Predictive Maintenance using Deep Reinforcement Learning

Aravind A/L Supramaniam [1], Sharifah Sakinah Syed Ahmad [1,*], Zeratul Izzah Mohd Yusoh [1]

[1] Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka
Durian Tunggal, Melaka
*Email: sakinah@utem.edu.my

2024 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET) | 979-8-3503-8969-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IICAIET62352.2024.10730350

***Abstract*—The Fourth Industrial Revolution has impacted various sectors significantly. In machine and equipment-related industries, ensuring an uninterrupted supply to customers is crucial. Therefore, machines and equipment must always be ready and undergo maintenance at the right time to prevent downtime. However, industries believe that simply predicting failures based on historical data is insufficient for effective predictive maintenance. The prediction model must learn from the actions taken on the machines or equipment to develop an optimal maintenance policy. Deep Reinforcement Learning (DRL), a combination of Deep Learning (DL) and Reinforcement Learning (RL), harnesses the strengths of both techniques to handle complex and high-dimensional data. Despite its potential, DRL's application in Predictive Maintenance (PdM) is still emerging. Hence, Predictive Maintenance using Deep Reinforcement Learning (DRL) is proposed. By leveraging the advantages of deep learning (DL) and reinforcement learning (RL), it is possible to predict appropriate maintenance timing and determine the optimal maintenance policy. This paper will discuss the performance of Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) in prediction and policy optimization.**

Keywords—Predictive Maintenance, Reinforcement Learning, Deep Reinforcement Learning, CNN, RNN, NASA C-MAPSS, PdM, DRL, DQN, RL, DL

I. INTRODUCTION

Predictive Maintenance (PdM) is a condition-based maintenance technique that leverages prediction based on technologies to determine the right time to perform maintenance [1]. The maintenance actions must be performed before a failure occurs, aiming to minimize the maintenance cost and maximize the availability of the machine or equipment. However, industries believe that simply predicting failures is not sufficient for effective PdM. To develop an optimal maintenance policy, the prediction model must learn from the actions taken on the machine or equipment. This approach ensures that maintenance decisions account for failure likelihood and necessary repairs while minimizing costs and maximizing production. Furthermore, Deep Reinforcement Learning (DRL), which combines deep learning (DL) and reinforcement learning (RL), leverages the strengths of both techniques to handle complex and high-dimensional data. DRL has shown exceptional performance in applications like AlphaGo and playing Atari, which involve handling large state and action spaces. PdM is similarly complex, with high-dimensional sensor data and the need for continuous learning and adaptation over time to improve maintenance actions. While DRL appears well-suited for these challenges, its application in PdM is still emerging, with limited evidence and exploration in existing literature.

In this work, Predictive Maintenance using Deep Reinforcement Learning is proposed. DRL combines RL and DL to learn an optimal maintenance policy based on the sensor data from the machine or equipment. DRL works well with complex patterns and is fast at adapting to changes, enhancing the PdM approach. It not only uses historical data, but it also learns continuously from the actions it takes to form an optimal maintenance policy. RL is good at learning using the trial-and-error method, while DL is good at remembering and generalizing patterns from large datasets. Several studies have shown that DRL can perform well in complex and diverse tasks involving high-dimensional inputs. In the past decade, DRL has made substantial advances in many tasks that require perceiving high-dimensional input and making optimal or near-optimal decisions [2]. Experimental evaluation shows that maintenance policy using an RL-based approach outperforms traditional methods in terms of failure prevention and downtime, improving 75% of total performance [3]. As DRL can perform well on complex and large-scale problems, predictive maintenance can be said to be a complex problem as it involves analyzing the sensor data, historical maintenance records, and other relevant details to predict when the machine or equipment will fail or need maintenance. Despite its potential, few DRL-based maintenance frameworks have been proposed so far in the literature or are limited in several respects.

