Domain Adaptation Remaining Useful Life Prediction Method Based on AdaBN-DCNN

Jialin Li
School of Mechanical Engineering and Automation
Northeastern University
Shenyang, China
Jialinli_neu@163.com

Xueyi Li
School of Mechanical Engineering and Automation
Northeastern University
Shenyang, China
lixueyineu@gmail.com

David He
Department of Mechanical and Industrial Engineering
University of Illinois at Chicago
Chicago, IL, USA
davidhe@uic.edu

Abstract—**Prognostics and health management (PHM) has received much attention as an emerging discipline. And the prediction of remaining useful life (RUL) is the core of the PHM. The data-driven RUL prediction methods are more favored because they can be developed faster and cheaper. However, the existing data-driven prediction models usually can only be used under the same data domain (DD), and they require a lot of labeled data to retrain a new prediction model. So a domain adaptation prediction model is more desirable. In this paper, a domain adaptation RUL prediction model is proposed by integrating the adaptive batch normalization (AdaBN) into deep convolutional neural network (DCNN). The improved AdaBN-DCNN model can not only improve the accuracy of the prediction, but also adapt to the prognostic tasks under different DDs. The sliding time window (TW) and the improved piecewise linear RUL function are also used in this paper to improve the prediction capability of the model. The proposed RUL prediction model is validated using the C-MAPSS turbofan engine dataset provided by NASA. The prediction results show that the proposed model not only has a strong predictive power but also adapts to different DDs.**

Keywords-adaptive batch normalization; deep convolutional neural network; time domain adaptation; remaining useful life

I. INTRODUCTION

The remaining useful life (RUL) prediction has received much attention as the core of Prognostics and Health Management (PHM). The maintenance strategy can be given in advance according to the RUL prediction result, and the equipment running status also can be adjusted in time. The RUL prediction model can be divided into 3 types: model-based method, data-driven method, and hybrid method. The data-driven prediction model is more attracted because it does not need to consider the system structure when it is developed.

Developing a data-driven model requires a large amount of training data, and it also takes a lot of time and effort to obtain the completely running data of a new system. Therefore, the public datasets such as Lithium-Ion batteries dataset provided by NASA, turbofan engine simulation dataset carried out using C-MAPSS, and FEMTO bearing dataset are more preferred used to validate prediction model.

REN et al. [1] proposed a prediction method based on spare autoencoder (SAE) and deep neural network (DNN) to predict the RUL of lithium-ion battery. The extracted features are first fused by an SAE, and then RUL is predicted by DNN. Considering the capacity unmeasurable problem of operating battery, Liu et al. [2] combined indirect health indicator (HI) with multiple Gaussian process regression (GPR) for RUL prediction. Considering that statistical features and have different contribute to the health indicators, Guo et al. [3] used recurrent neural network (RNN) based health indicator for bearings RUL prediction. The existing work generally believes that the two-stages of degradation processes are independent of each other which is not match the actual situation. To address this problem, Wang et al. [4] proposed novel two-stage Wiener process model with Bayesian method and stage correlation for RUL prediction. Li et al. [5] proposed a new data-driven prediction method based on deep convolutional neural networks (DCNN), and used C-MAPSS simulation data for validation. Babu et al. [6] also used a novel DCNN based regression method for RUL estimating.

The data-driven prediction methods rely solely on collected data to develop a model. The models developed under different working conditions and fault types are also very different. The dataset used to develop a model with different working conditions and fault types is called the data domain (DD). The prediction models developed in different DDs are often not universal to each other. Several RUL prediction methods mentioned above can only work under same DD. Once the predicted DD changes, it is necessary to retrain the network, which is time consuming and laborious. In order to improve the

This work was funded by the National Natural Science Foundation of China (Grant no. 51675089).

adaptability of the model to different DDs, the researchers have made many attempts. The transfer learning (TL) method fine-tune a part of parameters in the network with less time and training samples to improve the prediction accuracy of the network. Zhang et al. [7] used the BiLSTM network to develop a prediction model. When the prediction working condition changes, the TL is used to fine-tune the network with the target domain data. In addition, adaptive batch normalization (AdaBN) is also a domain adaptation algorithm. Initially, the batch normalization (BN) [8] is used to help SGD optimization by adjusting the distribution of training data. Later, it was found that the data distribution can be adjusted by BN to reduce the difference between the target domain and the source domain, and then achieve domain adaptation.

