 doi:10.20944/preprints202107.0040.v1

Article

Towards Making Predictive Maintenance System Adaptive and Interpretable

Wenbo Yu *, Xuejiao Li and Yong Sun

Isuzu Technical Center of America, 46401 Commerce Center Dr, Plymouth, MI 48170
* Correspondence: wenbo.yu@isza.com

Abstract: Using data-driven models to solve predictive maintenance problems has been prevalent for original equipment manufacturers (OEMs). However, such models fail to solve two tasks that OEMs are interested in: (1) Making the well-built failure prediction models working on existing scenarios (vehicles, working conditions) adaptive to target scenarios. (2) Finding out the failure causes, furthermore, determining whether a model generates failure predictions based on reasonable causes. This paper investigates a comprehensive architecture towards making the predictive maintenance system adaptive and interpretable by proposing (1) an ensemble model dealing with time-series data consisting of a long short-term memory (LSTM) neural network and Gaussian threshold to achieve failure prediction one week in advance and (2) an online transfer learning algorithm and a meta learning algorithm, which render existing models adaptive to new vehicles with limited data volumes. (3) Furthermore, the Local Interpretable Model-agnostic Explanations (LIME) interpretation tool and super-feature methods are applied to interpret individual and general failure causes. Vehicle data from Isuzu Motors, Ltd., are adopted to validate our method, which include time-series data and histogram data. The proposed ensemble model yields predictions with 100% accuracy for our test data on engine stalling problem and is more rapidly adaptive to new vehicles with smaller error following application of either online transfer learning or the meta learning method. The interpretation methods help elucidate the global and individual failure causes, confirming the model credibility.

Keywords: predictive maintenance; transfer learning; interpretable machine learning

1. Introduction

Predictive maintenance is essential for commercial-vehicle original equipment manufacturers (OEMs). Commercial vehicles are mostly used for heavy-duty, long-distance transportation. If failures occur during transportation, additional service time costs or even casualties may be generated, because the vehicles are typically large in sizes. However, if such failures could be predicted in advance, maintenance and target checks could be scheduled before failure occurrence. Hence, money-saving and increased road safety would be facilitated. However, it is difficult to train highly accurate failure prediction models, as the ratio of failure samples to normal samples is extremely low in most scenarios. For example, for a dataset considered by the authors that contained data for 3 years (i.e., 156 weeks) and 30 vehicles, only 9 failure cases occurred. Furthermore, vehicles may only generate abnormal data for two weeks ahead of failure and the percentage of failure cases could be as low as 0.7%. As there is a limited number of failure samples, traditional binary classification is not suitable for such datasets. An alternative approach is to use only normal data to train models and to discover the patterns within the data. Regression techniques have been widely adopted in this context [1] [2]. Applying this idea, we firstly developed an ensemble method that only trained with normal data for failure prediction.

With the predictive maintenance system well built, commercial vehicle OEMs must also consider how to make existing models adaptive to different vehicles for new

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

customers. Application of the maximum amount of current knowledge to a new vehicle is preferable, as this reduces time and manpower costs and, thus, reduces the financial cost. Two optimizations could be made through this process. The first idea is to make the training process contain as many different scenarios as possible. The factors of the scenarios include vehicle types, working conditions, i.e whether it is for long distance transportation or short distance. Mixing all the data together without considering the different types of the data could misguide the model from uncovering of the real pattern. So it is preferred to add more detailed labels to the data. The problem that come with it is that a large number of classes with little data. The unbalance may be brought again since for some scenarios, it may have far more data than others. To solve these issues, we implemented a meta learning method to train a model using multiple existing vehicle datasets, to render the model broadly suitable for many vehicles. The second idea is to reduce the cost when making the existing scenarios adaptive to new scenarios. The cost includes both time and data amount needed for the adaptation. We proposed an online transfer learning method named HomPlusOTLMS, which is a modified version of HomOTLMS [3]. The developed model can be quickly adapted to a new vehicle with a small number of datasets or procedures.

Another issue for most high prediction accuracy models are "black box" machine learning or deep learning models Therefore it is hard to obtain critical features from these "black box" directly. Engineers or other people who use these model cannot know how does the model work. Therefore, the interpretation model is needed for these "black box" models. Not only for finding out important reasons for the prediction, domain experts also could evaluate whether the high accuracy model predicts failure based on correct reasons.

To summarize the contents in the paper, we developed a novel ensemble model, which contains a Gaussian threshold and long short-term memory (LSTM) network [4] for failure prediction tasks. During the training process, the LSTM network is first trained to reflect the vehicle data patterns. Some values of vehicle features can then be predicted as expected phenomena and compared to the actual values. Failures can be predicted according to a threshold derived from a Gaussian distribution that is adaptive to each vehicle. To validate this approach, data from Isuzu Motors, Ltd. were used to predict the probability of engine stalling in the next seven days. (A seven-day time frame was selected because this is expected to be a sufficient period for the potential risk vehicles to be checked.). Furthermore, the model were used to verify the transfer learning and meta learning methods while adaptive target scenarios. As for the interpretable study, the Local Interpretable Model-agnostic Explanations (LIME) interpretation tool and super-feature methods are applied to interpret individual and general failure causes. Histogram data collected by Isuzu Motors, Ltd were used to verify this method.

To summarize, the contributions of this paper are the following:

- We proposed a complete predictive maintenance system including a novel ensemble network for failure prediction, fast adaptive to new scenarios and global and individual interpretation methods to obtain global and individual failure reasons.
- The approach offered an extension of previous work on transfer learning, meta learning and interpretable machine learning from classification to regression of previous work.
- We tested the system with real world data and give comparable results.
- Demonstrate interpretation results for "black box" models, also integrate prediction model and interpretation modules as pipeline with user friendly interface for application.

The remainder of this study is organized as follows. In Section 2, a brief review of literature on imbalanced data analysis and time-series data anomaly detection is provided. In the Data section, the relevant data pre-processing methods are described in detail. In the Methodology section, the logic of the methods as well as detailed network and statistic methods are illustrated. In the Experiment Results section, we present the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

experimental test setups, followed by the test results and analysis. Finally, we report our conclusions and note future research directions in the Conclusion section.

2. Related Work

In this section, we summarize background work needed for this paper from data process to deep neural networks design for anomaly detection. The state of the arts of Adaptive learning and meta learning, methods to improve neural network performance, as well as interpretable machine learning work are reviewed.

2.1. Deep Learning-Based Anomaly Detection Methods

In the broader field of ML, deep learning techniques have undergone fast development and implementation. Different kinds of deep neural networks have been utilized for anomaly detection tasks, such as CNNs [5], generative adversarial networks (GANs) [6], and variational autoencoders (VAEs) [7]. Among the neural networks, LSTMs, which are a kind of recurrent neural network (RNN), have exhibited good performances in many fields, including research on aero engines [8], controller area network (CAN) buses [9], and light curves [10]. LSTMs achieve powerful handling of time-series data; notably, most anomaly detection tasks involve data in this format.

In a similar work to the present study, Malhotra et al. [11] used stacked LSTMs to model the normal behavior of time-series data. In that case, the model was used to predict future sensor values and to compare them with real values. This logic is not applicable to vehicles, however, as the sensor data are transient. That is, the vehicle may be accelerating in one second but stop in the next second. Another flaw is that the results fail to evaluate the effect of a model with a threshold to find out anomaly. For our model, the predicted values used for comparison with the real values are all from the same time as the inputs; therefore, calculations are performed for each individual moment. Additionally, we obtain failure prediction results instead of regression scores.

