Simultaneous Prediction of Remaining-Useful-Life and Failure-Likelihood with GRU-based Deep Networks for Predictive Maintenance Analysis

Ali Yuce Kaleli*, Aras Firat Unal*, Sedat Ozer
Ozer Lab., Bilkent University

2021 44th International Conference on Telecommunications and Signal Processing (TSP) | 978-1-6654-2933-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/TSP52935.2021.9522592

***Abstract*—Predictive maintenance (PdM) has been an integral part of large industrial sites collecting data from multiple sensors to reduce the maintenance power and costs with the advent of Industry 4.0. Two of the major problems in PdM used at large industrial sites are: (i) the prediction of remaining useful life (RUL); (ii) the prediction of the likelihood of failing within a predefined time period. While data oriented maintenance predictions were heavily focused on using classical techniques for such problems, recent interest shifted towards utilizing AI based solutions due to the better generalization capabilities of deep solutions. Among the time-sequence based deep networks, RNN, GRU and LSTM based networks are the most frequently used solutions. GRUs have demonstrated their faster learning capabilities with near or better prediction performance on certain tasks already. However, predicting multiple PdM tasks including both RUL and failure detection, simultaneously within the same network in an end to end manner with GRUs has not been much studied in the literature before. In this paper, we introduce a solution to predict those two tasks simultaneously within the same network based on GRUs. In our experiments we compare the performance of GRU layers to LSTM and RNN layers and report their performance on NASA dataset.**

***Index Terms*—GRUs, Remaining useful life prediction, failure prediction, predictive maintenance, time-sequence analysis**

I. Introduction

Recent interest in industry 4.0 has increased the attention on monitoring the condition of equipment in large production sites [1]. Today, with the introduction of predictive maintenance, it becomes a norm to collect data by installing multiple sensors at various locations in such production sites where aiming reduction in the maintenance related cost including time, manual labour and the lost energy. The arrival of data collection brings the need for better data processing and data analysis tools to improve the performance of such production sites while reducing the required resources for maintenance.

Predictive Maintenance (PdM) aims to decrease the number of failure occurrences and the maintenance costs by predicting such failures even before they occur. Artificial Intelligence (AI) based techniques have been recent interests to predict such occurrences. Among the AI techniques, the time sequence analysis based techniques such as Recurrent Neural Networks (RNN) [2], [3] and Long Short Term Memory (LSTM) [4] networks have been widely used. Recent interest mainly evolved around LSTM and Gated Recurrent Unit (GRU) [5]–[8].

In this paper, we focus on Remaining Useful Life (RUL) prediction and detecting the likelihood of a machinery to fail within a specific time period. Unlike the existing literature, we propose a combined GRU-based architecture to predict both RUL and likelihood of failure in an end-to-end manner. While the literature mainly studied the performance of different architectures for RUL prediction and machine failure within a specified time period individually, there are not many papers focusing on developing an end-to-end solution that predicts both RUL (regression problem) and failure likelihood (classification problem) within the same network. In this paper, we focus on that problem to reduce the computation and the required network size, while introducing a GRU-based network to solve both problems together simultaneously.

* Both authors contributed equally.

II. Related Work

Predictive maintenance has been a key subject for smart manufacturing for many years. Therefore, the relevant literature covers a large set of approaches proposed. Most of such approaches include classical techniques and deep learning based techniques are more limited when compared to the available classical techniques. For example, a rolling element bearings RUL has been estimated using dynamic regression in [9]. Other examples of predicting RUL in various setups using classical techniques can be found in [10]–[13]. However, as we introduce a deep learning based solution, our main focus here is listing the relevant work from the deep learning perspective.

Deep learning based solutions have been recent focus in predictive maintenance applications due to their superior performance over classical techniques in many applications. Among the available deep learning architectures, the ones with the capability of sequential processing took the main attention as PdM data is typically a time sequence. For example, in [2] the use of RNN for RUL estimation problem is studied and it is reported that RNNs can obtain better results when compared to other classical methods. As another line of work, the performance of LSTM based models was studied for PdM as in [3], [4]. As an alternative to LSTMs, the GRU based models have also been studied in the relevant literature recently. For example, a two-step GRU based solution was proposed in [6] for the RUL prediction problem. Another work in [5] also supported the results of [6] stating that GRU based networks perform faster with near or better performance when compared to LSTM networks. LSTM and GRU being two successful architectures, some papers focus on the comparison of those two techniques as in [5], [8], [14]. It has been mainly reported that the learning speed of the GRU based models has exceeded the learning speed of LSTM based models and that they both performed well with high efficiency in various PdM problems.