To select the DL network, several research studies were reviewed. The Convolutional Neural Network (CNN) model has a finite set of processing layers that enable learning various features of input. It can perform feature extraction from input samples through the convolutional layer [4]. In contrast, the Recurrent Neural Network (RNN) model is very good at learning sequences in time-series data, such as stock prediction and text generation [5]. This work utilizes both CNN and RNN architectures in the DRL approach, analyzing their performance in obtaining the optimal policy. The main contributions of this article are as follows:

1) Novel application of DRL for PdM is introduced, which uses a combination of DL and RL techniques to predict optimal maintenance actions.

2) The DRL algorithm derives an optimal maintenance policy for machines and equipment. It can indicate the best times to perform maintenance actions, which can maximize uptime and minimize the maintenance cost.

979-8-3503-8969-2/24/$31.00 ©2024 IEEE

Authorized licensed use limited to: Universiti Teknikal Malaysia Melaka-UTEM. Downloaded on November 28,2024 at 03:58:14 UTC from IEEE Xplore. Restrictions apply.

3) The DRL algorithm, which is Deep Q-Network (DQN), is tailored for PdM tasks. The comparative analysis of the performance of CNN and RNN within the framework is performed to access their effectiveness in PdM strategy.

II. Background and Related Work

This section contains two main objectives: first, to provide essential background information on DRL and PDM concepts; second, it presents the literature review of what has been done on topics related to DRL and PdM.

A. Background

1) Reinforcement Learning

Reinforcement Learning is a learning process in which an agent can periodically make decisions, adapt with the results, and automatically adjust its strategy to achieve the optimal policy in the environment. RL has the objective of maximizing expected cumulative rewards over the learning process [6]. The main components of RL are as below:

- Agent: Agent, also known as performer of actions in the environment, represents the learning algorithm.
- Action: Action is the activity performed by the agent in the environment; action space refers to the actions the agent can take.
- State: A state is an observation or situation of an agent in the environment.
- Environment: The place where the agent interacts in RL.
- Reward: Also known as reward function, used to provide numerical value based on the feedback from the environment based on action. Positive/desired feedback will get a reward; negative feedback will receive punishment.
- Transition state: moving from a state to another. Epsilon will be used as a transition probability, which will be used for agents to move from one state to another state in RL.

One of the learning algorithms for RL is Q-Learning. Q-Learning is a model-free, value-based, off-policy algorithm that makes the agent learn the value of taking an action at a particular state. DRL uses deep neural network (DNN) to approximate the Q-values. These Q-values will be updated iteratively when the agent interacts with the environment [7].

2) Deep Learning

In the context of DRL, deep learning needs to be applied to obtain the benefits of data-driven neural networks, which extract the complex features from the unstructured data to improve the RL strategy. There are several types of DNN that can be applied to this work. Since the datasets involved are complex, the DNN should be capable of handling them. CNN and RNN will be utilized in this work.

a) CNN

Convolutional Neural Network (CNN) is a type of DNN that uses feed-forward architecture, visualized in Figure 1. It is usually used for image recognition and processing tasks. However, some studies have successfully applied CNNs to time-series forecasting problems. This is because CNNs can automatically extract features without extensive feature engineering. The data need to be represented in 2D so that the model is able to extract the correlation among the features in the dataset [8].

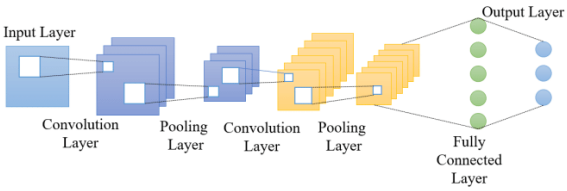

Fig. 1. A representation of CNN [9]

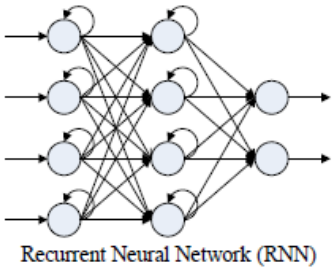

Fig. 2. A representation of RNN

b) RNN

Recurrent Neural Network (RNN) is a type of DNN that is used to process or recognize patterns in sequences of data. The connection of nodes in RNN forms a directed graph along a sequence like feature links from a layer to another, which allows information from one layer to its previous layer. In simple words, RNNs contain loops that allow information to be stored in the network and use their reasoning from previous experiences to inform upcoming events. Generally, RNN is used for time series data like stock prediction, text generation, machine translation, and much more [10].

