Interpretable Hidden Markov Model-Based Deep Reinforcement Learning Hierarchical Framework for Predictive Maintenance of Turbofan Engines

Ammar N. Abbas [1] Georgios Chasparis [1] John D. Kelleher [2]

Abstract

An open research question in deep reinforcement learning is how to focus the policy learning of key decisions within a sparse domain. This paper emphasizes combining the advantages of input-output hidden Markov models and reinforcement learning towards interpretable maintenance decisions. We propose a novel hierarchical-modeling methodology that, at a high level, detects and interprets the root cause of a failure as well as the health degradation of the turbofan engine, while, at a low level, it provides the optimal replacement policy. It outperforms the baseline performance of deep reinforcement learning methods applied directly to the raw data or when using a hidden Markov model without such a specialized hierarchy. It also provides comparable performance to prior work, however, with the additional benefit of interpretability.

1. Introduction

Machine learning has the potential to improve the performance of equipment maintenance systems by providing accurate predictions regarding the type of equipment that should be replaced as well as the optimal replacement time. Predictive maintenance can be categorized as (i) *Prognosis*: predicting failure and notifying for replacement or repair ahead of time (*Remaining Useful Life* (RUL) is usually used as a prognosis approach, which is the estimation of the remaining life of equipment or a system at any point in time before which it is no longer in a functional state (Sikorska et al., 2011)); (ii) *Diagnosis*: predicting the actual cause of failure in the future through cause-effect analysis, or (iii) *Proactive Maintenance*: anticipate and mitigate the failure modes and conditions before they develop within a certain equipment. (Do et al., 2015). In this paper, the aforementioned questions will be investigated in the context of predictive maintenance of turbofan engines (Saxena & Goebel, 2008; Chao, 2021).

Reinforcement Learning (RL) is a natural approach to solving time-series-based stochastic decision problems, such as predictive maintenance (Skordilis & Moghaddass, 2020) and recently, it has shown promising results. RL systems learn by interacting with the environment and can learn in an online setting without having the predefined dataset beforehand; incorporating stochastic events (Sutton & Barto, 2018). However, when the key policy decision learned by an RL agent is relatively rare in a dataset (such as the decision of when to change the equipment before failure while maximizing the use of each piece of equipment) the policy can be dominated by irrelevant phenomena in the data, resulting in inefficient training. At the same time, the derived optimal policy does not provide interpretations or the root cause of the failure, and therefore, it keeps humans out of the loop with limited collaborative intelligence. Furthermore, in real-world industrial environments, RL learns directly from the observed raw sensor data that does not provide information about the unobserved hidden factors responsible for the decision-making of the system such as its health, which limits the agent to behave sub-optimally.

Hidden Markov Model (HMM) (Rabiner & Juang, 1986) can overcome the challenges faced by RL through (i) learning unobserved states and interpretation based on those hidden states, (ii) combining multiple sensor data and leveraging on the covariance between the data, defining the state of the system and its hierarchical distribution, and (iii) dimensionality reduction based on the number of latent states that defines the model and reduces the size and complexity of the raw data (Yoon et al., 2019b). In order to address the need for a more direct and specialized data-based optimization, while maintaining the interpretability of the derived policies, we propose an unsupervised hierarchical

[1] Software Competence Center Hagenberg, Hagenberg, Austria [2] Department of Computer Science, Technological University of Dublin, Dublin, Ireland. Correspondence to: Ammar Abbas <ammar.abbas@scch.at>.

This research is supported by the Collaborative Intelligence for Safety-Critical systems (CISC) project; the CISC project has received funding from the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie grant agreement no. 955901. The work of Kelleher is also partly funded by the ADAPT Centre which is funded under the Science Foundation Ireland (SFI) Research Centres Program (Grant No. 13/RC/2106_P2).

modeling technique that combines a high-level input-output hidden Markov model (IOHMM) with a low-level deep RL methodology for predictive maintenance. *Hierarchical Reinforcement Learning (HRL)* is a solution towards sample efficient RL, which decomposes the long-horizon enormous state space into several short-horizon specialized tasks. At a first step, the IOHMM prefilters large amounts of non-relevant data generated during the normal running of the equipment and detects the state at which failure is imminent. At a second step, the deep RL agent learns a policy on equipment replacement conditioned on these (close to failure) states. Our experimental results indicate that the proposed state-/event-based approach with dynamic data pre-filtering has comparable performance to prior work that trains RL agents directly on the full dataset, hence increasing the training efficiency. Lastly, it allows for more explicit interpretability of the derived policies by learning the latent state spaces. Specifically, the IOHMM learns the hidden state representation of the system (x_t) and the DRL constructs the state-action pair modeling of the environment (s_t, a_t).

Structure: Section 2 provides the literature review on hierarchical and event-driven RL as well as applications of DRL and HMM in interpretable industrial maintenance along with this paper's contributions. Section 3 provides the overview of the use case. Section 4 frames predictive maintenance as an RL problem. Section 5 proposes the novel methodology. Section 6 explains the experimental setup and baseline architectures. Section 7 provides the interpretability aspect of the proposed methodology. Finally, Section 8 compares the proposed architecture with baseline and prior work.

2. Related Work

The major challenge for HRL is the ability to learn the hierarchical structure that requires a priori knowledge or supervision from experts as discussed by (Pateria et al., 2021; Yu, 2018). References (Xu & Fekri, 2021; Lyu et al., 2019) mention data inefficiency and lack of interpretability as the major challenges of RL, hence a new hierarchical RL framework is proposed by the authors that uses symbolic RL. similarly, (Lee & Choi, 2021) solves the same aforementioned problem through HRL by decomposing states into pretrained primitives. The effectiveness of an adaptive event-driven RL strategy and its convergence proof is shown in (Meng et al., 2019). Further, (Parra-Ullauri et al., 2021) proposes an event-driven explainable RL methodology.

