Prognostic Health Management Using Semi-Supervised Machine Learning

Anastacia MacAllister, Jordan Belknap,
Danielle Clement, Stephen Summers, George Hellstern

Lockheed Martin Corporation

Fort Worth, TX

anastacia.m.macallister@lmco.com, jordan.l.belknap@lmco.com, danielle.m.clement@lmco.com,
stephen.t.summers@lmco.com, george.hellstern@lmco.com

ABSTRACT

A report by McKinsey & Company indicates that by the year 2025 machine learning (ML) based predictive maintenance could save organizations $630 billion a year. The benefits associated with ML driven maintenance is not lost on the U.S. military. They are expected to spend billions to develop such systems. While ML does show incredible promise, applying it to military problems can be challenging. ML approaches popular in industries such as web services can be data hungry, requiring millions of well-balanced data points to be successful. The military and the defense industry, however, face many challenges and often do not have pristine data sets. As a result, ML algorithms must be carefully selected to effectively deal with a data set's limitations. This work describes the development of a ML model that identifies performance anomalies in an aircraft subsystem. The subsystem is responsible for power and thermal management of on-board equipment. While anomalies in the system are rare, it is important to identify them early, but due to the subsystem's complexity, finding anomalies is challenging for a human. This made building a ML model difficult since there were few examples of anomalies and little was known about how anomalies presented themselves. To combat the limited data problem a semi-supervised ML algorithm based on Self-Organizing Maps (SOMs) was used to cluster known anomalies. Using the SOM clustering method, uncategorized examples that fell into clusters with high numbers of known anomalies were categorized as anomalous themselves. Testing results show the SOM based classifier can detect anomalous subsystem behavior with over 90% accuracy. However, analysis also shows that results are sensitive to hyperparameter changes, indicating that careful tuning is required based on the ML model use case.

ABOUT THE AUTHORS

Anastacia MacAllister, Ph.D., is a Machine Learning Researcher at Lockheed Martin Skunk Works. Her research focuses on developing machine learning algorithms using sparse or imbalanced data sets. Dr. MacAllister is currently working on developing machine learning methods for prognostic health management. She is also developing machine learning tools to help reduce pilot's cognitive load. Dr. MacAllister received her Ph.D. from Iowa State University of Science and Technology in Mechanical Engineering and Human-Computer Interaction.

Jordan Belknap is an Artificial Intelligence Researcher at Lockheed Martin Skunk Works. She is currently the Principal Investigator of AI Based Prognostic Health Management. She specializes in developing AI solutions for highly complex systems with noisy, partially labeled data. She received her Masters' from Georgia Institute of Technology in Computer Science with a focus in Interactive Intelligence.

Danielle Clement, Ph.D., is a Technical Fellow at Lockheed Martin Skunk Works. She is currently researching reinforcement learning and game theoretic techniques applied to dynamic multi-agent collaborative and competitive environments. Dr. Clement received her Ph.D. from the University of Texas at Arlington in Computer Science Engineering.

George Hellstern is a Program Manager at Lockheed Martin Skunk Works. Mr. Hellstern has over 18 years of experience developing Battle Management applications at LM Aero. He has served as a Program Manager on several CRAD and IR&D program in addition to leading several development teams while at Skunk Works. Mr. Hellstern is responsible for Integrating an Alternative Speech Recognition package onto the F-35 Block 2B. He designed and developed analysis software suites for use in the JSF Virtual Simulation environment, managing computer generated forces and providing a white controller capability for both Red and Blue simulation entities.

Stephen Summers is a Systems/Software Engineer at Lockheed Martin Aeronautics Company on the Power and Thermal Management System (PTMS). He works with production and field support for system hardware failures looking for trends and undetected failures. Stephen received a B.S. degree in Computer Engineering from Texas A&M University and a M.S. degree in Electrical Engineering from University of Texas at Arlington.

Prognostic Health Management Using Semi-Supervised Machine Learning

Anastacia MacAllister, Jordan Belknap,
Danielle Clement, Stephen Summers, George Hellstern

Lockheed Martin Corporation

Fort Worth, TX

anastacia.m.macallister@lmco.com, jordan.l.belknap@lmco.com, danielle.m.clement@lmco.com, stephen.t.summers@lmco.com, george.hellstern@lmco.com

INTRODUCTION

New technologies and capabilities are being rapidly integrated into battlefield technology, providing warfighters with the tools necessary to address 21st century challenges and threats. While this increase in capability can provide immense benefit to the warfighter, it also adds complexity (Drezner, 2009; Grammich, Arena, Younossi, Brancato, & Blickstein, 2008). This complexity can introduce numerous engineering and human factors challenges that need to be addressed to ensure the equipment performs as intended (Iriarte, 2018; Hawley & Swehla, 2018). Specifically, in the engineering field new technology often requires increased maintenance monitoring and highly skilled maintainers. Previously, traditional one size fits all maintenance monitoring and scheduling was driven by engineering models of degradation and skilled operator observations. This conservative strategy was moderately effective when little information was recorded about aircraft operations. However, this strategy occasionally led to unnecessary replacement of equipment and to extensive equipment downtime when an unexpected failure occurred. To combat these issues, manufacturers are beginning to test and implement increasingly affordable commodity sensors to add health reporting capabilities to new equipment. The sensors capture performance and operation data that can then be leveraged to predict when a piece of equipment is performing sub-optimally or predict when a part failure may occur. Ultimately, the goal is to be prognostic rather than reactive when conducting maintenance. However, the amount of sensor data being generated is outpacing a human's ability to analyze it. In addition, even with smaller amounts of data, many times humans are not able to identify higher order interactions between recorded variables. As a result, events might go unnoticed and propagate into larger issues, meaning the sensors are not being fully utilized.

To combat this flood of data, organizations are turning to machine learning (ML) and artificial intelligence (AI) (Bean, 2018; Anshuk Gandh, Carmen Magar, 2013; How real-time data is transforming five industries, 2019; Libert & Beck, 2018). Reports by the consulting company Accenture predicted that ML and AI have the potential to become a 14 trillion-dollar industry by 2035 (Purdy & Daugherty, How AI Boosts Industry Profits and Innovation, 2017). The Department of Defense alone plans to spend two billion dollars over the next five years to develop and deploy ML and AI methods (Fryer-Biggs, 2018). One of the first areas the DOD anticipates ML and AI methods to make an impact for the military is in the prognostic health management of equipment.

