Characterizing the current state of a propulsion engine: A comparison of machine learning frameworks

CASI AERO 2019 – 22nd CASI Aerospace Propulsion Symposium

Srishti Sehgal [1,3]
srishti.sehgal@nrc-cnrc.gc.ca
OR
srishti.sehgal@mail.utoronto.ca

Catherine Cheung [2]
catherine.cheung@nrc-cnrc.gc.ca

Julio J Valdes [2]
julio.valdes@nrc-cnrc.gc.ca

[1] *Full-time undergraduate at University of Toronto – National Research Council Canada Co-op Student*
[2] *National Research Council Canada*
[3] *Corresponding author*

***Abstract* – In this digital age, the number of sensors installed on equipment to monitor operator inputs and associated outputs has increased. One of the ways that equipment health is monitored is by using sensor data with machine learning techniques. An empirical model can examine relationships between equipment steady state physical characteristics and other indicators of equipment health to systematically generate failure predictions when these parameters deviate from normal operating ranges. The main objective of this work is to identify healthy and failed states of a propulsion engine. This paper describes an analysis of two sensor data-driven frameworks. The *baseline framework* involves training a neural network directly with the input dataset for binary classification. It is intended to be the control group whose results will be used as a baseline for comparison. In the *multi-task framework*, an autoencoder is jointly trained with a neural network for binary classification. This allows analysts to generate and feed a compressed version of the input dataset through the neural network for classification and pool multiple loss functions together as a single objective for training. To quantify each framework's success in achieving the main goal, classification results are evaluated using binary classification performance metrics such as sensitivity and precision. The study demonstrated that the compressed feature space was more linearly separable than the original data space. This would indicate that better classification results can be obtained if the compressed feature space is used to train a classification model. However, the multi-task framework only achieved results of similar accuracy as the baseline framework. In spite of this shortcoming, the multi-task framework was able to yield those results in a third of the computation time taken by the baseline framework. That observation was the most significant outcome of the study. The techniques presented in this paper are expected to contribute to the creation of tools for real-time or quasi-real-time model updates and expand the data space in order to analyze sensor information and provide updated, adaptive predictions.**

Keywords: deep learning, failure modelling, multi-task learning, equipment health monitoring, dimension reduction

1. Introduction

Due to low cost and reduced sensor sizes, engineers and technologists have the ability to install numerous sensors on equipment, like a propulsion diesel engine (PDE). Using the sensor data, the engine health can be monitored and studied to improve its safety, economy and dependability. Typically, a machine learning model can examine relationships between equipment steady state physical characteristics and other indicators of equipment health to systematically generate failure predictions when these parameters deviate from normal operating ranges. However, developing strategies and capabilities to extract useful information from the tremendous amounts of healthy and failed steady state examples collected is very challenging. To add, like many complex systems, the distribution of failed to healthy steady state examples is skewed as failure events are not common. This class imbalance poses difficulties for these engineered algorithms, as they will be biased towards the majority healthy class. To address this challenge, NRC has developed many machine learning models to extract useful information from the data. When the class imbalance is reduced, these models have resulted in better outcomes.

In previous work, initial analysis of the sensor data related to the PDE was carried out [1]. The recorded data was able to capture an incident where the turbocharger was seized. In the analysis of this event, semi-supervised and unsupervised anomaly detection methods were engaged to

identify deviations in equipment parameters with a known outcome. These models classified engine sensor data as part of a healthy system or as part of a failed engine system [1]. To combat the imbalanced dataset, NRC also studied the impact of *data reduction* on the same classification task [1]. Although some classification and anomaly detection methods proved to be successful, there was interest in extending the study into the effects of *dimension reduction* on classification performance. In this process, a dataset with a large feature set was transformed into a dataset with a smaller feature set while ensuring that the smaller feature set conveyed the same information concisely. The benefits include a reduction in computation time and a potential increase in accuracy.

To evaluate this methodology, NRC is investigating the use of a multi-task framework (MTF) that generates an unsupervised model to extract features that may share complex non-linear relationships, while simultaneously training a classification model. If these learning tasks are related, training them jointly can lead to a greater performance improvement than if they were trained separately. Multi-task learning aims to improve the generalization performance of several related tasks and share knowledge to develop a common feature set among tasks. Similar to human learning, the knowledge contained in one task can be leveraged by other tasks [2, 3].

This study describes the structure of two frameworks. The first framework used a neural network for direct classification from an input dataset. The second framework, the MTF, reduced the dimension of the data prior to classification. The objective of this effort was to see if the MTF would improve training time, improve classification accuracy and set a direction towards the future development of efficient frameworks. To quantify the success of these frameworks, classification results were evaluated with binary classification performance metrics such as precision and recall.

This paper is organized as follows: Section 2 provides details of the propulsion engine data used, Section 3 describes the methods and tools used for data analysis, Section 4 details the data pre-processing steps and experimental settings, Section 5 presents the results of the two frameworks developed, and Section 6 summarizes the findings.

2. Propulsion engine data

The data analyzed in this work came from a propulsion diesel engine (PDE) system. The diesel engine system consists of two banks of 10-cylinders, referred to as Bank A and Bank B. The PDE also contains twin air-cooled turbochargers, denoted Turbo A and Turbo B, each assigned to one of the banks of 10-cylinders.

The PDE sensor system includes 238 sensors that record information related to operator inputs, equipment outputs, and sensor data. This data is recorded at rates up to 2 Hz. A subset of 51 signals from the full set of 238 sensors were selected and included in the analysis. This selection was made using expert and domain-specific knowledge. Table 1 lists the 51 sensors included in this analysis, encompassing parameters such as speeds, temperatures, pressures, and activation of specific alarms.

