Enhancing failure prediction from time series histogram data:
through fine-tuned lower-dimensional representations

M.Sc. in Computer Science & Engineering (120 ECTs)

Halmstad 2023-09-29
Vijay Jayaraman
"Dedicated to the unwavering support and love of my family:
To my beloved son Kabir, whose laughter and curiosity inspire me daily,
To my loving wife Anisha, whose strength and encouragement sustain me,
To my father Jayaram, whose guidance and support have shaped my journey.
And to my late mother Manjula, whose memory lives on in my heart,
This thesis is a testament to the love and inspiration you have all brought into my life."
Histogram data are widely used for compressing high-frequency time-series signals due to their ability to capture distributional information. However, this compression comes at the cost of increased dimensionality and loss of contextual details from the original features. This study addresses the challenge of effectively capturing changes in distributions over time and their contribution to failure prediction. Specifically, we focus on the task of predicting Time to Event (TTE) for turbocharger failures.

In this thesis, we propose a novel approach to improve failure prediction by fine-tuning lower-dimensional representations of bi-variate histograms. The goal is to optimize these representations in a way that enhances their ability to predict component failure. Moreover, we compare the performance of our learned representations with hand-crafted histogram features to assess the efficacy of both approaches.

We evaluate the different representations using the Weibull Time To Event - Recurrent Neural Network (WTTE-RNN) framework, which is a popular choice for TTE prediction tasks. By conducting extensive experiments, we demonstrate that the fine-tuning approach yields superior results compared to general lower-dimensional learned features. Notably, our approach achieves performance levels close to state-of-the-art results.

This research contributes to the understanding of effective failure prediction from time series histogram data. The findings highlight the significance of fine-tuning lower-dimensional representations for improving predictive capabilities in real-world applications. The insights gained from this study can potentially impact various industries, where failure prediction is crucial for proactive maintenance and reliability enhancement.
It isn’t what you don’t know that gets you into trouble,
it’s what you know for sure that just isn’t so.
— Mark Twain

ACKNOWLEDGEMENTS

Firstly, I would like to thank the entire Strategic Product Planning & Advanced Analytics team at the Service & Maintenance function at Scania, for giving me the opportunity to do my thesis with them. Also, I’m grateful to my thesis advisor at Scania - Olof Steinert, for his guidance and insights on predictive maintenance. I would also like to thank Maharshi Dhada, for passing on his knowledge from his research work, which has formed the premise to this thesis. Secondly, I would like to thank my academic supervisors Thorsteinn Rögnvaldsson and Abdallah Alabdallah for their relentless support, constructive feedback and their constant nudge to do my best. I also would like to thank Peyman Mashhadi and Yuantao Fan for supporting in spite of their limited time. Lastly, I would like to thank my family and friends for always standing by side and rooting for me.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left: Survival studies in actual time, Right: survival studies in normalized timelines to start at $t=0$. Image credits: <a href="https://scikit-survival.readthedocs.io">https://scikit-survival.readthedocs.io</a></td>
</tr>
<tr>
<td>2</td>
<td>Illustration of Time-series histogram for a particular Operational attribute for different subject vehicles (e.g. $v_1,v_2,v_3$)</td>
</tr>
<tr>
<td>3</td>
<td>Left: Boost Pressure 1-D Histogram Snapshots Visualized. Right: Turbine Speed 1-D Histogram Snapshots Visualized</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Illustration of Histogram Snapshots for each vehicle sample</td>
</tr>
<tr>
<td>6</td>
<td>Data preparation pipeline including preprocessing and data preparation for feature learning and survival modeling.</td>
</tr>
<tr>
<td>7</td>
<td>Left: Average of 1000 2-D histogram snapshots; Center: Individual 1-D histogram snapshots after summarising 2-D histogram along the y-axis, indicating the 1-D histogram of the load; Right: Individual 1-D histogram snapshots after summarising 2-D histogram along the x-axis, indicating the 1-D histogram of the engine RPM.</td>
</tr>
<tr>
<td>8</td>
<td>Feature engineering pipeline for generating Benchmark Features</td>
</tr>
<tr>
<td>9</td>
<td>Autoencoder feature learning process</td>
</tr>
<tr>
<td>10</td>
<td>Feature Learning architecture</td>
</tr>
<tr>
<td>11</td>
<td>Flow matrix feature preparation process</td>
</tr>
<tr>
<td>12</td>
<td>EMD Flow matrix between the first snapshot and the last snapshot of the vehicle. Color coding: red indicates a change in the starting distribution, green indicates a change in the ending distribution, and yellow indicates unchanged.</td>
</tr>
<tr>
<td>13</td>
<td>Prediction of failure distribution using the WTTE-RNN network</td>
</tr>
<tr>
<td>14</td>
<td>Illustration of Padding and Masking process to deal with variable sequence length</td>
</tr>
<tr>
<td>15</td>
<td>Time-segment based evaluation</td>
</tr>
<tr>
<td>16</td>
<td>Bottleneck layer comparison of train and validation reconstruction loss for Convolutional Neural Network - Autoencoder (CNN-AE)</td>
</tr>
<tr>
<td>17</td>
<td>Bottleneck layer comparison of train and validation reconstruction loss for Autoencoder (AE)</td>
</tr>
</tbody>
</table>
Figure 18  Comparative analysis of lower dimensions versus the amount of minimum reconstruction error achieved. 30

Figure 19  Comparison of training and validation losses of CNN-AE and the AE model after fine-tuning. 31

Figure 20  Comparison of the error on a test set of CNN-AE and AE model after fine-tuning. 31

Figure 21  WTTE Results comparing the tuned and untuned CNN-AE features with 95% prediction interval on test set. 34

Figure 22  WTTE Results With 95% prediction interval on test set. 35

Figure 23  Survival score using c-index with 95% prediction interval on test set. 36

Figure 24  Error in prediction RUL, when trained on different features: 1) Only CNN fine-tuned features, 2) Benchmark Features + CNN fine-tuned features, 3) Benchmark features. 36

Figure 25  Different predicted RUL and the True RUL averaged for all the vehicles in the dataset. 37

Figure 26  WTTE-RNN vs RNN performance on benchmark features. 37

Figure 27  Earth Mover Distance (EMD) distance visualization of the 2-D Histogram of Engine load on the y-axis versus the Engine RPM on the x-axis of trucks that experienced a failure. Color coding: Red channel indicates a change in the starting distribution; the green channel indicates a change in the ending distribution; yellow indicates "unchanged". The arrows indicate the flow determined using EMD, where the thickness of the arrow is proportional to the amount of mass flow. 38

Figure 28  EMD distance visualization of the 2-D Histogram of Engine load on the y-axis versus the Engine RPM on the x-axis of trucks that did not experience a failure. Color coding: The red channel indicates a change in the starting distribution; the green channel indicates a change in the ending distribution; yellow indicates "unchanged". The arrows indicate the flow determined using EMD, where the thickness of the arrow is proportional to the amount of mass flow. 39

Figure 29  Time-Segment based mean absolute error comparing the two feature types 1) EMD flow features; 2) Benchmark features. 39
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Baseline experiments on a simple WTTE-RNN Network to understand feature importance; Benchmark features refer to the hand-crafted features used in [3].</td>
<td>26</td>
</tr>
<tr>
<td>Table 2</td>
<td>Simple WTTE-RNN Architecture</td>
<td>27</td>
</tr>
<tr>
<td>Table 3</td>
<td>CNN-AE Encode-Decoder Architecture</td>
<td>28</td>
</tr>
<tr>
<td>Table 4</td>
<td>AE Encode-Decoder Architecture</td>
<td>28</td>
</tr>
<tr>
<td>Table 5</td>
<td>Fine Tuning Architecture for CNN-AE</td>
<td>29</td>
</tr>
<tr>
<td>Table 6</td>
<td>WTTE-RNN Architecture</td>
<td>32</td>
</tr>
<tr>
<td>Table 7</td>
<td>RNN Architecture</td>
<td>33</td>
</tr>
<tr>
<td>Table 8</td>
<td>Summary of WTTE-RNN results with 95% prediction interval</td>
<td>35</td>
</tr>
<tr>
<td>Table 9</td>
<td>Summary of results</td>
<td>35</td>
</tr>
<tr>
<td>Table 10</td>
<td>Modelling of Survival SVM on 1) EMD flow features; 2) Benchmark features. Modelling is static, meaning each snapshot's Remaining Useful Life (RUL) is estimated</td>
<td>40</td>
</tr>
</tbody>
</table>
ACRONYMS

DTC  Diagnostic Trouble Code
RUL  Remaining Useful Life
SVM  Support Vector Machine
DRSA Deep Recurrent Survival Analysis
RPM Rotations Per Minute
TTE Time to Event
PR Precision Recall
MSE Mean Squared Error
PDF Probability Density Function
RSF Random Survival Forest
AUC Area Under the Curve
SSVM Survival Support Vector Machine
AE Autoencoder
RNN Recurrent Neural Network
LSTM Long Short Term Memory
WTTE-RNN Weibull Time To Event - Recurrent Neural Network
T-SNE T-Distributed Stochastic Neighbor Embedding
CNN Convolutional Neural Network
CNN-AE Convolutional Neural Network - Autoencoder
EMD Earth Mover Distance
MAE Mean Absolute Error
INTRODUCTION

1.1 PROBLEM CONTEXT

Data-driven prognostics pose a significant challenge for automotive manufacturers as they seek to enhance fleet managers’ experience and mitigate losses caused by unexpected vehicle downtime. The core of the problem lies in optimizing maintenance schedules and proactively addressing vehicle component failures, thereby improving vehicle service prognostics. Unplanned failures of vehicle components result in substantial losses, primarily due to extended downtime and potential delays in obtaining spare parts, as well as increased expenses for roadside assistance or towing. An effective approach to failure prognostics can empower fleet owners to plan service visits strategically and address existing issues promptly. This, in turn, enables vehicle manufacturers to better prepare for situations where components might not exhibit apparent issues but are prone to failure in the near future. Thus, modeling failure prognosis for different vehicle components becomes a crucial factor in enhancing the overall vehicle service experience for both suppliers and consumers.