Multi-Objective Reinforcement Learning (MORL) is used by (Lepenioti et al., 2020) as a predictive maintenance strategy in the steel industry, where the model learns from both its own experience through environment interaction as well as from the human experience feedback using a policy-shaping approach (Griffith et al., 2013). Double Deep Q-Learning (DDQN) approach (Van Hasselt et al., 2016) for developing a general-purpose predictive maintenance architecture is used by (Ong et al., 2020). They highlight the difference between a traditional regression model and a self-learning agent that provides the recommended actions for each piece of equipment in the system. Authors have used turbofan engines (Saxena & Goebel, 2008) as their case study and have discussed the limitations of prior work that just estimate the RUL of a system, giving no cause-effect relationship between the failure and the components of the equipment. Instead of using raw sensor data, they have dimensionally reduced it into one principal component indicating the Health Index (HI) of the equipment. Using the same case study, (Skordilis & Moghaddass, 2020), provide an optimal maintenance decision and RUL prediction-based alarm system at any predefined cycle before failure, using Bayesian Network-based deep RL. *Bayesian particle filtering*; a Bayesian approach based on sequential Monte Carlo simulation (Chen et al., 2003) is used on top of DRL to map the raw sensor data into latent belief degradation states.

The use of HMM is proposed by (Giantomassi et al., 2011) for predicting RUL of turbofan engines that demonstrated its effectiveness towards the interpretation of the fault point with a sudden decrease in RUL and transition of HMM state. Similarly, (Hofmann & Tashman, 2020) uses HMM for predicting a failure event by using hierarchical mixtures of distributions to predict the overall failure rate and degradation path through individual assets. Input-Output Hidden Markov Model (IOHMM) (Bengio & Frasconi, 1995) is explored by (Klingelschmidt et al., 2017) for failure diagnosis, prognosis, and health state monitoring of a diesel generator in an online setting. Similarly, (Shahin et al., 2019) have shown how IOHMM can be used for prognosis and diagnosis by predicting the hidden (health) state and RUL through a simulated example. The effectiveness of online HMM estimation-based Q learning that converges to a higher mean reward for *Partially Observable Markov Decision Process (POMDP)*, where certain variables are hidden and not directly observable is proved mathematically by (Yoon et al., 2019a).

Literature Gap and Research Contributions: The majority of the research in predictive maintenance using RL is focused on prognosis based on the RUL estimation from multivariate raw sensor readings. However, the interpretability of the faults of the machine (at the equipment level) is missing. Furthermore, realistic environments often have partial observability, where learning from raw data might lead to suboptimal decisions. Additionally, RL encounters learning inefficiency when trained with limited samples and in an online setting (Dulac-Arnold et al., 2021). In this paper, a novel methodology for maintenance decisions and interpretability is proposed that is based on a hierarchical DRL. At a high level, an IOHMM is designed for detecting imminent-to-failure states, while at a low level, a DRL is designed for optimizing the optimal replacement policy. We further present an extensive comparative analysis with prior work that demonstrates the effectiveness of the proposed methodology in terms of both performance and interpretation.

3. Use Case: Turbofan Engines

NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), turbofan engine degradation dataset (Saxena & Goebel, 2008) is widely used in the community of predictive maintenance. The dataset consists of several engine units with multivariate time-series sensor readings and operating conditions discretized based on the flight cycles. Each unit observes some initial degradation at the start of the equipment failure, after which the health of the equipment degrades exponentially until it reaches a final failure state, hence, having a run-to-failure simulation. However, these degrees of wear are unknown. Recently, NASA published an updated version of the dataset (Chao, 2021) that records the real-time flight data and appends the operational history to the degradation modeling. This dataset additionally provides the ground truth values for the health state of the engine based on the component failure modes. These subsets of the datasets will be used in this paper: **FD001** with 1 operating condition and 1 failure mode. **FD002** with 6 operating conditions and 1 failure mode. **FD003** with 1 operating condition and 2 failure modes. **DS01 (version 2)** with ground truth values of degradation.

4. Framing Predictive Maintenance as a Reinforcement Learning Problem

In this section, the decision-making problem associated with optimal predictive maintenance is framed as an RL problem. A general modeling technique is described here followed by its simplified version for our use case.

4.1. Environment Dynamics and Modeling

(Ong et al., 2020) consider three actions as a general methodology for any decision-making maintenance model; hold, repair, and replace. The constraints can be the maintenance budget and the objective function can be the maximum uptime of the equipment. We propose a general framework for modeling such environments with state transitions based on the actions selected under stochasticity (uncertainty of failure, and randomness of replacement by new equipment) at any state, as illustrated in Figure 1. An action to hold transitions to the next state in time, under uncertainty of ending up in a failure state. An action to repair transitions the current state of the equipment back in its life cycle to an arbitrary state as defined by the type of repair or through some standards either from experience, reference manual, or history of data. An action to replace the equipment taken at the current state, transitions it to the initial state of the next equipment (introducing randomness), however, if the equipment reaches its final (failure) state regardless of the action chosen; the equipment must be replaced now. Algorithm A.1 of Appendix A defines the modeling of such stochastic dynamics in terms of RL used within the Open AI gym environment (Brockman et al., 2016). For simplicity and lack of data for repair actions, the action space consists of just two actions (hold or replace).

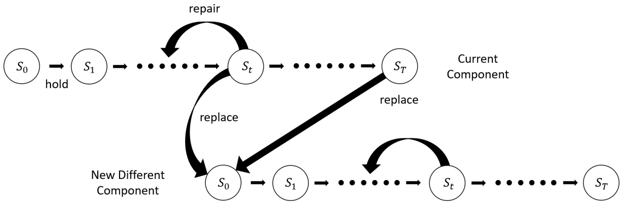


Figure 1. Environment model dynamics

4.2. Reward Formulation

The reward signal is one of the main factors responsible for an effective RL, therefore, it should be chosen carefully. For the maintenance decision in a simplified case, having just the replacement or hold actions, a dynamic reward structure has been formulated as shown in Equation (1) (Skordilis & Moghaddass, 2020). This cost formulation maintains the trade-off between the early replacement (c_r) and replacement after failure ($c_r + c_f$).

$$r_t = \begin{cases} 0, & \mathrm{a}_t = \text{ Hold } & \& \ \mathrm{t} < T_j, \\ -\frac{c_r}{t}, & \mathrm{a}_t = \text{ Replace } & \& \ \mathrm{t} < T_j, \\ -\frac{c_r + c_f}{T_j}, & \mathrm{a}_t = \text{ Hold } & \& \ \mathrm{t} = T_j, \\ -\frac{c_r + c_f}{T_j}, & \mathrm{a}_t = \text{ Replace } & \& \ \mathrm{t} = T_j. \end{cases} \tag{1}$$

4.3. Evaluation Criteria

To evaluate the performance of the RL agent, numerical values were chosen for better comparison.