Operational data was collected from 6 vehicles featuring this PDE system. Several failures experienced by these vehicles related to the diesel engine system were identified:

- Turbocharger failure (9482 data points)
- Minor governor fault (1440 data points)
- B9 cylinder head replacement (4454 data points)

The data related to these incidents (labelled 'failed') as well as data corresponding to normal operating conditions (considered 'healthy') were analyzed. The total data available was 230159 samples, with 15376 "failed data" points and 214783 "healthy" data points. The data was separated into training and testing sets as detailed in Table 2.

All the input variables in the training data were standardized by converting them to z-scores so that all variables had a mean of 0 and a standard deviation of 1. By ensuring that all variables have the same unit variance after standardizing, the influence of each variable in similarities, distances, etc. is the same. The mean and standard deviation of each signal from the training data were then used to pseudo-standardize corresponding signals in the testing data so that the models could be applied to the unseen testing data. Some common training/testing data splits evaluated included: 80/20, 75/25, 70/30, etc… Various splits within the 80/20 and 70/30 were applied without much variance in results. Thus, arbitrary splits of 75/25 and 70/30 were chosen for healthy data and failed data, respectively. A breakdown of the data is shown schematically in Figure 1.

Table 1 Subset of PDE parameters included in analysis

Signal #	Signal	Signal #	Signal
1	Turbo A speed	27	B10 cylinder exhaust gas temperature
2	Turbo B speed	28	Main bearing temperature 1
3	Turbo B inlet temperature	29	Main bearing temperature 2
4	Turbo B outlet temperature	30	Main bearing temperature 3
5	Turbo A inlet temperature	31	Main bearing temperature 4
6	Turbo A outlet temperature	32	Main bearing temperature 5
7	Charge air manifold pressure	33	Main bearing temperature 6
8	A1 cylinder exhaust gas temperature	34	Main bearing temperature 7
9	B1 cylinder exhaust gas temperature	35	Main bearing temperature 8
10	A2 cylinder exhaust gas temperature	36	Main bearing temperature 9
11	B2 cylinder exhaust gas temperature	37	Main bearing temperature 10
12	A3 cylinder exhaust gas temperature	38	Main bearing temperature 11
13	B3 cylinder exhaust gas temperature	39	Lube oil sump low level alarm
14	A4 cylinder exhaust gas temperature	40	Fuel supply pressure "low"
15	B4 cylinder exhaust gas temperature	41	High pressure fuel lines leakage
16	A5 cylinder exhaust gas temperature	42	Fuel rack position
17	B5 cylinder exhaust gas temperature	43	P1 P2 bypass failure
18	A6 cylinder exhaust gas temperature	44	Engine speed
19	B6 cylinder exhaust gas temperature	45	Engine emergency stop
20	A7 cylinder exhaust gas temperature	46	PDE load control
21	B7 cylinder exhaust gas temperature	47	PDE clutch engaged
22	A8 cylinder exhaust gas temperature	48	Major governor fault
23	B8 cylinder exhaust gas temperature	49	Minor governor fault
24	A9 cylinder exhaust gas temperature	50	Actual vehicle speed 1
25	B9 cylinder exhaust gas temperature	51	Actual vehicle speed 2
26	A10 cylinder exhaust gas temperature		

Table 2: Details of training and testing data for PDE system

	Healthy data	Failure data
Training data	161091 samples	10763 samples
Testing data	53692 samples	4613 samples

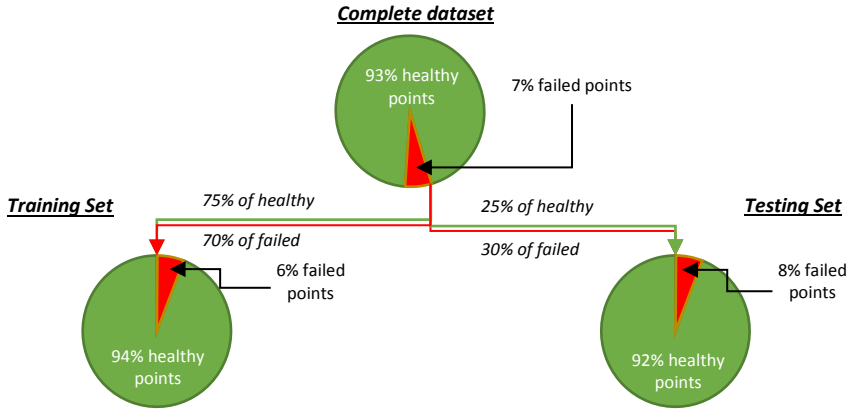

Figure 1 Distribution of healthy and failed system points for each type of dataset

3. Analytical Techniques

a. Intrinsic Dimension

The characterization of healthy and failed states of the propulsion engine system is an important topic to address, in particular, the transition between the two main states. Its description is based on sensor data integrated into a 51-dimensional time-series containing a mixture of variables affected by noise, irrelevancies and redundancies. As is typical from multidimensional information coming from real-world complex systems, there is a difference between the dimensionality of the original descriptor space and that of the lower dimensional manifold actually containing the core of the data. That is why transformations targeting the dimensionality of that subspace are useful for uncovering the underlying structure of the information. When such dimensionality is low, it is possible to construct visual approximations of those subspaces for data exploration.