In this research, we present an approach focused on estimating the RUL of a specific time-critical component: the turbocharger. While this might seem straightforward, we encounter several challenges:

- **Lack of Direct Health Status Data:** Direct health status measurements for the turbocharger are unavailable, making it difficult to assess its condition accurately.

- **Irregular Sampling and Histogram Representation:** The sensor data related to the turbocharger is irregularly sampled and presented in the form of histograms, posing a unique data processing challenge.

- **Component-Level Data Resolution:** The data is at the component level rather than representing specific fault modes or Diagnostic Trouble Codes (DTCs), making it challenging to pinpoint the exact cause of potential failures.

- **Rare Occurrence of Turbocharger Failures:** Turbocharger failures are infrequent events, further complicating the development of accurate prognostic models.

While most of the above challenges can be addressed during the modeling process, the absence of direct health status indicators emerges as the most significant hurdle in solving this prognostic problem. Business and data contract decisions have led to the summarization of raw sensor data from different vehicles into histogram distributions. This summarization results in the loss of specific sensor values for
each operational variable, posing challenges in extracting meaningful information using feature learning and subsequently evaluating their effectiveness in failure prognostics.

In this research, we focus on failure prognostics through time-to-event prediction, also known as remaining useful life estimation. This area is closely related to predictive maintenance, which requires actionable failure prognostics for real-world applications. As a comprehensive and in-depth study, our aim with time-to-event prediction is to take a step forward in developing practical and effective prognostic solutions.

1.1.1 Histogram Feature Learning

Feature learning is essential in understanding the various variations present in histogram distributions and their relationship to depicting a vehicle’s state as faulty or not. However, a significant challenge arises when dealing with time series histogram data, where the temporal information of variables is discarded, resulting in time-series of histograms capturing temporal changes in variable distribution. Despite this challenge, such data holds promise for prognostic modeling, necessitating additional feature engineering to ensure effective model input.

Capturing distinguishability between histograms is crucial, as it indicates variations or trends towards degradation or excessive strain in the long or short term. Furthermore, reducing dimensions in the spatial domain becomes vital for improved efficiency and better modeling practices.

The prevalence of histogram usage in the automotive industry, particularly in optimizing maintenance schedules and reducing maintenance costs through improved prognostic modeling is well-established. While predicting part reliability with histogram data may be challenging and not the focus of this research or Scania’s specific goals, the need for feature learning remains critical for effective failure prognostic modeling.

In this research, we propose a deep learning-based feature learning method aimed at effectively learning time series histogram data of different operational variables to best estimate the survivability of turbochargers in a population of trucks.

1.2 Research Questions

After describing the problem statement and providing background information in this area, the scope of this research is defined by addressing the following questions:

1. What representations can be learned from histogram features, including both hand-crafted and neural-network-based learned features, that would be beneficial in failure prognostics?
2. How does the performance of failure prognostic models compare when using hand-crafted histogram features versus neural-network-based learned representations?

1.3 Novelty

When modeling failure prognostics using histogram features, the focus has predominantly been on improving modeling practices by utilizing various methods to represent the data in static or time-dependent modes. However, limited attention has been given to understanding how to extract the most informative features to enhance performance.

Feature learning can be achieved either through manual crafting or representation learning methods such as AE. The problem with hand-crafted features lies in the vast number of potential feature combinations applicable in a single scenario, resulting in a daunting number of experiments to identify the most effective methods. An example of hand-crafted features can be found in [3], where histograms were transformed into point features using similarity measures. While this approach yielded seemingly reasonable results, questions remain regarding the trade-off between compression and retention of relevant information.

On the other hand, AE offers neural network-based trainable feature extractors that often learn low-dimensional general representations of the data. An example in this domain is presented in [12], where the focus was on using these general representations in a multi-task learning scenario. However, the encoder-decoder architecture of AE models tends to focus on reproducing the input, lacking a specific focus on features tailored for failure prognostics. To address this, transfer learning and fine-tuning can be employed with labeled target variables. In this case, however, we only have censoring information and RUL data, which are not applicable to non-failed vehicles. Therefore, using these labels would not be empirically accurate.

This work proposes a novel approach that involves fine-tuning lower-dimensional representations to facilitate the selection of useful features for the failure prognostics problem. Given that we lack any information on the "health" status of a vehicle during its operation, fine-tuning the network requires an alternate approach. The proposed fine-tuning approach outperforms the network learning general representations when evaluated on the WTTE-RNN network for time-to-event prediction.

The upcoming chapters will delve into the essential aspects of the problem being addressed and provide in-depth explanations of the methodology employed in this research.
1.4 Background

1.4.1 Censoring

In survival analysis, censoring is an important part of how the data is interpreted. Out of the sample set of vehicles that we use in this research, it comprise of vehicles; 1) Ones that in the period of the study underwent a repair, which we will interpret as failure as we don’t have the actual time of failure; 2) Ones that did not experience a failure during the entire study period; 3) Ones that left the study mid-way, and now we have no information on whether this vehicle experienced a failure or not in the future. Here cases 2 & 3 would be classified as right censoring of the data, wherein, we have data from the start of its lifetime but not until the actual end of its lifetime. We can see from figure 1(left), how our typical studies look like. There are a few cases that experienced a failure and some did not or were not captured due to censoring. But for ease of modeling, we normalize the enrollment time as depicted in 1(right).

Formally, we can define the samples with right censoring in this way, where \( y \) is given as the observable time and \( \delta \) is given as the event indicator \([0,1]\), \( t > 0 \) is the time when an event occurred and \( c > 0 \) is the time of censoring. Here the event indicates whether a failure happened or not.

\[
y = \begin{cases} 
  t, & \text{if } \delta = 1 \\
  c, & \text{if } \delta = 0 
\end{cases}
\]  

(1)

Figure 1: Left: Survival studies in actual time, Right: survival studies in normalized timelines to start at \( t=0 \). Image credits: https://scikit-survival.readthedocs.io
2 RELATED WORK

2.1 HISTOGRAM FEATURE LEARNING

In previous work, feature learning histogram data in prognostic modeling scenarios has previously found applications of using similarity or distance measures to understand deviations in distributions and how they indicate faulty behavior. The main reasoning behind using such metrics is its capacity to reduce the number of dimensions drastically to point features and statistically still hold meaning. In [5], the authors did a comparative study of various distance metrics in an unsupervised setting, in which the state of the histogram snapshot is derived based on its similarity from a central healthy snapshot. The results showed Hellinger and Cosine distance measures being the most sensitive to variations in snapshots of histogram data. This study was however performed on a high-frequency regular sampling setting (1Hz), which is different from the very low-frequency irregular sampling setting we have here (more about the data in chapter 3). In [3], the authors in collaboration with Scania, modeled failure prognosis on very similar data as used in this research. They used distance measures to depict variation in the state of time-series histogram snapshots by using; 1) the Distance of each snapshot in the series from its respective sample’s first snapshot, and 2) the Distance of each snapshot from the antecedent snapshot in the series. For this, they found reasonable success using entropy-based similarity measures like intersection distance and also proposed using the K-L Divergence for pre-processing the histogram data. In [2], the authors introduce a comprehensive study on different similarity measures to be used to compare normalized distributions which also include the above metrics. One key issue with working with similarity measures pointed as by [11] is that most of the similarity measures don’t give a distance value for non-overlapping parts when comparing two histograms, hence we could potentially be missing out on information. As mentioned, learning such features by comparing different similarity measures can be time-consuming to understand which features could prove to be more useful than others and in which scenarios.

In previous research, the focus has been on feature learning from histogram data in prognostic modeling scenarios. This has led to the application of similarity or distance measures to understand deviations in distributions and their indication of faulty behavior. The use of such metrics is motivated by their ability to drastically reduce the number of dimensions to point features while retaining statistical significance.

For instance, in [5], the authors conducted a comparative study of various distance metrics in an unsupervised setting, deriving the state of histogram snapshots based on their similarity to a central healthy
The study revealed that Hellinger and Cosine distance measures were the most sensitive to variations in histogram snapshot data. However, it’s worth noting that this research was conducted on data with high-frequency regular sampling (1Hz), which contrasts with the low-frequency irregular sampling used in our study (detailed in Chapter 3).

Similarly, in [3], in collaboration with Scania, modeled failure prognosis on comparable data to what we are using in this research. They utilized distance measures to depict variations in the state of time-series histogram snapshots by measuring the distance of each snapshot in the series from its respective sample’s first snapshot, and the distance of each snapshot from the antecedent snapshot in the series. Entropy-based similarity measures, like intersection distance, were found to be effective, and they also proposed using K-L Divergence for preprocessing the histogram data.

In a comprehensive study by [2], different similarity measures for comparing normalized distributions were discussed, including those mentioned earlier. An important issue raised by [11] in working with similarity measures is that most of them do not provide a distance value for non-overlapping parts when comparing two histograms, potentially leading to information loss.

