Data-driven Detection of Engine Faults in Infrequently-driven Ground Vehicles

Ethan Kohrt[1], Matthew Moon[2], Matthew Sullivan[3], Sri Das[4], Michael Thurston[5] and Nenad G. Nenadic[6]

[1,2,3,4,5,6] *Rochester Institute of Technology, Rochester, NY, 14623, USA*
eakgis@rit.edu memgis@rit.edu mrsgis@rit.edu spdgis@rit.edu mgtasp@rit.edu nxnasp@rit.edu

ABSTRACT

We investigated the detection of engine faults in infrequently driven ground vehicles using data-driven methods based on neural network autoencoders. Multivariate time-series data from the infrequently driven vehicles under investigation had limited coverage of operating conditions. Hence, a considerable part of this work focused on identifying suitable vehicles, relevant signals, and pre-processing the data. We trained autoencoder models on eight vehicles with known faults and detected faults in six. Four of the faults were detectable under idle conditions and four were detectable under driving conditions. Model evaluations required human inspection to distinguish fault detections from other anomalies. We detail our procedures for pre-processing, model development, and post-processing, and we include a discussion on our interpretations of the model results.

1. INTRODUCTION

A fault detection system deployed on a ground vehicle processes data from vehicle sensors and alerts appropriate stakeholders (e.g., maintainers, operators, or logisticians) about a developing fault. Early detection saves resources by reducing unnecessary maintenance, reducing unplanned downtime, and lessening the need for redundancy, which in turn allows for a smaller fleet size. Fault detection systems allow maintainers to correct issues before they become severe and can even prevent injury by detecting the development of a catastrophic failure before it happens(Arena, Collotta, Luca, Ruggieri, & Termine, 2022). Such systems are usually only concerned with detecting faults that are low-frequency and high severity because high-frequency faults suggest that a fundamental design change is needed — or otherwise can be mitigated with regularly scheduled maintenance — and low-cost events are generally not worth the expense of developing and installing a detection system.

Ground vehicles are complex machines with many interrelated systems, and these systems are expected to perform a wide range of tasks in a variety of environments. This complexity makes it challenging for human experts to establish indicators that can be used to consistently determine a vehicle's condition. There are a large number of sensor signals to take into account, each with varying relevance (Giordano et al., 2022). Furthermore, faults often manifest in the relationship between signals rather than in the values of an individual signal, meaning that checking whether a signal has exceeded its nominal range is insufficient. Instead, one can learn the patterns of a fault automatically with a data-driven approach. This involves collecting large amounts of healthy and faulty operating data and creating a model that can distinguish the two.

Unfortunately, it is rare to find data that is cleanly labeled as 'healthy' or 'faulty' because collecting this data often involves running a machine to failure, which can be prohibitively expensive and time-consuming(Theissler, Pérez-Velázquez, Kettelgerdes, & Elger, 2021). Such a procedure also results in a highly skewed dataset since nearly all of the data will be 'healthy,' and it is usually infeasible to account for all possible failure modes. For these reasons, a supervised data-driven classification approach is often untenable for fault detection.

The lack of labeled data motivated us to frame the problem as anomaly detection instead of classification. In anomaly detection, sensor data that is known to be from a period of normal operation is used to model the baseline behavior, then deviation from the baseline behavior can be used as an indicator of the vehicle's condition. If the deviation increases past some threshold, an anomaly is detected - possibly indicating the development of a fault. Broadly speaking, *prognostic health management* (PHM) capabilities are, in increasing order: anomaly detection, diagnostics, and prognostics (Vachtsevanos, Lewis, Roemer, Hess, & Wu, 2006; Goebel et al., 2017). While at the lowest level of PHM capability, anomaly detection is very important in its own right and can,

over time, be used to attain higher levels of PHM capability (Sikorska, Hodkiewicz, & Ma, 2011).

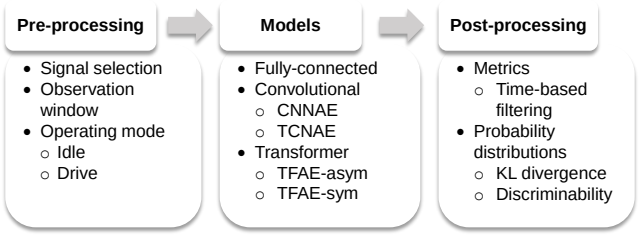

Figure 1. Outline of our approach

The selected approach was based on autoencoders. Traditionally, autoencoders were used for dimensionality reduction and feature learning (Goodfellow, Bengio, & Courville, 2016). Early successful implementation and deployment of autoencoders for anomaly detection predate the emergence of *deep learning* (Japkowicz, Myers, & Gluck, 1995). More recently, autoencoder-based anomaly detectors have been shown to have considerable promise because, unlike classical classifiers that demand balanced datasets, their training can be based on data associated with normal operation, which comes in abundance, as opposed to data associated with failures, which is difficult to come by (Eklund, 2018; Yan & Yu, 2015).

Autoencoders encode their input data in a lower dimensional space and then reconstruct the original input from the compressed representation. This topology encourages the model to learn the most important underlying features and ignore some of the noise (Darban, Webb, Pan, Aggarwal, & Salehi, 2024). For anomaly detection, an autoencoder is trained on healthy data so it is expected to accurately reconstruct other healthy data and poorly reconstruct faulty data. Thus, data samples with high reconstruction errors correspond to anomalies (Giannoulidis, Gounaris, & Constantinou, 2024). However, it is important to note that the training data for an autoencoder must contain all relevant operating and environmental conditions; otherwise, the data samples associated with different operating conditions will also result in a large autoencoder error. Operating and environmental conditions affect measurements and features. Without intelligent feature extraction, the sensitivity to damage is directly proportional to the sensitivity to changes in the operating conditions (Worden, Farrar, Manson, & Park, 2007). Data-driven features, e.g., encodings of autoencoders, require that the training and validation data include representative operating conditions that the model will be exposed to in use. For example, if an autoencoder model is trained using the data associated with low-speed driving conditions only, high-speed driving will likely induce a model anomaly.

This study adopted an iterative approach, which started with setting up an end-to-end workflow consisting of pre-processing, model development, and post-processing, depicted in Figure 1, and then continued with exploring the associated hyperparameter space.

The rest of the document is organized into nine sections. Section 2: *Dataset* describes the format and content of the vehicle operating data provided to us. Section 3: *Signal Selection* provides the domain knowledge that motivated our choice of signals to use in the model. Section 4: *Pre-processing* explains our procedure for cleaning and structuring the raw sensor data. Section 5: *Model* describes the autoencoder architectures. Section 6: *Post-processing* describes how we use the model outputs for anomaly detection and discusses some of the key hyperparameters and variations that were considered at each of these steps. Section 7: *Experiments* explains our observations on how different hyperparameters affected model performance, Section 8: *Results* presents metrics for the best configuration we tested, and Section 9: *Discussion* reviews our interpretation of these results. Finally, Section 10: *Conclusions* recapitulates the key findings and outlines potential future work.

2. DATASET

The original dataset contained operating data for 828 vehicles. Signals from each vehicle's engine control unit (ECU) were collected during operation at a 1Hz sampling rate using the controller area network (CAN) bus interface. These signals are defined by the society of automotive engineers (SAE) J1939 standard, which defines the range, scaling, units, and information included in each signal. The signals collected were a mixture of measured sensor values and ECU calculated variables. Diagnostic trouble codes (DTCs) were also logged and provided to us as a general message describing the issue. Each DTC event logged was accompanied with a timestamp of the event, the associated vehicle subsystem that was the source of the message and the status of that fault, if it was active or inactive. The signals, DTCs, and corresponding timestamps were provided to us in a collection of computable document format (CDF) files.

