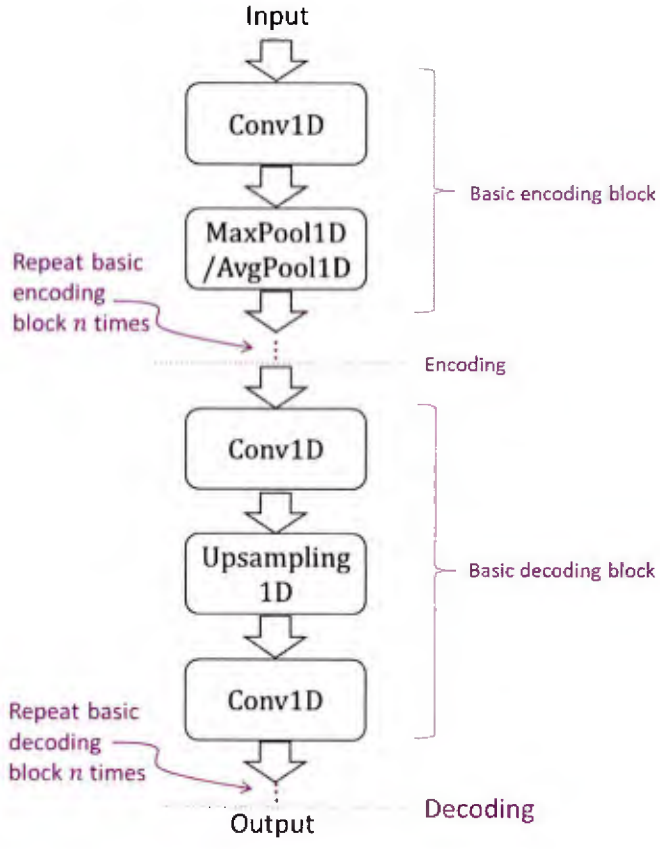


Figure 64: Basic CNN model topology

The basic block for encoding consists of Conv1D layer and a MaxPool1D/AvgPool1D layer, whereas the basic decoding block consists of Conv1D layer and an Upsampling1D layer. Conv1D implements a series of functions given by Eq. (16) and depicted in Figure 65. MaxPool1D downsamples the data by taking a maximum and Upsampling1D is its nonlinear inverse (it repeats the same value multiple times).

A 1D CNN model was built as an alternative to the MLP. Here, a hidden layer unit, ϕ_m, which is a part of a hidden layer

$$\bar{\phi} = [\phi_1 \dots \phi_m \dots \phi_M]^T, \tag{15}$$

is

$$\phi_m = f_m\left(b_m + \sum_{n=1}^{N} s_n * h_{n,m}\right), \tag{16}$$

where f_m is the activation function (e.g. ReLU), b_m is the bias, $h_{n,m}$ are coefficients of 1D digital filters, and $*$ is 1D convolution operator

$$s_n[k] * h_{n,m}[k] = \sum_{i=-\infty}^{\infty} s[k-i]h_{n,m}[i] \tag{17}$$

Graphical representation of Eq. (16) is shown in Figure 65.

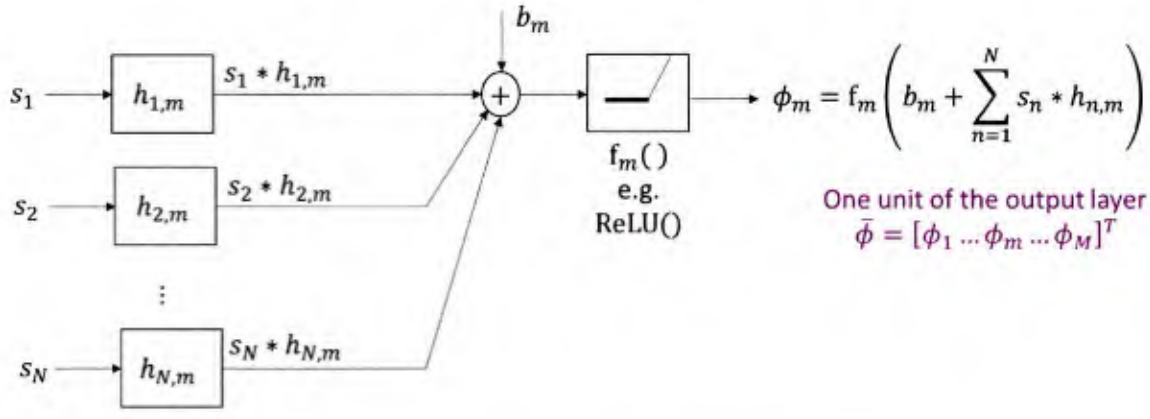


Figure 65: A graphical representation of a hidden unit in 1D CNN.

5.2.1.1 Data Preprocessing

The 1D CNN model employed a different input size than the MLP: MLP's input was a concatenated 1D array (see Eq. (3) and Eq. (13)), whereas 1D CNN's input sample was a 2D array, with one 1D array per signal given by

$$\overline{\overline{X}}_{n\times m} = \left[\overline{\omega}_e^T \quad \overline{v}^T \quad \overline{T}_{oil}^T\right]. \tag{18}$$

The size of the 2D array is $n \times m$, where in the present example $m = 3$, corresponding to three input signals (engine speed ω_e, vehicle speed v, and transmission oil temperature T_{to}). Each signal is represented by n consecutive samples, e.g. $n = 64$. Thus, the input sample is a 2D array of size 64×3.

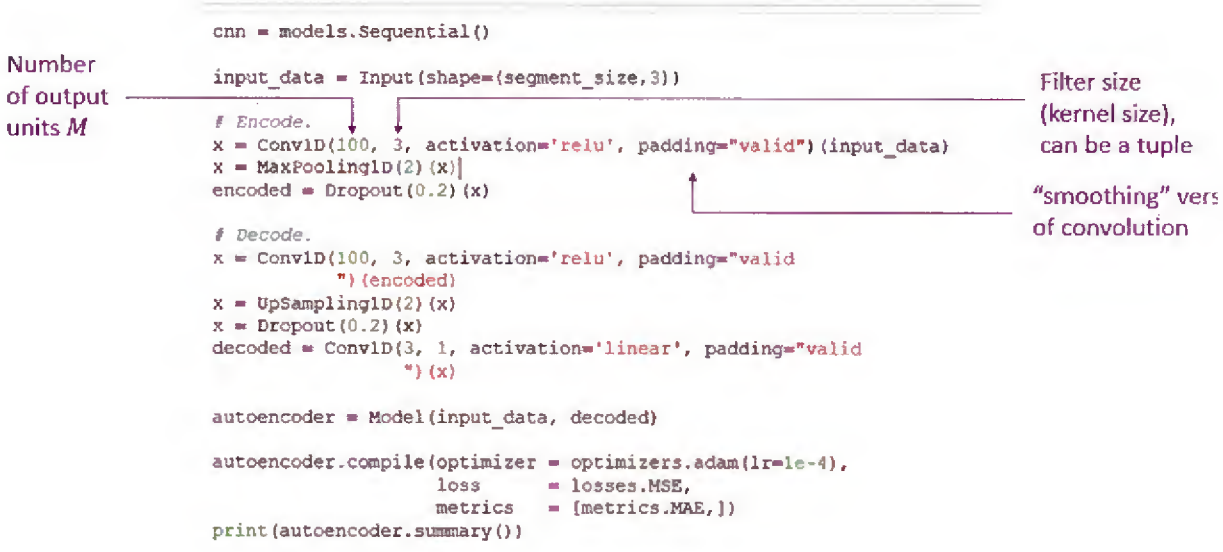
```
def segment_data_into_3d_array(data_2d, segment_size=32):
    '''Take samples of size 'segment_size' of 2D array and stack in a new 3rd array dimension.

    For example, if the original array is (64 x 3) then it will be segmented into (2 x 32 x 3)
    where the 2 are 2 samples of size 32x3 from the original array.
    '''
    rem = len(data_2d) % segment_size          # Get number of points to truncate to make an even segmentation.
    print(f"{rem} data points truncated to make equal segments")
    if rem == 0:
        pass                                   # Don't truncate data.
    else:
        data_2d = data_2d[:-rem]               # Truncate data.
                                               # Set up zeros array to collect the segments along a new 3rd axis
    segmented_data = np.zeros(shape=(len(data_2d)//segment_size, segment_size, 3))
    for i in range(segmented_data.shape[0]): # Iterate over the "sample" dimension (1st axis).
                                               # Add 32-size segments to new 3D array.
        segmented_data[i] = data_2d[i*segment_size:i*segment_size+segment_size]
    return(segmented_data)
```

Figure 66: Preprocessing for 1D CNN

The function for preprocessing is shown in Figure 66. It builds a 3D array by stacking 2D input samples into a third dimension.

A simple, shallow implementation of an autoencoder based on 1D CNN in Keras is shown in Figure 67. This model is not deep, but it is relatively wide – the first Conv1D layer had $M = 100$ units (see Eq. (15)). Figure 68 shows the model summary and the number of parameters. The total number of parameters in the model is 31,403, which is comparable to 41,376 of MLP, shown in Figure 54.




```
cnn = models.Sequential()

input_data = Input(shape=(segment_size,3))

# Encode.
x = Conv1D(100, 3, activation='relu', padding="valid")(input_data)
x = MaxPooling1D(2)(x)
encoded = Dropout(0.2)(x)

# Decode.
x = Conv1D(100, 3, activation='relu', padding="valid
          ")(encoded)
x = UpSampling1D(2)(x)
x = Dropout(0.2)(x)
decoded = Conv1D(3, 1, activation='linear', padding="valid
                ")(x)

autoencoder = Model(input_data, decoded)

autoencoder.compile(optimizer = optimizers.adam(lr=1e-4),
                    loss      = losses.MSE,
                    metrics   = [metrics.MAE,])
print(autoencoder.summary())
```

Figure 67: Keras implementation of 1D CNN

5.2.2 1D CNN Model Performance

The CNN model performance was very similar to the performance of the MLP model. Figure 69 shows the model prediction overlaying the input data for the three signals and overall MAE. Figure 69 shows the fidelity of the model on the training data. This may be compared to Figure 55, which illustrates the fidelity accomplished using an autoencoder based upon MLP. In both the MLP and CNN models, the MAE was typically less than 0.05. In both cases, the models were trained on baseline data to learn normal behavior. Thus, when the system starts to behave abnormally, the input and predicted values will begin to grow apart. However, to properly train a model, it is important that baseline data include all relevant

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 32, 3)	0
conv1d_1 (Conv1D)	(None, 32, 100)	1000
max_pooling1d_1 (MaxPooling1	(None, 16, 100)	0
dropout_1 (Dropout)	(None, 16, 100)	0
conv1d_2 (Conv1D)	(None, 16, 100)	30100
up_sampling1d_1 (UpSampling1	(None, 32, 100)	0
dropout_2 (Dropout)	(None, 32, 100)	0
conv1d_3 (Conv1D)	(None, 32, 3)	303

Total params: 31,403
Trainable params: 31,403
Non-trainable params: 0

Figure 68: Keras 1D CNN model summary

operating and environmental conditions to avoid situations where the model responds to the context changes.

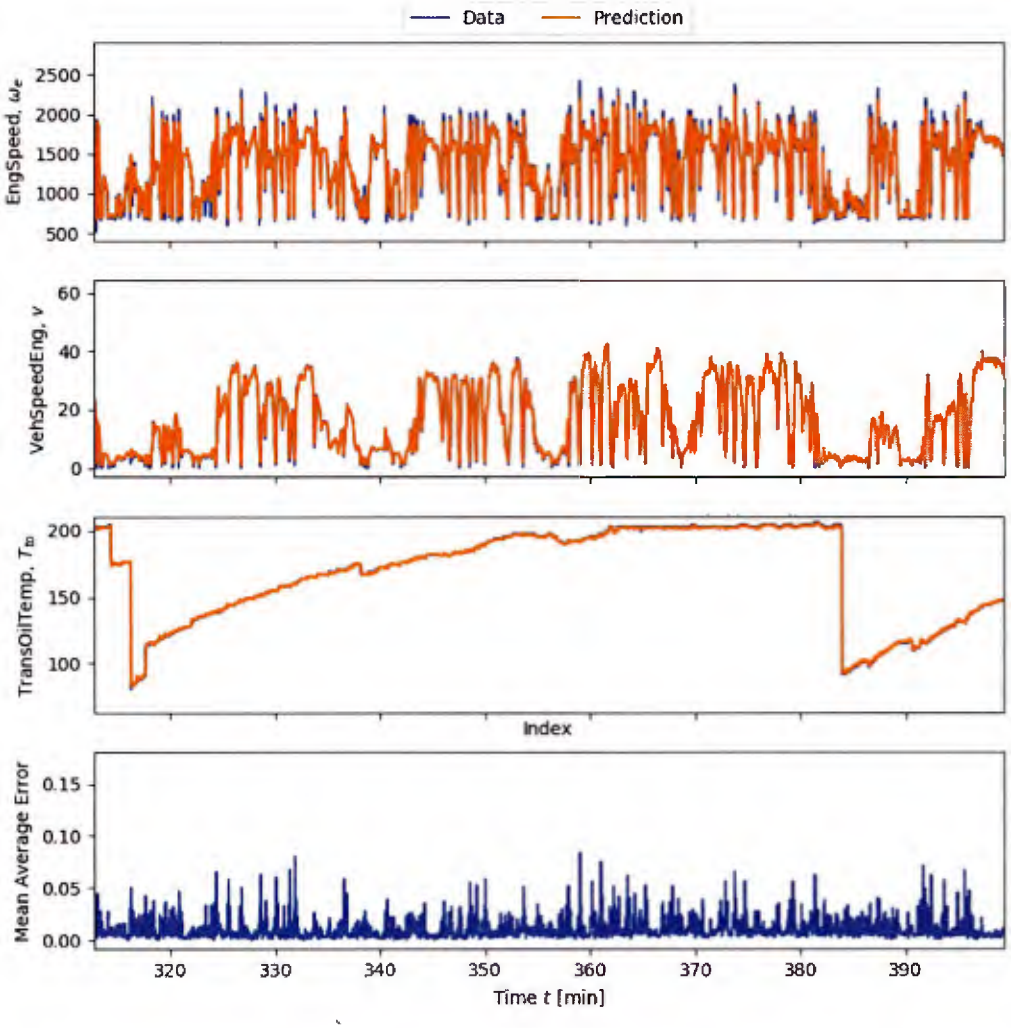


Figure 69: Representation capability of 1D CNN

5.3 Condition Indicators using Autoencoders

Whereas the previous sections describe the suitability of deep learning autoencoders in anomaly detection applications using data that may be separated into normal and anomalous behavior, the Navy data cannot be easily separated in such a manner. Because the Navy ship data consisted of the HUMS data that preceded failures, with no delineation of normal operation, the modeling approach had to be modified. That is, instead of training on normal data, the models were trained on an anomaly in progress (or incipient failure), with the objective to train the model encodings (the outputs of the middle, inner

layer) to be sensitive to the failure, as conceptually depicted in Figure 70. This approach was intended to train the Condition Indicators (CIs) associated with the event type (i.e. failure mode).

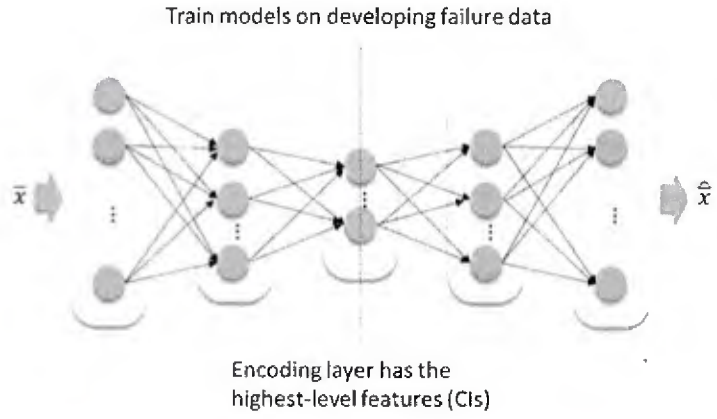


Figure 70: Innermost layer– the encodings – of an autoencoder trained on incipient failure are CI candidates

5.3.1 Data Description

The data consisted of three types of events, denoted by integers 1-3. Each event signified a unique failure. The 55 events are summarized in Table 57. This table shows the distribution of the events across the engines within a ship.

Table 57: Summary of events

Engine	Event type	# Events
1	1	8
1	2	10
1	3	3
2	1	2
2	3	3
3	1	4
3	3	1
2	2	3
3	2	6
4	1	5
4	2	9
4	3	1

The recognition that differences among ship engines and their instrumentation can be significant (see Section 5.5) guided the decision to start building a model for the case where the same engine on the same ship had experienced the same event type multiple times.

Table 58: Events by ship and engine.

Hull	Engine	Event Type	Data Start	Data End	# Data Rows
A	1	1	[2013-04-18]	[2013-07-01]	[595]
A	1	2	[2013-04-18]	[2014-05-08]	[1049]
A	1	3	[2013-04-18]	[2015-01-16]	[1284]
A	2	1	[2013-06-06]	[2015-08-07]	[3711]
A	2	3	[2015-03-06]	[2016-03-11]	[3771]
A	3	1	[2014-10-30]	[2015-08-07]	[858]
A	3	3	[2014-10-30]	[2016-03-11]	[1241]
B	1	1	[2013-05-03]	[2013-12-13]	[548]
B	1	2	[2013-05-03, 2013-05-03]	[2015-02-22, 2015-10-06]	[1012, 2728]
B	2	2	[2015-02-18]	[2015-10-08]	[1442]
B	3	2	[2013-05-03, 2013-10-15, 2013-12-17]	[2013-09-03, 2015-10-08, 2016-02-26]	[904, 3067, 4142]
B	4	1	[2013-08-28, 2013-10-10, 2013-11-02]	[2013-12-13, 2015-08-25, 2015-11-10]	[1641, 2346, 3826]
B	4	2	[2013-05-03, 2014-03-09, 2014-03-16]	[2013-09-02, 2015-09-05, 2015-10-06]	[850, 2115, 2903]
C	1	3	[2014-05-16]	[2015-03-28]	[2188]
C	2	1	[2013-05-24]	[2013-09-16]	[317]
C	2	2	[2013-05-24, 2014-06-08]	[2013-08-21, 2016-06-16]	[243, 2478]
C	3	2	[2013-05-10]	[2013-08-21]	[168]
C	4	2	[2013-05-10, 2014-07-07]	[2013-08-21, 2017-01-21]	[203, 3201]
C	4	3	[2013-05-10]	[2013-07-31]	[168]
D	1	1	[2015-02-13, 2015-03-27, 2015-04-26]	[2015-07-04, 2015-08-11, 2015-09-08]	[3281, 3303, 2824]
D	1	2	[2013-05-03, 2015-04-26, 2016-06-28]	[2013-08-31, 2015-09-08, 2017-01-23]	[205, 2824, 3958]
D	2	3	[2013-05-02, 2013-05-02]	[2015-02-17, 2016-10-12]	[1718, 2881]
D	4	2	[2013-05-02, 2015-03-24, 2015-04-03]	[2015-03-13, 2016-06-29, 2016-09-13]	[1933, 3950, 4030]
F	1	1	[2013-05-17, 2013-05-17, 2016-04-12]	[2013-11-01, 2014-01-25, 2017-05-23]	[529, 640, 9225]
F	1	2	[2013-05-17, 2014-04-02, 2015-02-05]	[2014-01-25, 2014-10-31, 2016-08-07]	[640, 3092, 4032]
F	1	3	[2013-05-17]	[2014-02-04]	[654]
F	3	1	[2013-05-16, 2016-03-09]	[2014-04-10, 2017-03-07]	[924, 6413]
F	3	2	[2013-05-17, 2014-05-11]	[2014-01-25, 2014-11-02]	[594, 3284]
F	4	1	[2013-05-17, 2015-04-10]	[2014-01-25, 2017-03-07]	[594, 6968]
F	4	2	[2016-04-12]	[2017-05-02]	[7077]
I	1	2	[2016-11-05]	[2017-04-10]	[10525]
I	3	1	[2014-02-20]	[2015-02-05]	[1333]

Table 58 provides a summary of the event data in a format that facilitates the selection of the events to build the model. All events are shown for each distinct Hull-Engine-Event Type combination. The second consideration was that there would be significant amount of data leading into the events. A good candidate was found for Hull=B, Engine=3, and Event type=2, with three occurrences (refer to the

highlighted row of Table 58). Furthermore, one of the three occurrences associated with this Hull-Engine-Event Type combination had the largest data set associated with one of the events.

The relevant signals are listed in Table 59. The entire set of 49 signals was used as input for the autoencoder model. Because the sampling period was 15 minutes, capturing dynamics was not considered in the model. Thus, the input vector consisted of a single sample of each signal.

Table 59: Navy engine signals.

Signal Description	No. Signals
Bearing Temperatures 1-9	9
Thrust Bearing Temperature	1
Cylinder Temperatures 1-16	16
Central Fresh Water Left Out & Right In Temperatures	2
Crankcase Pressure	1
Engine Speed	1
Fuel Rack	1
Jacket Water Cooler Out Temperature	1
Left and Right Bank Air Temperature	2
Left and Right Bank Air Pressure	2
Left Bank Exhaust Temperature	1
Left and Right Turbocharger Exhaust Temperature	2
Left and Right Turbocharger Oil Temperature	2
Left and Right Turbocharger Speed	2
Right Turbocharger Air Out Pressure	1
Lube Oil Cooler In Temperature	1
Lube Oil Inlet Temperature	1
Rocker Lube Oil Supply Pressure	1
Rocker Lube Oil Supply Temperature	1
Rocker Lube Oil Strainer Delta Pressure	1
Total Signal Count	49

5.3.2 Experimentally Determined Condition Indicators for Failure

5.3.2.1 Motivation and Approach

The middle layer of an autoencoder neural network contains the most compressed data representation and is referred to as *codings* [28]. Because they are the most efficient data representation for a given model, the codings outputs are considered to be good feature (or CI) candidates [16]. Data-driven PHM is based on the assumption that data collected before a maintenance event contains information on fault precursors, which evolve together with the degradation [29]. Idealized degradation is depicted in Figure 71.

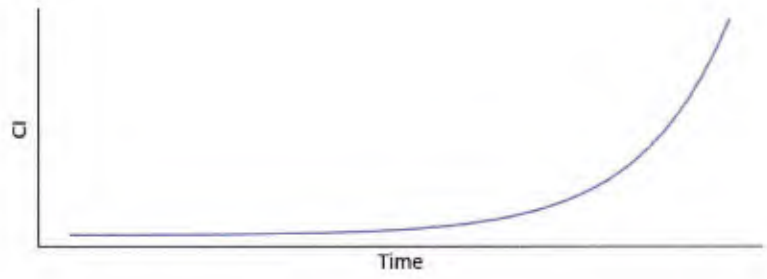


Figure 71 - Idealized failure degradation

CI candidates were selected experimentally, based on consistency among the models, because the exact onset of failure was unknown. Furthermore, the heuristic used to identify plausible CI candidates was based on the assumption that the degradation increased as the system approached the maintenance event. Several different topologies of an autoencoder were run on data from a given ship, event type, and date, and model structure and trained weights were saved in a file. The following sections describe the modeling process, the choice of metric, and discuss the results.

5.3.2.2 Modeling

Auto-encoders were generated using Keras [30, 31], an open source Python library for modeling neural networks, with a TensorFlow [32] backend. In general, two models were created, each with different topologies (labeled Model I and Model II), and five runs of each model (sub-labeled A, B, C, D and E) were performed. Neural networks are inherently random in training weights, so running the same model more than once will return similar results, with differently learned weights.

Figure 72 shows the amount of data collected over time for the ship (or hull) labeled "B". This event was identified as event type 2 on engine 3 on 02-26-2016.

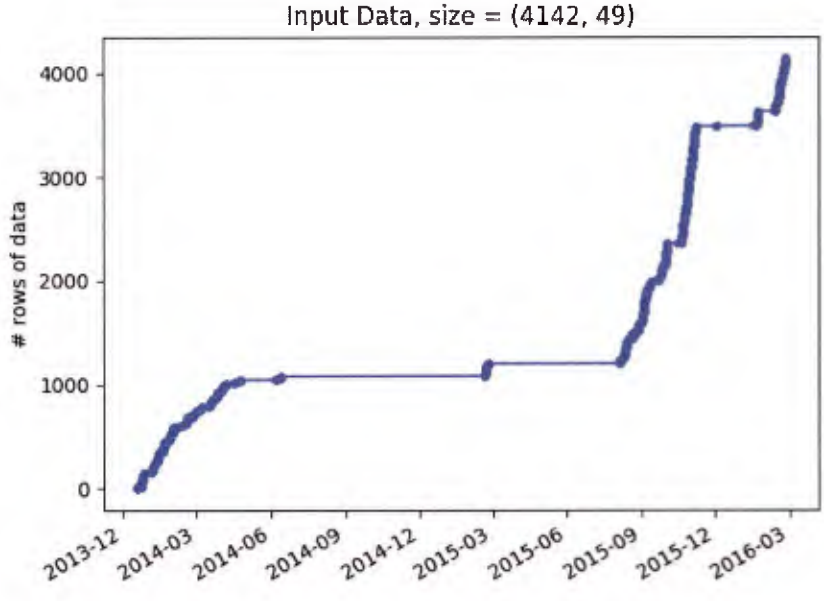


Figure 72: Hull = "B", Engine = 3, Event Type = 2, Event Date = 2016-02-26.

The data was prepared using min-max scaling. Model I used exponential linear unit activations with 20% dropout at each layer. The model utilized an 'adam' optimizer, a variation of first order stochastic gradient decent, which works well with large amounts of data. All signals from the supplied data were used in the model.

Figure 73 shows the model topology for all runs (A, B, C, D, and E) of Model I in a summary table printed by Keras. Layer types are shown in the first column, followed by output shapes and the number of parameters to learn during training for each layer. The input size was equal to 49, the number of signals in the data. The first layer output increased this to 64, then reduced it down to eight at the center of the model. Data was fed into the model in batches of size 32, for 50 rounds (epochs) of training. The layer sizes (including the input layer) were 49, 64, 32, 8, 32, 64, and 49. The Keras summary table adds rows to display dropouts, but in training, these are applied to the dense layers above them in the table. In practice, dropouts are not considered layers. This model had seven layers including input, and had to learn 11,129 parameters.

Layer (type)	Output Shape	Param #
dense_86 (Dense)	(None, 64)	3200
dropout_70 (Dropout)	(None, 64)	0
dense_87 (Dense)	(None, 32)	2080
dropout_71 (Dropout)	(None, 32)	0
dense_88 (Dense)	(None, 8)	264
dropout_72 (Dropout)	(None, 8)	0
dense_89 (Dense)	(None, 32)	288
dropout_73 (Dropout)	(None, 32)	0
dense_90 (Dense)	(None, 64)	2112
dropout_74 (Dropout)	(None, 64)	0
dense_91 (Dense)	(None, 49)	3185

Total params: 11,129
Trainable params: 11,129
Non-trainable params: 0

Figure 73: Keras summary of Model I topology.

The training loss was determined using MSE. MAE was considered as well, but was not used to choose when to stop training. This is shown in Figure 74. Training was stopped after 50 epochs, because after this point, the MSE no longer decreased significantly. MAE was used to visualize training error across the input signals and is shown in Figure 75.

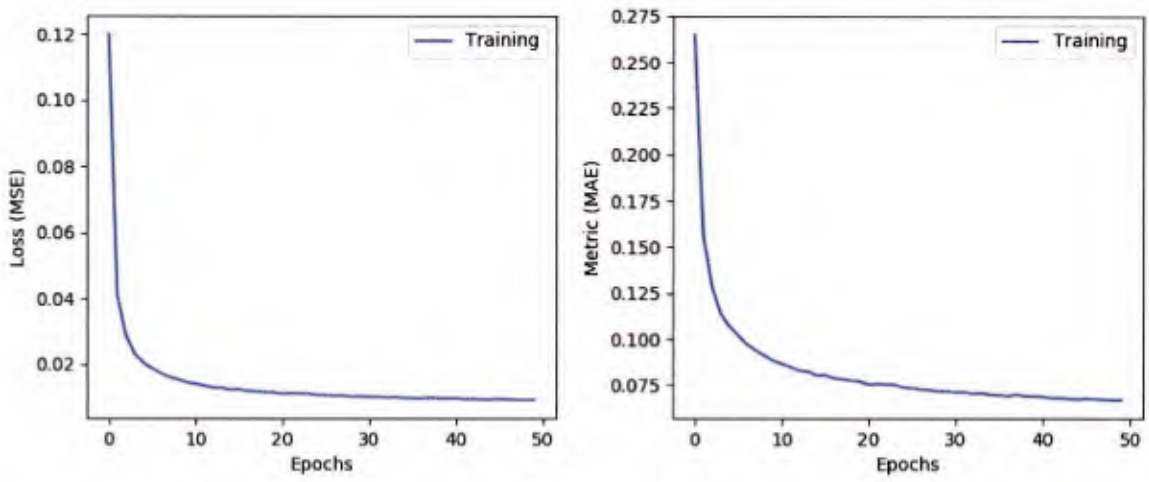


Figure 74: Training loss via MSE and MAE.

The model error was low enough (just above 5% on average) to consider condition indicators from the inner layer's encodings.

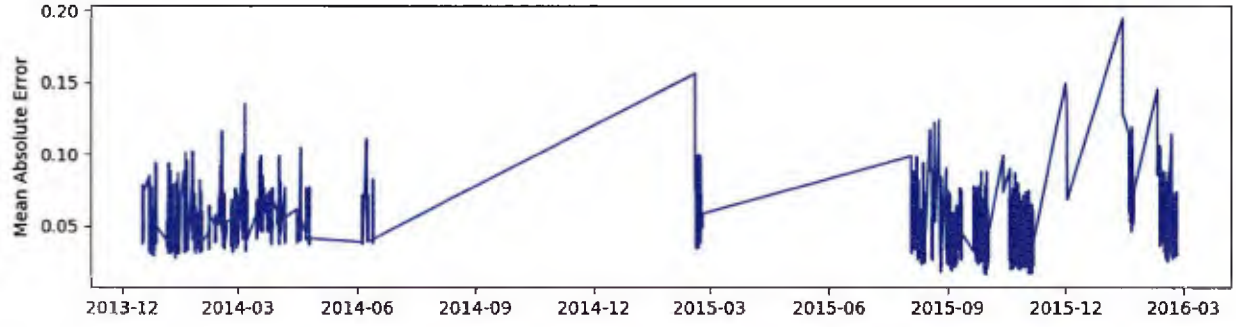


Figure 75: MAE of model evaluation on training data.

Highest Encoded Layer's Outputs

By running the code shown in Figure 76, the model was truncated to output at the smallest layer, rather than pushing the data all the way through to the last layer.

```
encoding_layer_model = Model(inputs=auto_enc.input,
                             outputs=auto_enc.layers[4].output)

encoding_output = encoding_layer_model.predict(scaled_input_data)
print(encoding_output.shape)
```

```
(4142, 8)
```

Figure 76: Python code to extract inner layer encodings from an auto-encoder.

Notice the output shape (4142, 8) has eight columns, instead of 49 like the input. These eight columns are compressed representations of the input data. The variable *encoding output* allows us to look at these outputs to see if the CI hypothesis is valid. The eight encodings are displayed in Figure 77. A good condition indicator should vary in proportion to the known failure degradation. ϕ_1 appears to be the best indicator, if one exists, because it is the only coding output that appears to change in average value from the start to the end. Judging by-eye, the failure appears to begin at the point indicated by orange box in the figure. The data before that region is mostly above zero, and the data after is mostly below. The y-axis values are results of a series of non-linear transformation and activation on the input data and should not be interpreted as anything more. Of most interest is the nature of how these representations drift over time. This raises the question of a metric for assessing the validity of the hypothesis.

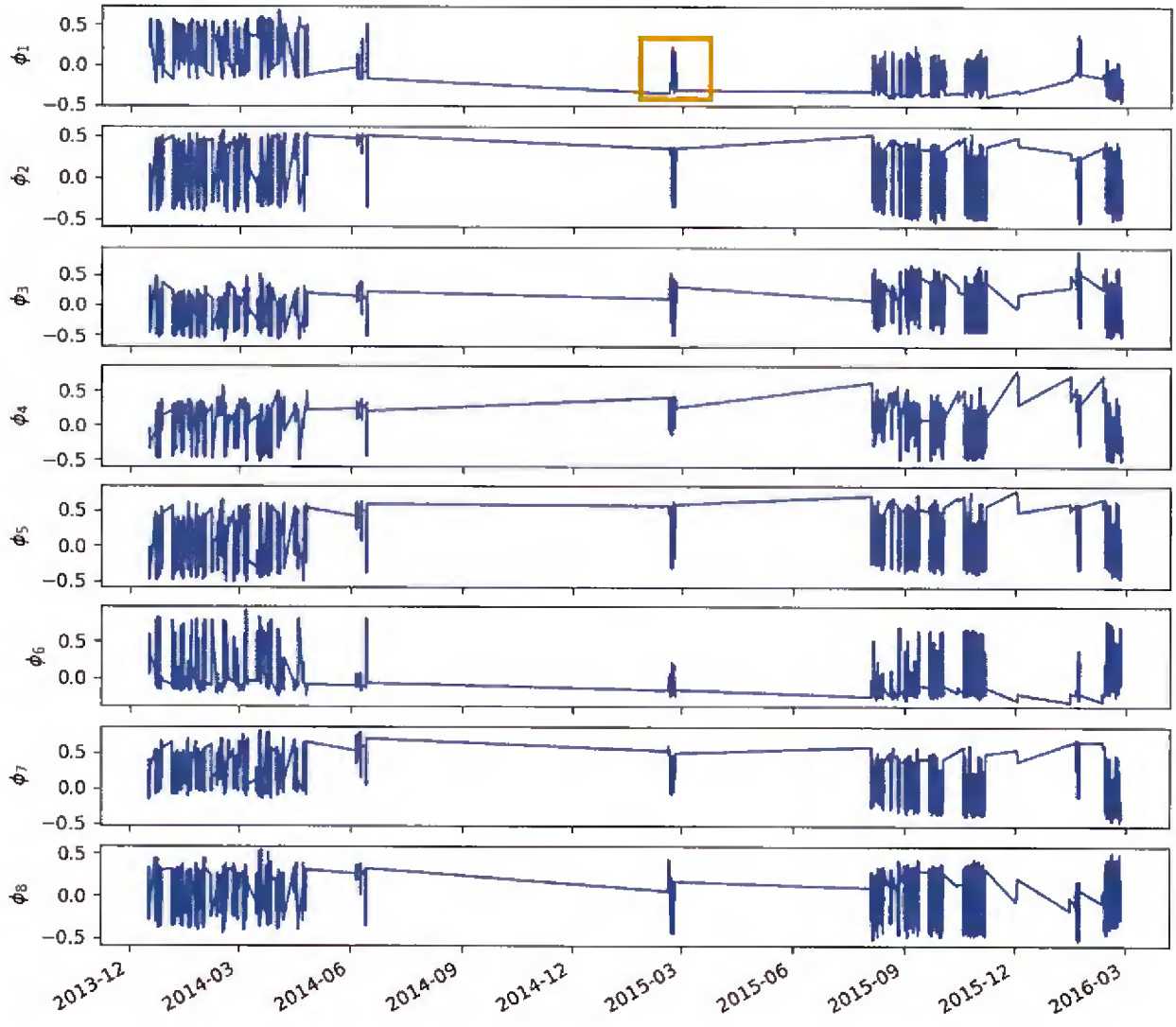


Figure 77: Highest encoded layer of Model I-A output.

5.3.2.3 The Quantitative Metric

The first step was to determine a metric that would accept or reject features for CIs. The metric would need to take into account the mean and standard deviation, and possibly minimum and maximum of the

output over time. This was the first heuristic of this hypothesis. When plotted, the inner layer encoding looked like time-series data, as shown in Figure 77. The chosen metric was a form of *discriminability*, d, which took the difference in group means, μ_1, and μ_2, separated by a selected boundary in the data and divided them by the square root of the average variation of the two sides. The equation for this metric is shown below in Eq. (19).

$$d = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} \qquad (19)$$

By placing the boundary between each data point, d was computed for all possible boundary locations for signal ϕ_1. The highest value of discriminability shows where the two sides of the boundary are most different (i.e. the moment where the data changes enough to be considered incipient failure). Figure 78 shows the results of the boundary search. The orange circles indicate where the discriminability was greatest.