While using similarity measures for feature learning can be valuable, it can also be time-consuming to understand which features may be more useful in specific scenarios.

### 2.2 Representation Learning and Predictive Maintenance

Recently, representation learning using neural networks has achieved significant success. Its popularity initially emerged from computer vision applications, where high-dimensional image and video data necessitate effective dimensionality reduction, as demonstrated in works such as [14] and [4].

In the domain of Predictive Maintenance, researchers in [12] employed representation learning to extract low-dimensional bi-variate histogram data in a multi-task supervised learning setting. Notably, they achieved promising results by combining T-Distributed Stochastic Neighbor Embedding (T-SNE) and AE for the detection task. However, it should be noted that T-SNE exhibited computational challenges for higher-dimensional representations.

For higher-dimensional bottleneck scenarios, both AE and Convolutional Neural Network (CNN)-AE showed favorable performance in the multitask setting. Despite these advancements, the potential utility of learned low-dimensional representations from autoencoders in failure prognostics remains to be thoroughly explored. We need to determine whether these learned representations are more effective in combination with hand-crafted features or even outperform them entirely.
Several studies have modeled irregularly sampled histogram data with limited emphasis on feature learning. In [6], the authors explored the failure prediction of NOx sensors using histogram data. However, their approach was static, relying on the most recent snapshot of histogram features and employing Random Forest and Random Survival methods without considering the temporal dimension. Although this work was one of the early explorations into using histogram data for failure prognostics, feature learning was not a major focus.

Similarly, in [8], the author modeled turbochargers’ failure using sparse irregularly sampled histogram data and applied Survival Phase-Long Short Term Memory (LSTM). Though the Survival Phase-LSTM model outperformed Random Survival Forest in differentiating failed cases from censored cases and predicting risk, there was little emphasis on feature learning.

In [13], the authors employed LSTM to build a prognostic model for predicting the failure of lead acid batteries using histogram data.

[9] introduced the novel WTTE-RNN, which gained popularity in churn prediction where customer event sampling is irregular. It has also been applied in vehicle prognostics by [10], using manually crafted point features derived from original histogram features. The WTTE-RNN network’s main advantage lies in its inherent design for solving time-to-event prediction problems with multi-variate features, making it suitable for failure prognostics. The network learns to address both censored and uncensored samples, as taken into account in its proposed loss function, providing parameters of the Weibull distribution.

However, further investigations are required to determine which additional features can enhance the existing hand-crafted features and improve modeling performance.
3 DATA PRE-PROCESSING

3.1 DATASET

The dataset for this project is provided by Scania CV and consists of a pool of at least 31,000 vehicles. Each vehicle contains approximately 1 to 130 time snapshots of data collected during the study period. In this study, we are specifically dealing with right-censored vehicles for the prognostic analysis.

Notably, all vehicles in the dataset have operated on a shared timeline, but with different study start and end dates. This common timeline implies that the components used by these vehicles, as well as the data collection and processing strategies, are similar, eliminating the need for data-cleaning methods.

Figure 2 illustrates time-series histogram data used in this study. The figure depicts the timeline of one operational attribute (e.g. engine RPM) for different vehicle subjects in the study. It also depicts the irregularity in the frequency of binned histogram snapshots.

![Figure 2: Illustration of Time-series histogram for a particular Operational attribute for different subject vehicles (e.g. v1,v2,v3)](image)

3.2 DATA DESCRIPTION

An illustration of the data is shown in Figure 5. This research primarily focuses on testing the efficacy of operational data; therefore, no other specification or numerical features were used.

- **Operational Data**: These features comprise the main operational data captured from various sensors of the vehicles. The operational data mostly consists of histogram data, accumulated over different time snapshots. The histogram data is irregularly sampled, depending on the frequency of the vehicle’s service visits and data contract, which determines how often the vehicle’s
data is uploaded to the servers. This group includes two categories of data: 1-D histogram features and 2-D histogram features. The features in this group are as follows:

- **1-D Histogram Features**: Boost Pressure (10 bins), Turbine Speed (10 bins)

![Figure 3: Left: Boost Pressure 1-D Histogram Snapshots Visualized. Right: Turbine Speed 1-D Histogram Snapshots Visualized](image)

- **2-D Histogram Features**: Engine Load Matrix (Engine Load Percentage (11 bins) x Rotations Per Minute (RPM) (12 bins) = 132 cells)

![Figure 4](image)

It’s important to note that the operational data doesn’t directly measure the health status but is instead the frequency of raw sensor values that have been binned into histograms. Therefore, a measure of health has to be derived from these bin values.

- **Operating Time**: The operating time feature can be derived by summing the turbine speed data along the histogram axis for each vehicle, which would give the operating time for that snapshot. This derived feature will be helpful for future analysis. The operating time is used to derive the countdown function of time, which will give us the **RUL** at each snapshot. The derived **RUL** is used as one of the targets for the survival modeling.

- **Censor**: We are also provided with a feature that indicates whether the vehicle underwent a repair (failure event) during the study period or was censored. The data consists of only right-censored cases. This forms the second target for the survival modeling.
Population Pruning for Feature Learning

To ensure a clear understanding of modeling performances in this research, certain decisions were made in collaboration with the decision-makers at Scania CV to down-sample clean data. The population pruning was conducted as follows:

1. **Missing Values**: Out of the original population, approximately 7,000 vehicles were found to have at least one null value in any of their bin features. Since the underlying reasons for these null value occurrences are still unknown, and to avoid introducing potential biases through imputation, we chose to drop all samples that had a single null value in any of their snapshots.

2. **Lifetime Operating Time**: To ensure a minimum runtime in vehicles and to have sufficient snapshots, a decision was made to filter vehicles that have accumulated at least 100 units of operating time in their lifetime.

3. **Minimum and Maximum Snapshot Frequency**: The data we have is irregularly sampled over a calendar period, which does not directly indicate run-time or operation period. For consistency, we consider the operating time of the vehicles as our time indicator. Thus, we filter vehicles that have a sampling duration of 60 operation time units in at least one instance. Additionally, we combine snapshots that have a sampling duration of less than 7 operation time units. Finally, we have a sampling duration ranging between 7 and 60 operation time units.

After this pruning, we are left with approximately 16,000 vehicles with snapshots ranging between 3 and 42, resulting in about 200,000 rows in total. This pruned population of data will be used for feature-learning purposes.

Population Pruning for WTTE-RNN Training

In the experiments explored by the authors on the effect of WTTE-RNN [9] and in other works like [10], it was found that the WTTE-RNN network is prone to instability when using an imbalanced set
of censored to uncensored cases during training. To address this, we use a ratio of 1:1 of uncensored to censored samples for training. After applying this pruning, we are left with approximately 2,200 vehicle samples, resulting in about 25,000 rows in total.

### 3.4 Data Pipeline

![Data preparation pipeline including preprocessing and data preparation for feature learning and survival modeling.](image)

The data pipeline is depicted in Figure 6. The pipeline illustrates the different steps applied to the data before feature learning and survival modeling. The data is first received in the form of accumulative frequency counts, after which population pruning is performed as mentioned above. The remaining steps are explained below:

1. **Difference Histograms**: The histograms from the original accumulative forms are differenced between consecutive histograms specific to the samples. This process yields a profiled snapshot, providing insights into the occurrences during the interval of the snapshots.
   
   \[ H_{\text{delta}}^{t} = H_{i}^{t} - H_{i}^{t-1}, \text{ where } H_{i}^{t} > H_{i}^{t-1} \text{ for all snapshots.} \]
   
   Here: \( i \) refers to a bin feature, \( t \) refers to the time index of the snapshots.

2. **Summarising 2-D Histograms**: The dataset contains two types of histogram features - 1-D histograms and 2-D histograms. Optionally, the 2-D histograms can be summarised along the axes to obtain two constituent 1-D histograms. While this step is performed for benchmark features in [3], it is not done for the feature learning process. Figure 7 shows the depiction of 2-D histograms and their summarised 1-D histograms.

3. **Normalising Histograms**: The histograms are normalized to depict a probability density function, with values in the range \([0, 1]\), instead of the existing frequency counts. This is achieved by dividing each bin feature by the sum of all bin values.

   \[ H_{\text{norm}}^{t} = \frac{H_{i}^{t}}{H_{\text{sum}}^{t}}, \text{ for every } i \text{th bin for each histogram.} \]

   Here: \( H_{\text{sum}}^{t} = \sum_{i=0}^{N} H_{i}^{t}, \ N \rightarrow \text{number of bins.} \)
Figure 7: Left: Average of 1000 2-D histogram snapshots; Center: Individual 1-D histogram snapshots after summarising 2-D histogram along the y-axis, indicating the 1-D histogram of the load; Right: Individual 1-D histogram snapshots after summarising 2-D histogram along the x-axis, indicating the 1-D histogram of the engine RPM.

4. **Data Split**: For the purpose of feature learning, the data is split into train, validation, and test sets stratified based on the censor target variable. This split is performed to retain the distribution of the data across all subsets, with a ratio of 70:10:20 for train, validation, and test splits, respectively.
In this section, we will describe different representations that will be used for the 1-D and 2-D histogram features as part of the feature engineering process. Our ultimate goal is to perform time-to-event prediction, and thus, we will discuss the prognostic models, how we model them, and how we evaluate their performance.

4.1 FEATURE REPRESENTATIONS

Histogram features, as discussed previously, provide limited contextual information and therefore require feature engineering to extract their full potential. In this section, we will explore several representations of histogram data, aiming to capture variations that can potentially improve the estimation of remaining useful life. Starting with the work developed by the authors of [3], we will build upon their Benchmark Features to incorporate additional information. These Benchmark Features are essentially hand-crafted features, and we will evaluate them using the WTTE-RNN prognostic model.