4.3.1. Cost

The average optimal total return/cost $(\widetilde{Q^*})$ serves as a single numeric value used and compared with the upper and lower bounds of cost possible for such conditions (Skordilis & Moghaddass, 2020).

Ideal Maintenance Cost (IMC) serves as the lower bound and the ideal cost in such maintenance applications.

It is the incurred cost when the replacement action is performed one cycle before the failure, utilizing its maximum potential with the minimum cost as shown in Equation (2).

$$\phi_{IMC} \approx \frac{N \cdot c_r}{N \cdot (\mathbb{E}(T) - 1)} \approx \frac{N \cdot c_r}{\sum_{j=1}^{N} (T_j - 1)} \qquad (2)$$

Corrective Maintenance Cost (CMC) serves as the upper bound and the maximum cost in such maintenance applications. It is the incurred cost when the replacement action is performed after the equipment has failed as shown in Equation (3).

$$\phi_{CMC} \approx \frac{(c_r + c_f)}{\mathbb{E}(T)} \approx \frac{N \cdot (c_r + c_f)}{\sum_{j=1}^{N} T_j} \qquad (3)$$

Average Optimal Cost ($\widetilde{Q^*}$) is the average cost that the agent receives as its performance on the test set as shown in Equation (4).

$$\widetilde{Q^*}(s, a) = \frac{1}{N} \sum \left[r(s, a) + \gamma \max_{a'} Q^* (s', a') \right] \qquad (4)$$

4.3.2. Average useful life before replacement

It quantifies; how many useful life cycles on average are remaining when the replacement action is proposed by the agent. Ideally, it should be 1 according to our defined optimization criteria.

5. Proposed Methodology

The proposed methodology is a hierarchical model integrating an IOHMM and DRL agent. Within this hierarchical model, the purpose of the IOHMM is to identify when the system is approaching a desired (in our case: failure) state. Once the IOHMM has entered this failure state, the task of the DRL agent is to optimize the decision of when to replace the equipment to maximize its total useful life. This IOHMM-DRL model introduces hierarchical levels into the information space and allows for the state- or event-based optimization. This further allows for a more efficient DRL training, since the training dataset is restricted to the imminent-to-failure states. Such agents can be deployed under situation-dependent adaptations as mentioned in (Panzer & Bender, 2021). Beyond the performance considerations of the model, the IOHMM component provides a level of interpretability in terms of identifying failure states (leading towards RUL estimation), root cause of failure, and health degradation stages. Figure 2 illustrates the proposed hierarchical model which we name Specialized Reinforcement Learning Agent (SRLA).

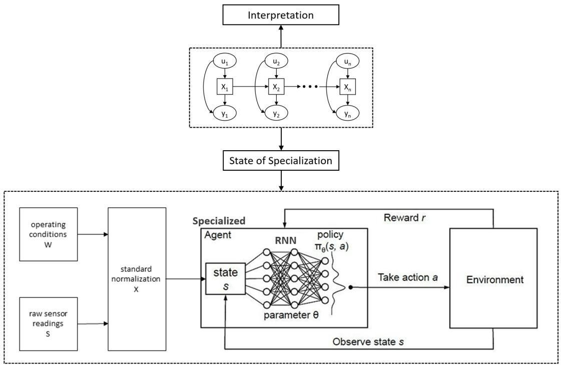


Figure 2. Specialized Reinforcement Learning Agent (SRLA).

The DRL training and optimation process is relatively standard. We use Deep Learning (DL) as a function approximator that generalizes effectively to enormous state-action spaces through the approximation of unvisited states (Bertsekas & Tsitsiklis, 1996) as shown in Equation (5).

$$L_i(\theta_i) = \mathbb{E}_{a \sim \mu} \left[(y_i - Q(s, a; \theta_i))^2 \right];$$
$$y_i := \mathbb{E}_{a' \sim \pi} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid S_t = s, A_t = a \right] \qquad (5)$$

At a high level, an IOHMM is employed that is an extension to a standard HMM model (Bengio & Frasconi, 1995). In a standard HMM model (as described (Rabiner, 1989)) the training optimization objective is to identify the model parameters that best determine the given sequence of observations. To predict the probability of being in a particular hidden state, given the observation sequence Y and trained model parameters λ (initial state, transition, and emission probability matrices), Equation 6 is used. γ is the vector defining the probability of being in each hidden state at a particular time, which will be used as the input to DRL in our baseline extension. Equation 7 predicts the most probable hidden state that in this context leads to the health degradation state given the sequence of sensor observations. However, this does not provide the information of the most probable sequence of states; as it might be possible that the two most probable states at a particular time step may not be the most optimal state sequence. This problem is solved by the Viterbi algorithm (Forney, 1973) as shown in Equation 8, where, in this context, δ is used to predict the health degradation sequence, where the last cycle of each equipment determines the failure state.

$$\begin{aligned} \gamma_t(i) &= P(x_t = S_i \mid Y, \lambda) && (6) \\ x_t &= \operatorname*{argmax}_{1 \le i \le N} [\gamma_t(i)], \quad 1 \le t \le T && (7) \\ \delta_t(i) &= \max_{x_1, \cdots, x_{t-1}} P[x_1 \cdots x_t = i, Y_1 \cdots Y_t \mid \lambda] && (8) \end{aligned}$$

One of the limitations of HMM is that the mathematical model does not take into account any input conditions that affect the state transition and the emission probability distribution of the observations (outputs). In the context of industrial settings, these inputs are the operating conditions that heavily influence the state of the system and control the system's behavior. Therefore, IOHMM is used to have a more general model architecture that can utilize the information of operating conditions, which modifies Equation (6 and 8) to Equation (9 and 10) with λ being conditioned on the input (U) as well.

$$\gamma_t(i) = P\left(x_t = S_i \mid U, Y, \lambda\right) \tag{9}$$

$$\delta_t(i) = \max_{x_1, \cdots, x_{t-1}} P\left[x_1 \cdots x_t = i, Y_1 \cdots Y_t \mid U, \lambda\right] \tag{10}$$

