**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** | **Catherine Cheung 2** | **Julio J Valdes 2** |
| [*srishti.sehgal@nrc-cnrc.gc.ca*](mailto:srishti.sehgal@nrc-cnrc.gc.ca)  *OR* [*srishti.sehgal@mail.utoronto.ca*](mailto:srishti.sehgal@mail.utoronto.ca) *(613) 991 9646* | [catherine.cheung@nrc-cnrc.gc.ca](mailto:catherine.cheung@nrc-cnrc.gc.ca) | [julio.valdes@nrc-cnrc.gc.ca](mailto: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 pre-defined settings. The main objective of this work is to identify healthy and failed states of a propulsion engine. Being able to detect failures near-real-time during critical operation is significantly useful to: operators who will plan their next set of actions to avoid failure, maintainers who will 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. This work describes an analysis of two sensor data-driven frameworks. In one framework, several machine learning architectures are directly trained with the input dataset for binary classification. In the second framework, an autoencoder is used to reduce the dimension of the input dataset before training a model. To quantify each framework’s success in achieving the main objective, classification results are evaluated with binary classification performance metrics such as sensitivity and precision. In previous work, the average sensitivity rate of classification models was 75% [1]. Since data reduction improves training time and prediction accuracy [2], the autoencoder framework is anticipated to reduce training time, exceed the average prediction sensitivity rate of 75% from previous work and set a direction towards the future development of efficient frameworks. The techniques presented are expected to contribute to creating tools for real-time or quasi-real-time model updates and hence to expand the data space in order to analyze sensor information and to provide more up-to-date, adaptive predictions.

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

[2] Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection (L. P. Kaelbling, Ed.) [Abstract]. *Journal of Machine Learning Research,* *3*, 1157-1182. Retrieved from <http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf>

# Introduction

Given the ability to install sensors on equipment, like a propulsion diesel engine (PDE), equipment 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 pre-defined settings. The challenge is that during training, many models assume an equal proportion of healthy and failed state examples. However, for many complex systems, the distribution of failed to healthy steady state examples is skewed as failure events are not common. This poses a difficulty for these algorithms, as they will be biased towards the majority healthy class.

To combat the imbalanced dataset, NRC Aerospace studied the impact of *data reduction* on a classification task for sample PDE data [1]. Although some classification and anomaly detection methods proved fruitful, there is interest to extend the study into the effects of *dimension reduction* on classification performance. In this process, a dataset with a large feature set is transformed into a dataset with a smaller feature set while ensuring to convey the same information concisely.

NRC Aerospace is investigating the use of a multi-task framework (MTF) to extract features that may share a complex non-linear relationship. If 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 paper describes the structure of two frameworks. The first framework uses a Neural Network for direct classification from an input dataset. The second framework is the MTF where dimension reduction will be used prior to classification. The objective of this effort is to see if the MTF can 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 are evaluated with binary classification performance metrics such as True Positive Rate and Positive Predictive Value.

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

[2] Zhang, Y., Wei, Y., and Yang, Q. (2018). Learning to multitask. *Conference on Neural Information Processing Systems.*

[3] Zhang, Y., and Yang, Q. (2018). A survey on multi-task learning.

# Turbocharger Data

The analyzed turbocharger system contains twin air-cooled turbochargers providing the air-charge to the medium-speed diesel engine system. The diesel engine system consists of two banks of 10-cylinders, denoted Bank A and Bank B. A single turbocharger is assigned to each 10-cylinder bank; the two turbochargers are differentiated as Turbo A and Turbo B, where the letter ‘A’ or ‘B’ identifies their respective cylinder bank.

One of the incident recorded by the engines sensor system and analyzed within this work relates to the seizure of Turbo A [1]. The sensor data recorded for this particular incident was available for the month leading up to and including the time of the incident. From the resulting investigation of the Turbo A seizure, a series of key events leading to the incident were noted. The chronology of the incident’s events is detailed below, with the time of occurrence indicated as (hh:mm).