The next part of this section delves into the autoencoder-based representation learning method. We propose a novel approach to fine-tune the network, and the prognostic model used to evaluate these features is the WTTE-RNN.

Lastly, we explore the use of EMD Flow features, which contain the flow information between distributions. These features will be evaluated using a static survival regression model, such as SSVM, instead of the temporal WTTE-RNN. In this approach, each snapshot is treated as an individual sample.

4.1.1 Benchmark Features

The feature engineering method adopted by [3] will serve as our Benchmark features. The processing of these features is depicted in Figure 8, an extension of the pipeline shown in Figure 6. The 1-D Histogram features of Boost Pressure and Turbine Speed were used as is, while the 2-D Histogram of the engine-load matrix was summarized along each axis to obtain two 1-D constituent Histograms, one for the...
engine and the other for the load. These constituent histograms were then converted into two major point features for each operational variable. To achieve this conversion, the authors tracked their entropy across two features using the intersection similarity measure.

The point features created from each histogram variable are as follows:

1. **Similarity feature from healthy snapshot** - The intersection similarity measure was used to compare each snapshot in a sample with its corresponding first snapshot. Mathematically, this is illustrated as the intersection distance of $A_{t_0}$ (sample’s initial snapshot) with $A_t$ (current snapshot):

$$\text{Dist}(A_{t_0}, A_t) = \sum_{i=0}^{N} \min(A_{t_0}, A_t)$$

2. **Similarity feature from antecedent snapshot** - The intersection similarity measure was used to compare each snapshot with its preceding snapshot within the same sample. Mathematically, this can be represented as the intersection distance of $A_{t-1}$ (antecedent snapshot) with $A_t$ (current snapshot), both from the same sample.

$$\text{Dist}(A_{t-1}, A_t) = \sum_{i=0}^{N} \min(A_{t-1}, A_t)$$

After converting them into point features, the temporal dimension of the features is kept intact, giving us 2 point features for every snapshot in every sample for every operational variable. While the aim of their research was to compress the histogram features into point features, the loss of information from the 2-D histogram was done to further simplify the dimensions instead of dealing with varying dimensional data. This leads us to explore the gap of using the 2-D histogram without summarizing along axes using the autoencoder-based feature learning method to automatically select important features that may have been previously missed during summarization.

An alternative to the author’s decision to attribute ‘healthy’ snapshot status to each vehicle’s first snapshot is to use a central healthy snapshot. The central healthy snapshot is the average of all the first snapshots for each vehicle sample. This helps give each sample a deviation from a certain constant, rather than using individual ‘healthy’ snapshots. It also promotes more collaborative learning among different samples in the dataset.

### 4.1.2 Autoencoder Feature Learning

Before discussing the architecture in detail, let’s highlight the different components of the feature learning method, as depicted in Figure 9. As mentioned earlier, the 2-D histogram has been unexplored by the authors of [3]. Therefore, in the autoencoder feature learning process, we take advantage of this and face the challenge of effectively reducing the features, originally 132 dimensions, to lower-dimensional representations. To achieve this, we draw inspiration from [12] and apply the Autoencoder (AE) or Convolutional Autoencoder (CNN-AE) to learn the bottleneck features.
For the AE-based feature learning, we propose a novel method of performing transfer learning and fine-tuning on the trained encoder. In this scenario, we do not have a reasonable target feature to tune our networks. Instead, we tune the encoded representation to model the intersection similarity between the 2-D histogram and the central 'healthy' snapshot. This helps the model learn the deviation of the histograms from a healthy state. The general architecture of this approach is depicted in Figure 10.

The first part of the network depicts the standard autoencoder architecture, where we intend to use either the CNN-AE or the plain AE. Here, the role of the autoencoder is to learn lower dimensions by reconstructing itself from the input. The second part of the network displays transfer learning of the lower-dimensional representations. The second part uses only the encoder in inference mode (by making the parameters non-trainable) and adds another dense layer of the same shape as that of the bottleneck layer. After that, there is a
sigmoid layer as we proceed to perform regression to model the one-dimensional intersection distance between the input snapshot and the central ’healthy’ snapshot. The extra dense layer that is added acts as a moderator, focusing the low-dimensional representation from the previous layer and tuning it to learn deviations from the healthy status. The third part of the network is the inference of the fine-tuned network. The output of the added dense layer during the fine-tuning phase is then used to convert the 2-D histogram inputs into learned low-dimensional representations. This can now be used to train the WTTE-RNN to learn from the temporal dimension aspect of these features.

In the architecture shown in 10, the highlighted red box encompasses the entirety of my contribution. This work continues to build upon and improve the research conducted by [3]. In their work, the authors exclusively utilized 1-D time-series histograms, converting them into point features. One of these point features is the intersection distance from different histogram snapshots to each vehicle’s respective first histogram snapshot. However, a significant gap in their research was the unexplored territory of 2-D histograms. Therefore, this thesis centers on the utilization of autoencoders to extract features from time-series 2-D histograms. To fine-tune the autoencoders, the distance to the average of all vehicles’ first snapshots was employed. The rationale behind this choice is as follows:

1. [5] used the distance from the average of histogram snapshots to detect deviations from the normal behavior of components. Calculating the average of all snapshots in the Scania dataset posed computational challenges due to the numerous snapshots from over 30,000 vehicles. Therefore, selecting the average of the first snapshots was a more feasible option.

2. In [3]’s work, they assumed that each vehicle’s first snapshot represented a “healthy” state of the component. However, experts at Scania believed that this assumption might not always hold true, as the data collection process could start at a time much later than the vehicle’s initial operation. Consequently, calculating the average of the first snapshots provides a more comprehensive understanding of the initial state compared to individual snapshots.

4.1.3 Flow matrix features

To understand the variations of the 2-D histogram data during its healthy phase and at the end of the study phase, we use flow matrices generated by using the Earth Mover’s Distance (EMD) between any two distributions. The EMD distance solves an optimization problem to determine the least amount of energy required to transform one distribution into another. An example of the EMD flow matrix between the first and last snapshot is visualized in Figure 12.

The feature preparation process is illustrated in Figure 11. As mentioned above, the EMD distance provides an estimate of the minimum
4.2 Prognostic Models

In the context of this research, our objective is to establish a model that captures the relationship between histogram features and their temporal variations leading to failures. Having explored various ways to represent histogram data effectively, we are now poised to assess these representations in estimating the remaining useful life of the turbocharger. In the subsequent sections, we delve into the discussion of prognostic models for this purpose.
4.2.1 Working of WTTE-RNN Network

In this section, we will discuss the theoretical framework underlying the operation of the WTTE-RNN network [9]. The WTTE-RNN is particularly notable for its capability to incorporate survival analysis within an RNN architecture. While the original WTTE-RNN model introduced by the authors in [9] addresses the general case of re-occurring events for a single subject, it is important to clarify whether this applies to your data in this study.

The primary innovation in the WTTE-RNN lies in its novel loss function, which is minimized during the neural network’s training process to estimate the parameters of the Weibull probability distribution for time-to-event predictions.

The specific structure of the loss function is outlined below:

\[
\log(\mathcal{L}) = \sum_{n=1}^{N} \sum_{t=0}^{T_n} \left[ u_n \log \left( \Pr(Y^n_t = y^n_{t|0:t}) \right) + (1 - u_n) \log \left( \Pr(Y^n_t > y^n_{t|0:t}) \right) \right]
\]

(2)

Where: \( u^n_t \) : Indicates the censoring label; 1 -> observable, 0 -> non-observable
\( u^n_t \log \left( \Pr(Y^n_t = y^n_{t|0:t}) \right) \): Indicates the case where we have the sample that is observable, in such a case we want the loss function to converge the predicted time-to-event \( Y^n_t \) to the provided time-to-event \( y^n_t \) target feature as it is the actual information of failure (or repair).
\( (1 - u^n_t) \log \left( \Pr(Y^n_t = y^n_{t|0:t}) \right) \): This term considers the censored cases, where the target time-to-event that we have does not indicate failure or repair but indicates an end of study therefore, the loss should be minimized for cases where \( Y^n_t \) is greater than that of the end of study event \( y^n_t \).
\( T_n \): Indicates the number of snapshots in this sample
\( N \): Indicates the number of samples in the dataset

The probabilities that we have in this equation is a probability distribution from survival analysis, so we assume the distribution to be a Weibull distribution, the loss function employed in the WTTE-RNN is defined as follows:

\[
\log(L_{\lambda}) = \sum_{n=1}^{N} \sum_{t=0}^{T_n} \left( u^n_t \left[ \exp \left( \frac{y^n_t + 1}{\alpha^n_t} \right)^{\beta^n_t} - \left( \frac{y^n_t}{\alpha^n_t} \right)^{\beta^n_t} \right] - \frac{y^n_t + 1}{\alpha^n_t} \right) \beta^n_t
\]

Here, the variables used in the equation are explained as follows:

- \( u^n_t \): Censoring label; 1 for observable, 0 for non-observable
- \( y^n_t \): Target time-to-event feature
- \( \alpha^n_t \) and \( \beta^n_t \): Parameters of the Weibull distribution
- \( T_n \): Number of snapshots for the sample \( n \)
- \( N \): Total number of samples in the dataset
The parameters $\alpha$ and $\beta$ of the Weibull distribution are learned through optimization of the loss function. The $\alpha$ parameter reflects the anticipated time until event occurrence, while $\beta$ indicates the confidence level of the prediction. The WTTE-RNN outputs these parameters, which are then used to construct a probability distribution for failure prediction. The time-to-event estimation is obtained from the mode of this distribution. The WTTE-RNN takes into account the historical information up to the current time-step to predict the remaining useful life, as depicted in Figure 13.