This paper combines AdaBN and DCNN to develop a RUL prediction model that can predict to different DDs. Meanwhile, a sliding time window (TW) and the piecewise RUL function are used to improve the capacity of prediction model. The rest of the paper is organized as follows: In Section 2, the related knowledge is introduced. In Section 3, the proposed method is given. In Section 4, the turbofan engine validation dataset carried out using C-MAPSS is described. In Section 5, the validation results and discussion is presented. Finally, Section 6 gives a conclusion of the paper.

II. Related Knowledge

A. *Convolutional Neural Network (CNN)*

The proposed of the convolutional neural network (CNN) is inspired by the visual nerves of animals. The CNN are widely used, especially in image processing and speech recognition. It can use 2-dimensional (2D) multi-channel data as input, which can avoid the impact of feature extraction and data reconstruction. The kernels of CNN with a same weight matrix that makes it closer to the principle of biological neural networks. Moreover, it can reduce the complexity of the network and the training parameters. The convolutional layer is the core of CNN, including convolution operation and activation operation.

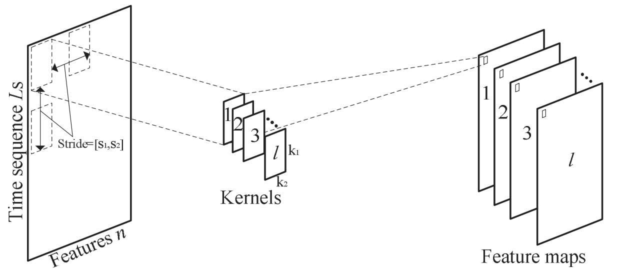

Figure 1. Operations of 2D-convolution

Fig. 1 shows the 2D convolution operation. The vertical direction of the input data is the length of the time sequence L_s, and the horizontal direction is n features. The number of convolution kernels is l, the size is [k$_1$, k$_2$], and the stride is set to [s$_1$, s$_2$]. When the input data dimension can be divisible by the stride of convolution kernel, the size of obtained feature map is [L_S/s$_1$, n/s$_2$]. Otherwise, the padding process will be performed, and the padding mode include 'same padding', 'zero padding' and so on.

The operations of convolution and activation can expressed by two equations as following:

$$z_{ij}^{l} = \mathrm{sum}(\mathbf{k}_l \otimes \mathbf{x}_{ij}) + b_l \tag{1}$$

$$\mathbf{Y}^l = \varphi(\mathbf{Z}^l) \tag{2}$$

where $\otimes$ represents the matrix multiplication, Z_{ij}^{l} represents the output of convolution operation, l is the number of feature maps, i and j represent the stride in the directions horizontal and vertical, sum (*) is an operation that adds all of the elements in matrix *, $\mathbf{k}_l$ is l-th the kernel matrix, $\mathbf{x}_{ij}$ represents the filter matrix, b_l represents the bias term, and $\varphi(\ )$ represents the activation function.

B. *Adaptive Batch Normalization (AdaBN)*

Initially, the batch normalization (BN) algorithm was proposed to speed up training and improve network performance by solving internal covariate shift problem. And the specific operation of the BN is to normalize the data in the same batch to achieve the purpose of adjusting the data distribution. The BN layer is usually added before the active layer after the convolutional layer or fully connected layer. Supposing add a BN layer after the convolutional layer, the input dimension of the BN layer is $[m_1, m_2, c]\times b_s$, which m_1 and m_2 are the two dimensions of the feature map, c is the number of channel, b_s is the batchsize. The normalization process can be implemented by (3), and Equation (4) is applied to restore the nonlinearity of the sample.