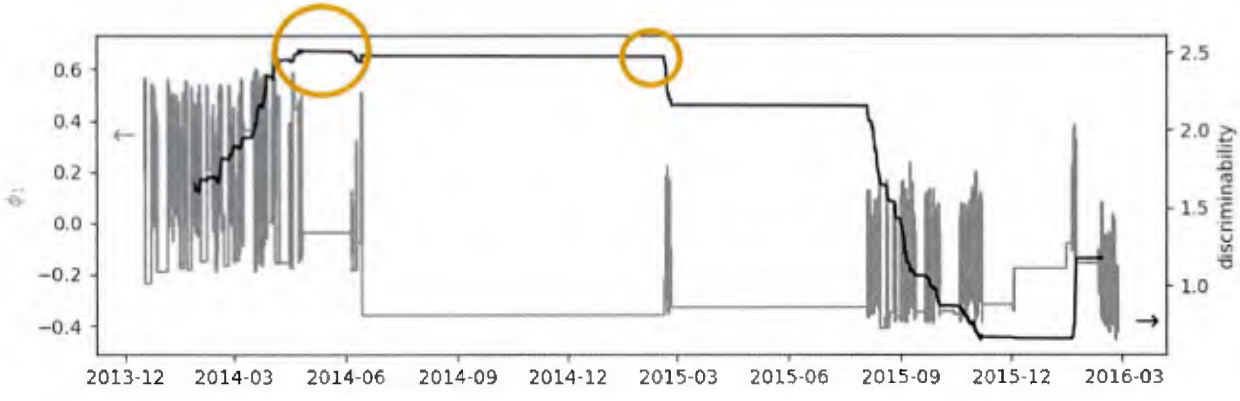


Figure 78: Discriminability across the inner layer encoding of Model I-A.

A margin of data (500 points) was ignored at the ends for this computation due to edge effects. Based on information about the failures, it was assumed that they were not the first or last data values, but rather somewhere in the middle of the data. This margin ignored the first and last 500 points (a small fraction of the data) and therefore was the second heuristic choice, making the two heuristic choices so far: 1) the discriminability metric, and 2) the size of the margin.

ϕ_1 in Figure 77 was annotated with an orange box where the system was thought to begin failing. The same location was computed in the boundary search, indicated by the second orange circle in Figure 78. Judging by the value of ~2.5, the first feature, ϕ_1, is the best feature for a CI. It is reasonable to assume failure started at the first orange circle shown in Figure 78, since there was little to no data between it and the second orange circle.

As a counter-example, Figure 79 shows the worst looking feature as an indicator, of the eight inner encodings. The areas of interest (where the orange circles are in Figure 78) show a much lower discriminability, suggesting there is not a significant inflection along this encoding to adequately indicate the incipient failure.

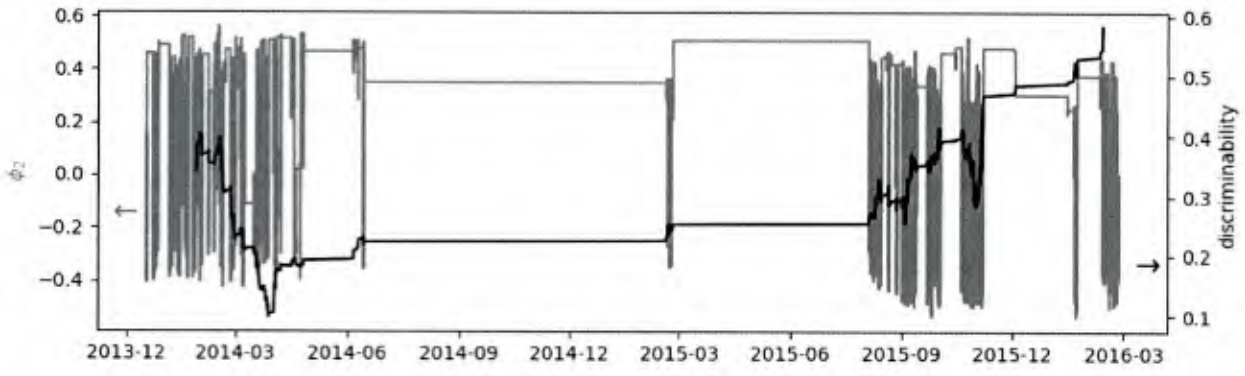


Figure 79: Counter-example to finding good CIs in the auto-encoder's inner layer.

Model I was re-run four more times (B, C, D, and E) to get the five versions mentioned above; features for Model I-B are shown in Figure 80. A few of the features, in particular ϕ_1 and ϕ_4 appear to agree with the hypothesis that condition indicators may be developed from autoencoders that show fault precursors and their progression over time.

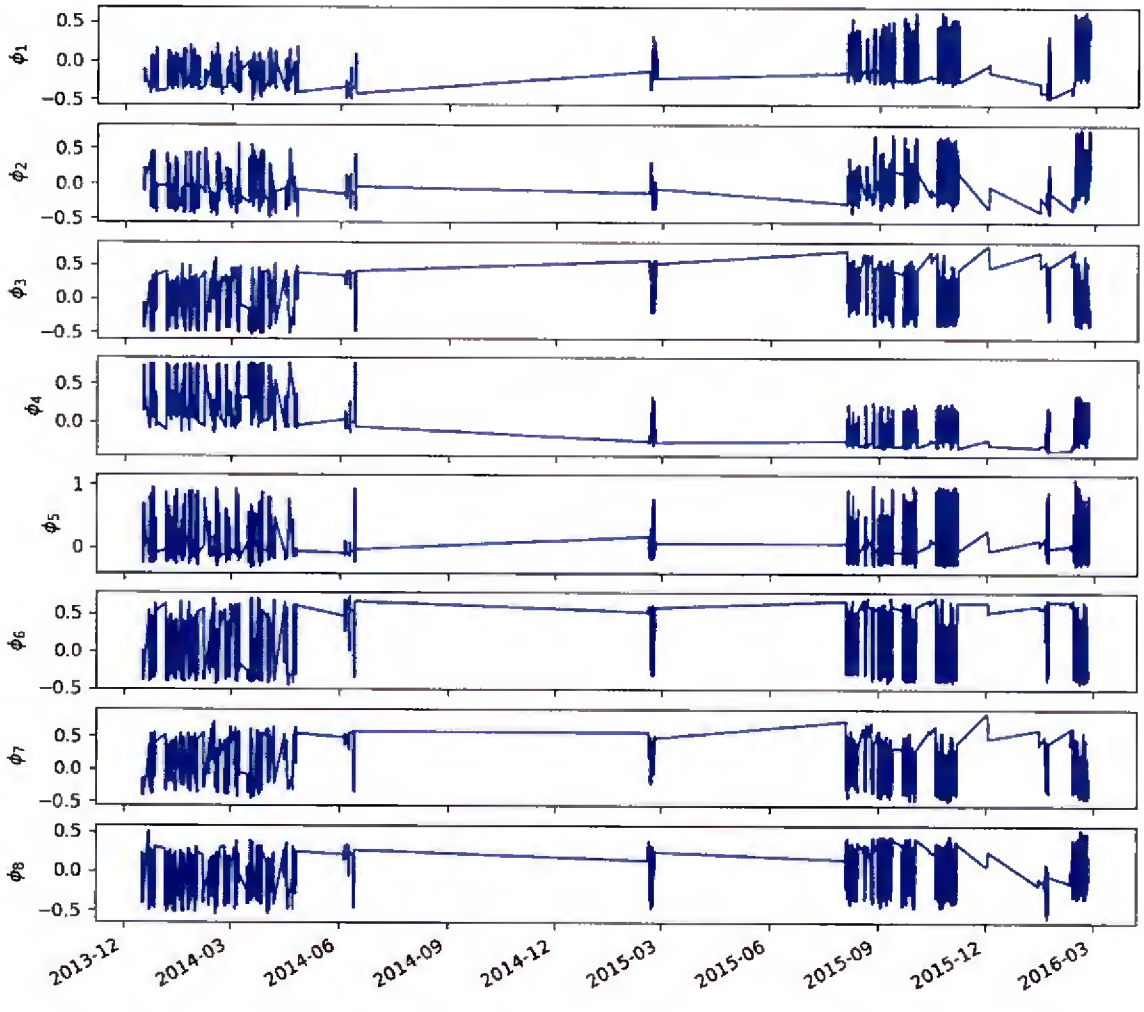


Figure 80: Model I-B inner layer encodings of the input data.

Model II was a deeper version of Model I with two extra hidden layers of size 16. Figure 81 shows the Keras summary of this model. Similar to Model I, min-max scaling was used for preparing the data. The 'adam' optimizer with a learning rate of 5e-4, and 20% layer dropout were used in this model as well. The activation function however, was changed to hyperbolic tangent. It was important to vary some aspect of the auto-encoder, such as the activation functions, to compare with Model I in support of proving the hypothesis. The number of training iterations (epochs) was increased to 100 and data was processed using the same batch size of 32.

Layer (type)	Output Shape	Param #
dense_37 (Dense)	(None, 64)	3200
dropout_31 (Dropout)	(None, 64)	0
dense_38 (Dense)	(None, 32)	2080
dropout_32 (Dropout)	(None, 32)	0
dense_39 (Dense)	(None, 16)	528
dropout_33 (Dropout)	(None, 16)	0
dense_40 (Dense)	(None, 8)	136
dropout_34 (Dropout)	(None, 8)	0
dense_41 (Dense)	(None, 16)	144
dropout_35 (Dropout)	(None, 16)	0
dense_42 (Dense)	(None, 32)	544
dropout_36 (Dropout)	(None, 32)	0
dense_43 (Dense)	(None, 64)	2112
dropout_37 (Dropout)	(None, 64)	0
dense_44 (Dense)	(None, 49)	3185

Total params: 11,929
Trainable params: 11,929
Non-trainable params: 0

Figure 81: Keras summary of Model II topology.

5.3.2.4 Results Summary

The collected CIs from each model are shown in Table 60 and Table 61 below.

Table 60: Model I collected CIs.

Model	Layer Sizes	Best CI	Max d Start -> End
I – A	49-64-32-8-32-64-49	1	6/12/14 – 2/17/15
I – B	49-64-32-8-32-64-49	1	6/13/14 – 2/18/15
I – C	49-64-32-8-32-64-49	8	6/13/14 – 2/17/15
I – D	49-64-32-8-32-64-49	6	2/21/15 – 8/10/15
I – E	49-64-32-8-32-64-49	6	6/13/14 – 2/18/15

Table 61: Model II collected CIs.

Model	Layer Sizes	Best CI	Max d Start -> End
II – A	49-64-32-16-8-16-32-64-49	8	6/13/14 – 2/18/15
II – B	49-64-32-16-8-16-32-64-49	3	6/13/14 – 2/17/15
II – C	49-64-32-16-8-16-32-64-49	4	6/13/14 – 2/17/15
II – D	49-64-32-16-8-16-32-64-49	3	6/13/14 – 2/17/15
II – E	49-64-32-16-8-16-32-64-49	1	6/13/14 – 2/18/15

The CIs are visually compared, both in time and with respect to their relationship, in Figure 82. There were several strongly correlated features, but only one pair (II-E, II-D) was so strong that one of the CIs could be considered redundant.

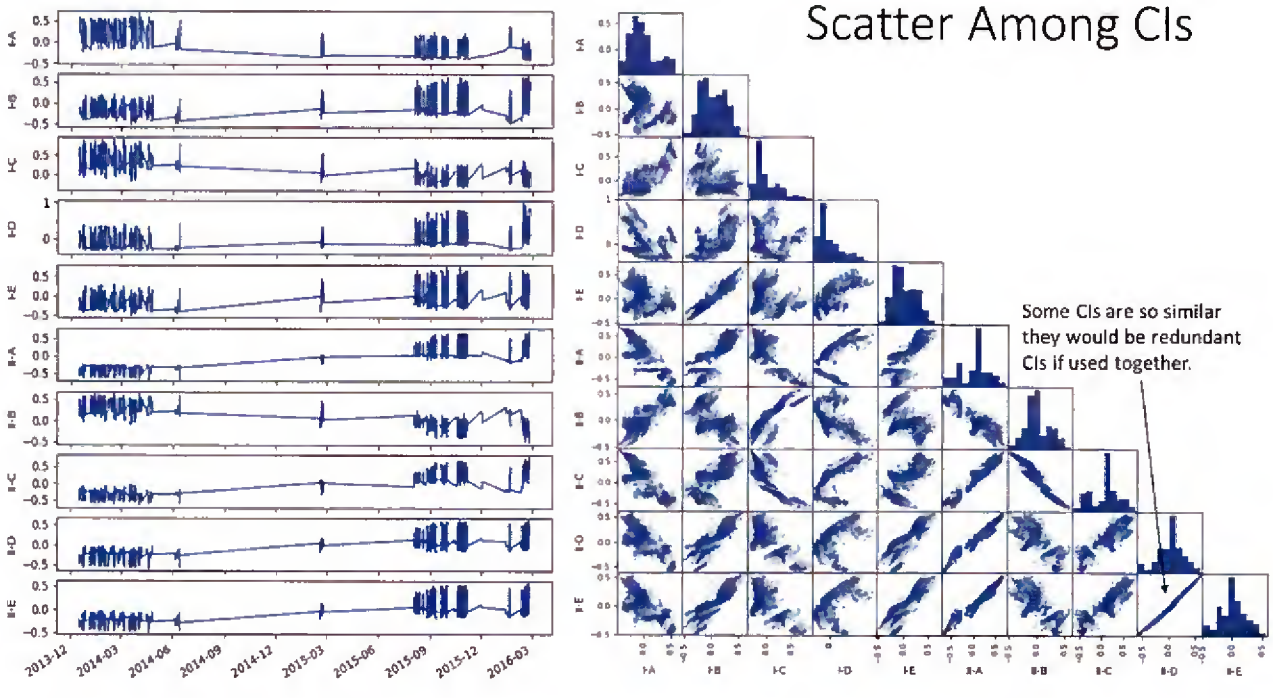


Figure 82: Comparison among CIs

5.3.2.5 Additional Ship Analysis

During evaluation of the data, it was determined that the three separate "Event Type = 2" events, which were specifically associated with Ship=B and Engine=3, were greatly overlapped in time. The identification of this overlapping event data prompted a closer temporal analyses of all of the events. As was mentioned previously, the Navy ship data included a series of failure event types, identified as Event 1, Event 2, and Event 3. As there can be significant differences between different ships (or even between different engines of the same type on the same ship), it was desired to identify a particular engine asset with multiple (preferably three or more) failures of the same type. Among the maintenance events, there were eight unique ship – engine – maintenance type combinations that had at least three associated maintenance events. The remaining combinations had two or fewer maintenance events. Figure 83shows these eight cases (denoted in the y-labels); each event occurs at the end of the marked timeline. Only D-1-2 (ship D, engine 1, event type 2) and F-1-2 had data ranges associated with the maintenance events, which did not overlap other maintenance events. The identification of non-overlapping events is critical to ensure that the data selected as training data for an event does not contain a failure progression for a separate event.

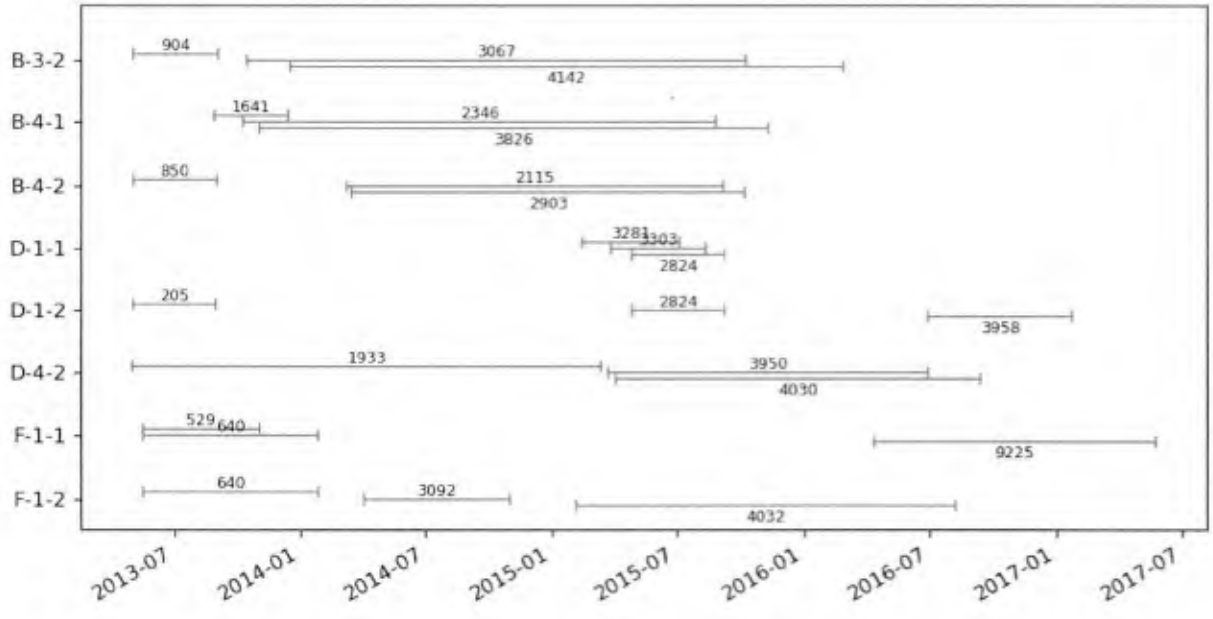


Figure 83: Data ranges associated with 8 maintenance events.

The similar modeling approach was taken on two events on ships with separate data (D-1-2 and F-1-2), but the results were not as promising as the initial results described above.

The two other data sets were chosen, because they were the only ones where each event of data was completely separate in time. Testing on these data sets was necessary to support the hypothesis on different examples. Overlapping data may return partially similar results, and therefore are not good for testing among different events within a ship-engine-event type. It was unclear as to whether the encodings extracted from these two data sets (D-1-2 and F-1-2) were CI candidates. In the first data set tested (B-3-2), there was a clear notion of the encoding values drifting, which was compatible with a degradation. Plausible CI candidates were not observed in these two examples (D-1-2 and F-1-2). This

prompted the effort described in section 5.4, where a battery simulation was created to provide an accessible ground truth of system degradation that could be directly compared to the extracted encodings for CI determination.

5.3.2.6 Summary of CI Estimation from Ship Data

The potential of autoencoders to learn the progression of failure by capturing the degradation within its internal representation layer was examined. Specifically, the representations consisted of the outputs of the encoding layers and were the highest encoded representation of the data within the model. To validate the consistency of these CIs, RIT ran multiple models of the same topology, and multiple models of similar topology, and compared the results both visually and using an objective metric based on discriminability. Given a boundary, which indicated the start of failure growth, the discriminability was defined as the difference in the means of the data distributions before and after the boundary, scaled by a particular form of their variances. High discriminability meant the data drifted away from its average position before the boundary, once failure began, and therefore was an indicator of failure growth. Initial results were promising, because the inner layer's outputs showed this phenomenon, and returned high discriminability between the data before and after the hypothesized incipient failure. However, upon closer examination of the temporal nature of the data, it was observed that many of the failure events were too close together, and therefore the associated HUMS data for the failures overlapped, preventing confidence in the validity of the CIs. While the attempts to apply this approach to few select non-overlapping faults did not yield compelling results, the general approach is believed to have merit; this judgement motivated the study in section 6.5.

5.4 Further Study of 1D CNNs

Although the 1D CNN showed promise in the evaluation of FMTV transmission, the lack of ground truth of the events and the operating conditions limit the validity of the models. As such, an additional study was performed on a simulated battery system, enabling the study of a 1D CNN autoencoder, built to detect anomalous behavior in the system, to be conducted with ground truth knowledge of how the system degraded up to failure. Degradation was set in the form of changes to the battery's cell capacities. A driving profile in the form of current was input into the simulation, where alternating cycles of charging and driving with regenerative braking were present.

5.4.1 Data Preparation

Data from the simulation was stored in an HDF5 file. The raw data contained 36 signals including time, voltages, currents, resistances, and many others. Most of the signals provided from the simulation would not be accessible in a real scenario, so the only two signals that were used in modeling were those denoted V_{batt} and I_{batt}, the voltage and current that would be measured on the battery terminal.

The simulation recorded measurements at a non-uniform rate, faster than 1Hz. There were 1,622,314 rows of data, recorded from a simulation runtime of about 400,000 seconds. The CNN computes convolutions between the data and the model's convolutional filters. In 2D convolutions, this amounts to convolutions over fixed-distance pixels, i.e. a pixel is always one array element from its neighboring pixels. In the time domain, this amounts to a constant interstitial spacing, or sampling rate. A resampling function was created to extract integer-second measurements from the data. After resampling the data to 1Hz, it was truncated at 250,000 seconds. This was known to be where approximately 70% of the cell's capacity had degraded, and was labeled "failure". Data was split into training, testing, and anomaly sets, shown in Figure 84. Ten percent of the training data was used for validation to train the autoencoder. A small test

data set was used to confirm the model's performance as seen by the validation metric returned when training finished, as well as for computing decision support metrics, discussed in Section 5.4.3. The anomaly range covered the start of cell degradation up to where it was truncated at 70% degradation, labeled failure. Of the 250,000 seconds of data, the split resulted in 100,000 training points, 25,000 test points, and 125,000 points where the failure grew, labeled anomaly.

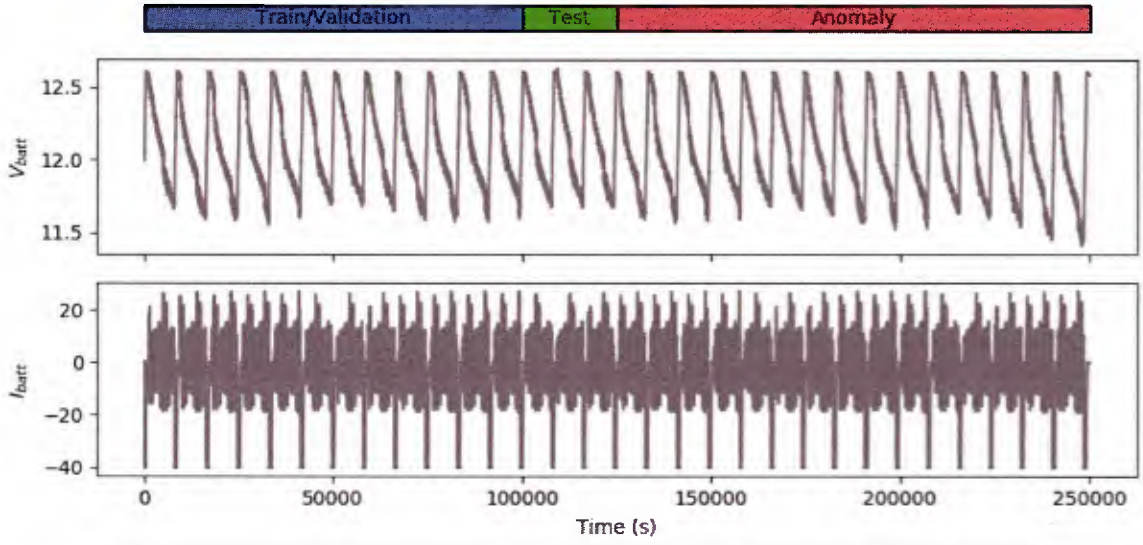


Figure 84: Simulation output for voltage and current signals measured on the battery terminals.

A close up of one charge cycle (negative current) and one driving segment (positive current with negative current at short regenerative breaking moments) is shown in Figure 85.

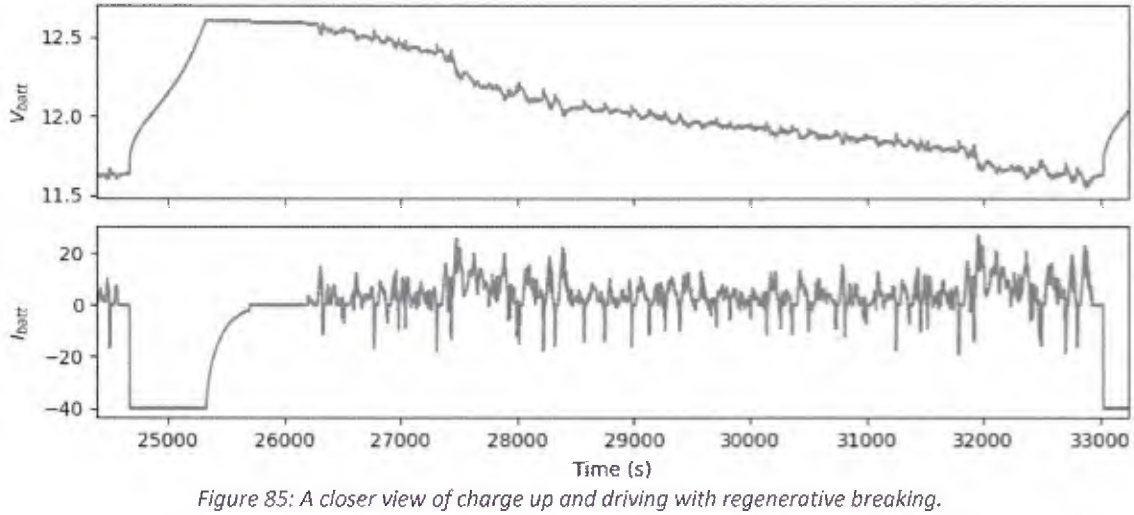


Figure 85: A closer view of charge up and driving with regenerative breaking.

5.4.1.1 Segmenting

The model's input data size was (256, 2), i.e., 256 seconds by 2 signals, voltage and current. This allowed 4-5 minutes of operation for each training sample so that the model could learn dynamical relations in the data. This sample time was chosen based on the fact that a full charge up was about 1-2 hours, and

anything smaller might not capture the system's behavior as well. Any longer and the number of training samples decreased. A segmenting function was created to generate these samples. Data sizes that were not divisible by the segment size were truncated by up to one less that of the segment size. For example, 2D data of size (100000, 2) would be segmented into a 3D array of size (390, 256, 2) with 160 points discarded. Segmented, it becomes 390 samples of (256, 2) data snapshots.

To properly split the training interval into training and validation, it needed to be segmented first, then split, so the randomly selected 10% for validation were whole samples of (256, 2).

5.4.1.2 Scaling

Most machine learning models require rescaling of data, so that one signal does not dominate another simply due to the units selected. In addition, machine learning models avoid large signals, especially when non-linear transformation are used, to avoid saturation. Data was scaled into the [0, 1] range, for each signal. To avoid "data snooping" (see [19] for the discussion), it was important to compute the scaling parameters from the training data set only, excluding validation and test data. If the validation metric was to be interpreted meaningfully, it could not have been included in fitting the scaling parameters.

5.4.2 Modeling

Several 1D CNN model configurations were tested. In summary, there were a two layer with 100 filters model, a two layer with 10 filters model, a two layer with 3 filters model, and a three channel multi-sized filter model. Each is described in turn below, with results, comparisons, and effect on the decision support process, which is described after the first model.

5.4.2.1 Model 1: Two Layers, 100 Filters Each

This model consisted of one layer for encoding with a convolution, max-pooling, and 10% dropout, and one layer for decoding, with a convolution, up-sampling, and 10% dropout. A convolution followed by up-sampling is referred to as a 'de-convolution'. Figure 86 shows the code used to generate this model and Figure 87 shows the model summary, which gives details on layer sizes and number of parameters. In the Keras syntax, the argument "filters" denoted the number of convolution filters to apply to the layer. In convolution, the integral is computed from negative infinity to infinity, and the only non-zero terms are those returned when the functions overlap. This means the filter starts some time before the start of the signal and ends sometime after the last data point in the signal. In Tensorflow, the convolution filter does not do this. Instead, it moves from the start of the signal to the end without going outside the bounds of the signal time. In the Keras syntax, "zero-padding" is used to describe the addition of zeros to the ends of the data to acquire the desired output size of that layer's convolution. Based on the way Tensorflow computes the convolutions, it is essentially functioning more similar to true mathematical convolution to achieve a specific output shape. This is accomplished using the command "padding = same", which can be seen on line 5, 10, and 13 in Figure 86. The model had 653,202 trainable parameters, shown at the bottom of Figure 87. The syntax "kernel_size" denoted the size of each convolution filter.

```
# Shallow-Wide Model Code.
input_data = Input(shape=(segment_size,num_signals))

# Encode.
x = Conv1D(filters=100, kernel_size=64, activation='relu', padding='same')(input_data)
x = MaxPooling1D(pool_size=2)(x)
encoded = Dropout(rate=0.1)(x)

# Decode.
x = Conv1D(filters=100, kernel_size=64, activation='relu', padding='same')(encoded)
x = UpSampling1D(size=2)(x)
x = Dropout(rate=0.1)(x)
decoded = Conv1D(filters=2, kernel_size=1, activation='linear', padding='same')(x)

autoencoder = Model(input_data, decoded)

autoencoder.compile(optimizer = optimizers.Adam(lr=1e-3),
                    loss      = losses.MSE,
                    metrics   = [metrics.MAE,])
```