While technology companies like Google, Amazon, and Microsoft have had great success applying machine learning to their domain (Columbus, 2019; Kobielus, 2019; Spencer, 2019; Wiggers, 2019), the military domain provides unique challenges when developing machine learning systems for prognostics. Specifically, tech companies often deal with consumer data that is large scale and simple to collect. They can collect data sets on consumer behavior that have millions or billions of data points, all which are labeled with the behavior or event being recorded. This large-scale dataset with corresponding labels is the gold standard for machine learning. Unfortunately, outside of the tech industry many applications present unique data collection challenges. For example, military equipment may only fail occasionally providing data on a much smaller scale. Due to these data limitations, constructing machine learning algorithms can be problematic.

While challenging, developing ML for these types of applications is not impossible. Often, however, these types of data sets require careful selection and tuning of ML algorithms to ensure accurate results. This paper explores the creation of an automated ML anomaly detection system based on Self-Organizing Maps (SOMs) that can identify problems within an aircraft subsystem. The subsystem studied was a power thermal management system (PTMS) on

a new aircraft variant (Martin, et al., 2018). This system occasionally exhibited anomalous behavior, which is challenging to detect for maintainers because of system complexity and the volume of data to analyze. Due to this challenge, very little anomalous labeled data was available for ML model creation. This resulted in a highly unbalanced data set between normal and anomalous data, making the problem very challenging for more traditional machine learning approaches. As a result, careful ML model construction was required to address these limitations. This paper will detail the construction and testing of a ML classifier that was created to detect these anomalous events in the PTMS subsystem.

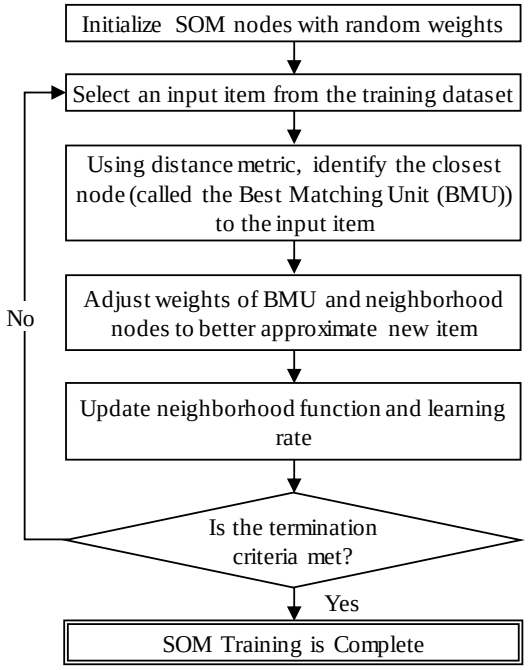

Figure 1. SOM Training Process

BACKGROUND

When constructing a machine learning classifier, characteristics of the available data can impact the approach used. Machine learning approaches can normally be categorized as supervised, unsupervised or semi-supervised. Supervised approaches use input and output pairs to generate a model which maps inputs to outputs, while unsupervised techniques group items by similarity and do not require knowledge of outputs. To generate a model with supervised learning techniques, a large and relatively balanced data set is often required. For example, a supervised learning technique looking to discriminate between normal and abnormal samples would require many labelled examples of both normal and abnormal readings. Since unsupervised techniques do not require these labels, they are frequently employed to gain better understanding of unfamiliar and unbalanced data sets (Russell & Norvig, 2016). Semi-supervised learning occupies the middle ground between supervised and unsupervised learning. In semi-supervised learning, some of the input data does have supporting labels, or output mappings. Semi-supervised techniques use this label information to either (a) apply the given labels to the unlabeled input data as in supervised approaches or (b) serve as limits or constraints on an unsupervised learning process (Chapelle, Schölkopf, & Zien, 2006). The dataset discussed in this work is comprised of sensor readings that are predominantly assumed to be of non-anomalous data, with limited examples of labeled anomalies. This limited amount of labeled data and highly unbalanced data set suggests a semi-supervised approach such as Self-Organizing Maps, discussed in the next section.

Self-Organizing Maps (SOMs)

Understanding key properties of the dataset can help drive machine learning model selection. Due to the properties of the PTMS data set, a semi-supervised classification approach was derived from unsupervised SOMs. Self-organizing maps, introduced by Kohonen in the early 1980s (Kohonen, 2012), are an unsupervised learning technique. They cluster similarly featured data together such that data items that are close together in the input space are also close together in the lower-dimensional output space. They are used extensively in anomaly detection because of their strengths in clustering, feature identification, data exploration, and visualization. When SOMS are used to cluster data, the resulting clusters can be labeled with known information, supporting semi-supervised techniques. This semi-supervised labeling of the unsupervised results is often called the contextual phase. SOMs approximate a multi-dimensional data set into a two-dimensional grid. SOMs are comprised of a set of nodes, also called neurons, that are defined by a typically two-dimensional index and a weight vector, which has the same number of elements as an input data item.

Figure 1 illustrates the training process for SOMs. SOMs are typically initialized with random weights. As the SOM is trained, the weights of each node are adjusted to approximate the provided input information. For each input item, the SOM node whose weight is the closest to the input item is identified. This node is called the Best Matching Unit (BMU). Once the BMU is found, the weights of the BMU and the nodes in the neighborhood of the BMU are updated to better approximate the new data. After the weights of the nodes are updated, the neighborhood function and the

learning rate, which are parameters that guide the learning process, are updated. This training process is repeated over the set of inputs until some termination criteria is met (typically number of inputs seen, minimization of error, or convergence properties met).

More formally, SOMs can be defined as a mapping from an input vector $\vec{x} = [x_1, x_2, \dots, x_m]$ to a node i with a weight $\overrightarrow{w_i} = [w_1, w_2, \dots, w_m]$. During training, weights are updated per equation (1), where $\overrightarrow{w_i}(t+1)$ is the updated weight for node i after processing input $\vec{x}$ in training timestep t. We define $d(\vec{x}, \vec{y})$ to be the distance between vector $\vec{x}$ and vector $\vec{y}$. Euclidean distance is frequently employed as the distance function.

$$\overrightarrow{\boldsymbol{w_i}}(\boldsymbol{t}+\mathbf{1}) = \overrightarrow{\boldsymbol{w_i}}(\boldsymbol{t}) + \boldsymbol{\Theta}(\boldsymbol{t}) * \boldsymbol{\sigma}(\boldsymbol{t}) * \boldsymbol{d}(\vec{\boldsymbol{x}}, \overrightarrow{\boldsymbol{w_i}}) \qquad (1)$$

