Adversarial Transfer Learning for Machine Remaining Useful Life Prediction

Mohamed Ragab*†, Zhenghua Chen†, Min Wu† , Chee Keong Kwoh* and Xiaoli Li †
*School of Computer Science and Engineering, Nanyang Technolgical University
†Institute for Infocomm Research (I2R),A*STAR

***Abstract*—Remaining useful life (RUL) prediction is a key task for realizing predictive maintenance for industrial machines/assets. Accurate RUL prediction enables prior maintenance scheduling that can reduce downtime, reduce maintenance costs, and increase machine availability. Data-driven approaches have a widely acclaimed performance on RUL prediction of industrial machines. Usually, they assume that data used in training and testing phases are drawn from the same distribution. However, machines may work under different conditions (i.e., data distribution) for training and testing phases. As a result, the model performing well during training can deteriorate significantly during testing. Naive recollection and re-annotation of data for each new working condition can be very expensive and obviously not a viable solution. To alleviate this problem, we rely on a transfer learning approach called domain adaptation to transfer the knowledge learned from one *labelled* operating condition (source domain) to another operating condition (target domain) *without labels*. Particularly, we propose a novel adversarial domain adaption approach for remaining useful life prediction, named ADARUL, which can work on the data from different working conditions or different fault modes. This approach is built on top of a bidirectional deep long short-term memory (LSTM) network that can model the temporal dependency and extract representative features. Moreover, it derives invariant representation among the working conditions by removing the domain-specific information while keeping the task-specific information. We have conducted comprehensive experiments among four different datasets of turbofan engines. The experiments show that our proposed method significantly outperforms the state-of-the-art methods..**

***Index Terms*—Remaining useful life, Domain Adaptation, Transfer Learning, Deep Learning**

I. Introduction

Predictive maintenance or condition-based monitoring (CBM) enables industries to have prior maintenance planning and thus prevent catastrophic failures. Realizing predictive maintenance, can lessen downtime, reduce maintenance expenses, increase reliability, and increase machine availability. Remaining useful life (RUL) prediction aims to forecast the exact failure time of an industrial asset, which is a key task in predictive maintenance. Three main approaches have been developed to estimate the RUL of industrial machines, namely, model-driven approaches [1], data-driven approaches [2], and hybrid approaches [3]. Particularly, model-driven approaches rely on strong domain expertise to mathematically model the physics of deterioration process and predict the RUL. But, with the evolving complexity of industrial machines, this approach can be very challenging even for experts to accurately model the deterioration process of advanced industrial machines.

Differently, data-driven approaches rely on the huge data availability in the digital era to model the deterioration process. Data-driven approaches include traditional machine learning approaches and deep learning based approaches. In particular, various traditional machine learning approaches have been utilized to estimate the RUL of industrial machines including, support vector regressors [4], Markovian models [5], and shallow neural networks [6]. However, these approaches suffer from extensive feature engineering that requires advanced domain knowledge. Recently, deep learning, with its automatic ability to extract salient features, has been applied for the fault prognosis problems [7]. For example, Li *et al.,* developed a deep CNN with 1-D kernels to accurately estimate the RUL of machine [8]. Zhu *et al.,* predicted the deterioration pattern of bearings using CNN with a multi-scale features [9]. However, CNN based approaches cannot explicitly model the temporal dependency among the data. Recurrent neural networks (RNN) have showed great potential in modeling the dynamic systems. Long short-term memory (LSTM), as a strong variant of RNN for modelling long sequences [10], has been exploited extensively for RUL estimation problems. In [11], multi-layer LSTM network is employed to extract features and predict the RUL of turbofan engines. Miao *et el.* employed LSTM networks to jointly predict the RUL and evaluate the degradation pattern [12]. Huang *et al.,* proposed bidirectional LSTM coupled with auxiliary inputs to estimate the RUL under varying working conditions [13]. In [14], LSTM has been integrated with CNN to extract more representative features.

Nevertheless, all the above approaches work under the assumption training phase and deployment phase are under the same working environment, which does not hold in many practical scenarios. We consider two working environments A and B each with certain parameters (e.g., pressure, temperature, and rotational speed) from different distributions. The model trained on working condition A usually fails to generalize to working condition B. In reality, this situation is very likely to happen due to the varying environments. Naïve solution is to re-annotate the data for each new working environment and train new model independently. But, annotating the data for each new working condition is labour-intensive and unscalable solution. To better handle this issue, there is a need for a model that can handle new working environment with no labels available.