![Figure 13: Prediction of failure distribution using the WTTE-RNN network](image)

To accommodate multi-length sequences, padding is applied to ensure uniform snapshot counts for all sequences during batch training [7]. A masking layer is also introduced in the network to disregard values that are masked. For padding, an arbitrary value of -99 is used to prevent confusion with valid data, as our dataset lacks values in this range. Masked snapshots are handled during evaluation by using the last valid snapshot for prediction continuation. To ensure accurate error evaluation, masked snapshots are excluded to prevent bias. The padding and masking mechanisms are illustrated in Figure 14.

![Figure 14: Illustration of Padding and Masking process to deal with variable sequence length](image)

### 4.2.2 Survival Support Vector Machine (SSVM)

The SSVM is an extension of the original Support Vector Machine tailored for time-to-event data in survival analysis. Much like the SVM,
the SSVM offers computational efficiency and is effective in capturing underlying non-linear relationships by utilizing kernel functions. Kernel functions such as the Radial Basis Function (RBF) map the data into higher-dimensional hyperplanes, allowing the identification of a survival decision boundary.

While the SSVM shares advantages with the SVM, its focus on survival analysis leads to specific applications:

1. Ranking samples with short survival times
2. Regression by predicting time-to-event values for samples

For the purpose of this study, we opt for using the SSVM in a regression setup, enabling a direct comparison of its performance with that of the WTTE-RNN. In some experiments involving hand-crafted features, we utilize a static survival model. Given the nature of our data-time-series histograms- and the specific experiment requirements, the SSVM is a suitable choice. Its computation efficiency further contributes to its selection.

4.3 Evaluation Methods

To assess the performance of our prognostic models, we employ the following evaluation methods:

4.3.1 Mean Absolute error (MAE) and Time Segment Evaluation

The prediction of time-to-event (TTE) involves solving a survival regression problem, and the mean absolute error (MAE) is a suitable metric for quantifying its performance. However, to comprehensively analyze the networks’ performance across different time segments, we conduct a time segment-based evaluation on the outcomes. We divide the entire timeline of the vehicles into uniform segments of 15 time units and assess the results accordingly. This approach also accommodates the irregular sampling of our time-series data. During the evaluation, we calculate the mean absolute error only for the uncensored cases. This decision is driven by the lack of reliable information about the actual failure times for censored cases, rendering the use of the end-of-study timeline inappropriate.

An illustration of the time-segment-based evaluation is provided in Figure 15.
4.3.2 Concordance Index (c-index) and c-index Decomposition

The concordance index (c-index) stands as the prevalent metric utilized in survival analysis. It serves as a measure of rank correlation between the predicted risk scores and the observed time-to-event points. To illustrate, consider two samples, A and B. If only A experiences an event during the study while B is censored, the pair of points (A, B) is deemed concordant if the estimated risk of failure for A surpasses that of B, and simultaneously, the time to event of B exceeds that of A.

We employ the c-index to assess the performance of the time-to-event network in terms of censored features. Although, the MAE provides a reasonably effective evaluation for uncensored cases, extending our understanding to real-life scenarios—where most cases are censored—requires an evaluation of how well the network ranks censored cases against uncensored ones.

In addition to the c-index, we intend to conduct an evaluation using c-index decomposition [1]. This metric dissects the c-index by separately ranking censored and uncensored cases. This approach offers a detailed analysis of the performance of survival models in each case.
5.1 BASELINE EXPERIMENTS

To lay the foundation for our future experiments, it is essential to gain a comprehensive understanding of how specific features perform within the context of RNN modeling. In this regard, we conduct a comparative analysis of various sets of hand-crafted features and subject them to evaluation.

5.1.1 Experiment Design

Hence, we conduct the following baseline experiments using a scaled-down version of the WTTE-RNN network. The simplified WTTE-RNN architecture is outlined in Table 2. This scaled-down version of the network shares similarities with the main experiments, but with the distinction that it exclusively employs the last remaining useful life (RUL) as the target. Thus, this network operates as a sequence-to-one model for the baseline comparison. The experiments we undertake are as follows:

1. Average Model - In this experiment, the model emulates the behavior of constantly outputting the average last remaining useful life from the training set (only for uncensored cases). The performance error is then measured by comparing against the actual remaining useful life. This serves as a baseline for assessing the relative performance of other models.

2. Age & Usage Features - This baseline experiment aims to directly evaluate the impact of truck age and usage on the remaining useful life. The experiment employs only two features: truck age and truck usage between snapshots. Both features are expressed in terms of operating time units and have been scaled to fit within the range of 0 to 1. Since this experiment is intended for comparison with other models, only the last snapshots are utilized. Therefore, a static model of Survival SVM in regression configuration is applied to predict the remaining useful life. Unlike the RNN experiments, the static model cannot incorporate information from other time snapshots.

3. Raw Histogram Features - In this experiment, the original raw histogram features are used in their normalized form. The feature space encompasses 152 dimensions, which corresponds to the total number of bin features for each operational variable.

4. Benchmark Features - This experiment employs the hand-crafted features as described in [3]. The features consist solely of 1-
D histograms, including the 2-D histograms that were summarized along the axes and used as individual 1-D histograms.

5. **Benchmark Features with 2-D Histogram Point Features** - Building on the benchmark features, this experiment applies the similarity measure in a similar manner to the benchmark features, but on the 2-D histograms without prior summarization. The newly created point features from this process are used alongside the existing features from the benchmark study, and their combined performance is evaluated.

6. **Benchmark Features using Hellinger Distance** - In this experiment, the benchmark features, which were originally transformed using intersection distance, are now modified to employ Hellinger distance instead.

7. **Benchmark Features with Modified Health Feature** - The benchmark features were converted into two point features, one of which signifies the similarity to each vehicle’s first snapshot as an indicator of deviation from its healthy state. This experiment explores the effects of using a centralized healthy histogram, which is calculated as the average of all the first snapshots:

\[
\text{Dist}(A_{\text{central}}, A^t) = \sum_{i=0}^{n} \min(A_{\text{central}}, A^t_i)
\]

where; \(n\):= number of snapshots in a sample, \(t\):= time index in a sample

\[
A_{\text{central}} = \frac{\sum_{i=0}^{N} A^t_i}{N}
\]

where; \(N\) represents the number of samples.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Input Config</th>
<th>Train MAE</th>
<th>Val MAE</th>
<th>Test MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>Avg. model</td>
<td>11.15 ± 1.21</td>
<td>11.40 ± 1.25</td>
<td>-</td>
</tr>
<tr>
<td>Static</td>
<td>Age, Usage</td>
<td>12.28 ± 1.5</td>
<td>12.29 ± 1.5</td>
<td>-</td>
</tr>
<tr>
<td>RNN</td>
<td>Raw Histogram</td>
<td>21.48 ± 2.45</td>
<td>21.21 ± 2.30</td>
<td>22.29 ± 2.35</td>
</tr>
<tr>
<td>RNN</td>
<td>Benchmark</td>
<td>9.51 ± 0.02</td>
<td>9.86 ± 1.21</td>
<td>9.55 ± 1.2</td>
</tr>
<tr>
<td>RNN</td>
<td>Benchmark + 2-D Hist Point</td>
<td>9.35 ± 1.25</td>
<td>9.92 ± 2.1</td>
<td>9.99 ± 2.5</td>
</tr>
<tr>
<td>RNN</td>
<td>Benchmark + Hellinger Dist</td>
<td>11.82 ± 1.34</td>
<td>11.48 ± 1.25</td>
<td>11.04 ± 1.15</td>
</tr>
<tr>
<td>RNN</td>
<td>Benchmark + Modified Healthy Sim.</td>
<td>9.28 ± 1.12</td>
<td>10.28 ± 1.8</td>
<td>9.85 ± 1.32</td>
</tr>
</tbody>
</table>

Table 1: Baseline experiments on a simple WTTE-RNN Network to understand feature importance; Benchmark features refer to the handcrafted features used in [3].

The results of the baseline experiments are presented in Table 1. Upon reviewing the outcomes, it becomes evident that the network struggles to effectively learn from the higher-dimensional raw histogram data. This observation underscores the necessity for feature learning in addressing this problem.
|
|---|---|---|---|
| Input Shape | Layer type | Hidden Layer | Output Shape |
| (None, None, 10) | Mask | - | (None, None, 10) |
| (None, None, 10) | LSTM | 24 | (None, 24) |
| (None, 24) | Dense | 30 | (None, 30) |
| (None, 30) | Dense | 20 | (None, 20) |
| (None, 20) | Dense | 10 | (None, 10) |
| (None, 10) | Dense | 2 | (None, 2) |
| (None, 2) | Lambda | - | (None, 2) |

Table 2: Simple WTTE-RNN Architecture

Another noteworthy finding pertains to the replacement of intersection distance with Hellinger distance in the creation of point features. Curiously, the error rate slightly increases when utilizing Hellinger distance. This outcome is intriguing considering the near-linear relationship between intersection distance and Hellinger distance.