When seeking an appropriate transformation, important elements to consider are i) the preservation of a property of the data used for the interpretation and ii) an intuitive metric [4]. Combined, these elements would produce a mapping of the original high dimensional objects into a lower dimension space. Properties characterizing data structure could be conditional probability distributions around neighbourhoods and similarity relations, to name a few. Typically, dimensionality reduction operations inevitably imply information loss or errors that the transformation attempts to minimize. A successful transformation generates a new set of features out of the original ones, while preserving the desired property of the data in a lower dimension space, mitigating the curse of dimensionality.

Knowing the intrinsic dimensionality of the original information, usually understood as the minimum number of features required to account for the observed properties of the data [5-8], is crucial when choosing the dimension of the target space. Once a measure of information loss between the original and the target spaces is chosen, different algorithms provide estimators to the minimal number of dimensions required for representing the data. Some approaches focus on local properties of the data whereas others emphasize the behavior of global properties. Most real-world systems are complex due to the nonlinear nature of the interdependencies between their state

variables. Under such conditions, linear estimators of intrinsic dimensionality use to be less powerful than their nonlinear counterparts, which usually are computationally more expensive.

In this work, the intrinsic dimension of the propulsion engine data is estimated using four nonlinear methods: maximum likelihood estimation (MLE), correlation dimension (CD), geodesic minimum spanning tree (GMST), and nearest neighbour estimator (NN) and one linear technique (principal component analysis, PCA).

The maximum likelihood estimator (MLE) is a nonlinear estimate derived from the assumption that the points in the vicinity of every object follows a Poisson probability distribution. From that distribution, the log-likelihood function is constructed, leading to actual dimensionality estimate [9].

One of the most commonly used techniques for estimating the intrinsic dimension is the correlation dimension [10], which is among the fractal dimension functions. The measure is based on counting pairs of objects whose pairwise distances are within an equivalent class of those smaller than a given threshold. The dependency between the correlation integral values and the distance threshold values is approximated with a linear regression whose slope is the actual intrinsic dimension estimate.

The geodesic minimum spanning tree (GMST) is another nonlinear estimator [11], derived from the following assumptions: i) The data objects are contained in a lower dimensional subspace embedded within a higher dimensional one defined by the descriptor variables, ii) the data objects are coming from a multivariate probability distribution. This technique does not require the reconstruction of the lower dimensional manifold. Rather, it seeks for an asymptotically consistent estimate. The idea is to construct a graph based on either k-neighbourhood densities, or neighbourhood distances, with the requirement that every object is connected with its neighbours. Once the graph is constructed, its associated minimum spanning tree (MST) serves as the base from which manifold parameters are estimated (entropy measures and the intrinsic dimension), using the distances along the edges of the tree [11].

The nearest neighbour estimator is based on the idea of investigating how the properties of a neighbourhood around each object change when the size of that neighbourhood is expanded. In that sense, it resembles what is done by the correlation dimension approach. However, the nearest neighbour estimator tries to approximate the multivariate probability distribution by estimating neighbourhood densities obtained by normalizing object counts within hyper-spherical neighbourhoods by their corresponding volumes. The dependency between the number of neighbours and the radius of the containing hyperspheres is approximated with a log-log regression from which the intrinsic dimension is estimated [12].

Principal component analysis (PCA) is a classical linear, unsupervised technique which can be used as a rough estimate of intrinsic dimensionality. The estimate is obtained by looking at the number of components required to achieve a specified value of the overall cumulated variance. The solution of the associated eigenvalue problem requires singular value decomposition techniques or diagonalization of covariance/correlation matrices, where the orthogonal (uncorrelated) components are linear combinations of the original descriptor variables [13].

b. Classification Methods

Neural Networks are inspired by the human brain. Each node in a neural network identifies with neurons in the brains. The connections between nodes correspond to synapses. Single layer perceptrons, a hidden-layer with back-propagation neural network, can be used for classification and regression style problems. It is also easy to implement neural networks with multiple hidden layers, or multi-layer perceptrons (MLP) [14] for classification and regression problems. For each class, there is an output node in the output layer that corresponds to a value between 0 to 1 [15].

For example, autoencoders are neural networks models that encode then decode the data. The encoder compresses the input data while the decoder transforms the encoded data back to the original format. The objective here is to train the model to be able to reproduce the input dataset with as little error as possible. Autoencoders have wide applications in tasks such as dimensionality reduction, generative models and feature extraction in machine learning [16].

c. Multi-task Learning

In machine learning, multi-task learning (MTL) is a learning paradigm. In MTL, there are multiple learning tasks such as: data classification and input data reconstruction. Among these learning tasks, all of them or at least a subset of them are assumed to be related to each other. In our study, NRC found that jointly learning these tasks can improve performance when compared with learning each task separately. MTL leverages useful information contained in multiple tasks to help improve the generalization performance of all the tasks. A quick mathematical definition of MTL is provided below:

> *Given m learning tasks $\{T_i\}_{i=1}^{m}$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for T_i by using the knowledge contained in all or some of the m tasks. Contrast with a deep network used for single-task learning, there exists m outputs (1 output for each of the m tasks) in an MTL deep network [3].*

It is intuitive to assume that the related tasks share a common feature set stemming from the original set of features. As shown in Figure 2, the common feature set is a nonlinear transformation of the original feature set that is shaped by the training of each task in the network.