* Noted loss of sensor reading on Turbo A speed sensor
* Engine shut down for inspection
* Turbo A and B speed sensors switched
* Engine restarted at idle, still no Turbo A speed reading indicated (01:12)
* Engaged diesel engine (01:41)
* Higher speed setting requested (01:42), engine exhaust temperatures increased beyond alarm threshold, with no speed increase achieved (01:43 - 01:44)
* Diesel engine disengaged and shut down (01:44 - 01:45)

Following the incident, an inspection of Turbo A was conducted. From the inspection, it was determined that the seizure of Turbo A occurred due to a sensor installation error, which occurred when the speed sensors for Turbo A and B were switched. Insufficient spacing between the speed sensor and the turbine’s thrust collar led to rubbing and eventually the turbocharger seizure [1].

Although this failure was caused by installation error rather than gradual deterioration of a system element, the ability to characterize and distinguish the healthy state from the seized state of the turbocharger system using data analysis tools is of significant value. This type of analysis could aid in establishing failure models for further predictive work.

The sensor system of the diesel engine is comprised of 238 sensors that capture information related to operator inputs, equipment outputs, and sensor data. The sensor system provides a means for staff to control system components, monitor the systems status, or be notified via alarm when various pieces of equipment operate outside of pre-set threshold values. In addition, the sensor system allows for the recording and archiving of the operational data measured from various instruments at rates up to 2 Hz. From the 238 sensors relevant to the diesel engine, a subset of 51 signals relating to the operation of the turbochargers was selected. The 51 sensors considered within this analysis, listed in Table 1 [1], encompass parameters such as speeds, temperatures (inlet, outlet, and exhaust), pressures, and activations of specific alarms.

Table 51 Turbocharger parameters

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

# Analytical Techniques

# Intrinsic Dimension

An important aspect of this work was the characterization of the relationship between the healthy and failed states of the turbocharger system, particularly the transition between the two states. The original sensor data is described by a multidimensional time-series composed of the 51 signals. Typical from these kind of data is the presence of noise, irrelevancies and redundancies between the descriptor variables. The core of the data represents a subspace of lower dimension embedded within the higher dimensional descriptor space. In such situations, transformations to lower dimensional spaces are useful for highlighting and visualizing the underlying structure of the information.

To that end, a suitable transformation, preferably with an intuitive metric [9] would produce a mapping of the original high dimensional objects into a lower dimension one, such that a certain property of the data is preserved by the representation. Desired properties characterizing data structure could be conditional probability distributions around neighbourhoods, similarity relations and others. Under normal circumstances, such transformations imply some information loss or error that should be minimized. If successful, the transformation would generate a new set of features out of the original ones which would preserve the desired property, but in a lower dimension representation space that mitigates the curse of dimensionality.

When choosing the dimension of the target space, it is important to consider the intrinsic dimensionality of the original information which is typically understood as the minimum number of variables required to account for the observed properties of the data [10-13]. Given a functional measure of information loss, it is the minimum number of dimensions (descriptor variables) required to describe the data that minimizes that measure. This concept could be understood in several ways, which results in different algorithms aiming at producing such an estimation. Some approaches focus on local properties of the data, whereas other techniques emphasize the analysis of global properties of the data. Most complex systems in the real world exhibit nonlinear relations among their state variables, which make linear estimators of intrinsic dimensionality at a global scale less powerful than their nonlinear estimation counterparts. However, some of them are computationally expensive.

From the practical point of view, the smaller the choice of the target dimension with respect to the intrinsic one, the higher the representation error would be. On the other hand, choosing values higher than the chosen dimension introduce unnecessary attributes, redundancies and possibly noise. For visualization purposes, three or two dimensions are forcibly required. In these cases, the value of the intrinsic dimension provides a useful guide to the level of caution required when making inferences based on the visualization space.

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

The maximum likelihood estimator [14] is based on the assumption of a Poisson distribution for the *k* neighbour points and a constant behavior of the probability density function around a given point. The actual estimate of the dimension is derived from the log-likelihood function.

Correlation dimension [15] is one type of fractal dimension and it is one of the most commonly used techniques for estimating the intrinsic dimension. The idea is to compare objects from the point of view of their pairwise distances, producing a normalized count of those pairs whose distance does not exceed a given threshold (the correlation integral). The estimate is given by the slope of a log-log linear regression of the correlation integral values vs. the different distance thresholds *r*.