2.2. Transfer Learning

Transfer learning has been widely applied to CNNs [12] [13]. However, for time-series data, there have been few investigations of the general framework underlying such transfer learning methods. Laptev et al. [14] proposed an LSTM auto-encoder that was trained on a large-scale time-series dataset. The well-trained auto-encoder was used to train and learn time-series features. Furthermore, Fawaz et al. [15] investigated transfer of deep CNNs for time-series tasks [15].

To implement transfer learning, effective combination of existing models must be investigated, considering their domains. Previously, Wu et al. [3] proposed an online transfer learning algorithm for both homogeneous and heterogeneous data, which is a combination of several models. However, the proposed model is not adaptive to new data; thus, a new model cannot be derived from an existing model. Similar work is reported in [16], which presents a model that implements a prediction, assessment, and weight updating process. However, that approach does not classify sources into homogeneous and heterogeneous sources. Instead, a multi-view approach is adopted. The data considered in this study are from different, homogeneous buses; therefore, our algorithm is constructed based on the approach detailed in [3].

2.3. Meta Learning

Meta learning methods, also known as “learning to learn" methods, are usually trained over a variety of learning tasks. This allows the models to function for multiple tasks or to rapidly adapt to a new task with few data elements or procedures. These methods can be divided into (1) metric-, (2) model-, and (3) optimization-based approaches. (1) Metric-based approaches learn the distance metric between the target and a set of known labels and are mostly used for few-shot training examples; therefore, there are limited examples for each class. (2) Model-based methods achieve fast parameter

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

updating when supplied with a small number of samples for a new task by a specifically designed model architecture; examples include MANN [17] and MetaNet [18]. (3) Optimization-based methods achieve fast learning for multiple tasks by adjusting optimization algorithms. For example, Ravi et al. [19] used an LSTM network to update parameters to rapidly adapt to a new task. Model-agnostic meta-learning (MAML) has also been developed, which implements a two-step updating process for efficient weight updating with few samples [20]. Note that Nichol et al. proposed Reptile, which is a first-order MAML with a simplified update procedure [21]. We used Reptile in our work.

2.4. Interpretable Machine Learning

ML has become a critical tool for vehicle predictive maintenance. Vehicle failures such as engine stalls may generate costs and even casualties. Therefore, understanding of ML models related to such failures is critical. ML models are usually defined as "black boxes," which are difficult to understand. Therefore, interpretation techniques have attracted research attention.

There are two major types of interpretability method, intrinsic and post hoc [22]. An intrinsic interpretability method is one where the ML models are considered to be explainable because the model structure and parameters are easy to understand. Post hoc interpretability refers to use of additional methods after model training; that is, additional models are required to analyze the trained model. Intrinsic interpretation methods are only applied to limited ML models such as those featuring linear regression, logistic regression, generalized linear models, and generalized additive models. For complex on-board diagnostic prediction, post hoc methods can be applied to most ML models. Thus, a post hoc method is most suitable for the topic considered in the current study.

Post hoc methods involve interpretation of both global and individual instances. The results yielded by global interpretation methods can explain general failure reasons, whereas those for individual instances can give reasons why a specific instance yields normal behavior or failure. When global interpretation methods are implemented, partial dependence plots [23], individual conditional expectations [24], accumulated local effect plots [25], permutation feature importance analyses, and statistical interpretations are given. As these methods evaluate features individually, they are more suitable for low-dimensional, flattened structure data. When the data dimension increases and the structure becomes complex, the data structure is ignored by these methods. No interpretation method for complex-structured high-dimensional data has been reported to date, to the best of the authors' knowledge. Therefore, a statistical visualization method remains the first option for these data. Local surrogate models are used in interpretation methods for individual instances. Local Interpretable Model-agnostic Explanations (LIME)[26] and Shapley Additive Explanations (SHAP) [27] are two major local surrogate models. LIME is based on a surrogate model trained to approximate the prediction, which is used instead of the original, complex model. The concept behind the LIME method is that the locality of a complex model can be explained by a simple model such as linear model. Therefore, a simple and explainable model is used for data points generated around the test sample. SHAP is based on game-theoretic optimal Shapley values. The Shapely value is the average of all marginal contributions to all possible coalitions.

3. Methodology

In this section, we unfold all the methodologies utilized in this paper from the general logic of the ensemble models to details for each part.

3.1. Logic Flow

In this subsection, we introduce our general logic for an anomaly detection algorithm for a specific vehicle, and that for an adaptive learning method applicable

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

when models of several vehicles are required to be transferred to a new vehicle. The detailed network structures, mathematical derivations, and algorithms are presented in the following subsections.

3.1.1. Anomaly Detection

The logic flow is shown in Fig. 1. The anomaly detection process can be divided into three stages: training, validation, and testing. In the training stage, the vehicle status patterns are uncovered by training a regression model, where the features and labels are the control and dependent sensor variables, respectively. As discussed in the Data section, the control sensor variables can represent the normal vehicle pattern used to predict the dependent sensor values. During the training stage, only normal data are used to train the regression model from the control sensor variables, to obtain the dependent sensor variables.

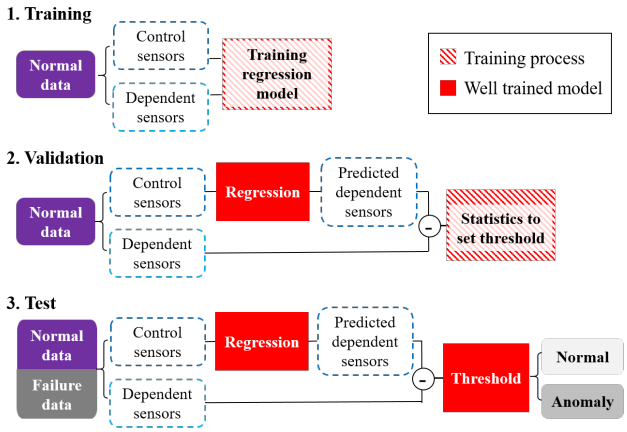

Figure 1. Logic flow for failure prediction

Once the model for analyzing the normal patterns in the dataset is well designed, whenever failure occurs in the system, the prediction given by the control sensor variables no longer tracks the values of some dependent sensor variables. Thus, larger differences appear between the predicted values and the actual values given by those sensors. In this way, a threshold can be set to identify abnormal cases. In other words, the differences between the predicted and actual dependent sensor variables are used to train the threshold. This step is shown as Step 2 in Fig. 1. As in Step 1, only normal data are used to set the threshold. When both the regression model and threshold are trained, the network is used for testing considering both normal and abnormal data, and to further calculate the evaluation criteria such as the recall or precision.

In accordance with the above analysis, the dataset employed in this study was divided as follows:

- Test data (abnormal): data two weeks ahead of the fail date and selected from among the abnormal data;
- Test data (normal): 10% of the normal data in biweekly form;
- Validation data: 20% of the normal data in biweekly form;
- Training data: 70% of the normal data in daily form.

3.2. Neural Network Structure

RNNs have two main benefits for handling of time-series data:

- The input data are in time order and all previous data points are tracked;
- The model patterns are learned over time.

In this work, we adopted the widely used RNN network sub-model, LSTM, for regression. LSTM has an internal memory state and uses three gates (the input, output,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

and forget gates) to propose and pass the input as well as the hidden states. We used an LSTM network with the structure shown in Fig. 2.