$$\hat{\mathbf{x}}_j^c = \frac{\mathbf{x}_j^c - \mathrm{E}[\mathbf{x}^c]}{\sqrt{\mathrm{Var}[\mathbf{x}^c] + \varepsilon}} \tag{3}$$

$$\mathbf{y}_j^c = \gamma^c \cdot \hat{\mathbf{x}}_j^c - \beta^c \tag{4}$$

where $\mathbf{x}_j^c$ is j-th (j=1,…, b_s) input in a batch of channel c, E[$\mathbf{x}^c$] and Var[$\mathbf{x}^c$] are the mean and variance of all element in channel c, ε is a very small positive number, scale factor γ^c, and offset β^c.

It is worth mentioned that the network is trained in batches, where E[$\mathbf{x}^c$] and Var[$\mathbf{x}^c$] are the average and variance of a batch. Scale factor γ^c and offset β^c are also updated during network training. When predicting with the trained network, E[$\mathbf{x}^c$] and Var[$\mathbf{x}^c$] in (3) should be replaced by the mean and variance of all training data.

When the target domain data is input to the network trained by the data from source domain, the E[$\mathbf{x}^c$] and Var[$\mathbf{x}^c$] are first replaced by target domain Z as shown in (5), and then the parameters γ_z and β_z are corrected by the target domain data, as shown in (6).

$$\tilde{\mathbf{x}}_j = \frac{\mathbf{x}_j - \mathrm{E}[\mathbf{z}]}{\sqrt{\mathrm{Var}[\mathbf{z}] + \varepsilon}} \tag{5}$$

$$\mathbf{y}_j = \gamma_z \cdot \tilde{\mathbf{x}}_j - \beta_z \tag{6}$$

where the E[**z**]and Var[**z**] are the mean and variance of all element in target domain, γ_z and β_z corrected parameters for target domain.

III. The Proposed Method

This paper proposes a novel RUL prediction model for adaptive DDs. The framework of the prediction model is divided into two parts as shown in Fig. 2. When the training data and test data are from the same DD, only Part A works in the model framework. Conversely, when the test domain is different from the training domain, the Part B in the model will be used to fine tune the trained model.

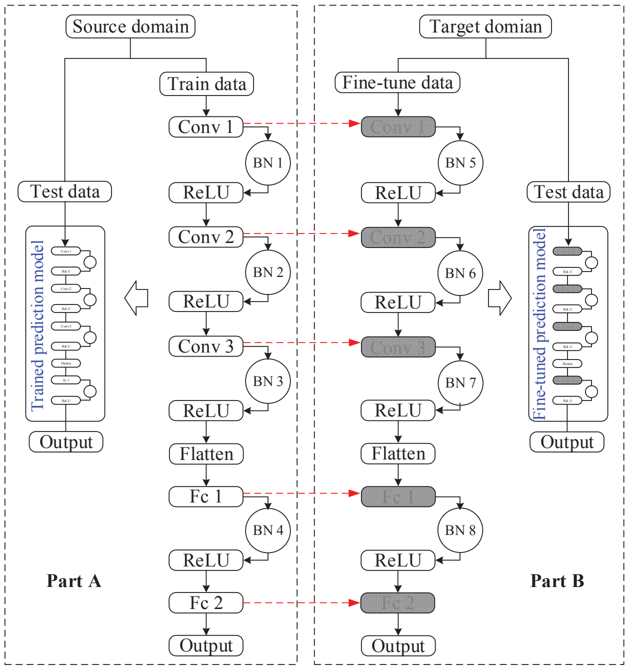

Figure 2. Framework of proposed prediction method

A. Part A

Part A in the framework works when trained and tested with data from same DD. The model consists of three convolutional (Conv) layers, a flatten layer, and two fully connected (Fc) layers. The convolutional layer and the fully connected layer are followed by a BN layer to adjust the distribution of the output data. The training process of the network is the parameter adjustment process. The parameters that need to be saved and used during testing are $\mathbf{K}_l$ and b_l of the convolution layer, γ and β of the BN layer, weight $\mathbf{W}_{fc}$ and bias $\mathbf{b}_{fc}$ of fully connected layer. The E[**x**] and Var[**x**] of each batch of the BN layer are saved in the training, and the average of all batches is used in testing.