Figure 86: Model 1 code.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 256, 2)            0
_________________________________________________________________
conv1d (Conv1D)              (None, 256, 100)          12900
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 128, 100)          0
_________________________________________________________________
dropout (Dropout)            (None, 128, 100)          0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 128, 100)          640100
_________________________________________________________________
up_sampling1d (UpSampling1D) (None, 256, 100)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256, 100)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 256, 2)            202
=================================================================
Total params: 653,202
Trainable params: 653,202
Non-trainable params: 0
_________________________________________________________________
```

Figure 87: Model 1 summary.

Figure 88 shows the model loss as mean squared error for all training steps. Due to the dropout layers in training, the model was able to return a smaller loss on the validation data than the training data, because it was using all neurons during validation, and only the active neurons that did not drop out during training. If dropout is removed from the model, the validation loss is in fact greater than the training loss, but over-fitting is unregulated during training.

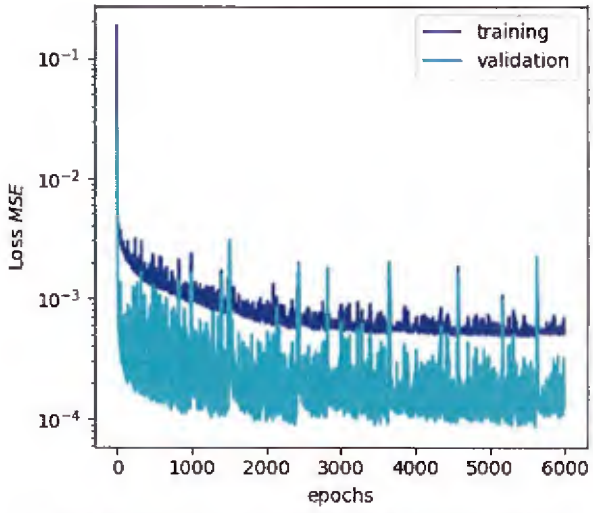


Figure 88: Training and validation mean squared error.

5.4.2.2 Model Metric 1: Average Prediction Error across Signals

The first choice of model metric was to compute the average scaled error of the model's prediction. This was done by taking the absolute difference between input and prediction for both the scaled voltage and current signals, and averaging their values across signals, resulting in a single error value for each time in the data.

Figure 89 shows predictions (orange dotted line) plotted over the two input signals i_{Batt} and v_{Batt}, as well as the average scaled error for the data used in training and testing. Validation segments that were taken at the start can be seen scattered within the training data.

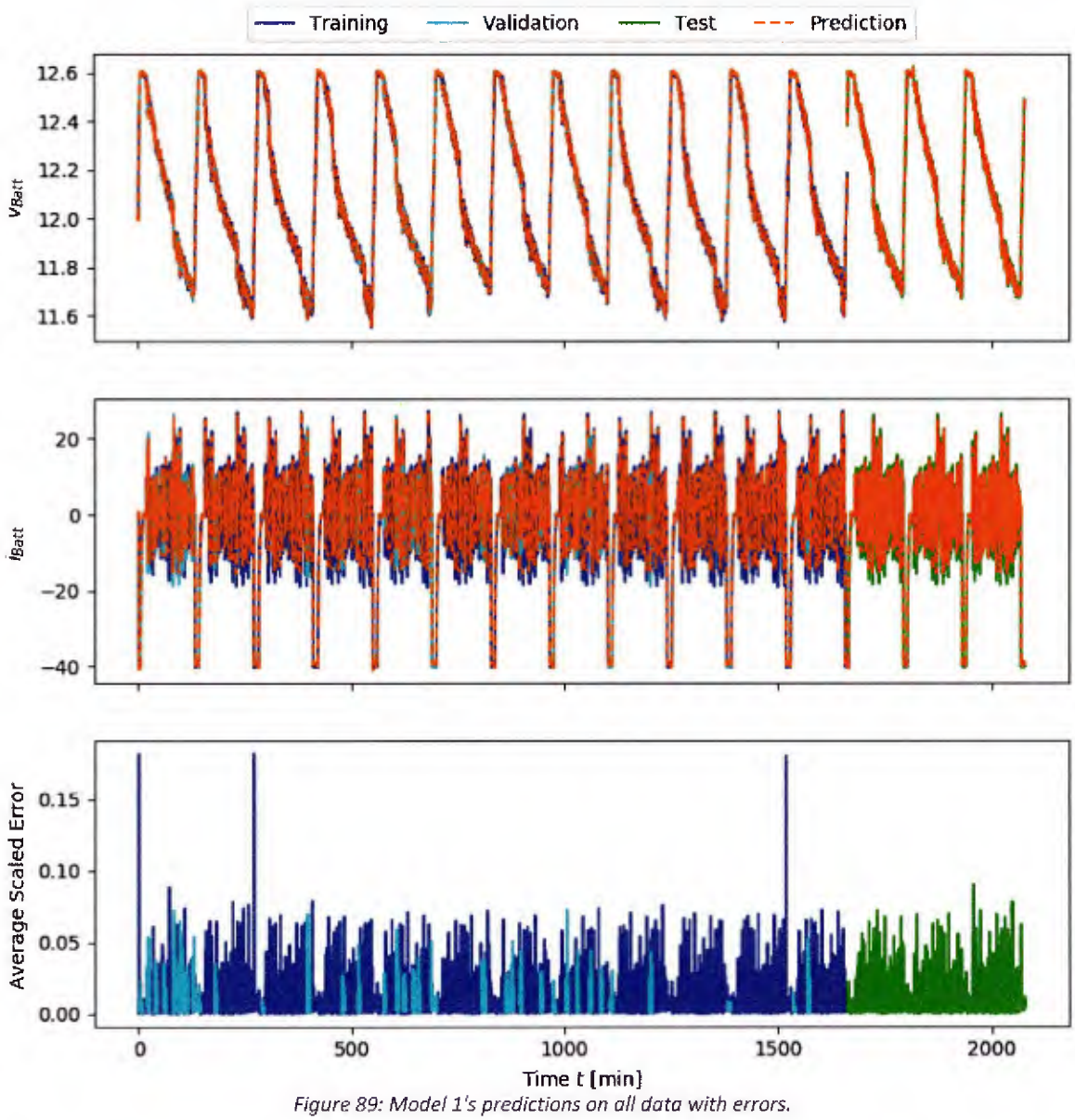


Figure 89: Model 1's predictions on all data with errors.

Figure 90 shows a close up of the test region for both voltage and current.

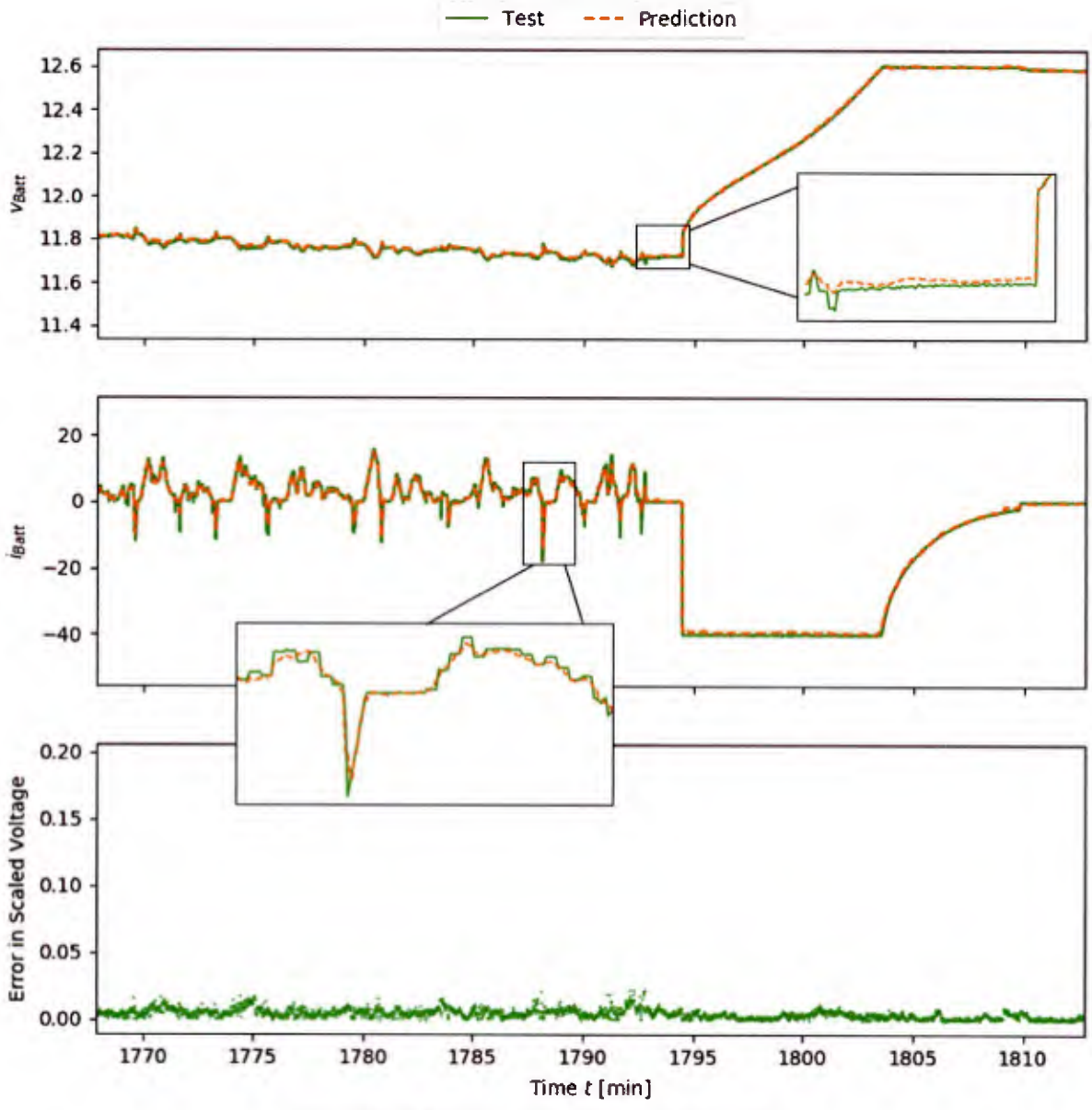


Figure 90: Close-up of model performance in test region.

5.4.2.3 Model Metric 2: Average Filter Size N

The second choice of model metric was the length average window, N, used to filter error. A value of $N = 60$ seconds was chosen. This value controlled the values in the filtered error, which fed into the decision support process, explained in the following section. In practice, the process of choosing model metrics and decision support metrics was iterative because each choice affected each other choice of metric, and the overall goal was to attain a robust desired level of anomaly detection.

5.4.3 Decision Support Process

5.4.3.1 Decision Support Metric 1: Filter + Threshold

The model informed the decision support process via average scaled error of the two output signals. The first choice of metric for the decision support process was to take the filtered model error and compute a threshold above its maximum. The process was defined as follows:

The maximum filtered error value, M, observed in both training and testing was used to determine a threshold, θ, by which any value of the filtered error in new data predictions greater than this threshold were to be considered anomalous. The threshold was determined by increasing M by a factor α, where $\alpha > 1$, thus

$$\theta = \alpha M \tag{20}$$

5.4.3.2 Decision Support Metric 2: A Value for α

The first value to choose for the decision support process was α. The value was chosen to make θ be 30% greater than M. This meant $\alpha = 1.3$. Figure 91 shows how the model metrics, i.e., average scaled error filtered with a 60 second moving average, informed the decision support parameter M so that the threshold for anomaly, θ, was determined.

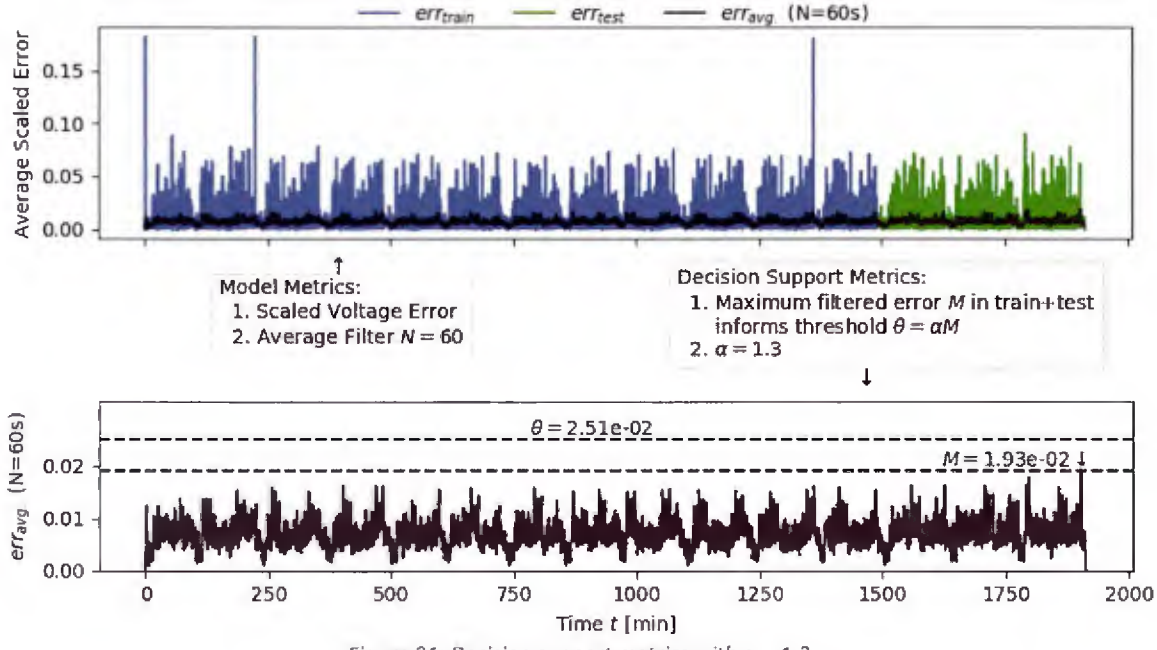


Figure 91: Decision support metrics with α = 1.3.

5.4.3.3 Detection Horizon

The detection horizon was defined as the length of time from the first detection of anomaly to the system failure. In the simulation used for Model 1, it was known that 70% of the batteries' cell capacities were degraded by the end of the data. This was the time in the data labeled as "failure" for this study. The detection horizon of this autoencoder model and decision support process was therefore the amount of time between the first trigger of anomaly and the end of the data. Figure 92 shows the detection horizon for Model 1 with the $N = 60$ and $\alpha = 1.3$.

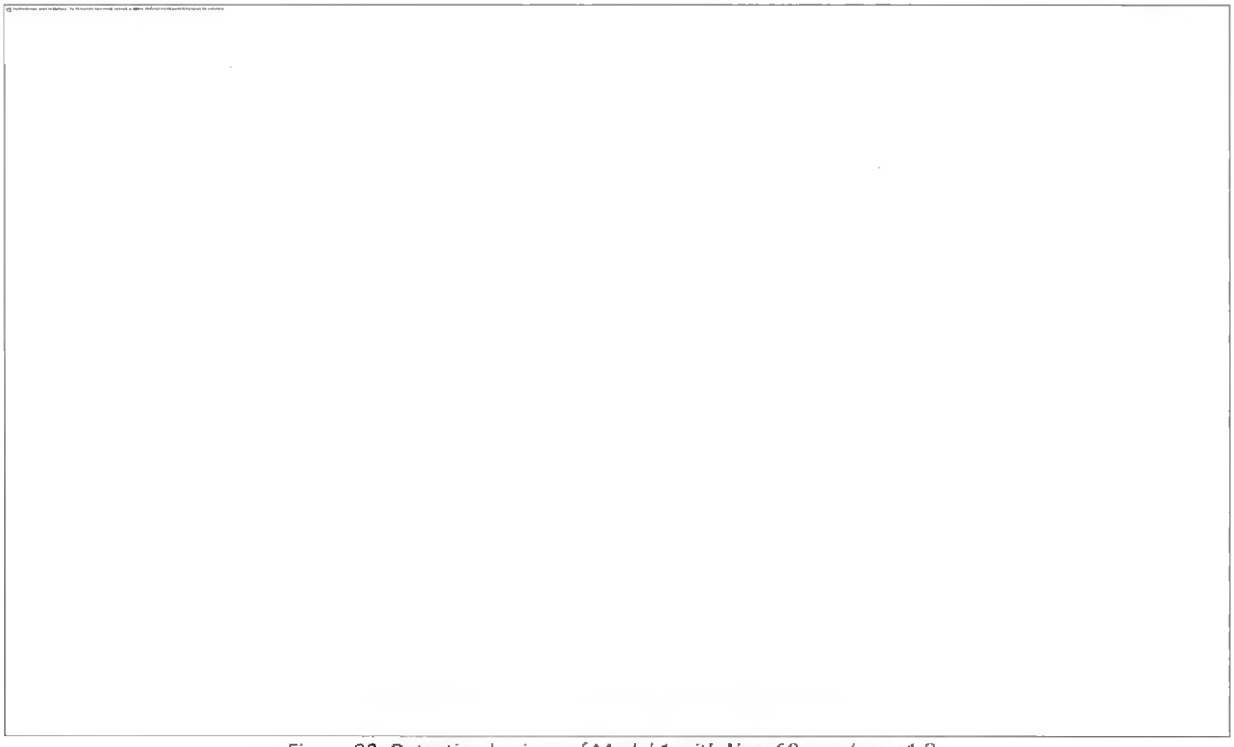


Figure 92: Detection horizon of Model 1 with $N = 60s$ and $\alpha = 1.3$.

5.4.4 Different Models

The two layer model with 100 convolution filters each had 653,202 parameters to learn. A model with significantly less parameters, but the same detection capability would be preferred. For this reason, the following models were explored. The first, a copy of the two layer model but with 10 filters each, simply tested to see if a similar model with significantly less parameters to learn could achieve similar performance. After this, an even smaller model, identical to the first two but with 3 filters in each layer, was tested to see the limit of reducing parameters in this type of model. The last, a model with several layers, three channels in parallel and dealing with different frequencies in the data, was built to test if a more complex model inspired by physics and domain knowledge could outperform all two-layer models while maintaining a low number of parameters.

5.4.4.1 Model 2: Two Layers 10 Filters Each

Compared to Model 1, this model was identical in every way except the number of convolution filters in each layer. The number of filters was reduced from 100 in Model 1 to 10 in this model.

The code for this model was identical to Figure 86 with the replacement of 'filters=100' with 'filters=10' on lines 5 and 9. Figure 93 shows the model summary with layer sizes and number of parameters. This model had 7,722 model parameters, two orders of magnitude less than Model 1.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 256, 2)	0
conv1d (Conv1D)	(None, 256, 10)	1290
max_pooling1d (MaxPooling1D)	(None, 128, 10)	0
conv1d_1 (Conv1D)	(None, 128, 10)	6410
up_sampling1d (UpSampling1D)	(None, 256, 10)	0
conv1d_2 (Conv1D)	(None, 256, 2)	22

Total params: 7,722
Trainable params: 7,722
Non-trainable params: 0

Figure 93: Model 2 summary.

This model returned a detection horizon virtually the same as Model 1. Figure 94 shows this for Model 2, using the same model and decision support metrics. The detection horizon of this model was 854.5 minutes, compared to the 849.9 minutes of Model 1.

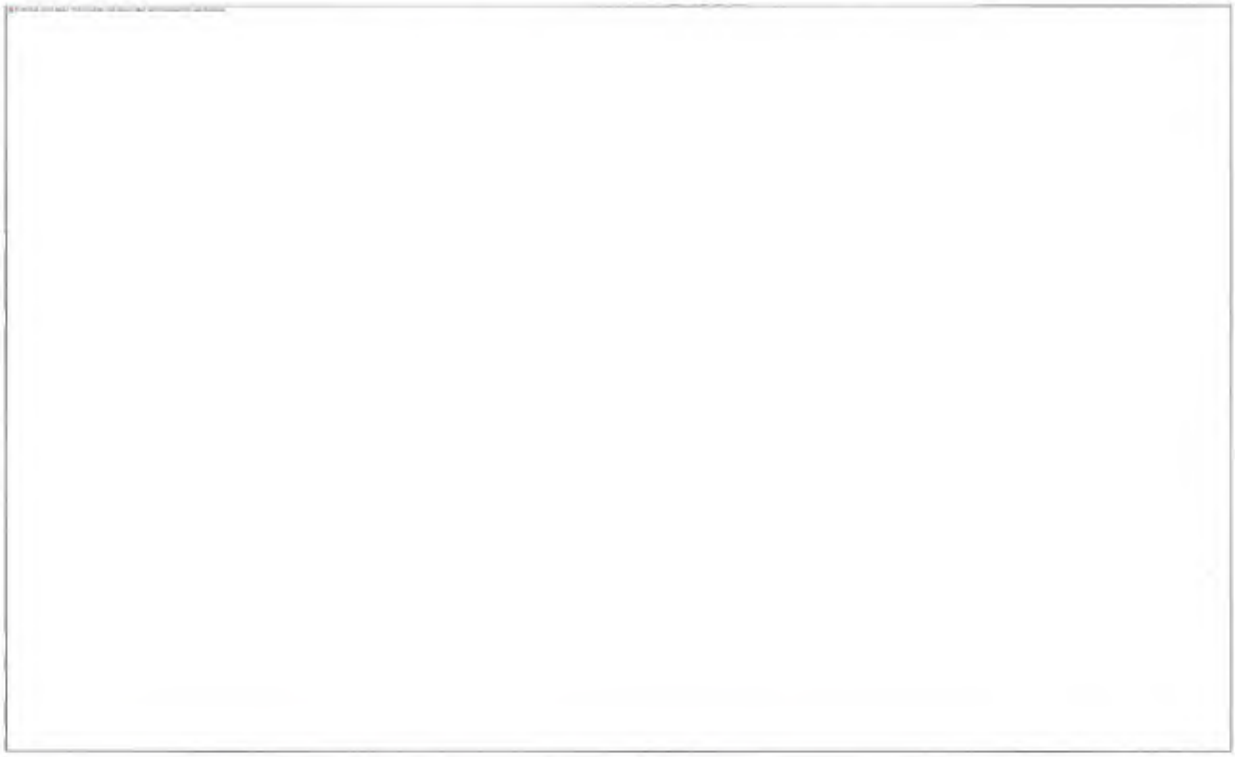


Figure 94: Detection horizon of Model 2 with $N = 60s$ and $\alpha = 1.3$.

5.4.4.2 *Model 2a: Two Layers 3 Filters Each*

To test the limitations of having a significantly lower number of parameters while maintaining the desired level of anomaly detection, Model 2a was created identical to Model 2, but used only 3 filters in each layer. Figure 95 shows the model summary. The model had 974 parameters. The detection horizon of this

model was 167.1 minutes (Figure 96), which was significantly closer to the failure than Model 1 or 2. Fewer filters were tested (1-2 in each layer) but were unable to predict current and voltage accurately, even with many more training steps, indicating that this model was near the limit of the 1D CNN autoencoder's ability to inform anomaly detection at the desired level.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 256, 2)	0
conv1d (Conv1D)	(None, 256, 3)	387
max_pooling1d (MaxPooling1D)	(None, 128, 3)	0
conv1d_1 (Conv1D)	(None, 128, 3)	579
up_sampling1d (UpSampling1D)	(None, 256, 3)	0
conv1d_2 (Conv1D)	(None, 256, 2)	8

Total params: 974
Trainable params: 974
Non-trainable params: 0

Figure 95: Model 2a summary.

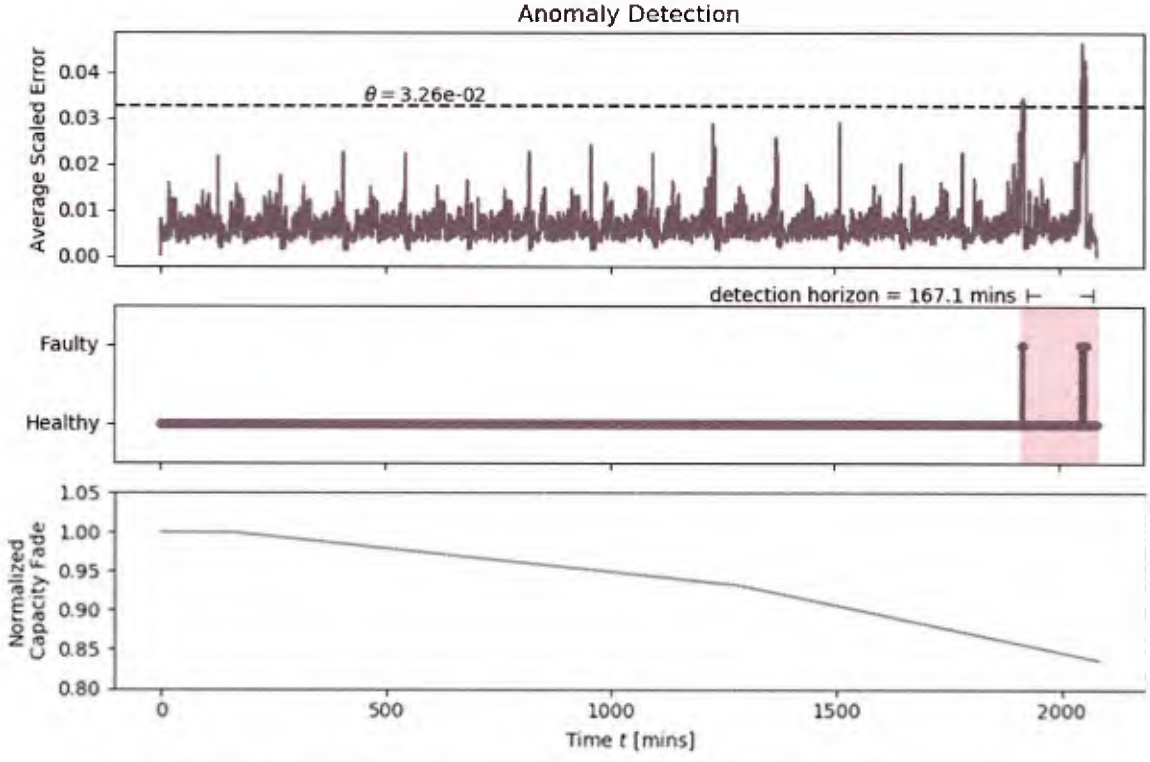


Figure 96: Detection horizon of Model 2a (Model 2 with 3 filters) with N = 60s and α = 1.3.

The general goal of these autoencoders was to allow for anomaly detection with minimal modeling effort and domain knowledge. No domain knowledge was applied to the construction of these models. To

demonstrate potential improvements in anomaly detection by including domain knowledge to the problem, Model 3 was developed.

5.4.4.3 Model 3: Three Channel Multi-Sized Filters Non-Symmetric

This model takes the input along three channels. Each channel consists of a convolution layer with 10 filters, max-pooling of 2, and a 10% dropout. They differ only in what Keras refers to as dilation rates. Dilation rate is the distance between elements in the convolution filter. A default dilation rate of 1 was used by all models up to this point. That meant the convolution filter passed over every point in the data. A dilation rate of 2 means the convolution filter passes over every other point in the data. A dilation rate of 3, every third, a dilation rate of 4, every fourth, and so on. Figure 97 shows the code for Model 3. It used the same number of filters as Model 2 since the goal was to keep the total number of model parameters low. The dilation rate increased for each filter. For example, filter 3 (line 12 in Figure 97) had a filter length of 16 with a dilation rate of 16. For an input of length 256, this filter would cover every 16th point and move through a total of 16 positions for the convolution. This is shown in Figure 98, which shows the model summary, in the size of the output of filter 3 in the row with variable name "conv_1d_2" under column "Output Shape".

```
1  # Model 3: Three Channel Mutli-Sized Filter Non-Symmetric Model Code.
2  input_data = Input(shape=(segment_size,num_signals))
3
4  filter1 = Conv1D(filters=10, kernel_size=16, dilation_rate=1, activation='relu')(input_data)
5  filter1 = MaxPooling1D(pool_size=2)(filter1)
6  filter1 = Dropout(rate=0.1)(filter1)
7
8  filter2 = Conv1D(filters=10, kernel_size=16, dilation_rate=4, activation='relu')(input_data)
9  filter2 = MaxPooling1D(pool_size=2)(filter2)
10 filter2 = Dropout(rate=0.1)(filter2)
11
12 filter3 = Conv1D(filters=10, kernel_size=16, dilation_rate=16, activation='relu')(input_data)
13 filter3 = MaxPooling1D(pool_size=2)(filter3)
14 filter3 = Dropout(rate=0.1)(filter3)
15
16 merged = Concatenate(axis=1)([filter1, filter2, filter3]) # <- encoded.
17
18 upsample = UpSampling1D(size=2)(merged)
19
20 fully_connected = Dense(units=16, activation='linear')(upsample)
21
22 fully_connected2 = Dense(units=4, activation='linear')(fully_connected)
23
24 out_filter = Conv1D(filters=2, kernel_size=61, activation='linear')(fully_connected2)
25
26 autoencoder = Model(input_data, out_filter)
27
28 autoencoder.compile(optimizer = optimizers.Adam(lr=1e-3),
29                     loss      = losses.MSE,
30                     metrics   = [metrics.MAE,])
```

Figure 97: Model 3 code.

Model 3 had several thousand less parameters than Model 2 and only a few hundred more than 2a with a total of 1,724.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 2)	0	
conv1d (Conv1D)	(None, 241, 10)	330	input_1[0][0]
conv1d_1 (Conv1D)	(None, 61, 10)	330	input_1[0][0]
conv1d_2 (Conv1D)	(None, 16, 10)	330	input_1[0][0]
max_pooling1d (MaxPooling1D)	(None, 120, 10)	0	conv1d[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 30, 10)	0	conv1d_1[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 8, 10)	0	conv1d_2[0][0]
dropout (Dropout)	(None, 120, 10)	0	max_pooling1d[0][0]
dropout_1 (Dropout)	(None, 30, 10)	0	max_pooling1d_1[0][0]
dropout_2 (Dropout)	(None, 8, 10)	0	max_pooling1d_2[0][0]
concatenate (Concatenate)	(None, 158, 10)	0	dropout[0][0] dropout_1[0][0] dropout_2[0][0]
up_sampling1d (UpSampling1D)	(None, 316, 10)	0	concatenate[0][0]
dense (Dense)	(None, 316, 16)	176	up_sampling1d[0][0]
dense_1 (Dense)	(None, 316, 4)	68	dense[0][0]
conv1d_3 (Conv1D)	(None, 256, 2)	490	dense_1[0][0]

Total params: 1,724
Trainable params: 1,724
Non-trainable params: 0

Figure 98: Model 3 summary.

Figure 99 shows the detection horizon for Model 3. It is nearly the same as Model 1 and Model 2; however, this model only required ~2,000 parameters. Model 3 has slightly more parameters than model 2a, yet has a significantly longer detection horizon (846.1 minutes vs. 167.1) and is therefore a more effective solution to anomaly detection.

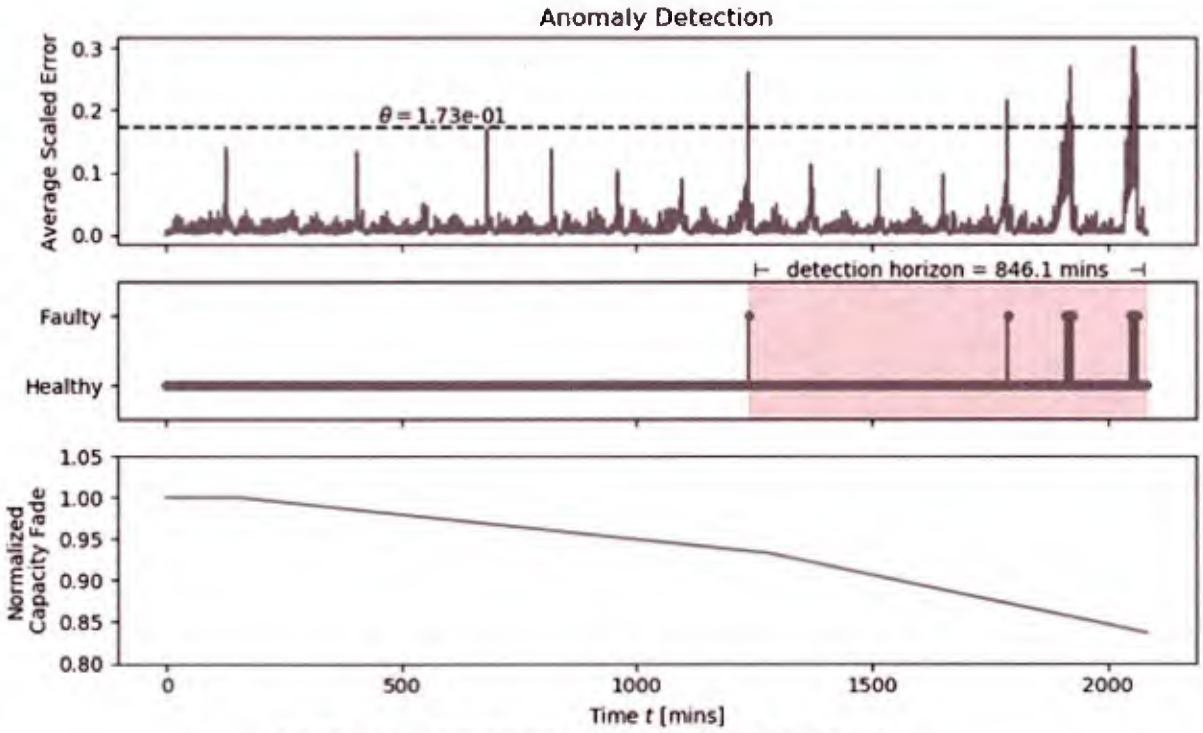


Figure 99: Detection horizon of Model 3 with N=60s and α=1.25.

Table 62: Comparing model's by architecture, model and decision support metrics, and detection horizon.

Model #	Description	Training Steps	Number of Parameters	Decision Support Parameters	Detection Horizon (minutes)
1	2 layers, 100 filters each	6,000	653,202	N = 60, α = 1.3 θ = 0.0251	849.9
2	2 layers, 10 filters each	6,000	7,722	N = 60, α = 1.3 θ = 0.0329	854.5
2a	2 layers, 3 filters each	6,000	974	N = 60, α = 1.3 θ = 0.0326	167.1
3	Three channel, multi-sized filters, non-symmetric	6,000	1,724	N = 60, α = 1.3 θ = 0.173	846.1

5.4.5 Battery Simulation 1D CNN Conclusions and Future Work

Battery simulation was used to provide a system for analysis where the ground truth of the system degradation was known. The simulation was built upon a published Modelica battery module, which was extended to be able to model degradation. For future CBM work, simulation refinements will include an increase in the equivalent series resistor that accompanies the capacity fade. The simulation improvement is expected to further expose the degradation to the terminal (observable) variables (voltage and current), which will in turn help the deep learning model to detect the degradation-in-progress.

Even with these limitations of the simulation (the lack of dependence of the equivalent serial resistance on the degradation), a 1D CNN autoencoder was capable of detecting the anomaly. This was achieved by

training the model on many samples of operation with healthy ground truth knowledge and feeding the model's prediction error into a decision support process to indicate when the operation is no longer behaving as expected. With the current simulation limitations, the deep learning investigation was restricted to anomaly detection based upon the prediction error. Future research, based upon higher-fidelity simulations, will further explore representation learning for extracting CIs. The CIs will be extracted from the model error as well as from the encodings, as originally intended with the battery simulation model. The appropriate CIs in conjunction with the degradation models may be utilized to develop a prognostic model.

5.5 Exploratory Analysis of System Level Models of the Navy Ships

The analysis of system level models of Navy ships started with a high-level domain knowledge model and then proceeded to develop a detector for irregular behavior of the engine system. The first task was to visually explore data on various sub-systems to understand which signals best characterized system behaviors. Overall, the data sets were smaller, which guided the decision to engineer features using expert knowledge and employ classical machine learning techniques.

Classical machine learning models, supported by handcrafted features, can be used to build effective models. The workflow of this approach is described below. Figure 100 shows a high-level view of the signal modeling process. This was the pipeline followed in modeling these engine signals.

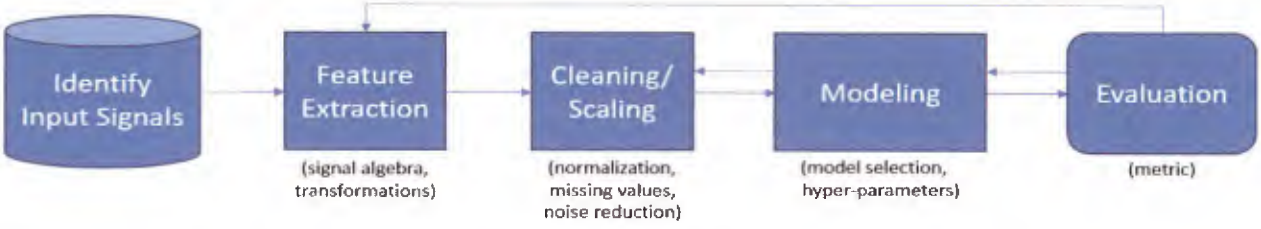


Figure 100: High-level pipeline for modeling signal measurements.

In the first stage, a subset of signals are selected for input and for output such that the input signals expose as much information as possible to the model for prediction or classification. This typically requires a combination of data exploration through visualization and domain knowledge of the subsystems. The next step is to consider feature extraction. Deriving features from signals using mathematical transformations or signal combinations (e.g. fuel consumption = fuel inlet – fuel outlet) should also be considered. The end goal may be anomaly detection, in which the model's features are what indicate a healthy to unhealthy drift in operation, or classification in which the model's features are specifically used to make decision boundaries for classifying new data. In any case, it is important to develop strong features that are appropriate for the task. For this reason, this stage is often revisited after the model evaluation. Mathematical transformations of these inner layer outputs may help as well. Cleaning and scaling are necessary in any machine learning process, like normalization, dimensionality reduction, noise reductions, numerical encodings of nominal variables, etc. Choice of model, selection of hyper-parameters, and evaluation follow the cleaning step. In Figure 100, they are all connected by arrows pointing in both directions, because these stages are usually done iteratively. After preparing the data for a selected model, it may be realized that a different algorithm fits the problem better. In this case, cleaning and preparation may require changes, and the model parameters would be entirely different.

5.5.1 System Selection and Description

The diesel engine was selected for analysis on the Navy LPD, because it is one of the few subsystems present on both Navy ships and in Army trucks. The official subsystem name is Main Propulsion Diesel Engine (MPDE). Figure 101 shows a diagram of the MPDE and related subsystems that are included in the Navy HUMS data. The naming nomenclature in the Navy system is based upon the dominant subsystem, with the propulsion system being named MPDE.

The MPDE subsystem had four engines, denoted 1A, 2A, 1B, and 2B. The HUMS signal data consists of two signal subsets: PG1 and PG2. All signals within a single subset always had identical timestamps, whereas signals across the subsets may have a small number (~0.1%) of non-matching timestamps. The two subsets were first joined by interpolating the signals with inconsistent sampling.

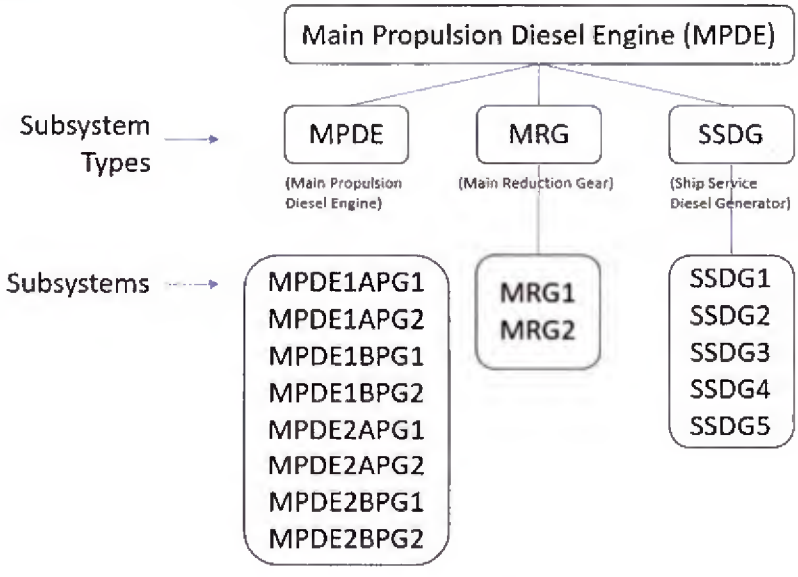


Figure 101: MPDE ship sub-system breakdown.

Figure 102 shows the signals (including both PG1 and PG2 signals) classified in the three categories: 'pressure', 'temperature', and 'other'. Temperature and pressure measurements made up the majority of the data. Signals denoted by 'other' were measurements of delta-pressures, speeds, and levels. This layout of available signals served as a guide to develop an approximate system diagram of the engine subsystem as discussed below.

Pressure Signals

```
'CFW CLR 1A&1B OUT TEMP',
'1A THRUST BRG TEMP',      Thrust Bearing Temp
'1A RKR LO SPLY TEMP',     Rocker Lube Oil Supply Temp
'1A R T/C OIL TEMP',       Right Turbo Charger Oil Temp
'1A R T/C EXHAUST TEMP',   Right Turbo Charger Exhaust Temp
'1A R BANK EXHAUST TEMP', Right Bank Exhaust Temp
'1A R BANK AIR TEMP',      Right Bank Air Temp
'1A LO INLET TEMP',        Lubricant Oil Inlet Temp
'1A LO CLR OUT TEMP',      Lubricant Oil Cooler Out Temp
'1A LO CLR IN TEMP',       Lubricant Oil Cooler In Temp
'1A L T/C OIL TEMP',       Left Turbo Charger Oil Temp
'1A L T/C EXHAUST TEMP',   Left Turbo Charger Exhaust Temp
'1A L BANK EXHAUST TEMP', Left Bank Exhaust Temp
'1A L BANK AIR TEMP',      Left Bank Air Temp
'1A JW OUT TEMP',          Jacket Water Out Temp
'1A JW IN TEMP',           Jacket Water In Temp
'1A JW CLR OUT TEMP',      Jacket Water Cooler Out Temp
'1A CYL 9 EXHAUST TEMP',
'1A CYL 8 EXHAUST TEMP',
'1A CYL 7 EXHAUST TEMP',
'1A CYL 6 EXHAUST TEMP',
'1A CYL 5 EXHAUST TEMP',
'1A CYL 4 EXHAUST TEMP',
'1A CYL 3 EXHAUST TEMP',
'1A CYL 2 EXHAUST TEMP',
'1A CYL 16 EXHAUST TEMP',
'1A CYL 15 EXHAUST TEMP',
'1A CYL 14 EXHAUST TEMP',
'1A CYL 13 EXHAUST TEMP',
'1A CYL 12 EXHAUST TEMP',
'1A CYL 11 EXHAUST TEMP',
'1A CYL 10 EXHAUST TEMP',
'1A CYL 1 EXHAUST TEMP',
'1A CFW RT CAC OUT TEMP', Central Fresh Water Right Charge Air Cooler Out Temp
'1A CFW LFT CAC OUT TEMP', Central Fresh Water Right Charge Air Cooler Out Temp
'1A BRG 9 TEMP',
'1A BRG 8 TEMP',
'1A BRG 7 TEMP',
'1A BRG 6 TEMP',
'1A BRG 5 TEMP',
'1A BRG 4 TEMP',
'1A BRG 3 TEMP',
'1A BRG 2 TEMP',
'1A BRG 1 TEMP',
'1A RKR LO TEMP',
'1A COMB EXH TEMP',        Combined(?) exhaust temp
'1A CFW CLR OUTLET TEMP'   Central Fresh Water Cooler Outlet
```

Temperature Signals

```
'1A/1B CTRL AIR PRES',     Control Air Pressure (not sure what this
would be, for general control applications?  Diesels may have starter
air reqd (at 20-30bar)
'1A RKR LO SPLY PRES',     Rocker Arm Lube Oil Supply Pressure
'1A R T/C AIR OUT PRES',   Right Turbo Air Out Pressure
'1A R BANK AIR PRES',      Right Bank Air Pressure
'1A LOSP 2 DISCH PRES',    Lube Oil Supply 2 Disch Pressure
'1A LOSP 1 DISCH PRES',    Lube Oil Supply 1 Disch Pressure
'1A LO HEADER PRES',       Lube Oil Header Pressure
'1A L T/C AIR OUT PRES',   Left Turbo Air Out Pressure
'1A L BANK AIR PRES',      Left Bank Air Pressure
'1A JW OUTLET PRES',       Jacket Water Outlet Pressure
'1A JW INLET PRESS',       Jacket Water Inlet Pressure
'1A FO MANIFOLD PRES',     FuelOil(?) Manifold Pressure
'1A EXH BACK PRES',        Exhaust Back Pressure
'1A CRKC PRES',            Crankcase Pressure
'1A RKR LO PMP DIS PRES', Rocker Lube Oil Pump Discharge Pressure
'1A FO PMP DISCH PRES',    FuelOil(?) Pump Discharge Pressure
'1A CFW PMP DISCH PRES'    Central Fresh Water Pump Disch Pressure
```

Other Signals