B. Related Work

The application of machine learning (ML) in PdM has recently attracted a lot of interest. Massimiliano De Benedetti, Fabio Leonardi, Fabrizio Messina, Corrado Santoro, and Athanasios Vasilakos presented a learning approach using Artificial Neural Networks (ANNs) to predict AC power production based on solar irradiance and PV panel temperature for photovoltaic (PV) systems and detect anomalies by analyzing residuals between actual power prediction and predicted values [11]. This approach aims to enable early fault detection, preventing performance drops and power loss by ensuring timely maintenance.

Ayvaz and Alpay proposed a data-driven PdM system for production lines in manufacturing [12]. The system uses data from IoT sensors in real-time to detect signals for potential failures before they occur using machine learning methods such as Random Forest, XGBoost, Gradient Boosting, MLP Regressor, Support Vector Regressor, and AdaBoost. Comparative evaluation in their study showed that Random Forest and XGBoost outperformed in the assessment. This method helps address the issues by notifying operators of the maintenance actions to be done before production stops.

Silva and Capretz proposed a CNN-based framework for asset predictive maintenance (PdM) [13]. Their approach uses CNNs to identify abnormal patterns in sensor data and

Authorized licensed use limited to: Universiti Teknikal Malaysia Melaka-UTEM. Downloaded on November 28,2024 at 03:58:14 UTC from IEEE Xplore. Restrictions apply.

provide maintenance advice. They transform one-dimensional data into a two-dimensional format to leverage CNN's feature extraction capabilities. Their CNN-based framework outperforms traditional machine learning techniques like Random Forest, Support Vector Machine, and Multi-Layer Perceptron.

Recent work by Habib and Mohamed presents a machine learning-based PdM approach using a combined CNN-LSTM network for manufacturing facilities [14]. This approach leverages machine learning to predict maintenance needs, utilizing historical data to detect anomalies in machine operations and determine when maintenance actions are necessary. The paper highlights that CNNs are adept at extracting spatial features, while LSTMs are effective in capturing temporal dependencies, making the combined model suitable for PdM tasks in complex, nonlinear systems like those in manufacturing and aerospace industries.

Nguyen and Medjaher present a comprehensive predictive maintenance framework that integrates prognostics and maintenance decision-making processes [15]. The framework uses LSTM networks to predict the remaining useful life (RUL) based on the sensor measurement, estimating the probabilities of equipment failure. This approach is designed to assist operation planners in scheduling maintenance activities effectively, reducing downtime, and optimizing inventory management.

The above PdM studies focus on predicting maintenance before failure using ML methods. However, relying solely on predictions without a maintenance policy may not effectively minimize downtime and maximize profit. An optimal maintenance policy combined with failure prediction is crucial for a PdM strategy, enabling more accurate and informed maintenance actions.

III. Case Study

This work proposes two frameworks for the PdM strategy, one using RL with CNN and the other using RL with RNN.

A. Dataset

The dataset used in this work is the NASA Turbofan Jet Engine dataset. This is a synthetic dataset generated by the Prognostic Center of Excellence at NASA Ames using the C-MAPSS simulator. It contains multivariate time series sensor measurements from jet engines under various operational conditions [16]. This study focuses on the data from equipment F002. A total of 260 engine units, in which each engine had run until failure. A total of 53759 rows of data were used. The first 208 engine unit data is used for training, and the remaining 52 engine unit data is used for testing.

B. Data Preprocessing

In this part, columns ‘unit’ and ‘cycle’ were used for iteration, while the other 24 sensor data were normalized using the Min-Max scaler, transforming all values to a range of [0, 1], for faster convergence during the training step [17].