Domain adaptation (DA), which is a subgroup of transfer learning that leverages labelled source domain data to

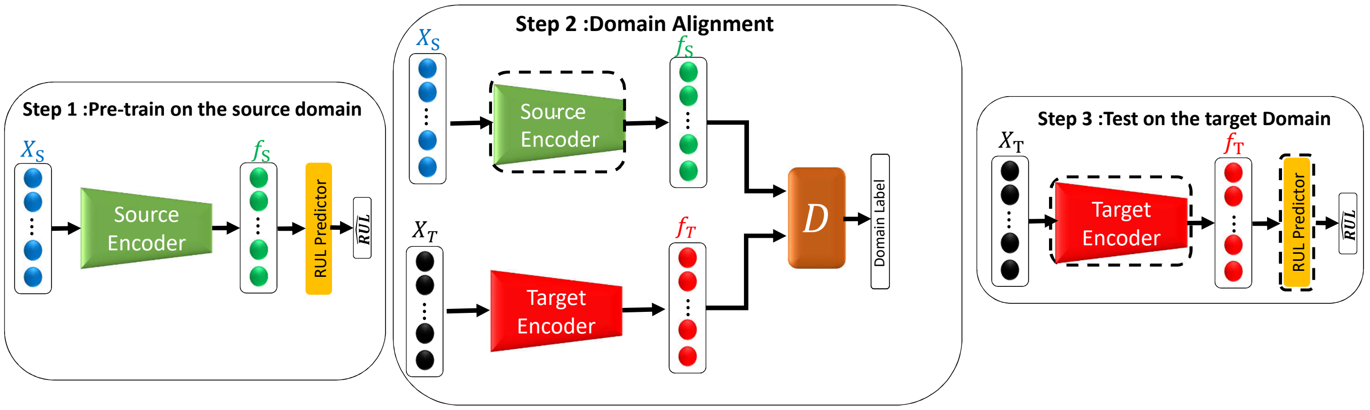

Fig. 1: Adversarial domain adaptation approach for RUL prediction.

generalize well for unlabelled target domain with a different distribution. Domain Adaptation has been applied extensively for image classification problems [15]. Very recently, it has been applied for fault diagnosis problems [16]–[20]. However, little attention has been paid to domain adaptation for prognostic problems such as remaining useful life estimation. Only few works have been developed recently, for instance, Fan *et al.*, employed group deviation detection method called self-organized models (COSMO) to create transferable features and used Random forest to predict the RUL labels [19]. Costa *et al.* integrated LSTM model with domain classification loss to reduce the deviation between source and target distributions [21].

In this paper, we propose an unsupervised adversarial domain adaptation method named ADARUL to efficiently transfer knowledge from labeled working condition (source domain) to a new unlabelled working condition (target domain). Inspired by the successful work done by Tzeng *et el.* [22] for image classification from different domains, ADARUL is an adversarial domain adaptation approach for time series regression problems. It derives domain invariant representation for the unlabelled target domain based on a two-player adversarial game. In particular, ADARUL aims to find a domain invariant feature representation such that the domain discriminator network can no longer distinguish between the source domain and the target domain. As such, the predictor network trained on the source domain can generalize well to the target domain network. The contributions of this work are summarized as follows.

- We propose a novel adversarial domain adaptation approach for remaining useful life prediction, which can handle domain shift problem (e.g., different working conditions).
- Different from existing approaches, our proposed approach is an end-to-end model that can generalize to new working environment without any labels.
- Extensive experiments have been conducted and experimental results demonstrate that our proposed approach significantly outperform the state-of-the-art approaches.

II. Methodology

In this section, we introduce our proposed ADARUL for RUL prediction in details.

A. Overview of ADARUL

In unsupervised domain adaption settings, we have a labeled source domain $\mathcal{D}_s = \{X_s^i, y_s^i\}_{i=1}^{n_s}$, and unlabeled target domain $\mathcal{D}_t = \{X_t^j\}_{j=1}^{n_t}$, where n_s, n_t are the number of source and target samples respectively. The source domain and target domain have different marginal distribution $P_s(X) \neq P_t(X)$ due to the variation of working environments. Our task in this paper is to build a RUL prediction model based on the labelled source domain, which can perform well on the unlabelled target domain.