B. Part B

Part B of the model will works when the trained model is used to predict data under other conditions. The fine-tuning process is as shown in the Fig. 2. All parameters of the convolutional layer and fully connected layer do not need to be changed. Only the γ and β of the BN layer need to be retrained. And the E[**z**] and Var[**z**] of target domain are used in the fine-tuned model.

IV. Experimental

The simulated turbofan engine run to failure data is used to validate the proposed prediction model. The simulated data produced using the simulation program C-MAPSS developed by NASA. This section will provide a detailed description of C-MAPSS dataset, data processing, and model evaluation method.

A. Turbofan Engine Simulated Validation Dataset

As shown in Fig. 1, the structure of the simulation engine is mainly composed of five main parts: fan, low pressure compressor (LPC), high pressure compressor (HPC), combustor, high pressure turbine (HPT), and low pressure turbine (LPT).

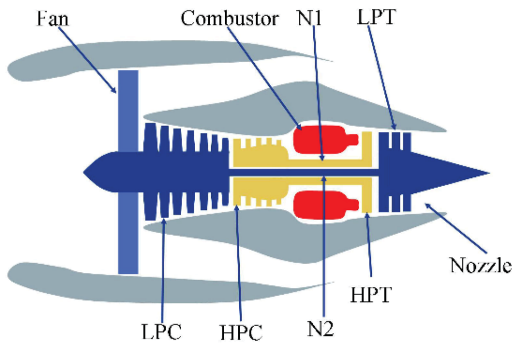

Figure 3. A simplified diagram of the simulation engine in C-MAPSS [3]

The CMAPSS simulation program takes 14 parameters that affect engine degradation as input, and 21 outputs represent states of engine such as: total temperature at fan inlet, pressure at fan inlet, and physical fan speed. The CMAPSS simulated dataset contains 4 sub-datasets, and the specific description as shown in Table I.

TABLE I. Description of the Turbofan Engine Dataset

Dataset Contents	**Sub-datasets**			
	FD001	***FD002***	***FD003***	***FD004***
Engine units in training set	100	260	100	249
Engine units in testing set	100	259	100	248
Max/min cycles in training set	362/128	378/128	525/145	543/128
Max/min cycles in testing set	303/31	367/21	475/38	486/19
Operating conditions	1	6	1	6
Fault types	1	1	2	2
TW length	30	21	36	18
Training samples	17731	48558	21220	56815
Testing samples	100	259	100	248

B. Data Preparation

In this paper, 14 sensors signal with regular trend from all of the 21 outputs of C-MAPSS simulation program was used as the input of prediction model. And then, a sliding time window (TW) was first performed to cut a long time series into several short time series, as shown in Fig.2. The obtained short time series length is equal to the time window length L_{TW}, and the number of short sequence is m-L_{TW}+1.

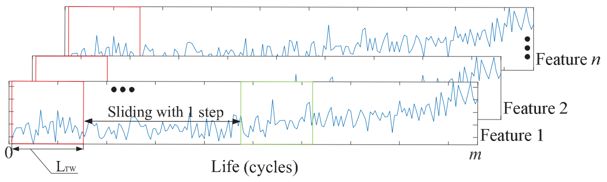

Figure 4. Operation of sliding time window

Because the feature values have different range, the short sequence data extracted by the sliding TW needs to be normalized. This paper uses the z-score normalization to process the raw data, and the mathematical implementation is given in (7).

$$y_i^n = (x_i^n - \bar{x}^n) / \sigma^n \tag{7}$$

where x_i^n is the i-th value of feature n, $\bar{x}^n$ is the average of feature n, σ^n is the standard deviation of feature n, and y_i^n is the normalized data.