C. RL Environment

To fit the data into reinforcement learning, it is necessary to initialize a custom RL environment, allowing the agent to learn to choose the best action over time based on the RL strategy. The variables for RL were defined as below:

- Agent: The agent interacts in the environment with the given state and returns action.
- Action: The action space is 2, one is perform maintenance, and another is continue operation.
- State: The 24 columns of sensor data used as the state
- Environment: All required variables by RL
- Reward: -100 if engine failure, -50 divided by the number of cycles completed for maintenance action, and 0 for continued operation.
- Transition state: Transition of engine unit will happen when an engine undergoes maintenance or failure.

These are the required variables for implementing RL. Next, the step function needs to be defined. In this function, the agent interacts with the environment in the given state and will give action to this step function. Then, a calculation of reward based on action will be performed. Then, it will return the next state, reward, transition flag, and information. This output will be used to enhance the learning of the RL strategy in the DL model.

D. DL Network Architecture

CNN and RNN architectures are used in this work. The RL environment remains unchanged for both. The dataset will be reshaped to fit each model as explained below:

1) CNN Framework

The model consists of a Conv1D layer that performs convolution operations to extract the important features from the data. It uses 64 units of filter, which functions as the feature extractor. Next, MaxPooling1D is used to reduce the dimensionality of the input by taking the maximum value based on the pool size; a pool size of 2 is used. Then, a flatten layer is used to flatten multi-dimensional input into a one-dimensional vector. This layer ensures that the output is ready to be fed into fully connected, dense layers. The dense layer consists of 50 units of neurons. The output layer is a dense layer with 2 neurons, which gives out the approximation of the action value. The input layer and hidden dense layer use the ReLU activation function to introduce non-linearity in the model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 119, 64)           3136

 max_pooling1d (MaxPooling1  (None, 59, 64)            0
 D)

 flatten (Flatten)           (None, 3776)              0

 dense (Dense)               (None, 50)                188850

 dense_1 (Dense)             (None, 2)                 102

=================================================================
Total params: 192088 (750.34 KB)
Trainable params: 192088 (750.34 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 3. CNN network architecture

Authorized licensed use limited to: Universiti Teknikal Malaysia Melaka-UTEM. Downloaded on November 28,2024 at 03:58:14 UTC from IEEE Xplore. Restrictions apply.

The CNN model is used for Q-network in DRL, and the weights of the network are updated over the training process. The Q-Learning process computes the gradients of loss with respect to the network's trainable variables and applies weights to update the network. After training, the model is tested with testing data. CNN models are effective at feature extraction, dimensionality reduction, and handling sequential and high-dimensional datasets.

2) RNN Framework

The input shape of RNN is (50, 24), representing 50 time steps and 24 features. The first RNN layer has 24 units, and each of them maintains a hidden state that gets updated every time step. Tanh activation function is used to deal with vanishing gradient problem and to capture more complex patterns. The next layer is the RNN layer with 48 units and tanh activation function. The output layer is defined with 2 units for action predictions.

The RNN model is used for Q-network in DRL, and the weights of the network are updated over the training process. During training, the network weights are updated based on the computed gradients of the loss function. After training, the model is tested with testing data. RNN is suitable for sequential data and time series prediction.

E. Training

1) Get action

The agent chooses action to be performed using the Epsilon-greedy method. If the epsilon value is greater than the random value, the action will be chosen randomly between 0 and 1, else the argmax value of the value function prediction based on state will be chosen as the action.

2) Q-learning

The agent begins learning by deciding on an action, which is passed to the step function, this function will return the next state, reward, and done flag. The information of state, action, reward, next state, and done will be stored in a list called replay for network training later. Then, the total reward and state values are updated, and the agent performs the next action based on the next state. If maintenance action or failure occurs, the total reward is displayed and saved into the session rewards list. Then the engine unit transition will occur. This process continues until the loop reaches 208 iterations, corresponding to the 208 engine data.