As reflected in equation (1), the update is based on a gaussian neighborhood decay rate $\Theta(t)$, defined in equation (2), as well as a learning rate $\sigma(t)$, defined in equation (3). In early stages of training, the neighborhood around the BMU might be very large, but as the map grows more mature, the neighborhood shrinks. This means that early updates to the map have a more global impact, while later updates act on a more local level. The learning rate defines the impact of the changes. In this way, the neighborhood defines the scope of the changes based on a given input and the learning rate defines the scale of those changes. Often the initial values and bounds of these parameters, such as neighborhood decay rate and learning rate, that result in an optimally trained SOM are not known beforehand. These parameters, frequently referred to as hyperparameters, are usually computed through experimentation and testing.

$$\boldsymbol{\Theta}(\boldsymbol{t}) = \boldsymbol{e}^{-\frac{\boldsymbol{d}(\vec{\boldsymbol{x}}, \overrightarrow{\boldsymbol{w_i}})^2}{2\boldsymbol{\sigma}(\boldsymbol{t})^2}} \qquad (2)$$

$$\boldsymbol{\sigma}(\boldsymbol{t}) = \boldsymbol{\sigma_0} \boldsymbol{e}^{-\frac{\boldsymbol{t}}{\boldsymbol{\lambda}}} \qquad (3)$$

Once trained, SOMs are frequently employed to assist humans in visualizing complex data sets. Methods of visualization include U-matrix, P-matrix, and U*-matrix visualizations (Ultsch & Vetter, 1995). The U-matrix, which stands for unified distance matrix, typically defines the SOM as a grid of square or hexagonal cells, each representing a node. The distance between a node and its neighbors is encoded in the U-matrix as a height value, which is commonly displayed as a grayscale or color encoding on the map. A similar visualization is shown in Figure 2, where the individual neighbor distances from node to node are encoded. Representations such as these allow analysts to quickly identify clusters or regions of interest in the data set.

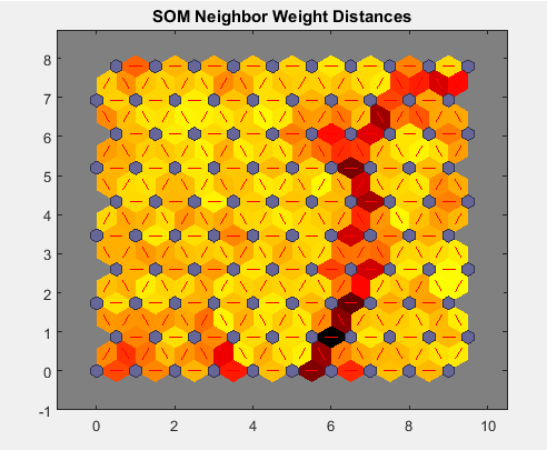

Figure 2. Example SOM Visualization

Self-Organizing Maps for Anomaly Detection

The work presented uses SOMs for early anomaly detection, which is critical to prognostic health management. Anomaly detection is the identification of abnormal or unexpected behavior or outputs, in comparison with some baseline of expected behavior or outputs. Supervised learning techniques can be limited when applied to anomaly detection, as identifying which data is normal to assign labels can be a challenge. Unsupervised techniques are frequently used for anomaly detection, as they do not necessarily require a pre-established identification of normal.

When SOMs are used for anomaly detection, they are typically applied in one of two ways. The first approach is to use only normal data in the creation of the SOM. When new input data is assessed, if the distance to the BMU for that point is above a pre-set threshold, then the data does not fit the normal patterns, and is classified as abnormal. This technique has been applied to anomalies in Unix (Hoglund, Hatonen, & Sorvari, 2000), network traffic (González & Dasgupta, 2002), intrusion detection (Greensmith, Feyereisl, & Aickelin, 2009), system health monitoring (Tan, Liu, Yang, Wei, & Qiu, 2018), fuel consumption under rationing (Aquize, Emery, Buarque, & Neto), engine health monitoring (Vanem & Brandsater, 2019), mining machines (Fan, Chiong, Hu, & Lin, 2018) and aircraft anomaly data

(Megatroika, Galinium, Mahendra, & Ruseno, 2016). This approach is limited in the same way that supervised techniques can be limited: in order to train solely on the normal data, anomalous data needs to be identified and removed from the dataset. In the subsystem dataset discussed here, some anomalous data can be identified, but there is a potential that undetected anomalies exist.

An alternative approach to applying SOMs is to use a map to cluster all the data (normal and abnormal samples). Then, the nodes which contain abnormal inputs can be identified. Future input data that best matches to an abnormal node is classified as abnormal. Langin, et al. (2009) applied this process to botnet detection, and it has also been applied to intrusion detection (Megatroika, Galinium, Mahendra, & Ruseno, 2016; Sarasamma, Zhu, & Huff, 2005; Langin, et al., 2009) and insider threat applications (Le & Zincir-Heywood, 2018). This approach supports noisy data with potentially unknown anomalies, such as the data set used for this research. As a result, for the work presented this approach was employed and described below.

METHODOLOGY

There are multiple challenges which must be addressed when constructing and deploying ML algorithms. The first is data. A ML classifier's accuracy is highly dependent on the quality of the training data. Therefore, clean quality data must exist to generate a robust ML model. Once the data is collected, it then must be formatted properly to be ingested by the algorithm for training. This ingestion algorithm must also be thoughtfully selected to match the characteristics of the available data. Only after the completion of these collection, pre-processing, and model selection steps can a ML model be trained and tested. This section describes these steps in detail then discusses the training and testing methodology employed.

Data Collection and Pre-Processing

To build a quality data set for ML model creation, flight test data was pulled off the aircraft post flight. From here the subsystem subject matter expert (SME) then converted data to readable format, filtered out irrelevant features, and time synced the data. Features of the final data set included temperatures, pressures, derivatives, and set points determined by deterministic models. This preprocessing was an important set, since it ensured that the ML model could have access to clean data that accurately represented the subsystem behavior the SME was trying to classify.

The data set consisted of 120,000 flight hours, shared between 342 tail numbers. Amongst those 342 tail numbers, 2 aircraft each exhibited a unique identified anomaly. Both anomalies specifically dealt with air pressure leakage during two different subsystem configurations. The remaining data set was unlabeled and contained several types of data including: aircraft performing at baseline, performing with minor deviations from baseline, and potentially unidentified anomalies. These characteristics mean the data set had a significant amount of noise. There were 3,594 files, 543 of which contained known anomalous behavior. Each data file had a frequency of 1 Hz, meaning there was a data entry for every second of the flight run, and for every data entry there were 197 variables. With a frequency of 1 Hz, 120,000 hours of flight time, and 197 variables, this resulted in 1.4 billion data points. This amount of data is impossible for a human to examine manually and training an ML algorithm on a fraction of this data set far exceeds the processing power of any standard machine.