A log of maintenance events was also provided for each vehicle. The maintenance logs included text descriptions of the fault, corrective actions taken, dates associated with the start and completion of the repair, as well as specific parts and quantities required. All text descriptions were manually written by the vehicle maintainers. We grouped the descriptions by fault type for our own organization, using a combination of manual labeling and clustering via the *sentence-transformers* Python library. We operated on the assumption that engine-related maintenance events corresponded to a real engine-related fault, which might be detectable in the period leading up to the maintenance event.

We selected eight vehicles with engine-related maintenance events for our investigation. The pool of candidate vehicles

Table 1. List of Vehicles

Vehicle ID	Maintenance Event	Fault Comment	Context preference
E01	Engine Overheating	Replaced cooling system parts: thermostat, gasket, filler cap	-
E02	Water Pump Inoperable	Replaced inoperable water pump - unknown failure mode	Idle
E03	Thermostat Failure	Three work orders in short time frame; multiple parts replaced	Idle
E04	Injector Assembly	Four injectors replaced - unknown failure mode.	Idle
E05	#4 Fuel Injector Failure	One injector replaced - unknown failure mode.	Idle
E06	Low Eng. Coolant	Unknown amount of coolant added.	Drive
E07	Low Eng. Oil & Coolant	Unknown amounts of coolant and oil added.	Drive
E08	Low Eng. Oil & Coolant	Unknown amounts of coolant and oil added.	Drive

consisted of those with both an engine-related fault and a corrective action taken to repair the issue. From those candidates, a domain expert prioritized faults that were thought to have a higher probability of success in the fault detection process. This assessment was based on the severity of the fault, a lack of other confounding faults in the maintenance record, available signals, and whether enough data was available for both training and evaluation of the model. This meant that there was sufficient data available such that the operating conditions present in the evaluation period were also covered in the period of healthy training data. The eight selected vehicles and the fault information from the provided maintenance record is listed in Table 1. The table also includes our own comments about each fault and whether we believe the fault is more likely to manifest in idle or driving mode.

A period of data preceding each vehicle's maintenance event was reserved for model evaluation. This evaluation period was determined by an expert based on a safe estimate of how much time and usage that type of fault typically takes to develop. The usage history of vehicle E04 is shown in Figure 2. It cannot be assumed that all of the data inside the evaluation period is faulty, but we expected our models to detect the fault somewhere within that period. We confirmed that there were no relevant maintenance records before the evaluation period, and we assumed that all data preceding the evaluation period was healthy. This healthy baseline data was used for model training or validation.

3. SIGNAL SELECTION

At their core, diesel engines generate a series of combustion events using a mixture of compressed air and diesel fuel oil. The energy from those events is converted to rotational force and heat by the engine. In a healthy engine, for any given engine speed, torque demand, coolant temperature, and intake air mass flow, the ECU will command a given amount of fuel to be injected. The combustion creates the intended amount of torque output from the engine along with a given amount of heat as a byproduct, which is transferred to the surrounding environment via the engine's radiator and exhaust gas. If the engine is experiencing a fault of sufficient magnitude, those levels of torque and heat generated would differ from the output of a healthy engine, which would be detectable.

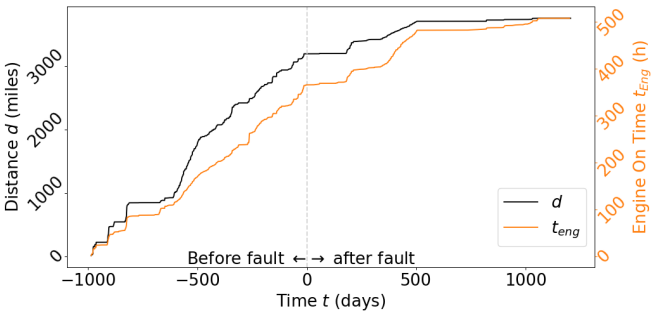

Figure 2. Usage history for vehicle E04

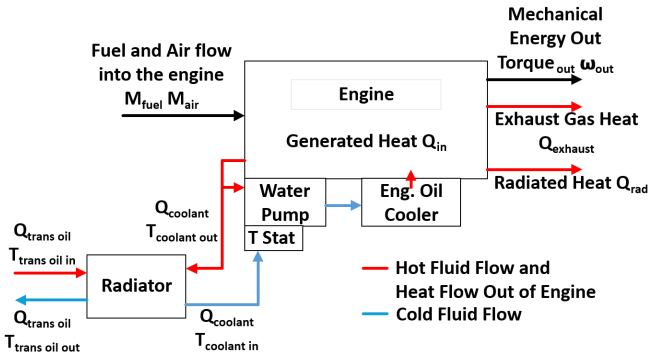

Figure 3. Block diagram of the engine and cooling system

Two engine fault scenarios we expected to be detectable are:

- If the fuel system experiences a fault, the quantity or timing of the fuel injected will change the amount of actual torque and excess heat that the engine generates.
- If the cooling system experiences a fault, its ability to transfer heat away from the engine will be affected, typically resulting in elevated coolant temperatures relative to a healthy engine at the same engine speed and torque.

Signals for the models were selected based on our understanding of the physics governing the operation of the engine, and what was available on the vehicle's ECU. A block

Table 2. List of input signals

Signal	Symbol	Signal Source	Unit	Min	Max
Vehicle Speed	v_v	ECU Measured	mph	0	100
Engine Speed	ω_e	ECU Measured	rpm	0	3000
Engine Torque	τ_e	ECU Calculated	%	0	100
Engine Coolant Temp.	T_{ec}	ECU Measured	°F	-40	300
Engine Coolant Temp. Rate of Change	$\dot{T}_{ec}$	Model Calculated	°F/s		
Engine Fan Speed	v_{ef}	ECU Status	%	0	100
Intake Manifold Temp.	T_{im}	ECU Measured	°F	-30	300
Intake Manifold Pressure	p_{im}	ECU Measured	psi	0	45
Engine Oil Pressure	p_{eo}	ECU Measured	psi	0	140

diagram of the engines cooling system is shown in Figure 3, showing the overall flow of engine coolant and heat through the components of the cooling system. For the engine thermal model there were three core pieces of information we had to understand: the quantity of heat generated, the system temperatures, and the capacity of the cooling sub-system. To monitor the heat generated, the model required engine speed and engine torque, as well as intake manifold temperature and pressure. Engine torque is available in several forms on the ECU. Of those available, *Actual Engine Percent Torque* is the signal used here. This represents the total percentage of torque that the engine is producing in relation to its defined reference torque, including all torque requests from the driver, as well as other vehicle and engine requests. System temperatures are represented by the engine coolant temperature and the calculated rate of change of engine coolant temperature. Finally, vehicle speed and engine fan speed were used as proxies for mass airflow over the engine's radiator (the primary source of heat rejection for the engine). Access to ambient temperature would have been a welcomed addition to the set of signals, but was not available to us.

We collected 30 or 60 second windows of data from each signal as input to the model. The lengths of the observation windows were chosen to accommodate an inherent delay in the thermal response to changes in engine speed and torque due to the thermal mass of the engine and its cooling system. In a steady operating condition, it can take minutes to reach thermal equilibrium, but in normal operation, the engine's speed and torque behave much more dynamically (even the provided data rate of 1 Hz removes some of the dynamic nature of the engine behavior). The length of the observation window needed to be long enough to capture some portion of the thermal response. However, longer windows require larger models, and cause more data to be dropped during preprocessing. We chose the observation window length to balance these considerations.