The geodesic minimum spanning tree (GMST) estimator [16] assumes that i) the set of multivariate objects are in a smooth manifold embedded within the higher dimensional space determined by the original descriptor variables and ii) these objects are realizations of a random process from an unknown multivariate probability density distribution. This technique produces an asymptotically consistent estimate of the manifold dimension without requiring the reconstruction of the manifold or the estimation of the multivariate distribution of the objects. The first step is to construct a graph based on *k*-neighbourhood density (or neighbourhood distances) where every object is connected with the others nearby. The second step is to build a minimal spanning tree (MST). The distances along its edges and the overall length are used to estimate parameters of the manifold, like entropy and dimension.

The nearest neighbour estimator [17] presents some similarities with the correlation dimension. It is motivated by the possibility of approximating the unknown probability density of the set of multidimensional objects, by normalizing the relative number of nearest neighbours by the volume of the hypersphere containing the objects. The procedure computes the smallest radius *r* required to cover *k* nearest neighbours via a linear log-log regression of the average minimum radius vs. *k*.

Principal component analysis (PCA) is an unsupervised, classical method that is among others, it is used to estimate intrinsic dimensionality. The estimation is simply constructed by obtaining the number of eigenvalues whose relative contributions to the overall variance exceeds a predetermined threshold (e.g. 0.975). Singular value decomposition techniques or diagonalization of covariance/correlation matrices are the typical approaches used for finding the components, which are linear combination of the original set of features. The former approach was used in this paper following the algorithm described in [18].

# Classification Methods

Supervised classification methods were attempted to build models to distinguish unhealthy turbocharger data from healthy turbocharger data. In these methods, a representative training set was given to the model with the training data labelled as “healthy" or “failed". A model was trained on this data to achieve high overall classification (true positives and true negatives) accuracy and then tested on an unseen testing set.

Neural Networks (NN) were initially developed as models for the human brain, with each node in the network representing neurons in the brain and the connections between nodes corresponding to synapses. Single hidden-layer back-propagation neural networks, also called single layer perceptrons, can be used for both regression and classification in a two-stage model. Neural networks with multiple hidden layers, or multi-layer perceptrons (MLP) [18], are also commonly used for classification and regression. For classification of K classes [13], there are K output nodes in the output layer corresponding to 0 - 1 values of each node for each of the K classes.

For example, autoencoders are neural networks models that encode and decode data where the output should look like the input. Autoencoders have wide applications in tasks such as dimensionality reduction, generative models and feature extraction in machine learning. The encoder compresses the input data while the decoder transform the encoded data back to the original format. The objective here is to train the model to be able to reproduce the output which looks like the input.

# Multi-task Learning <https://arxiv.org/pdf/1707.08114.pdf>

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, the context of our study. Among these learning tasks, all of them or at least a subset of them are assumed to be related to each other. In this case, it is found that jointly learning these tasks can improve performance compared with learning each task separately. Hence, the aim of MTL is to leverage useful information contained in multiple tasks to help improve the generalization performance of all the tasks. A quick mathematical definition is provided below:

*Given learning tasks where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for by using the knowledge contained in all or some of the tasks.* Contrast with a deep network used for single-task learning, there exists outputs (1 output for each of the tasks) in an MTL deep network.

It is intuitive to assume that the related tasks share a common feature set stemming from the original set of features. As shown in Fure3, 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.