Furthermore, an experiment involving the alteration of the definition of a healthy histogram has been conducted. The comparison involves using a centrally averaged healthy histogram against the conventional approach of using individual samples’ first snapshot histograms as indicators of a healthy state. While this modification does not notably reduce the error rate, the results remain comparable to those achieved with the benchmark model. This insight suggests that, within this context, the concept of an average healthy histogram aligns with the histograms derived from individual samples’ first snapshots.

5.2 Evaluating Representations on Time-to-Event Prediction Models

Having conducted preliminary baseline experiments, we are now prepared to outline our main experimental approach. For these primary experiments, we will employ the WTTE-RNN in its full sequence-to-sequence mode. This configuration allows us to leverage the network’s capability to estimate the RUL across the entire trajectory of each sample. In this endeavor, we aim to juxtapose the performance of the benchmark features against that of the features learned through the Autoencoder methodology.

5.2.1 Autoencoder Feature Learning

In our pursuit of feature learning, our initial step involves identifying a suitably lower-dimensional space that can efficiently reconstruct the input while incurring minimal loss during the process. To this end, we delve into both the AE and the CNN-AE methodologies. Our objective is to contrast their performances during the subsequent fine-tuning phase, thereby gaining insights into which autoencoder configuration is optimal for the estimation of RUL.
5.2.1.1 Experiment Design

The architecture of the CNN encoder-decoder, employed for the purpose of learning lower-dimensional representations, is presented in Table 3. The activation function ‘ReLU’ was applied to all layers, except for the output layer, which was set to ‘sigmoid’. Notably, the use of max pooling appeared to effectively summarize the convolution outputs, surpassing the strategy of employing a stride of 2. This contrasts with the assertion made by the authors of [12], who suggested that a higher stride performs better for down-sampling. It’s worth noting that their work dealt with histograms of nearly twice the size (24x24), whereas this study focuses on histograms of dimensions (12x12).

<table>
<thead>
<tr>
<th>Input Shape</th>
<th>Layer type</th>
<th>Hidden layer</th>
<th>Output Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>(None , 12, 12, 1)</td>
<td>Conv2D(3x3/1)(Input)</td>
<td>24</td>
<td>(None , 12, 12, 24)</td>
</tr>
<tr>
<td>(None , 12, 12, 24)</td>
<td>MaxPool2D</td>
<td>-</td>
<td>(None , 6, 6, 24)</td>
</tr>
<tr>
<td>(None , 6, 6, 24)</td>
<td>Flatten</td>
<td>-</td>
<td>(None , 864)</td>
</tr>
<tr>
<td>(None , 864)</td>
<td>Dense(Bottleneck)</td>
<td>16</td>
<td>(None , 16)</td>
</tr>
<tr>
<td>(None , 16)</td>
<td>Reshape</td>
<td>-</td>
<td>(None , 6, 6, 24)</td>
</tr>
<tr>
<td>(None , 6, 6, 24)</td>
<td>Conv2DTranspose(3x3/1)</td>
<td>24</td>
<td>(None , 12, 12, 24)</td>
</tr>
<tr>
<td>(None , 12, 12, 24)</td>
<td>Conv2D(3x3/1)(Output)</td>
<td>1</td>
<td>(None , 12, 12, 1)</td>
</tr>
</tbody>
</table>

Table 3: CNN-AE Encode-Decoder Architecture

The architecture of the Dense encoder-decoder, employed for the purpose of learning lower-dimensional representations, is depicted in Table 4. All layers utilized the ‘ReLU’ activation function, with the exception of the output layer, which employed the ‘sigmoid’ activation function. While a ‘linear’ activation layer was tested at the bottleneck, it was found that ‘ReLU’ performed better in this context.

<table>
<thead>
<tr>
<th>Input Shape</th>
<th>Layer type</th>
<th>Hidden layer</th>
<th>Output Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>(None, 132)</td>
<td>Dense(Input)</td>
<td>64</td>
<td>(None, 64)</td>
</tr>
<tr>
<td>(None, 64)</td>
<td>Dense</td>
<td>32</td>
<td>(None, 32)</td>
</tr>
<tr>
<td>(None, 32)</td>
<td>Dense(Bottleneck)</td>
<td>16</td>
<td>(None, 16)</td>
</tr>
<tr>
<td>(None, 16)</td>
<td>Dense</td>
<td>32</td>
<td>(None, 32)</td>
</tr>
<tr>
<td>(None, 32)</td>
<td>Dense</td>
<td>64</td>
<td>(None, 64)</td>
</tr>
<tr>
<td>(None, 64)</td>
<td>Dense(output)</td>
<td>132</td>
<td>(None, 132)</td>
</tr>
</tbody>
</table>

Table 4: AE Encode-Decoder Architecture

The chosen combination of layers in each of the networks proved to be most effective in this scenario and was validated by comparing their reconstruction errors on the test set. No Dropout or Batch Normalisation layers were implemented in both networks, as they did not contribute to improving the validation score or enhancing model generalization in this case. The networks were trained for 200 epochs, utilizing the Adam optimizer with a learning rate of 0.0005. Several combinations of other optimizers and learning rates were tested, but this particular set of hyperparameters yielded the best results in this context.
The fine-tuning architecture capitalizes on the acquired representations from the preceding step. The encoder segment of the autoencoder was utilized by freezing its layers and employing them in inference mode. For the fine-tuning process, we replicated the shape of the dense layer, as we sought to study the impact of the fine-tuning process. In this approach, only the learned representations from the CNN-AE were employed for fine-tuning. The target output for fine-tuning is the intersection distance between the input and the central healthy histogram, making it a regression problem. Given that the output is confined within the range of 0 and 1, the output layer employs a ‘sigmoid’ activation function to constrain the output within this interval.

In Table 5, the fine-tuning architecture is outlined. Based on the autoencoder experiments for both CNN-AE and AE, a 16-dimensional bottleneck layer demonstrated the best performance. Consequently, a new trainable dense layer was appended with 16 dimensions, utilizing the ‘ReLU’ activation function, while the output layer was configured with a ‘sigmoid’ activation function to restrict the output between 0 and 1. The encoder network was maintained in a non-trainable state (inference mode).

<table>
<thead>
<tr>
<th>Input Shape</th>
<th>Layer type</th>
<th>Hidden layer</th>
<th>Output Shape</th>
<th>Trainable</th>
</tr>
</thead>
<tbody>
<tr>
<td>(None, 12, 12, 1) or (None, 132)</td>
<td>Encoder Input</td>
<td>-</td>
<td>-</td>
<td>False</td>
</tr>
<tr>
<td>(None, 16)</td>
<td>Dense(Bottleneck)</td>
<td>16</td>
<td>(None, 16)</td>
<td>False</td>
</tr>
<tr>
<td>(None, 16)</td>
<td>Dense</td>
<td>16</td>
<td>(None, 16)</td>
<td>True</td>
</tr>
<tr>
<td>(None, 16)</td>
<td>Dense</td>
<td>1</td>
<td>(None, 1)</td>
<td>True</td>
</tr>
</tbody>
</table>

Table 5: Fine Tuning Architecture for CNN-AE

5.2.1.2 Autoencoder-based Lower Dimensional Representations

In this section, an experiment is conducted on the CNN-AE and AE networks described in Chapter 4. These networks are trained to reconstruct 2-dimensional inputs and 1-dimensional outputs, respectively. The experiments involve varying the bottleneck dimensions from 4 to 20. The results are then compared and presented in Figures 16 and 17. The reconstruction loss exhibits a decreasing trend as the dimensions are increased, starting from a reasonably low value. However, it eventually saturates, indicating that further increases in dimensions would either insignificantly reduce the loss or, as observed in these cases, increase the error. Notably, the 16-dimensional bottleneck dimension demonstrates the most effective ability to reproduce the input with the lowest reconstruction error. As a result, we proceed with this bottleneck dimension for the subsequent step of fine-tuning the network.
Analysis lower Dimensional Representations

An analysis of the reconstruction error, measured as MAE on the test set, was conducted for various bottleneck dimensions. A significant drop in error is observed when comparing the 4-dimensional and 8-dimensional bottleneck dimensions. Subsequently, for each increase of 4 dimensions, a consistent linear decrease in error is observed. This trend highlights the trade-off between adding more dimensions to maximize the potential of the bottleneck layer in capturing input data and learning general representations. High error values would not be ideal for subsequent steps in the feature learning process and ultimately for RUL estimation. Additionally, it is noteworthy that the CNN-AE outperforms the AE in terms of reconstruction capabilities. A visual representation of these comparisons is presented in Figure 18.
5.2.1.4 Fine-tuning Bottleneck Dimensions

After acquiring lower-dimensional representations of the input data through bottleneck features, it is essential to guide these general representations towards effectively discerning the similarity of the input 2-D histograms from their original healthy state. As no explicit feature indicates the histograms contributing to the maximum deviation from their healthy state, it becomes necessary to introduce a feature representing the intersection distance between the input histogram and the central healthy histogram. Fine-tuning was conducted on both the CNN-AE and the AE, with the results of the two configurations compared below.

![Figure 19](image1.png) Comparison of training and validation losses of CNN-AE and the AE model after fine-tuning

![Figure 20](image2.png) Comparison of the error on a test set of CNN-AE and AE model after fine-tuning

The tuned CNN-AE model will serve as the learned features for modeling time-to-event prediction. These learned features will be integrated into the WTTE-RNN in two ways: 1) as standalone features; and 2) in tandem with hand-crafted benchmark features. Comparative assessments of their performances will be made against the benchmark features.