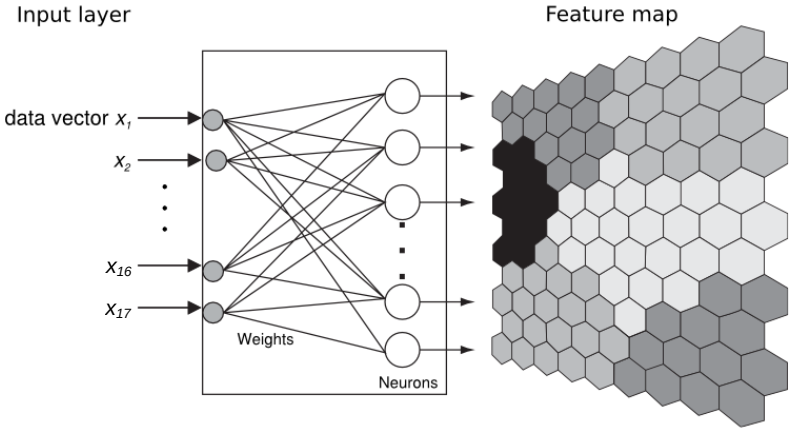

Figure 3: SOM Feature Vector

Reprinted from the AI for 5G: Research Directions and Paradigms, by the Science China Information Sciences, 2018, Retrieved from https://arxiv.org/pdf/1807.08671.pdf

Feature Vector Creation

Due to the large number of variables in the data set a down selection was required. Down selecting ensures that a ML classifier is only being trained on the most relevant variables in the data set, reducing the chances of the model picking up on unwanted trends. The variables used to build a ML classifier are often referred to as the feature vector. The feature vector is a list of features, or variables, whose values are considered by the ML algorithm for classification. As shown in Figure 3, the feature vector is fed into the ML algorithm, and the algorithm generates a model based on the information provided by the feature vector. This means that the robustness and accuracy of the model depends on the appropriate selection of features.

Multiple methods for downsizing a feature vector exist. At a fundamental level, all of these techniques fall in the realm of dimensionality reduction- a technique used to identify features of high value and discard features of low value. Two of the most common dimensionality reduction algorithms are Principal Components Analysis (PCA) and Independent Components Analysis (ICA). However, while both algorithms can produce viable results, the resulting features are not human readable (Lever, 2017; Hyvarinen, 1999). For this application it was important to the SME that they be able to interpret features used by the ML model. To address the human readability problem, the authors chose to have the dimensionality reduction driven by engineering design and subsystem expertise. To start the down selection, derivative values were discarded due to repetition in captured information. Then the subsystem engineer identified salient features that could describe a low-pressure anomalous event. These sensors and values impacted by a low-pressure events were identified as potentially high value indicators of system behavior that the ML model should be provided with. At the culmination of the down selection process, 17 features consisting of pressure and temperature readings throughout the subsystem were selected to be part of the feature vector.

Machine Learning Model Selection

After data collection and feature vector creation the next step was to choose a ML algorithm. Machine learning algorithms often fall into three separate categories: supervised, semi-supervised, and unsupervised. Each category of algorithm performs well with specific types of data, which means that algorithm selection is driven by data characteristics. Supervised learning requires fully labeled, evenly distributed data. Semi-supervised requires partially labeled data and can deal with unbalanced data. Unsupervised requires no labels and can also deal with unbalanced data. Due to the characteristics of the PTMS data -partially labeled and unbalanced - supervised algorithms were not a viable option for anomaly detection. Semi-supervised and unsupervised learning algorithms would both perform well with the data. However, to leverage the information about known anomalies, a semi-supervised classifier was the ideal option and was selected for the work.

Within the realm of semi-supervised learning, one-class support vector machines (SVMs), kmeans, and self-organizing maps (SOMs) are frequently used methods of anomaly detection. SVMs are traditionally a supervised learning method but can be adapted to make use of limited known information. However, they exhibit extreme sensitivity to hyperparameter selection, and adequate hyperparameter selection has proven to be challenging (Hsu & Chang, 2003). For these reasons, SVMs were not chosen for initial method exploration. Kmeans is also a commonly used clustering technique and is quick to implement. It is a proven method for anomaly detection and allows for use

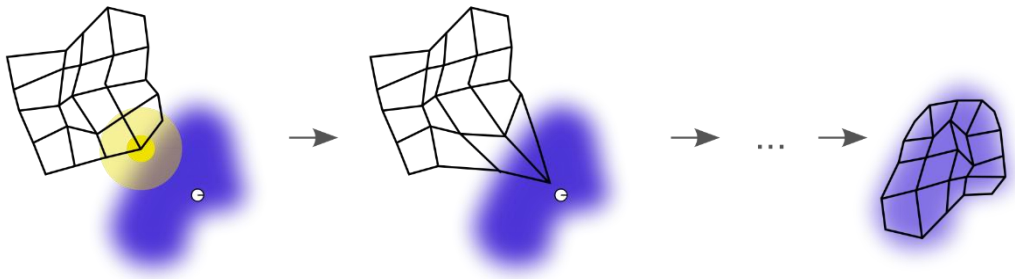

Figure 4: SOM Training

Reprinted from Wikimedia Commons, by Mcld, 2010, Retrieved from https://commons.wikimedia.org/wiki/File:Somtraining.svg

of limited known information. The weaknesses of kmeans are that the algorithm is very sensitive to initial hyperparameter values, namely the number of clusters and their initialization locations (Chen, Qin, Liu, Liu, & Li, 2010). Kmeans assigns data points to randomly initialized clusters based on which cluster is closest in n-dimensional space. However, the updates to each cluster have no effect on neighbors; each cluster independently shifts and grows. This means that reaching a global optimum is challenging (Ng & Han, 1992). Self-organizing maps (SOMs), perform similarly to kmeans with some notable differences. It is also a clustering algorithm, and randomly initializes locations of clusters. However, the method is less sensitive to initial hyperparameters. As the algorithm trains on a growing data set, the location of each cluster is updated based upon the data points assigned to it. Early in the training cycle, newly assigned data points have a significant effect on the location of the cluster. As training continues, the effect of a newly assigned data point lessens, resulting in fine tuning of cluster shape and location, as shown in Figure 4. This molding of the clusters to match the data means that SOMs outperform kmeans when finding the global optimum. Additionally, SOMs are more suitable for visualization of the data- providing clear insight into cluster behavior, which allows users to identify what makes an anomaly an anomaly (Chen, Qin, Liu, Liu, & Li, 2010). This information has the potential to inform engineering experts about behaviors and anomalies previously unidentified. Resulting in a decrease in data noise.