4. PRE-PROCESSING

Before training models, it was necessary to clean the data and package it in a more accessible way. This routine is outlined at a high level in Algorithm 1. First, a plausible range of values for each signal was determined, as shown in Table 2. Signal values outside the range were treated as sensor glitches and deleted. We deleted any data associated with duplicate timestamps, except for cases in which a duplicate timestamp was immediately followed by a gap of matching size. In such cases, the repeated timestamp was adjusted to obtain a continuous sequence. Individual missing values in a signal were imputed by linear interpolation, and gaps of size 2 were imputed by copying the values from either side. Files containing a decreasing time step were considered corrupt and discarded entirely. Cumulative metrics were calculated for each timestamp, such as total distance driven, total engine-on time, and total driven time. These metrics were each centered such that the maintenance event fell at zero.

The engine coolant temperature rate of change was not part of the original dataset so it had to be estimated. The J1939 specification defined the resolution of the engine coolant temperature to be 1 degree Celsius. The resulting rate of change was a flat line of zero degrees/second with sporadic spikes to 1 degree/second at the data points where the temperature steps were present. We applied a Gaussian filter to smooth the signal before calculating the rate of change, resulting in a more realistic representation that showed the constant fluctuation of the coolant temperature.

To construct a set of data samples for model development, we split all continuous stretches of signal data into windows with lengths equal to our observation window (30 or 60 seconds). Either a sliding window or consecutive window approach was used, depending on the experiment. Since the ECU data was recorded at 1Hz, a 30-second window would capture 30 consecutive values from each signal; then, if five signals were chosen for the model, for example, one data sample would contain 150 floating point values total. Each resulting sample was added as a row in a Pandas dataframe (McKinney, 2022) alongside its starting timestamp and cumulative metrics.

We further divided the data from each vehicle into idle and driving modes in order to model each mode separately. We defined the 'idle' mode as having vehicle speed of 0 mph and engine speed between 400 and 900 rpm. We considered the

Algorithm 1 Pre-Processing Routine

Require: Multivariate time-series data D, window length M, operating condition OC
Ensure: M-Windows of cleaned data satisfying OC
1: Remove data with invalid timestamps
2: Repair repeated timestamps
3: Impute missing signal values
4: Impute short runs of signal outliers
5: Remove long runs of signal outliers
6: Divide remaining chunks of time-consecutive data into M-second (possibly sliding) windows
7: **for** each window **do**
8: Compute and insert virtual signals
9: Accept windows satisfying OC
10: Adjust rejected window bounds to borrow samples from neighbors
11: Accept adjusted windows satisfying OC
12: **end for**
13: **return** Accepted windows

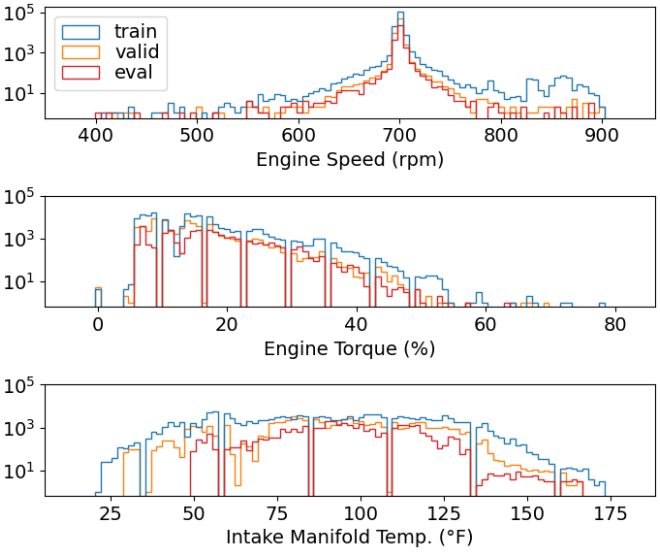

Figure 4. Distribution of signal values for vehicle E07

vehicle to be 'driving' if the vehicle speed was at least 2 mph and the engine speed was at least 400 rpm. When constructing a set of samples for one of these operating modes, we checked that these conditions were met at all timestamps in a sample, and the whole sample was discarded if the check failed. If too many samples were discarded this way, we performed an additional step; for each discarded sample, we checked whether the window bounds could be shifted left or right to overlap an adjacent window and construct a new sample that met the idle or driving conditions.

We could not always split the healthy period into a contiguous training set followed by a contiguous validation set because the resulting distributions of operating data of the two sets were too different. On the other hand, randomly selecting samples for training and validation opens the potential for data leaks because adjacent data samples are similar. To mitigate data leaks and ensure that the full range of operating conditions are represented in both training and validation sets, we first split the data into consecutive chunks of samples and then randomly assigned each chunk to either the training or validation set.

It was important to verify that our training set covered the range of operating and environmental conditions seen in the evaluation period. This was because the autoencoder would flag novel operating and environmental conditions as anomalies. Figure 4 shows how we plotted histograms of each signal to verify that the full range of values was represented.

We scaled each signal separately before model training, either by normalization or standardization. Normalization involved subtracting the minimum value of the signal in the training set and dividing by the range (i.e. the maximum minus the minimum). Standardization involved subtracting the mean value of the signal in the training set and dividing by the standard deviation. The choice of scaling procedure was treated as a hyperparameter.

5. MODEL

For each vehicle we trained a neural network autoencoder on data from healthy operating periods so that the autoencoder could be expected to produce low reconstruction error on data from other presumably similar healthy periods and high error on faulty periods. The autoencoder was trained to minimize the mean squared error (MSE) between the input and output. Five main autoencoder architectures were investigated; fully-connected autoencoders (FCAE), convolutional autoencoders (CNNAE), temporal convolutional network autoencoders (TCNAE), and two types of transformer-based autoencoders (TFAE-asym, TFAE-sym). We also tried implementing the popular TranAD architecture (Tuli, Casale, & Jennings, 2022), but had difficulty adapting it to our particular dataset.

The fully-connected model maps the input vector $\boldsymbol{x}$ to the output vector of the same dimensions $\boldsymbol{y}$ via $n-1$ hidden layers (Figure 5)

$$\boldsymbol{x} \mapsto \boldsymbol{h}_1 \mapsto \boldsymbol{h}_2 \mapsto \ldots \mapsto \boldsymbol{h}_{n-1} \mapsto \boldsymbol{y} \equiv \hat{\boldsymbol{x}} \tag{1}$$

The model parameters were organized in a set of n weights and biases $\{(\boldsymbol{W}_k, \boldsymbol{b}_k)\}_{k=1}^{n}$ for fully connected layers. Activation functions $\sigma()$ associated with the hidden layers were $\mathrm{ReLU}()$,

$$\mathrm{ReLU}(z) = \max(z, 0) \tag{2}$$

and the output layer was linear. The output of a fully-connected model was

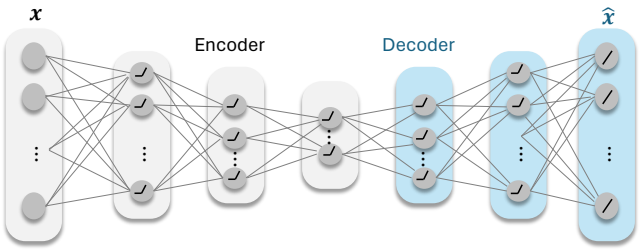

Figure 5. Conceptual diagram of an autoencoder with fully-connected layers (FCAE)

$$\boldsymbol{y} = \boldsymbol{W}_{n-1} h_{n-1} + \boldsymbol{b}_n \tag{3}$$

where $\boldsymbol{h}_{n-1}$ was the last hidden layer. The hidden layers were

$$\begin{aligned} \boldsymbol{h}_1 &= \mathrm{ReLU}\left(\boldsymbol{W}_o \boldsymbol{x} + \boldsymbol{b}_o\right) \\ &\vdots \\ h_{n-1} &= \mathrm{ReLU}\left(\boldsymbol{W}_{n-2} \boldsymbol{h}_{n-2} + \boldsymbol{b}_{n-1}\right) \end{aligned} \tag{4}$$

The additional topology-related hyperparameters that required specification were the lengths of hidden layers

$$N_k = \dim\left(\boldsymbol{h}_k\right) \quad k \in \{1, 2, \ldots, n-1\} \tag{5}$$

and dropout regularization parameters for each of the hidden layers. Dropout and batch normalization were applied after every layer except for the final output layer, before the activation function.