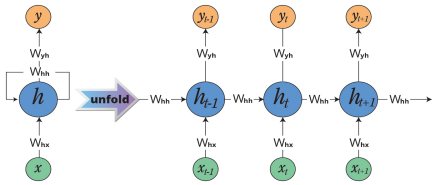
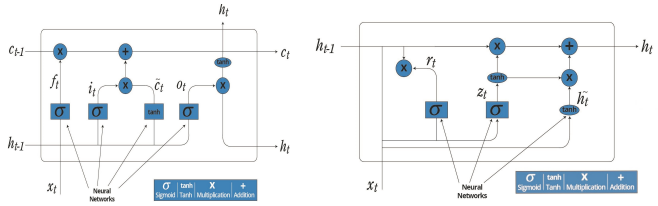


Fig. 1: An RNN layer unfolding onto itself over time.



(a) (b)

Fig. 2: Illustration of LSTM (a) and GRU (b) respectively

When the goal is only predicting RUL, the work in [14] reported that GRU can perform better than LSTM.

A more recent focus is using deep models that can jointly learn multiple tasks simultaneously. In such simultaneous learning problem, the goal is not predicting only one type of task (such as detecting failure or detecting RUL individually), but predicting multiple outputs jointly, e.g., detecting RUL and likelihood of the failure of a given machinery. There are not many papers available in the literature focusing on such simultaneous learning problems in PdM. A close relevant paper is [15] where the authors introduced an LSTM based network proposed to combine both regression and classification tasks for age prediction. Our work differs from all the above-mentioned literature in multiple ways: (1) we introduce using a GRU based architecture that jointly predicts both RUL and likelihood of failure within a given time period; (2) we compare the performance of three different networks based on LSTM, GRU and RNN on the NASA dataset [16] for the joint learning problem of RUL and the likelihood of the failure.

III. Background

Recurrent Neural Networks (RNN): Recurrent Neural Networks are considered to be self-connected hidden layers where each node within a layer is connected not only to the nodes within the next hidden layer but also connected to itself. Fig. 1 illustrates the structure of a typical RNN layer (on the left) and how it unfolds over consecutive time steps (on the right). The figure is illustrated based on [6]. The RNNs include 3 main aspects, i.e. input vector x, hidden layers h and the output vector y. The hidden and output layers are updated via the following formulas with the activation function f:

$$h_t = f(W_{hh}h_{t-1} + W_{hx}x_t), \quad y_t = f(W_{yh}h_t) \qquad (1)$$

where W_{yh}, W_{hh} and W_{hx} are the weight matrices for output, recurrent and input layers, respectively.

Long Short Term Memory (LSTM): Long Short Term Memory has been proposed by Hochreiter and Schmidhuber in 1997 [17]. LSTM based networks have been successfully used in many applications where the data is a time series as in [18]. The top illustration in Fig. 2, based on [6], shows the relation between the inputs and outputs of an LSTM architecture. Together with the feedback connections and gates, LSTM constructs a cell structure to act like a memory and pass information from start to the end. The input-output relationship for each gate of LSTM is given below, where x is the input and h is the output. f_t is the forget gate's activation, while i_t is the update gate's activation vector. The vectors o_t, $\tilde{c}_t$ and c_t are output gate's activation, cell input activation and state vectors respectively. c_{t-1} is the state from the previous time step. W and U are weight matrices. The $*$ corresponds to the Hadamard product (element-wise multiplication).

$$f_t = \sigma(W_f x_t + U_f h_{t-1}), \quad i_t = \sigma(W_i x_t + U_i h_{t-1}) \qquad (2)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1}), \quad \tilde{c}_t = tanh(W_c x_t + U_c h_{t-1}) \qquad (3)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t, \quad h_t = o_t * tanh(c_t) \qquad (4)$$

Gated Recurrent Unit (GRU): GRU is proposed as an alternative to LSTM in [19]. It is considered a simplified version of LSTM where some components within the LSTM model are deleted. The bottom illustration in Fig. 2 shows the general structure of a GRU model. In a typical GRU model (as in [6]) there exists only a reset layer in addition to RNN. The remember and forget layers of LSTM do not exist in the GRU architecture. Instead, there exists an update gate vector. Below equations define the reset gate, candidate activation vector, update gate vector and output vector.

$$z_t = \sigma(W_z x_t + U_f h_{t-1}), \qquad (5)$$

$$\hat{h}_t = tanh(W_h x_t + U_h(r_t * h_{t-1})) \qquad (6)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}), \quad h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t \qquad (7)$$

where x_t represents the input vector, h_t stands for output vector and $\tilde{h}_t$ is for candidate activation vector. z_t and r_t are update and reset gate vectors respectively. W and U are the weight matrices. The $*$ represents Hadamard product.