```
'1A RKR LO SUMP LVL',      Rocker Arm Lube Oil Sump Level
'1A RKR LO STR DP',        Rocher Arm Lube Oil Strainer Delta Pressure
'1A R T/C SPEED',          Right Turbo Speed
'1A LO SUMP LVL',          Lube Oil Sump Level
'1A LO CLR OIL DP',        Lube Oil Cooler Oil Delta Pressure
'1A L T/C SPEED',          Left Turbo Speed
'1A FUEL SPLY FLOW',
'1A FUEL RTN FLOW',
'1A FUEL RACK',            Fuel rack is a mechanical device which is
                           adjusted to meter flow at individual injectors
'1A ENGINE SPEED',
'1A AIR INTK FLTR DP',     Air Intake Filter Delta Pressure
'1A LO STRAINER DP',       Lube Oil Strainer Delta Pressure
'1A LO FILTER DP',         Lube Oil Filter Delta Pressure
'1A JW EXPANSION TNK LVL', Jacket Water Expansion Tank Level
'1A FO FILTER DP'          Fuel Oil Filter Delta Pressure
```

Figure 102: Engine signals on MPDE ships.

5.5.2 System-level Diagram

The system diagram is shown below in Figure 103. The blocks in the figure represent subsystem components in the engine. Arrows show which components are connected by mass and energy flows, and the letters S (speed), P (pressure), dp (delta-pressure), and T (temperature) denote the location of HUMS measurements for these components or flows. For instance, oil exits the engine and flows through the following components: lube oil (LO) sump, LO strainer, LO pump 1 and 2, LO cooler, LO filter, LO header, and finally back into the engine and the turbochargers. Oil level is measured on the LO sump, pressure is measured on the LO pumps and on the LO header, and delta-pressure measurements are made on the LO strainer, cooler and filter.

Exhaust temperatures are measured on the right and left engine banks (Lft Bank and Rt Bank in the diagram) and at the exhaust of the turbochargers. Pressure measurements are made where air is directed back into the engine from the turbocharger. Additional signals on the engine are listed in the bottom left of the diagram. The turbocharger process was captured in speed, temperature, and pressure measurements. As an example of signal choice, oil problems could lead to unwanted friction in the turbocharger turbines, and may be observed in measurements of speed, which would in turn affect measurements on the air flow, like pressure and temperature. If an oil problem led to a turbocharger problem then all signals mentioned here, as well as some related to the LO pumps would be desired in building a model to capture the relevant subsystem. There were no direct measurements of air flow mass in the subsystems, which would have provided a turbocharger model with significantly more useful information. Thus, pressures and temperatures were considered in its place.

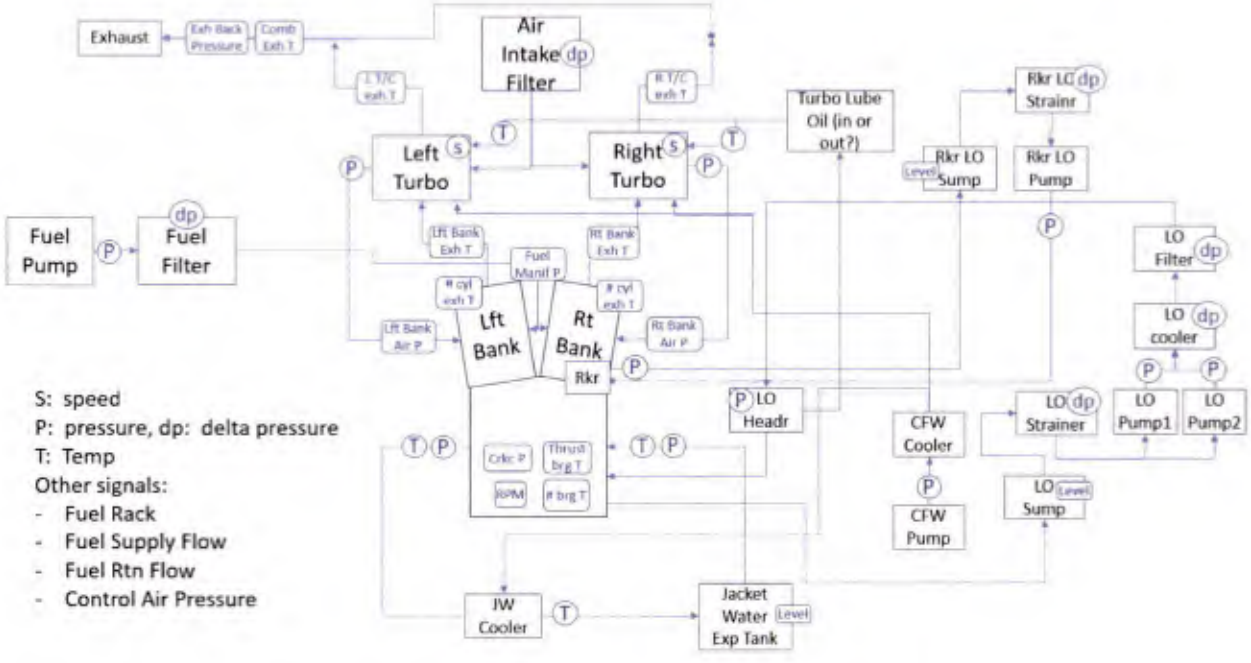


Figure 103: High level system diagram of diesel engine.

The domain knowledge captured in the system diagram and examination of the signals prompted several investigations. The first effort was to review data on various systems to understand which signals best

characterized system behaviors (and might indicate certain failures) and to look for specific anomalies that could be used in additional machine learning exercises. Visualizing the signals is the first step to analyzing the system's behavior. Irregularities in the visualized data directed further investigation. For example, the left and right turbocharger components were observed to operate in two states. Figure 104 shows a scatter plot of the two signals, and Figure 105 shows stacked time series plots for those same signals with engine speed included. When engine speed was below 400 rpms, the right turbocharger remained at near-zero, low speeds (< 1,000 rpms). The right turbocharger never fully stopped as the turbine was always spinning, presumably due to residual air flow. These two states were considered active and non-active, and are better explained through investigation of the time series plot shown in Figure 105.

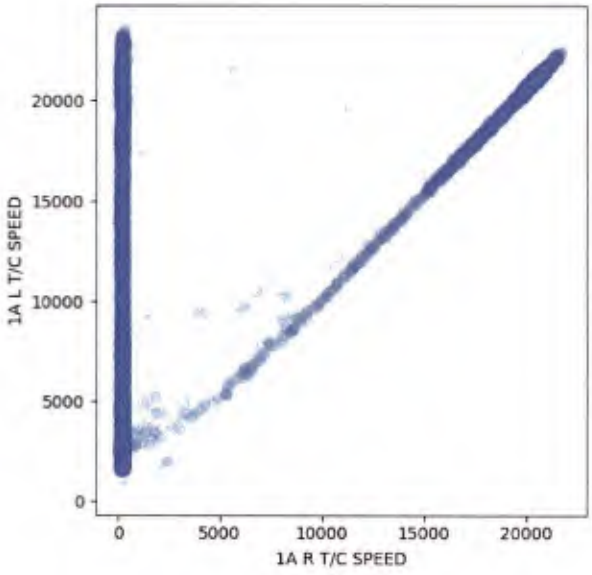


Figure 104: Engine 1A – Ship 0. Scatter plot of left turbocharger speed vs. right turbocharger speed for all available data.

The gray regions of Figure 105 indicate areas of interest in this window of measurements. The thick vertical line of points in Figure 104 represents times where the right turbocharger speed was under 1,000 rpms and corresponds to Figure 105 where the right turbocharger (1A R T/C SPEED) showed very low or no speed at the same time the left turbocharger (1A L T/C SPEED) was active. The highlighted gray areas in Figure 105 show that the right turbocharger was only active when the engine speed (bottom subplot) was above ~400 rpm.

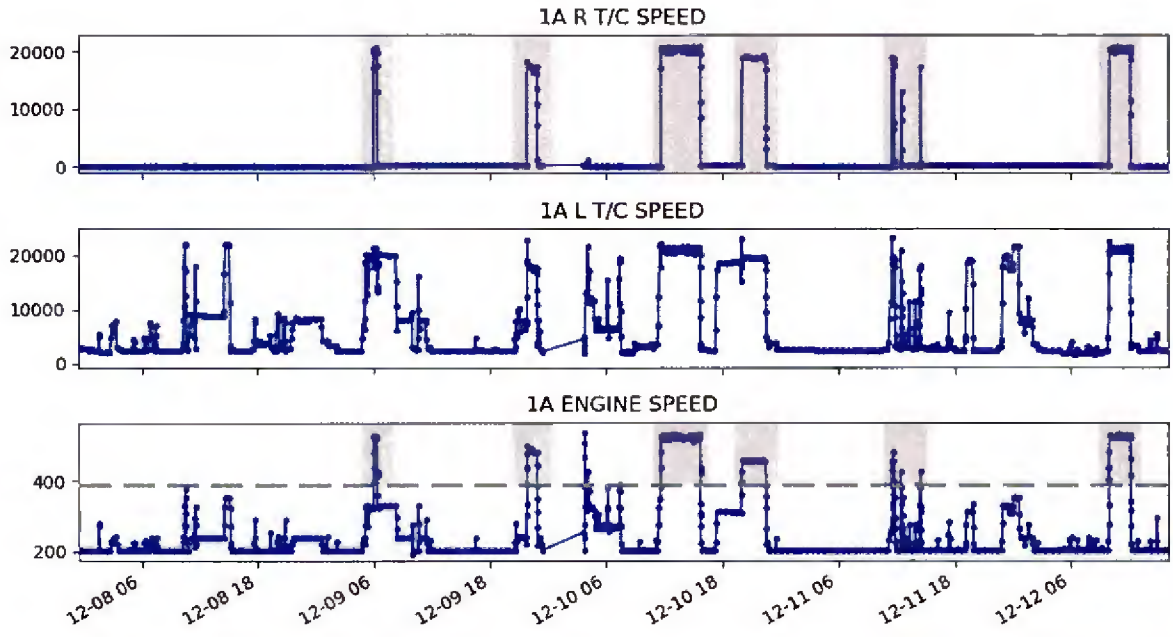
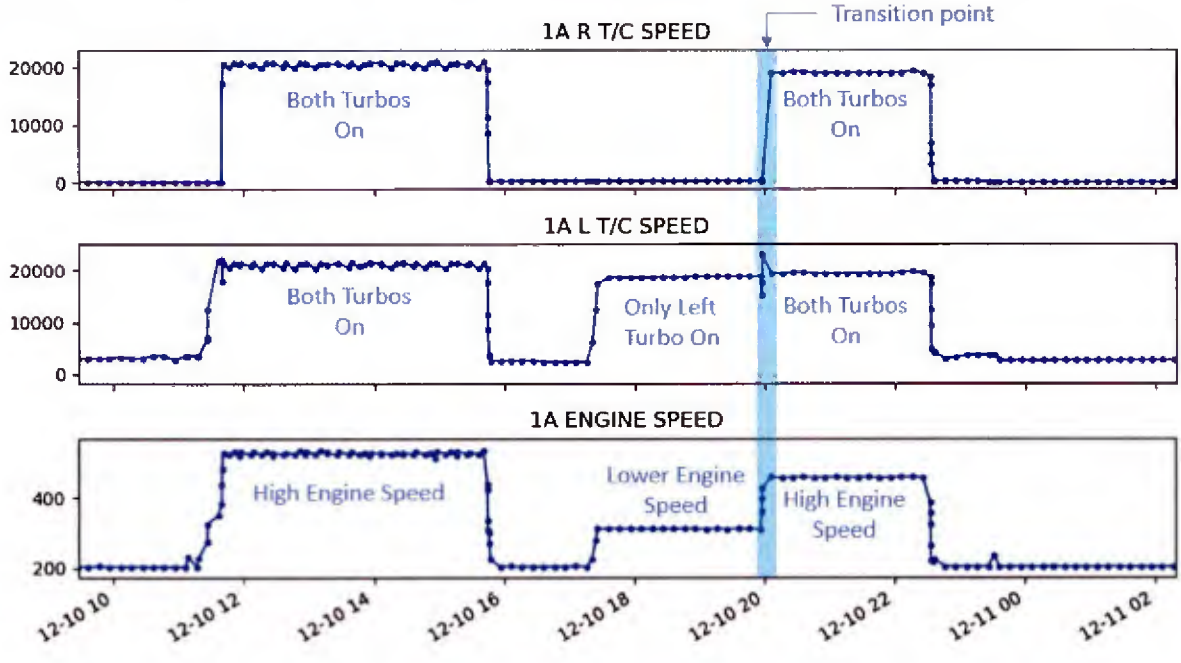


Figure 105: Engine 1A – Ship 0. Time series plot of left and right turbo speed with engine speed for a small window of the available data.

Two gray regions were expanded in Figure 106 for the same three signals. The transient points floating between the dark vertical and diagonal line of points in Figure 104, were those that existed between transitions of the activation of the right turbocharger. In Figure 106, the thin blue shade labeled "Transition point" was an example of this instance. The few points seen there were the points between the two states, right turbo pressure at 0 and right turbo pressure tracking the left turbo, of Figure 104.



only

Figure 106: Zoomed in region of right turbocharger's response to engine speed alongside left turbocharger.

Figure 107 shows that the left turbocharger was always tracking with engine speed, revealing that the right turbocharger only supported the left (main) turbocharger. The exhaust gas from the engine is the only force driving the compressor of the turbocharger, so exhaust temperatures and pressures are expected to be highly correlated with turbocharger speeds. The first subplot shows a daily correlation value between left turbo speed and engine speed. The second and third subplots show the same with larger filters.

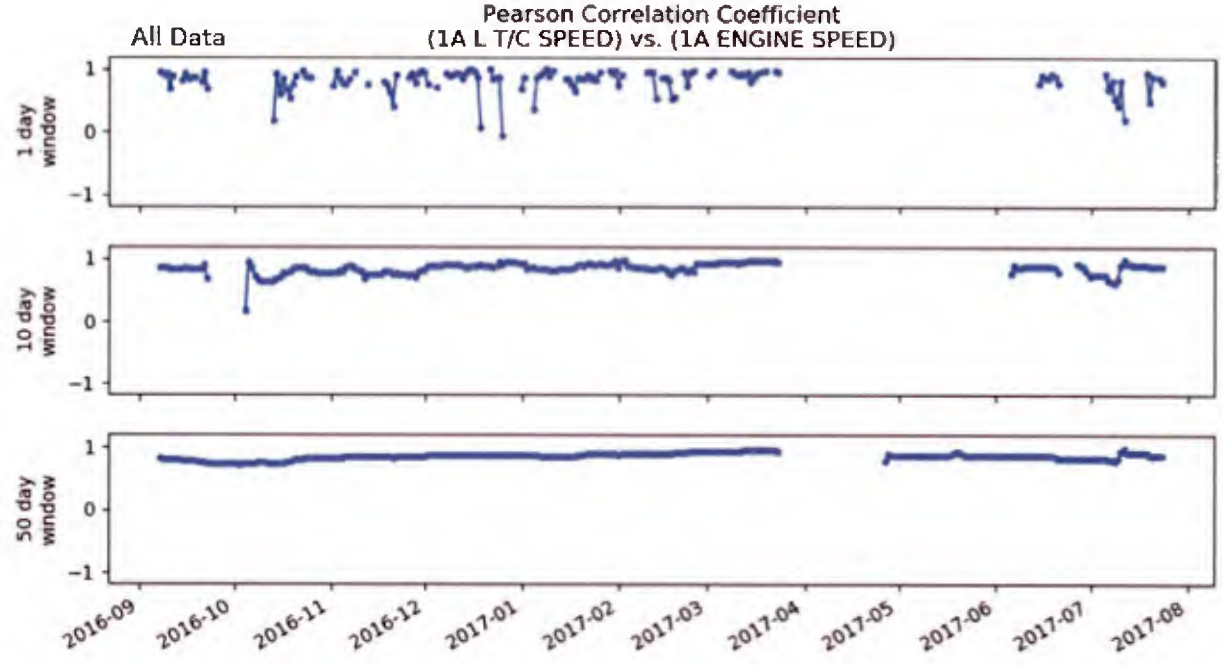


Figure 107: Correlation between engine speed and the left turbo charger, which is always on.

The left turbo charger and engine speed operated together for the entire span of data. The Figure 107 plots only show engine 1A on Ship 0. Figure 108 shows that all engines on both diesel ships behave similar to Ship 0 engine 1A in this regard. MPDE1B on ship 0 contained some large negative data that is obviously errant data. Data segments with obvious errors were not used for subsequent analysis.

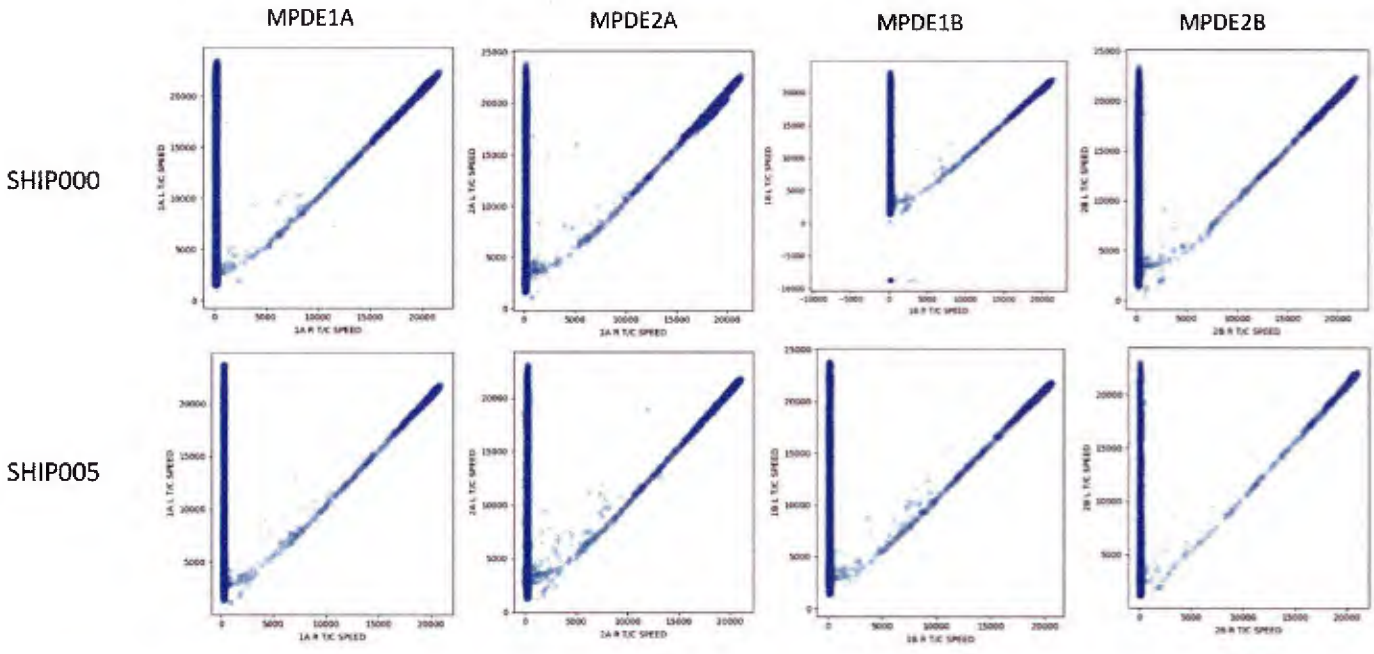


Figure 108: All engines on both diesel ships show right turbocharger spinning only some of the time, and left turbocharger always spinning.

As turbo outlet pressure is a function of turbo speed, it is logical to look at the functional relationship between these signals (pressure vs. speed was scattered for all data on ship 0 – engine MPDE1A in Figure 109). The curve suggests that other variables affect this relationship, or possibly that the turbo may be in different states across the range of sample times.

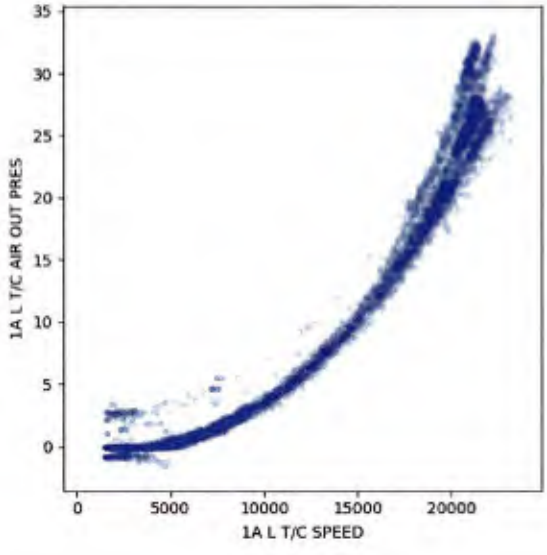


Figure 109: Air out pressure vs. speed for left turbocharger.

Figure 110 and Figure 111 show data for a narrower time range. The yellow and blue dotted lines surround a grouping of points in the scatter plot (Figure 110). The corresponding points to those groups are

surrounded by yellow and blue dotted lines in the time series plot (Figure 111). The blue surrounded points correspond to lower engine speeds, and the yellow surrounded points correspond to high engine speeds. This explains the apparent two-mode relationship between these two signals for the left turbocharger. The higher mode occurs when the right turbocharger is on (second half of the third subplot in Figure 111), at high engine speeds.

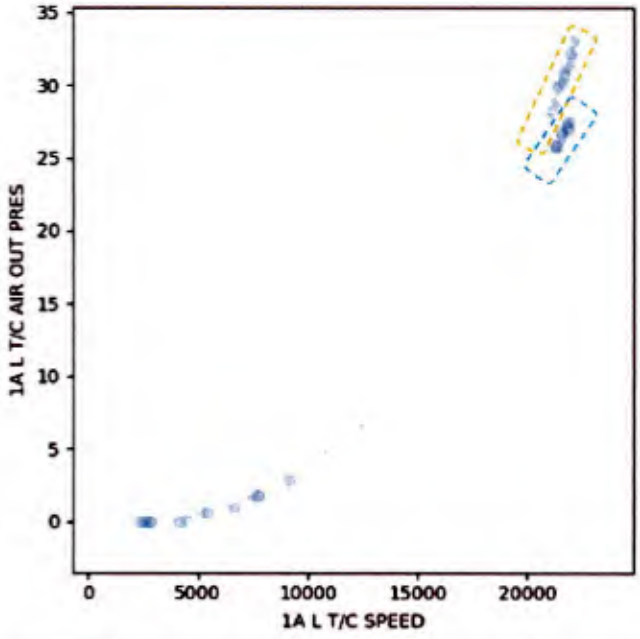


Figure 110: Scatter plot of left turbocharger air out pressure vs. speed.

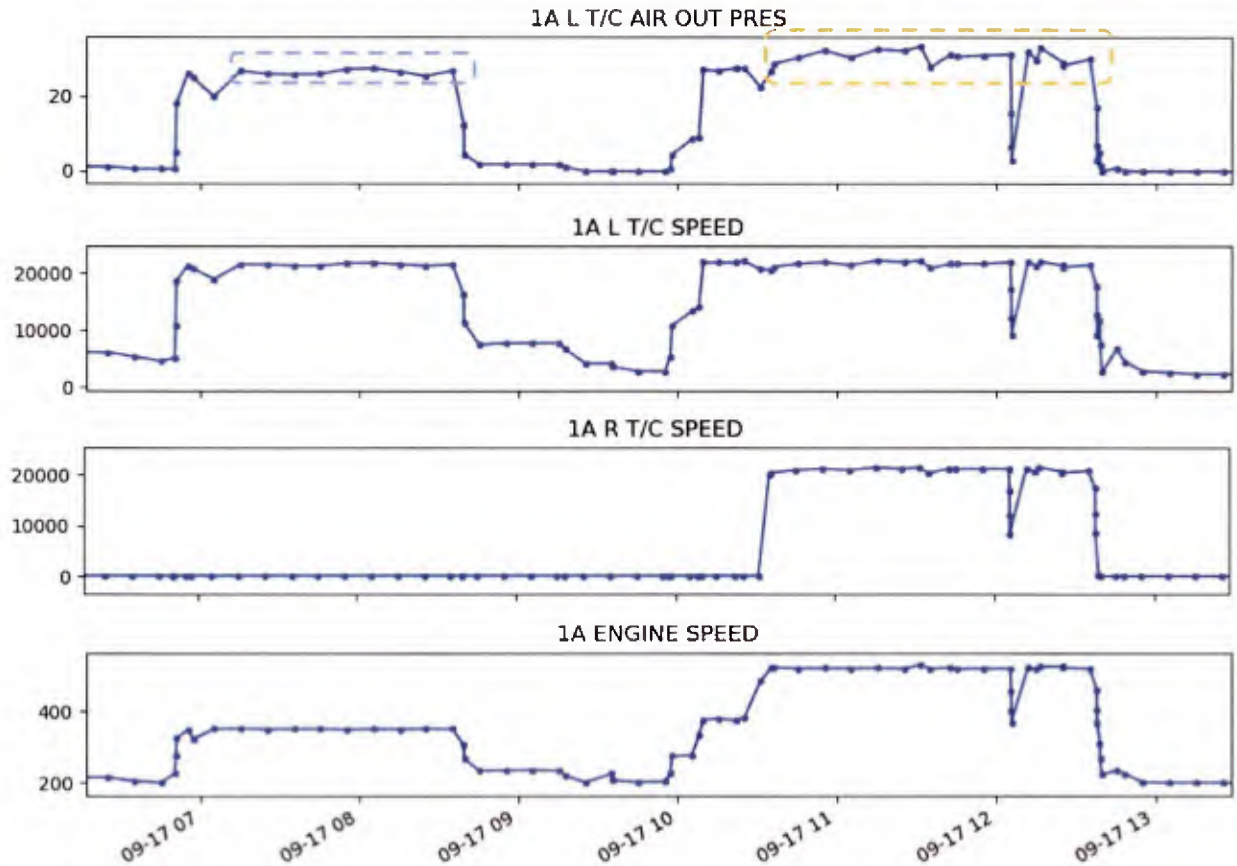


Figure 111: Time series plot of left turbocharger speed and air out pressure, and engine speed.

To better demonstrate this, Figure 112 shows the split in the left turbocharger's output pressure colored by the state of the right turbocharger's speed. It is clear that the exhaust pressure measured on the left turbocharger moves between the two modes depending on the right turbocharger's state. Based on the pressure speed characteristic, it was hypothesized that this relationship could be used to detect turbocharger anomalies. This relationship between the measurements made on the left turbocharger and the state of the right turbocharger would not have been immediately obvious, unless the observer was a domain expert. The above exploration exposed some of the systems properties, but for a general approach, the benefit of classical machine learning methods at this level are to expose these relationships, in this case by learning regression boundaries.

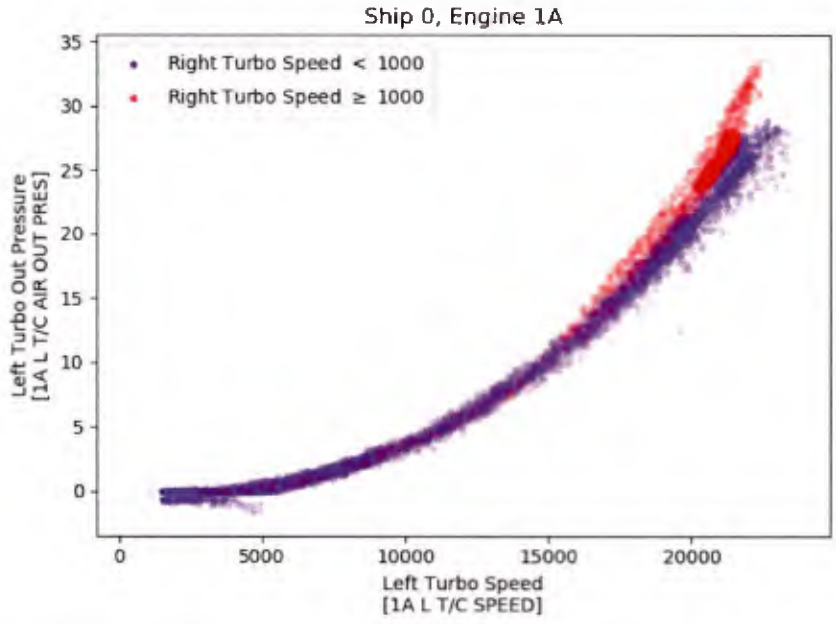


Figure 112: Left turbocharger output pressure colored depending on the right turbocharger's speed.

There are several different types of machine learning algorithms for modeling these types of relationships. One robust method is a Support Vector Machine (SVM). SVMs have some advantages over other methods in that they are efficient on computer memory since only the support vectors, or closest points to the decision boundary, are need in determining the boundary. They are robust because they use kernel spaces to transform the feature space (in this case of signals used to predict another signal) into a higher dimensional linearly separable space. The SVM can be extended from classification to regression for continuous space predictions. These are referred to as Support Vector Regressors (SVRs). Another noteworthy algorithm, which was not used here, is the Random Forest (RF). Using information gain calculations, the model creates a decision tree where the most informative features are closest to the top. RF classifiers combine many differently built decision trees and use a voting schema to determine the most likely prediction value. Its advantages over the SVM is that 1) there is only one parameter to tune, i.e., the number of trees in the forest, 2) a lower sensitivity to this one parameter, 3) it's based on probability functions, and 4) can be extended more easily to a multiple class output (which is not the case here). A shortcoming of the decision tree used in RF are that they tend to over fit the data if not properly monitored and trained. For this reason, the simpler option of SVR, which performed with high fidelity on this data, was chosen. The following model uses an SVR to predict turbocharger air output pressure.

5.5.2.1 Support Vector Regression (SVR) Model

A Support Vector Regression (SVR) model was used to predict turbocharger exhaust air pressure based on input features, turbocharger speed and engine speed. Figure 113 shows where the data was split into training (4 months of densely populated data) and testing (7 months of less densely populated data). The idea was to see if the model could learn the relationship and be able to predict air output pressure later. Furthermore, if the model were to make incorrect predictions, either 1) the model was not sufficiently trained well enough on the signal relations, or 2) something changed in the system and the model was no longer exposed to the data it was trained on. It is paramount in anomaly detection to train on data during normal operations. New data that does not lend itself to accurate model predictions is thus labeled as

abnormal, or anomalous. Due to systems changing over time, it is important to maintain models that have learned normal operation data of the system at the current time. In addition to anomaly detection, there is the general need to use the model trained on one engine for many different engines across the fleet without significant re-tuning to individual assets. For a model to be effective in this way, it must correctly generalize to the overall system behavior.

The model was tested first on the left turbo for engine 1A (MPDE1A) on a specific ship denoted by "0" (see Figure 113). The model was subsequently tested on other engines of the same type, both those installed on the same ship and those installed on other ships.

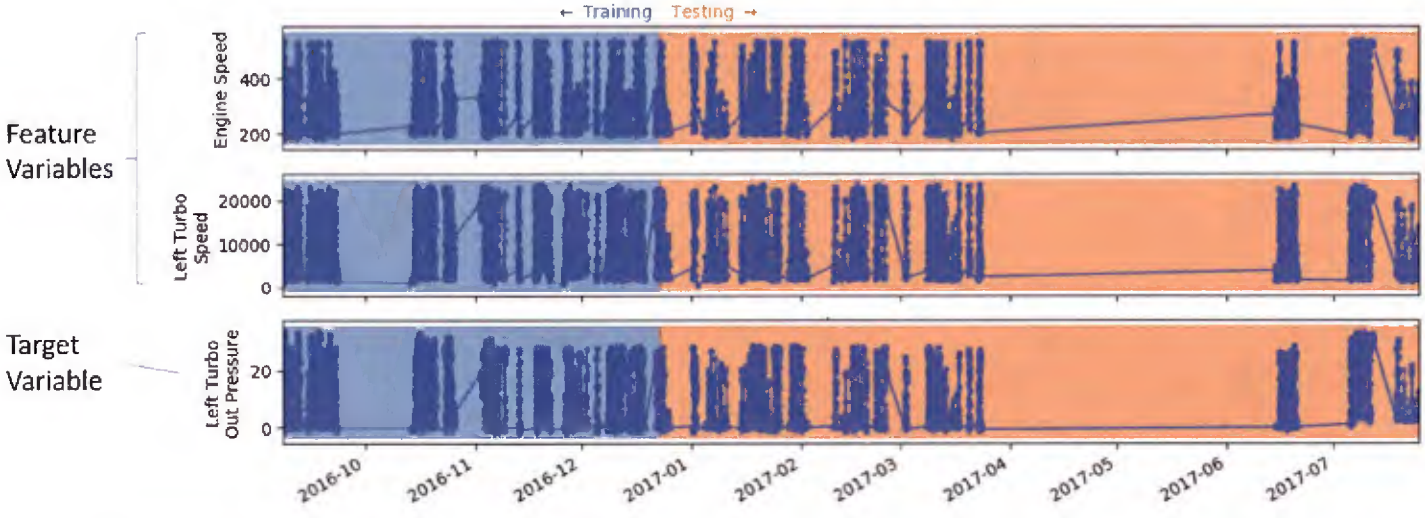


Figure 113: Training and testing split for predicting turbocharger air out pressure from turbocharger and engine speeds on Ship 0 engine 1A.

The potential for re-use of models on the same subsystems in other engines was also explored. Two different interpretations (sensor fault and platform differences, see Section 5.5.2.1.3) of the SVR's predictions on this data were made in light of anomaly detectors and predictions across other engines.

5.5.2.1.1 Testing on the Original Engine

Testing on the original engine revealed the model accuracy was very small (~1 psi average absolute error). Because isolated single-point error (the green trace in the bottom subplot of Figure 114) could be high during transients, performing a moving average over single-point absolute errors (the red trace in the bottom subplot of Figure 114) yielded a more stable indicator. This figure shows the results for the whole span of testing data (orange section in Figure 113). The top subplots of Figure 114 show the test data in blue, along with the predictions in orange, which shows general agreement.

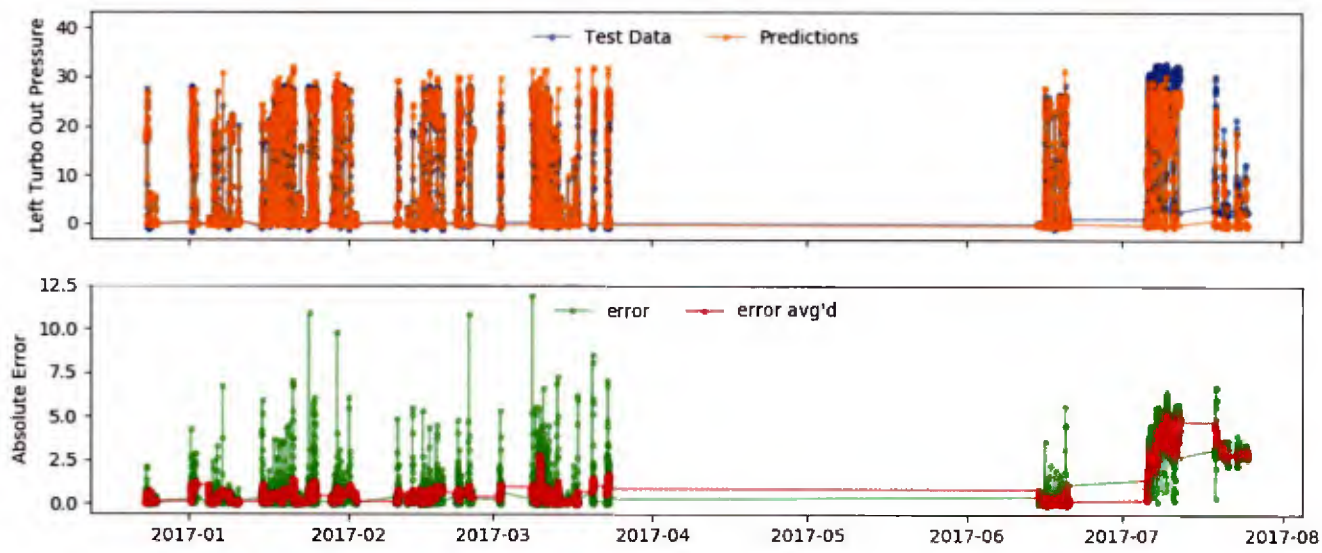


Figure 114: SVR Results on predicting left turbocharger air output pressure.

Figure 115 provides a zoomed in look at the results over a narrower timeframe. The bottom subplot shows the absolute error between the test data and the predictions in green, the averaged error is shown in red.

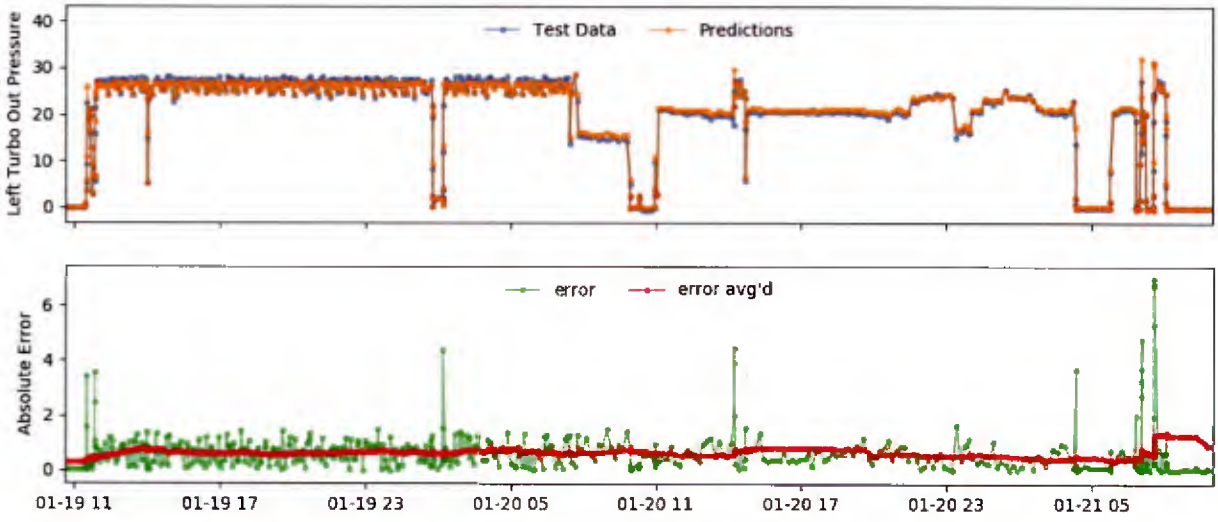


Figure 115: Test data, predictions, and errors for the SVR on predicting left turbocharger air out pressure from January of 2017 where prediction error was low.

Around July 2017 there was a spike in model error. The predicted values were about 2-5 psi lower on average than the measured values.

5.5.2.1.2 Error Analysis

In Figure 116 (an annotated copy of Figure 109), a region of points is highlighted that was not previously discussed in detail.

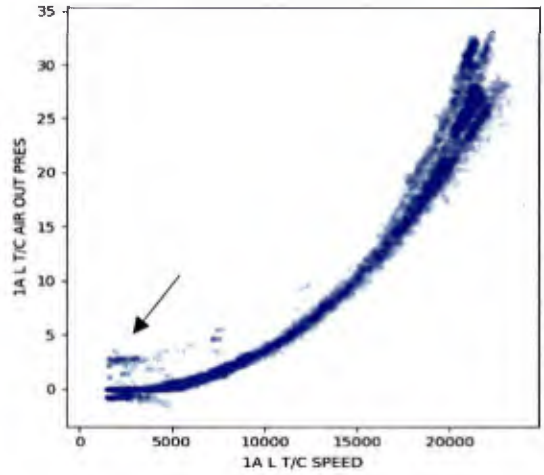


Figure 116: Left turbocharger output pressure vs. speed repeated.

These indicated points are all from after July of 2017. This is also where the model deviation begins to increase. Figure 117 shows three intervals of data that were evaluated by the SVR model. The green shaded region was designated the anomaly region based on the error spike from the initial model evaluation (Figure 115) and is the region from which the points indicated in Figure 116 are derived.

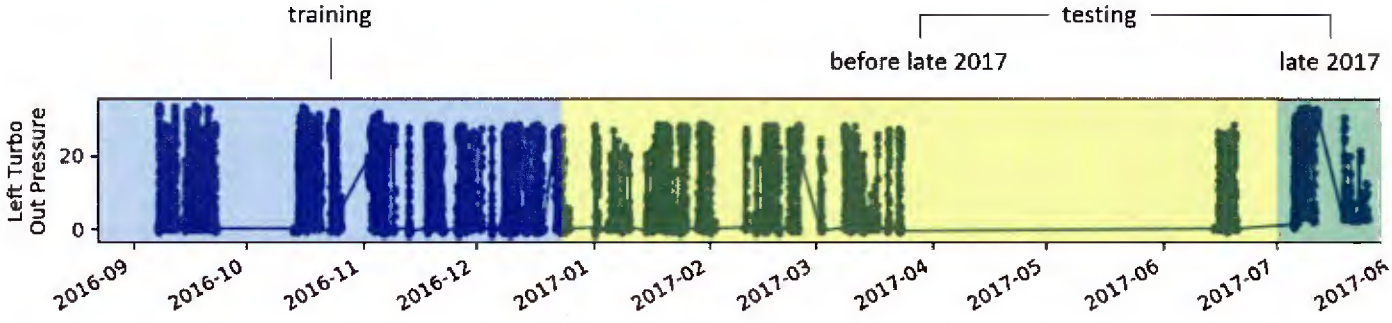


Figure 117: Turbocharger output pressure predictions for training, testing, and anomaly data.

Figure 118 shows the absolute error of the model evaluated on each region. The anomaly region (green) is highly distinguished from the training region (blue) and early 2017 testing region (yellow) in this figure.

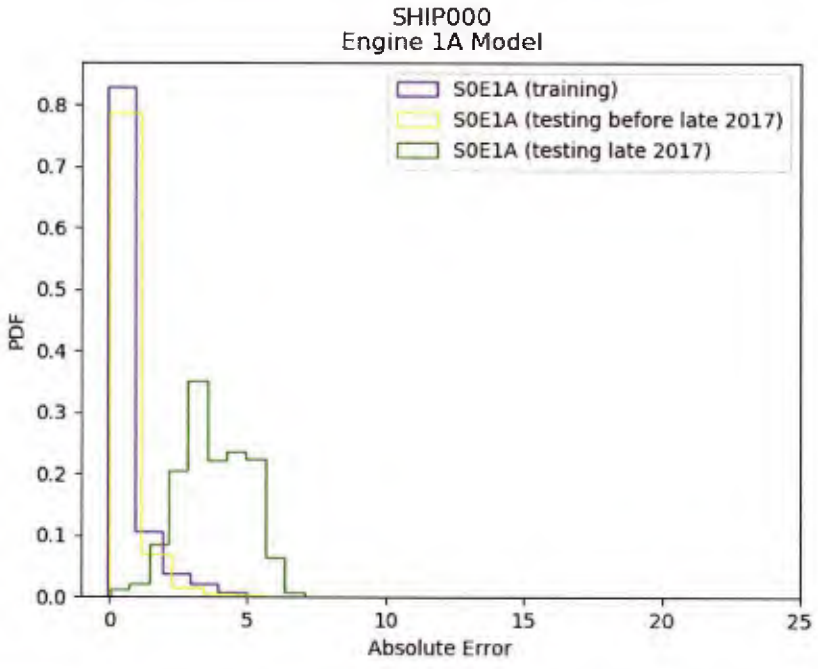


Figure 118: Histograms of each region of model evaluation shows the model's ability to detect anomalous behavior.

The model is therefore indicating a significant change in the behavior of the HUMS data. Referring to Figure 116, the data indicates an elevated turbo pressure when the engine is at very low rotation speeds, and similarly an increase in turbo pressure of about the same amount at higher engine speeds. The most likely explanation of this is that the sensor was out of calibration.

From a modeling standpoint, the SVR model has shown that it can accurately model the turbo pressure/speed relationship. In addition, a single layer Multi-Layer Perceptron (MLP) with 8 neurons was trained on the same data and achieved similar results.

5.5.2.1.3 Testing on Other Engines

As the model was built on engine 1A of Ship 0, the next natural step was to determine the applicability of the model to the three remaining engines on that ship. Figure 119 shows error distributions for the application of the model to the other engines.

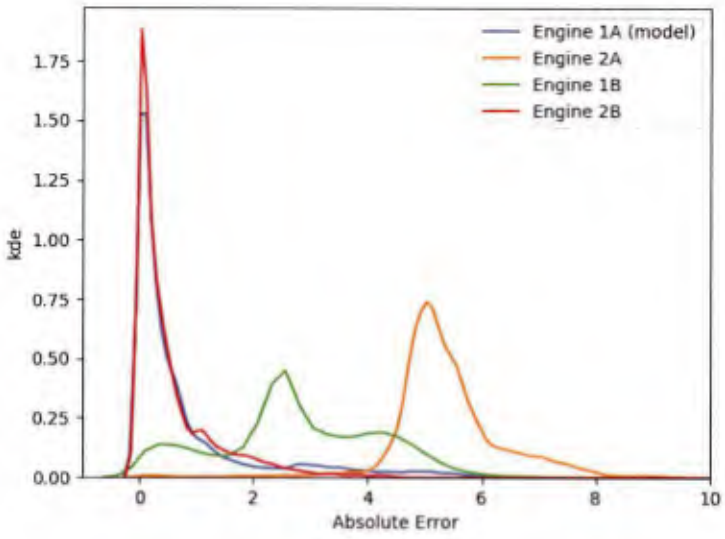


Figure 119: Engine 1A's model predicting all of Ship 0's engine's left turbocharger air out pressures.

The blue line indicates the model application to engine 1A, the engine that the model was trained on. Engine 1A has the lowest error alongside engine 2B, which had very similar results. Tuning a model for individual vehicles/subsystems is an important aspect of the digital twin concept. Applied to engines 1B and 2A, the model has significantly higher errors. An analysis of data from engine 2A indicates a likely calibration error or sensor fault due to an offset of about 5 psi in data values. There is not an obvious explanation for the error shown above for engine 1B.

These results indicate a physical understanding of system behavior is often important to inform model structure and interpretation, and that relatively simple models can provide effective anomaly detectors. The errors shown for engine 1B suggest that in some cases the normal model must be tuned to the individual asset – this is in line with the digital twin strategy that suggests a unique digital twin model for each individual asset.

5.6 Conclusions of the PHM Opportunity Investigation

The PHM Opportunity Investigation uncovered the prospect of data-driven PHM development, specifically at the first level of PHM capability, anomaly detection, utilizing modern deep learning methods like MLP and 1D CNNs as well as some classical machine learning methods, like SVR.

The approach used for this study was based on comparing and contrasting normal operation to the operation associated with failure in progress, where the data was supplied by HUMS and the ground truth by the recorded maintenance events. The opportunities were identified by the availability of the ground truth of failures, which came from the analysis of maintenance data. The first such opportunity was related to engine fuel injector failures and an anomaly detector was developed based upon a Multi-Layer Perceptron autoencoder model. Unlike classical machine learning models, which require high balance between healthy and fault-related data, autoencoders can learn the structure of the system just based on the normal operation. This feature is extremely valuable in PHM, where a large amount of data on normal operation is readily available, but data related to failures are scarce. While the general model developed

for this failure mode was not able to detect anomalies ahead of the maintenance events associated with the engine fuel injector failure, the same model worked very well when applied to transmission ECU failures. Moreover, the model trained on the HUMS data from one vehicle was demonstrated to detect anomaly in a transmission on another vehicle with no additional tuning.

MLPs, also referred to as *dense* or *fully-connected* neural network layers, are the first choice for new machine learning tasks. They were used in vision problems before CNNs and for speech recognition before Long-Short Term Memory (LSTM) networks. Because they are based on a set of unknown filters whose weights are determined during training from data, 1d CNN layers have natural ability to encode dynamical systems. In addition, these models tend to be more compact and better exploit the computational advantages to GPUs.

In addition to being useful as anomaly detectors, deep learning autoencoders have the potential to extract condition indicators when trained on failure-in-progress. In this case, the encodings (the outputs of the innermost, narrowest layer) are CI candidates. This method was explored using Navy data that consisted of maintenance events accompanied by HUMS data leading into the maintenance events. RIT developed approaches to evaluate plausible CI candidates, using model invariance as the criteria to distinguish potential degradation from spurious drifting encoding outputs: only the CI candidates that were consistent across multiple models were accepted. The approach showed promise when applied to the first selected set of events (ship-engine-event type), but no good CI candidates were found when the process was repeated on two more similar data sets.

To explore this modeling approach further, a simulation of a physical system was developed, which provided direct access to the ground truth of the evolving degradation, which was not available in the real-world data set. The physical system was a battery stack, which was selected based upon its relevance to CBM and its rich internal structure, featuring non-linear relationships among its states. The simulation was based upon a published Modelica ESS model. The model was extended to include the degradation, and the terminal (physically observable) variables (voltage and current) were used as the inputs for the machine learning model. The first level of success was demonstrated by the ability of the model to perform as an anomaly detector. Multiple autoencoders were tested to identify the impact of the number of parameters on the detection horizon of the failure. Future work will focus on the CI extraction, where the CI candidates will be assessed using the methods developed in this project, then evaluated by comparing them to the actual degradation.

After the identification of an anomaly and its classification with respect to the failure mode that caused it (i.e. diagnostics), the next level of capability is the damage assessment. The damage assessment can be either explicit, if it can be related to some ground truth, or implicit, indicating just the state that leads towards the failure. The implicit damage assessment is far more common. Analysis of failure progression, and the estimation of the future states of damage, falls in the domain of prognostics. While damage assessment was explored using autoencoders in the context of a specific example. More generally, the data-driven approach to damage assessment and prognostics based upon representation learning demands multiple examples of failure progression in order to learn from the data. However, while the large amounts of data are needed for learning of the general representation, tuning the model to learn degradation is expected to be faster and less demanding with respect to the data volumes. After the general representation is learned, the model is fine-tuned using few progressions of failure. It is reasonable to expect that the innermost layer -- the encodings – will be good candidates for data-driven

CIs. The forecasting of the progression of failure still must be conditioned on the assumptions on future operating and environmental conditions [29]. In a similar vein, because all CIs are sensitive to the operating conditions [8], whether they are engineered, or data-driven, it seems plausible that the representation learning should be trained to infer the operating conditions in addition to CIs [33]. It is worth reemphasizing that the physics that describes a progression of an incipient failure, while related to the physics of normal operation, are distinct phenomena and that must be either 1) learned from the data (that consists of multiple instances), 2) characterized empirically in a form of a phenomenological model, or 3) supplied by physics, with empirical determination of some parameters.

A system level model for a ship engine was developed to show how signal relationships and operating conditions can impact the learning of behavior over time. A classical machine learning model, the SVR, was built to predict output pressures on a turbocharger where different operating states, due to the dependent behavior of a left and right turbocharger on engine speed, were present. Classical machine learning methods are still indispensable when the data sets are relatively small, because deep learning requires large data volumes and domain knowledge can be exploited to make up for the lack of data.

6 Best Practices and Emerging Trends

6.1 Layers of PHM Capability: anomaly detection, diagnostics, and prognostics

The capability within a PHM system may be broken down into three layers, as shown in Figure 120, that provide progressively increasing value at the cost of increased data requirements and complexity. Implementation of the layers of PHM may be performed over-time with increasing capability and sophistication [6, 7]. In general, the progression from anomaly detection, to diagnostics, to prognostics, gives increasing resolution and refinement to both operational and maintenance decisions. Anomaly detection says simply that something is wrong, but does not address the severity of potential failure, nor the timeliness of action required. Diagnosis adds information about what is going wrong, so that severity of the potential failure can be understood, as well as better guiding maintenance activity to the root cause. Finally, prognosis aims to give an estimate of how timely a repair or operational accommodation is - do I have seconds, minutes, or hours until my operation is affected. Findings from the literature review performed are provided below and cover the current capabilities within each of the PHS system layers, a review of applicable systems, an overview of activities in related fields, and the best practices identified.

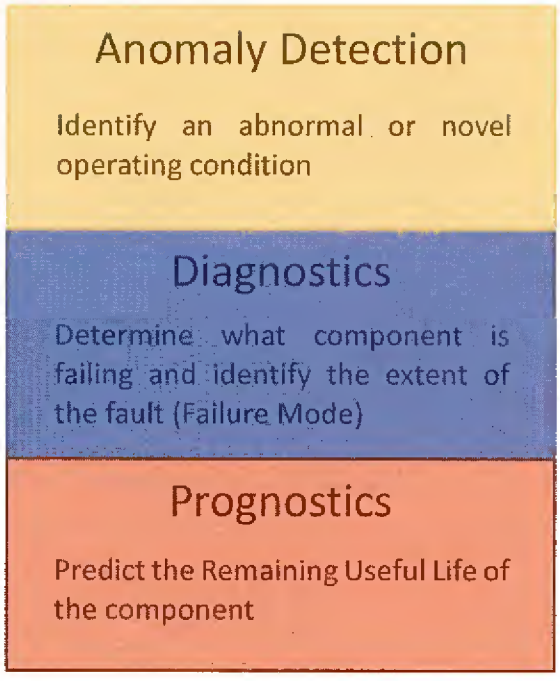


Figure 120 - The Layers of PHM Capability

6.2 State of PHM

In their seminal and highly influential paper, Engel *et al.* discussed details of computing remaining useful life in 2000 [9]. Worden *et al.* introduced *fundamental theorems of structural health monitoring* in 2007 [8], which is particularly useful to identify realistic and achievable goals for a PHM system.

A recent comprehensive review of state of PHM is provided by the Prognostics Center of Excellence at NASA Ames Research Center [34].

6.2.1 Current Capabilities (and limitations)

6.2.1.1 Anomaly Detection

Anomaly detection can be focused on simply identifying a behavior as "different from normal" – also termed novelty detection. Anomaly detectors can be developed without significant physical understanding of system behavior, although they do need an understanding (through data signals) of the operational context of the system (e.g. speed, load, etc.). In traditional machine learning, the key distinguishing factor of the success of a project is *feature engineering* [15], which represents ≈90% of industrial machine learning development effort [16]. The emerging trend (since mid-2000s) is to attempt to automate this process using *representation learning*. This idea had been very successfully applied before 2000. For example, Japkowicz, Myers, and Gluck [17] employed an autoencoding neural network that takes a high dimensional feature vector as the input, compresses it via a lower-dimensional hidden layer, and reconstructs it with the output layer of the same dimension as the input. This solution had very impressive performance in practice. The approach has considerable generality: the neural net learns a lower dimensional representation of the input data. The neural net is trained on "normal" inputs from a machine, so when an input that represents any of a multiplicity of possible "fault" conditions is input to the neural net it will fail to accurately reconstruct the input at its output. Deviation between measured output signals and the model results is used to develop a measure of likelihood of the presence of anomalous behavior, and/or of degree of the anomaly.

This model can be further enhanced using the recent advances in machine learning. The neural networks experienced a renaissance in the mid-2000s, when systematic approaches to deep learning architectures emerged [35]. While it has been known for some time that, given the same number of non-linear (neural network) units, a deep architecture is more expressive than a shallow one [36], deep nets were difficult to train in the mid-1990s. That difficulty and the emergence of support vector machines [37, 38] caused the second crises of neural networks in the mid-1990s. The situation changed in the early 2000s when new training algorithms for deep nets were discovered [39, 40]. The successful applications are growing in numbers and algorithms continue to improve [41]. The key distinguishing feature of the deep networks are layer-wise, pre-training via an unsupervised learning algorithm, use of new, simpler activation functions (ReLU), and new approaches to regularization (dropout [20]). The most popular approach is to use the restrictive Boltzmann machine and contrastive divergence. The promise of this technology is that