The convolutional neural network autoencoder (CNNAE) architecture (Figure 6) began with some number of convolutional blocks, ended with an equal number of deconvolutional blocks, and had a fully-connected model in between (as defined earlier). A convolutional block consisted of a depth-wise convolution layer that applied filters to each signal separately to increase the number of channels, followed by max pooling. A deconvolutional block performed nearest neighbor upsampling, followed by a depth-wise convolution layer that decreased the number of channels. All blocks except the final (output) deconvolutional block performed dropout, batch normalization and ReLU. The motivation for this architecture was to learn features from each signal waveform separately via the convolutional layers and then learn relationships among the signals via the fully-connected layers.

Another convolutional architecture was explored, called a Temporal Convolutional Network Autoencoder (TCNAE). The dilated convolutions employed by this architecture aim to model long temporal patterns more naturally than traditional convolutions (Darban et al., 2024). We followed the design offered by (Thill, Konen, Wang, & Bäck, 2021), using the python library *pytorch-tcn* (Krug, 2023) to implement the dilated convolutional layers.

Motivated by the success of transformers in modeling sequential data (Vaswani, Shazeer, Parmar, Uszkoreit, & Jones, 2017), two transformer-based autoencoder architectures were explored. The first architecture, which we refer to as TFAE-asym, followed the implementation in (Liang, Knutsen, Vanem, Æsøy, & Zhang, 2023) in which the output from a series of transformer encoder blocks was flattened and passed through a series of fully-connected layers. Because this architecture has residual connections that could allow the model to learn the identity function, we chose one of the fully-connected layers to act as a bottleneck by setting its size to be smaller than the size of the input data.

The second transformer architecture (TFAE-sym, Figure 7) similarly begins with a series of transformer encoder blocks, but instead of fully-connected layers, a series of transformer decoder blocks were introduced to reconstruct the data. We again addressed the problem of the residual connections by introducing a fully-connected bottleneck. Following the pattern of the other fully-connected layers in the transformer model, which are applied positionwise instead of flattening the data, so too are these bottleneck layers applied to each timestep separately.

All models were implemented in *pytorch* (Paszke et al., 2019). Final hyperparameters are listed in Table 7 in the appendix.

6. POST-PROCESSING

From a trained autoencoder, we compute the reconstruction error and compute a Condition Indicator (CI) over that error sequence. In this paper, the CI is the rolling average of the mean absolute error (MAE) of each model input sample. MSE was also considered but showed no improvement, and MAE is more interpretable. The threshold for anomaly detection is the maximum of this CI over the baseline samples (training and validation) plus some percent margin to prevent false positives (Figure 9). An anomaly is detected when the CI crosses the threshold.

We compared the distributions of MAE values between training, validation, and evaluation periods, using Kullback-Leibler (KL) divergence D_{KL} (Bishop, 2006). For example, the KL divergence between the estimated training error distribution $\hat{p}_t(MAE)$ and validation error distribution $\hat{p}_v(MAE)$ is given by

$$D_{KL} = -\int \hat{p}_t(MAE) \ln\left(\frac{\hat{p}_v(MAE)}{\hat{p}_t(MAE)}\right) d(MAE) \tag{6}$$

The estimated error distributions are shown in the middle subplot of Figure 8. They were using estimated using *kernel density estimation* (KDE) where the distribution $\hat{p}(MAE)$ is a scaled sum of kernels $g(.)$, centered at individual N data

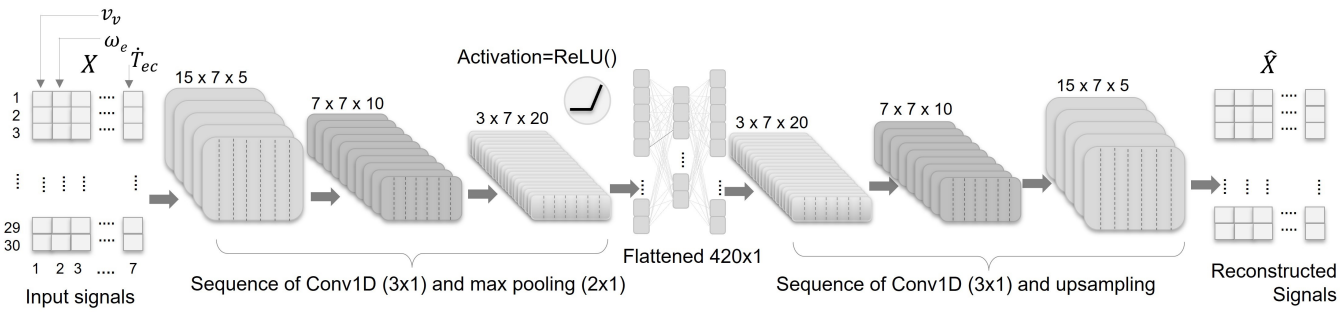

Figure 6. A block diagram of a convolutional autoencoder (CNNAE)

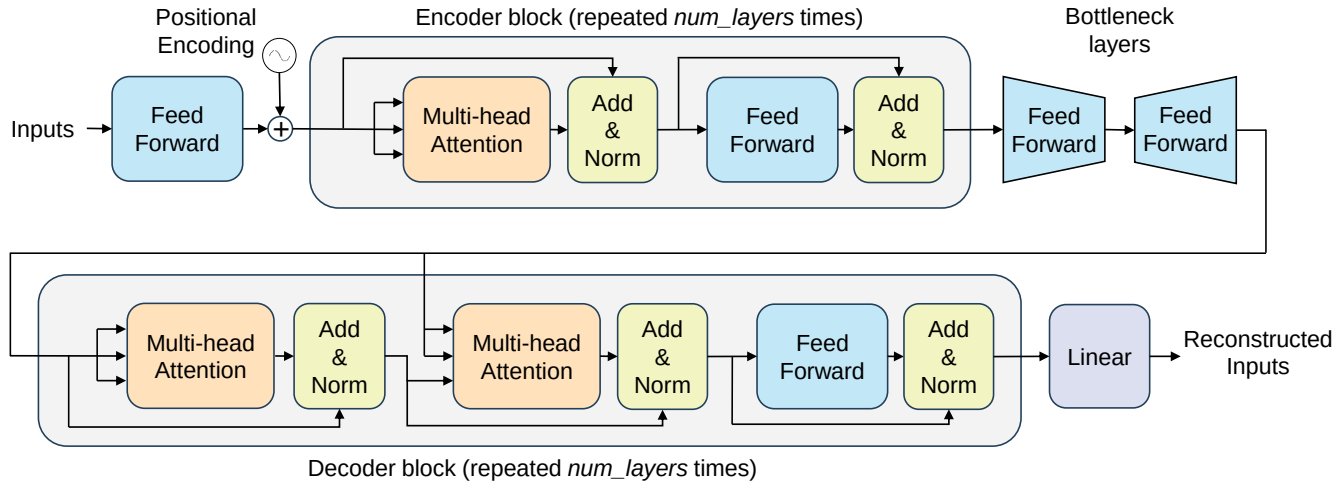

Figure 7. A block diagram of a symmetric transformer autoencoder (TFAE-sym)

points MAE_n

$$\hat{p}(MAE) = \frac{1}{N\Delta(MAE)} \sum_{n=1}^{N} g\left(\frac{MAE - MAE_n}{\Delta(MAE)}\right). \tag{7}$$

We used *scipy*'s (Virtanen et al., 2020) Gaussian KDE from *scipy*'s *stats* module, with default parameters.