As shown in Fig. 1, our proposed ADARUL has three main steps. The first step is the supervised training on the source domain using the available labels. The second step is to train the target feature encoder to produce the target features that are similar to the source. In particular, the target feature encoder is initialized by the weights of the source encoder. We then train the discriminator against the target feature extractor adversarially, to mitigate the distribution shift between the source and target. Last step is to integrate the target encoder with source RUL predictor to predict the RUL labels for the target domain. Note that, the model parameters with dashed box is fixed and those without dash box are trained in each step. In the following subsections we will provide detailed explanations for each step.

B. Supervised pre-training on the source domain

1) LSTM for Feature Extraction: Recurrent neural networks (RNN) have wide acclaimed performance in modeling the system dynamics. Long short-term memory (LSTM) is a special type of RNNs that can model longer dependencies without vanishing gradient [10]. To model the temporal dependency of the multivariate time series data, we develop a 3-layer bidirectional LSTM model to extract representative features from the input sensor readings, as shown in Fig. 2. Given an input sample $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) \in \mathbb{R}^{n \times T}$, where n

is number of sensors and T is the total number of time steps for each sample. The vector $\mathbf{x}_t \in \mathbb{R}^n$ is passed as input to the LSTM network for each time step t, where $(1 \leq t \leq T)$. Then, the network will process this input to produce a sequence of hidden states $(\mathbf{h}_1^j, \ldots, \mathbf{h}_T^j)$, where $\mathbf{h}^j$ denotes the output hidden states of j^{th} layer, $(1 \leq j \leq M)$. M is total number of LSTM layers and it is set as 3 in our experiments. Subsequently, the output hidden states are passed to the next LSTM layer until reaching the last layer. The last hidden state of the last layer $\mathbf{h}_T^M$ summarizes all the information from the input sequence among all the layers, which can be considered as our feature representation for the whole input sequence. The following equations illustrate the working principles of single-layer LSTM at each time step t for the clarity and simplification and the same process can be applied for the other LSTM layers. Given the input vector $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{(t-1)}$, the equations of can be as follows:

$$\mathbf{b}_t = \sigma(V_b \mathbf{x}_t + W_b \mathbf{h}_{t-1} + \mathbf{b}_b), \tag{1}$$
$$\mathbf{e}_t = \sigma(V_e \mathbf{x}_t + W_e \mathbf{h}_{t-1} + \mathbf{b}_e), \tag{2}$$
$$\mathbf{i}_t = \sigma(V_i \mathbf{x}_t + W_i \mathbf{h}_{t-1} + \mathbf{b}_i), \tag{3}$$
$$\mathbf{g}_t = tanh(V_g \mathbf{x}_t + W_g \mathbf{h}_t - 1 + \mathbf{b}_g), \tag{4}$$
$$\mathbf{c}_t = \mathbf{e}_t \odot \mathbf{c}_{t-1} + \mathbf{b}_t \odot \mathbf{g}_t, \tag{5}$$
$$\mathbf{h}_t = \mathbf{i}_t \odot tanh(\mathbf{c}_t), \tag{6}$$

where $\odot$ is an operator for element-wise multiplication, σ represents a *sigmoid* function to realize non linearity. Let the dimension of the hidden states as d, $V_* \in \mathbb{R}^{n \times d}$ (i.e., V_b, V_e, V_f and V_g), are the parameters that matrices that transform the data from n input dimension to d hidden dimension. Similarly, $W_* \in \mathbb{R}^{d \times d}$ (i.e., W_b, W_e, W_f and W_g) are parameter matrices to obtain the current hidden state from the previous hidden state.

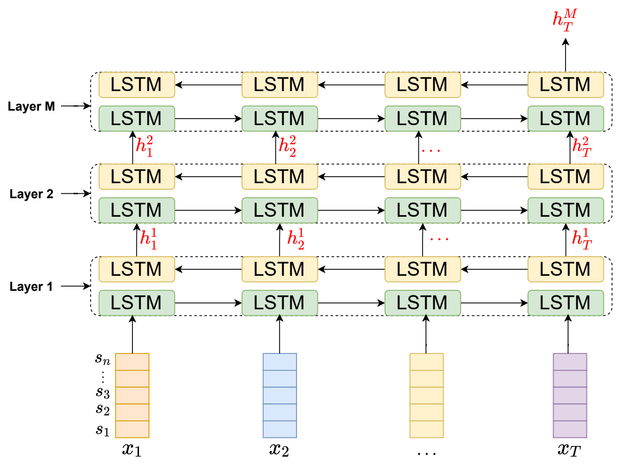

Fig. 2: Multi-layer bidirectional LSTM as feature encoder.