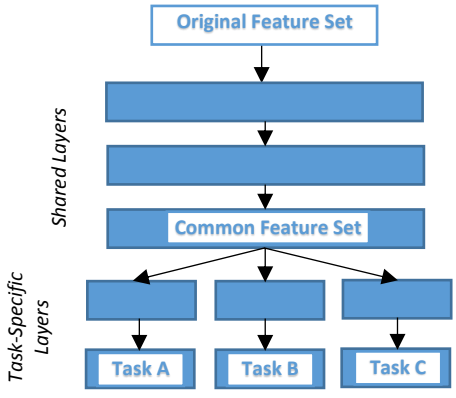

Figure 2 Sample Multi-task Learning (MTL) Deep Network

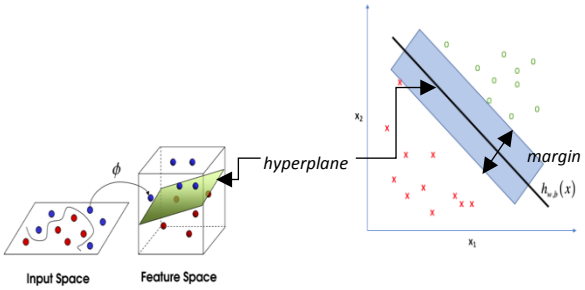

Figure 3 Hyperplane and margin for a 2D sample dataset [17]

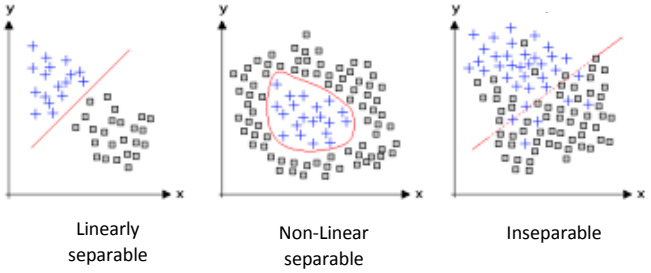

Figure 4 Comparison between datasets in 2D space that are either separable with a linear or non-linear decision boundary or inseparable [17]

d. Linear Separability and Support Vector Machines

In the field of Euclidean geometry, if two or more sets of points are linearly separable, this means that there exists a hyperplane that is able to separate the points in the dataspace. For instance, given two sets of points in 2D space, these two sets are linearly separable if there exists at least one line in the plane with one set of points on one side of the line and all points of the other sets on the other side. In machine learning, a dataset that contains classes of points that are linearly separable is indicative of a good dataset to use for classification training problems. A more mathematical definition is given below:

> *Let there be two sets of points denoted by X_0 and X_1 in an n-dimensional Euclidean space. These two sets are linearly separable if there exists n+1 real numbers $w_1, w_2, \dots, w_n, k$, such that $x \in X_0$ satisfies $\sum_{i=1}^{n} w_i x_i > k$ and every point $x \in X_1$ satisfies $\sum_{i=1}^{n} w_i x_i < k$, where x is a n-dimensional vector and x_i is the ith component of x* [18].

We wish to create a model that will classify a new data point as part of one of the classes exhibited in the training set. When support vector machines (SVMs) are employed, a data point is effectively an *n*-dimensional vector, and we would like to know if we can separate these points by class with a (n-1)-dimensional hyperplane. We do this because our goal is to determine some establishing *decision boundary* which is capable of separating the data into homogenous groups based on the class that they belong to. Typically, the best decision boundary is one that has the highest distance between the hyperplane and the nearest data point on each side, also known as margin, between the sets of points that it separates in order to increase the total confidence of predictions. An example is shown in Figure 3. If it exists, this is known as a maximum-margin hyperplane and the linear classifier (such as SVM) would be known as a maximum-margin classifier [18].

Linear separability is also seen as a feature space quality measure. Using the example provided in Figure 4, if the dataset is not as linearly separable, it means that it is much harder to classify a new data point. Therefore, we can use linear separability and SVM classifiers to see which dataset can be easily separated to help train and build a better classification model.

e. Evaluation Metrics

To evaluate binary classification models, receiver operating characteristic (ROC) metrics are often used. Some metrics include the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), false negative rate (FNR) and positive predictive value (PPV) or precision [19]. These metrics can be derived from a confusion matrix for a binary classifier as in Table 3, and in Equation 1 through Equation 4. TNR was not used in this study; thus, its formula is not shown. A perfect classifier will have only elements along the main diagonal, that is, a TP rate equal to 1 and a TN rate equal to 1. The total number of evaluated samples is equal to the sum of true positives, true negatives, false positives and false negatives. In this study, a true positive was defined as a failure event predicted as a failure event, while a true negative corresponds to a healthy system state predicted as such. Minimizing false positives will lead to a decrease in complacency in operator response. Minimizing false negatives was also important to ensure that system failures were not overlooked.

Table 3 Classification Metrics

		Ground Truth/Observed	
		Failed	**Healthy**
Model Prediction	**Predicted failed**	True Positive (TP)	False Positive (FP)
	Predicted healthy	False Negative (FN)	True Negative (TN)

	$TPR = \frac{TP}{(TP + FN)}$	$FPR = \frac{FP}{(TN + FP)}$	$FNR = \frac{FN}{(TP + FN)}$	$PPV = \frac{TP}{(TP + FP)}$
	Equation 1 True Positive Rate (TPR)	*Equation 2 False Positive Rate (FPR)*	*Equation 3 False Negative Rate (FNR)*	*Equation 4 Positive Predictive Value (PPV)*
Criteria:	*Closer to 1, the better.*	*Closer to 0, the better.*	*Closer to 0, the better.*	*Closer to 1 the better.*

4. Experimental Settings

a. Data Pre-processing

As the sensor network data was not originally designed to be used for maintenance or safety applications but for real-time equipment monitoring, the recorded data was in need of processing and consolidation before data analysis could be performed. Each signal was extracted individually from the full database. On-change values were used to fill in incomplete rows where possible and erroneous readings were removed. Next, a window of time was selected to ensure that within the window, all signals have on-change data. Afterwards, each signal was linearly interpolated and sampled at one-minute intervals within the previously selected time window. These data pre-processing steps are illustrated in Figure 5.