The divergence between training and validation MAE distributions $D_{KL}(t, v)$ should be low, indicating that the model generalizes well, and the divergence between validation and evaluation MAE distributions $D_{KL}(v, e)$ should be high, indicating the development of a fault. Since it was also informative to note the mean shift between the validation and evaluation MAE distributions, we measured the discriminability (Duda, Hart, & Stork, 2001) between them as:

$$d(e, v) = \frac{|\mu_e - \mu_v|}{\sqrt{\sigma_e^2 + \sigma_v^2}} \tag{8}$$

Figure 8 illustrates a sample model evaluation of a single-vehicle model. The leftmost subplot shows the evolution of the autoencoder error leading up to a maintenance event. Zero hours (t=0) corresponds to the maintenance event. The dots denote individual samples with blue, orange, and red correspond to training, validation, and evaluation respectively. The solid line is the filtered error, using a rolling average $\langle MAE \rangle_{N=20}$. The maximum of the baseline $\langle MAE \rangle$ is shown for reference, and in this example a 100% margin is added to determine the detection threshold. The middle subplot shows the error distributions, and the rightmost subplot shows the corresponding mean plus and minus one standard deviation width ($\mu \pm \sigma$). The KL divergence between training and validation error distributions $D_{KL}(t, v) = 0.01$ was much smaller than the KL divergence between evaluation and validation error distributions $D_{KL}(v, e) = 0.21$, as expected for successful detection.

7. EXPERIMENTS

Exploration of the hyperparameter space spanned across all stages of the work flow: pre-processing, model, and post-processing. Each stage had its own set of parameters, and heuristic optimization of this set of parameters was an iterative and time-consuming process. To simplify this process

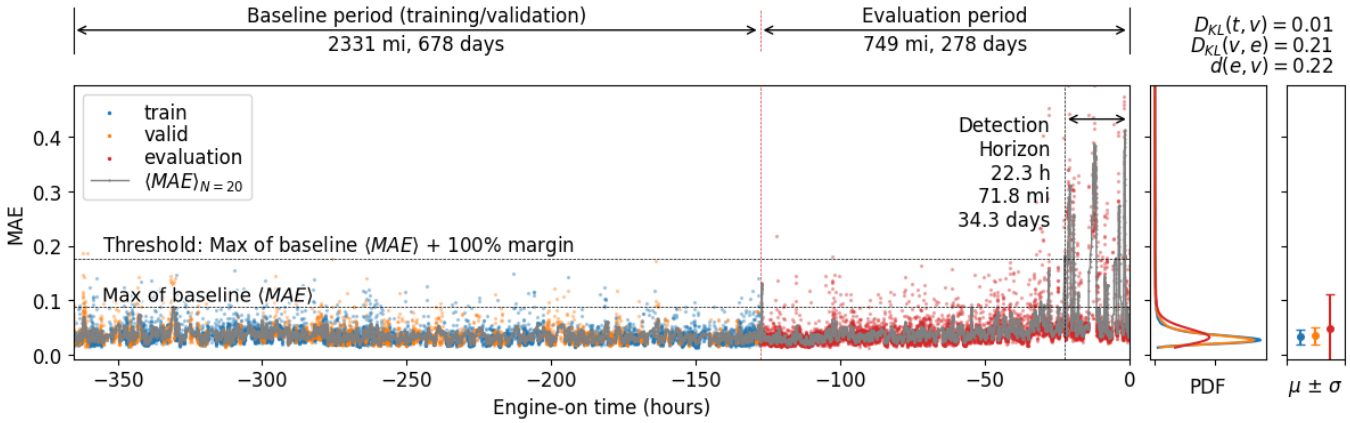

Figure 8. Example model evaluation (vehicle E04, idle data, TFAE-sym)

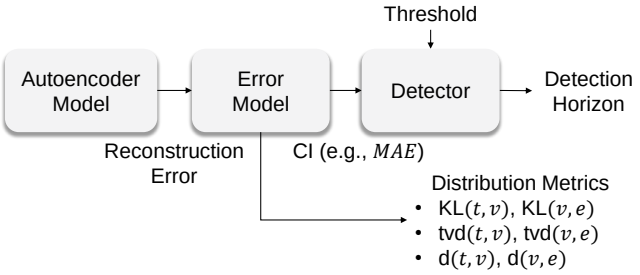

Figure 9. Post-Processing Block Diagram

we devised experiments where different combinations of parameters were used to train the models. In each experiment, we trained a variety of models for each vehicle and compared the performance against previous experiments; we looked for detection on new vehicles and for improved performance on vehicles that showed good detection previously. Based on the results of these experiments, a final set of hyperparameters was selected. The results from this experiment are discussed in the next section.

The key hyperparameters in the pre-processing stage were operating mode, choice of input signals, and observation window length. Models trained on the entire dataset (idle and driving data together) resulted in bimodal error distributions, which indicated the idle and driving modes. Hence, we split the data into idle and driving subsets, and subsequent experiments were carried out on each mode separately.

We saw significant improvements in model performance by separating idle and driving data. The data collected during the idle operation condition benefited from the fact that it is a singular fixed operating point with a set engine speed and a relatively fixed load that has a dependency on the engine operating temperature. This stability is beneficial for model performance as this state is both highly repeatable and frequently seen. Data collected when the vehicle is actively being driven has to cover a wide range of engine speeds and torque loads. The advantage of the driving mode is that the differences between the healthy state and a faulted state for a number of failure modes are increased or only occur as a result of the additional load of driving.

The choice of signals also had a strong effect on model performance. Nine signals were initially identified as relevant to the thermal system of these vehicles, and we investigated several subsets. Some signals were judged to be too sparse, such as Engine Fan Speed, which was zero at the large majority of points. That signal also contained frequent sensor glitches across vehicles, so leaving it out resulted in more usable data — this alone improved some models. Some signals were regarded as fundamental and so were included in all models, such as the Engine Coolant Temperature, its rate of change, Engine Speed, and Engine Torque.

Initial experiments used a 30-second observation window, but we also experimented with a longer 60-second window. The motivation for expanding the observation window was to potentially capture more of the dynamics in the slow-moving temperature signal. No significant improvement was observed, and some models suffered because dividing our data into longer windows resulted in more data discarded during pre-processing and fewer total training samples. To bolster the number of samples for training we switched from the consecutive-window approach to a sliding-window when using the 60-second observation window. None of these variations showed improvement over the initial consecutive 30-second approach.

Hyperparameters for model development included the scaling method, model architecture, regularization, number and width of layers, learning rate, and batch size. No difference in performance was observed between the normalization and standardization scaling methods. The choice of model architecture was only somewhat impactful; the highest threshold margins were achieved by the TFAE-sym architecture, but in

general, models for a particular vehicle tended to perform either strongly or poorly regardless of the architecture.

Models tended to overfit if their number of parameters was large, especially for vehicles with less training data. Our strategies for mitigating overfitting included dropout, adding Gaussian noise, reducing model size, and early stopping. A high dropout ratio caused models to converge at a higher validation error, so it was limited to 10% if used at all. Adding a small amount of Gaussian noise to the input data also helped prevent overfitting but otherwise we observed no improvement in either validation error or anomaly detection. It was also effective to simply stop model training when the validation loss began to trend upwards. The number of layers, layer widths, learning rate, and batch size were adjusted as necessary per model, but none showed a consistent overall relationship to model detection.

Post-processing hyperparameters defined the procedure for converting raw model errors into condition indicators for anomaly detection. They included considerations such as MAE vs MSE and the number of filter taps. The choice of using MAE or MSE did not show any overall difference in performance, so MAE was chosen for ease of interpretability. We generally used 20 taps for the running mean to smooth out short-term spikes in MAE, but this number could be adjusted freely as needed. Greatly increasing the number of taps does become a practical issue for deployed models running in real-time because it adds an initial delay to the prediction and assumes continuity that may not be present.