**Original Feature Set**

*Shared Layers*

*Task-Specific*

*Layers*

**Common Feature Set**

**Task A**

**Task B**

**Task C**

Figure Sample Multi-task Learning (MTL) Deep Network

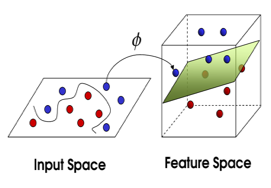
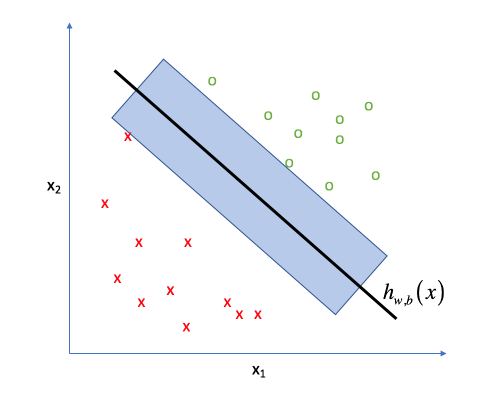
# Linear Separability and Support Vector Machines

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 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 and in an n-dimensional Euclidean space. These two sets are linearly separable if there exists n+1 real numbers , such that and every point x where is a n-dimensional vector and is the th component of .

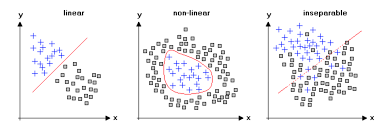
Suppose we wish to create a model that will classify a new data point as part of one of the classes exhibited in the training set. When support vector machines (SVMs) are employed, a data point is effectively a n-dimensional vector, and we would like to know if we can separate these points by class with 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. The best decision boundary is one that has the highest margin, the distance between the hyperplane and the nearest data point on each side, between the sets of points that it separates in order to increase the total confidence in our predictions. This is shown in Figure 2. If it exists, this is known as a maximum-margin hyperplane. The linear classifier (such as SVM) would be known as a maximum-margin classifier.

Linear separability is also seen as a feature space quality measure. Using the example provided in Figure 3, if the “x” and “o” dataset were not as linearly separable as shown, it means that it is much harder to classify a new data point. Thus, using a dataset whose classes can be separated easily can help to train and build a better classification model.



*hyperplane*

*margin*



Linearly separable

Non-Linear separable

Inseparable

Figure Hyperplane and margin for a 2D sample dataset

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

# Evaluation Metrics

The Receiver Operating Characteristic Metrics, also known as ROC metrics, are often used in binary classification problems to evaluate models, including the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR) and false negative rate (FNR). PPV, positive predictive value or precision, is also considered in conjunction with TP rates for classification problems that are highly imbalanced 4. The formulae for each metric are listed in Equation 1 through Equation 4. Table 2 describes these terms as part of a confusion matrix, summarizing the output of a classifier. A perfect classifier will have elements along the main diagonal only. The sum of all elements correspond to the total number of samples evaluated. In this work, a true positive is defined as a failed event predicted as a failed event, while a true negative corresponds to a healthy event predicted as a healthy event. False positives or false alarms should be minimized, since if too frequent, it could lead to complacency in operator response. False negatives also need to be minimized and avoided if possible so that any system failure is not overlooked. For prediction of failure states in a mechanical system, it is important that not only is the classification accurate but also that false alarms (false positives) are minimized.

Table 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) |

|  |  |  |  |
| --- | --- | --- | --- |
| Equation True Positive Rate (TPR) | Equation False Positive Rate (FPR) | Equation False Negative Rate (FNR) | Equation Positive Predictive Value (PPV) |

# Experimental Settings

# 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 are used to fill in incomplete rows where possible and erroneous readings were removed. Next, a window of time is 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. The table was altered to include only data corresponding to the diesel engine being activated. These data pre-processing steps are illustrated in Figure 4.

Figure Data pre-processing steps

# Training and testing sets

Turbocharger data from similar propulsion systems on entirely different vehicles were pooled together to create a large dataset. Training of the classification frameworks was carried out using the turbocharger data containing the following subset of failures: turbocharger failure, pre-lube oil pump leak, preheat pump leak, minor governor fault, and cylinder head on B9 replacement. These failed points include all of the data points after the failure event as they correspond to data related to the seized subsystem. All of the data points preceding the failure event were labelled as “healthy” data points. All the input variables in the training data were standardized, 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.