IV. Proposed Architecture

Our proposed two-branch GRU-based solution predicts both RUL and failures simultaneously in an end-to-end fashion. In our network, we have the first two layers used commonly for both regression and classification branches. After those two layers (GRU and drop out layers), our network branches into two different networks. That is, we have a regression branch for RUL prediction and a classification branch for failure detection. In this work, we mainly compare the performance of three different and recent blocks including RNN, LSTM and GRU blocks (layers). Therefore, to analyze the affect of using each of those three types in our experiments, we keep the model (the connections, the number of branches and the layer numbers) the same in our base network. The overall architecture of our network is given in Fig. 3 including all three of those types. The input for our model is a 2 dimensional array with size bxd where b is the window width and d is the sample dimension. The b value (the window size representing the amount of samples used at each time) is decided after performing a heuristic search on our dataset where we train our network at different window sizes (see Table I). In our approach, we utilize separate loss functions for each branch.

The classification branch starts with a GRU (or an LSTM or a simple-RNN) layer followed by a dropout layer. Then,

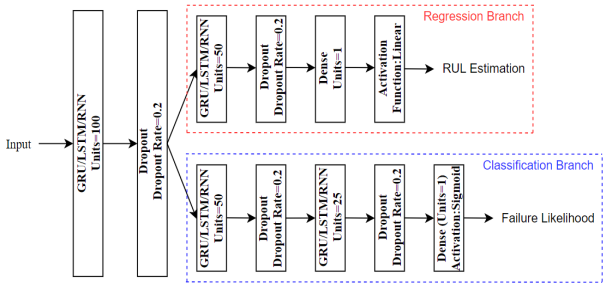


Fig. 3: Our proposed two-branch architecture.

those two layers repeat themselves again. The last layer of the classification branch is a dense layer whose activation is set to sigmoid. The output of the branch is a one dimensional (scalar) number, indicating whether there is a failure or not. The ADAM optimizer is used for the classification branch in all experiments. Binary Cross Entropy (BCE) is the loss function that is used to train the classification branch which is calculated as: $BCE = -\frac{1}{n}\sum_{i=1}^{n} y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)$, where n refers to the total number of samples, y_i is the ground truth for i^{th} sample and $\hat{y}_i$ is the prediction for i^{th} sample.

The branch that is used for regression problem differs from the branch that is used for classification in terms of number of algorithm layers (i.e. GRU, LSTM or RNN) and the activation layer. The regression branch starts with a GRU layer followed by a dropout layer. That dropout layer is connected to a dense layer which uses linear activation function to yield our final scalar prediction. The scalar output represents the remaining useful life of a machine. The RMSPROP optimizer is used for the regression branch in all experiments. We used Mean Squared Error (MSE) as the loss function and it is defined as: $MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$, where n refers to number of samples, y_i refers to the i^{th} ground truth RUL value and $\hat{y}_i$ is the predicted RUL value.

V. Experiments

We design our experiments to analyze the performance of GRU, LSTM and RNN layers for PdM, respectively, when we predict both RUL value and failure detection simultaneously. For that purpose, we utilized the NASA Predictive Maintenance Dataset[1]. In our experiments, Adam optimization with learning rate of 0.001 is used in all the classification branches and RMSprop with learning rate of 0.001 is used in all the regression branches. Algorithms are trained over 100 epochs and the best epoch value is reported in our tables. In our experiments, b is set to 50 and d is set to 25.

1) Dataset: Throughout the experiment the focus is directed to the NASA dataset [16] called FD001. The time series dataset is split into training and test data. The training dataset contains a total of 20631 instances while the test dataset contains 13096 instances. Each sample includes the cycle index and 25 different feature data (21 from sensors, 3 from settings and 1 from cycle number) for 100 different machines along with their corresponding RUL values. For failure detection, we create binary classification labels according to the given ground truth RUL values and obtain the final two dimensional output labels for each sample normalized between 0 and 1.

[1]https://ti.arc.nasa.gov/c/6/

2) Used Metrics: We utilize multiple metrics in our experiments. Accuracy, Precision, Recall, F1, Area-Under-Curve (AUC) are used to analyze the performance of classification branch (failure prediction); Mean-Absolute-Error (MAE) and Coefficient-of-Determination (R^2) are used to measure the performance of the RUL prediction (regression) branch. Definition of each of those metrics can be found in [20], [21].