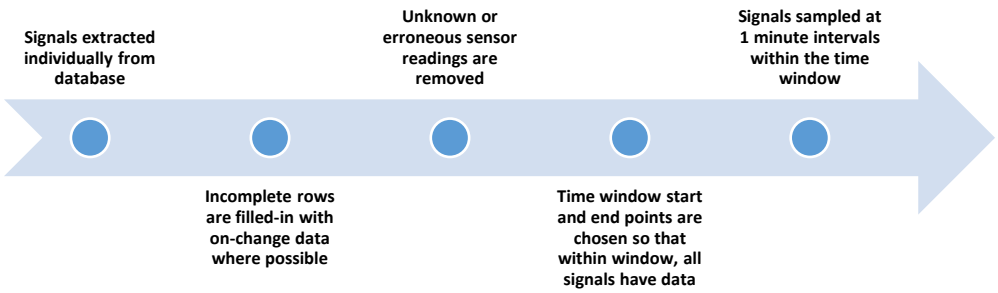

Figure 5 Data pre-processing steps

b. Description of Frameworks

i. Baseline Framework (F1)

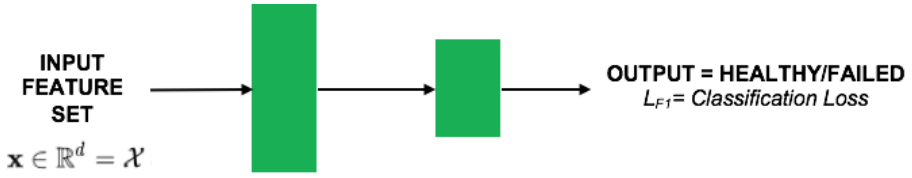

Figure 6 Baseline Framework structure

The baseline framework (F1) as shown in Figure 6 is a neural network. The architecture of the NN-MLP consisted of a 3-layer network (15 nodes, 10 nodes, 1 node). Each layer used rectified linear activation, the learning rate was 0.0001 and the optimizer used was Adam. The input feature set used was the original dataset of 51 features as detailed in Table 1. This framework was used to help evaluate the multi-task framework.

ii. Multi-task Framework (MTF)

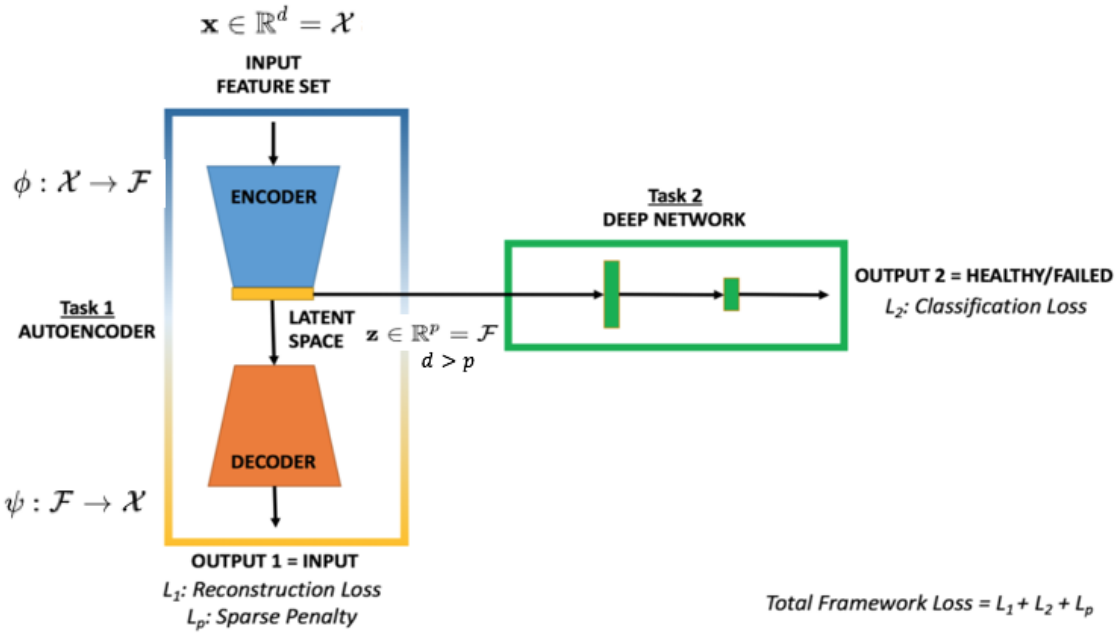

Figure 7 Multi-task Framework structure

The multi-task framework (MTF) is shown in Figure 7. In this MTF, the first task was to recreate the input dataset with a sparse autoencoder. The autoencoder, a type of neural network, contains 3 layers (60 nodes, 30 nodes, 51 nodes). The first layer was the encoder and the last two layers made up the decoder. Mean squared error (MSE) loss from Task 2 was combined with the sparsity penalty function and MSE loss from Task 1 for optimization. The objective function is formulated in Equation 5.