After the data sampling, 171836 data points, with 10763 of them representing failed system states, were available for training. 58323 data points, with 4613 of them representing failed system states, were available for testing. Similar distributions between healthy and failed points were maintained. The breakdown is shown in Figure 2. Some common training/testing data splits include 80/20, 75/25, 70/30, etc… Various splits within 80/20 and 70/30 were applied without much variance in results. Thus, arbitrary splits of 75/25 and 70/30 was chosen for healthy data and failed data selected at random, respectively.

Figure Breakdown of healthy and failed point distributions in each type of dataset

7% failed points

93% healthy points

6% failed points

94% healthy points

8% failed points

92% healthy points

***Training Set***

***Testing Set***

***Complete dataset***

*25% of healthy*

*75% of healthy*

*70% of failed*

*30% of failed*

Description of Frameworks

Baseline Framework (F1)

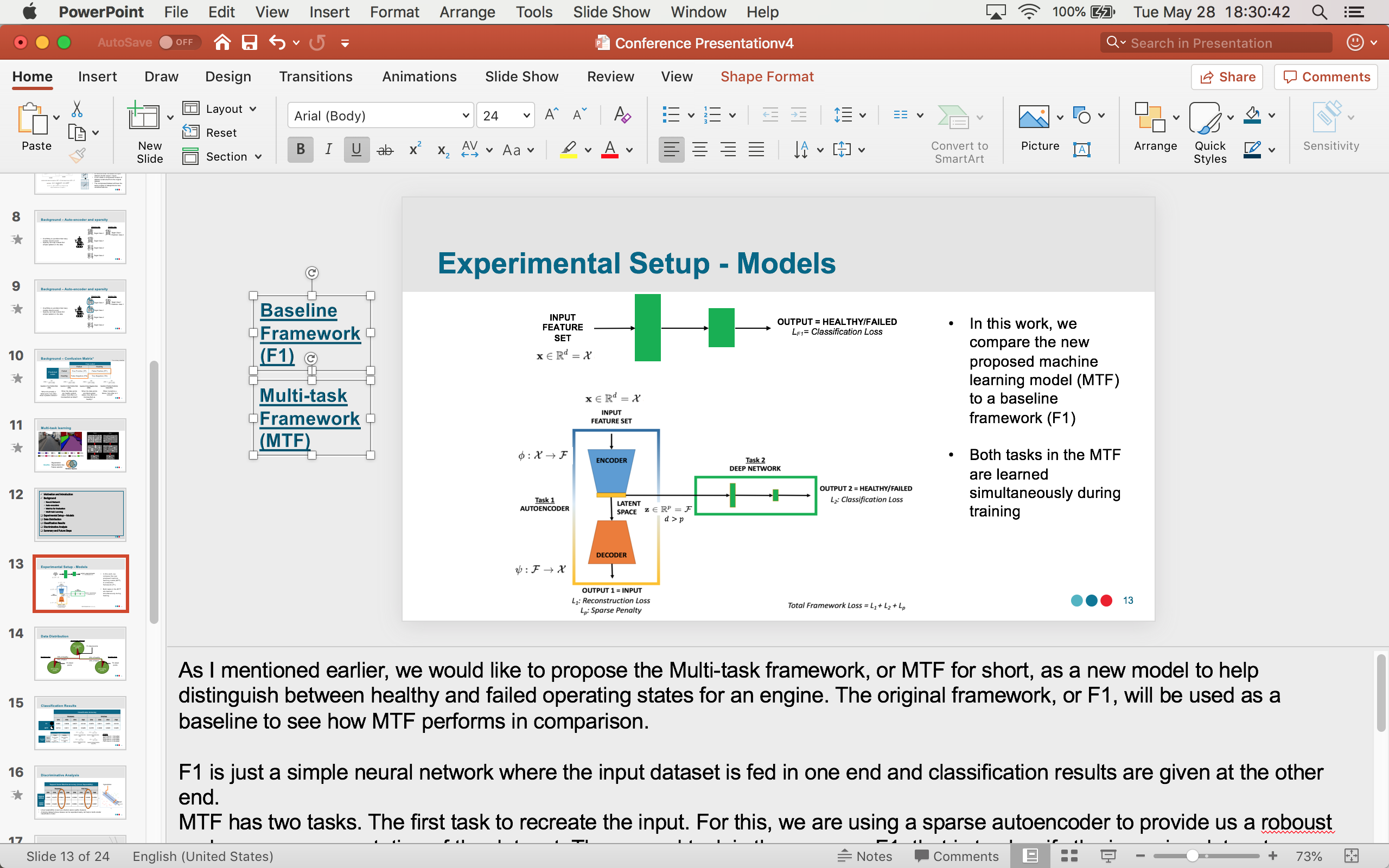


Figure Baseline Framework structure

The Baseline framework as shown in Figure is a neural network, a supervised classification method. The architecture of the NN-MLP consists 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 was Adam. The input feature set is the original dataset of 51 features as detailed in Turbocharger Data. This framework is used to help evaluate the Multi-task framework as described below.

Multi-task Framework (MTF)