3) Network Training and Apply Gradient

The previously saved replay list is used in this step. If the replay has more than the specified batch, a random sample is drawn for gradient calculation. It involves recalculating the q-value for the current state and the q-value for the next

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 50, 24)            1176

 simple_rnn_1 (SimpleRNN)    (None, 48)                3504

 dense (Dense)               (None, 2)                 98

=================================================================
Total params: 4778 (18.66 KB)
Trainable params: 4778 (18.66 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 4. RNN network architecture

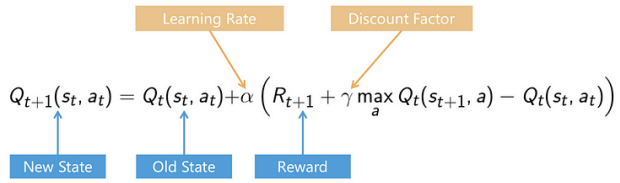

Fig. 5. The Bellman equation was used to calculate the target q-value [18].

state. The target q-value for action in the current state is calculated using the Bellman equation.

Then, the mean-squared error loss between the target q-value and predicted q-value is calculated. The loss value used as the gradient is applied to the network for backpropagation. The loss and epoch values are saved in a list for evaluation. Epsilon decay occurs after 100 epochs, resetting to initial epsilon if it goes below 0.016. This training method applies to both CNN and RNN models.

F. Testing

The trained model is tested on unseen data with training set to false. Two empty lists are created to store reward and session reward. For each testing unit, the action for the agent will be chosen based on the trained model. The chosen action will be passed to the step function to get the next state, reward, done flag, and info. Next, total reward and state will be updated. If the engine unit reaches failure or maintenance, the done flag will be true, the session reward will be stored, and the engine unit transition will happen. A total of 52 engine units is used for testing.

G. Evaluation

The trained model needs to be evaluated by comparing the training and testing results. Three methods of evaluation are visualized as below:

1) Loss Curve

The loss curve provides insights into the learning efficiency of the training model. During training, the model's loss is calculated using the mean squared error (MSE) technique, which measures the mean squared difference between the actual and predicted Q-values. This iterative loss calculation continues until the training process completes.

The loss values are recorded in an array of list for visualization purposes. The loss curve helps in understanding how well the model is fitting the data and whether the learning process is improving over epochs. Here, the loss curve of CNN is compared with the loss curve of RNN.

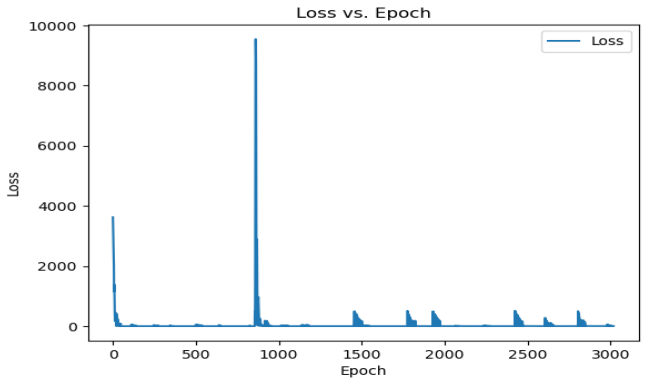

Fig. 6. Loss curve of CNN model

Authorized licensed use limited to: Universiti Teknikal Malaysia Melaka-UTEM. Downloaded on November 28,2024 at 03:58:14 UTC from IEEE Xplore. Restrictions apply.

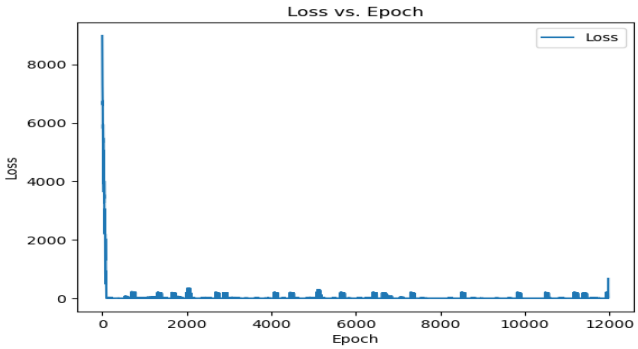

Fig. 7. Loss curve of RNN model

Based on the loss curves of CNN and RNN, it can be observed that the RNN model learns better compared to the CNN model, as it shows a comparatively lower loss value over the epochs, which indicates the RNN model is fitting the data over epochs.