while learning of the *representation* requires large amounts of data, the data required for this learning corresponds to "normal operations", which is more readily available [42] (as opposed to requiring sets of normal and abnormal training data). The layer-wise pre-training using an unsupervised scheme requires relatively large datasets. Typically, data corresponding to known failure conditions is orders of magnitude smaller in volume, and often it is not available (or too limited) making supervised learning impractical. Thus, the unsupervised approach described above uses available data to learn the representation of normal operations, which enables anomaly detection immediately, and later, as data associated with faults become available, the system can be expanded to more sophisticated capabilities: diagnostics (classification of labeled faults) and prognostics (regression of fault evolutions). State-of-the-art implementations of deep learning algorithms are now widely available [32, 43-45] and will be used to expedite the training of the initial model. Building a deeper version of an autoencoder is now feasible and more layers will likely improve the performance over the model used in 1990s [17, 46].

Neural networks are *black boxes*, or *algorithmic models*, which produce eminently useful results (see Breiman [47] for the discussion on *two cultures* in statistical modeling). In fact, neural nets have recently outperformed other models in a number of cognition tasks [35]; however, they do not lend themselves to human interpretation. The price of achieving very high levels of performance is the model complexity: current neural networks employ enormous number of parameters. For example, Taigman [48] *et al.* trained a nine-layer network with 120 million parameters for 3D face modeling. In addition to neural nets, random forests [49], and boosted regression trees [50] are among algorithmic models that can be used to learn certain tasks or skills similar to cognition, or what is in psychology referred to as System 1 (or human sub-consciousness) [51]. Their successful performance must be validated on a data set separate from the data set used for training the model. All these models require large amount of data and considerable amount of validation. This project focused on deep learning and specifically anomaly detectors based on autoencoders because of their unique capability to learn the structure of a dynamical system from that data associated with normal operating conditions.

New ideas are currently being introduced and explored in the context of deep learning, such *adversarial learning*, which corresponds to a mini-max two player game, where one (generative) model attempts to synthesize data that can fool the other (generative) model [52] and *multimodal learning,* which learns multiple modalities (text, image, audio) simultaneously and can later predict audio based on text and vice versa. This advanced paradigm in machine learning has the capability to artificially synthesize high-fidelity data, which is otherwise scarce. For example, the anomaly data is far less accessible than normal operation, and the existing datasets can be artificially expanded in this manner.

6.2.1.2 Diagnostics

Assuming that proper instrumentation is in place to enable a comprehensive anomaly detection, the anomaly detection can, in principle, lead to a purely data-driven diagnostics over time. In this proposed process, maintainers and other domain experts label the detected anomalies to specific faults and no-trouble-found for those that could not be related to a failure or an incipient failure. This growing data set of labeled anomalies can be then used to train a machine learning classifier. Furthermore, if the anomaly detection is based on a deep learning autoencoder, the encodings can be potentially used as features for the classifier.

Classical approaches to diagnostics rely on large quantities of data that show progression from normal operation to failure. Vachtsevanos et Al.[53] provide an overview of the various methods of fault

diagnosis, including statistical classification and clustering[54, 55], data driven diagnostics[56], dynamic systems modeling[57], physical model-based methods[58, 59], model based reasoning[60, 61], and case based reasoning (CBR) [62, 63]. Diagnostics can be an important pathway to prognostics, as some of the prognostics methodologies first require a diagnosis of what failure(s) are thought to be developing.

6.2.1.3 Prognostics

Prognostics rely on an accurate assessment of the current state of health of the component or system. Estimation of remaining useful life (RUL) is then projected from the current condition to the failure threshold. Goebel *et al.*[29] identified four methods for reasoning about predictions of failure: 1) reliability analysis, using population statistics from control experiments and usage data; 2) damage accumulation, using specific load history and comparing it to population empirical models; 3) data analytics, using machine learning models and applying them on HUMS data; and 4) condition monitoring, using specific degradation models and applying them on load history. Because condition indicators depend on operating and environmental conditions (see Axiom IV in [8]), the degradation prognostics models are conditioned on assumed future operating and environmental conditions [29]. State estimation methods (e.g. unscented Kalman filter, particle filter, hidden Markov models) are often employed for predicting RUL because of their ability to integrate physics-based model for prediction of future states and measurements for correction. Prognostics models are generally broken down into three types: physics-based, data-driven, and hybrid [53]. They are described in turn below.

Physics-based models rely on a physical understanding of the degradation of the equipment, and the effects of operation on the mechanism of failure. Utilizing a physical damage model combined with measured data, predictions are made of the remaining useful life of the component or system. Crack-growth modeling[64] or wear modeling[65] are types of physics based prognostic models. The challenge in physics based modelling is that complex systems do not always lend themselves to accurate analytical or numerical treatment. The more complex the model becomes, the more system and material knowledge (and associated model parameters) are necessary, thus increasing requirements on model development efforts. In addition, prognostic models need deeper visibility into the system state, driving increased signal measurement requirements and associated parameter estimation and feature extraction. However, advances continue to be made that may further enable physics-based prognostics, such as the ability to facilitate system-level engineering solutions and collaborations by sharing models via the Functional Mock-up Interface (FMI) paradigm, and increasingly robust processors that can handle high degree of freedom analytical models.

Data-driven prognostics are black box models that learn equipment behavior. Typical data-driven models may be broken down into artificial intelligence (AI) or statistical approaches. Artificial intelligence approaches include neural networks[66-68] and fuzzy logic[69, 70]. Gaussian process regression[71, 72], least squares regression[73], support vector machines[74, 75], and hidden Markov models[76, 77] are all types of statistical prognostic approaches. A few of the challenges related to data-driven models are: 1) the need for large and expensive data sets that correspond to normal operation, 2) data sets often have few labeled faults and even fewer progressions to failure, and 3) expensive test rig data collections may not always capture relevant failures. As described earlier, recent advances have been made that allow models to learn from normal operations without having collected specific fault data, such as autoencoders.

Finally, the last type of prognostic is a hybrid model. Hybrid models integrate elements of both physics-based and data-driven models. These models may be done in series or in parallel[78]. Series models utilize the data-driven approaches to fine-tune the parameters of the physics-based models, wherein parallel models perform their calculations separately and join the results to make a prediction on RUL. Zhou et Al.[79] present a data fusion approach that combines data driven and physics based prognostics to improve prognostics predictions relating to performance degradation of PEM fuel cells. This hybrid approach utilizes the advantages of each separate approach to improve the predictions over the entire life of the system. The challenge with hybrid models is that they can suffer the same shortcomings as either physics-based or data-based prognostic models.

6.2.1.4 Decision Support

For decision support systems, with humans in the loop, opaqueness of algorithmic models is a significant deficiency. For example, it was shown above that an autoencoder error can be a good health indicator. However, the existence of a health indicator is not enough - a decision support layer that interprets the indicator is needed to make the indicator actionable. In the prior examples, simple automated reasoners, such as filter and threshold, or SPRT were used. For some practical implementation, more on this in Section 6.5, the CI thresholds are intentionally set low to avoid false alarms and engineers are carefully reviewing the data and its context to assess the presence of a fault.

As more models based on neural networks emerge, the treatment of their outputs is becoming critically important. There is a considerable reluctance to use the results from a black box model that cannot be readily interpreted, except in cases where the data-driven model has been demonstrated to work on a very large number of examples and with high confidence. This problem of trust without interpretation is at the frontier of current research in deep learning [80].

Humans arrive at critical decisions using deductive reasoning[3] or logic. This is also referred to as reasoning at the conscious level, or "System 2" as defined in [51]. The natural extension of explicit, reason-based decision-making, with full transparency and interpretability can be accomplished by Bayesian formulation. Bayesian formulation provides a principled approach to automate decision process in the presence of uncertainty and insufficient knowledge.

It is helpful to consider the timeline of the development of this approach. The formal logic dates back to antiquity. It started in the fourth century BCE; Aristotle introduced syllogisms, or logical arguments, to arrive at rational decisions (e.g. $A \Rightarrow B$ and $B \Rightarrow C$ thus $A \Rightarrow C$). Recognition that probability is just "common sense reduced to calculation" was due to Laplace (circa 1814), but formal connection between the Bayesian view of probability was established more than a century later. While formal logic employs deduction, plausible inference (an extension of logic) employs induction, Polya [81] . Cox [82, 83] captured the qualitative rules of the extended logic and showed that they obey the rules of Bayesian probability. These findings, together with earlier work of Jeffreys [84], who extended the original Laplace's objective priors from location parameters to scale parameters (Jeffrey priors), and Shannon's information theory [85] were synthesized by Jaynes [86] to form a coherent *objective* logical reasoning in the presence of uncertainty. This approach has been adopted in physics [87-90], machine learning [91, 92], information theory [93], and other fields. In parallel, approaches for reasoning under uncertainty with appropriate

[3] "A decision was wise, even though it lead to disastrous consequences, if the evidence at hand indicated that it was the best one to make; and a decision was foolish, even though it lead to the happiest possible consequences, if it was unreasonable to expect those consequences." -Herodotus, the fifth century BCE

accounting of prior belief based on individual experiences (or personal probabilities) also adopted the Bayesian framework [94, 95]. While the probability rules are intuitive[4], formulating them for practical use can be challenging[5]. Probabilistic graphical models exploit the intuitiveness of logical connections, ensure the coherency and consistency of the calculations, and take care of the necessary computations. The potential of Bayesian networks has been recognized in the 1980s [96, 97]. More recently excellent tutorials [98, 99] and books [91, 92, 100-102] have become available. This powerful method is now ready to be more widely deployed in cases where high-level decisions have to be made in the absence of all desirable information.

The decision-support system in Bayesian framework consists of two components: computational engine, and static expert (domain) knowledge. They are considered below in turn.

6.2.1.4.1 Computational Engine

Stochastic simulation, based upon Markov Chain Monte Carlo (MCMC) , has been identified as a powerful technique for coherent inferencing [96]. Although MCMC method dates back to 1940s [103], up until recently the problem with stochastic simulations was their computational requirements. Significant advances in semiconductor technology and associated computing devices have enabled faster computation, and coupled with software developments changed the situation. In the recent past there has been a great deal of development of software tools that enable Bayesian computations, including BUGS [104], JAGS [105], PyMC [106, 107], Church[108], Stan [109], Infer.NET [110] and others. Thus, the first component of a Bayesian decision-support system is already available: these computational frameworks greatly accelerate implementation of a decision-support system, ensuring that the underlying computational principles are properly implemented.

6.2.1.4.2 Static Expert Knowledge

Although subjective, expert knowledge is extremely valuable and should be integrated into the overall decision-support system. It can provide statistically independent knowledge that can be fused with the outputs of the algorithmic systems. However, this knowledge lacks quantification. The approach to capture the quantification is to make the assumption of uncertainties explicit, in the form of plausibility distributions. For example, a domain expert can be queried for his/her belief in certain outcomes within certain contextual scenarios. It is important to stress that it is not necessary to fully characterize the space of possibilities. An attempt to be exhaustive can be paralyzing. Instead, a pragmatic approach, which exploits knowledge where it is available, and admits ignorance where it is missing, has the promise to be a good starting point for development of a decision-support system.

6.3 Additional Literature for Selected Subsystems for Military Vehicles

From the vehicle perspective, certain subsystems are common across all platforms and mixed vehicle fleets. When evaluating ground vehicles and ships, the following subsystems were identified as common systems that have a history of PHM research: engines, gears/gearboxes, and electrical power systems. Provided below are a few select papers, which contribute to or summarize PHM research in those areas.

---

[4] Pierre-Simon Laplace recognized the potential: "Probability theory is nothing but common sense reduced to calculation."

[5] François-Marie Arouet Voltair recognized the problem: "Common sense is not so common". In the more recent past Court of Appeal bans Bayesian probability http://www.bailii.org/ew/cases/EWCA/Civ/2013/15.html

6.3.1 Engines

There is a long history for engine aircraft PHM. Rice's 1994 patent [111] describes methods for detecting partial and full engine failures. Xu *et al.* [112] described a framework for fusing data-driven with experience-based approaches to prognostics in the context of aircraft engines. An overview of different approaches to PHM of complex aerospace components is given by Patnaik *et al.* [113]. He and Feng [114] present a fault diagnosis approach for on-line detection of diesel injection faults utilizing fuzzy pattern recognition on the injection pressures. Kimmich et al.[115] describe model-based approaches for fault detection in diesel engine intake, injection and exhaust systems utilizing residual generation semiphysical models and neural networks. Lebold et al. [116] present multiple approaches for fault analysis of diesel fuel injectors, with the goal of embedding a low computational power algorithm into the vehicle engine controller. Li et al. [117] provide a review of approaches to prognostics on rotating machinery, covering Bayesian theory models, proportional hazard models, neural network based models, support vector machine based models, and similarity based models. Delvecchio et al. [118] review approaches to monitoring internal combustion engines with vibro-acoustic signals, which includes a number of common faults and analysis techniques.

6.3.2 Gears

Of the four dominant modes of gear tooth failure (breakage, wear, pitting, and scoring), the breakage is the most catastrophic and occurs precipitously, with no advanced warning. From the fatigue viewpoint, the lifetime of a gear can has two phases: crack initiation and crack propagation [119-121]. The gear research community has developed many vibration-based, condition indicators (CIs), to detect these features and assess damage, as summarized in [122-124]. To validate the performance of these CIs, researchers have been employing crack propagation (CP) sensors (see e.g. [125]). Typically, the crack propagation sensors are used to measure crack lengths on the surface of mechanical structures. A study focused on the analysis of signals from CP sensors implemented on spur gears considered two types of tests: crack propagation in a single-tooth fatigue-based tester and crack propagation in a dynamometer. In both cases, the CP sensors provide measurable ground truth for the level of damage. After researchers started seeding cracks on fatigue fixtures [126, 127], it was found that propagating cracks on a fatigue tester are much easier to observe than those in a gearbox.

6.3.3 Bearings

Reliability and failure analysis of bearings have a long history, with the early life models emerging in the 1940s [128]. A recent review of the life models of bearings is provided in [129]. The details of geometry and kinematics of bearings are described in [130]. The dominant causes of failure are fatigue, wear, plastic deformation, corrosion, brinelling, poor lubrication, faulty installation, and incorrect design [131]. Bearing PHM is chiefly concerned with fatigue and vibration-based CIs have been traditionally used for detection of incipient failures (e.g., kurtosis was first introduced in the context of bearings [132]). Detection of incipient failures due to different failure modes (old grease, flaws in the inner race, outer race, and a ball) has been investigated since the 1960s [133]. Good classical reviews of the bearing research can be found in [131, 134] and a recent authoritative tutorial is [135]. Traditionally, CIs were based on vibration, but Acoustic Emission (AE) based CI have attracted the attention of bearings PHM in the more recent past [136-138].

6.3.4 Electronics and Electrical Power Systems

Keller *et al.* [139] introduced The Aircraft Electrical Power Systems Prognostics and Health Management (AEPHM) program to demonstrate PHM technologies and enable CBM. The selected systems were electrical actuation and a fuel pump and valve. Arc fault and pump degradation were particular failure modes of interest. The framework was based on Bayesian network models and a probabilistic reasoner.

De Martin *et al.* [140] presented physics-motivated (but data-driven) PHM for turn-to-turn shorts in brushless electric motors (dominant failure in flight actuators), with an interesting discussion on pre-conceived vs. better-performing features.

For semiconductor devices, there are four main failure mechanics: *electromigration* (EM), *hot carrier injection* (HCI), *time-dependent dielectric breakdown* (TDDB), and *bias temperature instability* (BTI) [141, 142]. In addition to these, total ionization dose (TID) may also play a role. Though not typically critical, soft errors due to radiation are also of concern for the reliability of terrestrial electronics systems (e.g. satellites) based on CMOS technology [143]. A comprehensive review of physics of failure-based modeling was provided by White and Bernstein of NASA [144]. Suhir proposed improvements to Arrhenius model [145, 146].

Wang *et al.* [147] discussed future trends of reliability of power electronics, including physics of failure criteria of power electronics components.

Few examples exist of successful component electronics prognostics. Changes in *equivalent serial resistor* (ESR) R_{ESR} over time is a good indicator of electrolytic capacitors [148]. The reduction in the collector-emitter ON voltage $V_{CE(ON)}$ was found to be an aging indicator and precursors of failure for *insulated gate bipolar transistors* (IGBTs) [149, 150]. Xiong *et al.* [151] observed a degradation trace on $V_{CE_{sat}}$ of IGBTs. The system consisted of both hardware & software architecture and utilized simulation for verification.

6.4 Learning from Related Fields

Several other application domains require probabilistic estimates for future states. Evaluating how PHM is applied in related fields may enhance the capabilities on PHM relative to ground vehicles. Two areas that were further investigated were medicine and transportation systems.

6.4.1 Medicine

In the area of medicine, application of machine learning is a highly studied area due to the large number of available datasets (Healthdata.gov, https://www.kaggle.com, https://data.medicare.gov/, https://hsric.nlm.nih.gov, and http://apps.who.int/gho/data/?theme=main are just a few of the many data repository providers). Esfandiari *et al.* [152] break down the activities in medicine into six "medical tasks": screening, diagnosis, treatment, prognosis, monitoring, and management. Data mining and machine learning may be applied during each of the six tasks.

The algorithms that are used within medical data mining fall into four types: Regression, Classification, Clustering, and Association Rule. Some studies have been done on hybrid algorithms, but these should not be confused with hybrid data/physics-based models. The hybrid approaches discussed here are hybrid models that utilize more than one technique of machine learning/data mining. The majority of the literature on machine learning in medicine has been focused on classification algorithms, as seen in Table 63.

Table 63 - Application of Machine Learning to Healthcare as Studied by Esfandiari et al. [152]

	Screening	Diagnosis	Treatment	Prognosis	Monitoring	Management	% of papers
Regression	2	3	3	6	2	1	7%
Classification	23	47	16	15	7	18	51%
Clustering	3	7	4	1	4	5	10%
Association Rules	4	7	5	2	4	1	9%
Hybrid Machine Learning	8	33	4	5	4	3	23%

Regression models are based on traditional statistical techniques and are most useful to studies on prognosis. Among classification algorithms, decision trees[153-156] are the most popular because the expert is able to understand the extracted information. Support vector machines and artificial neural networks are also popular in study, but are less favorable due to the incomprehensibility of the outputs. Data clustering performed using hierarchical or fuzzy clustering methods were useful in management due to their ability to their ability to classify gene expression. [157, 158]

Hybrid approaches are being used to increase performance and reduce the problem of incomprehensibility of the results. Many different models have been studied in combination: fuzzy –SVM and artificial neural networks[159], clustering and SVM[160], decision tree and genetic algorithms[161], and logistic regression and random forest[162]. Many more approaches have been studied and are provided in Table 64.

Table 64: Hybrid Algorithms Utilized in Medicine

Reference	ANN	SVM	Fuzzy	Clustering	Classification	K-Means	Case Based Reasoning	Genetic Algorithms	Decision Tree	Evolutionary Algorithms	Random Forest	Regression
[163]		*					*					
[164]						*		*				
[165]		*	*					*	*			
[166]	*									*		
[167]	*	*	*									
[168]		*	*			*						
[169]		*	*									
[170]								*				*
[161, 171]								*	*			
[162]											*	*
[159]	*	*	*									
[160]		*		*								
[172]			*		*						*	*

Seera et. al. [172] proposes a medical classification system utilizing a hybrid approach of Fuzzy Min-Max Neural Networks, Classification and Regression Tree and Random Forest models. As is typical of a NN, the

outputs are not easy for a human to interpret what the system did. Therefore, the hybrid approach is taken to allow the system to incrementally learn from the data (NN), explain the outputs (regression tree), and achieve high classification performance (RF).

Sharaf-Al-Deen et. al. [173] proposes a hybrid Case Based Reasoning system that utilizes automated adaptation and reasoning rules extraction and reasoning to improve the outcomes for new cases. This process reduces the need for domain experts in the adaptation step of a typical CBR system.

6.4.2 Transportation

Traffic forecasting is a critical component of future intelligent transportation systems, yet is one of the more difficult predictions to make. Most predictions are not highly accurate, due to the fluctuations in traffic dynamics. As many models may be used to make traffic predictions, one approach is to combine the individual estimates, as opposed to selecting one. Granger, et Al. [174] determined that combining predictions performed better than any individual predictor. El Faouzi [175] provides a framework for combining traffic predictions into a single, improved indicator.

Another area requiring predictions in intelligent vehicle systems is accurate position estimation. Although GPS has become commonplace to many in their travels, the potential for a loss of GPS signal in certain areas of the country still exists. When GPS is lost, the system must keep an accurate estimate of position utilizing inertial navigation techniques. These techniques include multi-layer perceptron and radial basis function neural networks[176, 177] and adaptive neuro-fuzzy techniques[178].

6.5 Contemporary Trends and Drivers in PHM

As discussed in Section 1 and illustrated in Figure 2, development of PHM is appropriate for systems that are critical for overall operational readiness but do not fail frequently. The cost of development and implementation of PHM is one of the largest drivers for continuing advancement of PHM approaches. While more than one approach was cited in this review, PHM development is expensive irrespective of the approach. Key reasons for the high-costs of the development for three main methods, viz. physics-based, empirical, and data-driven are discussed here in turn.

The cost of developing of physics-based models for complex electro-mechanical systems is driven by the sheer enormity of the endeavor: a complex system consists of multiple subsystems and components that require different domain expertise (and multiple expert modelers) but the overall model must be fully integrated because the interaction among the subsystems and components are strong and the failures often occur at the interfaces (e.g. most failures in aerospace occurs at the interfaces of subsystems). In addition to the ability of modeling normal operation of the system, the physics-based PHM requires separate physics-based degradation models for different failure modes. Note, however, that many degradation models are actually empirical in nature (e.g. Paris's law [179], Lundberg-Palmgren model [128], various Arrhenius models for electronics components [145, 180], etc.). When the failure mechanism is understood, it often needs to capture second- and higher-order effects of the models, which in turn requires subtle modeling techniques and high resolution models at the component level, thus yielding larger system-level models that are slow to run and difficult to maintain. The digital twin paradigm has the potential to overcome critical barriers to physics-based PHM components by distributing the cost of the development of physics-based models. The Functional Mock-up Units (FMU)/Functional Mock-up Interface paradigm (Section 7.2.3) is a promising framework for sharing the models and building system-level capability, further reducing PHM development costs.