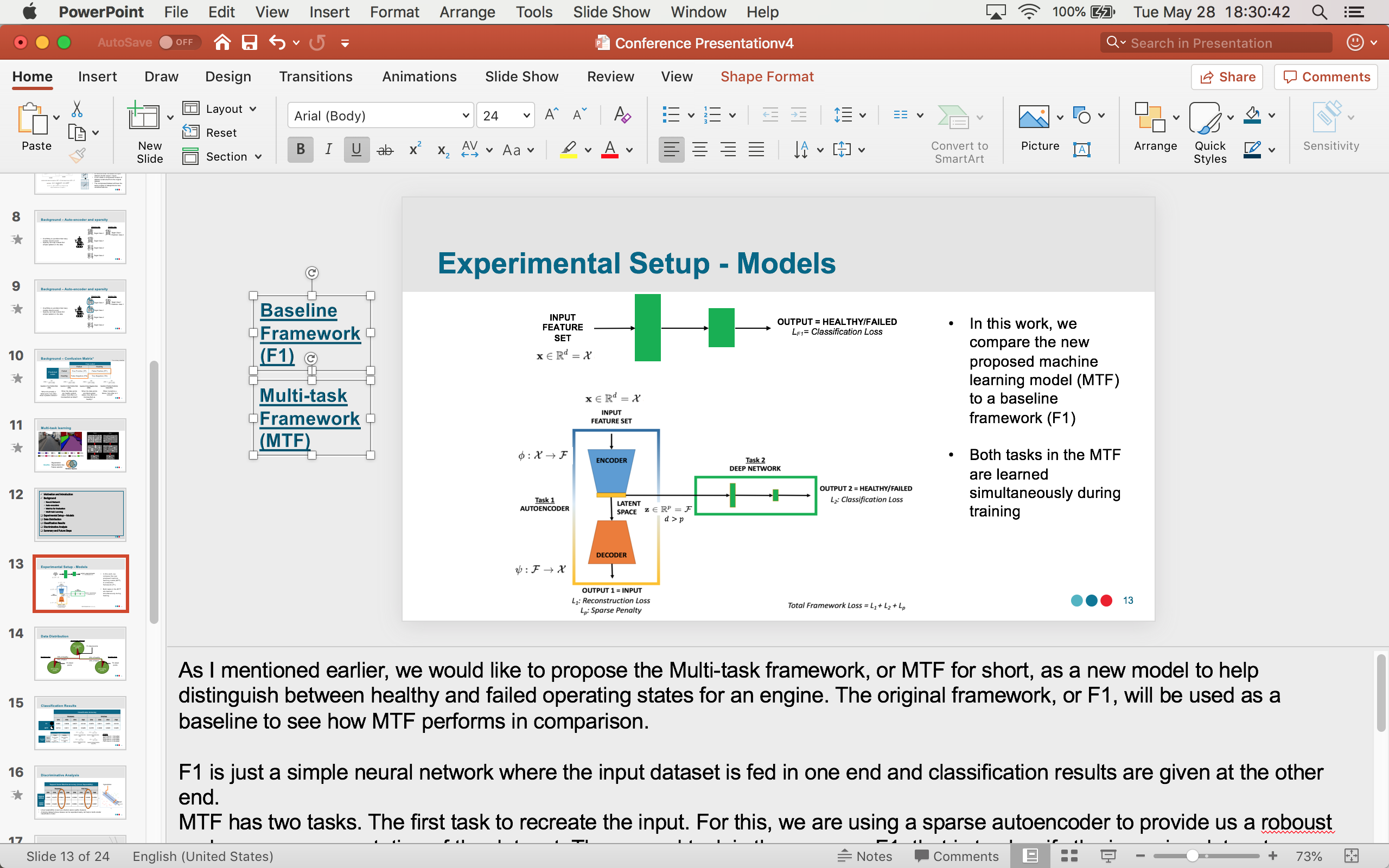


Figure Multi-task Framework structure

The multi-task framework (MTF) takes shape of typical multi-task models as shown in Figure 1. In this MTF, the first task is 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 is the encoder and the last two layers make up the decoder. Mean Squared Error (MSE) Loss, from Task 2, was pooled with the sparsity objective function and MSE Loss from Task 1 for optimization. This objective function is formulated in Equation 5.

Equation Multi-task Framework (MTF) objective function

Equation 6 Average activation of hidden unit j

Sparsity was implemented by using the Kullback-Leibler Divergence between a Bernoulli random variable with mean and a Bernoulli random variable with mean , shown by the first term in Equation 5. Hence, the sparsity distribution parameter, , was 0.1 and its regularization factor, , was 0.0008. This regularization component is summed over, , the total number of nodes in the encoding layer of the autoencoder. The term , described in Equation 6, is the average activation of a hidden unit over all training examples, . Sparsity was applied only during training to strengthen the model. The learning rate was 0.0001 and the optimizer used was Adam.

The second and third terms in Equation 5 are MSE Loss functions, one for each task in the MTF, regulated by and .

As shown in Figure 7, there are two tasks in the MTF. Using an Autoencoder, the first task is to reconstruct the input dataset. At the bottleneck layer, the first layer of the decoder, the network is compelled to learn a compressed representation of the input. If some of the input features are related, then this algorithm will be able to discover some of those non-linear relationships. At the same time, a second learning task performs a classification of the samples as either healthy or failed states. The primary difference between the MTF and the other framework is that the second learning task will attempt classification on the latent space produced by the first task.

The architecture of the NN-MLP consisted of 3 hidden layers (15 nodes, 10 nodes, 5 nodes). A recti\_ed linear activation was used and the optimization function was lbfgs (limited-memory Broyden-Fletcher-Goldfarb-Shanno [10] which is an optimizer in the family of quasi-Newton methods. An L2 penalty of 0.0001 was used for regularization, and maximum of 200 iterations with convergence criteria (tolerance of 1e􀀀4).