Performing this review indicated that SOMs were the best fit for the problem space and data. Robustness to ambiguities of error identification and steady state performance, handling unbalanced data, and the possibility of gleaning new information about the data set made it a promising method.

Constructing and Training the SOM

After model selection was completed and the feature vector selected, the next step was the SOM training process. In order to train the SOM, first a number of training hyperparameters needed to be set. Training hyperparameters for SOMs often include setting the size of the map, setting the learning rate, neighborhood decay rate, and the number of training iterations. The optimal parameters often depend on the data set characteristics and are often determined through experimentation and analysis of results. Once the training parameters were decided upon the SOM training process began using what is called the training data set. For this work, a subset of approximately 3,500 flight run files were used as the training data set for SOMs. This training data was pulled at random, with the remaining portion set aside to use for testing and validation of the trained classifier. The training data set serves to establish the behavior of the SOM by allowing it to cluster “like” data points and change the weights of each node to match the training data characteristics.

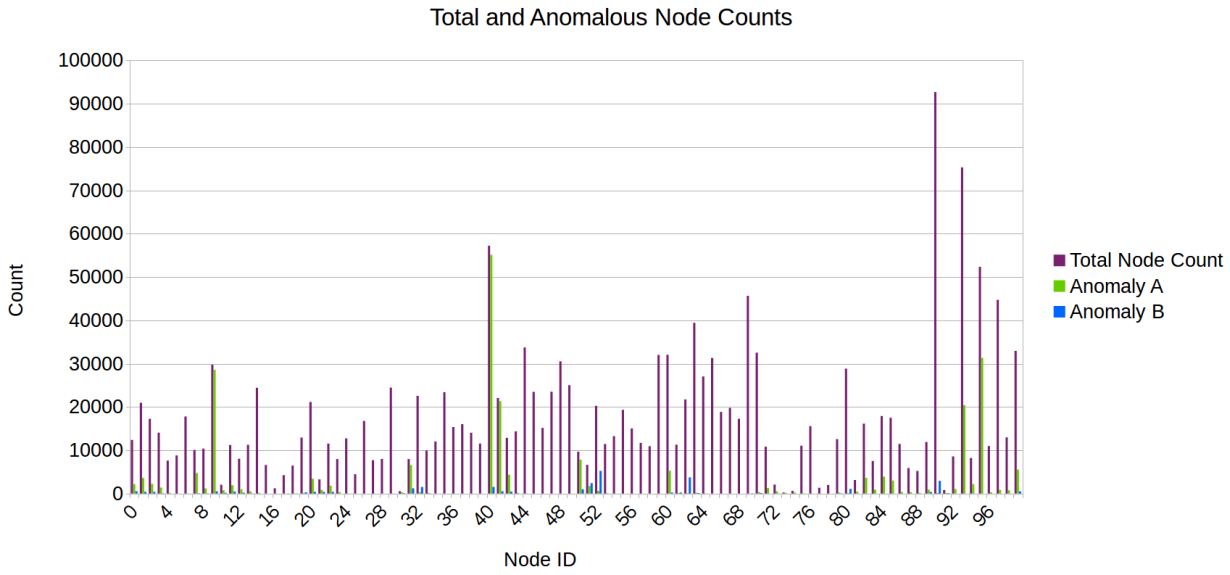

Figure 5: Anomaly Proportion by Node

A training data point for the SOM consisted of a time series data point measuring the 17 expert selected pressure and temperature sensor data from the PTMS system flight run files. Time, however, was not a part of the feature vector used to train the SOM because it would have made the initial analysis more complex. In addition, since the subject matter expert labeling resolution of the data was based on a flight run, the data was not broken down enough to use time in the feature vector. This limitation was a large challenge in the analysis and limited the resolution of the anomaly detection to flight run file not time periods within a file.

Once an unsupervised SOM was trained, a contextual phase was applied. The contextual phase utilized semi-supervised machine learning methodology to overlay known anomalies back onto the SOM. Sample output from the contextual phase is shown in Figure 5. This figure shows the node ID, the number of data points in each node, and the proportion of anomalous points in each node. The hope was that the anomalies would cluster in like areas of the SOM, demonstrating that anomalous flight runs contained distinguishably different behavior. Comparing the total vs anomalous counts in each node, Figure 5 shows that nodes 10, 41, and 42 contain a high proportion of data from anomalous flight runs. Preliminary analysis of the results, like those in Figure 5, suggested that the SOM could in fact differentiate between anomalous and non-anomalous flight data. The ability of the SOM to differentiate between the anomalous and non-anomalous flight run data meant that the method could be used to build an automatic classifier. This automatic detection could help aid the maintainers by identifying potential issues in the aircraft subsystem before they become critical. In addition, an automatic method of detection increases the likelihood of an anomaly being identified, since the current amount of data is too large for the maintainer team to sort through and is too complex for standard deterministic methods.

Constructing and Testing the Semi-Supervised Classifier

Once preliminary training results demonstrated the SOM's ability to separate anomalous and non-anomalous data, the next step was to build a classifier. This classifier automatically determines if a flight run was anomalous or not. Building this classifier required a way to use the time series data from within a flight run file, to classify an entire flight run as anomalous or not. To accomplish this, the authors used the information from Figure 5. Inspecting the graphs for each individually trained SOM, often showed that much of the data from the anomalous flight runs fell into a handful of nodes. These nodes were designated as anomaly indicators. A flight run file was classified as anomalous if a certain number of its data points fell into these anomalous nodes. This certain number of data points was referred to as the threshold. Initially, the threshold determining the optimal number of data points was unclear, so the threshold value for making the anomaly decision was turned into a hyperparameter and optimized. Designating the threshold as a hyperparameter allowed the authors to experimentally determine the threshold corresponding to the maximum classification accuracy. This threshold method was necessary in part because of the uncertainty associated with the labeled data. Recall that a flight run was designated anomalous or not anomalous based on SME analysis. The SME generally labeled a flight run anomalous only after a known issue was found. While the flight run file was known to have anomalous behavior, one could not be certain that all the measurements within a file were truly anomalous. Some of the data could in fact be partially normal. As a result of this coarse labeling, uncertainty was injected into the