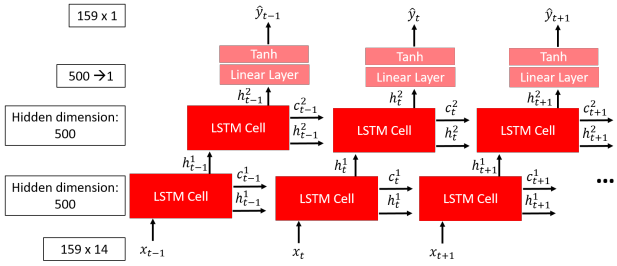

Figure 2. LSTM structure.

The network used in this study was stacked with two LSTMs. The input was $X \in \mathbb{R}^{159\times14}$, as discussed in the Data section. The hidden layers of each LSTM cell were set to h^1 and h^2, the dimensions of which were both equal to 500. One linear layer was set at the end of the second LSTM cell to change the tensor dimension to $\mathbf{y} \in \mathbb{R}^{159\times16}$. The *tanh* function was used to normalize the values in $[-1, 1]$.

3.3. Adaptive Gaussian Threshold

In this subsection, we unfold the overall logic behind formation of the adaptive generated threshold. For the LSTM network, $\forall i, j \in l$, $y^{(i)}$ is correlated with $\mathbf{x}^{(j)}$. A multivariate Gaussian distribution was used to express the dependent sensors, $\mathbf{y}$. Our purpose was to learn the normal pattern of the vehicle; thus, the training set collection had independent and identically distributed properties. The Gaussian distribution was

$$p(\mathbf{y}|X, \mathbf{w}, \mathbf{\Sigma}) = \mathcal{N}(\mathbf{y}|f(X, \mathbf{w}), \mathbf{\Sigma}) \tag{1}$$

where f represents the LSTM network with parameters of $\mathbf{w}$ and $\mathbf{\Sigma}$ is an $l \times l$ covariance matrix. For the dataset with N samples $\{\mathbf{X}, \mathbf{Y}\}$, the input value is $\mathbf{X} = \{X_1, ..., X_N\}$ while $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$. The likelihood function is given by

$$p(\mathbf{Y}|\mathbf{X}, \mathbf{w}, \mathbf{\Sigma}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{y}|f(X, \mathbf{w}), \mathbf{\Sigma}) \tag{2}$$

The maximum likelihood of $\mathbf{w}$, $\mathbf{w_{ML}}$, was found from regression. Obtaining the maximum likelihood was equivalent to minimizing the mean square error (MSE) loss, after which $\mathbf{\Sigma}$ could be determined from the relation

$$\mathbf{\Sigma_{ML}} = \frac{1}{N} \sum_{n=1}^{N} (f(\mathbf{X_n}, \mathbf{w_{ML}}) - \mathbf{y_n})(f(\mathbf{X_n}, \mathbf{w_{ML}}) - \mathbf{y_n})^T \tag{3}$$

Here, $\mathbf{w}$ and $\mathbf{\Sigma}$ reflect the model. That is, when failure occurs, the normal pattern is changed; this is reflected by a parameter change. An easy approach is to use the determinant of $\mathbf{\Sigma}$, $|\mathbf{\Sigma}|$, to reflect the value. In this work, we used β and t to represent $|\mathbf{\Sigma}|$ and the adaptive generated threshold, respectively. Any difference larger than t was judged to be a failure case, with

$$f(\beta) = \begin{cases} sgn(\beta)(|\beta| - t), & |\beta| \geq t \\ 0, & |\beta| < t \end{cases} \tag{4}$$

The mean β value was calculated for each day and from a two-week-long data array. We hoped to achieve predictions for one week in advance; therefore, we chose data from the first week, with $\mathbf{b} = \{\bar{\beta}_1, ...\bar{\beta}_7\}$. As the data were expressed as a Gaussian distribution, 3σ was set as the threshold. The validation set was used to learn the statistical variables and to set the adaptive threshold. If the absolute difference for the test data was larger than the threshold, the data were classified as abnormal.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

3.4. Online Transfer Learning

The LSTM models trained for each vehicle were taken from the homogeneous domain reported in [3]. Therefore, we modified the HomOTLMS algorithm proposed in that work, which was designed for classification problems. As the model reported in this study was designed for a regression algorithm, we modified HomOTLMS to obtain our own Regression HomOTLMS (HomPlusOTLMS) method, as detailed in Algorithm 1.

Compared with the HomOTLMS model, HomPlusOTLMS (1) explores homogeneous transfer learning adjustment to become suitable for regression problems besides classification problems and (2) enables updating of model parameters from the source domain. To enhance the target task performance, we leverage knowledge extracted from offline source data. In the validation task performed in this study, we treat each model built for a certain vehicle as a source domain and explore online transfer learning to make those models adaptive to a new vehicle.

Algorithm 1 HomPlusOTLMS

Input: LSTMs from source domains $f_1^S = \{f_1^{S_1}, f_1^{S_2}, ..., f_1^{S_n}\}$, weight discount parameter $\beta \in (0,1)$, online data stream $\mathcal{X}$ and labels $\mathcal{Y}$, and data-stream length m.
Initialize: f_1^T with random weights, $u_1^1, ..., u_1^n = 1/(n+1), v_1 = 1/(n+1)$, loss function L, and optimization function *optimizer*.

for $d = 1, 2, ..., m$, **do**
  receive instance: $x_d \in \mathcal{X},\ y_d \in \mathcal{Y}$
  $loss_d^v = L(f_d^T(x_d), y_d)$
  $f_{d+1}^T = optimizer(loss_d^v)$
  for $i = 1, 2, ..., n$, **do**
    $loss_d^{u_i} = L(f_d^{S_i}(x_d), y_d)$
    $f_{d+1}^{S_i} = optimizer(loss_d^{u_i})$
  $r_d^{u_1}, ..., r_d^{u_n}, r_d^v = rank(loss_d^{u_1}, ..., loss_d^{u_n}, loss_d^v)$
  for $i = 1, 2, ..., n$, **do**
    $p_d^i = u_d^i / \left(\sum_{i=1}^n u_d^i + v_d\right)$
    $u_{d+1}^i = u_d^i \beta^{r_d^{u_i}}$
  $p_d^v = v_d / \left(\sum_{i=1}^n u_d^i + v_d\right)$
  $v_{d+1} = v_d \beta^{r_d^v}$
  $f_d(x_d) = \left(\sum_{i=1}^n p_d^i f_d^{S_i}(x_d) + p_d^v f_d^T(x_d)\right)$

Output $f_m(.)$

In accordance with the raw algorithm, we used the domain to represent the vehicle. In the pseudo-code, the input includes a list of models built for different domains in an offline batch learning paradigm, which are labeled f^{S_1} to f^{S_n}. Note that β is the weight discount parameter for resetting the weights for each step. The input also contains the online data stream $\mathcal{X}$, the corresponding labels $\mathcal{Y}$, and the ending iteration m.

In our proposed approach, the target model f^T is first initialized with random weights. Then, a weight vector $\mathbf{u} = (u^1, u^2, ..., u^n)^T$ and a variable v are constructed to represent the contributions of the source and target model. They are initialized with equal weights. Then they are updated by multiplying damped exponential calculated by the rank of accuracy predicted by different models. Next, the loss function and optimizer are selected.