6. Experimental Setup

The baseline systems defined in this paper are distinguished and designed by varying each of these four stages: (i) input, (ii) feature engineering, (iii) RL architecture, and (iv) output. System 1 just uses the raw sensor data. System 2 signifies the importance of using raw sensor data along with operating conditions. It is used to set the cost of failure to be used for the rest of the experiments. System 3 uses HMM as the layer between the raw sensor readings and the DRL. Its significance is to (i) determine the optimal number of HMM states to be used in the experiments, and (ii) demonstrate the effectiveness of HMM and compare it with system 1. Finally, System 4 uses IOHMM in place of HMM to introduce a more general architecture that can take into account varying operating conditions. Implementation of HMM and IOHMM is done through libraries (Lee et al., 2015; Yin & Silva, 2017). The summary of the training parameters are shown in Appendix A.1 of Appendix A.

System 1: Baseline
(i) Raw sensor data as the input, (ii) Standard normalization as the feature extraction module, (iii) DNN as the RL architecture, and (iv) Action policy at the output.

System 2: Baseline + Operating Conditions
(i) Raw sensor data and operating conditions as the input, (ii) Standard normalization as the feature extraction module, (iii) DNN as the RL architecture, and (iv) Action policy at the output.

System 3: Baseline + HMM
(i) Raw sensor data as the input, (ii) MinMax normalization and HMM as the feature extraction module, (iii) DNN as the RL architecture, and (iv) Action policy, RUL estimation, and event-based unsupervised clustering and interpretation at the output; as shown in Figure 3.

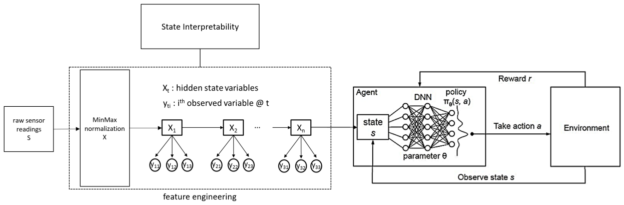


Figure 3. HMM posterior probabilities as the input to DRL.

System 4: Baseline + Operating Conditions + IOHMM
(i) Raw sensor data and operating conditions as the input, (ii) MinMax normalization and IOHMM as the feature engineering module, (iii) RNN as the RL architecture, and (iv) Action policy, RUL estimation, and event-based unsupervised clustering and interpretation at the output; as shown in Figure 4.

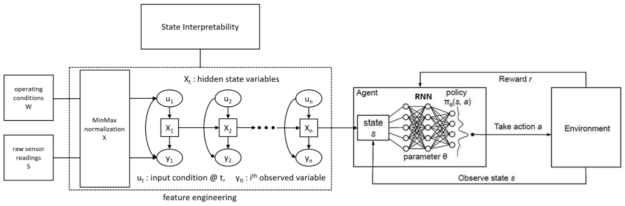


Figure 4. IOHMM posterior probabilities as the input to DRL.

6.1. Model's Hyperparameters Search

This section consists of determining the hyperparameters (i) cost of failure (c_f) and (ii) HMM states. While searching for these parameters, we also observe the effectiveness of HMM and address the question of how well DRL can perform by learning through hidden states? Hence, the effectiveness of the architectures has been evaluated as described in Section 4.3. The dataset used for this part of the experiment is FD001, which is split into an 80:20 (train:test) ratio.

6.1.1. Calculating the cost of failure

The reward function (Equation 1) for RL agent assumes a *cost of failure* (c_f) and *cost of replacement* (c_r) to be specified. However, the NASA C-MAPSS dataset does not specify this parameter. To fix a value for this, we train System 2 using a range of different c_f, while fixing c_r and then comparing and identifying the c_f that minimizes the average of total optimal cost per episode ($\widetilde{Q^*}$). We used System 2 because this system has the baseline architecture while at the same time using the full set of input parameters

Table 1. Comparative evaluation and hyperparameter search.

Fail cost	Avg Q*	IMC	CMC	Average remaining cycles	Failed units
			System 2		
25	0.54	0.45	0.56	2.4	45%
500	0.61	0.45	2.68	7.5	5%
1000	0.49	0.45	4.92	7.0	0%
			System 1		
1000	0.51	0.45	4.92	12.0	0%
			System 3		
HMM states					
5	0.60	0.45	4.92	44.8	0%
10	0.54	0.45	4.92	24.2	0%
15	0.49	0.45	4.92	6.8	0%
20	0.53	0.45	4.92	20.2	0%
30	0.55	0.45	4.92	28.5	0%

available in the dataset. c_r is fixed (100) and the comparison is based on the different c_f (25, 500, and 1000) as shown in Table 1. It was observed that as the c_f increases, $\widetilde{Q^*}$ gets closer to the ideal cost as well as the number of failed units decreases to 0%. However, the agent becomes more cautious suggesting replacement action earlier in the lifetime of the engine; thereby, increasing the average remaining cycles. Therefore, a balance between the dynamic replacement cost (decreasing cost with increased life cycles) and the higher failure cost must be maintained. Table 1 also shows the results of the optimal action policy learned by the agent through System 1. As a comparison, it can be concluded that the additional information of the operating conditions helps the model to learn a better maintenance policy.

6.1.2. Calculating the number of hidden states

System 3 has been used here to find the number of states of HMM model that maximizes the likelihood for our state space as well as the performance of DRL through an iterative process. 15 states of the HMM showed better performance results than the rest. The model trained through HMM gives the posterior probability distribution for every state as shown in Equation 6, which is then fed as an input to the DRL agent to be able to learn the optimal maintenance (replacement) policy. The experiment was performed on the test set using the failure cost of 1000 and with the same parameters as the previous experiment for better comparison. The model with the HMM outperforms System 1 and System 2 as shown in Table 1.