5.2.2 Time-series Modelling

As the next step towards time-to-event prediction, we will proceed with utilizing the complete WTTE-RNN network, which predicts the
RUL for all snapshots of a given sample. Subsequent sections will delve into the architecture details and present the outcomes of the prediction models.

5.2.2.1 Experiment Design

1. WTTE-RNN Model

The network architecture employed for the WTTE-RNN network is provided in Table 6. This architecture is derived from the works of [3] and [10], where their network designs were utilized without alteration. The rationale behind using their architectures was to isolate the effects of the newly introduced learned representations.

The architecture begins with a masking layer that facilitates the network’s recognition of time-sequence inputs with certain values masked using an arbitrary value beyond the range of the existing feature values. This masking enables the network to disregard these values and label them as masked. Subsequently, an LSTM layer is employed, as recommended in [9]. Although the authors propose any RNN layer can be used, the LSTM layer yielded the best results for this particular scenario. Following the LSTM layer, a series of dense layers are incorporated. All layers in the network are activated using the ‘tanh’ function. Finally, the last dense layer should possess two dimensions, as the intention is to apply the Weibull loss function, thereby obtaining ‘alpha’ and ‘beta’ values. These values enable the generation of a Weibull distribution for the projected remaining useful life. The median of this distribution is then extracted.

<table>
<thead>
<tr>
<th>Input Shape</th>
<th>Layer type</th>
<th>Hidden layer</th>
<th>Output Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>(None, None, 43)</td>
<td>Mask</td>
<td>20</td>
<td>(None, None, 43)</td>
</tr>
<tr>
<td>(None, None, 43)</td>
<td>LSTM</td>
<td>20</td>
<td>(None, None, 20)</td>
</tr>
<tr>
<td>(None, None, 30)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 30)</td>
</tr>
<tr>
<td>(None, None, 20)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 20)</td>
</tr>
<tr>
<td>(None, None, 10)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 10)</td>
</tr>
<tr>
<td>(None, None, 2)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 2)</td>
</tr>
<tr>
<td>(None, None, 2)</td>
<td>Lambda</td>
<td>-</td>
<td>(None, None, 2)</td>
</tr>
</tbody>
</table>

Table 6: WTTE-RNN Architecture

2. Direct Regression of Survival Time with RNN

The performance of WTTE-RNN was assessed by comparing it to that of a RNN network (direct regression of survival time). In order to use the plain RNN network in the context of survival analysis, we use a special loss function as shown below;

\[
\ell (x) = m \cdot \begin{cases} 
|y - \hat{y}|, & \text{if } e_x = 1 \\
\max(0, y - \hat{y}), & \text{if } e_x = 0 
\end{cases}
\]  

(3)
where:

- x: Index of the subject vehicle in data
- m: the masking weight (only use the non-masked values for loss calculation)
- \( e_x \): censor value (1 or 0)
- y: True TTE
- \( \hat{y} \): Predicted TTE

This loss function first and foremost factors in the censored and non-censored cases differently. Secondly, it takes into consideration the masking of the values, which is specific to our case working with non-fixed time sequences.

As can be seen in table 7, the architecture used for the RNN is not very different from that of the WTTE-RNN architecture (table 6) except for the Lambda layer in the WTTE-RNN network. This will help us compare the behavior of the two networks objectively. Other training hyper-parameters also were kept constant.

<table>
<thead>
<tr>
<th>Input Shape</th>
<th>Layer type</th>
<th>Hidden layer</th>
<th>Output Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>(None, None, 43)</td>
<td>Mask</td>
<td>20</td>
<td>(None, None, 43)</td>
</tr>
<tr>
<td>(None, None, 43)</td>
<td>LSTM</td>
<td>20</td>
<td>(None, None, 20)</td>
</tr>
<tr>
<td>(None, None, 30)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 30)</td>
</tr>
<tr>
<td>(None, None, 20)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 20)</td>
</tr>
<tr>
<td>(None, None, 10)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 10)</td>
</tr>
<tr>
<td>(None, None, 2)</td>
<td>Dense</td>
<td>30</td>
<td>(None, None, 2)</td>
</tr>
</tbody>
</table>

Table 7: RNN Architecture

5.2.2.2 Time-To-Event Prediction

1. Performance of fine-tuned vs. non-fine-tuned features on WTTE-RNN

Once the lower-dimensional representations have been fine-tuned, the estimation of the turbocharger’s remaining useful life becomes feasible. In the initial stage, we will model both the fine-tuned features and the un-tuned features (encoded features following AE reconstruction) to assess the effectiveness of our fine-tuning approach.

The outcomes of this experiment are depicted in Figure ???. The results clearly indicate that the fine-tuned encoded features outperform the un-tuned encoded features.

2. Performance of Benchmark vs. fine-tuned vs. both combined features on WTTE-RNN

For the subsequent experiment, we will assess the performance of the learned features from the tuned network in comparison to the benchmark features. The outcomes of the WTTE-RNN modeling are illustrated in Figure 22. The WTTE-RNN results
Figure 21: WTTE Results comparing the tuned and un-tuned CNN-AE features with 95% prediction interval on test set.

indicate that the combination of both feature sets, as well as the CNN features alone, do not exhibit improvements beyond the benchmark features’ performance across the segmented timeline.

A noteworthy observation is that the CNN features alone match the results of the benchmark features, but they fail to surpass them. This is intriguing given that the CNN features encode larger non-summarized 2-D histograms in contrast to the benchmark model’s utilization of summarized 1-D histograms. Further investigation into how these 2-D histograms encapsulate information would be crucial to comprehend this behavior.

A summary of the results can be found in Table 8. Additionally, the results portray the 95% prediction intervals, derived from running 10 ensembles of the same network using the three different feature types. The prediction interval serves to assess the robustness of the WTTE-RNN network. Interestingly, the prediction interval scores indicate a certain degree of network instability, possibly attributed to the variable sequence lengths and the network’s handling of such instances. Further exploration of this phenomenon is left for future work.

The evaluation of the WTTE network results is carried out using the MAE. However, this evaluation can only be conducted on the uncensored cases, as the error is well-defined solely for such instances. Consequently, to gauge the model’s performance on the censored cases, we employed the survival ranking score of the c-index. To achieve this, we utilized the c-index decomposition method introduced in Chapter 4. This approach allows us to evaluate the c-index for censored, uncensored, and overall scenarios. The outcomes are presented in Figure 23.

Observing the results, we note a remarkable similarity in scores across all three feature types. Benchmark features exhibit a slightly superior performance compared to the others. An interesting observation is that the model generally excels in ranking the
Figure 22: WTTE Results With 95% prediction interval on test set

Table 8: Summary of WTTE-RNN results with 95% prediction interval

<table>
<thead>
<tr>
<th>Time Segments</th>
<th>Only Benchmark Features</th>
<th>CNN features + Benchmark features</th>
<th>Only CNN features</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>15.58 ± 2.79</td>
<td>17.77 ± 3.10</td>
<td>12.24 ± 2.00</td>
</tr>
<tr>
<td>30</td>
<td>12.46 ± 2.65</td>
<td>14.00 ± 2.95</td>
<td>12.20 ± 2.15</td>
</tr>
<tr>
<td>45</td>
<td>12.68 ± 2.30</td>
<td>13.76 ± 4.52</td>
<td>13.39 ± 3.47</td>
</tr>
<tr>
<td>60</td>
<td>13.81 ± 6.73</td>
<td>14.08 ± 4.63</td>
<td>14.65 ± 5.44</td>
</tr>
<tr>
<td>75</td>
<td>17.29 ± 3.79</td>
<td>16.99 ± 5.21</td>
<td>17.77 ± 4.33</td>
</tr>
<tr>
<td>90</td>
<td>24.35 ± 1.77</td>
<td>22.84 ± 4.10</td>
<td>23.64 ± 2.30</td>
</tr>
<tr>
<td>105</td>
<td>35.55 ± 0.90</td>
<td>34.04 ± 2.66</td>
<td>34.68 ± 1.41</td>
</tr>
<tr>
<td>120</td>
<td>48.24 ± 1.35</td>
<td>46.82 ± 2.14</td>
<td>47.60 ± 1.83</td>
</tr>
<tr>
<td>135</td>
<td>61.37 ± 1.38</td>
<td>60.28 ± 2.24</td>
<td>60.99 ± 2.12</td>
</tr>
<tr>
<td>150</td>
<td>73.58 ± 1.62</td>
<td>73.26 ± 1.83</td>
<td>73.75 ± 1.99</td>
</tr>
</tbody>
</table>

Table 9: Summary of results

<table>
<thead>
<tr>
<th>Features</th>
<th>c-index</th>
<th>c-index-uncensored</th>
<th>c-index-censored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Benchmark Features</td>
<td>0.775 ± 0.0565</td>
<td>0.796 ± 0.061</td>
<td>0.752 ± 0.055</td>
</tr>
<tr>
<td>Combined CNN Features and Benchmark Features</td>
<td>0.741 ± 0.086</td>
<td>0.788 ± 0.090</td>
<td>0.69±0.090</td>
</tr>
<tr>
<td>Only CNN Features</td>
<td>0.747±0.095</td>
<td>0.793±0.085</td>
<td>0.69±0.110</td>
</tr>
</tbody>
</table>

3. Understanding the WTTE-RNN

When analyzing the WTTE performances in figure 21 and 22, we can notice strange behavior in the MAE values of the non-uncensored features as opposed to the censored ones. This disparity suggests that the WTTE-RNN model exhibits a bias toward uncensored cases. This phenomenon aligns with the claim made by the authors of the network, who noted that incorporating censored samples during training may introduce a degree of instability to the network.