8. RESULTS

Here we report results for two pre-processing configurations that we found to have the best-performing models overall, one for idle mode and one for driving mode. The idle model used five signals: Engine Speed, Torque, Engine Coolant Temperature, Engine Coolant Temperature Rate of Change, and Intake Manifold Temperature. The driving model used the same five signals plus Vehicle Speed (six total). Both used consecutive 30-second observation windows, a symmetric transformer architecture, and 20 taps for MAE smoothing.

Table 3. TFAE-sym detection per vehicle in idle mode

ID	Horiz.	Margin	$D_{KL}(v,e)$	$d(e,v)$	Verdict
E01	33.6 h	24%	0.32	0.33	yes
E02	-	0%	0.04	0.08	no
E03	30.2 h	53%	0.05	0.04	yes
E04	21.0 h	225%	0.21	0.22	yes
E05	-	0%	1.61	0.09	no
E06	213.5 h	82%	0.04	0.15	yes
E07	-	0%	0.34	0.07	no
E08	-	0%	0.06	0.07	no

Table 3 and Table 4 show performance metrics for the TFAE-sym architecture on each vehicle. The recorded metrics are the detection horizon in hours of engine-on time, the maximum possible margin that allows detection at that detection horizon, the KL divergence between validation and evaluation MAE distributions, and the discriminability between those distributions. The verdict column shows our interpretation of whether we are confident that the model successfully detected the fault (yes), or if we believe there is an alternate explanation for the detected anomaly (no).

Table 4. TFAE-sym detection per vehicle in driving mode

ID	Horiz.	Margin	$D_{KL}(v,e)$	$d(e,v)$	Verdict
E01	-	0%	0.44	0.20	no
E02	177.5 h	3%	0.04	0.09	no
E03	41.9 h	120%	0.19	0.38	yes
E04	31.9 h	39%	0.08	0.26	yes
E05	-	0%	0.29	0.29	no
E06	159.7 h	170%	0.06	0.24	yes
E07	-	0%	0.07	0.00	no
E08	13.9 h	22%	0.40	0.47	yes

Table 5. Detection per architecture on E04 in idle mode

Mod	Horiz.	Margin	$D_{KL}(v,e)$	$d(e,v)$
FCAE	21.0 h	146%	0.19	0.26
CNNAE	21.0 h	59%	0.06	0.19
TCNAE	21.0 h	143%	0.14	0.25
TFAE-asym	21.0 h	120%	0.13	0.24
TFAE-sym	21.0 h	225%	0.21	0.22

Table 5 compares the performances of different autoencoder architectures on specifically vehicle E04 in idle mode. All models were able to detect this fault at the same detection horizon, but the TFAE-sym architecture had the highest maximum possible margin. This trend held for the other vehicles as well, with TFAE-sym achieving higher margins than other architectures.

In an ideal case, we would expect the $\langle MAE \rangle$ to increase monotonically leading up to the fault. There would then be a trade-off between the detection horizon and margin where decreasing the margin would result in an earlier detection and vice versa. Our models generally did not align with this archetype, instead showing distinct spikes of $\langle MAE \rangle$ that cross the detection threshold briefly. The timing of the major spikes was often consistent across model runs, so measuring the maximum margin at the earliest major spike was useful to us as an indicator of model quality. For example, the evaluation plot in Figure 8 shows a threshold with a margin of 100%, but that margin could be increased as far as 225% with only slightly shorter detection horizon.

9. DISCUSSION

An anomaly detected by an autoencoder requires further interpretation because an anomaly could either be an indication of a fault or simply an unrecognized operating condition. This was particularly likely to occur in our dataset because the vehicles were operated infrequently. Notably, several of the vehicles suffered from a lack of data across seasons, which caused our models to flag unseen seasonal temperature changes as anomalies, thereby obscuring any true detection of faults. For vehicles where the baseline period did not span all seasons, we shrank the evaluation period until the full range of temperature values was represented in the training set. If this was not possible we discarded the vehicle since we could not be confident that its detected anomalies were truly indicative of a fault.

For example, initial models for vehicle E07 appeared to be a resounding success according to the metrics; the model flagged an anomaly early, with high margin, and the MAE distribution in the evaluation period was markedly different from the baseline. However, closer inspection revealed that seasonal temperature changes were the source of the anomaly. The baseline period spanned from April to November and the evaluation period spanned from November to July, meaning that the model had been trained without any data from the colder winter months. We observed that the evaluation period contained low values of intake manifold temperature that were not represented in the training set, which corresponded exactly with the major MAE spikes. Thus, we shifted the end of the baseline period forward into March, verified that the range of signal values was represented in the training data, and trained models again. The new idle and driving models for vehicle E07, as seen in Table 3 and Table 4, did not detect at all.

We sometimes also passed a negative verdict if we judged that a model detected too early and too briefly, in a way that was not consistent with the expected behavior of the fault. As human observers we were skeptical when we saw a model's $\langle MAE \rangle$ cross the detection threshold only once, briefly, and then return to normal levels. This was the case with models for vehicle E02, which briefly crossed the detection threshold long before the fault and soon returned to baseline levels. In general, multiple brief detections in succession were more convincing, as were long periods spent above the threshold (or even a long period at elevated MAE levels), especially if this behavior occurred leading up to the fault time. Confidence of a true fault detection was further increased if that vehicle showed detection on both the idle and driving models. Our process required a human in the loop for interpretation, making detection verdicts somewhat subjective.

This analysis is complicated by the fact that some faults only manifest under certain operating conditions; for example, a thermostat failure would likely only reveal itself after the vehicle reaches the temperature at which the thermostat is meant to open. Such under-representation of operating conditions is one possible explanation for why the model MAE might return to normal levels after crossing the threshold.

A single metric proved to be insufficient to serve as an objective measure of model quality. As our primary condition indicator, $\langle MAE \rangle$ crossing a threshold with a high margin was promising, but it was also important that $KL(v, e)$ and $d(e, v)$ were high. Even if $\langle MAE \rangle$ never exceeded the validation threshold, markedly different validation and evaluation distributions would indicate a lasting change, possibly representing a developing fault. Conversely, $\langle MAE \rangle$ crossing the threshold while the distributions remain similar could be cause for suspicion. Early detections were generally better, but if the model did not also detect close to the fault time then it was less convincing. It was not clear how to combine these considerations into a single representative score, so models were always subject to human inspection and interpretation.

A limitation in working with this dataset was the lack of crisp ground-truth labels. We relied on the maintenance logs for some amount of ground truth, i.e. whether and when an engine fault occurred, but previous work on this dataset identified some flaws with this approach (Bond et al., 2020). For example, maintenance may have been conducted when there was no fault, the maintainer may have incorrectly diagnosed the fault, the corrective action may have failed to repair the underlying fault, or a fault may have occurred for which no maintenance was performed.

Our work encountered some further limitations with this dataset. We observed that maintenance logs for some vehicles were likely to be incomplete, as evidenced by gaps as long as 2 years. The fault descriptions and corrective narratives were often short and vague, so it was difficult to determine exactly what happened or how severe was the fault. The 1Hz sample rate meant that the more dynamic signals lost resolution, which motivated our decision to investigate thermal-related faults because temperature changes occur over a longer time scale. Some variables like ambient temperature were missing from the dataset, which is a significant gap in any model of a vehicle's thermal system. And some vehicles had less than one calendar year of data before the evaluation period, which introduced the possibility of flagging seasonal temperature changes as an anomaly.