2) Reward Graph

The graph of total reward per episode illustrates the agent's learning performance based on the rewards received from each episode. Each episode corresponds to an individual engine unit. The total reward per episode is computed by adding up the rewards that the agent accumulates while interacting with the engine unit. The total reward is computed and kept in an array of list for reference and visualization purposes. A higher total reward indicates that the RL agent has learned effectively.

Based on the total reward per episode during the training phase, the RNN model performs better than the CNN model. This is because higher rewards per episode are consistently achieved by the RNN model compared to the CNN model. The total reward per episode serves as an indicator of the RL agent's learning effectiveness, with higher rewards indicating the model could give a better performance.

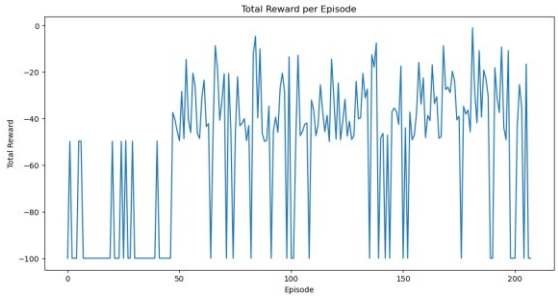

Fig. 8. Total reward per episode based on CNN model training phase

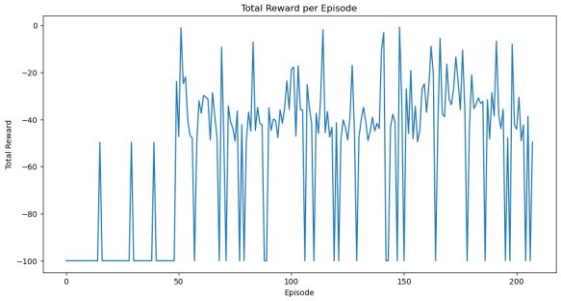

Fig. 9. Total reward per episode based on RNN model training phase

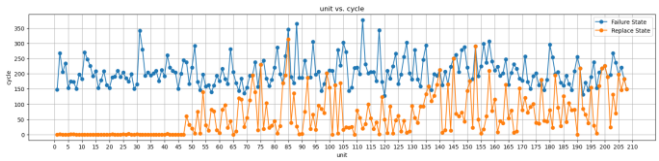

Fig. 10. Policy comparison of failure state and replace state of CNN model training phase

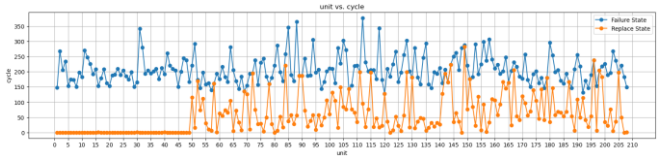

Fig. 11. Policy comparison of failure state and replace state of RNN model training phase

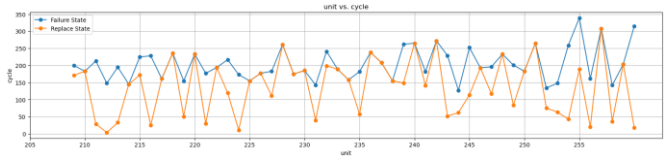

Fig. 12. Policy comparison of failure state and replace state of CNN model testing phase

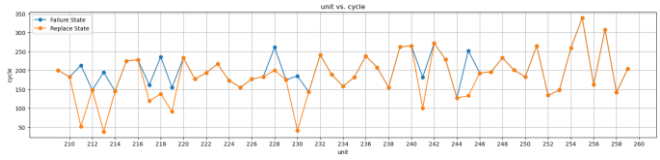

Fig. 13. Policy comparison of failure state and replace state of RNN model testing phase

3) Policy graph

The policy graph depicts the policy learned throughout the training and testing phases. According to the PdM concept, a policy that is closer to the failure state is considered efficient, provided it does not result in actual failures. Each engine unit, along with its replacement and failure states, is plotted on the same axis to facilitate a better comparison of the learned policies.