C. Piece-wise Linear RUL Function

It is generally believed that the machine's RUL linearly reduced with time. However, this does not match the actual degradation process of the machine. The performance degradation of the machine in the early stage is not obvious, so it can be assumed that the machine RUL has not changed. As shown in Fig. 5, the linear RUL function is improved to a piece-wise linear function based on the trend of the signal.

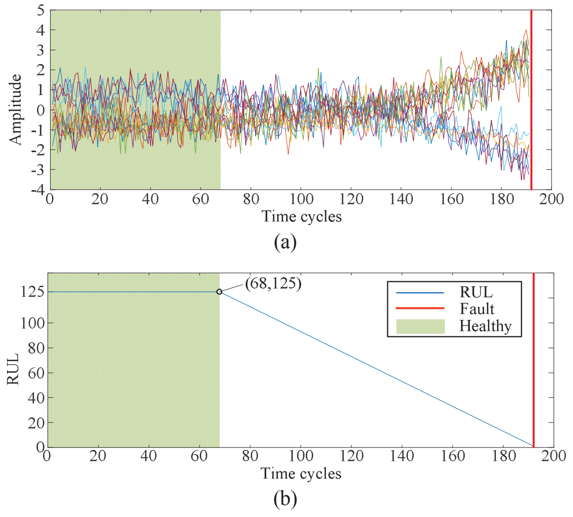

Figure 5. Engine unit 1 of FD001 dataset: (a) normalized 14 signals and (b) piecewise linear RUL function

D. Model Evaluation

When a trained prediction model was used to predict the RUL of machine, the prediction error is define as h ($h = \mathrm{RUL}^{\mathrm{predict}} - \mathrm{RUL}^{\mathrm{actual}}$). In order to make an overall evaluation of the predictive model, two evaluation methods are presented below:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \cdot \sum_{j=1}^{N} h_j^2} \tag{8}$$

$$\mathrm{score} = \sum_{j=1}^{N} s_j, s_j = \begin{cases} e^{-\frac{h_j}{13}} - 1, & h_j < 0 \\ e^{\frac{h_j}{10}} - 1, & h_j \geq 0 \end{cases} \tag{9}$$

where h_j is the prediction error of j-th engine units, N is number of the engine units in testing set.

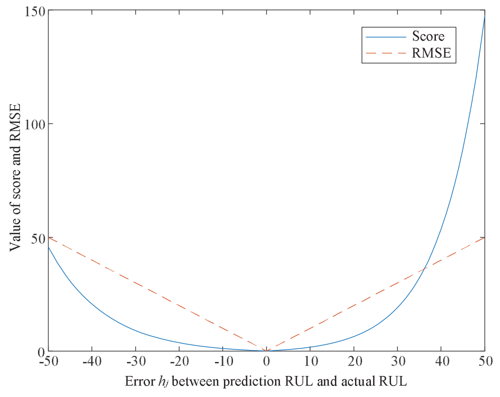

Figure 6. Graph of two evaluation methods

The graphs of the two evaluation methods are shown in Fig. 6. It can be known that the closer the prediction error to 0, the lower the value of RMSE and score. Comparing the two evaluation methods, it can be found that the RMSE has the same punishment for early and late prediction, and the Score has more penalized for late evaluation.

V. Results and Analysis

This section presents the results and analysis of using the C-MAPSS simulation turbofan engine data validate the proposed prediction model. It can be seen from Table I that the turbofan engine dataset contains 4 sub-datasets. The main difference between the 4 sub-datasets is the simulated operating conditions and the fault modes, i.e., the DD is different. The validation can be divided into two parts: 1) the data used for training and test has the same DD, 2) training data and test data has different DD.