Table 6 shows the number of days and operating hours leading up to the maintenance event for each vehicle. Many vehicles have data recorded across several years, but only a fraction of those days show the vehicle in operation ('Active'). For some vehicles there are spans of months or years where there is no recorded operating data at all. Table 6 also shows that these vehicles spend a large portion of their operating time in idle mode. The limited or deficient coverage of all possible operating and environmental conditions is only fur-

Table 6. Pre-fault usage statistics

Vehicle	Calendar Days		Active Hours		Miles
	Elapsed	Active	Idle	Driving	
E01	1,405	200	223	100	2,043
E02	505	211	538	61	817
E03	265	118	82	63	1,078
E04	981	322	207	158	3,080
E05	415	116	326	210	6,179
E06	854	253	238	142	2,770
E07	455	126	74	46	1,165
E08	382	103	54	55	1,699

ther exacerbated when the data is split into training, validation, and evaluation sets.

10. CONCLUSION

This work investigated detection of high-severity and low-frequency engine faults in ground vehicles. Scarcity of data associated with such faults along with sporadic usage of these vehicles were the biggest challenges. To overcome these challenges a multi-step approach was proposed. First, a study of the system is carried out to identify most relevant input signals. Second, an autoencoder-based anomaly detection was used to isolate the anomalous data. Finally, the anomalous data was evaluated based on multiple metrics.

We found that pre-processing considerations affected model performance more than the autoencoder architecture and the model hyperparameters. Namely, the decision to model the idle and driving modes separately had a strong impact, as did the choice of signals to use in the model. Though we were conscious of the necessity for widely covered operating and environmental conditions, we could not always anticipate the nature of these conditions. As we explored different data filters and signal sets, previously unknown data limitations were an early pitfall that caused models to flag novel operations conditions as anomalies. A manual review of these detections led us to dismiss some anomalies and gain confidence about others.

Future work will extend the findings along three main paths: investigating fleet models, improving post-processing, and developing elements of diagnostics. For example, the fleet models may be trained and validated using healthy data segments from eight to twelve vehicles, where a subset of randomly selected vehicles will provide the training data, and the remaining vehicles will provide the validation data. The models will then be evaluated on vehicles with faults, and their performances will be compared to those of single-vehicle models. The post-processing can potentially improve upon using a moving average and a threshold by exploring more advanced statistical methods, such as those based on likelihood ratios or Bayesian reasoning. In addition, there is an opportunity to add objectivity to fault interpretation to facilitate decision support. Decomposition and subsequent analyses of MAE by signals may provide insights into the anomaly. In addition, as stated above, autoencoders are traditionally used for feature learning. The innermost layer – the encoding or bottleneck – presumably has the most compact representation of the data and can be used as an input for diagnostics development.

FUNDING

This research was sponsored by the Department of the Navy, Office of Naval Research under ONR award number N00014-19-1-2600.

DISCLAIMER

NOMENCLATURE

AE	autoencoder
CAN	controller area network
CDF	computable document format
CI	condition indicator
CNNAE	convolutional autoencoder
DTC	diagnostic trouble codes
d	discriminability
D_{KL}	Kullback-Leibler divergence operator
e	data samples associated with evaluation
E##	Engine-faulted vehicle ID
ECU	engine controller unit
FCAE	fully-connected autoencoder
KL	Kullback-Leibler (divergence)
MAE	mean absolute error
MSE	mean square error
μ	mean
p_{eo}	engine oil pressure signal
p_{im}	intake manifold pressure signal
$\hat{p}()$	estimated probability density function
ReLU	rectifier linear unit (activation function)
σ	standard deviation
t	data samples associated with training
T_{ec}	engine coolant temperature signal
T_{im}	intake manifold temperature signal
$\dot{T}_{ec}$	engine coolant rate of change signal
TCNAE	temporal convolutional autoencoders
TFAE	transformer-based autoencoders
tvd	total variation distance
τ_e	engine torque signal
v	data samples associated with validation
v_v	vehicle speed signal
v_{ef}	engine fan speed signal
ω_e	engine speed signal

REFERENCES

Arena, F., Collotta, M., Luca, L., Ruggieri, M., & Termine, F. G. (2022). Predictive maintenance in the automotive sector: A literature review. *Mathematical and Computational Applications*, *27*(1).

Bishop, C. M. (2006). Relative entropy and mutual information. In *Pattern recognition and machine learning* (p. 55-58). Springer.

Bond, W. G., Dozier, H., Arnold, T. L., Silas, A., Shukla, I., Dong, Q. T., ... Mize, C. H. (2020). A hybrid learning approach to prognostics and health management applied to military ground vehicles using time-series and maintenance event data. In *Proceedings of the annual conference of the prognostics and health management society.*

Darban, Z. Z., Webb, G. I., Pan, S., Aggarwal, C. C., & Salehi, M. (2024). Deep learning for time series anomaly detection: A survey. In *Acm computing surveys. association for computing machinery (acm).*

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). Signal detection theory and operating characteristics. In *Pattern classification* (Second ed., p. 48-51). John Wiley.

Eklund, N. (2018). Anomaly detection tutorial. In *Proceedings of the annual conference of the prognostics and health management society.*

Giannoulidis, A., Gounaris, A., & Constantinou, I. (2024). Exploring unsupervised anomaly detection for vehicle predictive maintenance with partial information. In *Proceedings of the 27th international conference on extending database technology (edbt).*

Giordano, D., Giobergia, F., Pastor, E., La Macchia, A., Cerquitelli, T., Baralis, E., ... Tricarico, D. (2022). Data-driven strategies for predictive maintenance: Lesson learned from an automotive use case. *Computers in Industry*, *134*, 103554.

Goebel, K., Daigle, M. J., Saxena, A., Roychoudhury, I., Sankararaman, S., & Celaya, J. R. (2017). *Prognostics: The science of making predictions*. CreateSpace Independent Publishing Platform.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Japkowicz, N., Myers, C., & Gluck, M. (1995). A novelty detection approach to classification. In *Ijcai* (Vol. 1, pp. 518–523).

Krug, P. (2023). *Pytorch-tcn: Streamable (real-time) temporal convolutional networks in pytorch.* `https://github.com/paul-krug/pytorch-tcn`. (GitHub repository)

Liang, Q., Knutsen, K. E., Vanem, E., Æsøy, V., & Zhang, H. (2023). Unsupervised anomaly detection in marine diesel engines using transformer neural networks and residual analysis. *Proceedings of the Asia Pacific Conference of the PHM Society*, *4*(1).

McKinney, W. (2022). *Python for data analysis*. " O'Reilly Media, Inc.".

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, *25*(5), 1803-1836.

Theissler, A., Pérez-Velázquez, J., Kettelgerdes, M., & Elger, G. (2021). Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. *Reliability Engineering & System Safety*, *215*, 107864.

Thill, M., Konen, W., Wang, H., & Bäck, T. (2021). Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing*, *112*.

Tuli, S., Casale, G., & Jennings, N. R. (2022). Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment*, *15*.

Vachtsevanos, G., Lewis, F. L., Roemer, M., Hess, A., & Wu, B. (2006). Intelligent fault diagnosis and prognosis for engineering systems. In *1st ed. hoboken.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., & Jones, L. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000-6010.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... others (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, *17*(3), 261–272.

Worden, K., Farrar, C. R., Manson, G., & Park, G. (2007). The fundamental axioms of structural health monitoring. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *463*(2082), 1639–1664.

Yan, W., & Yu, L. (2015). On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Proceedings of the annual conference of the prognostics and health management society.*

Ethan A. Kohrt received his B.S. in Computer Science from Furman University (Greenville, SC, USA) in 2021 and his M.S. in Computer Science from Northwestern University (Evanston, IL, USA) in 2022. In 2023 he joined the Golisano Institute for Sustainability where he is currently a Data Science Engineer.

Matthew E. Moon received his B.S. in Computer Engineering and Mathematics from Rose-Hulman Institute of Technology (Terre Haute, IN, USA) in 2018 and his MEng in Electrical Engineering from Rice University (Houston, TX, USA) in 2019. He joined the Golisano Institute for Sustainability (GIS) at Rochester Institute of Technology in 2023 where he is currently working as a Data Science Engineer.