During the training phase, both models began learning the policy from the 50th engine unit. The RNN-based policy was able to learn closer to the failure state compared to the CNN-based policy, indicating that the RNN model performed better than the CNN model during this phase.

Based on testing, CNN model-based policy is estimated to be closer to its failure state compared to the RNN model-based policy. This suggests that CNN model performed well compared to RNN model during the training phase.

IV. Results and Discussion

The results from the loss curve, reward graph, and policy graph provided insights into the efficiency and performance of the models used in this work. During the training phase, the loss curve, reward graph, and policy graph indicated that the RNN-based model outperformed the CNN-based model. This suggests that the RNN model, with its ability to handle

Authorized licensed use limited to: Universiti Teknikal Malaysia Melaka-UTEM. Downloaded on November 28,2024 at 03:58:14 UTC from IEEE Xplore. Restrictions apply.

sequential data, effectively captures temporal dependencies and learns patterns indicative of failures.

However, during the testing phase, the CNN-based model demonstrated better performance compared to the RNN-based model. The CNN model's performance can be attributed to its effective feature extraction, robustness to noise, and ability to handle multidimensional data.

While the RNN-based model showed better performance during training by learning policies closer to failure states, it did not generalize as effectively during testing. The RNN-based model might have overfitted to the training data by learning noise, leading to poor performance in the testing phase. Conversely, the CNN-based model demonstrated better generalization and robustness during testing.

For PdM strategy, a model that generalizes well with new data and performs consistently in both training and testing phases is needed. Thus, the CNN-based model would be the better choice in this scenario, given its consistent performance across both phases. However, neither model is optimal as the replace state is still far from the failure state.

V. Reflection and Future Works

In this paper, it was recognized that the both developed models weren't able to perform better than the results that were achieved by previous research, as in [19]. However, the model is able to come out with a maintenance policy, although it is not as desired. The DRL application in PdM strategy to learn optimized maintenance policy is a good start for future development. The optimal maintenance policy should suggest maintenance between 10 and 40 cycles before the engine reaches failure state [20]. In the future, hyperparameter tuning and optimization of DL network architecture will be done to improve the current result stated in this work. Also, testing this method on different datasets will be done to look at the areas to apply to this strategy.

VI. Conclusion

This paper explored using a CNN network and a RNN network to predict the maintenance and get the optimal maintenance policy in the NASA C-MAPSS dataset FD002. The model framework and used data were chosen based on the analysis of the latest research in DRL and PdM strategy. During the evaluation of the CNN and RNN models, the results did not surpass the currently available research results. However, they highlighted areas for improvement and development for future work.

Acknowledgement

Thank you to the Ministry of Education (MOE) and Universiti Teknikal Malaysia Melaka (UTeM) for the financial supports through FRGS Vote No: FRGS/1/2020/ICT06/UTEM/02/1.

References

[1] M. Baptista, S. Sankararaman, I. P. de Medeiros, C. Nascimento,H. Prendinger, and E. M. Henriques, "Forecasting fault events for predictive maintenance using data-driven techniques and ARMA modeling,"

[2] Khorasgani, H.,Wang, H., & Gupta, C. (2020). Challenges of applying deep reinforcement learning in dynamic dispatching. arXiv preprint arXiv:2011.05570.

[3] M. L. R. Rodríguez, S. Kubler, A. De Giorgio, M. Cordy, J. Robert, and Y. L. Traon, "Multi-agent deep reinforcement learning based Predictive Maintenance on parallel machines," Robotics and Computer-integrated Manufacturing, vol. 78, p. 102406, Dec. 2022, doi: 10.1016/j.rcim.2022.102406.