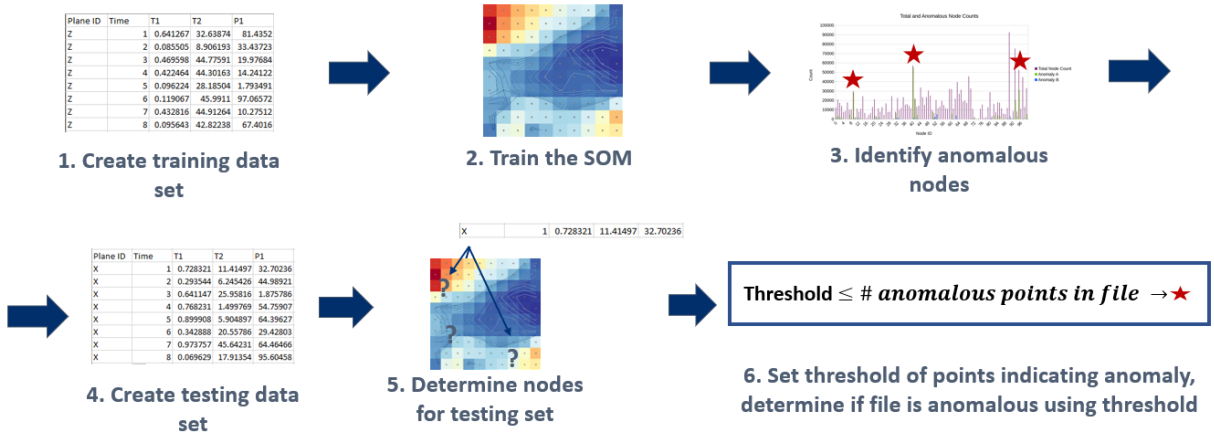

Figure 6: SOM Training and Classification Process

system. As such, the system needed to be tolerant to the existence of some non-anomalous data contained in a flight run labeled anomalous. The threshold method allowed for this to be considered.

The entire training and testing process for the classic cluster classifier is shown in Figure 6. The first training step involves creating a data set that contains examples of both anomalous and non-anomalous data. The existence of anomalous data in the data set is important since it allows the researchers to determine where the known anomalies cluster. The second step trains the SOM using the training method outlined in the Background section. After the SOM is trained, the next step involves determining where the limited number of data points from known anomalous files fall. From here the last step is to test how accurate the trained classifier is at predicting anomalies in a testing data set using a predetermined threshold. The next section provides the results from this process and discusses their implications.

RESULTS AND DISCUSSION

To test the SOM classifier, setting several parameters were necessary. Often ML requires tuning what are called hyperparameters. These hyperparameters vary depending on the machine learning method selected. For the SOM classifier the hyperparameters consisted of map size, and threshold for anomalous file designation. For the results presented below these hyperparameters were varied in an attempt to find the optimal combination, resulting in the most accurate classifier. Accuracy results presented below include overall accuracy, recall, and precision. The overall accuracy is a measure of how many flight run files the classifier correctly labeled as anomalous or non-anomalous. This metric, however, does not provide a complete picture of classifier performance, especially with highly unbalanced data sets, so recall and precision are also used to augment the results presentation. Recall can be thought of as a measure of how many true positives a classifier misses. If a classifier has a very low recall, then it is missing many of the anomalous flight runs. Precision can be thought of as a measure of how accurate the classifier is when it predicts a flight run contains anomalous behavior. If a classifier has a low precision it is not very accurate when it flags a file.

Use cases influence the level of precision and recall required in ML classifiers. For example, say letting a defective part off an assembly line is very expensive. In that case a ML system would require high recall. It would be conservative and flag parts as defective even if there was only a slight suspicion. In another application, taking a machine out of service for maintenance may be very expensive. In that situation, a ML classifier with high precision would be needed since high confidence would be required when taking a machine off-line. The testing results discussed below present the implications of these tradeoffs as they correspond to the tabulated results. Note that classifier accuracies presented are an average based on three independent SOM classifiers for a given hyperparameter combination. This average measure is intended to damp out any variation associated with randomly selecting testing and training data.

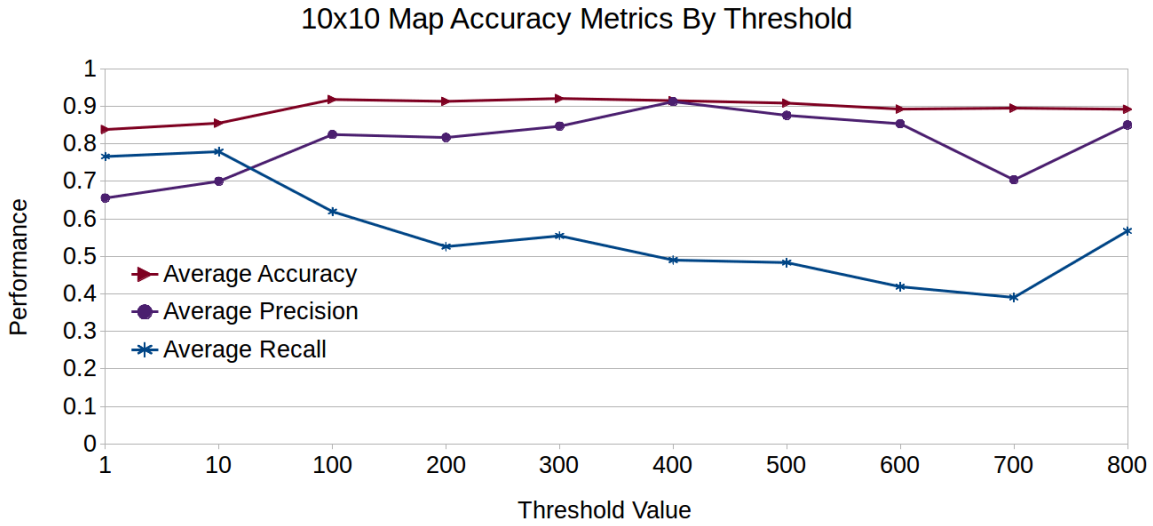

Figure 7. 10x10 Map Accuracy Results

The first SOM based flight run classifier used a ten by ten SOM map. This map size was fixed and the threshold for classifying a file as anomalous was varied. In this classifier, each parameter combination was tested using three independently trained SOMs whose results were then averaged. For each run, seven-hundred flight files were used for training and five-hundred for testing. The number of files used for training was dictated by the memory constraints of the computer. Figure 7 shows the overall accuracy, precision, and recall values for each threshold. On the graph, a score of one indicates perfect performance or one-hundred percent correct answers. In practice, a classifier rarely reaches one-hundred percent accuracy. In order to deploy a classifier, generally accuracy is desired to be above the high eighties. Looking at the results, the graph suggests that for a ten by ten map size a threshold of one-hundred produces the highest overall accuracy while also providing a high recall and precision. The graph also shows that a threshold increase results in a decrease in recall and an asymptotic increase in precision. This means that as the threshold goes up, it requires that more points fall into the anomalous nodes before a flight is flagged as anomalous. This results in higher prediction precision. Remember, however, in some cases a higher recall might be more desirable. It is up to the user to make the tradeoff depending on the application. For the SME's application a higher precision is required since repairing the subsystem can be time consuming and expensive. As a result, when a system is flagged they desire high confidence that something is not performing as expected. This desire for high confidence drove the selection of a higher precision with at the expense of recall for the ten by ten map.