During the loop, the online data stream is input in each iteration. All the source and target models are used to generate predictions and to compare the loss. The loss is used to update the networks and rank the performance. Then, the weights are updated using β. The updated weights are then normalized to reflect the possibility. The final model f is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

$$f(x) = \left(\sum_{i=1}^{n} p^i f^{S_i}(x) + p^v f^T(x) \right). \tag{5}$$

3.5. Meta Learning

Our meta learning algorithm is presented in Algorithm 2, and was developed with reference to the Reptile method [21]. Considering a distribution over tasks, $p(\mathcal{T})$, Reptile calculates the gradient parameters by several steps for each task and then update once. For the sample application reported in this work, the different tasks correspond to different vehicles. We hoped to train a single model that could be used to finish regression tasks for different vehicles. Meta learning is expected to help us on training step to gain better performance for the network without messing up data from different sources. The model could also be taken as the initialization for different vehicles and that could adapt to new vehicles with input from only a few real samples.

Algorithm 2 Modified Reptile algorithm for our task

Input: Distribution over tasks $p(\mathcal{T})$, learning rate for each step α, learning rate for update γ, dataset for different tasks $\{\mathcal{X}, \mathcal{Y}\}$
Initialize: f_θ with random weights, number of steps N, loss function L, and optimization function *optimizer*

for Iteration $= 1, 2, \ldots$ **do**
  Set task order K to update the model
  for k in K **do**
    Sample batch $\{\mathbf{X}, \mathbf{Y}\}$ of tasks $\mathcal{T}_k \sim p(\mathcal{T})$
    for steps $= 1, 2, \ldots, N$ **do**
      $\theta_k \leftarrow optimizer(L(f_{\theta_k}(X), Y), \alpha)$
    $\theta \leftarrow \theta + \gamma(\theta_k - \theta)$

Output θ

3.6. Interpretable Machine Learning

Recall that global interpretation means the failure reasons are general reasons for one specific issue, such as engine stalling. In contrast, individual interpretation means the failure reasons for one specified instance, such as one specified vehicle. In this study, we proposed both methods to explain the potential global and individual vehicle failure reasons.

As histogram data includes more sensors' data than time-series data, the current study used histogram data to explain failure reasons after the prediction with time-series data. The logic flow is shown in Fig. 3.

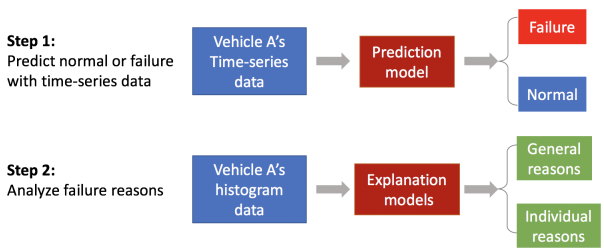

Figure 3. Logic flow for failure prediction

For general reasons, probability density function (PDF) method was used to compare all normal histogram data and failure data. First of all, potential features were selected by experts of the histogram data. Then for each feature, one PDF method was applied on normal data and failure data in one figure to compare the difference between

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

normal and failure. Then if there was an obvious difference between the distributions of normal data and failure data, we would supposed that the feature was one of the critical causes to the failure.

For the individual interpretation method, LIME was selected to explain failure reason for one specific vehicle. LIME method firstly perturbed the instances that we were interested. Then it generated a simple linear model instead of complex 'black-box' model. As the model just predicted the one point, simple linear model could predict with high accuracy. The output of LIME was part of input features with explanation, reflecting the contribution of each feature to the prediction of a data sample such as which feature changes will have most impact on the prediction.

4. Data

In this section, we introduce two datasets together with some data processing methods. One of the dataset is time-series data for training prediction model. The other is histogram data used for failure reason analysis.

4.1. Data Description

The time-series dataset considered in this study contained 3 years of sensor data records collected from 30 buses. Records of 40 s in length were collected while each vehicle was operating, with 4 FPS every 12 min. Data were collected from 75 sensors. Based on domain experts' suggestions, two groups of sensors were selected from the 75 sensors. The first group of sensors were considered as control sensors and the recorded data of which were used to predict the values of the dependent sensors of the vehicle. The dependent sensors were considered to be the target sensors and belonged to the second sensor group. The values of the target sensors could be regarded as the regression results of the sensors from the first group. In [11], the data from the first and second groups were named the control and dependent sensor variables, respectively, and that approach was also adopted here. The 14 control and 16 dependent sensor variables are listed in Tables 1 and 2, respectively. This group either contains sensors recording the basic condition of the vehicle (speed, curret gear) or target (target fuel injection), neither of which could cause failure.

Table 1: Control sensor variables.

Variable Name	Unit
Accelerator-pedal opening degree	%
Actual engine torque	%
Engine speed	rpm
Target fuel injection amount	mm^3 $stroke^{-1}$
Current gear	-
Vehicle speed (25 pulses)	km h^{-1}
Clutch SW	MT only
Brake SW	-
Cruise control status	-
EGR target position	%
Intake throttle target position	%
Target rail pressure	MPa
Target boost	%
Exhaust pipe INJ ON / OFF state	%

Histogram data were used to explain the global and individual failure reasons, as it is easier to collect and process histogram data than time-series data from vehicles. The histogram data frame is given in Table 3.

The histogram data contained seven super features, as detailed in Table 3. A super feature is a group of single features having similar physical meanings under continuous operating conditions. In this work, each super feature included 441 to 1368 single features. As mileage is a key factor, it was listed as an individual super feature. All other features defined by two variables, such as single feature in INTPRES is determined by

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

Table 2: Dependent sensor variables.

Variable Name	Unit
Coolant temperature	°C
Fuel temperature	°C
Post injection Q	mm^3
Common rail pressure	MPa
DPF differential pressure	kPa
Intake air temperature	°C
Boost pressure	*kPa*
DOC inlet temperature	°C
MAF	g cyl^{-1}
Cylinder-1 balancing fuel compensation	mm^3 $stroke^{-1}$
Cylinder-2 balancing fuel compensation	mm^3 $stroke^{-1}$
Cylinder-3 balancing fuel compensation	mm^3 $stroke^{-1}$
Cylinder-4 balancing fuel compensation	mm^3 $stroke^{-1}$
Cylinder-5 balancing fuel compensation	mm^3 $stroke^{-1}$
Cylinder-6 balancing fuel compensation	mm^3 $stroke^{-1}$
Sensor value O2	%

Table 3: MMU data frame.

No.	Super Feature	Single Features	Unit
1	Mileage	1	mileage
2	INTPRES	441	intake pressure (kPa), rpm
3	COMRAIL	1071	inject pressure (MPa), rpm
4	INT_TEMP	777	intake temperature (°C), rpm
5	ENG_WTEMP	1368	water temperature (°C), rpm
6	FUEL_EXH	861	exhaust pressure (MPa), RPM
7	WATER_FUEL	653	water temperature (°C), fuel (mm^3/s)

intake pressure and rpm at the same time. Therefore, 5172 single features, seven super features, are used to explain vehicle failure reasons.

4.2. Data Pre-processing

The raw time-series sensor data need pre-processing to set and uniform the length and unit of the data. We achieved this by applying rolling window methods and normalization.