[4] S. Gafsi, "Convolutional Neural Networks: Hyperparameters tuning and numerical results-A case study," *Technical Report*, May 2018, Clemson University.

[5] N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Networking: a survey," arXiv (Cornell University), Jan. 2018, doi: 10.48550/arxiv.1810.07862.

[6] C. C. Keat and S. S. Syed Ahmad, "Simulation of Smart Traffic Light By Using Image Processing and Reinforcement Learning," 2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), Kota Kinabalu, Malaysia, 2022, pp. 1-6, doi: 10.1109/IICAIET55139.2022.9936772.

[7] H. Li, R. Cai, N. Liu, X. Lin, and Y. Wang, "Deep reinforcement learning: Algorithm, applications, and ultra-low-power implementation," Nano Communication Networks, vol. 16, pp. 81–90, Jun. 2018, doi: 10.1016/j.nancom.2018.02.003.

[8] M. K. Habib and K. Mohamed, "Machine Learning-Based Predictive Maintenance: using CNN – LSTM Network," International Conference on Mechatronics and Automation, Aug. 2023, doi: 10.1109/icma57826.2023.10216091.

[9] H. Gu, Y. Wang, S. Hong, and G. Gui, "Blind channel identification aided generalized automatic modulation recognition based on deep learning," IEEE Access, vol. 7, pp. 110722–110729, Jan. 2019, doi: 10.1109/access.2019.2934354.

[10] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions," *Preprint*, arXiv:1909.00590, Sep. 2019. DOI: 10.48550/arXiv.1909.00590.

[11] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro, and A. Vasilakos, "Anomaly detection and predictive maintenance for photovoltaic systems," Neurocomputing, vol. 310, pp. 59–68, Oct. 2018, doi: 10.1016/j.neucom.2018.05.017.

[12] S. Ayvaz and K. Alpay, "Predictive maintenance system for production lines in manufacturing: A machine learning approach using IoT data in real-time," Expert Systems With Applications, vol. 173, p. 114598, Jul. 2021, doi: 10.1016/j.eswa.2021.114598.

[13] W. Silva and M. Capretz, "Assets predictive maintenance using convolutional neural networks," IEEE Computer Society, Jul. 2019, doi: 10.1109/snpd.2019.8935752.

[14] M. K. Habib and K. Mohamed, Machine Learning-Based Predictive Maintenance: using CNN – LSTM Network. 2023. doi: 10.1109/icma57826.2023.10216091.

[15] K. T. P. Nguyen and K. Medjaher, "A new dynamic predictive maintenance framework using deep learning for failure prognostics," Reliability Engineering & System Safety, vol. 188, pp. 251–262, Aug. 2019, doi: 10.1016/j.ress.2019.03.018.

[16] "NASA Turbofan Jet Engine data set," Kaggle, Jul. 26, 2019. https://www.kaggle.com/datasets/behrad3d/nasa-cmaps

[17] J. Brownlee, "How to Use StandardScaler and MinMaxScaler Transforms in Python," Machine Learning Mastery, Jun. 09, 2020. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

[18] F. Bansac, "Overview of deep Reinforcement learning," *AILEPHANT*, Oct. 23, 2019. https://ailephant.com/overview-deep-reinforcement-learning/

[19] H. Khorasgani, H. Wang, C. Gupta, and A. K. Farahat, "An offline deep reinforcement learning for maintenance Decision-Making," *arXiv (Cornell University)*, Jan. 2021, doi: 10.48550/arxiv.2109.15050.

[20] S. S. Afshari, B. Jonnadula, X. Xu, X. Liang, and Z. Yang, "Jet engine optimal preventive maintenance scheduling using Golden Section search and genetic algorithm," Journal of Prognostics and Health Management, vol. 2, no. 1, pp. 45–71, Jul. 2022, doi: 10.22215/jphm.v2i1.3321.

Authorized licensed use limited to: Universiti Teknikal Malaysia Melaka-UTEM. Downloaded on November 28,2024 at 03:58:14 UTC from IEEE Xplore. Restrictions apply.