The second SOM based flight run classifier tested used a twenty by twenty SOM map. Again, the map size was fixed and the threshold for classifying a file as anomalous was varied. Each parameter combination was tested using three independently trained SOMs whose results were then averaged. For each run, seven-hundred flight files were used for training and five-hundred for testing. Figure 8 shows the overall accuracy, precision, and recall values for each threshold. Looking at the results, the graph suggests that for a twenty by twenty map size a threshold of ten produces the most accurate overall accuracy while also providing a high recall and precision. As with the ten by ten map, testing results generally show that increasing the threshold decreases recall and increases precision to a point. While both classifiers ultimately produce similar overall accuracies, the twenty by twenty classifier provides less of a tradeoff between precision and recall. As a result, it is likely the twenty by twenty map size would be used. This result suggests that the map size hyperparameter plays an important role determining the accuracy of the SOM based classifier. This could also suggest that a larger map size is better able to differentiate between normal and anomalous system behavior.

To test the large map size hypothesis the authors also trained a fifty by fifty map. Again, the map size was fixed and the threshold for classifying a file as anomalous was varied. Results showed that the fifty by fifty map classification accuracy and precision were similar to the ten by ten and twenty by twenty. However, the recall for the map hovered around a tenth. Much less than the ten by ten and twenty by twenty maps. Upon inspection of the trained SOM, the authors noticed that many of the nodes were empty of training data. This suggests that the amount of data being fed into the SOM for training is not enough for the large map size. In addition, such a large map size might cause the SOM

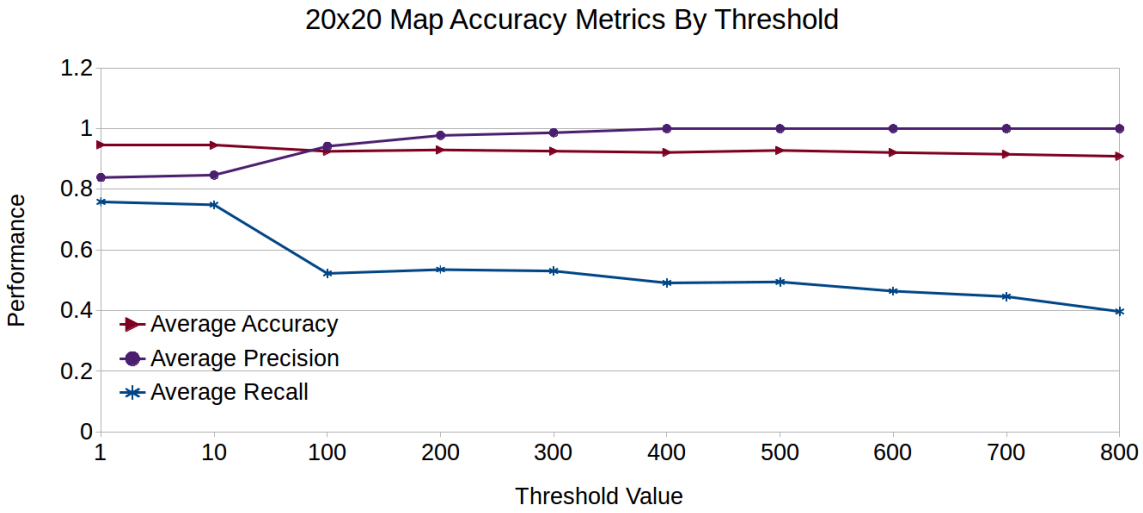

Figure 8. 20x20 Map Accuracy Results

to cluster on random noise in the data rather than meaningful differentiators. The impact that map size and amount of training data have on classifier accuracy should be explored more in the future. Overall, the results demonstrate that a fairly accurate ML classifier can be created using challenging data. While there is room for improvement, the ML classifier as is can help maintainers identify anomalous flights that may have gone unnoticed previously. Ultimately, the tool is a step in the right direction and a demonstration of how carful ML model creation can help provide benefits even when presented with challenging data.

CONCLUSION AND FUTURE WORK

With increases in equipment complexity organizations are turning to data driven analysis via machine learning to help pinpoint issues. This monitoring is made possible by increasingly economical data collection. While the technology industry has demonstrated success analyzing data for trends using traditional ML and artificial intelligence, there exist unique challenges outside of this domain. Challenges like data collection and lack of labeled data make using traditional machine learning methods infeasible. Careful method selection, however, can result in an accurate machine learning model, able to help pinpoint and identify when equipment is behaving sub-optimally.

The work presented in this paper discusses the development of a semi-supervised ML method to identify anomalous performance in an aircraft subsystem. Occasional anomalous performance and data labeling difficulties made it challenging to construct a machine learning model. To combat this the authors developed a Self-Organizing Map (SOM) based classifier. Classifier testing results demonstrated that it was able to detect anomalous PTMS performance around 90% of the time using both a ten by ten and a twenty by twenty map. However, results also showed that classifier hyperparameters play an important role determining more nuanced measures of performance. As a result, these hyperparameters need to be carefully tuned based on the application of the classifier. Ultimately, the tool helps to decrease maintainer analysis work load when looking for issues in the subsystem output data. Moving forward, future work on the classifier will focus on pinpointing specific causes of anomalous system behavior, further aiding the maintainers and hopefully reducing equipment downtime.

REFERENCES

Anshuk Gandh, Carmen Magar, R. (2013). *How technology can drive the next wave of mass customization.* McKinsey on Business Technology.

Aquize, V., Emery, E., Buarque, F., & Neto, L. (n.d.). *Self-organizing Maps for Anomaly Detection in fuel consumption. Case Study:Illegal fuel storage in Bolivia*.