$$\min\left(\alpha\cdot\left(\sum_{j=1}^{d}\left(\rho\log\frac{\rho}{\hat{\rho}_j}+(1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right)\right)+\beta\cdot\left(\frac{1}{n}\sum_{i=1}^{n}(\varsigma_i-\hat{\varsigma}_i)^2\right)+\gamma\cdot\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2\right)\right)$$

Equation 5 Multi-task Framework (MTF) objective function

$$\hat{p}_j=\frac{1}{n}\sum_{i=1}^{n}\left[a_j^{(2)}\left(x^{(i)}\right)\right]$$

Equation 6 Average activation of hidden unit j

Sparsity was implemented into Task 1 using the Kullback-Leibler Divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\hat{\rho}_j$, shown by the first set of terms in Equation 5. As a result, the sparsity distribution parameter, ρ, was 0.1 and its regularization factor, α, was 0.0008. This penalty component was summed over, d, the total number of nodes in the encoding layer of the autoencoder. The term $\hat{p}_j$, shown in Equation 6, represents the average activation of a hidden unit j over all training examples, n. Sparsity was applied only during training to encourage the model to find simpler patterns in the data and hence, be more robust. The second and third terms in Equation 5 are MSE loss functions, one for each task in the MTF, regulated by β and γ that were each equal to 1.

Note that there were two tasks in the MTF shown in Figure 7. Using an autoencoder, the first task was to reconstruct the input dataset. At the bottleneck layer, the first layer of the decoder, the network was compelled to learn a compressed representation of the input. If some of the input features are related, then this algorithm should be able to discover some of those non-linear relationships. At the same time, a second learning task was used to perform a classification of the samples into either healthy or failed states. The architecture of Task 2's NN-MLP consisted of a 3-layer network (15 nodes, 10 nodes, 1 node). The primary difference between the MTF and F1 was that the MTF's second learning task attempted classification on the compressed data space produced by the autoencoder. The learning rate of the MTF was 0.0001 and the optimizer of MTF was Adam.

5. Results

a. Intrinsic dimension results

Estimates of the intrinsic dimension of the turbocharger data were calculated using the 5 techniques detailed in section 3.a. These estimates are listed in Table 4. The first 3 principal components represent 96.6% of the total variance in the data.

Table 4 Estimates of Intrinsic Dimension

Intrinsic dimension Method	Estimate
Number of eigenvalues using Principal Component Analysis	3
Maximum Likelihood Estimator	7.60
Correlation Dimension	1.51
Geodesic Minimum Spanning Tree	2.312
Nearest Neighbor Dimension	0.11

Based on the maximum likelihood estimator, the autoencoder in the MTF can be tuned to transform the dataset to a lower dimensional space with as little as 8 dimensions without suffering from major information loss. If the chosen dimension of the target space is smaller than the intrinsic dimension, the information loss increases.

b. Task 1 (MTF) Results

The data reproduced by the autoencoder was very close to the input. To visualize this, NRC has plotted the data in Figure 8 and Figure 9. The final MSE loss of the test set for Task 1 was 0.0012 and its initial loss was 0.035. Different configurations were also run, where the learning rate, number of layers, number of nodes, loss function and sparsity parameters were changed. The ratio between final and initial loss for this task did not change significantly when these hyperparameters were changed in Task 1.

Figure 8 Visualization of 100 random rows of the original input dataset

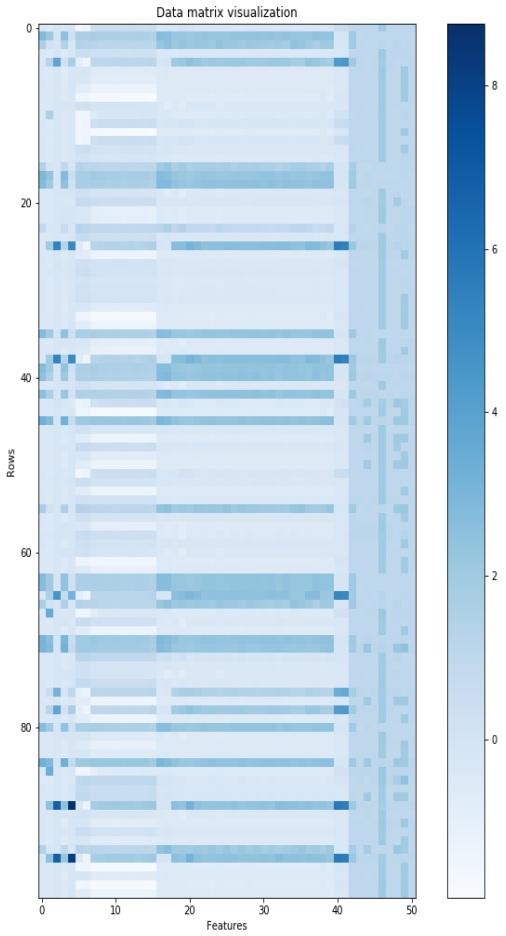

Figure 9 Visualization of the corresponding rows of input dataset as reproduced by the MTF's autoencoder. With an exception of a few visible errors, the reproduction is very close to the input dataset

c. Classification Results

Since the model did not learn the trivial identity function to reconstruct the input dataset, the compressed representation retained information from the original input dataset with only 30 features instead of the original 51. The MTF's second learning task used the compressed dataspace to train a neural network. The architecture of its neural network was very similar to the architecture of the neural network in F1. Other hyperparameters such as choice of optimizer or choice of learning rate or number of nodes in the network were kept constant to ensure that the only difference between the two frameworks would be the use of a latent dataspace and how the framework was trained. Ultimately, the goal was to develop a framework that is able to classify the propulsion engine data as a healthy system state or as a failed system state. The confusion matrices for the test dataset are provided in Figure 10 and Figure 11.