A summarized presentation of the c-index results is provided in Table 10.
Figure 23: Survival score using c-index with 95% prediction interval on test set

fine tuned model (figure 21: red bars), where the MAE values go down and then go back up as the model is predicting values closer to end-of-life. In other models (figure 21: yellow), this phenomenon is also visible although with a smaller magnitude. We need to break this down to understand this further.

Figure 24: Error in prediction RUL, when trained on different features: 1) Only CNN fine-tuned features, 2) Benchmark Features + CNN fine-tuned features, 3) Benchmark features

In addition to the MAE of the RUL, results presented in 22, we also need to look at the error values presented in figure 24. This illustrates the errors (average true - predicted RUL) for three different feature sets (benchmark features + cnn fine-tuned features, cnn fine-tuned features and benchmark features). In the figure, it is noticeable that the errors are initially negative but trend towards becoming positive as the trucks approach the point of failure. This trend is also closely observed in Figure 25, which displays the true RUL time steps alongside the predicted RUL time steps averaged for all vehicles in the dataset.

Furthermore, in figure 25, all the predicted RULs appear to be diverging slightly from the true RUL, indicating that the WTTERNN tends to under-predicts failures. This behavior aligns with the results presented by [3]. Notably, the predicted RULs based on the benchmark features and the tuned Autoencoder (AE) RUL shows a tendency to converge towards the true RUL, whereas the non-fine-tuned AE RUL appears to diverge from the true RUL. Consequently, a significant deviation in MAE is
observed in the non-fine-tuned AE RUL beyond the 60 segments leading up to the time of failure.

4. WTTE-RNN vs Direct Survival Time Regression with RNN

In figure 26, the $\text{MAE}_{\text{WTTE}}$ consistently remains significantly lower than $\text{MAE}_{\text{RNN}}$ across all time segments. Notably, the $\text{MAE}_{\text{WTTE}}$ indicates an error lower than 20 time units from 90 time units onward to failure whereas the $\text{MAE}_{\text{RNN}}$ reaches a similar error level only from 30 time units onward. This displays the superiority of the WTTE-RNN over the RNN model in terms of predictability and preparation for failure. Hence, the WTTE-RNN having the alpha and beta parameters responsible for predicting the Weibull distribution would better suit the survival modelling problem than just a typical RNN in this case.
**Figure 27**: EMD distance visualization of the 2-D Histogram of Engine load on the y-axis versus the Engine RPM on the x-axis of trucks that experienced a failure. Color coding: Red channel indicates a change in the starting distribution; the green channel indicates a change in the ending distribution; yellow indicates "unchanged". The arrows indicate the flow determined using EMD, where the thickness of the arrow is proportional to the amount of mass flow.

5.2.2.3 *Analysis*

The 2-D histograms for the Engine and Load variables consist of 132 dimensions, which is essential to capture a more comprehensive understanding of the turbocharger’s behavioral characteristics, given their dependency on its changes. To differentiate between snapshots, we utilize the Earth Mover’s Distance (EMD) distance metric to visualize the distribution shifts between the first and penultimate snapshots of the subject. The EMD calculates dissimilarities between two distributions.

In Figure 27 and 28, we present the distributions of the first snapshot, the penultimate snapshot, and the corresponding EMD distance between them. Figure 27 pertains to cases where a failure event occurred during the subject’s lifespan, while Figure 28 relates to subjects where no failure was recorded. In the EMD visualization, colors indicate distribution changes, with intensity reflecting the magnitude of change. Specifically, red and green denote the first and penultimate snapshot distributions, respectively. Arrows represent the flow from the distribution of the first snapshot to that of the penultimate one.

Upon analyzing these visualizations, it becomes evident that the variation in the 2-D histogram data primarily occurs along each of the axes rather than obliquely. This observation helps explain why the performance of the CNN features does not surpass that of the benchmark features, which summarized the 2-D histograms along the axes.

Furthermore, we are interested in expanding this study to explore the potential of flow matrices derived from the EMD computations in replacing the need for an RNN in survival modeling. Instead, we aim to employ a more general survival model, such as survival SVM, to estimate the remaining useful life. However, it’s important to note that in this approach, there would be no temporal encoding between snapshots of a sample.
5.2.3 EMD flow with Survival SVM

The EMD flow captures changes between distributions, essentially representing the flow in time. To investigate its potential in simulating temporal changes and possibly replacing the need for an RNN network in this time-series problem, we conduct a comparative analysis. As part of this analysis, we aim to apply a static survival regression model on the flow data.

To establish a benchmark for this exercise, we leverage the results obtained from the RNN. Specifically, we consider the RNN error for uncensored data and its c-index scores, which indicate how well it ranks censored cases compared to uncensored data. These benchmark results will serve as a reference point for our analysis.

![Figure 29: Time-Segment based mean absolute error comparing the two feature types 1)EMD flow features; 2) Benchmark features](image)

The results presented in Table 10 and the plots in Figure 29 clearly demonstrate that the model’s performance is far from achieving re-
Table 10: Modelling of Survival SVM on 1) EMD flow features; 2) Benchmark features. Modelling is static, meaning each snapshot’s **RUL** is estimated

<table>
<thead>
<tr>
<th>Model</th>
<th>Input</th>
<th>Train c-index</th>
<th>Test c-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Survival- SVM</td>
<td>EMD Flow features</td>
<td>0.53</td>
<td>0.53</td>
</tr>
<tr>
<td>Survival- SVM</td>
<td>Benchmark features</td>
<td>0.53</td>
<td>0.63</td>
</tr>
</tbody>
</table>

results comparable to those of the WTTE-RNN. The model consistently predicts a constant value of approximately 134 units, which corresponds to the median **RUL** present in the dataset. This trend also explains the increasing **MAE** observed across the earliest time segments to the lowest ones.

While the flow data does visually capture variations in distributions and encodes the flow between different distributions, it doesn’t inherently encode temporal information. Similarly, the benchmark features also represent histogram variations over time but in a much lower-dimensional point feature space. Despite these differences, neither the high-dimensional flow matrix nor the low-dimensional benchmark features perform well with the static survival model. This further underscores the importance of utilizing an **RNN** for accurately estimating **RUL** in this context.
DISCUSSION

6.1 DISCUSSION AND INTERPRETATION OF RESULTS

This chapter aims to provide a comprehensive analysis of the results presented in Chapter 5. It elucidates the objectives of this research and how they were achieved.

While the exploration of feature learning for histogram data in survival analysis is not novel, it remains an intricate problem. Earlier efforts, as mentioned before, often involved converting histograms into point features and tracing their dynamics. Notably, the work of [3] transformed histograms into point features for estimating remaining useful life. Building upon this foundation, we extended our investigation to features that were previously summarized, specifically the 2-D histograms. Although authors like [12] delved into learning low-dimensional representations of 2-D histograms, this was predominantly executed within a supervised learning context. However, in our research, the estimation of remaining useful life is framed as a survival regression problem, necessitating an indicator for both the time-to-event and the censoring label to enable the model to formulate a credible survivability estimation. Given the high dimensionality of 2-D histograms, we sought efficient and meaningful representations to unveil deterioration in an unsupervised manner. To this end, we adopted a CNN-based autoencoder approach to uncover lower-dimensional general representations of input 2-D histograms. Subsequently, we fine-tuned this approach to perceive deviations from the central normal histogram. This fine-tuning approach allowed the network to learn problem-specific representations for remaining useful life estimation. Our results highlight that the performance of the autoencoder-based feature learning method is comparable to that of benchmark hand-crafted features. Evaluation segmented over time indicates that while the error is relatively high for initial snapshots, the RNN network improves performance as more snapshots become available, rendering reasonable estimates of remaining useful life. This progression can be applied to predictive maintenance applications, where inference is monitored at each snapshot but only acted upon after a certain number of snapshots with the lowest error are accumulated.

The utilization of the EMD flow estimation enabled us to comprehend the variations between a vehicle’s initial state and its end state. The flow estimation provides a close approximation of how these variations transpired. Investigation of the flow reveals that most variations between histograms occur along axes that are either latitudinal or longitudinal in nature, with minimal variations in oblique directions. This insight indicates that the matrix can be summarized into subsequent 1-D histograms in this scenario, although this ap-
proach might not hold true for all cases. Furthermore, a preliminary exploration aimed to ascertain whether the flow matrices could negate the necessity for RNNs, possibly reflecting variation displacement through flow information. However, the results only slightly exceeded random behavior, demonstrating that static flow information lacks temporal context or insight into variable changes over time.

6.2 FUTURE WORK

This research, due to its limited scope, could not delve into various other ideas, some of which are highlighted in this section.

The study’s focus was on achieving a generalized approach in modeling different populations. The dataset comprised various vehicle types, yet these distinctions were not considered in assessing the model’s overall performance. This approach aimed to ascertain whether distinct populations exhibit similar failure characteristics concerning turbocharger modeling. A potential avenue for future research involves investigating methods to homogenize the populations or incorporating their heterogeneity into the modeling process.

The division of the dataset into training, validation, and testing subsets was stratified based on censor labels, ensuring label distribution consistency across the subsets. However, a more beneficial approach could involve stratifying the split based on the lifetime operational period in addition to censor distribution. This enhancement or future work could lead to improved model performance.

This research introduces a novel fine-tuning method that models snapshot similarities. From empirical findings and experimentation, it is evident that the intersection distance similarity measure yields optimal results due to its inherent ability to measure entropy accurately. Thus, an avenue for future research could be the exploration of alternative similarity measures tailored for this purpose.