Artificial Intelligence (AI) in Workspace Industry 2019 Review. (2019). Retrieved from Market Watch: https://www.marketwatch.com/press-release/artificial-intelligence-ai-in-workspace-industry-2019-review-future-growth-global-survey-in-depth-analysis-share-key-findings-company-profiles-2019-03-25

Bean, R. (2018). *The State of Machine Learning in Business Today*. Retrieved from Forbes: https://www.forbes.com/sites/ciocentral/2018/09/17/the-state-of-machine-learning-in-business-today/#5dbf6f93b1de

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning.* Cambridge, MA.: MIT Press.

Chen, Y., Qin, B., Liu, T., Liu, Y., & Li, S. (2010). *The Comparison of SOM and K-means for Text Clustering.* 268-274: Canadian Center for Sciences and Education.

Columbus, L. (2019). *The Most Innovative Companies of 2019 According to BCG*. Retrieved from Forbes: https://www.forbes.com/sites/louiscolumbus/2019/03/24/the-most-innovative-companies-of-2019-according-to-bcg/#50b7ee82486d

Drezner, J. (2009). *COMPETITION AND INNOVATION UNDER COMPLEXITY.* RAND.

Fan, Z., Chiong, R., Hu, Z., & Lin, Y. (2018). Investigating the effects of varying cluster numbers on anomalies detected in mining machines. *1st International Conference on Computer and Drone Applications: Ethical Integration of Computer and Drone Technology for Humanity Sustainability, IConDA 2017.*

Fryer-Biggs, Z. (2018). *The Pentagon plans to spend $2 billion to put more artificial intelligence into its weaponry - The Verge*. Retrieved from The Verge: https://www.theverge.com/2018/9/8/17833160/pentagon-darpa-artificial-intelligence-ai-investment

González, F., & Dasgupta, D. (2002). *Neuro-Immune and Self-Organizing Map Approaches to Anomaly Detection: A Comparison.*

Grammich, C., Arena, M., Younossi, O., Brancato, K., & Blickstein, I. (2008). *A Macroscopic Examination of the Trends in U.S. Military Aircraft Costs over the Past Several Decades.*

Greensmith, J., Feyereisl, J., & Aickelin, U. (2009). The DCA: SOMe comparison. *Evolutionary Intelligence*.

Hawley, J., & Swehla, M. (2018). *The New Equipment is Here, Now Comes the Hard Part: Cognitive and Sociotechnical Challenges in Network-Enabled Mission Command.* ARL.

Hoglund, A., Hatonen, K., & Sorvari, A. (2000). *A COMPUTER HOST-BASED USER ANOMALY DETCTION SYSTEM USING THE SELF-ORGANIZING MAP.*

How real-time data is transforming five industries. (2019). Retrieved from Innovation Enterprise Channels: https://channels.theinnovationenterprise.com/articles/how-real-time-data-is-transforming-five-industries

Hsu, C.-W., & Chang, C.-C. &.-J. (2003). *A Practical Guide to Support Vector Classification.* Department of Computer Science and Information Engineering, National Taiwan University.

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 626-634.

Iriarte, M. (2018). *Test and measurement sector grapples with standardization, complex systems*. Retrieved from Military Embeded Systems: http://mil-embedded.com/articles/test-measurement-grapples-standardization-complex-systems/

Kobielus, J. (2019). *Looking ahead to Next '19, Google puts AI at the center of cloud hyperscaling*. Retrieved from Silicon Angle: https://siliconangle.com/2019/03/25/looking-ahead-next-19-google-puts-ai-center-cloud-hyperscaling/

Kohonen, T. (2012). *Self-organizing maps* (Vol. 30 ed.). Springer Science & Business Media.

Langin, C., Zhou, H., Rahimi, S., Gupta, B., Zargham, M., & Sayeh, M. (2009). A self-organizing map and its modeling for discovering malignant network traffic. *2009 IEEE Symposium on Computational Intelligence in Cyber Security, CICS 2009 - Proceedings.*

Le, D., & Zincir-Heywood, A. (2018). Evaluating insider threat detection workflow using supervised and unsupervised learning. *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018.*

Lever, J. K. (2017). Principal Components Analysis. *Nature*, 641-642.

Libert, B., & Beck, M. (2018). *Machine Learning Is A Moneyball Moment For Companies*. Retrieved from Forbes: https://www.forbes.com/sites/barrylibert/2018/08/31/machine-learning-is-a-moneyball-moment-for-companies/#4bac378e44ec

Martin, N., Bobalik, J., Wall, K., Robbins, D., De Stena, D., Rail, K., & Plag, K. (2018). F-35 Subsystems Design, Development & Verification.

Megatroika, A., Galinium, M., Mahendra, A., & Ruseno, N. (2016). Aircraft anomaly detection using algorithmic model and data model trained on FOQA data. *Proceedings of 2015 International Conference on Data and Software Engineering, ICODSE 2015.*

Ng, R., & Han, J. (1992). *Efficient and Effective Clustering Methods for Spatial Data Mining.* 144-155: VLDB Conference.

Purdy, M., & Daugherty, P. (2017). *How AI Boosts Industry Profits and Innovation.* Accenture.

Russell, S., & Norvig, P. (2016). *Artificial intelligence: a modern approach.* Malaysia: Pearson Education Limited.

Sarasamma, S., Zhu, Q., & Huff, J. (2005). Hierarchical Kohonenen Net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*.

Spencer, M. (2019). *Artificial Intelligence Hype Is Real*. Retrieved from Forbes: https://www.forbes.com/sites/cognitiveworld/2019/02/25/artificial-intelligence-hype-is-real/#33aa935d25fa

Tan, X., Liu, X., Yang, Y., Wei, M., & Qiu, B. (2018). Condition Monitoring for the Marine Diesel Engine Economic Performance Analysis with Degradation Contribution.

Ultsch, A., & Vetter, C. (1995). *Self-Organizing-Feature-Maps versus Statistical Clustering Methods: A Benchmark.*

Vanem, E., & Brandsater, A. (2019). Cluster-Based Anomaly Detection in Condition Monitoring of a Marine Engine System.

Wiggers, K. (2019). *Google researchers improve reinforcement learning by having their AI play Pong | VentureBeat*. Retrieved from Venture Beat: https://venturebeat.com/2019/03/25/google-researchers-improve-reinforcement-learning-by-having-their-ai-play-pong/

You, X., Zhang, C., Tan, X., Jin, S., Wu, H. (2018). AI for 5G: Research Directions and Paradigms. *Science China Information Sciences, arXiv:1807.08671v1*. Retrieved from https://arxiv.org/pdf/1807.08671.pdf.