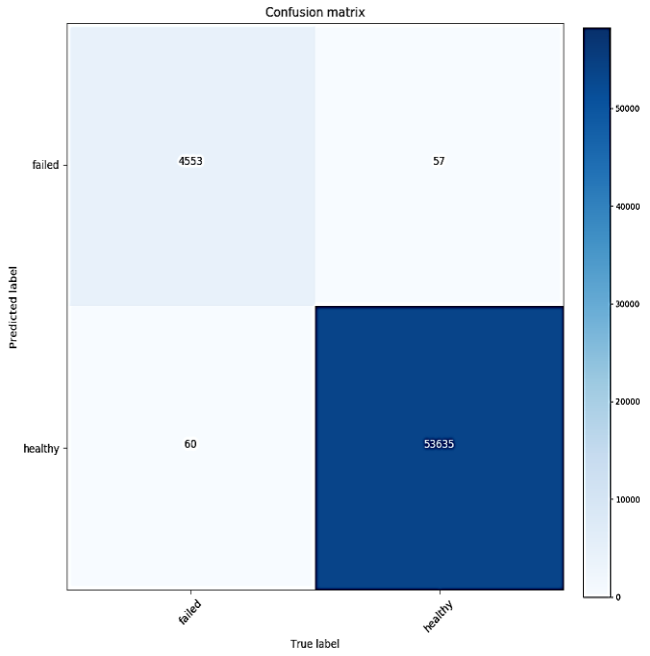

Figure 10 Confusion Matrix for Baseline Framework

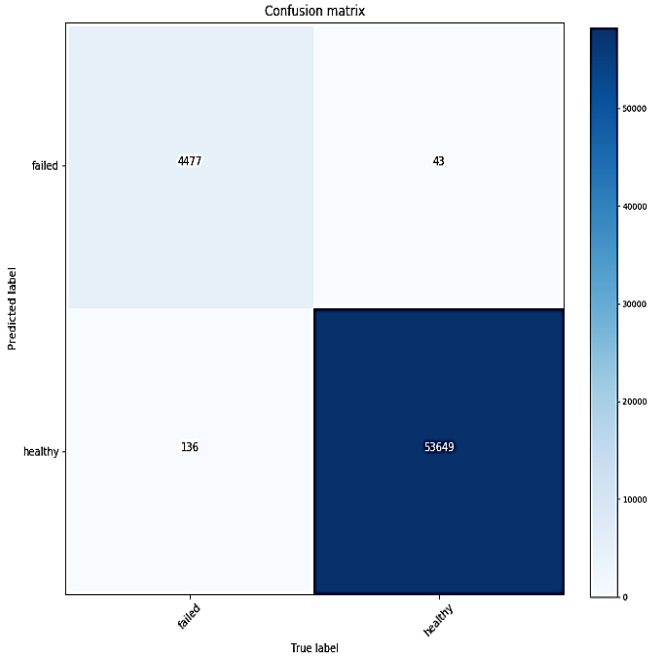

Figure 11 Confusion Matrix for Multi-task Framework's Task 2

Table 5 Classification results produced by the MTF (Task 2) and the F1

	Classification Accuracy							
	Training				Testing			
	TPR	FPR	PPV	FNR	TPR	FPR	PPV	FNR
F1	0.9851	0.0009	0.9871	0.0149	0.9870	0.0011	0.9876	0.0130
MTF	0.9712	0.0011	0.9833	0.0288	0.9705	0.0008	0.9905	0.0295

With respect to the criteria of these evaluation metrics as described in Equation 1-4, all metrics in Table 5 indicate that F1 produced better results than the MTF in training. In testing, F1 had better TPR and FNR rates, but the MTF had a better FPR and PPV rates. An ideal classifier would result in Equation 7 and Equation 8 having calculated values as close to 0 as possible. In the situation where these equations do not lead to the same conclusion, Equation 8 would take

precedence because this study involves imbalanced datasets. Based on the calculated values in Table 6, F1 did slightly better in testing and training.

$$d_1 = (1 - TPR)^2 + (0 - FPR)^2$$

Equation 7 Distance between point (TPR, FPR) to the point (1, 0)

$$d_2 = (1 - PPV)^2 + (0 - FNR)^2$$

Equation 8 Distance between point (PPV, FNR) to the point (1, 0)

Table 6 Framework evaluation using Equations 7 and 8 where final values closer to 0 are preferred

	d_1	d_2
F1	0.00017	0.000323
MTF	0.000871	0.00096
F1 < MTF?	***True***	***True***

On the other hand, the MTF was able to produce results of similar accuracy to F1 and the efficiency of the MTF was significantly better. For the same datasets, the MTF only required a third of the computation time. This result was important because as datasets get larger, computation time becomes critical. Hence, it may be beneficial to incur a small, possibly negligible, loss in accuracy for a large gain in efficiency to use the MTF for training.

d. Linear Separability Results

Table 7 Using classification results to evaluate feature spaces and linear separability

	Classification Accuracy							
	Training				Testing			
	TPR	**FPR**	**PPV**	**FNR**	**TPR**	**FPR**	**PPV**	**FNR**
Original Dataset	1.0000	0.0475	0.5844	0.0149	0.9998	0.0482	0.6406	0.0002
Compressed Dataset	0.9999	0.0249	0.7287	0.0001	0.9993	0.0249	0.8379	0.0007

In this section of the analysis, Support Vector Machines (SVMs) were employed to determine if there was any significant information loss caused by the act of compressing the original dataset to produce the compressed dataset. As explained in section 3.d, the evaluation metrics produced by SVM classifiers describe how linearly separable the dataset was. If the dataset is more linearly separable, it would be easier to distinguish between healthy and failed points in the data space. This is shown in Figure 4. Classifying new, unseen data points would be easier and the analyst would have more confidence if the dataset is linearly separable. In contrast to the situation where the dataset is inseparable, it would be more difficult to separate the data into homogenous groups. Moreover, class predictions of new, unseen data points would be less confident.