Table II shows the structure and parameters (Para.) of the proposed method. The column information of the Table II includes: *Layers*-the layer information of the model, *State*-the data state of each layer (the convolution kernel size and stride), *Learnable*-the parameters to be learned in each layer, *Para.*-the total number parameters in each layer, *TL*-the number of retrained parameters by transfer learning, and *Ada.*-the number of retrained parameters by proposed AdaBN method. It can be seen from the table that when the data condition of target domain changes, the amount of parameters that need to be fine-tuned by the AdaBN method is much smaller than retraining a new network or the conventional transfer learning (TL) algorithm. Therefore, the proposed method can achieve self-adjustment faster.

TABLE II. STRUCTURE AND PARAMETERS OF THE PROPOSED PREDICTION MODEL

Layers	State	Learnable	Para.	Retrained Para.	
				TL	*Ada.*
Input	14×30×1	-	0	0	0
Cov 1	[1,6], [1,2]	Weights 1×6×1×3	21	0	0
	14×15×3	Bias 1×1×3			
BN 1	14×15×3	Offset 1×1×3	6	0	6
		Scale 1×1×3			
ReLU	14×15×3	-	0	0	0
Cov 2	[1,4], [1,2]	Weights 1×4×3×3	39	0	0
	14×8×3	Bias 1×1×3			
BN 2	14×8×3	Offset 1×1×3	6	0	6
		Scale 1×1×3			
ReLU	14×8×3	-	0	0	0
Cov 3	[1,2], [1,2]	Weights 1×2×3×3	21	0	0
	14×4×3	Bias 1×1×3			
BN 3	14×4×3	Offset 1×1×3	6	0	6
		Scale 1×1×3			
ReLU	14×4×3	-	0	0	0
Flatten	1×1×168	-	0	0	0
Fc 1	1×1×30	Weights 30×168	5070	5070	0
		Bias 30×1			
BN 4	1×1×30	Offset 1×1×30	60	0	60
		Scale 1×1×30			
ReLU	1×1×30	-	0	0	0
Fc 1	1×1×1	Weights 1×30	31	31	0
		Bias 1×1			
Output	1×1×1	-	-	-	-
Total Parameters			5260	5101	**78**

A. *The target domain and the source domain have the same data condition*

When the data condition of the target domain and the source domain are the same, only part A of the model runs. The sliding TW with one step size was used to process the raw signals. In order to validate the model with all the data in the testing set, the signals are processed with different sliding window lengths in 4 sub-datasets. The obtained data size is [14, L_{TW}], and the RUL corresponds to the last moment extracted by the TW. The specific structure and parameter of the proposed prediction model are shown in Table II. The 'rmsprop' was selected as the optimizer, the learning rate and mini-batch are set to 0.005 and 400 respectively. The results of the RUL prediction methods in the past 4 years that also used C-MAPSS dataset for validation are presented in Tables III and IV.

TABLE III. COMPARISON OF DIFFERENT METHODS RMSE RESULTS

Methods	Years	RMSE			
		FD001	*FD002*	*FD003*	*FD004*
MLP [6]	2016	37.56	80.03	37.39	77.37
SVR [6]	2016	20.96	42.0	21.05	45.35
RVR [6]	2016	23.80	31.30	22.37	34.34
CNN [6]	2016	18.45	30.29	19.82	29.16
LSTM [9]	2017	16.14	24.49	16.18	28.17
ELM [10]	2017	17.27	37.28	18.47	30.96
DBN [10]	2017	15.21	27.12	14.71	29.88
MODBNE [10]	2017	15.04	25.05	**12.51**	28.66
BLSTM [7]	2018	14.26	21.7	16.33	25.9
Proposed method	2019	**13.17**	**20.87**	14.97	**24.57**