4.2.1. Rolling Window Methods

For our data to reflect the time-series properties, we adopted rolling window methods to process the raw data; these methods reshaped the data into small time intervals of fixed size. As the data elements were 40-s continuous records with 4 FPS, these 40-s data segments could be regarded as a unit of time-series data and were used to set the rolling window size. Note that a well-selected window size can overcome the delay phenomena present in automotive systems, which cause some variables to be affected by others from a few seconds previously. Rolling windows contain a sensor-data time dimension and are intended to rectify this problem. The control sensor variable data in each window were expressed as a matrix $X = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(l)})^T \in \mathbb{R}^{l \times m}$, where $\mathbf{x}^{(t)} = (x_1^{(t)}, x_2^{(t)}, ..., x_m^{(t)})^T \in \mathbb{R}^m$. Here, l refers to the time-series length, which was 159 in this study (40 s with 4 FPS but missing 1 data point), and m refers to the number of control sensors, which was 14. One example of this is shown in Fig. 4, where the different colours correspond to different sensors. Delays are clearly apparent for some sensors. Similarly, dependent sensor variables were represented by $\mathbf{y} = (y^{(1)}, y^{(2)}, ..., y^{(l)})^T \in \mathbb{R}^l$. Note that the sensor number was only 1 as we constructed different models for each dependent sensor variable.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

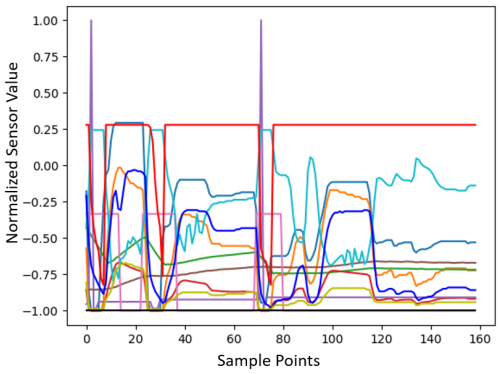

Figure 4. Data samples in time window (the colours represent different control sensors).

4.2.2. Normalization

To unify the scales of the data from different sensors, we found the upper and lower bounds of the values for each sensor and normalized all values to $[-1, 1]$, based on the following equation:

5. Experiment Results

In this section, we show the experiment results for each part of the ensemble model to support the theoretical section.

5.1. Data Description

From the dataset, we selected 16 buses, 9 of which experienced failure cases and 7 of which operated consistently well. The data on these buses are listed in Table 4. The training, validation, and test data roughly corresponded to 70%, 20%, and 10% of the dataset. Note that the samples in training set are summarized in units of days (d), while the validation and test sets were organized biweekly (biw), with the data from the first week being adopted for prediction, and that for the second week being neglected.

Table 4: Data summary for training, validation, and test datasets.

Vehicle idx	Training (d)	Validation (biw)	Test Normal (biw)	Test Abnormal (biw)
1	580	11	6	1
2	509	10	5	1
3	425	9	3	1
4	225	6	3	1
5	219	4	2	1
6	164	4	2	1
7	160	4	2	1
8	64	2	1	1
9	58	2	1	1
10	449	14	7	-
11	366	6	3	-
12	241	5	3	-
13	232	6	4	-
14	218	14	7	-
15	216	6	3	-
16	195	6	3	-
Total	**4325**	**109**	**55**	**9**

5.2. Failure Prediction

We used the Nvidia Titan RTX graphics card for training, with the batch size set to 1. Each update during training required a period of 0.025 s. The test results are listed in Table 5, where “Case," “Ab," and “N" indicate the test set samples, abnormal

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

items, and normal items, respectively. After classification, for each case, the sensor-level anomaly detection results were obtained. The numbers of abnormal sensor readings were summed and recorded under "Num". The numbers of failure cases were higher than the normal cases except for N1 and Ab9, which were equal. Cases with numbers over 3 were considered to be failures; the failure prediction results are listed in Table 6

Table 5: Test results with all sensors considered (“Ab" and “N" indicate abnormal and normal, respectively).

Case	Num	Case	Num	Case	Num	Case	Num
Ab 1	13	N 8	2	N 24	1	N 40	0
Ab 2	9	N 9	2	N 25	1	N 41	0
Ab 3	9	N 10	2	N 26	1	N 42	0
Ab 4	8	N 11	2	N 27	1	N 43	0
Ab 5	8	N 12	2	N 28	1	N 44	0
Ab 6	7	N 13	2	N 29	1	N 45	0
Ab 7	6	N 14	1	N 30	1	N 46	0
Ab 8	6	N 15	1	N 31	0	N 47	0
Ab 9	4	N 16	1	N 32	0	N 48	0
N 1	4	N 17	1	N 33	0	N 49	0
N 2	3	N 18	1	N 34	0	N 50	0
N 3	3	N 19	1	N 35	0	N 51	0
N 4	3	N 20	1	N 36	0	N 52	0
N 5	3	N 21	1	N 37	0	N 53	0
N 6	3	N 22	1	N 38	0	N 54	0
N 7	2	N 23	1	N 39	0	N 55	0

Table 6: Failure prediction results considering all sensors.

	Positive	Negative	Total Number
Normal	1	54	55
Failure	9	0	9

From Table 6, we obtained 98.4% $accuracy = (TP + TN)/(TP + TN + FN + FP)$, 90% $precision$= TP /(TP + FP), and100%recall $= TP/(TP + FN)$, where TP, TN, FP, and FN are true positive, true negative, false positive and false negative, respectively.

Table 7 lists the sensors related to the time-series data. Each sensor’s rank indicates its relationship with engine stalling, as defined by domain experts.

Table 7: Time-series sensors.

Idx	Sensor Name	
1	50	Cylinder-2 balancing fuel compensation
2	53	Cylinder-5 balancing fuel compensation
3	49	Cylinder-1 balancing fuel compensation
4	14	DPF differential pressure
5	54	Cylinder-6 balancing fuel compensation
6	19	DOC inlet temperature
7	12	Post injection Q
8	29	MAF
9	16	Intake air temperature
10	11	Fuel temperature
11	10	Coolant temperature
12	51	Cylinder-3 balancing fuel compensation
13	87	Sensor O2 value
14	52	Cylinder-4 balancing fuel compensation
15	17	Boost pressure
16	13	Common rail pressure

Based on the critical sensor ranks, only the five highest-ranking sensors were considered in the summary of the sensors for each case. The results are listed in Table 8.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

In that table, the numbers of abnormal cases are all higher than the normal cases. Cases with numbers over 1 were taken as failures; the failure prediction results are listed in Table 9. Hence, we achieved 100% *accuracy*, *precision*, and *recall*.

Table 8: Test results considering five highest-ranking critical sensors ("Ab" and "N" indicate abnormal and normal, respectively).

Case	Num	Case	Num	Case	Num	Case	Num
Ab 1	4	N 8	1	N 24	0	N 40	0
Ab 2	4	N 9	1	N 25	0	N 41	0
Ab 3	3	N 10	1	N 26	0	N 42	0
Ab 4	3	N 11	1	N 27	0	N 43	0
Ab 5	3	N 12	1	N 28	0	N 44	0
Ab 6	3	N 13	1	N 29	0	N 45	0
Ab 7	3	N 14	1	N 30	0	N 46	0
Ab 8	3	N 15	1	N 31	0	N 47	0
Ab 9	2	N 16	1	N 32	0	N 48	0
N 1	1	N 17	1	N 33	0	N 49	0
N 2	1	N 18	1	N 34	0	N 50	0
N 3	1	N 19	1	N 35	0	N 51	0
N 4	1	N 20	1	N 36	0	N 52	0
N 5	1	N 21	0	N 37	0	N 53	0
N 6	1	N 22	0	N 38	0	N 54	0
N 7	1	N 23	0	N 39	0	N 55	0

Table 9: Failure prediction results considering five highest-ranking sensors.