3) Window Size Selection: To consider the relation between the consecutive information in the time series data, we select a window size that combines all the samples in that window to form a chunk of samples. We studied the performance of changing the window size on RUL prediction and reported the results in Table I in two different metrics (MAE and R^2 . While the best results are obtained when the window size is 100, the value 100 is a too big value for the existing dataset as not all the machines had that long samples. Therefore we picked the next best and usable value: 50.

Results: Here we first report our experimental results for simultaneous (joint) training of classification and regression branches. Then we also report individual training of networks for only classification and then for only regression branches.

4) Joint Training Results: Table II reports our results of jointly training both branches for three different configurations of using LSTM, RNN and GRU layers. In our experiments, we kept the architecture (the number of layers, the number of used dropout layers) the same and only used one of the LSTM, RNN or GRU options to create an LSTM, an RNN and a GRU based network. Each network simultaneously predicts both failure likelihood (as the output of the final sigmoid function of the classification branch) and the RUL value as the regression output of the regression branch. Table II compares the performance of each network for both branches (under regression and classification columns). The best values are shown in bold. The total number of trainable parameters for each network is also given in the lost row. The lower number of trainable parameters is better from the computational aspect.

As the table shows, for both MAE and R^2 metrics, i.e., for the regression branch, GRU based network yielded the best results when compared to both LSTM and RNN networks. For the classification branch, the results were very similar between GRU and LSTM networks. However, they both performed better than RNN based network in all five metrics. From the computational perspective, the GRU network yielded the least number of trainable parameters (as shown in the table).

5) Individual Training of Each Branch: In addition to jointly training regression and classification branches in an end to end manner, we also trained the networks for each branch

TABLE I: RUL prediction at different window lengths.

	1	**2**	**5**	**10**	**25**	**50**	**100**
MAE	56.62	24.21	23.82	22.11	19.25	11.41	**8.19**
R^2	-0.25	0.67	0.68	0.67	0.75	0.86	**0.90**
Number of machines	100	100	100	100	100	93	70

TABLE II: Comparison of LSTM, RNN and GRU based network results for joint training of both branches.

	Classification						Regression		
	Ace.	Prec.	Rec	F_1	AUC	Param	MAE	R^2	Param
GRU	**0.99**	**0.99**	0.97	**0.98**	**0.99**	66701	**11.29**	**0.86**	60951
LSTM	**0.99**	0.98	**0.98**	**0.98**	**0.99**	88226	12.20	0.80	80651
RNN	0.94	0.84	0.88	0.86	0.92	22076	39.19	−0.09	20201

TABLE III: Comparison of the individually trained model results.

	Classification							Regression			
	Acc.	Prec.	Rec	F_1	AUC	Param	Time	MAE	R^2	Param	Time
GRU	**0.99**	**0.99**	0.98	**0.99**	**0.98**	66701	**85.21**	**15.63**	0.74	60951	**81.48**
LSTM	**0.99**	0.97	**0.99**	0.98	**0.98**	88226	92.43	15.78	**0.75**	80651	87.88
RNN	0.96	0.96	0.85	0.90	0.97	22076	698.91	45.96	−0.94	20201	628.61

individually. First, we experimented on training only the classification branch for each network on the NASA dataset. For the classification branch, we utilized all five classification metrics (Accuracy, Precision, Recall, F_1, AUC) to obtain the results on the test dataset. Those results are summarized in Table III. As Table III demonstrates, both LSTM and GRU based networks yielded very similar results. They both performed significantly better than RNN classification.During the training, RNN required the least number of trainable parameters, however, its training time was significantly higher.

Next, we studied the performance of each network for only the regression task (for RUL prediction). For that purpose, we trained only the regression branch for each network on the NASA dataset from the Figure 3. For the regression branch, we utilized two metrics (MAE, R^2) in our experiments on the test dataset. Those results are summarized in Table III. As the table demonstrates, GRU and LSTM yielded very similar results while training of the GRU network was the shortest. Both GRU and LSTM networks performed significantly better than the RNN network.