TABLE IV. COMPARISON OF DIFFERENT METHODS SCORE RESULTS

Methods	Score			
	FD001	*FD002*	*FD003*	*FD004*
MLP [6]	1.80×10^4	7.80×10^6	1.74×10^4	5.62×10^6
SVR [6]	1.38×10^3	5.90×10^5	1.60×10^3	3.71×10^5
RVR [6]	1.50×10^3	1.74×10^4	1.43×10^3	2.65×10^4
CNN [6]	1.29×10^3	1.36×10^4	1.60×10^3	7.89×10^3
LSTM [9]	3.38×10^2	4.45×10^3	8.52×10^2	5.55×10^3
ELM [10]	5.23×10^2	4.98×10^5	5.74×10^2	1.21×10^5
DBN [10]	4.18×10^2	9.03×10^3	4.42×10^2	7.95×10^3
MODBNE [10]	3.34×10^2	5.59×10^3	$\mathbf{4.22\times10^2}$	6.56×10^3
Proposed method	$\mathbf{2.79\times10^2}$	$\mathbf{2.02\times10^3}$	8.17×10^2	$\mathbf{3.69\times10^3}$

The RMSE and score performance of different methods are shown in Tables III and IV. Comparing the prediction results of different methods, it can be found that the methods proposed in this paper have the best diagnosis results except for the sub-dataset FD003.

B. *The data condition of the target domain is different from the source domain*

When the data condition of the target domain and the source domain are different, it is necessary to perform the part B to fine-tune the model built by Part A. The cross prediction results for different DD are shown in Tables V and VI. In order to achieve cross prediction between different sub-datasets, the length of the TW for the 4 sub-datasets is set to 18. This results in a certain change in the predicted results compared to the RMSE and SCORE in Tables III and IV. The Tables V and VI are the RMSE and SCORE of prediction results with and without the AdaBN algorithm. Since the sub-datasets FD001 and FD003 are similar, the cross prediction error between the two datasets is smaller as shown in Table V. Similarly, the prediction error between FD002 and FD004 is also smaller due to similar data condition.

TABLE V. THE CROSS PREDICTION RMSE WITH DIFFERENT DD

RMSE			Source domain			
			FD001	*FD002*	*FD003*	*FD004*
Target domain	*FD001*	*Without*	16.0425	516.7504	24.1745	224.5615
		AdaBN		20.6615	19.8594	21.7056
	FD002	*Without*	51.7574	20.5952	52.8622	26.5664
		AdaBN	30.5424		40.3857	24.0074
	FD003	*Without*	23.1527	395.3741	18.078	382.1581
		AdaBN	18.4032	29.2461		35.0254
	FD004	*Without*	47.1612	30.2956	49.9685	26.2416
		AdaBN	36.8525	29.6395	39.9672	

TABLE VI. THE CROSS PREDICTION SCORE WITH DIFFERENT DD

Score		Source domain			
		FD001	*FD002*	*FD003*	*FD004*
Target domain	*FD001*	5.87×10^2	Inf	4.55×10^3	5.64×10^{28}
			1.01×10^3	1.92×10^3	1.58×10^3
	FD002	1.38×10^6	2.35×10^3	1.11×10^6	1.15×10^4
		1.34×10^4		7.85×10^4	4.76×10^3
	FD003	1.84×10^3	4.5×10^{37}	1.73×10^3	Inf
		1.68×10^3	7.0×10^3		1.77×10^4
	FD004	3.65×10^5	8.88×10^3	8.16×10^5	6.26×10^3
		1.0×10^5	1.41×10^4	1.34×10^5	