	Positive	Negative	Total Number
Normal	0	55	55
Failure	9	0	9

5.3. Online Transfer Learning

The transfer learning method is used to fast adaptive to a new vehicle and decrease the loss while learning process

We validated our algorithm by transferring two well-trained models to a new vehicle. This was equivalent to training the LSTM neural networks. We selected the MSE loss and Adam optimization algorithm. The β value was set to 0.9. We used the Nvidia Titan RTX graphics card and tested the run time of our algorithm. The relationship between the number of existing networks used for transfer learning and the run time when one sample was input is detailed in Table 11. The time increased with the number of existing models. Note that the reported time is that required for both the training process and the distribution setting, and was sufficiently fast for online transfer learning.

Table 10: Number of existing models used for online transfer learning.

	One Model	Two Models	Three Models
Time	0.283 s	0.465 s	0.651 s

Our online transfer learning results are visualized in Fig. 5. The losses of a model trained from random weights ($f_.^T$ in Algorithm **??**), a model obtained from transfer learning ($f_.$ in Algorithm **??**), and a well-trained model are plotted. These models were tested on a test set of 1000 elements obtained from new-vehicle data. The results for the Cylinder-1 balancing fuel compensation sensor only are shown.

Note that the curve of the model incorporating our online transfer learning algorithm indicates low loss at the beginning and has a steeper decline than the original models; this indicates that transfer learning can accelerate existing models. To clarify that the existing model assisted the training of the new model, the weights $\mathbf{u}$ and v are visualized in Fig. 6. The u_1 trend of Model 1 increases continuously with time, whereas

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

the trends for the other two models decrease; thus, the final model was trained based on Model 1. This finding further validates our online transfer learning algorithm.

Table 11: Number of existing models used for online transfer learning.

	One Model	Two Models	Three Models
Time (s)	0.283	0.465	0.651

Our online transfer learning results are visualized in Fig. 5. The losses of a model trained from random weights ($f_.^T$ in Algorithm **??**), a model obtained from transfer learning ($f_.$ in Algorithm **??**), and a well-trained model are plotted. These models were tested on a test set of 1000 elements obtained from new-vehicle data. The results for the Cylinder-1 balancing fuel compensation sensor only are shown.

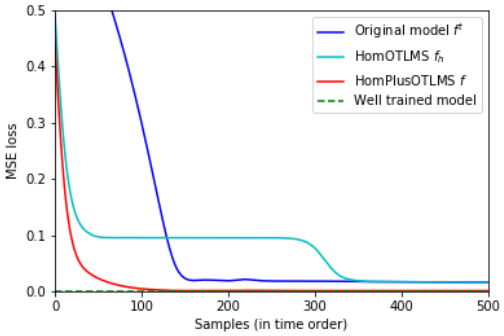

Figure 5. Losses of various models on test set as function of time order.

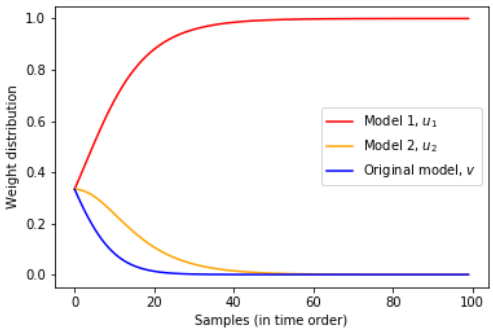

Figure 6. Weight distributions as functions of samples in time order.

5.4. Meta Learning

We used the Nvidia Titan RTX graphics card for training and set both the learning rate for each step α and update rate γ to 10^{-5}. The batch size was five and the Adam optimizer was used to train the models. In Fig. 7, we report the results for the model trained with ten vehicles, as well as those obtained for models trained with one of the ten vehicles only. The times required for each update were 0.023 and 0.101 s without and with meta learning training, respectively, for two tasks. In Fig. 7, the y-axis corresponds to the accuracy calculated based on the MSE loss. Data from seven different vehicles were used as test data. When the number of vehicles exceeded one, the accuracy was calculated using the mean value for the multiple vehicles. The model trained with data from multiple vehicles has the lowest mean value of MSE loss for the seven vehicles than others trained with single vehicle only. This verified our expectation that meta learning could be used to help the network gain better performance with training data from different sources without messing them up.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

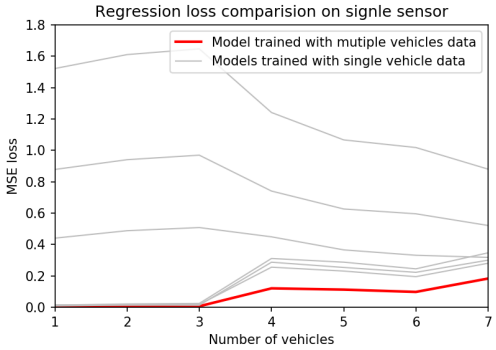

Figure 7. Mean accuracy as function of vehicle number.

To verify that meta learning could used to learn a model as the initialization for different vehicles. The model trained via meta learning was tested on data for three vehicles and compared to models trained using random weight. The model accuracy curves for the samples input for training are shown in Fig. 8. The model trained using meta learning methods had higher accuracy at the beginning. The accuracy was calculated using a test set that was separate from the training set.

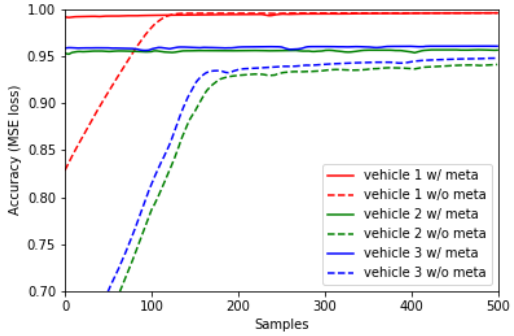

Figure 8. Accuracy variation as function of sample number.

5.5. Interpretation Analysis

In this section, global and individual interpretation results were reported. Three super features were considered in order to demonstrate the interpretations. For the global interpretation, one super feature was divided into four sections based on the ranges of the two variables. Figures. 9 show the four different combinations of intake pressure (kPa) and rpm. The red (blue) bars and curves indicated failure (normal) vehicles. Figure. 9(a) shows the results for the high intake pressure and rpm ranges. The x-axis represents for the time, indicating the number of hours worked by the vehicles in this range, while the y-axis represents for the probability density index, which could convey the probability density trend. From Figs. 9, there is an obvious difference between the results for the normal and failure vehicles in Fig. 9a. The failure vehicles exhibits longer working times under high intake pressure and rpm conditions. In the other three figures, however, the probability density distributions are quite similar. Therefore, from the visualized results, high intake pressure and high rpm are summarized as characteristics of failure vehicles.