The cost of development of empirical models is driven by the cost of physical hardware, the cost of set-up and instrumentation and the cost of specimens and the labor associated with data collection of failure progression. Multiple instances of failure progression are needed to develop a reliable understanding of the progression of failure for each failure mode. Moreover, system disassembly and reassembly – an indispensable parts of seeding-failure processes – invariably impart slight changes to the system, which can be difficult to distinguish from early degradation. Finally, the seeded faults may not be a good representative of the actual field failures: sometimes the anticipated failure modes do not prove to be dominant; other times the method of seeding the failure may not accurately represent the actual failure. The latter problem can be illustrated by the practices in seeding gear cracks: until recently [126, 181] traditional seeding of gear cracks employed wire Electrical Discharge Machining (EDM), where a gear is notched instead of cracked, but the crack radius is very different than the notch radius and consequently the cracked teeth generally propagate faster than the notched cracks of the same length.

The cost of the data-driven approach includes initial instrumentation cost, which needs to be more comprehensive than some targeted PHM because this approach starts with less knowledge of the system failures; higher requirements for bandwidth of the communication because not just refined features but high-resolution raw data may be need for development of data-driven PHM modules; large data storage requirements and the overall system for managing the storage. The development is slow because it takes a long time for the right examples to emerge. The stochastic emergence of failures over time and across the fleet is not the same, which is the random process that generates failures is not approximately ergodic across the fleet. In addition, classical machine learning approaches struggled with large disk data volumes and small statistically relevant datasets [182]. Representation learning using deep neural networks, discussed in Section 5.6, overcomes the problem at least in the context of anomaly detection because it can learn the normal behavior based upon abundantly available data associated with typical operating and environmental conditions. However, for the prognostics capability, the degradation models has to be learned, with its accompanied uncertainties, such as the uncertainty about the future operating and environmental conditions which affect both RUL [29] and the features (see Axiom IV of [8]).

In spite of the high development costs, a successful development and implementation of PHM was demonstrated for battery health management [183] with key early contributions in the late 2000s [184, 185] and is still attracting the interest of research (e.g. [186, 187]). While modeling and simulation was important (with empirical, multi-physics based, electrochemical and molecular models [188]), the key enabler of the development was the availability of large datasets: data associated with large number of charge-discharge cycles were used for time-to-discharge predictions and data associated with degradation of multiple battery modules for predictions of capacity fade.

Another successful, semi-automated approach is adopted by Navair for their monitoring of the health of helicopters. Navair employs human-in-the-loop and a large set of CIs with conservatively set thresholds. These CIs intentionally produce a large number of false alarms, but the system relies on the engineers to interpret the CIs (and also sometimes raw data) to determine if the system has an incipient failures. This semi-automated system has been successful with detecting many faults, with zero false alarms.

Prognostics Health Management encompasses a significant ecosystem, ranging from model building, data collection, to data management and analysis capabilities. The above research on PHM identified a few high level emerging trends that could be employed to enhance the application of PHM within the military. They are identified below and will be expanded upon in section 7.

1) High performance computing is impacting the breadth and depth of machine learning capabilities. The application of new algorithms and approaches have been increasing as the use of cluster and GPU computing have evolved. Future computing systems, such as quantum computing may provide the next big boost in computational power.

2) Software flexibility is crucial. The software environments, libraries and interfaces that are used in the PHM community have been ever evolving, so the ability to adapt is necessary. Counter to typical software IT control, the data scientist will need an environment where they can install, update and try various software and libraries without going through an extensive development and deployment process.

3) The need for open datasets has been identified as crucial to the development, testing and comparison of PHM approaches. Within specific PHM domains, such as deep learning and medical health management, the availability of open datasets significantly contributes to the research, development, and application of new approaches. Closed datasets may have too few failures to be able to truly model the system and failure progression within the system. However, a large, open dataset would consolidate many failures into a single dataset, enhancing model development.

7 Roadmap

PHM development needs include both the development of PHM science, and the development of effective PHM systems and system architectures. A roadmap is provided here that addresses the development of PHM systems, including multiple different system considerations.

7.1.1 Backdrop

PHM is not evolving in isolation. There are several important concepts that are developing in data science, engineering, and in synergetic industries. These advances can be leveraged in PHM systems and foster their growth.

7.1.2 Industry 4.0

A new PHM system must recognize and embrace the rapidly changing landscape in data science, business analytics, and computational hardware, both at the edge and in the cloud (server side). More broadly, society is experiencing the fourth industrial revolution, "Industry 4.0", including additive manufacturing, augmented reality, autonomous robots, cyber security, high-performance computing, big data and analytics, simulation, and software integration, as depicted in Figure 121. Each of the elements of Industry 4.0 have the potential to make a dramatic impact on PHM of the future. For example, additive manufacturing may revolutionize the way components are repairs. Augmented reality can enable effective domain-knowledge capture and better integration of the human in the knowledge loop. Development of AI technology and self-awareness of autonomous robots can translate to self-awareness of assets, which could include self-health-assessment. Cyber security will play an increasingly important role to protect data transport networks. High performance computing is already revolutionizing machine learning, which immediately impacts data-driven algorithms. Simulation is critical for physics-based modeling, and advances in software (and integration advances) will ensure effective system function.

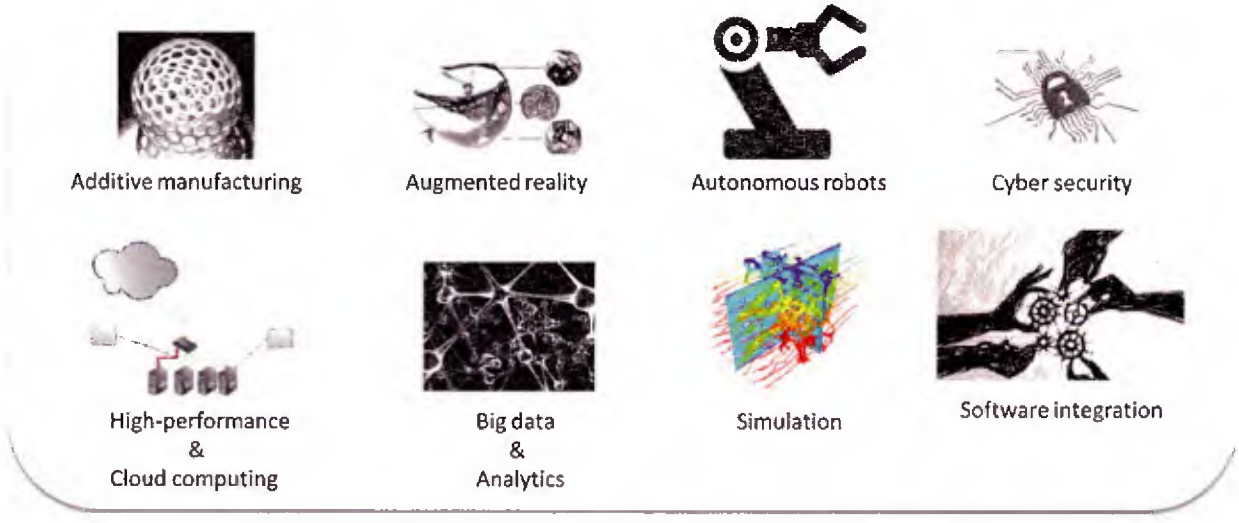


Figure 121: The fundamental elements of the fourth industrial revolution.

7.1.3 Open Source Software Environment

With recent rapid software development, open-source environments have been adopted to broadly engage the scientific, software, and engineering community. For example, the statistical community has embraced the R package [189], robotic development was greatly accelerated by the adoption of the open-source platform *Robotic Operating System* (ROS) [190], and simulation have benefitted by growth in use of Modelica [191]. Open-source PHM initiatives are emerging as well [192, 193].

The Python scientific computational ecosystem, with general-purpose libraries like NumPy [194], SciPy [195], Pandas[196], Ipython [197], machine learning Scikit-Learn [198], and visualization libraries Matplotlib [199], Mayavi [200], and others, is especially promising for new development. The great advantage of this environment is that it can integrate well with other frameworks, including for example, the aforementioned R and Modelica, as illustrated in Figure 122, further enabling hybrid prognostic approaches. The growth and popularity of this ecosystem is further solidified as Google and Facebook have both selected this environment for the primary interface of their new frameworks for deep learning, viz. TensorFlow [32] and PyTorch [201]. These frameworks were made open-source to further accelerate their development. Leveraging these tremendous works to build applied machine components for the future of PHM systems would allow these solutions to evolve and thrive.

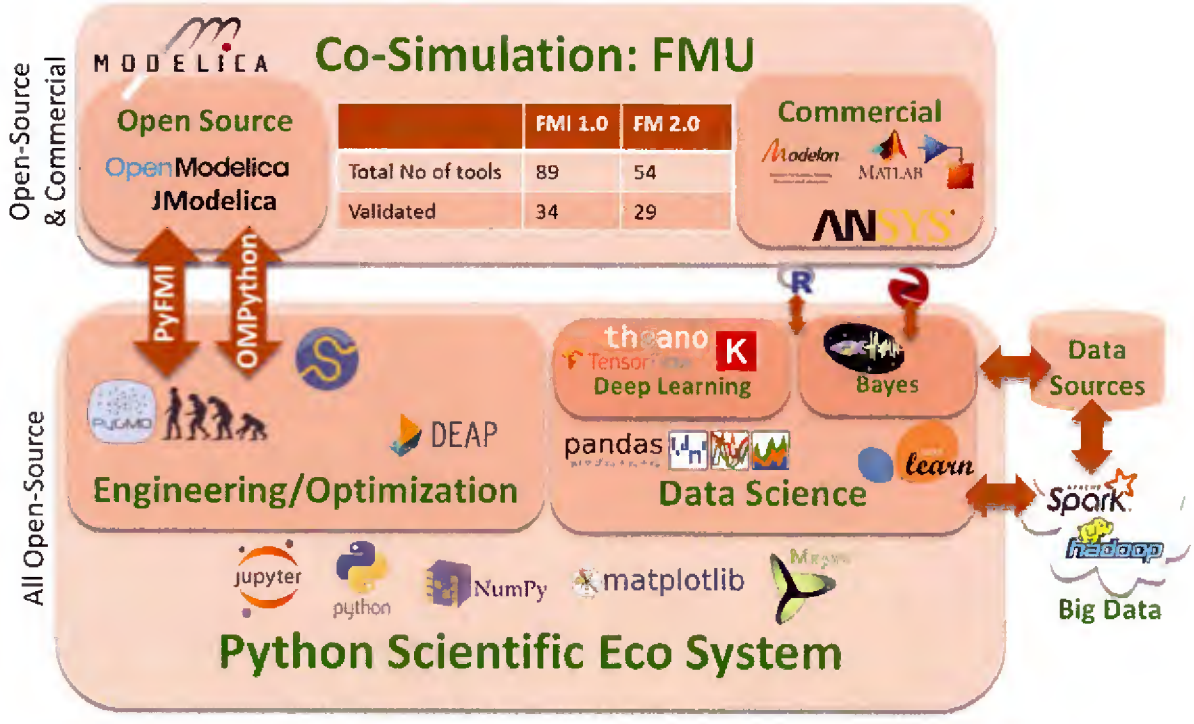


Figure 122: Open-source scientific ecosystem.

7.1.4 Advances in Hardware

In addition to the aforementioned advances in computational hardware, viz. Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs), new lower-cost sensors are becoming ubiquitous, widely deployed, and networked into internet-of-things (IOT) applications. The new engineering field – Cyber Physical Systems (CPS) – is becoming engaged in a variety of complex systems from microgrids, transportation systems, etc.

Micro-Electro-Mechanical Systems (MEMS) have been producing extremely low-cost inertial sensors commercially for decades. However, the trade-off between sensitivity and frequency bandwidth limited their applications in PHM to low-frequency monitoring and orientation sensing (MEMS accelerometers employ capacitive transducers that can measure constant acceleration). The recent advances made by Analog Devices, manufacturing low-cost accelerometers with bandwidth of 30 kHz, have the potential to bring a new paradigm shift in vibration sensing. While MEMS accelerometers are approaching the performance of traditional piezo accelerometers with the real potential to undercut them, piezo accelerometers are becoming smarter. They now facilitate integration of vibration measurements into HUMS through higher-level interfaces such as CAN databus and integrate more data processing features locally at the sensor, with preprocessing, such as Time Synchronous Averaging (TSA), built into the system[6].

Quantum Computing (QC), introduced in the 1980s, and studied theoretically at many universities in the 1990s, has burst into the public's imagination only over the past few years as the first proof-of-concept quantum computers emerged. This new paradigm has the potential to revolutionize back-end-

[6] https://www.dytran.com/CAN-MD-Phase-II/

computations, but some known technical challenges remain, and new challenges will likely be encountered.

Neuromorphic computing offers different opportunities in PHM system evolution; they are well suited for edge computing due to the low energy consumption. Like QC, the concept was introduced in the 1980s, but the field experienced a renaissance in the 2000s, with new prototypes. The computing is enabled by analog, digital, and mixed-signal Very Large Scale Integration (VLSI) circuits that mimic biological nervous system. A key feature is that the computations are asynchronous.

Table 65 Summary of the selected advances in hardware that have the potential to profoundly affect future PHM

	Technology	Advantage	Availability
Sensors	High-frequency MEMS	Low-cost vibration sensing	now
	Smart accelerometers	Integrated vibration CIs	now
Computing	Quantum computing	Fast, has the potential to change the back-end, research proof-of-concept	Near future
	Neuromorphic computing	Low energy consumption, has the potential to drive computation at the edge	First chips available (TrueNorth by IBM and Loihi by Intel)

7.2 PHM Trends and New Development in Related Fields

The key trends of PHM (identified in Section 6.2) are directly related to advanced development in related fields of study. Anomaly detection and diagnostics are connected to advances in deep learning and machine learning in general, with the importance of the development of open data sets for continuous advances and benchmarking being a major emphasis. Physics-based modeling is related to the development of system-simulation using the powerful new Functional-Mockup paradigm. Traditional PHM principles are being expanded to evaluate asset lifecycles in the form of digital twins.

A future PHM system needs to integrate its principal capabilities, viz. anomaly detection, diagnostics (including sensor fault detection), and prognostics. Integration of diagnostics and prognostics has been advocated for some time (see e.g. [202]). However, PHM is not all-or-nothing proposition; rather, the capabilities can be gradually added and as the performance improves and more relevant data becomes available [6, 7].

PHM systems are grouped according to the source of knowledge into three main categories [53, 203]: 1) data-driven, 2) physics-based, and 3) hybrid. Most practical systems are hybrid in nature. Irrespective of the type of model, prognostic capability is expensive to develop and is generally considered to be reserved for high-criticality systems with low-frequency failures. Reducing the development and deployment cost could allow the benefits of PHM and prognostics to be propagated to increasingly lower value hardware.

7.2.1 Data-Driven Anomaly Detection and Diagnostics

Because the frequency of failure mode occurrence in the field is low, it may take a long time, even with a large fleet, to build a statistically large data set that can be used for development of data-driven PHM systems. It is important to distinguish a *disk-large* dataset from a *statistically-large* dataset; the former refers to large quantity of data and the later large quantity of *relevant* and statistically independent data

from the problem at hand [182]. Insights in physics and, more generally, insights in domain expertise, can insert knowledge into the model and this knowledge can give rise to models that require less data.

An alternative approach is to develop PHM capabilities incrementally, starting with anomaly detection [6, 7], and progressing towards diagnostics and prognostics. This approach utilizes black-box systems, or algorithmic data learners [47], that include large decision trees (random forests [49] and boosted regression trees [50, 204]) and deep neural networks [35]. While it has been known for a long time that deeper neural networks are more expressive [36], the vanishing gradient problem [205] steered the practitioners away from more than one or two hidden layers until the renaissance in mid-2000s. However, Japkowicz *et al.* used an encoder-based deep neural network in the 1990s [17] to build a successful anomaly detector. Specifically, auto-encoder deep learning topology is particularly interesting from the PHM perspective, where operational data comes in abundance and failure data is scarce.

7.2.2 Open Datasets

Increasing the rate of development of prognostic algorithms would be facilitated by more benchmark and reference datasets. For example, the MNIST [206] data set played a significant role in the advancements of machine learning, including in the recent affirmation of deep neural networks [207]. There are many datasets for machine learning development in different domains [208]. Deep learning alone has multiple specialized datasets, viz. vision-natural images (CIFAR [209], Pascal VOC, etc.), vision-faces, music, and others [7].

This need has already been recognized by the PHM community and repositories of prognostics data exist [210, 211], with representation of some important systems such as lithium ion batteries and bearings. The availability of data for other subsystems would greatly increase the pace of development. This trend should be further supported in accordance to the recent acknowledgement from the National Academy of Sciences, which advocates for a new *open science* ecosystem, which includes scholarly publications accompanied by the analysis code and algorithms used to generate the results, and availability and usability of the associated data [212].

7.2.3 Physics-Based

Physics-based model development is an expensive and time-consuming activity. The models are domain-specific and difficult to generalize. Furthermore, the development is often fragmented across multiple analysis/simulation platforms. While mathematical formulation of physic-based models is software agnostic, a large portion of effort needed for a successful simulation is exerted within specific simulation package. Competing software environments, with the associated specialized engineering expertise, make co-simulation of a system, comprised of many subsystems developed by different entities, especially challenging. It is extremely valuable to be able to directly share models irrespective of the software development environment used to generate them.

[7] http://deeplearning.net/datasets/

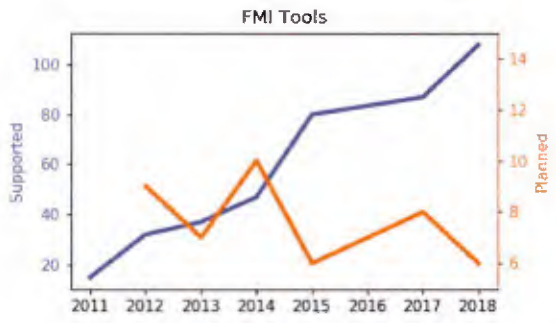


Figure 123: The growing number of simulation packages with FMI capability (adapted from [213]).

Functional Mock-up Interface [214, 215] is enabling and facilitating collaborations by allowing sophisticated models to be shared across multiple organizations, even when they do not share the same simulation environments. FMI provides a standardized, open, vendor-neutral Application Programming Interface (API), as well as a new paradigm. This new paradigm was embraced by the designers of complex systems for system-level simulations. In the automotive industry, FMI has been employed for some time [216] and is still considered state-of-the art [213]. OEMs and suppliers are exchanging models using Functional Mock-up Units. The FMUs are being integrated in engine test benches for real drive emissions tests. Additionally, the FMUs are being integrated in Software-in-the-Loop and Hardware-in-the-Loop applications. Aerospace industries are employing this technology for real-time simulation of aircraft systems [217]. The potential of FMU/FMI paradigm has been recognized for DOD: the National Defense Industrial Association (NDIA) Simulation Committee has recognized the importance of open standards and is tracking the overall adoption and implementation of FMI as an international standard. Many major simulation software systems have embraced this model exchange, including ANSYS, dSPACE, Dymola, MapleSim, Matlab/Simulink, Modelon, and NI LabVIEW. Figure 123 shows the number of software packages that have developed FMI support: the blue trace on the left axis shows the monotonic growth of the number of supported tools over time; the orange trace shows the planned expansions over time.

These physics-based models have a great potential not only for simulation of normal operation, but also for PHM development, especially as new platforms emerge in the future and system level simulation modeling, based on a variety subsystem/component FMUs becomes available. FMUs for system level simulation should be provided by the OEMs whenever possible, at least in the form of black box model that produces the responses to the defined inputs. The trade-off between model transparency and protection of intellectual property can affect how much detail an OEM and FMU paradigm allows the control of what states can be exposed to the system level simulation. FMUs for failure models, such as fatigue life prediction model for a specific mechanical component, are critical for full PHM development. These models are not always available from the OEMs and may need to be developed separately. Furthermore, the development of the failure models may be affected by the transparency of the FMUs involved in the system-level simulation.

7.3 PHM Relation to Digital Twin

There is considerable overlap between PHM and the concept of the digital twin. While it is generally understood that digital twin encompasses PHM, the boundary lines between the two are not immediately obvious. For example, consider a typical illustration of a digital twin of an asset, shown in Figure 124, where all of the attributes of the digital twin are also often used in PHM development. Before identifying distinctions between the concepts, it is instructive to briefly review the rapidly growing body of digital twin literature.

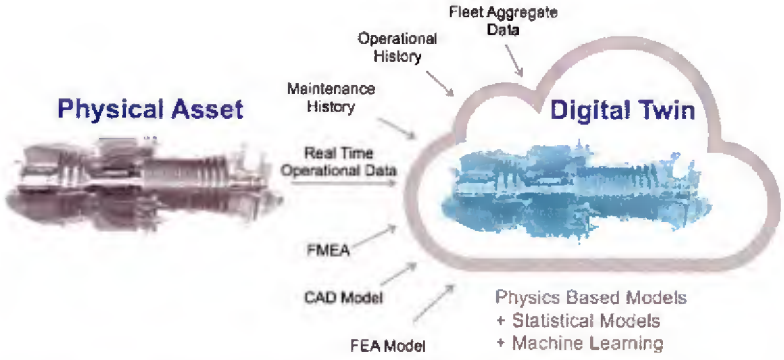


Figure 124: Illustration of a digital twin (source: https://www.loginworks.com/blog things-iot)

Table 66 illustrates the diversity of research under the digital twin umbrella. Most digital twin publications fall into two categories: visionary and enabling. Visionary papers often include a mockup implementation, while enabling papers are concerned with specific technologies, often simulations, for digital twin support.

Table 66: Literature Search Summary

Ref	Year	Vision	Implemented	Keywords	Digital Twin Components		
					Physics-Based Models	**Failure Models**	**Real Data**
[218]	2011	✓		Aircraft Structural Life Prediction	High level requirements for coupled physics models	Describes notionally fatigue crack models	None
[219]	2014		✓	Modeling as-manufactured geometry	None	Presents a use case for ductile fracture in a non-standardized material test specimen.	Fracture data on specimen
[220]	2016	✓	✓	AutomationML and IoT device (FIWARE)	AutomationML used for high-level models a valve with an actuator with temperature, battery level, and voltage sensor.	None	IoT middleware (FIWARE) used to communicate data.
[221]	2017	✓	✓	ABS (Braking)	Reduced order models generated from FEA of: • Electromagnetics model of the ABS valve solenoid actuator • Magnetic wheel speed sensor • Mechanical brake wear	Equation for Rate of Wear and a description of how missing teeth on the wheel sensor lead to abnormal ABS activation	None, proposed as future work

[222]	2017	✓	✓	Aircraft Wing	Gaussian process surrogate model to replace FEA to compute stress intensity factor K, and for computing crack growth.	Dynamic Bayesian network	Numerical simulations only.
[223]	2017	✓		Practical requirements for twin	All three columns covered in detailed, and philosophical vision, which goes deeper into asking important questions and demonstrating, via hypothetical scenarios related to real processes that happen now, practical use of the digital twin and what needs to change to move in that direction.		
[224]	2017		✓	JET divertor	FEA in Abaqus and ANSYS (commercial codes in C++) for 3 complementary applications of the JET divertor	Discusses need to evaluate impact of deviation from nominal geometry if changes in divertor are seen.	Temperature, heat flux
[225]	2017	✓		High level architecture, business use case	None	None	None
[226]	2017	✓	✓	Farming	None	Predictive analytics for specific applications in notes below	6 example in notes below
[227]	2017	✓		YouTube, Microsoft	Examples in video	None	Architecture for data collection, examples in video
[228]	2017	✓	✓	Modeling additive manufacturing	3D model for temperature, velocity fields, cooling rates, solidification parameters and deposit geometry.	Authors state predictive model outside scope of paper, still a long way in future	Laser power, material hardness, spacing/sizes

Historically, before the digital twin, there were *Physical* Twins: in the Apollo missions, two identical space vehicles were built to mirror the conditions of the space vehicles during the mission. The vehicle remaining on Earth was referred to as the "twin" and it was used extensively for training during flight preparation and during flight to simulate alternatives for mission critical situations. Similarly, an Iron Bird is a ground-based engineering tool used in aircraft industries to incorporate, optimize, and validate vital aircraft systems. Increasing power of simulation technologies and physical models have led to the replacement of the "hardware" twins with virtual models, "digital twins".

A simplified view of the history of digital twin is illustrated in Figure 125: the term is first introduced to the public in NASA's integrated technology roadmap in 2010 [229]. Another key motivation for the digital twin, which came in 2011, was a need for a design tool that would enable "prototype-free design" by using high-resolution multi-physics models [218]. In 2012, NASA the paradigm is introduced, with a formal definition: *The digital twin is an integrated multi-physics, multi-scale, probabilistic simulation of an as-*

build vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding twin [230]. In 2013, the digital twin became a part of the strategic initiative of the fourth industrial revolution [231]; the simulations aspects of Industry 4.0 and the role of digital twin is further examined in [232]; and considerations of human involvement are examined in [233].

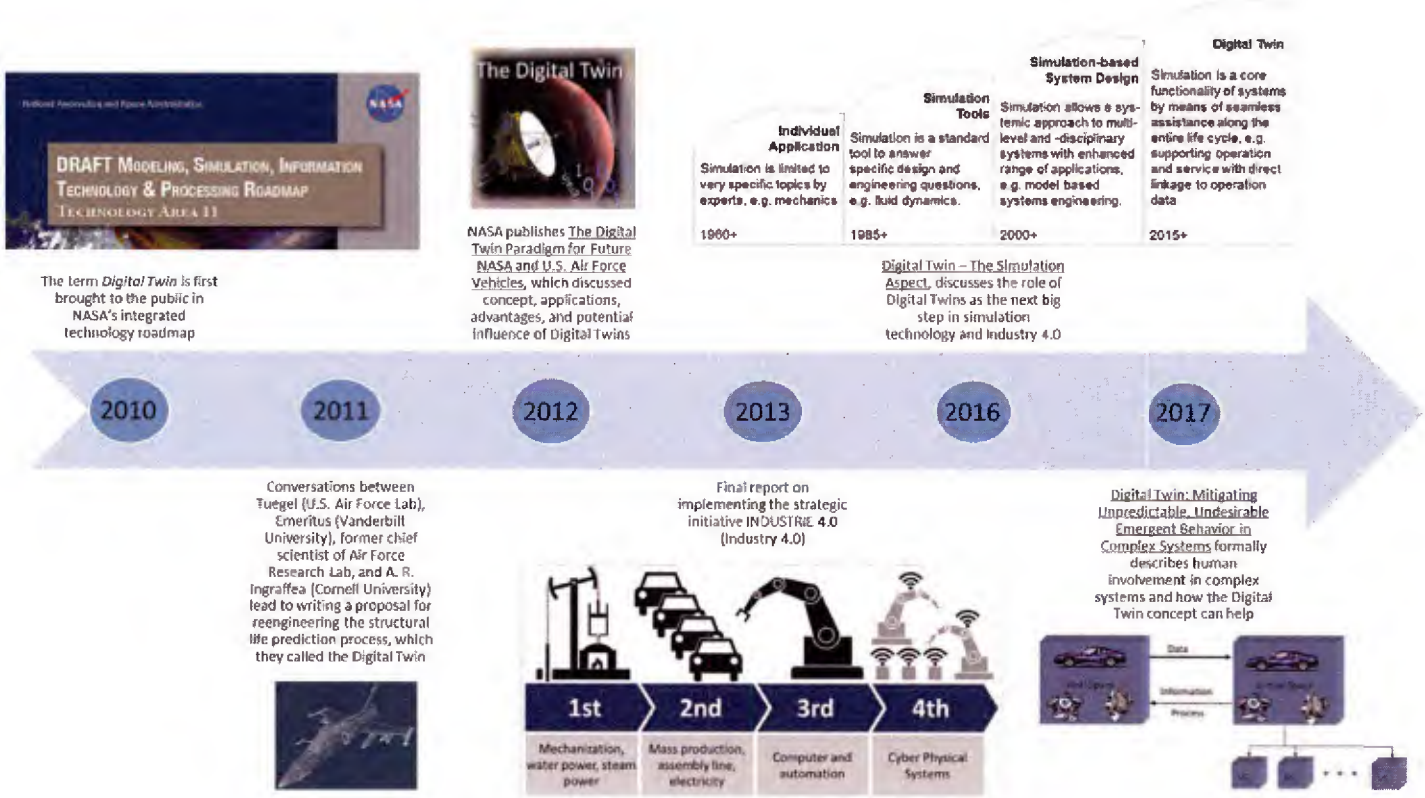


Figure 125: History of digital twin.

As seen in Figure 126, PHM is concerned primarily with operations and maintenance, whereas digital twin encompasses the entire life cycle of an asset. The diagram distinguishes between components and capabilities. Capabilities provide functionality that are enabled by components. For example, a PHM data-driven model is a component and anomaly detection is a capability. This view is greatly simplified and many interactions and relations were omitted. Table 67 lists PHM and digital twin references according to different system capabilities.

The significantly novel aspect of the digital twin is the recognition of individuality of an asset and related information that includes specific manufacturing parameters and operational and environmental history. In current practice, most physical models are representative of generic attributes of the product, and not of a particular asset. The idea of the digital twin offers the promise of capturing the as-manufactured state of an asset, which could be represented in the physical model, and then updating the model over the entire asset life cycle. A complete digital twin model would contain the asset data needed to build a robust prognostic model. In current practice the digital twin typically consists of a data based operational model, with a normative model with is specific to the serialized asset, and some tailored physical models to aid in assessment of remaining life, and performance and economic behavior. General Electric has developed demonstrations of digital twin technology through their Predix platform[234].

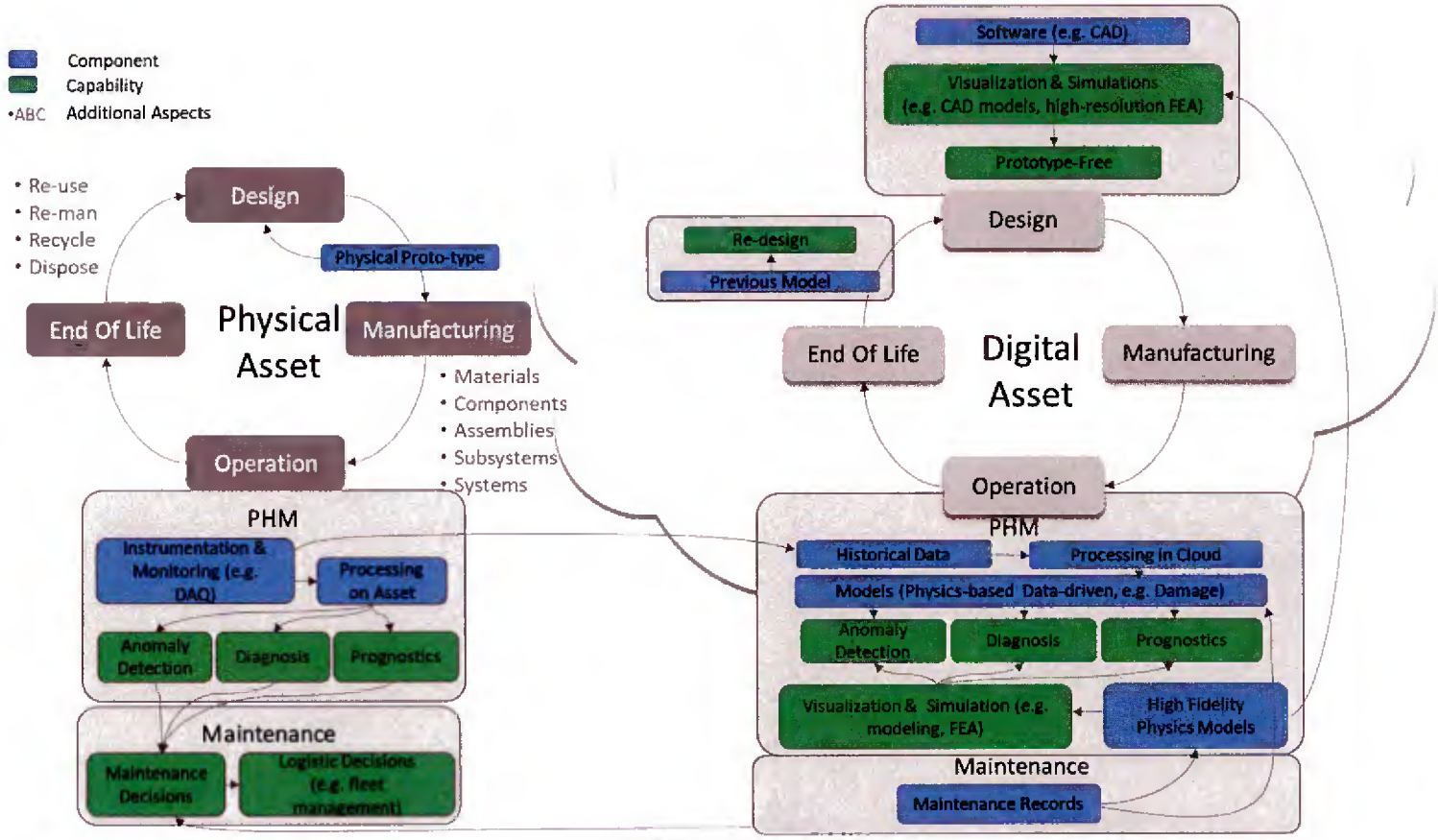


Figure 126: Digital twin and PHM.

Table 67: References by Digital Twin and PHM capabilities

Capability	Example	References in PHM	References in Digital Twin
Data Acquisition	On-board sensors, operational data	[235], [183], [236], [237]	[218], [220], [223], [224], [226], [227], [228]
	Environmental conditions (e.g. weather)		[218], [226], [227]
	Historical operational and maintenance data	[236], [237]	[218], [223], [224], [226], [227]
Data Processing	Located on asset (edge)	[235], [236], [237]	[220], [224]
	Located on a computer (cloud)	[237]	[218], [223], [224], [226], [227], [228]
	Mean-time-before-failure model	[235]	[218], [223]
	FEA, reduced order model	[235]	[221], [218], [222], [227]
	Parallel programming	[236]	
Anomaly detection	Rule-based (e.g. fault threshold)	[235]	[221], [223], [227]
	Data-driven (e.g. auto-encoders)	[235]	
Classification / Diagnosis	Classification / clustering methods		[218], [223], [226]
	Example of failure modes	[235]	[221], [219], [222], [223], [224], [227]
	Multiple failure mode models	[235]	[221]
Prognostics	Damage / lifing models (e.g. crack propagation)	[235], [236]	[218], [223], [219]
	Physical experiments		
	Numerical simulations on computer		[218], [219], [222], [224]
Decision Making & Human Interaction	FMEA – drives HUMS decisions	[236]	[228]
	Asset maintenance (e.g. scheduled repairs)	[235]	[223], [226], [227]
	Fleet management		[218], [226], [227]
	Digital replica on computer		[218], [223], [224], [226], [227]
	AR overlays with voice command		[218], [227]

7.4 ROS 2.0