VI. Conclusion

Signal and data processing on time sequences in large industrial sites has gained interest of many researchers recently due to the rise in the number of installed sensors and due to the data collection tools used in such industrial sites. With the recent developments in AI, the interest in processing time sequences shifted gradually towards the deep learning based solutions. While LSTM and RNN based solutions have been used widely to predict RUL or to classify failure individually, the field of predicting multiple error types and their lengths is a relatively new field and, therefore, there are not many literature available simultaneously predicting such multiple outputs in an end to end way. This work aims to help on that subject by introducing a GRU based solution to predict both RUL and failure simultaneously in an end to end manner. We compared our GRU based network's performance to its LSTM and RNN based counterparts and observed that utilizing a GRU based network has benefits over utilizing an LSTM or a simple-RNN based network for RUL prediction and failure detection simultaneously. Those benefits include: (i) lesser training time which can be useful where re-training is done frequently or in semi-online training environments, (ii) similar or better performance for both classification and regression tasks when compared to LSTM as our results indicate.

Acknowledgment

This paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No:118C356). However, the entire responsibility of the paper belongs to the owner of the paper. The financial support received from TÜBİTAK does not mean that the content of the publication is approved in a scientific sense by TÜBİTAK. The authors would like to thank Teknopar researchers for their valuable support during this study.

References

[1] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li, "Predictive maintenance in the industry 4.0: A systematic literature review," *Computers & Industrial Engineering*, p. 106889, 2020.

[2] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98–109, 2017.

[3] S. Zhao, Y. Zhang, S. Wang, B. Zhou, and C. Cheng, "A recurrent neural network approach for remaining useful life prediction utilizing a novel trend features construction method," *Measurement*, vol. 146, pp. 279–288, 2019.

[4] M. Yuan, Y. Wu, and L. Lin, "Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network," in *2016 IEEE international conference on aircraft utility systems (AUS)*. IEEE, 2016, pp. 135–140.

[5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[6] J. Chen, H. Jing, Y. Chang, and Q. Liu, "Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process," *Reliability Engineering & System Safety*, vol. 185, pp. 372–382, 2019.

[7] Z. Qu, L. Su, X. Wang, S. Zheng, X. Song, and X. Song, "A unsupervised learning method of anomaly detection using gru," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2018, pp. 685–688.

[8] C. Wang, W. Du, Z. Zhu, and Z. Yue, "The real-time big data processing method based on lstm or gru for the smart job shop production process," *Journal of Algorithms & Computational Technology*, vol. 14, p. 1748302620962390, 2020.

[9] W. Ahmad, S. A. Khan, M. M. Islam, and J.-M. Kim, "A reliable technique for remaining useful life estimation of rolling element bearings using dynamic regression models," *Reliability Engineering & System Safety*, vol. 184, pp. 67–76, 2019.

[10] Y. Hu, H. Li, P. Shi, Z. Chai, K. Wang, X. Xie, and Z. Chen, "A prediction method for the real-time remaining useful life of wind turbine bearings based on the wiener process," *Renewable energy*, vol. 127, pp. 452–460, 2018.

[11] K. Le Son, M. Fouladirad, and A. Barros, "Remaining useful lifetime estimation and noisy gamma deterioration process," *Reliability Engineering & System Safety*, vol. 149, pp. 76–87, 2016.

[12] M. H. Ling, H. Ng, and K. L. Tsui, "Bayesian and likelihood inferences on remaining useful life in two-phase degradation models under gamma process," *Reliability Engineering & System Safety*, vol. 184, pp. 77–85, 2019.

[13] Y. Chen, Y. He, Z. Li, L. Chen, and C. Zhang, "Remaining useful life prediction and state of health diagnosis of lithium-ion battery based on second-order central difference particle filter," *IEEE Access*, vol. 8, pp. 37 305–37 313, 2020.

[14] R. Naren and J. Subhashini, "Comparison of deep learning models for predictive maintenance," in *IOP Conference Series: Materials Science and Engineering*, vol. 912, no. 2. IOP Publishing, 2020, p. 022029.

[15] J. Chen, L. Cheng, X. Yang, J. Liang, B. Quan, and S. Li, "Joint learning with both classification and regression models for age prediction," in *Journal of Physics: Conference Series*, vol. 1168, no. 3. IOP Publishing, 2019, p. 032016.

[16] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set," *NASA Ames Prognostics Data Repository*, pp. 1551–3203, 2008.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] R. Valiente, M. Zaman, S. Ozer, and Y. P. Fallah, "Controlling steering angle for cooperative self-driving vehicles utilizing cnn and lstm-based deep networks," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2423–2428.

[19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[20] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[21] A. Botchkarev, "Evaluating performance of regression machine learning models using multiple error metrics in azure machine learning studio," *Available at SSRN 3177507*, 2018.