Figs. 10 show results for combinations of intake temperature and rpm. Hence, the failure vehicles have more working conditions in the high-intake-temperature range, with both high and low rpm. However, more obvious differences appear between the normal and failure vehicles for the high-rpm case.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

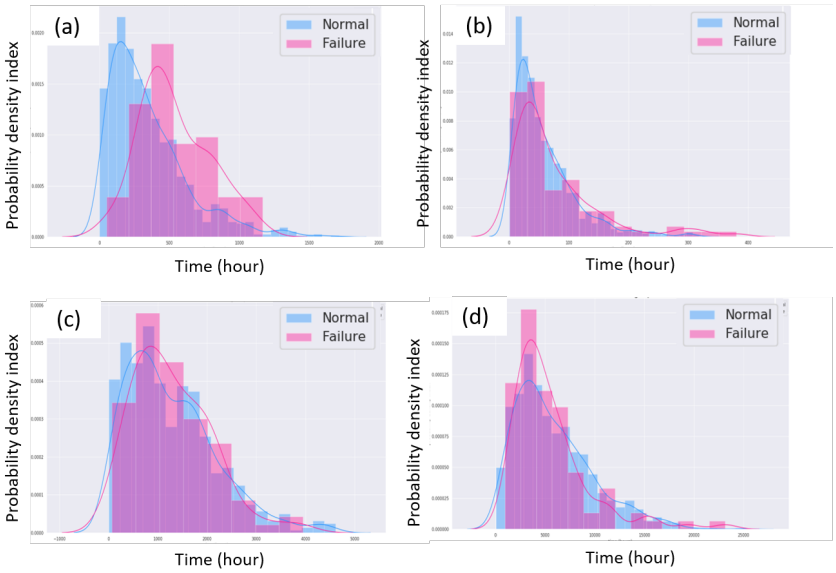

Figure 9. Intake pressure and rpm: **(a)** high intake pressure high rpm. **(b)** high intake pressure low rpm. **(c)** low intake pressure high rpm. **(d)** low intake pressure low rpm.

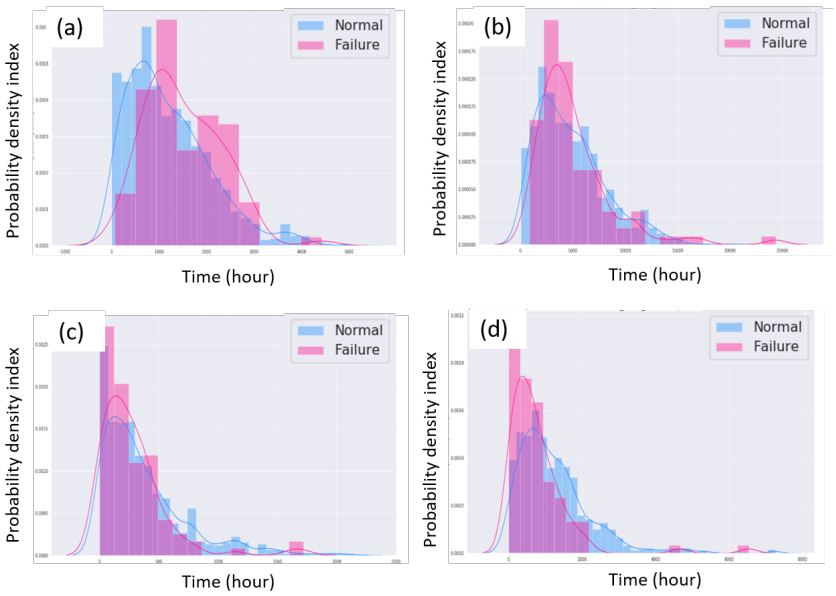

Figure 10. Intake pressure and rpm: **(a)** high intake temperature high rpm. **(b)** high intake temperature low rpm. **(c)** low intake temperature high rpm. **(d)** low intake temperature low rpm.

Figs. 11 show results for the combination of engine water temperature and rpm. Distinct differences are apparent for three of four figures. Figures. 11a and Figures. 11b both show results for high engine water temperature. From these two figures, it is apparent that a large portion of failure vehicles have long working periods under high-water-temperature conditions. Figure. 11d shows that failure vehicles also have a high probability of working under low-engine-water temperature and low-rpm conditions.

The global interpretation analysis provides a general explanation for all failure vehicles. If an instance is classified as failure based on these reasons, we can trust the classification model. However, if the classification is based on other, irrelevant reasons, even if the classification result is correct, we must doubt the classification method.

Figs. 12 to Figs. 14 show interpretation results for one failure vehicle obtained using models based on both old and new datasets. Old and new datasets have the same features and structures. However, the mileage feature in old dataset is a bias features,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

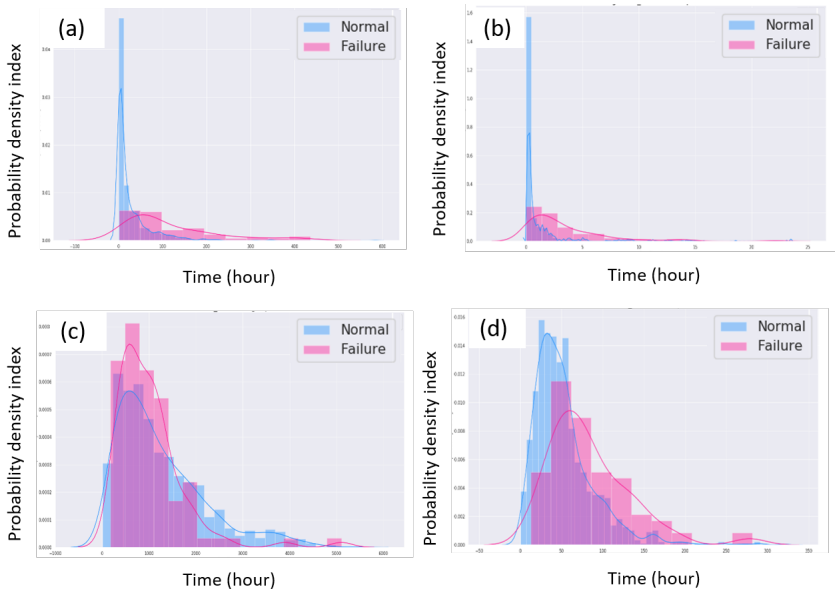

Figure 11. Intake pressure and rpm: **(a)** high engine water temperature high rpm. **(b)** high engine water temperature low rpm. **(c)** low engine water temperature high rpm. **(d)** low engine water temperature low rpm.

as all failure vehicles' mileages are higher than normal vehicls' mileage. For the new dataset, failure vehicles' and normal vehicles' mileages have the same range. LIME was applied to both datasets. In the figures, the orange and green bars represent reasons why the instance should be classified as failure or normal, respectively. The bar length indicates the importance of these factors. As each super feature includes hundreds of individual features, only the 20 most important features are shown in the figures, for improved readability.

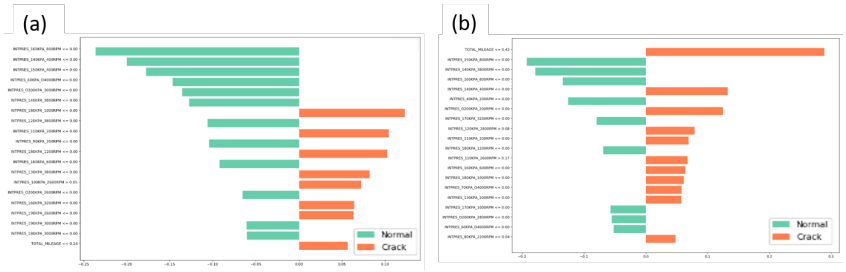

Figure 12. Comparison between new dataset and old dataset with intake pressure: **(a)** Intake pressure new dataset. **(b)** Intake pressure old dataset.

Figs. 12 show results for the intake pressure dataset and mileage. In Fig. 12b, the first reason for the instance is mileage, whereas in Fig. 12a, the highest-ranking feature is feature from intake pressure. From analysis of the global interpretation results, mileage is not one of the failure reasons. Mileage was selected as a failure reason for the old dataset because that dataset has obvious bias as mentioned. These results show that LIME can determine a failure reason based on the given model, and can help users understand how the model classify failure and normal vehicles.

Figs. 13 show individual interpretation results for the intake temperature dataset, whereas Figs. 14 are interpretation results for the engine water temperature dataset.