HUMS is becoming an integral part of military mobility platforms and its capability and level of integration are expected to increase in time. While initial HUMS systems were designed and built to act as non-intrusive, silent observers, new HUMS systems are expected to be able to send requests for information to other modules. As the platforms employ higher level of automation, the cybersecurity considerations are moving to towards the top of the requirements. US Army has identified ROS 2.0 as the integration framework for development of their autonomous vehicles [238]. Future HUMS systems may need to be compatible with this paradigm. This section briefly introduces ROS and ROS 2.0.

Robotic Operating System (ROS) is a set of software libraries and tools that facilitate building robotic applications. The name is a misnomer because ROS is not an operating system; instead, it is a flexible framework with a collection of tools. ROS facilitates collaboration and allows robots to learn from each

other, as discussed in [239]. ROS is developed primarily for research in Linux environment, with some implicit characteristics, such as existence of workstation-class computational resources on board and excellent network connectivity, which limits its applicability.

ROS 2.0 is a new version of the framework, built from ground up, with the similar capabilities that aims to overcome perceived limitations of ROS and extend it more broadly to broader automation needs. It leverages some new technologies (e.g. Distributed Data Services (DDS)), which enables building ROS-like middleware system using off-the-shelf libraries. Its inherent cybersecurity made it the framework of choice for the development of an autonomous military vehicle by the US Army.

7.5 Data Storage

PHM requires significant quantities of data to be collected for model training. The "big data" problem consists of two major facets: data storage and data processing. The identification of the appropriate data solution actually depends on a balance of both. Traditional Relational Database Management Systems (RDBMS) are good when ACID compliance (Atomicity, Consistency, Isolation and Durability) and transactional consistency are important. Additionally, the traditional RDBMS does not scale well to very large data sets, particularly when database sizes move past 1 TB. With HUMS data, the data is typically collected and stored, so the transactional nature and ACID compliance is not as critical.

Another consideration is the computational framework for data management and data processing. Apache Spark is a current popular framework where data is distributed over a cluster of machines in a fault tolerant way. Spark scales well to large datasets, supports both structured and unstructured data, and is designed for large, distributed data processing tasks. A presentation comparing PostgreSQL vs a 1 node and 4 node Spark system utilizing data from the Automated Characterization of Health Information at Large-scale Longitudinal Evidence Systems (ACHILLES) showed a significant reduction in analysis time on specific benchmarks.[240] Thomas Dinsmore claims that for scalable open source data science, Spark is the only fully integrated solution available. [241]

Many of the analytics that Spark provides are geared toward business analytics, as much of the prognostics community is utilizing some form of Python, including Anaconda. The machine learning libraries and tools available through Python make it the tool of choice in the data science community, and it is growing faster than other tools, see Figure 127.

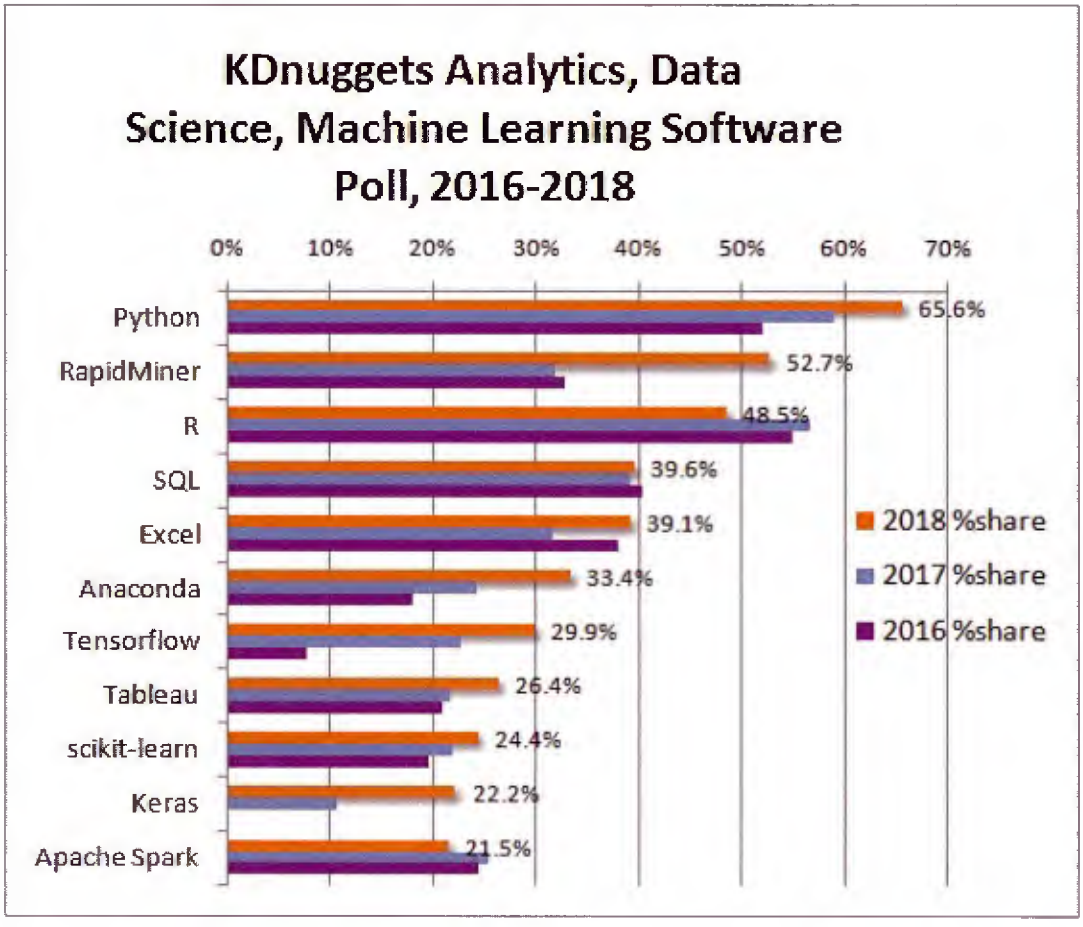


Figure 127 - KDNuggets Analytics Software Poll[242]

Although Spark is listed as a machine learning software, the availability of algorithm libraries is not as broad as with Python. However, Spark may still have a place in performing simple analytics, such as daily summaries of conditions or readiness reports. Python can access the data stored in the Spark distributed files system, allowing for big data storage and advanced scientific computing.

Another data format to be considered is the Hierarchical Data Format (HDF5), which is designed to store and organize large datasets. HDF5 can be accessed from Python and was recently incorporated into NetCDF, the next version of the CDF file format. NetCDF is essentially HDF5 with a few additional restrictions applied. Libraries are available for both HDF5 and NetCDF to access the data from many different languages: R, Perl, python, Ruby, Haskell, Mathematica, MATLAB, IDL, Octave, C, and C++.

In conclusion, data storage and processing requirements will differ throughout the CBM implementation. HUMS data, as seen in Section 4.1, is typically stored in some type of flat file format, such as ABCD. Off-board data may be joined into a merged, multiplatform data set in an environment like Apache Spark. Within Spark, typical business analytics reporting can be achieved, such as Availability reports. However, when performing a deep analytical dive into prognostics, retrieval of a specific subset of data relevant to the failure may make sense. This subset may be stored in a file based system, such as CDF or HDF5, allowing for rapid access and multiple trial runs over the smaller dataset. This can be seen in Figure 128,

which depicts the types of storage and processing being performed alongside the CBM+ Data Management Environments from the CBM+ Guidebook[2].

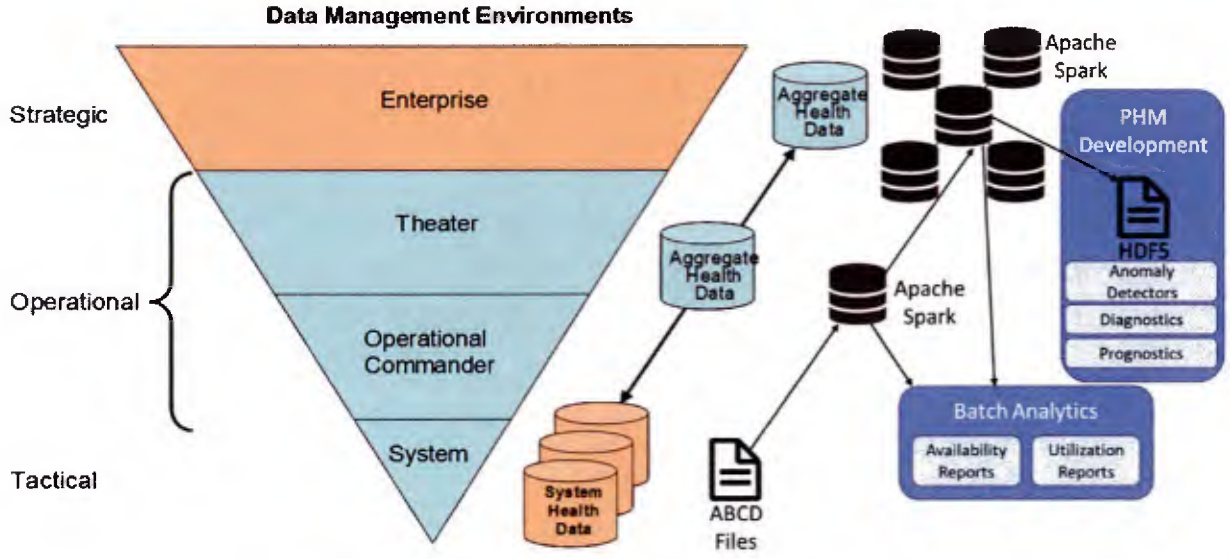


Figure 128 - Data management relative to storage and processing needs

8 References

[1] T. a. l. Under Secretary of Defense for Acquisition, "Department of Defense Instruction Number 4151.22," ed, 2012.

[2] U. D. o. Defense, "Condition Based Maintenance Plus (CBM+) DoD Guidebook," ed, 2012.

[3] M. Lebold and M. Thurston, "Open standards for condition-based maintenance and prognostic systems," in *Maintenance and Reliability Conference (MARCON)*, 2001.

[4] National Defense Industrial Association (NDIA), "Final Report of the Systems Engineering Division Integrated Diagnostics Committee," in *E-Prog II Electronic Prognostic Workshop*, Miami, FL, 2006.

[5] G. J. Vachtsevanos, *Intelligent fault diagnosis and prognosis for engineering systems*. Hoboken, N.J.: Wiley, 2006.

[6] J. Z. Sikorska, M. Hodkiewicz, and L. Ma, "Prognostic modelling options for remaining useful life estimation by industry," *Mechanical Systems and Signal Processing*, vol. 25, pp. 1803-1836, 2011.

[7] H. E. Bussey, N. G. Nenadic, P. A. Ardis, and M. G. Thurston, "Case Study: Models for Detecting Low Oil Pressure Anomalies on Commercial Vehicles," in *Annual Conference of the Prognostics and Health Management Society 2014*, **Fort Worth, TX**, 2014, pp. 252-264.

[8] K. Worden, C. R. Farrar, G. Manson, and G. Park, "The fundamental axioms of structural health monitoring," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 463, p. 1639, 2007.

[9] S. J. Engel and B. J. Gilmartin, "Bongort. K. and Hess, A.(2000)'Prognostics, The Real Issues Involved with Predicting Life Remaining'," pp. 457-469.

[10] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel, "Metrics for offline evaluation of prognostic performance," *International Journal of Prognostics and Health Management*, vol. 1, pp. 4-23, 2010.

[11] S. Uckun, K. Goebel, and P. J. F. Lucas, "Standardizing research methods for prognostics," in *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, 2008, pp. 1-10.

[12] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, ed: American Institute of Aeronautics and Astronautics, 2012.

[13] Spacy Physics Data Facility NASA / Goddard Space Flight Center. (2017, *CDF User's Guide, Version 3.6.4*. Available: https://spdf.gsfc.nasa.gov/pub/software/cdf/doc/cdf364/cdf364ug.pdf

[14] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the HDF5 technology suite and its applications," pp. 36-47.

[15] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, pp. 78-87, 2012.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*: MIT Press, 2016.

[17] N. Japkowicz, C. Myers, and M. Gluck, "A novelty detection approach to classification," in *IJCAI*, 1995, pp. 518-523.

[18] R. K. Jain, "Rajeswari Ponnuru (Intel), Ajit Kumar P.(Intel), and Ravi Keron N.(Intel). Cifar-10 classification using intel optimization for tensorflow," ed, 2017.

[19] Y. Abu-Mustafa, M. Magdon-Ismail, and H. T. Lin, "Learning from data: a short course," *Np: AMLbooks*, vol. 60, 2012.

[20] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.

[21] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807-814.
[22] T. Tieleman and G. Hinton, "Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.(2012)," *Google Scholar*, 2012.
[23] A. Wald and J. Wolfowitz, "Optimum character of the sequential probability ratio test," *The Annals of Mathematical Statistics*, pp. 326-339, 1948.
[24] A. Wald, *Sequential analysis*: Courier Corporation, 1973.
[25] K. C. Gross and K. E. Humenik, "Sequential probability ratio test for nuclear plant component surveillance," *Nuclear Technology*, vol. 93, pp. 131-137, 1991.
[26] K. J. Cassidy, K. C. Gross, and A. Malekpour, "Advanced pattern recognition for detection of complex software aging phenomena in online transaction processing servers," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 478-482.
[27] M. Pecht and R. Jaai, "A prognostics and health management roadmap for information and electronics-rich systems," *Microelectronics Reliability*, vol. 50, pp. 317-323, 3// 2010.
[28] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*: " O'Reilly Media, Inc.", 2017.
[29] K. Goebel, M. Daigle, A. Saxena, Sankararaman Shanakar, I. Roychoudhury, and J. R. Celaya, *Prognostics: the science of making predictions*: CreateSpace Independent Publishing Platform, 2017.
[30] F. Chollet, "Keras," ed, 2015.
[31] F. Chollet, *Deep Learning with Python*: Manning Publications, 2017.
[32] M. Abadi, *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467*, 2016.
[33] N. G. Nenadic, M. G. Thurston, and A. A. Hood, "Learning Operating Conditions for Gearbox Health Monitoring," in *American Helicopter Society 74th Annual Forum and Technology*

Display, Phoenix, AZ, 2018.
[34] K. Goebel, M. Daigle, A. Saxena, I. Roychoudhury, and J. R. Celaya, *Prognostics: the science of making predictions*: CreateSpace Independent Publishing Platform, 2017.
[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015.
[36] C. M. Bishop, *Neural networks for pattern recognition*. Oxford New York: Clarendon Press ; Oxford University Press, 1995.
[37] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273-297, 1995.
[38] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*: MIT press, 2002.
[39] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, pp. 1527-1554, 2006.
[40] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.
[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.
[42] W. Yan and L. Yu, "On accurate and reliable anomaly detection for gas turbine combustors: a deep learning approach," presented at the Annual Conference of the Prognostics and Health Management Society, San Diego, CA, 2015.

[43] F. Bastien, *et al.*, "Theano: new features and speed improvements," *arXiv preprint arXiv:1211.5590,* 2012.
[44] Y. Jia, *et al.*, "Caffe: Convolutional architecture for fast feature embedding," pp. 675-678.
[45] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," IDIAP2002.
[46] C. M. Bishop, "Novelty detection and neural network validation," *Vision, Image and Signal Processing, IEE Proceedings -,* vol. 141, pp. 217-222, 1994.
[47] L. Breiman, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical Science,* vol. 16, pp. 199-231, 2001.
[48] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701-1708.
[49] L. Breiman, "Random forests," *Machine learning,* vol. 45, pp. 5-32, 2001.
[50] J. Elith, J. R. Leathwick, and T. Hastie, "A working guide to boosted regression trees," *Journal of Animal Ecology,* vol. 77, pp. 802-813, 2008.
[51] D. Kahneman, *Thinking, fast and slow*: Macmillan, 2011.
[52] I. Goodfellow, *et al.*, "Generative adversarial nets," pp. 2672-2680.
[53] G. Vachtsevanos, F. L. Lewis, M. Roemer, A. Hess, and B. Wu, *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*: Wiley, 2006.
[54] S. Haykin, *Neural networks: a comprehensive foundation*: Prentice Hall PTR, 1994.
[55] S. Leonhardt and M. Ayoubi, "Methods of fault diagnosis," *Control Engineering Practice,* pp. 683-692, 1997.
[56] N. C. Propes and G. Vachtsevanos, "A fuzzy Petri-net-based mode identification algorithm for fault diagnosis of complex systems," *System Diagnosis and Prognosis: Security and Condition Monitoring Issues III,* pp. 44-54, 2003.
[57] V. A. Skormin, J. Apone, and J. J. Dunphy, "On-line diagnostics of a self-contained flight actuator," *IEEE Transactions on aerospace and electronic systems,* pp. 186-196, 1994.
[58] J. C. Newman Jr, "FASTRAN-2: A fatigue crack growth structural analysis program," NASA STI/Recon Technical Report N 921992.
[59] B. Wu, Saxena, A., Khawaja, T. S., Patrick, R., Vachtsevanos, G., & Sparis, P., "An approach to fault diagnosis of helicopter planetary gears," in *Proceedings AUTOTESTCON 2004*, 2004, pp. 475-481.
[60] J. a. W. de Kleer, B C, "Diagnosing Multiple Faults," *Artificial Intelligence,* vol. 32, pp. 97-130, April 1987.
[61] M. Gandy and K. Line, "Joint Strike Fighter - Prognostics and Health Management (PHM)," ed.
[62] C. K. Reisbeck and R. C. Shank, *Inside Case-Based Reasoning*. Hillsdale, NJ: L. Erlbaum Associates Inc., 1989.
[63] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and systems approaches," *AI Communications,* vol. 7, pp. 39-59, March 1994.
[64] M. E. Orchard, and George J. Vachtsevanos, "A particle-filtering approach for on-line fault diagnosis and failure prognosis.," in *Transactions of the Institute of Measurement and Control*, 2009, pp. 221-246.
[65] M. J. D. a. K. Goebel, "Model-Based Prognostics With Concurrent Damage Progression Processes," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2013.
[66] K. M. K Chakraborty, CK Mohan, S. Ranka, "Forecasting the behavior of multivariate time series using neural networks," *Neural Networks,* vol. 5, pp. 961-970, 1992.
[67] W. W. D Li, F. Ismail, "Enhanced fuzzy-filtered neural networks for material fatigue prognosis," *Applied Soft Computing,* vol. 13, pp. 283-291, 2013.

[68] J. L. F Ahmadzadeh, "Remaining useful life prediction of grinding mill liners using an artificial neural network," *Minerals Engineering,* vol. 53, pp. 1-8, 2013.

[69] R. G. R.E. Silva, S. Jemeï, D. Hissel, L. Boulon, K. Agbossou, N. Yousfi Steiner, "Proton exchange membrane fuel cell degradation prediction based on adaptive neuro-fuzzy inference systems," *International Journal of Hydrogen Energy,* vol. 39, pp. 11128-11144, 2014.

[70] F. M. E Zio, "A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of a nuclear system," *Reliability Engineering & System Safety,* vol. 95, pp. 49-57, 2010.

[71] A. Wilson, and Ryan Adams, "Gaussian process kernels for pattern discovery and extrapolation," in *International Conference on Machine Learning*, 2013.

[72] M. Seeger, "Gaussian processes for machine learning," *International journal of neural systems,* vol. 14, pp. 69-106, 2004.

[73] A. Coppe, Raphael T. Haftka, and Nam H. Kim, "Uncertainty identification of damage growth parameters using nonlinear regression," *AIAA journal,* vol. 49, pp. 2818-2821, 2011.

[74] B.-S. Yang, and Achmad Widodo, "Support vector machine for machine fault diagnosis and prognosis," *Journal of system design and dynamics,* vol. 2, pp. 12-23, 2008.

[75] H. T. Pham, Bo-Suk Yang, and Tan Tien Nguyen, "Machine performance degradation assessment and remaining useful life prediction using proportional hazard model and support vector machine," *Mechanical Systems and Signal Processing,* vol. 32, pp. 320-330, 2012.

[76] M. Dong, and David He, "Hidden semi-Markov model-based methodology for multi-sensor equipment health diagnosis and prognosis," *European Journal of Operational Research,* vol. 178, pp. 858-878, 2007.

[77] S. S. H. Zaidi, Aviyente, S., Salman, M., Shin, K. K., & Strangas, E. G., "Prognosis of gear failures in DC starter motors using hidden Markov models," *IEEE Transactions on industrial Electronics,* vol. 58, pp. 1695-1706, 2011.

[78] R. L. T.H. Penha, B.R. Upadhyaya, "Application of hybrid modeling for monitoring heat exchangers," in *3rd Meeting of the Americas - America's Nuclear Energy Symp.*, Miami, 2002.

[79] F. G. Daming Zhou, Elena Breaz, Alexandre Ravey, Abdellatif Miraoui, "Degradation prediction of PEM fuel cell using a moving window based hybrid prognostic approach," *Energy,* vol. 138, pp. 1175-1186, 2017.

[80] Z. C. Lipton, "The mythos of model interpretability," *arXiv preprint arXiv:1606.03490,* 2016.

[81] G. Polya, *Mathematics and Plausible Reasoning: Patterns of plausible inference* vol. 2: Princeton University Press, 1954.

[82] R. T. Cox, "Probability, frequency and reasonable expectation," *American journal of physics,* vol. 14, p. 1, 1946.

[83] R. T. Cox, *The algebra of probable inference*: Johns Hopkins Press Baltimore, 1961.

[84] H. Jeffreys, *The theory of probability*: Oxford University Press, 1939.

[85] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE,* vol. 37, pp. 10-21, 1949.

[86] E. T. Jaynes, *Probability theory: the logic of science*: Cambridge university press, 2003.

[87] S. F. Gull, "Bayesian inductive inference and maximum entropy," in *Maximum-entropy and Bayesian methods in science and engineering*, ed: Springer, 1988, pp. 53-74.

[88] T. J. Loredo, "From Laplace to supernova SN 1987A: Bayesian inference in astrophysics," in *Maximum entropy and Bayesian methods*, ed: Springer, 1990, pp. 81-142.

[89] G. L. Bretthorst, *Bayesian spectrum analysis and parameter estimation*. New York: Springer-Verlag, 1988.

[90] P. Gregory, *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with Mathematica® Support*: Cambridge University Press, 2005.

[91] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.
[92] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*: Academic Press, 2015.
[93] D. J. C. MacKay, *Information theory, inference and learning algorithms*: Cambridge university press, 2003.
[94] (2012, *After Blackouts, Connecticut Explores High-Tech Solutions*. Available: http://www.nbcnewyork.com/news/national-international/A-Year-After-Storms-Blackouts-A-State-Explores-High-Tech-Solutions-175871031.html
[95] B. De Finetti and B. de Finetti, "Theory of Probability, volume I," *Bull. Amer. Math. Soc,* vol. 2, p. 14, 1990.
[96] J. Pearl, "Evidential reasoning using stochastic simulation of causal models," *Artificial Intelligence,* vol. 32, pp. 245-257, 5// 1987.
[97] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* pp. 721-741, 1984.
[98] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning,* vol. 50, pp. 5-43, 2003.
[99] P. Diaconis, "The markov chain monte carlo revolution," *Bulletin of the American Mathematical Society,* vol. 46, pp. 179-205, 2009.
[100] M. I. Jordan, *Learning in Graphical Models*: London, 1998.
[101] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*. Boca Raton: Chapman & Hall/CRC, 2004.
[102] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*: MIT press, 2009.
[103] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association,* vol. 44, pp. 335-341, 1949.
[104] D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility," *Statistics and computing,* vol. 10, pp. 325-337, 2000.
[105] M. Plummer, "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling," p. 125.
[106] A. Patil, D. Huard, and C. J. Fonnesbeck, "PyMC: Bayesian stochastic modelling in Python," *Journal of statistical software,* vol. 35, p. 1, 2010.
[107] C. Davidson-Pilon, *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Methods*: ADDISON WESLEY Publishing Company Incorporated, 2015.
[108] N. Goodman, V. Mansinghka, D. Roy, K. Bonawitz, and D. Tarlow, "Church: a language for generative models," *arXiv preprint arXiv:1206.3255,* 2012.
[109] A. Gelman, D. Lee, and J. Guo, "Stan: A probabilistic programming language for Bayesian inference and optimization," 2015.
[110] T. Minka, J. Winn, J. Guiver, and D. Knowles, "Infer .NET 2.5," *Microsoft Research Cambridge,* 2012.
[111] R. W. Rice and D. H. Sweet, "Engine failure monitor for a multi-engine aircraft having partial engine failure and driveshaft failure detection," ed: Google Patents, 1994.
[112] J. Xu, Y. Wang, and L. Xu, "PHM-oriented integrated fusion prognostics for aircraft engines based on sensor data," *IEEE Sensors Journal,* vol. 14, pp. 1124-1132, 2014.
[113] A. R. Patnaik, A. Nayak, V. Narasimhan, and P. C. Patnaik, "An integrated PHM approach for gas turbine engines," pp. 2476-2479.
[114] Y. He and L. Feng, "Diesel Fuel injection System Faults Diagnosis Based on Fuzzy Injection Pressure Pattern Recognition," in *Proceedings of the 5th World Congress on Intelligent Control and Automation*, Hangzhou, 2004, pp. 1654-1657.

[115] F. Kimmich, A. Schwarte, and R. Isermann, "Fault detection for modern Diesel engines using signal- and process model-based methods," *Control Engineering Practice,* pp. 189-203, 2005.
[116] M. Lebold, *et al.*, "Detecting injector deactivation failure modes in diesel engines using time and order domain approaches," Pennsylvania State University, State College2012.
[117] X. Li, Duan, F, Mba, D, & Bennett, I, "Rotating machine prognostics using system-level models," in *Engineering Asset Management*, 2016, pp. 123-141.
[118] S. Delvecchio, Bonfiglio, P., & Pompoli, F., "Vibro-acoustic condition monitoring of Internal Combustion Engines: A critical review of existing techniques.," *Mechanical Systems and Signal Processing,* pp. 661-683, 2018.
[119] J. Kramberger, M. Sraml, S. Glodez, J. Flasker, and I. Potrc, "Computational model for the analysis of bending fatigue in gears," *Computers & Structures,* vol. 82, pp. 2261-2269, 2004/10//.
[120] J. Kramberger, M. Sraml, I. Potrc, and J. Flasker, "Numerical calculation of bending fatigue life of thin-rim spur gears," *Engineering Fracture Mechanics,* vol. 71, pp. 647-656.
[121] S. Glodez, M. Sraml, and J. Kramberger, "A computational model for determination of service life of gears," *International Journal of Fatigue,* vol. 24, pp. 1013-1020, 2002.
[122] J. J. Zakrajsek and D. G. Lewicki, "Detecting gear tooth fatigue cracks in advance of complete fracture," *Tribotest,* vol. 4, pp. 407-422, 1998.
[123] M. Lebold, K. McClintic, R. Campbell, C. Byington, and K. Maynard, "Review of vibration analysis methods for gearbox diagnostics and prognostics," in *54th Meeting of the Society for Machinery Failure Prevention Technology*, Virginia Beach VA, 2000, pp. 623-634.
[124] P. D. Samuel and D. J. Pines, "A review of vibration-based techniques for helicopter transmission diagnostics," *Journal of Sound and Vibration,* vol. 282, pp. 475-508, 2005.
[125] D. G. Lewicki and J. J. Zakrajsek, "Detecting Gear Tooth Fatigue Cracks in Advance of Complete Fracture," ed: Storming Media, 1996.
[126] N. G. Nenadic, J. A. Wodenscheck, M. G. Thurston, and D. G. Lewicki, "Seeding Cracks Using a Fatigue Tester for Accelerated Gear Tooth Breaking Rotating Machinery, Structural Health Monitoring, Shock and Vibration, Volume 5." vol. 8, T. Proulx, Ed., ed: Springer New York, 2011, pp. 349-357.
[127] D. q. B. q. Stringer, K. E. LaBerge, C. J. Burdick, and B. A. Fields, "Natural Fatigue Crack Initiation and Detection in High Quality Spur Gears," presented at the 68th American Helicopter Society Annual Forum, FT Worth, TX, 2012.
[128] A. Palmgren and G. Lundberg, "Dynamic capacity of rolling bearings," *Acta Polyteehnica,* 1947.
[129] E. V. Zaretsky, "Rolling bearing life prediction, theory, and application," 2013.
[130] P. Eschmann and L. Hasbargen, "Ball and roller bearings: theory, design, and application," 1985.
[131] I. Howard, "A Review of Rolling Element Bearing Vibration'Detection, Diagnosis and Prognosis'," DEFENCE SCIENCE AND TECHNOLOGY ORGANIZATION CANBERRA (AUSTRALIA)1994.
[132] D. Dyer and R. M. Stewart, "Detection of rolling element bearing damage by statistical vibration analysis," *Journal of mechanical design,* vol. 100, pp. 229-235, 1978.
[133] H. L. Balderston, "Incipient failure detection: Incipient failure detection in ball bearings," BOEING CO SEATTLE WA AEROSPACE SYSTEMS DIV1969.
[134] N. Tandon and A. Choudhury, "A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings," *Tribology International,* vol. 32, pp. 469-480, 1999.
[135] R. B. Randall and J. Antoni, "Rolling element bearing diagnostics—A tutorial," *Mechanical systems and signal processing,* vol. 25, pp. 485-520, 2011.
[136] J. Miettinen, P. Andersson, and V. Wikström, "Analysis of grease lubrication of a ball bearing using acoustic emission measurement," *Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology,* vol. 215, pp. 535-544, 2001/06/01 2001.

[137] D. Mba and R. B. K. N. Rao, "Development of Acoustic Emission Technology for Condition Monitoring and Diagnosis of Rotating Machines; Bearings, Pumps, Gearboxes, Engines and Rotating Structures," 2006.

[138] J. Z. Sikorska and D. Mba, "Challenges and obstacles in the application of acoustic emission to process machinery," *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering,* vol. 222, pp. 1-19, 2008.

[139] K. Keller, *et al.*, "Aircraft electrical power systems prognostics and health management," pp. 12-pp.

[140] A. De Martin, J. Giovanni, and V. George, "Windings fault detection and prognosis in electro-mechanical flight control actuators operating in active-active configuration," *INTERNATIONAL JOURNAL OF PROGNOSTICS AND HEALTH MANAGEMENT,* vol. 8, 2017.

[141] E. A. Amerasekera and F. N. Najm, *Failure mechanisms in semiconductor devices*, 2nd ed. Chichester ; New York: J. Wiley, 1997.

[142] E. Maricau and G. Gielen, *Analog IC reliability in nanometer CMOS*: Springer Science & Business Media, 2013.

[143] T. Heijmen, "Soft Errors from Space to Ground: Historical Overview, Empirical Evidence, and Future Trends," in *Soft Errors in Modern Electronic Systems*, M. Nicolaidis, Ed., ed Boston, MA: Springer US, 2011, pp. 1-25.

[144] M. White, *Microelectronics reliability: physics-of-failure based modeling and lifetime evaluation*: Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2008.

[145] E. Suhir, "Predicted reliability of aerospace electronics: Application of two advanced probabilistic concepts," in *2013 IEEE Aerospace Conference*, 2013, pp. 1-13.

[146] E. Suhir, "Aerospace-electronics reliability-assurance (AERA): Three-step prognostics-and-health-monitoring (PHM) modeling approach," in *2016 17th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, 2016, pp. 1-14.

[147] H. Wang, *et al.*, "Transitioning to physics-of-failure as a reliability driver in power electronics," *IEEE Journal of Emerging and Selected Topics in Power Electronics,* vol. 2, pp. 97-114, 2014.

[148] K. Harada, A. Katsuki, and M. Fujiwara, "Use of ESR for deterioration diagnosis of electrolytic capacitor," *IEEE Transactions on Power Electronics,* vol. 8, pp. 355-361, 1993.

[149] N. Patil, J. Celaya, D. Das, K. Goebel, and M. Pecht, "Precursor Parameter Identification for Insulated Gate Bipolar Transistor (IGBT) Prognostics," *IEEE Transactions on Reliability,* vol. 58, pp. 271-276, 2009.

[150] N. Patil, D. Das, K. Goebel, and M. Pecht, "Identification of failure precursor parameters for Insulated Gate Bipolar Transistors (IGBTs)," in *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, 2008, pp. 1-5.

[151] Y. Xiong, X. Cheng, Z. J. Shen, C. Mi, H. Wu, and V. K. Garg, "Prognostic and warning system for power-electronic modules in electric, hybrid electric, and fuel-cell vehicles," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2268-2276, 2008.

[152] N. Esfandiari, M. R. Babavalian, A.-M. E. Moghadam, and V. K. Tabar, "Knowledge discovery in medicine: Current issue and future trend," *Expert Systems with Applications,* vol. 41, pp. 4434-4463, 2014/07/01/ 2014.

[153] V. Zanten, A. P. Van Zanten, A. Twijnstra, A. A. M. Hart, and B. W. Ongerboer De Visser, "Cerebrospinal fluid lactate dehydrogenase activities in patients with central nervous system metastases," in *Clinica Chimica Acta*, 268, p. 259.

[154] T. H. Rainer, P. K. W. Lam, E. M. C. Wong, and R. A. Cocks, "Derivation of a prediction rule for post-traumatic acute lung injury," in *Resuscitation*, 187-196, p. 1999.

[155] Y. M. Chae, H. S. Kim, K. C. Tark, H. J. Park, and S. H. Ho, "Analysis of healthcare quality indicator using data mining and decision support system," in *Expert Systems with Applications*, 2003, pp. 167-172.