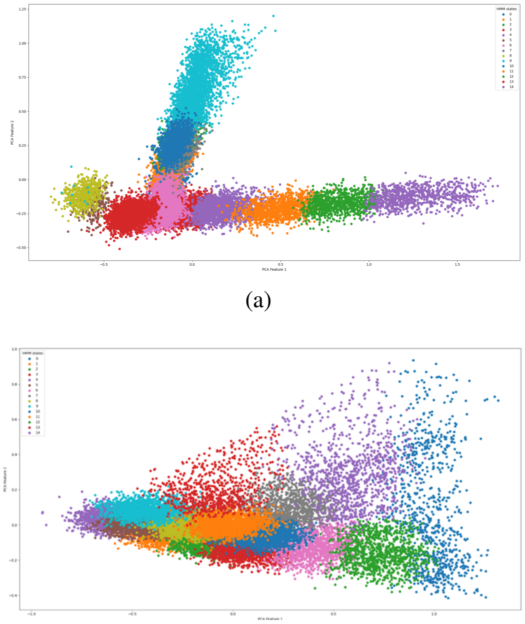
7. Experiment 1: Interpretations Based on the Hidden States

Datasets FD003 and DS01 are used in this section with System 3 for event-based hypothesis and state interpretations. The experiments performed here are to address the question, can the hidden state help towards interpretability?

7.1. Interpretability - Failure Event Hypothesis

Based on the state sequence distributions predicted by the HMM through Viterbi algorithm from Equation 8, each state of a particular event can be decoded, such as the failure mode or the degradation stage as shown in (Giantomassi et al., 2011). Due to the unavailability of the ground truth for other state mappings in FD003, just the failure states (last cycle state) were mapped in this experiment. Figure 5(a) plots state distributions for each data point based on the hidden states of the HMM on FD003. The data points are collected from the sensor readings of every engine per cycle; reduced to 2D features through Principal Component Analysis (PCA) for visualization. It can be hypothesized that each of these state clusters defines a particular event. The dataset was provided with the insight that it consists of 2 failure modes (HPC and fan degradation), however, the ground truth for the engines corresponding to which failure mode was not provided. Analyzing the failure states revealed two states that corresponded to the failure event (state 9 and 14) as visualized in Figure B.1 of Appendix B, which might be based on the two failure modes. Just to validate this hypothesis it was compared with FD001 as shown in Figure 5(b), where only one failure mode exists as defined in the dataset description and further analysis showed that only one state (state 0) was observed to be the failure state for each engine as shown in Figure B.2 of Appendix B. Therefore, giving an initial heuristics of HMM state distribution based on the failure events.

To discover the most relevant sensor readings corresponding to these failure states that triggered the HMM to predict such a state, feature importance was performed. Raw sensor readings were used as the input feature to the model and HMM state predictions based on the Viterbi algorithm were used as the target. After fitting the model, the importance of each sensor could be extracted for each HMM state. Table 2 shows the subset of the output of the feature importance for the failed states. The features with relatively higher score were selected from each class and its corresponding actual sensor information and description were extracted from (Saxena & Goebel, 2008) as described in Table 3. From the background information of the sensor descriptions, it was observed that the sensor importance for two different states showed a concrete failure event interpretation that corresponded to the failure described in the dataset (HPC and Fan degradation), as hypothesized in Table 4. Algorithm A.3 of Appendix A defines the feature importance usage in the context of HMM state interpretation.



(a)

(b)

Figure 5. HMM states clustering for (a) FD003 and (b) FD001.

Table 2. Feature (sensor) Importance.

STATE: 9	STATE: 14
$feature$ 5: -12.497	$feature$ 5: 4.211
$feature$ 6: -3.873	$feature$ 6: 0.268
$feature$ 7: -5.984	$feature$ 7: 0.175
$feature$ 8: 0.463	$feature$ 8: 19.697
$feature$ 9: -7.529	$feature$ 9: 0.325
$feature$ 10: -12.737	$feature$ 10: 3.973
$feature$ 11: -3.454	$feature$ 11: 0.153
$feature$ 12: -5.651	$feature$ 12: 0.097
$feature$ 13: 4.036	$feature$ 13: -3.555

7.1.1. Remaining Useful Life (RUL) estimation

Another additional benefit of using the HMMs is RUL prediction per every cycle in an unsupervised and online setting. The Viterbi algorithm was used to predict the optimal sequence of states until the cycle at a particular time step. Based on the last observed state, it can predict the next most probable state using the transition probability matrix. Using the emission probabilities, one can sample the sensor observations based on the predicted next state and append it to the previously seen observations sequence. This process was continued until the sequence predicted the next state to be the failure state as decoded through the methodology discussed in the earlier Section 7.1. The total number of transitions to the failure state gives the RUL at that particular cycle as elaborated in Algorithm A.4 of Appendix A. For each cycle, the trend can be predicted as shown in Figure 6.

Table 3. Feature to sensor description.

Feature	Sensor	Description
$feature$ 5	P_{30}	pressure at HPC outlet
$feature$ 8	epr	Engine pressure ratio
$feature$ 10	phi	fuel flow : pressure (HPC)
$feature$ 13	BPR	Bypass Ratio

Table 4. Sensor importance to failure event hypothesis.

HMM state	Important sensor reading	Failure event hypothesis (interpretation)
9	BPR	Fan degradation
14	P_{30}, epr, phi	HPC degradation

7.2. Interpretability - State Decoding and Mapping

Apart from the failure event hypothesis, it is necessary to measure the health state of the equipment at different points to generate an alarm for the user when the equipment reaches a critical point of its lifetime. However, the dataset used until now does not contain any such ground truth over which the performance of such interpretability of HMM states can be tested. Therefore, the second version of the dataset (Chao, 2021) was used to evaluate the state interpretability of HMM throughout the lifetime of the engine and the subset of which is shown in Figure 7. The dataset has the ground truth values of engine degradation per equipment, boolean health state value, and remaining useful life. The interpretations were based on the critical points along the equipment degradation curve as shown in Figure B.3 of Appendix B. Based on the degradation curve, and the range of HMM states observed during those conditions it was concluded that a relevant set of state hypotheses can be interpreted through the state distribution, as shown in Table 5.

8. Experiment 2: Comparison of SRLA with Baseline and Prior Work

Until now, the dataset used just consisted of 1 operating condition, however, in real-world cases, this is not the case. To adapt to a more general architecture, an Input-Output Hidden Markov Model (IOHMM) is used instead of the HMM. Dataset FD002 is used in this experiment section having 6 operating conditions, which will be used for the comparative evaluation with the prior work, i.e., to the best of our knowledge, the state-of-the-art methodology (Skordilis & Moghaddass, 2020) in this particular case.