Inf represents a number close to infinity

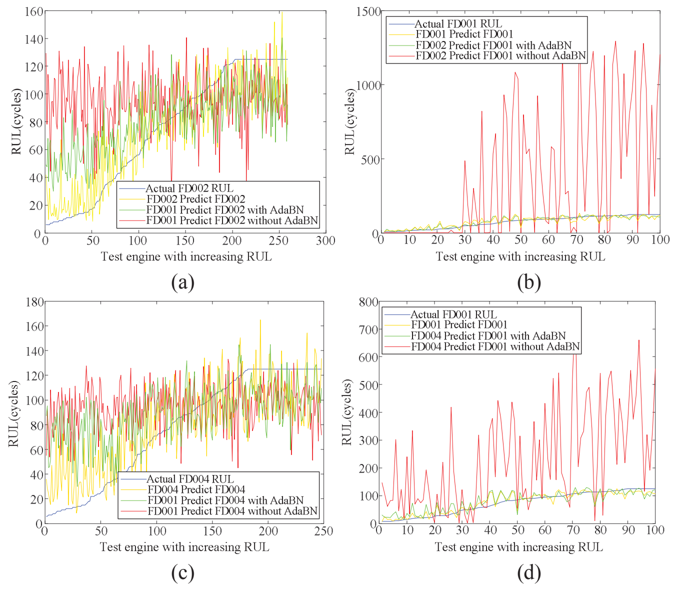

Figure 7. Comparison of RUL prediction results in different methods: (a) FD001→FD002, (b) FD002→FD001, (c) FD001→FD004, and (d) FD004→FD001.

Fig. 7 presents the RUL prediction results for the different methods in 4 cross-validation case. It can be seen that the proposed prediction model can significantly improve the prediction results under different DD, especially in the cases FD002→FD001 and FD004→FD001. The operating conditions and failure types in the 3 sub-datasets of FD001, FD002, and FD004 can be seen in Table I. The number of fault types of FD001 and FD002 is the same, but the FD002 dataset contains more working conditions. Compared with FD001 dataset, the FD004 dataset contains more fault types and operating conditions. Observing the 4 graphs in Fig. 7, the predicted results of 7b and 7d are the best, followed by 7c, 7a. It is concluded that models trained with complex data conditions are more easily adapted to simple data condition.

VI. Conclusion

In view of the fact that the existing prediction models cannot adapt to multiple DD, this paper developed a domain adaptation prediction model by adding the BN layer in CNN. And the TW and piecewise RUL function were used to facilitate the development of the model. The proposed RUL prediction model was validate by C-MAPSS turbofan engine dataset. It can be concluded that the proposed method not only has good predictive ability under the same DD, but also can improve the prediction power by fine-tuning parameters under different DDs. Especially, when using the prediction model trained by multi-working conditions to predict the data contains less working conditions, the proposed prediction model can greatly improves the prediction accuracy after simple fine-tuning, as in cases FD002→FD001 and FD004→FD001. Compared with other methods, the proposed method requires fewer parameters to be fine-tuned when predicting other DD, which means that it can adapt to the new DD more quickly.

References

[1] L. Ren, L. Zhao, S. Hong, S. Zhao, H. Wang, and L. Zhang, "Remaining Useful Life Prediction for Lithium-Ion Battery: A Deep Learning Approach," IEEE Access, vol. 6, pp. 50587–50598, 2018.

[2] J. Liu and Z. Chen, "Remaining Useful Life Prediction of Lithium-ion Batteries Based on Health Indicator and Gaussian Process Regression Model," IEEE Access, pp. 1–1, 2019.

[3] 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, May 2017.

[4] H. Wang, Y. Zhao, and X. Ma, "Remaining Useful Life Prediction Using a Novel Two-Stage Wiener Process With Stage Correlation," IEEE Access, vol. 6, pp. 65227–65238, 2018.

[5] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," Reliabil. Eng. Syst. Saf., vol. 172, pp. 1–11, Apr. 2018.

[6] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life," in Database Systems for Advanced Applications, vol. 9642, S. B. Navathe, W. Wu, S. Shekhar, X. Du, X. S. Wang, and H. Xiong, Eds. Cham: Springer International Publishing, 2016, pp. 214–228.

[7] A. Zhang, H. Wang, S. Li, Y. Cui, Z. Liu, G. Yang, and J. Hu, "Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation," Appl. Sci., vol. 8, no. 12, p. 2416, Nov. 2018.

[8] D. Xiao, Y. Huang, C. Qin, H. Shi, and Y. Li, "Fault Diagnosis of Induction Motors Using Recurrence Quantification Analysis and LSTM with Weighted BN," Shock Vib., vol. 2019, pp. 1–14, Jan. 2019.

[9] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Dallas, TX, USA, 2017, pp. 88–95.

[10] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, Oct. 2017.