[156] Y. F. Wang, M. Y. Chang, R. D. Chiang, L. J. Hwang, C. M. Lee, and Y. H. Wang, "Mining Medical Data: A Case Study of Endometriosis," in *Journal of Medical Systems*, 2013.

[157] W.-H. Au, K. Chan, A. Wong, and Y. Wang, "Attribute Clustering for Grouping, Selection, and Classification of Gene Expression Data," in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2005, pp. 83-101.

[158] J. Mueller, F. von Eggeling, D. Driesch, J. Schubert, C. Melle, and K. Junker, "ProteinChip Technology Reveals Distinctive Protein Expression Profiles in the Urine of Bladder Cancer Patients," in *European Urology*, 2005, pp. 885-893.

[159] R. Czabanski, J. Jezewski, A. Matonia, and M. Jezewski, "Computerized analysis of fetal heart rate signals as the predictor of neonatal acidemia," in *Expert Systems with Applications*, 2012, pp. 11846-11860.

[160] W. Zhong, R. Chow, and J. He, "Clinical charge profiles prediction for patients diagnosed with chronic diseases using Multi-level Support Vector Machine," in *Expert Systems with Applications*, 2012, pp. 1474-1483.

[161] A. R. Razavi, H. Gill, H. Ahlfeldt, and N. Shahsavar, "Predicting Metastasis in Breast Cancer: Comparing a Decision Tree with Domain Experts," in *Journal of medical systems*, 2007, pp. 263-273.

[162] J. I. Titapiccolo, *et al.*, "Artificial intelligence models to stratify cardiovascular risk in incident hemodialysis patients," in *Expert Systems with Applications*, 2013, pp. 4679-4686.

[163] F. C. Mohd Hanafi Ahmad Hijazi, Yalin Zheng, "Data mining techniques for the screening of age-related macular degeneration," *Knowledge-Based Systems,* pp. 83-92, 2012.

[164] C. D. L. Jourdan, E.-G. Talbi, S. Gallina, "A data mining approach to discover genetic and environmental factors involved in multifactorial diseases," *Knowledge-Based Systems,* pp. 235-242, 2002.

[165] V. A. Maryam Vatankhah, Reza Fazel-Rezai, "Perceptual pain classification using ANFIS adapted RBF kernel support vector machine for therapeutic usage," *Applied Soft Computing,* pp. 2537-2546, 2013.

[166] V. P. P. George D Magoulas, Michael N Vrahatis, "Neural network-based colonoscopic diagnosis using on-line learning and differential evolution," *Applied Soft Computing,* pp. 369-379, 2004.

[167] T.-h. K. Aboul Ella Hassanien, "Breast cancer MRI diagnosis approach using support vector machine and pulse coupled neural networks," *Journal of Applied Logic,* pp. 277-284, 2012.

[168] R. C. J. B.M. Patil, Durga Toshniwal, "Hybrid prediction model for Type-2 diabetic patients," *Expert Systems with Applications,* pp. 8102-8108, 2010.

[169] Y. C. T. Bo Jin, Yan-Qing Zhang, "Support vector machines with genetic fuzzy feature transformation for biomedical data classification," *Information Sciences,* pp. 476-489, 2007.

[170] S. B. Ugur Bilge, Sedat Durmaz, "Application of data mining techniques for detecting asymptomatic carotid artery stenosis," *Computers & Electrical Engineering,* pp. 1499-1505, 2013.

[171] M. Šprogar, Lenič, M. & Alayon, S., "Evolution in Medical Decision Making," *Journal of Medical Systems,* 2002.

[172] M. Seera and C. P. Lim, "A hybrid intelligent system for medical data classification," *Expert Systems with Applications,* vol. 41, pp. 2239-2249, 2014/04/01/ 2014.

[173] D. A. Sharaf-El-Deen, I. F. Moawad, and M. E. Khalifa, "A New Hybrid Case-Based Reasoning Approach for Medical Diagnosis Systems," *Journal of Medical Systems,* vol. 38, p. 9, 2014/01/28 2014.

[174] C. W. J. Granger, "Combining Forecasts – Twenty Years Later," *Journal of Forecasting,* pp. 167-173, 1989.
[175] N.-E. E. Faouzi, "Combining predictive schemes in short-term traffic forecasting," in *Proceedings of the 17th International Symposium on Transportation and Traffic theory (ISTTT),*, Jerusalem, 1999, pp. 471-487.
[176] K. C. N. El-Sheimy, A. Noureldin, "The utilization of artificial neural networks for multi-sensor system integration in navigation and positioning instruments," in *IEEE Transactions on Instrumentation and Measurement*, 2006, pp. 1606-1615.
[177] A. N. R. Sharaf, A. Osman, N. El-Sheimy, "Implementation of online INS/GPS integration utilizing radial basis function neural network," *IEEE Aerospace and Electronic Systems Magazine,* pp. 8-14, 2005.
[178] R. L. A. Hiliuta, F. Gagnon, "Fuzzy corrections in a GPS/INS hybrid navigation system," in *IEEE Transactions on Aerospace and Electronic Systems*, 2004, pp. 591-600.
[179] P. C. Paris and F. Erdogan, "A critical analysis of crack propagation laws," *Journal of Basic Engineering,* vol. 85, p. 528, 1963.
[180] R. W. Kuehl, "Stability of thin film resistors - Prediction and differences base on time-dependent Arrhenius law," *Microelectronics Reliability,* vol. 49, pp. 51-58, 2009.
[181] D. B. Stringer, K. E. LaBerge, C. J. Burdick, and B. A. Fields, "Natural Fatigue Crack Initiation and Detection in High Quality Spur Gears," presented at the 68th American Helicopter Society Annual Forum, FT Worth, TX, 2012.
[182] C. M. Bishop, "Model-based machine learning," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* vol. 371, p. 20120222, 2013.
[183] K. Goebel, B. Saha, A. Saxena, J. R. Celaya, and J. P. Christophersen, "Prognostics in battery health management," *IEEE instrumentation & measurement magazine,* vol. 11, 2008.
[184] B. Saha and K. Goebel, "Uncertainty Management for Diagnostics and Prognostics of Batteries using Bayesian Techniques," in *Aerospace Conference, 2008 IEEE*, 2008, pp. 1-8.
[185] B. Saha, K. Goebel, S. Poll, and J. Christophersen, "Prognostics methods for battery health monitoring using a Bayesian framework," *IEEE Transactions on instrumentation and measurement,* vol. 58, pp. 291-296, 2009.
[186] Y. Song, D. Liu, L. Li, and Y. Peng, "Lithium-Ion Battery Pack On-Line Degradation State Prognosis Based on the Empirical Model," in *2018 International Conference on Sensing,Diagnostics, Prognostics, and Control (SDPC)*, 2018, pp. 402-407.
[187] M. Eider, N. Bodenschatz, A. Berl, P. Danner, and H. de Meer, "A Novel Approach on Battery Health Monitoring."
[188] V. Ramadesigan, P. W. C. Northrop, S. De, S. Santhanagopalan, R. D. Braatz, and V. R. Subramanian, "Modeling and simulation of lithium-ion batteries from a systems engineering perspective," *Journal of The Electrochemical Society,* vol. 159, pp. R31-R45, 2012.
[189] R. C. Team, "R: A language and environment for statistical computing," 2013.
[190] (2017, *ROS.org | Powering the world's robots*. Available: http://www.ros.org/
[191] (2018, *OpenModelica*. Available: https://www.openmodelica.org/
[192] C. Teubert, M. J. Daigle, S. Sankararaman, K. Goebel, and J. Watkins, "A Generic Software Architecture for Prognostics," *International Journal of Prognostics and Health Management,* vol. 8, p. 26, 2017.
[193] M. Daigle. (2016, *Prognostics Algorithm Library*. Available: https://github.com/nasa/prognosticsalgorithmlibrary
[194] T. E. Oliphant, *A Guide to NumPy* vol. 1: Trelgol Publishing USA, 2006.
[195] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open source scientific tools for Python," *http://www. scipy. org/*, 2001.

[196] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*: " O'Reilly Media, Inc.", 2012.
[197] F. Perez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science & Engineering,* vol. 9, pp. 21-29, 2007.
[198] F. Pedregosa, *et al.*, "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.
[199] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering,* vol. 9, pp. 90-95, 2007.
[200] P. Ramachandran and G. Varoquaux, "Mayavi: 3D visualization of scientific data," *Computing in Science & Engineering,* vol. 13, pp. 40-51, 2011.
[201] N. Ketkar, "Introduction to PyTorch," in *Deep Learning with Python*, ed: Springer, 2017, pp. 195-208.
[202] M. Dong and D. He, "A segmental hidden semi-Markov model (HSMM)-based diagnostics and prognostics framework and methodology," *Mechanical Systems and Signal Processing,* vol. 21, pp. 2248-2266, 2007/07/01/ 2007.
[203] M. Roemer, C. Byington, G. Kacprzynski, and G. Vachtsevanos, "An overview of selected prognostic technologies with reference to an integrated PHM architecture."
[204] T. Chen, T. He, and M. Benesty, "Xgboost: extreme gradient boosting," *R package version 0.4-2,* pp. 1-4, 2015.
[205] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks,* vol. 5, pp. 157-166, 1994.
[206] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist,* vol. 2, 2010.
[207] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine,* vol. 29, pp. 141-142, 2012.
[208] D. J. Newman, S. Hettich, C. L. Blake, C. J. Merz, and D. W. Aha, "UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA."
[209] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 and cifar-100 datasets," *URl: https://www. cs. toronto. edu/kriz/cifar. html (vi sited on Mar. 1, 2016),* 2009.
[210] PHM Society. (2018, *Public Data Sets | PHM Society*. Available: https://www.phmsociety.org/references/datasets
[211] Prognostics Center of Excellence. (2018, *Prognostics Center - Data Repository*. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/
[212] Committee on Toward an Open Science Enterprise Board on Research Data and Information Policy and Global Affairs National Academies of Sciences Engineering and Medicine, *Open Science by Design: Realizing a Vision for 21st Century Research*: The National Academic Press, 2018.
[213] A. Junghanns and T. Blochwitzh. (2018, *10 Years of FMI: Where are we now? Where do we go?* Available: https://www.modelica.org/events/modelica2018japan/presentation/10_Years_of_FMI.pdf
[214] T. Blochwitz, *et al.*, "The functional mockup interface for tool independent exchange of simulation models," in *the 8th International MODELICA Conference*, Dresden, Germany, 2011, pp. 105-114.
[215] T. Blochwitz, *et al.*, "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models," in *The 9th International MODELICA Conference*, Munich, Germany, 2012, pp. 173-184.
[216] E. Chrisofakis, A. Junghanns, C. Kehrer, and A. Rink, "Simulation-based development of automotive control software with Modelica," pp. 1-7.
[217] A. Himmler, A. Pillekeit, B. Loyer, S. Mouvand, and V. Dezobry, "Using FMI-and FPGA-Based Models for the Real-Time Simulation of Aircraft Systems," p. 0125.

[218] E. J. Tuegel, A. R. Ingraffea, T. G. Eason, and S. M. Spottswood, "Reengineering Aircraft Structural Life Prediction Using a Digital Twin %J International Journal of Aerospace Engineering," vol. 2011, p. 14, 2011.
[219] A. Cerrone, J. Hochhalter, G. Heber, and A. Ingraffea, "On the Effects of Modeling As-Manufactured Geometry: Toward Digital Twin %J International Journal of Aerospace Engineering," vol. 2014, p. 10, 2014.
[220] G. N. Schroeder, C. Steinmetz, C. E. Pereira, and D. B. J. I.-P. Espindola, "Digital twin data modeling with automationML and a communication methodology for data exchange," vol. 49, pp. 12-17, 2016.
[221] R. Magargle, *et al.*, "A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System," in *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, 2017, pp. 35-46.
[222] C. Li, S. Mahadevan, Y. Ling, S. Choze, and L. Wang, "Dynamic Bayesian Network for Aircraft Wing Health Monitoring Digital Twin," *AIAA Journal*, vol. 55, pp. 930-941, 2017/03/01 2017.
[223] S. P. A. J. J. o. I. M. Datta, "Emergence of Digital Twins-Is this the march of reason?," vol. 5, pp. 14-33, 2017.
[224] D. Iglesias, *et al.*, "Digital twin applications for the JET divertor," vol. 125, pp. 71-76, 2017.
[225] A. Parrott and L. Warshaw, "Industry 4.0 and the digital twin," ed, 2017.
[226] C. Verdouw and J. Kruize, "Digital twins in farm management: illustrations from the FIWARE accelerators SmartAgriFood and Fractals."
[227] M. Envision. (2017, August 27 2018). *A New Class of Digital Twin for Industry 4.0*. Available: https://www.youtube.com/watch?v=tnho3se1K6k
[228] G. Knapp, *et al.*, "Building blocks for a digital twin of additive manufacturing," *Acta Materialia*, vol. 135, pp. 390-399, 2017.
[229] M. Shafto, *et al.*, "NASA technology roadmap: DRAFT modeling, simulation, information technology & processing roadmap technology area 11," ed: Nov, 2010.
[230] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and US Air Force vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, 2012, p. 1818.
[231] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*: Forschungsunion, 2013.
[232] INCOSE. (2007, *Systems Engineering Vision 2020*. Available: https://www.sebokwiki.org/wiki/Model-Based_Systems_Engineering_(MBSE)_(glossary)
[233] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, *et al.*, Eds., ed Cham: Springer International Publishing, 2017, pp. 85-113.
[234] G. Digital, "Minds + Machines: Meet a Digital Twin," ed, 2016.
[235] M. Pecht, "Prognostics and health management of electronics," *Encyclopedia of Structural Health Monitoring*, 2009.
[236] C. Teubert, M. J. Daigle, S. Sankararaman, K. Goebel, and J. Watkins, "A Generic Software Architecture For Prognostics," 2017.
[237] K. Swearingen, W. Majkowski, B. Bruggeman, D. Gilbertson, J. Dunsdon, and B. Sykes, "An open system architecture for condition based maintenance overview," in *Aerospace Conference, 2007 IEEE*, 2007, pp. 1-8.
[238] J. D. Labrado, B. A. Erol, J. Ortiz, P. Benavidez, M. Jamshidi, and B. Champion, "Proposed testbed for the modeling and control of a system of autonomous vehicles," pp. 1-6.

[239] G. A. Pratt, "Is a Cambrian explosion coming for robotics?," *The Journal of Economic Perspectives,* vol. 29, pp. 51-60, 2015.
[240] J. Powers, "Apache Spark and ACHILLES," ed.
[241] T. Dinsmore, "Spark is the Future of Analytics," ed, 2017.
[242] "Python eats away at R: Top Software for Analytics, Data Science, Machine Learning in 2018: Trends and Analysis," ed: KDnuggets, 2018.

Appendix A: Available Signal List for the FMTV

J1587 Signals

EngRetSwtch_1587
PrkBrkSwtchStat_1587
IdleShutDwnTimerStatDriverAlertMode_1587
VehSpeedEng_1587
VehSpeedBrakes_1587
PTOCtlSwt_1587
AccelPedalPos_1587
EngLoad_1587
EngOilPres_1587
BoostPres_1587
BarPres_Eng_1587
EngCoolantTemp_1587
Cylinder2EngRetStat_1587
ATCStatusLamp_1587
TrRgSelec_1587
TrRgAttai_1587
InjCtrlPres_1587
BatPotVEng_1587
TransOilTemp_1587
FuelRate_1587
EngSpeed_1587
TransOptShaftSpeedTrans_1587
EngRetLvlSwtch_1587
ABS_BrkCtl_1587
ABS_RetCont_1587
ABS_OffRoadSwtch_1587
IdleShutDwnTimerStatEngShtDwn_1587
IdleShutDwnTimerStatTimerOverride_1587
IdleShutDwnTimerStatTimerFunc_1587
IdleShutDwnTimeStat_1587
RoadSpeedLimStat_1587
PTOSetSwitch_1587
PTOCoastSwtch_1587
PTOResSwtch_1587

J1587 Signals

PTOStatAccelSwtch_1587
PTOStatBrkSwtch_1587
PTOStatClutchSwtch_1587
PTOMode_1587
IntManfTemp_1587
Cylinder3EngRetStat_1587
Cylinder4EngRetStat_1587
Cylinder6EngRetStat_1587
Cylinder8EngRetStat_1587
EngRetarderStat_1587
ATCBrkCtl_1587
ATCEngCtl_1587
ATCspinOutDetection_1587
ATCsnowMudSwtch_1587
VDCBrkCtl_1587
VDCEngCtrl_1587
InstFuelEco_1587
Avg_FuelEco_1587
AirInletPres_1587
AirInletTemp_1587
AmbAirTemp_1587
EngCoolantLvl_1587
EngOilTemp_1587
EngRetPct_1587
ExtRngBarPres_1587
FuelTemp_1587
OutputTorq_1587
PTOSetSpeed_1587
TorqLimFact_1587
RtdEngSpeed_1587
RtdEngPwr_1587
TotIdlHours_1587
TotalIdleFuelUsed_1587
TotVehDist2
TotVehHrs_1587
TotEngHrs_1587
TotPTOHrs_1587
TotEngRevs_1587
TotalFuelUsed_1587

J1587 Signals

TripDist_1587

TripFuel_1587

J1939 Signals

RetEnableBrkAsstSwtchRetExhaust

RetEnableShftAsstSwtchRetExhaust

ABS_OffRoadSwtchStatTires

ASROffRoadSwtchTire

ASRHillHolderSwtchTire

TractionCtrlOverrideSwtch_Tires

TransDrivEng

TransTorqConvLockupEngaged

TrShInProg

TrOutShaftSp

EngPctLdAtCurSpd

EngPctTorq

PctEngLoad

EngSpeed

EngDmdPctTorq

TrSelGr

RelativeSpdFrontAxleLftWheel

RelativeSpdFrontAxleRghtWheel

RelativeSpdRearAxle1LftWheel

RelativeSpdRearAxle1RghtWheel

EngDesiredOpSpd

EngOilPres

PTOGovernorEnableSwtch

EngRemPTOGovernorPreProgSpdCtrlSwtch

EngRemPTOGovernorVarSpdCtrlSwtch

PTOGovernorSetSwtchEng

PTOGovernorCoastDecelSwtch

PTOGovernorResSwtch

PTOGovernorAccelSwtch

BrakeSwitch

ClutchSwtch

PTOGovernorState

InstFuelEco

J1939 Signals

Ave_FuelEco
IntManfTemp
BatPotVEng
ReqPctFanSpdEng
Act_RetardPctTorqExh
AccelPedalPos
FrontAxleSpd
InjCtlPres
NomFrict-PtcTorq
EngCoolantTemp
VehSpeedEng
FuelRate
BoostPres
TransOilTemp
PitchAngleMean
PitchAngleStdv
RollAngleMean
RollAngleStdv
altitude
heading
speed_mph
unsprungMax
unsprungMean
unsprungMin
unsprungStandardDev
TotPTOHrs
HiResTotVehDist
TotVehDist
TotEngHrs
TotEngRevs
TotalFuelUsed
EngRtdPwr
EngRtdSpeed

Appendix B: Available Bus Data for the MTVR

All J1708 Signals

Engine #1

Engine Oil Pressure (kPa)
Boost Pressure (kPa)
Intake Manifold Temperature (Degrees F)
Barometric Pressure (kPa)
Engine Coolant Temperature (Degrees F)
Engine Retarder Status (Binary Bit-Mapped)
Battery Potential (Voltage) (V)
Fuel Temperature (Degrees F)
Engine Oil Temperature (Degrees F)
Fuel Rate (L/s)
Instantaneous Fuel Economy (km/L)
Average Fuel Economy (km/L)
Power Takeoff Set Speed (rpm)
Engine Speed (rpm)
Transmitter System Diagnostic Code and Occurrence Count Table (Binary Bit-Mapped)
Transmitter System Status (IEEE 1-byte Unsigned Integer)
Total Vehicle Distance (km)
Parking Brake Switch (Binary Bit-Mapped)
Idle Shutdown Timer Status (Binary Bit-Mapped)
Road Speed Limit Status (Binary Bit-Mapped)
Road Speed (km/h)
Cruise Control Status (Binary Bit-Mapped)
Cruise Control Set Speed (IEEE 1-byte Unsigned Integer)
Power Takeoff Status (Binary Bit-Mapped)
Percent Accelerator Pedal Position (%)
Percent Engine Load (%)
Component-specific request (IEEE 1-byte Unsigned Integer)
Cruise Control Switches Status (Binary Bit-Mapped)
Request Parameter (IEEE 1-byte Unsigned Integer)

Transmission

Invalid Data Parameter (Unknown Data Type ID)
Transmission Range Selected (Alphanumeric)
Transmission Range Attained (Alphanumeric)
Transmission Output Shaft Speed (rpm)
Transmitter System Diagnostic Code and Occurrence Count Table (Binary Bit-Mapped)
Auxiliary Vacuum Pressure Reading (kPa)
Component-specific request (IEEE 1-byte Unsigned Integer)

Brakes, Power Unit

ATC Control Status (Binary Bit-Mapped)
Battery Potential (Voltage) (V)
Transmitter System Diagnostic Code and Occurrence Count Table (Binary Bit-Mapped)
ABS Control Status (Binary Bit-Mapped)
Road Speed (km/h)

Defined by SAE J1708

Data Link Escape (Variable)

All J1939 Signals

FMPT

FMPT Pedal Position

Actual Pedal Position (%)
Modified Pedal Position (%)
FMPT Status (Bitmap)

FMPT Parameter 1

Pedal Alpha Parameter (IEEE_Single)
Pedal Fast Alpha Parameter (IEEE_Single)

FMPT Parameter 2

FMPT Version (Numeric)
Idle threshold (%)
Clip Threshold (%)
Velocity Threshold (km/hr)
Boost Threshold (kPa)
FMPT Config (Bitmap)

Engine #1

Electronic Engine Controller 2

Road Speed Limit Status (Bitmap)
Accelerator Pedal 1 Low Idle Switch (Bitmap)
Accelerator Pedal Kickdown Switch (Bitmap)
Accelerator Pedal Position 1 (%)
Percent Load At Current Speed (%)
Remote Accelerator Pedal Position (%)

Electronic Engine Controller 1

Source Address of Controlling Device for Engine Control (Numeric)
Engine Starter Mode (Bitmap)
Engine Speed (rpm)
Engine Demand - Percent Torque (%)
Driver's Demand Engine (%)
Actual Engine - Percent Torque (%)
Engine Torque Mode (Bitmap)

Electronic Engine Controller 3

Nominal Friction (%)
Engine's Desired Operating Speed (rpm)
Engine's Desired Operating Speed Asymmetry Adjustment (Numeric)

Engine Temperature 1

Engine Coolant Temperature (deg C)
Engine Intercooler Thermostat Opening (%)
Fuel Temperature (deg C)
Engine Oil Temperature 1 (deg C)
Turbo Oil Temperature (deg C)
Engine Intercooler Temperature (deg C)

Engine Fluid Level/Pressure 1

Engine Oil Pressure (kPa)
Crankcase Pressure (kPa)
Coolant Pressure (kPa)
Coolant Level (%)
Extended Crankcase Blow-by Pressure (kPa)
Fuel Delivery Pressure (kPa)
Engine Oil Level (%)

Engine #1

Cruise Control/Vehicle Speed

Engine Shutdown Override Switch (Bitmap)
Cruise Control Pause Switch (Bitmap)
Cruise Control States (Bitmap)
Cruise Control Active (Bitmap)
Cruise Control Enable Switch (Bitmap)
Brake Switch (Bitmap)
Clutch Switch (Bitmap)
Cruise Control Set Switch (Bitmap)
Cruise Control Coast (Decelerate) Switch (Bitmap)
Cruise Control Resume Switch (Bitmap)
Cruise Control Accelerate Switch (Bitmap)
Two Speed Axle Switch (Bitmap)
Parking Brake Switch (Bitmap)
Wheel-Based Vehicle Speed (km/h)
Cruise Control Set Speed (km/h)
Engine Test Mode Switch (Bitmap)
Idle Decrement Switch (Bitmap)
Idle Increment Switch (Bitmap)
PTO State (Bitmap)

Fuel Economy (Liquid)

Fuel Rate (L/h)
Instantaneous Fuel Economy (km/kg)
Average Fuel Economy (km/kg)
Throttle Position (%)

Inlet/Exhaust Conditions 1

Boost Pressure (kPa)
Intake Manifold 1 Temperature (deg C)
Air Inlet Pressure (kPa)
Air Filter 1 Differential Pressure (kPa)
Coolant Filter Differential Pressure (kPa)
Exhaust Gas Temperature (deg C)
Particulate Trap Inlet Pressure (kPa)

Transmission #1

Electronic Transmission Controller 1

Source Address of Controlling Device for Transmission Control (Numeric)
Input Shaft Speed (rpm)
Output Shaft Speed (rpm)
Percent Clutch Slip (%)
Driveline Engaged (Bitmap)
Torque Converter Lockup Engaged (Bitmap)
Shift In Process (Bitmap)
Momentary Engine Overspeed Enable (Bitmap)
Progressive Shift Disable (Bitmap)

Electronic Transmission Controller 2

Transmission Requested Range (per byte)
Transmission Current Range (per byte)
Current Gear (Numeric)
Selected Gear (Numeric)
Actual Gear Ratio (Numeric)

Transmission Fluids

Clutch Pressure (kPa)
Transmission Oil Level (%)
Transmission Filter Differential Pressure (kPa)
Transmission Oil Pressure (kPa)
Transmission Oil Temperature (deg C)

Torque/Speed Control 1

Requested Torque/Torque Limit (%)
Override Control Mode (Bitmap)
Requested Speed Control Conditions (Bitmap)
Override Control Mode Priority (Bitmap)
Requested Speed/Speed Limit (rpm)

Brakes - System Controller

Electronic Brake Controller 1

EBS Brake Switch (Bitmap)
Traction Control Override Switch (Bitmap)
ABS Fully Operational (Bitmap)
ABS/EBS Amber Warning Signal (Powered Vehicle) (Bitmap)
EBS Red Warning Signal (Bitmap)
Source Address of Controlling Device for Brake Control (Numeric)
Tractor-Mounted Trailer ABS Warning Signal (Bitmap)
ATC/ASR Information Signal (Bitmap)
Trailer ABS Status (Bitmap)
Brake Pedal Position (%)
ASR Engine Control Active (Bitmap)
ASR Brake Control Active (Bitmap)
Anti-Lock Braking (ABS) Active (Bitmap)
ABS Off-road Switch (Bitmap)
ASR Off-road Switch (Bitmap)
ASR "Hill Holder" Switch (Bitmap)
Remote Accelerator Enable Switch (Bitmap)
Auxiliary Engine Shutdown Switch (Bitmap)
Engine Derate Switch (Bitmap)
Accelerator Interlock Switch (Bitmap)
Engine Retarder Selection (%)

Wheel Speed Information

Front Axle Speed (km/h)
Relative Speed; Front Axle, Left Wheel (km/h)
Relative Speed; Front Axle, Right Wheel (km/h)
Relative Speed; Rear Axle #1, Left Wheel (km/h)
Relative Speed; Rear Axle #1, Right Wheel (km/h)
Relative Speed; Rear Axle #2, Left Wheel (km/h)
Relative Speed; Rear Axle #2, Right Wheel (km/h)

Torque/Speed Control 1

Requested Torque/Torque Limit (%)
Override Control Mode (Bitmap)
Requested Speed Control Conditions (Bitmap)
Override Control Mode Priority (Bitmap)
Requested Speed/Speed Limit (rpm)

Retarder - Engine

Electronic Retarder Controller 1

Engine Coolant Load Increase (Bitmap)
Intended Retarder Percent Torque (%)
Source Address of Controlling Device for Retarder Control (Numeric)
Retarder Requesting Brake Light (Numeric)
Drivers Demand Retarder (%)
Retarder Selection, non-engine (%)
Actual Maximum Available Retarder (%)
Actual Retarder (%)
Retarder Enable - Brake Assist Switch (Bitmap)
Retarder Enable - Shift Assist Switch (Bitmap)
Retarder Torque Mode (Bitmap)

Retarder - Driveline

Retarder fluids

Hydraulic Retarder Pressure (kPa)
Hydraulic Retarder Oil Temperature (deg C)

Tire Pressure Controller

Electronic Brake Controller 1

EBS Brake Switch (Bitmap)
Traction Control Override Switch (Bitmap)
ABS Fully Operational (Bitmap)
ABS/EBS Amber Warning Signal (Powered Vehicle) (Bitmap)
EBS Red Warning Signal (Bitmap)
Source Address of Controlling Device for Brake Control (Numeric)
Tractor-Mounted Trailer ABS Warning Signal (Bitmap)
ATC/ASR Information Signal (Bitmap)
Trailer ABS Status (Bitmap)
Brake Pedal Position (%)
ASR Engine Control Active (Bitmap)
ASR Brake Control Active (Bitmap)
Anti-Lock Braking (ABS) Active (Bitmap)
ABS Off-road Switch (Bitmap)
ASR Off-road Switch (Bitmap)
ASR "Hill Holder" Switch (Bitmap)
Remote Accelerator Enable Switch (Bitmap)
Auxiliary Engine Shutdown Switch (Bitmap)
Engine Derate Switch (Bitmap)
Accelerator Interlock Switch (Bitmap)
Engine Retarder Selection (%)

Appendix C: Available Signal Data for the PLS

Main Propulsion Diesel Engine (MPDE) Signals

Temperature Signals

'CFW CLR 1A&1B OUT TEMP',
'1A THRUST BRG TEMP',
'1A RKR LO SPLY TEMP',
'1A R T/C OIL TEMP',
'1A R T/C EXHAUST TEMP',
'1A R BANK EXHAUST TEMP',
'1A R BANK AIR TEMP',
'1A LO INLET TEMP',
'1A LO CLR OUT TEMP',
'1A LO CLR IN TEMP',
'1A L T/C OIL TEMP',
'1A L T/C EXHAUST TEMP',
'1A L BANK EXHAUST TEMP',
'1A L BANK AIR TEMP',
'1A JW OUT TEMP',
'1A JW IN TEMP',
'1A JW CLR OUT TEMP',
'1A CYL 9 EXHAUST TEMP',
'1A CYL 8 EXHAUST TEMP',
'1A CYL 7 EXHAUST TEMP',
'1A CYL 6 EXHAUST TEMP',
'1A CYL 5 EXHAUST TEMP',
'1A CYL 4 EXHAUST TEMP',
'1A CYL 3 EXHAUST TEMP',
'1A CYL 2 EXHAUST TEMP',
'1A CYL 16 EXHAUST TEMP',
'1A CYL 15 EXHAUST TEMP',
'1A CYL 14 EXHAUST TEMP',
'1A CYL 13 EXHAUST TEMP',
'1A CYL 12 EXHAUST TEMP',
'1A CYL 11 EXHAUST TEMP',
'1A CYL 10 EXHAUST TEMP',
'1A CYL 1 EXHAUST TEMP',
'1A CFW RT CAC OUT TEMP',
'1A CFW LFT CAC OUT TEMP',
'1A BRG 9 TEMP',
'1A BRG 8 TEMP',
'1A BRG 7 TEMP',
'1A BRG 6 TEMP',
'1A BRG 5 TEMP',

Temperature Signals

'1A BRG 4 TEMP',
'1A BRG 3 TEMP',
'1A BRG 2 TEMP',
'1A BRG 1 TEMP',
'1A RKR LO TEMP',
'1A COMB EXH TEMP',
'1A CFW CLR OUTLET TEMP'

Pressure Signals

'1A/1B CTRL AIR PRES',
'1A RKR LO SPLY PRES',
'1A R T/C AIR OUT PRES',
'1A R BANK AIR PRES',
'1A LOSP 2 DISCH PRES',
'1A LOSP 1 DISCH PRES',
'1A LO HEADER PRES',
'1A L T/C AIR OUT PRES',
'1A L BANK AIR PRES',
'1A JW OUTLET PRES',
'1A JW INLET PRESS',
'1A FO MANIFOLD PRES',
'1A EXH BACK PRES',
'1A CRKC PRES',
'1A RKR LO PMP DIS PRES', '1A FO PMP DISCH PRES',
'1A CFW PMP DISCH PRES'

Other Signals

'1A RKR LO SUMP LVL',
'1A RKR LO STR DP',
'1A R T/C SPEED',
'1A LO SUMP LVL',
'1A LO CLR OIL DP',
'1A L T/C SPEED',
'1A FUEL SPLY FLOW',
'1A FUEL RTN FLOW',
'1A FUEL RACK',
'1A ENGINE SPEED',
'1A AIR INTK FLTR DP',
'1A LO STRAINER DP',
'1A LO FILTER DP',
'1A JW EXPANSION TNK LVL',
'1A FO FILTER DP'

Main Reduction Gear (MRG) signals

Temperature Signals

'MRG1 THR BRG OIL TEMP',
'MRG1 THR BRG ASTN TEMP',
'MRG1 THR BRG AHEAD TEMP',
'MRG1 PN FWD SBD BG TEMP',
'MRG1 PN FWD PRT BG TEMP',
'MRG1 PN AFT SBD BG TEMP',
'MRG1 PN AFT PRT BG TEMP',
'MRG1 LS GR FWD BRG TEMP',
'MRG1 LS GR AFT BRG TEMP',
'MRG1 LO SUMP TEMP',
'MRG1 LO HDR TEMP',
'MRG1 LO CLR OUT TEMP',
'MRG1 LO CLR FW OUT TEMP'

Pressure Signals

'MRG1 LOSP DISCH PRES',
'MRG1 LO PMP B DISC PRES',
'MRG1 LO PMP A DISC PRES',
'MRG1 LO HDR PRES',
'MRG1 HMRB PT2 PRES',
'MRG1 HMRB PT1 PRES',
'MRG1 CLUTCH AIR PRES'

Other Signals

'MRG1 TN GEAR FWD BRG T',
'MRG1 TN GEAR AFT BRG T',
'MRG1 SEC TN GEAR ENGGED',
'MRG1 SEC TN GEAR DISENG',
'MRG1 PRI TN GEAR ENGGED',
'MRG1 PRI TN GEAR DISENG',
'MRG1 LO SUMP LVL',
'MRG1 LO STR DP'

Ship Service Diesel Generator (SSDG) signals

Temperature Signals

'SG1 PHASE C STATOR TEMP',
'SG1 PHASE B STATOR TEMP',
'SG1 PHASE A STATOR TEMP',
'SG1 FWD BRG TEMP',
'SG1 AIR COOLER OUT TEMP',
'SG1 AIR CLR INLET TEMP',
'SG1 AFT BRG TEMP',
'DG1 T/C EXHAUST TEMP',
'DG1 SW OVERBOARD TEMP',
'DG1 SW INJECTION TEMP',
'DG1 LO TO COOLER TEMP',
'DG1 LO TEMP TO ENG',
'DG1 LO TEMP FROM ENGINE',
'DG1 JW TEMP FROM ENG',
'DG1 INTK AIR MANF TEMP',
'DG1 FUEL SPLY TEMP',
'DG1 FUEL RTN TEMP',
'DG1 EXH MANF TEMP',
'DG1 CYL 8 EXHAUST TEMP',
'DG1 CYL 7 EXHAUST TEMP',
'DG1 CYL 6 EXHAUST TEMP',
'DG1 CYL 5 EXHAUST TEMP',
'DG1 CYL 4 EXHAUST TEMP',
'DG1 CYL 3 EXHAUST TEMP',
'DG1 CYL 2 EXHAUST TEMP',
'DG1 CYL 1 EXHAUST TEMP',
'DG1 COMBINED STACK TEMP',
'DG1 JW AFTCLR INLET TMP'

Pressure Signals

'SG1 GEN LO PRES',
'DG1 SW PRES',
'DG1 SW PMP DISCH PRES',
'DG1 LO PMP DISCH PRES',
'DG1 LO HEADER PRES',
'DG1 LO FLTR DIFF PRES',
'DG1 JW PUMP PRES',
'DG1 INTK AIR MANF PRES',
'DG1 FUEL PRES',
'DG1 EXH BACK PRES'

Other Signals

'SG1 VOLTAGE',
'SG1 POWER',
'SG1 PHASE C CURRENT',
'SG1 PHASE B CURRENT',
'SG1 PHASE A CURRENT',
'SG1 FREQUENCY',
'DG1 T/C SPEED',
'DG1 JW EXP TK LVL',
'DG1 GENERATOR OIL LEVEL',
'DG1 FUEL RACK POSITION',
'DG1 FO FILTER DP',
'DG1 ENG SPEED',
'DG1 CRKC VACUUM',
'DG1 AFTCLR PMP DISCH P'