If the results of the compressed dataset are poor, then it is possible that information may be lost in the transformation. Evaluation metrics applied on results produced by both datasets are noted in Table 7. The only metric that has noticed a significant gain was PPV. All other parameters

were still relatively close between the original dataset and the compressed dataset. Equation 7 and Equation 8 show that the compressed dataset produced better classification results using SVM classifiers than the original dataset. This was noteworthy because according to section 5.c, the MTF, which used the compressed dataspace for classification, did not yield results that improved the evaluation metrics produced by F1, which used the original dataset in classification. Consequently, this warrants an additional future investigation to see why the more linearly separable dataset was not able to produce better classification results.

6. Concluding Remarks

The main objective of this work was to correctly identify healthy and failed states of a propulsion diesel engine. Results of a multi-task framework (MTF) that generated a lower-dimensional latent data space for classification were compared against a framework that did not use dimension reduction for classification. The study shows considerable promise for using an MTF to generate a compressed data space for a classification problem with an imbalanced dataset. Though the prediction rates are similar to the baseline framework, the computation time elapsed was decreased by threefold. This result is very significant because as datasets get larger, the time required to train a model will increase too. Furthermore, future real-time tools that use faster-to-train frameworks, like the MTF, could be developed to analyze sensor information and to provide more up-to-date and adaptive predictions. Choosing a framework that could potentially decrease the amount of training time while attaining accurate results is important. Ultimately, being able to detect failures near-real-time during critical operation is meaningful to: operators who need to plan their next set of actions to avoid failure, maintenance workers who need to understand where their efforts should be concentrated and engineers who can initiate improvements in new engine designs based on areas of recurring anomalies in historical data.

7. Acknowledgements

This work was supported by Defence Research and Development Canada.

8. References

[1] C. Cheung, J. Valdes, S. Sehgal, and R. Chavez. “Failure modelling of a propulsion subsystem: unsupervised and semi-supervised approaches to anomaly detection” in International Journal of Pattern Recognition and Artificial Intelligence, 2019.

[2] Y. Zhang, Y. Wei, and Q. Yang. “Learning to multitask” in Conference on Neural Information Processing Systems, 2018.

[3] Y. Zhang, and Q. Yang. “A survey on multi-task learning”. 2018. https://arxiv.org/pdf/1707.08114.pdf

[4] J. Valdés, C. Y. S. Létourneau, “Data fusion via nonlinear space transformations" in 1st International Conference on Sensor Networks and Applications, San Francisco, USA, 2009.

[5] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press Professional Inc., 1990.

[6] E. Facco, M. d’Errico, A. Rodriguez, A. Laio, "Estimating the intrinsic dimension of datasets by a minimal neighborhood information" Sci. Rep.-UK, 7(1), 2017. https://www.doi.org/10.1038/s41598-017-11873-y

[7] D. Granata, V. Carnevale, "Accurate estimation of the intrinsic dimension using graph distances: Unraveling the geometric complexity of datasets" Sci. Rep.-UK, 6(1), 2016. https://doi.org/10.1038/srep31377

[8] P. Campadelli, E. Casiraghi, C. Ceruti, A. Rozza, "Intrinsic dimension estimation: Relevant techniques and a benchmark framework" Math. Probl. Eng., 2015. https://doi.org/10.1155/2015/759567

[9] E. Levina, P. J. Bickel, “Maximum likelihood estimation of intrinsic dimension" in 17th Conference in Neural Information Processing Systems, Vancouver, Canada, 2004.

[10] P. Grassberger, I. Procaccia, “Measuring the strangeness of strange attractors" Physica D, 9(1-2), 189-208, 1983. https://www.doi.org/10.1016/0167-2789(83)90298-1

[11] J.A. Costa, A.O. Hero, “Geodesic entropic graphs for dimension and entropy estimation in manifold learning" IEEE T. Signal Proces., 52(8), 2210–2221, 2004. https://www.doi.org/10.1109/tsp.2004.831130

[12] K. Pettis, T. Bailey, A. Jain, R. Dubes, “An intrinsic dimensionality estimator from near-neighbor information" IEEE T. Pattern Anal., 1(1), 25-37, 1979. https://www.doi.org/10.1109/tpami.1979.4766873

[13] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, Cambridge University Press, 1992.

[14] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by backpropagating errors, Nature 323 (1986) 533-536.

[15] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. (Springer, New York, NY, 2016).

[16] A. Ng. “Sparse Autoencoder”. n.d. https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

[17] J. Jordan, « Support vector machines”, 2017. https://www.jeremyjordan.me/support-vector-machines/

[18] M. Sugiyama. (2016). Chapter 27 - Support Vector Classification. In Introduction to Statistical Machine Learning (pp. 303-320). Waltham, MA: Elsevier. doi:10.1016/B978-0-12-802121-7.00003-0

[19] T. Fawcett. “An introduction to ROC analysis” in Pattern Recognition Letters, 861-874, 2006. doi:10.1016/j.patrec.2005.10.010.