Matthew R. Sullivan received his B.S. in Engineering Physics from Saint Bonaventure University (Olean, NY, USA) in 2002 and his Ph.D. in Mechanical Engineering from the State University of New York at Buffalo (Buffalo, NY, USA) in 2010. He joined the Golisano Institute for Sustainability (GIS) at Rochester Institute of Technology in 2019, where he is currently a Senior Prognostics and Health Management Engineer. His research interests include condition based maintenance and prognostics, and asset health management. He was previously an engineer at Torvec Inc. where he worked on development of prototype automotive driveline components and hydraulic pumps and motors. He holds one patent for electron transport in an atomic scale magnetoresistive device.

Sri Das received her B.S. in Electronics Engineering from J.N.T.U. (Hyderabad, India) and her M.S. (in Electrical Engineering) & Ph.D. (in Microsystems Engineering) both from Rochester Institute of Technology (Rochester, NY, USA). After specializing in Data Science & Machine learning, she worked at Megalabs (Palo Alto, CA) before joining the Golisano Institute for Sustainability (GIS) at Rochester Institute of Technology in 2023, where she is currently a Data Science Engineer. Her prior background includes postdoctoral research experience, business development, consulting for start-ups, counseling small business owners and cofounding her own small business. She has co-authored several peer-reviewed publications.

Michael G. Thurston received his B.S. and M.S. in Mechanical Engineering from Rochester Institute of Technology (Rochester, NY, USA) in 1988, and his Ph.D. in Mechanical and Aerospace Engineering from the University of Buffalo (Buffalo, NY, USA) in 1998. He is the Technical Director and Research Associate Professor at the Center of Integrated Manufacturing Studies at Rochester Institute of Technology. He formerly held positions in air conditioning system development at General Motor and Delphi, and as a Researcher at the Applied Research Laboratory at Penn State University. He holds 7 patents in the areas of air conditioning and asset health monitoring. His research interests include: sustainable design and production, condition based maintenance and prognostics, and asset health management. He is a member of the Society of Automotive Engineers, and was awarded the Boss Kettering Award for product innovation by Delphi.

Nenad G. Nenadic received his B.S. in Electrical Engineering from University of Novi Sad (Novi Sad, Serbia) in 1996 and his MS and Ph.D. in Electrical and Computer Engineering from University of Rochester (Rochester, NY, USA) in 1998 and 2001, respectively. He joined Kionix Inc. in 2001, where he worked on development of microelectromechanical inertial sensors. Since 2005, he has been at Rochester Institute of Technology, where he is currently a Research Associate Professor. His research interest include design, analysis, and monitoring of electromechanical devices and systems. He co-authored a textbook *Electromechanics and MEMS* and is a member of IEEE.

APPENDIX

The appendix contains information on hyperparameters. Table 7 summarizes the hyperparameters associated with the best-performing models shown in Table 5 and organizes them into three groups: preprocessing, modeling, and post-processing.

The appendix also contains evaluation plots (Figure 10) for each vehicle's idle data using the TFAE-sym architecture. These plots show $\langle MAE \rangle_{20}$ as the CI, and show the baseline threshold in grey. The red vertical line represents the beginning of the fault window.

Table 7. Hyperparameters

Hyperparameter	Value
Pre-processing	
Max signal outlier run	2
Window size	30
Window slide	30 (no overlap)
Operating condition	idle/driving
Modeling: FCAE	
learning rate	5e-4
batch size	32
p_dropout	0.05
noise_scale	0.1
fully-connected layers	[150, 128, 96, 80, 64, 48, 32, 48, 64, 80, 96, 128, 150]
Modeling: CNNAE	
learning rate	1e-4
batch size	32
p_dropout	0.0
noise_scale	0.1
kernel size	3
channels	[5, 25, 125]
fully-connected layers	[875, 128, 96, 128, 875]
Modeling: TCNAE	
learning rate	1e-3
batch size	32
p_dropout	0.0
noise_scale	0.0
blocks	3
k_size	3
latent_size	150
use skip connections	True
Modeling: TFAE-asym	
learning rate	1e-4
batch size	32
p_dropout	0.0
noise_scale	0.1
num_layers	2
nhead	1
model_dim	8
feedforward_dim	64
fully-connected layers	[240, 128, 64, 128, 150]
Modeling: TFAE-sym	
learning rate	5e-4
batch size	32
p_dropout	0.0
noise_scale	0.1
num_layers	2
nhead	1
feedforward_dim	16
model_dim	16
bottleneck_dim	4
Post-processing	
CI	smoothed MAE
# smoothing taps	20

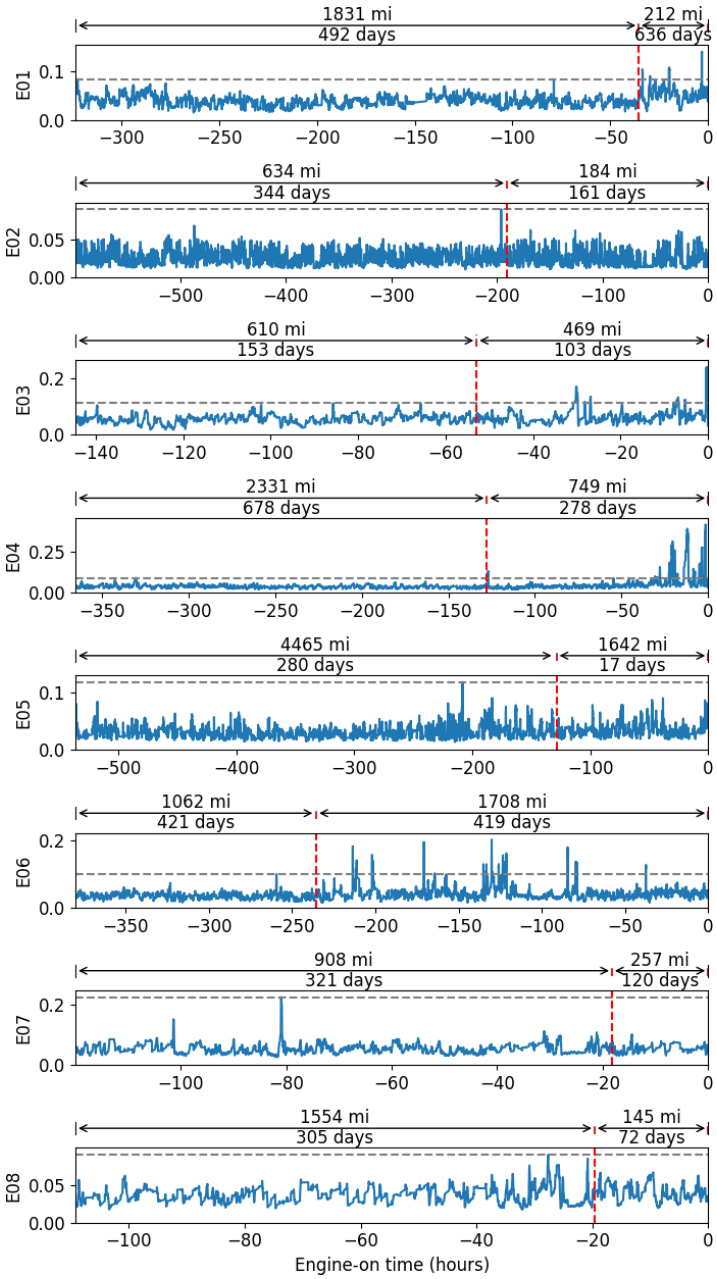

Figure 10. TFAE-sym evaluations on each vehicle's idle data with 0% margin thresholds dotted in grey.