8.1. Comparative Evaluation and Results

As seen in Section 7.2, HMM could distribute the states well according to the engine health state, however, the dis-

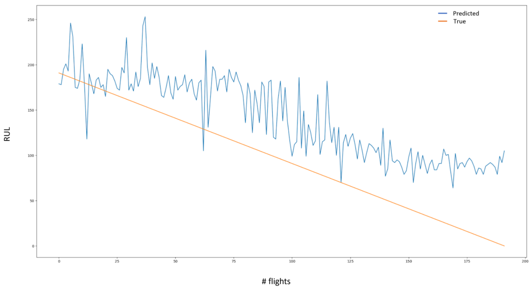
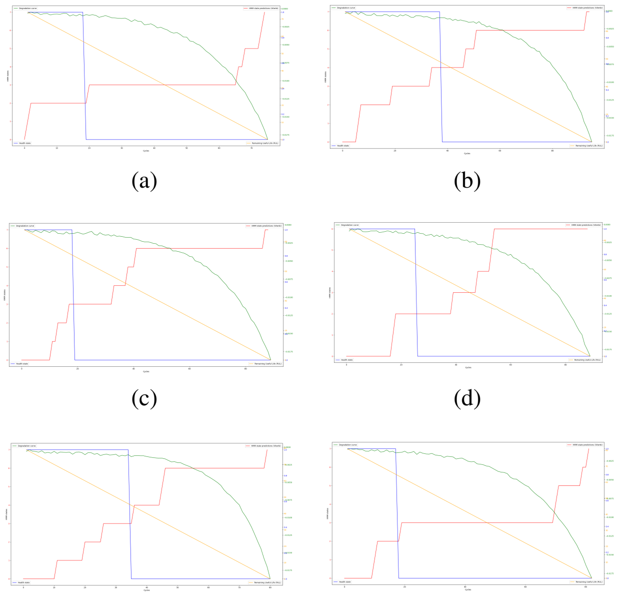


Figure 6. Remaining Useful Life estimation



(a) (b)

(c) (d)

(e) (f)

Figure 7. State decoding and mapping for dataset DS001.

tribution was not fine enough to replace the engine just 1 cycle before the failure, which is the most crucial point for the performance of the replacement policy as designed in this paper. Therefore, DRL is used to refine the granularity after state distribution based on IOHMM, resulting in a hierarchical model. To evaluate the performance, the results are compared with four baseline systems and prior work (Skordilis & Moghaddass, 2020), where the authors have used the Particle Filtering (PF) based-DRL. The authors used 80 engines as the training set and 20 as the test set out of 260 engines. However, the engines were selected randomly, therefore the exact comparison with the average cost of the agent could not be made. Therefore, the ratio of the Ideal Maintenance Cost (IMC) to the average cost of the agent ($\widetilde{Q^*}$) was compared in Table 6. As shown, SRLA outperforms the baseline systems and has a comparative performance with PF + DRL methodology with the added benefits of interpretability. Inference algorithm for SRLA is described in Algorithm A.5 of Appendix A.

Table 5. HMM state interpretability to equipment conditions.

EQUIPMENT CONDITION	HMM STATES
NORMAL EQUIPMENT	0 - 2
POTENTIAL FAULT POINT OF EQUIPMENT	2 - 4
FAILURE PROGRESSION	4 - 6
FAULT POINT OF EQUIPMENT FUNCTION	6 - 7
FAILURE	7

Conclusion and Future Direction

In this paper, a new hierarchical methodology was proposed utilizing the hidden Markov model-based deep reinforcement learning allowing the functionality of interpretability in the stochastic environment along with defining an optimal replacement policy and estimating remaining useful life without supervised annotations. Therefore, such a model can easily be used in industrial cases where the annotation of the fault type is difficult to obtain and the human supervisor in the loop can help define the state distribution according to the event-based analysis. To test the effectiveness of the model, NASA C-MAPSS (turbofan engines) dataset versions 1 and 2 were used. To evaluate the performance, it was compared with baseline models and prior work of Bayesian filtering based-deep reinforcement learning. The results were outperforming and comparable with the added benefits of interpretability with a less complex system model. In the future, the proposed architecture will be used on other open datasets to create a benchmark along with real-world case studies to measure its robustness.

References

Bengio, Y. and Frasconi, P. An input output hmm architecture. *Advances in neural information processing systems*, pp. 427–434, 1995.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-dynamic programming*. Athena Scientific, 1996.

Brockman, G. et al. Openai gym. preprint, arXiv, 2016.

Chao, A. Manuel. *et al. "Aircraft Engine Run-to-Failure Dataset under Real Flight Conditions for Prognostics and Diagnostics*, 6:1, 2021.

Chen, Z. et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

Do, P. et al. A proactive condition-based maintenance strategy with both perfect and imperfect maintenance actions. *Reliability Engineering & System Safety*, 133: 22–32, 2015.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, pp. 1–50, 2021.

Forney, G. D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

Giantomassi, A. et al. Hidden Markov model for health estimation and prognosis of turbofan engines. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 5480, 2011.

Griffith, S. et al. *Policy shaping: Integrating human feedback with reinforcement learning*. Georgia Institute of Technology, 2013.

Hofmann, P. and Tashman, Z. Hidden markov models and their application for predicting failure events. In *International Conference on Computational Science*, pp. 464–477. Springer, 2020.

Klingelschmidt, T., Weber, P., Simon, C., Theilliol, D., and Peysson, F. Fault diagnosis and prognosis by using input-output hidden markov models applied to a diesel generator. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, pp. 1326–1331, 2017. doi: 10.1109/MED.2017.7984302.

Lee, A. et al. hmmlearn. https://github.com/hmmlearn/hmmlearn, 2015.

Lee, J.-H. and Choi, J. Attaining interpretability in reinforcement learning via hierarchical primitive composition. *arXiv preprint arXiv:2110.01833*, 2021.

Lepenioti, K. et al. Machine learning for predictive and prescriptive analytics of operational data in smart manufacturing. In *International Conference on Advanced Information Systems Engineering. , Cham*, 2020.