# Results

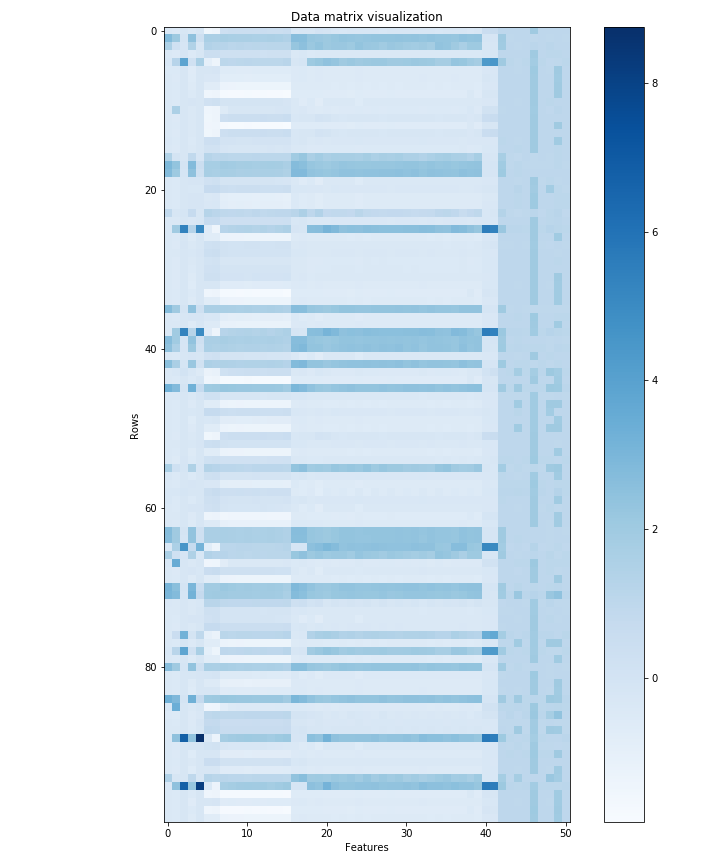
# Intrinsic dimension results

Estimates of the intrinsic dimension of the turbocharger data were calculated using the five techniques detailed in Intrinsic Dimension. These estimates are listed in Table 3. The first three principal components represent 96.6% of the total variance in the data.

Table 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 |

# Task 1

The data reproduced by the autoencoder was very close to the input. It is clear to see, when this data is visualized. 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 parameters were changed in Task 1.

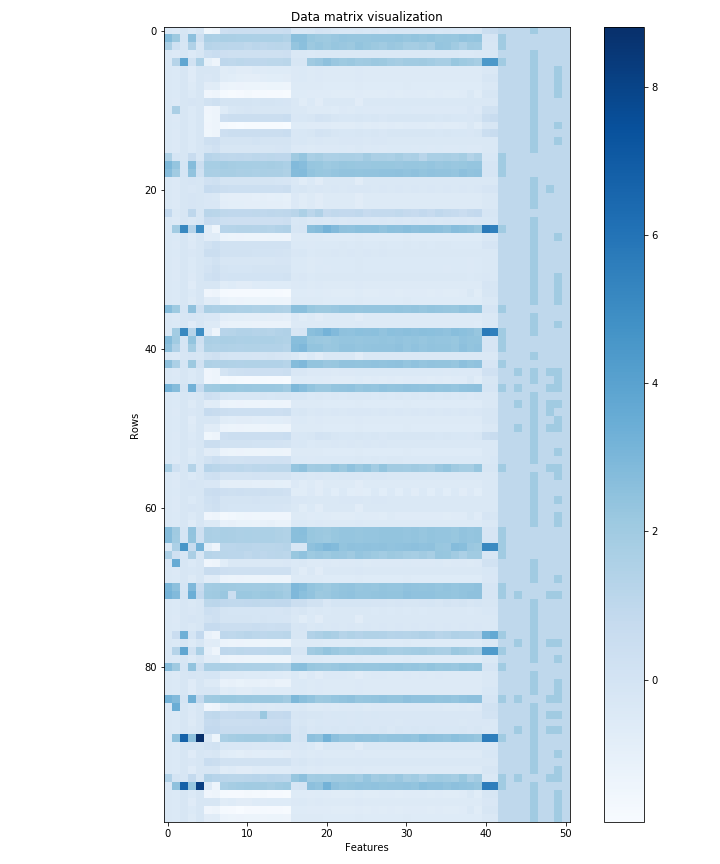


Figure 8 Visualization of the corresponding rows of input dataset as reproduced by the MTF's autoencoder

Figure 9 Visualization of 100 random rows of the original input dataset

* 1. Classification Results

Ultimately, the goal is to develop a framework that is able to classify the turbocharger data as a healthy system state or as a failed system state. The confusion matrices for test data are shown in Figure.

# Linear Separability Results

# Decision Plots

# Concluding Remarks

# Acknowledgements

# References

[1] C. Cheung, J. J. Valdés, A. Lehman Rubio, R. Salas

Figure Confusion Matrix for Multi-task Framework’s Task 2

Figure Confusion Matrix for Baseline Framework

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 |

Chavez, C. Bayley, “Low-dimensional spaces for the analysis of sensor network data: identifying behavioural changes in a propulsion system" in 5th International Symposium on Robotics and Intelligent Sensors, Ottawa, Canada, 2017. https://www.doi.org/10.1109/IRIS.2017.8250133

[9] 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.

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

[11] 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

[12] 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

[13] 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

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

[15] 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

[16] 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

[17] 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

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