The old and new datasets were both tested with LIME. For both old datasets, the top failure reasons are the same, i.e., mileage. The results given by all old dataset models indicate that mileage is the failure reason, which is unreasonable. Therefore, we cannot trust the model based on the old dataset. For the new-dataset models, the normal and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0040.v1

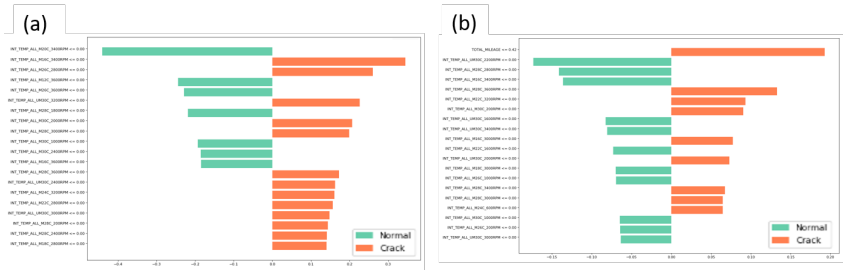

Figure 13. Comparison between new dataset and old dataset with intake temperature: **(a)** Intake temperature new dataset. **(b)** Intake temperature old dataset.

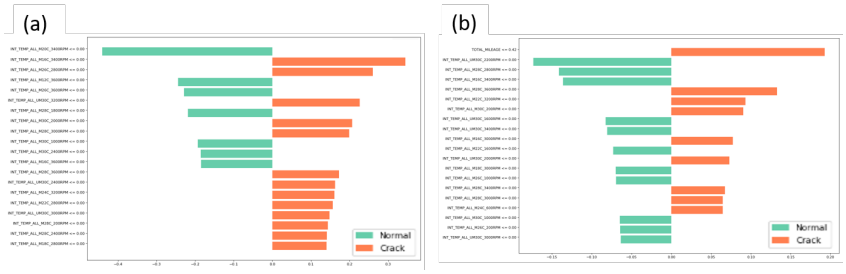

Figure 14. Comparison between new dataset and old dataset with engine water temperature: **(a)** Engine water temperature new dataset. **(b)** Engine water temperature old dataset.

failure reasons have similar portions. Therefore, from the LIME results, we believe the failure vehicle classification is not based on a few features, but rather on the combination of all failure reasons. This conclusion is consistent with the global interpretation result.

Based on the unique interpretation results, if the specific instance classification is based on a data bias, the LIME method can reveal this bias clearly. For unbiased data, the classification reasons can be taken as critical features for reference by domain experts.

6. Conclusion

This paper introduced an ensemble deep learning model for failure prediction. On-board time-series data pertaining to normal and failure cases were used to train and test the proposed model. A two-stack LSTM network and threshold were trained using normal data only. A Gaussian distribution method was used to express the prediction errors. A theoretical derivation was performed based on the threshold to set the loss function to train the entire model. For an actual vehicle dataset, the model predicted all 9 failure cases and 64 normal cases without errors. After the models were well trained for different vehicles, we introduced an online transfer learning algorithm to render the models rapidly adaptive to new targets with little errors. We conducted tests and used the results to verify the algorithm.

Global interpretation analysis revealed the common properties for engine stalling: high intake pressure with high rpm, high intake temperature with high rpm, high engine water temperature, and low engine water temperature with low rpm. Failure reasons for specific instances were obtained through individual interpretation analysis. Hence, the particular failure properties of each instance were illustrated. This approach could provide detailed information for domain experts to analyze problems leading to failure in vehicles.

References

1. Salem, O.; Guerassimov, A.; Mehaoua, A.; Marcus, A.; Furht, B. Anomaly detection in medical wireless sensor networks using SVM and linear regression models. *International Journal of E-Health and Medical Communications (IJEHMC)* **2014**, *5*, 20–45.
2. Liu, X.; Nielsen, P.S. Regression-based online anomaly detection for smart grid data. *arXiv preprint arXiv:1606.05781* **2016**.

3. Wu, Q.; Wu, H.; Zhou, X.; Tan, M.; Xu, Y.; Yan, Y.; Hao, T. Online transfer learning with multiple homogeneous or heterogeneous sources. *IEEE Transactions on Knowledge and Data Engineering* **2017**, *29*, 1494–1507.
4. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.
5. Zhang, Z.; Zhou, X.; Zhang, X.; Wang, L.; Wang, P. A model based on convolutional neural network for online transaction fraud detection. *Security and Communication Networks* **2018**, *2018*.
6. Sun, Y.; Yu, W.; Chen, Y.; Kadam, A. Time Series Anomaly Detection Based on GAN. 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS). IEEE, 2019, pp. 375–382.
7. Sweers, T.; Heskes, T.; Krijthe, J. Autoencoding Credit Card Fraud **2018**.
8. Yuan, M.; Wu, Y.; Lin, L. Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network. 2016 IEEE International Conference on Aircraft Utility Systems (AUS). IEEE, 2016, pp. 135–140.
9. Taylor, A.; Leblanc, S.; Japkowicz, N. Anomaly detection in automobile control network data with long short-term memory networks. 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2016, pp. 130–139.
10. Zhang, R.; Zou, Q. Time series prediction and anomaly detection of light curve using lstm neural network. Journal of Physics: Conference Series. IOP Publishing, 2018, Vol. 1061, p. 012012.
11. Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P. Long short term memory networks for anomaly detection in time series. Proceedings. Presses universitaires de Louvain, 2015, p. 89.
12. Yu, W.; Sun, Y.; Zhou, R.; Liu, X. GAN Based Method for Labeled Image Augmentation in Autonomous Driving. 2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE). IEEE, 2019, pp. 1–5.
13. Geng, M.; Wang, Y.; Xiang, T.; Tian, Y. Deep transfer learning for person re-identification. *arXiv preprint arXiv:1611.05244* **2016**.
14. Laptev, N.; Yu, J.; Rajagopal, R. Applied timeseries transfer learning **2018**.
15. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Transfer learning for time series classification. 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018, pp. 1367–1376.
16. Zhan, D.; Yi, S.; Xu, D.; Yu, X.; Jiang, D.; Yu, S.; Zhang, H.; Shangguan, W.; Zhang, W. Adaptive Transfer Learning of Multi-View Time Series Classification. *arXiv preprint arXiv:1910.07632* **2019**.
17. Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; Lillicrap, T. Meta-learning with memory-augmented neural networks. International conference on machine learning, 2016, pp. 1842–1850.
18. Munkhdalai, T.; Yu, H. Meta networks. Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 2554–2563.
19. Ravi, S.; Larochelle, H. Optimization as a model for few-shot learning **2016**.
20. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 1126–1135.
21. Nichol, A.; Achiam, J.; Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* **2018**.
22. Molnar, C. *Interpretable machine learning*; Lulu. com, 2019.
23. Friedman, J.H. Greedy function approximation: a gradient boosting machine. *Annals of statistics* **2001**, pp. 1189–1232.
24. Goldstein, A.; Kapelner, A.; Bleich, J.; Pitkin, E. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics* **2015**, *24*, 44–65.
25. Apley, D.W. Visualizing the effects of predictor variables in black box supervised learning models. *arXiv preprint arXiv:1612.08468* **2016**.
26. Ribeiro, M.T.; Singh, S.; Guestrin, C. " Why should i trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.
27. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. Advances in neural information processing systems, 2017, pp. 4765–4774.

 doi:10.20944/preprints202107.0040.v1