Lyu, D., Yang, F., Liu, B., and Gustafson, S. Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2970–2977, 2019.

Meng, F., An, A., Li, E., and Yang, S. Adaptive event-based reinforcement learning control. In *2019 Chinese Control And Decision Conference (CCDC)*, pp. 3471–3476. IEEE, 2019.

Ong, K. S. H., Niyato, D., and Yuen, C. Predictive maintenance for edge-based sensor networks: A deep reinforcement learning approach. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pp. 1–6. IEEE, 2020.

Panzer, M. and Bender, B. Deep reinforcement learning in production systems: a systematic literature review. *International Journal of Production Research*, pp. 1–26, 2021.

Parra-Ullauri, J. M., García-Domínguez, A., Bencomo, N., Zheng, C., Zhen, C., Boubeta-Puig, J., Ortiz, G., and Yang, S. Event-driven temporal models for explanations-etemox: explaining reinforcement learning. *Software and Systems Modeling*, pp. 1–23, 2021.

Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Rabiner, L. and Juang, B. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986. doi: 10.1109/MASSP.1986.1165342.

Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Saxena, A. and Goebel, K. Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository :*, pp. 878–887, 2008.

Shahin, K. I., Simon, C., and Weber, P. Estimating iohmm parameters to compute remaining useful life of system. In *Proceedings of the 29th European Safety and Reliability Conference, Hannover, Germany*, pp. 22–26, 2019.

Sikorska, J., Hodkiewicz, M., and Ma, L. Prognostic modelling options for remaining useful life estimation by

Table 6. Comparison of the proposed methodology with baseline systems and (Skordilis & Moghaddass, 2020) on dataset FD002.

Methodology	$\widetilde{Q}^*$	IMC	CMC	IMC/$\widetilde{Q}^*$	Failure	Average Remaining cycles	Interpretations
System 1	2.10	0.64	7.02	0.30	20%	5.9	No
System 2	6.87	0.64	7.02	0.09	90%	2.6	No
System 3	7.02	0.64	7.02	0.09	100%	0.0	Yes
System 4	0.77	0.64	7.02	0.83	0%	23.0	Yes
PF + DRL [15]	2.02	1.93	20.80	0.96	0%	-	No
SRLA	0.69	0.64	7.02	0.94	0%	6.4	Yes

industry. *Mechanical systems and signal processing*, 25 (5):1803–1836, 2011.

Skordilis, E. and Moghaddass, R. A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics. *Computers & Industrial Engineering*, 147, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. No. 1, 2016.

Xu, D. and Fekri, F. Interpretable model-based hierarchical reinforcement learning using inductive logic programming. *arXiv preprint arXiv:2106.11417*, 2021.

Yin, M. and Silva, T. Iohmm. https://github.com/Mogeng/IOHMM, 2017.

Yoon, H.-J., Lee, D., and Hovakimyan, N. Hidden markov model estimation-based q-learning for partially observable markov decision process. In *2019 American Control Conference (ACC)*, pp. 2366–2371. IEEE, 2019a.

Yoon, H.-J., Lee, D., and Hovakimyan, N. Hidden markov model estimation-based q-learning for partially observable markov decision process. *2019 American Control Conference (ACC)*, 2019b. doi: 10.23919/acc.2019.8814849.

Yu, Y. Towards sample efficient reinforcement learning. In *IJCAI*, pp. 5739–5743, 2018.

A. *Algorithms and Training Parameters*

Algorithm A.1 Environment Modeling

```
Input:
$S_t = s_0, ..., s_t, ..., s_T$: state space
$A_t = a_0, ..., a_n$: action space
$R_t(s_t, a_t)$: reward given current state and chosen action
repeat
  step in environment and sample observed state and reward
  if $hold$ then
    if equipment has reached the failure state then
      reward of failure
      end of episode
      replace to new different equipment and observe $S_0^{m+1}$
    else
      reward of hold
      increasing the age of the equipment by one step
      observed next state: $S_{t+1}$
    end if
  else if $replace$ then
    if equipment has reached the failure state then
      reward of failure
    else
      reward of replacement
    end if
    end of episode
    replace to new different equipment and observe $S_0^{m+1}$
  end if
  Output: $S_{t+1}$ : next state, $R_t$: reward, end of episode
until all states have been observed
```

Algorithm A.2 Approximate Reinforcement Learning

```
  Input:
  $T_{max}$: epochs
  $\epsilon_0$: initial exploration parameter
  $\gamma$: discount factor
  $\alpha$: learning rate
  $S_t = s_0, ..., s_t, ..., s_T$: state space
  $A_t = a_0, ..., a_n$: action space
  $R_t(s_t, a_t)$: reward given current state and chosen action
  for $t = 1$ to $T_{max}$ do
    if $training$ then
      $\epsilon = \epsilon_0$
    else
      $\epsilon = 0$
    end if
    if $exploration < \epsilon$ then
      choose action randomly
    else
      Choose action with maximum q-value
    end if
    find $S_{t+1}$ and $R_t$ given the chosen action
    approximate $Q$ for actions in current states $\hat{Q}(s_t, a_t)$
    sum approximated $Q$ for chosen actions $\mathbb{E}\{\hat{Q}(s_t, a_t)\}$
    approximate $Q$ for actions in next state $\hat{Q}(s_{t+1}|s_t, a_t)$
    compute $Q*(s_{t+1}|s_t, a_t) = max(\hat{Q}(s_{t+1}|s_t, a_t))$
    $Q_{target} = R_t + \gamma(Q*(s_{t+1}|s_t, a_t))$
    update $\hat{Q}$ by MSE between target and previous value;
    $\hat{Q}(s_t, a_t) \rightarrow \hat{Q}(s_t, a_t) + \alpha(\frac{1}{n}\Sigma(Q_{target} - \hat{Q}(s_t, a_t)))^2$
    total reward $= \Sigma(R_t)$
    $S_t = S_{t+1}$
    decay epsilon by defined percentage
    if last state or end of episode then
      $Q_{target} = R_t(s_t, a_t)$
      break the loop
    end if
  end for
  Output: $\hat{Q}*(s_t, a_t)$
```

Algorithm A.3 Feature Importance

```
  Input:
  Viterbi state predictions as target classes: $\hat{S}$
  Normalized sensor readings as features: $X$
  repeat
    Fit features with the classes
    Extract the relevance of features corresponding to every class
    Output: Feature relevance
  until all features are evaluated for concerned states
```

Algorithm A.4 RUL Estimation

```
Input:
$Y$: observation sequence till cycle '$t'$
$A$: Transition probability matrix
$B$: Emission probabilities
for $every\ cycle$ do
    useful cycles = 0
    while $predicted\ state\ is\ not\ in\ failure\ state$ do
        for $100\ iterations$ do
            Predict state sequence using Viterbi algorithm
            Select most probable next state
            Sample sensor observations of predicted state
            Append predicted state to state sequence
            Append sampled observation to sequence
            Add 1 to 'useful cycles'
        end for
        RUL = Average of the useful cycles
    end while
end for
RUL for each point = list of RULs
Output: RUL prediction
```

Algorithm A.5 Specialized Reinforcement Learning Agent (SRLA)

```
  STEP I: IOHMM Training
  Input:
  $n$: number of hidden states
  $Y$: output sequences
  $U$: input seauences
  Output: $\lambda$: model parameters (initial, transition, and emission probability)

  STEP II: Viterbi Algorithm (IOHMM inference)
  Input: $\lambda, U, Y$
  Output: $\delta_t(i) = \max_{x_1,\cdots,x_{t-1}} P\left[x_1 \cdots x_t = i, u_1 \cdots u_t, y_1 \cdots y_t \mid \lambda\right]$

  STEP III: DRL Training
  Input:
  $\delta_s$: specific event (such as failure)
  $S_t$: $u_t$ + $y_t$
  Algorithm A.1
  Algorithm A.2
  Output: $\hat{Q}*(S_t, A_t)$

  STEP IV: SRLA Inference
  Input: $\lambda$, $\hat{Q}*(S_t, A_t)$, $S_t$: $(U_t, Y_t)$
  Step II, Algorithm A.3, and Algorithm A.4 for interpretations
  $\delta \rightarrow$ Specialized state $(X_s) \rightarrow U_s, Y_s$
  if $S_t$ in $X_s$ then
     $\hat{Q}*(s_t, a_t)$
     Algorithm A.1
  else
     $a_t =$ do nothing (hold)
  end if
  Output: $\hat{Q}*(\delta_t, s_t, a_t)$
```

A.1. Training Parameters

The summary of the DL framework within the RL architectures are as follows: (a) Deep Neural Network (DNN) consisting of a total of 37,000 training parameters and fully-connected (dense) layers with 2 hidden layers having 128 and 256 neurons, respectively, with ReLU activation. (b) Recurrent Neural Network (RNN) consists of a total of 468,000 training parameters and fully connected (LSTM) layers with 2 hidden layers having 128 and 256 neurons, respectively. The output layer consists of the number of actions the agent can decide for decision-making with linear activation. The parameters of the DRL agent are as follows: discount rate = 0.95, learning rate = 1e-4, and the epsilon decay rate = 0.99 is selected with the initial epsilon = 0.5.

B. *Figures*

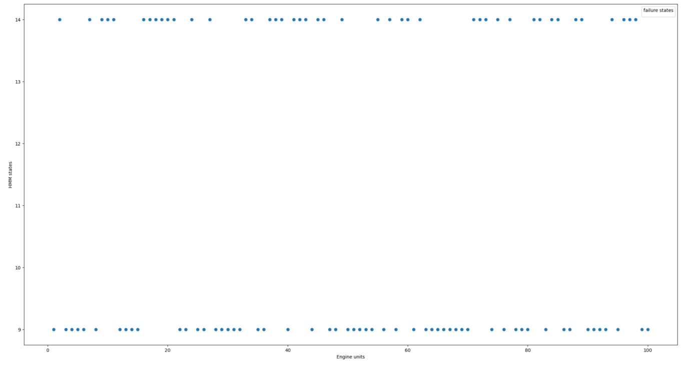


Figure B.1. States decoding and mapping for dataset FD003.

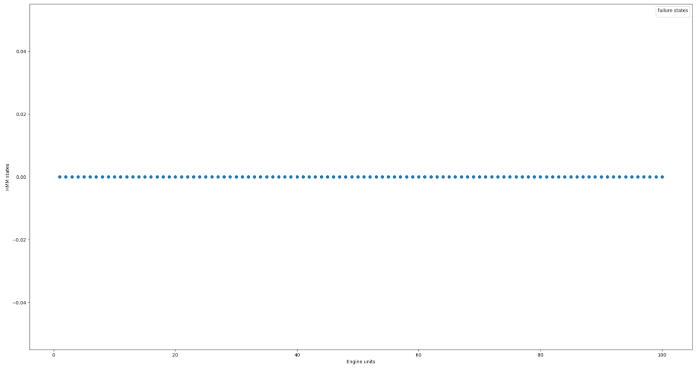


Figure B.2. States decoding and mapping for dataset FD001.

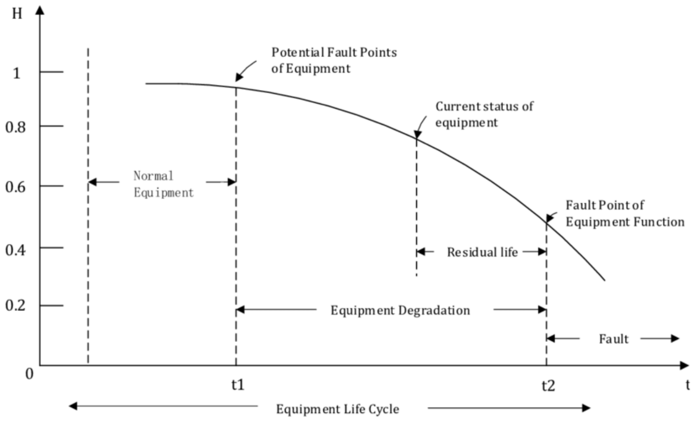

H
1
0.8
0.6
0.4
0.2
0
Potential Fault Points of Equipment
Current status of equipment
Normal Equipment
Fault Point of Equipment Function
Residual life
Equipment Degradation
Fault
t1
t2
t
Equipment Life Cycle

Figure B.3. Health degradation curve of equipment.