2) RUL Predictor Network: After obtaining the feature vectors for input sequences, the RUL predictor network harnesses these features to predict the RUL labels. The RUL predictor is a 3-layer fully connected neural network $O : \mathbb{R}^{(d)} \rightarrow \mathbb{R}$ that maps the input features to the corresponding RUL label. The RUL Network and the LSTM feature extractor are trained jointly in an end-to-end manner to minimize the mean square error loss (MSE) between the predicted RUL and the true label. The full training loss can be formulated as follows:

$$L_{rul} = \frac{1}{N} \sum_{i=1}^{N} (O(E_s({X^i}_s)), y^i)^2 \tag{7}$$

where E_s represents the LSTM encoder model, O is the RUL predictor network, X_s^i is the input sample from the source domain, $O(E_s({X^i}_s))$ is the predicted RUL for X_s^i and y^i is the corresponding true label.

C. Unsupervised Domain Adaptation for the Target Domain

The model trained on the labeled source domain has the knowledge to predict the RUL accurately. To inherit this knowledge, we use the parameters of the source encoder E_s as initialization for the target encoder E_t. Usually, the initialized target encoder E_t performs poorly on the target domain data due to the significant distribution shift between source and target. To mitigate this domain shift problem, the target feature extractor is then adversarially trained against discriminator network to produce target features $\mathbf{f}_t$ that can be similar to the source features $\mathbf{f}_s$(i.e, h_T^M). More specifically, for each iteration, the discriminator network D is updated to classify well between the source features $\mathbf{f}_s$ and target features $\mathbf{f}_t$. We manually assign ground-truth labels of 1 and 0 for the source and target features respectively. Then the discriminator network is trained to minimize the binary cross entropy loss between the predicted labels and the true labels. This can be formulated as follows:

$$\begin{aligned} \min_{D} \mathcal{L}_D = &- \mathbb{E}_{X_s \sim P_s}\big[\log D(\mathbf{f}_s)\big] \\ &- \mathbb{E}_{X_t \sim P_t}\big[\log(1 - D(\mathbf{f}_t))\big]. \end{aligned} \tag{8}$$

where $\mathbf{f}_s = E_s(X_s)$ and $\mathbf{f}_t = E_t(X_t)$ are the source and target features, respectively.

Given the optimal discriminator, the target feature extractor is then updated to maximize the discriminator by producing target features $\mathbf{f}_t$ that are indistinguishable from the source features. In particular, the target encoder is trained using inverted labels for the target features (i.e., true label 1 instead of 0) such that discriminator network is fooled to classify the target features as a source features. This can be formalized as follows:

$$\max_{E_t} \mathcal{L}_E = -\mathbb{E}_{X_t \sim P_t}\big[\log(1 - D(\mathbf{f}_t))\big], \tag{9}$$

where D is the discriminator network and $f_t = E_t(X_t)$ is the target domain features.

The discriminator D and target encoder E_t are updated in adversarial manner using L_D and L_E respectively. The alignment between source and target features is achieved when the discriminator network can no longer distinguish between source and target features, demonstrating that the distribution discrepancy between the source and target features is already minimized.

D. Evaluate the Target Domain

After minimizing the discrepancy between the source and target features, the source RUL predictor O can be directly used to predict the RUL labels of the adapted target features $\mathbf{f}_t$. Given single input sample X_t^i, the RUL prediction of the target domain data can be formulated as follows:

$$\widehat{rul}_t = O(E_t(X_t^i)) \tag{10}$$

where $\widehat{rul}_t$ is the predicted RUL for the sample X_t^i in the target domain, O is the pre-trained RUL predictor network, and E_t is the target feature extractor.

III. Experiments and Results

A. Data Description

We used the C-MAPSS dataset for experimental evaluation, which describes the degradation process of turbofan engines as shown in Fig. 3. To monitor the engines under varying working conditions, 21 parameters of total 58 outputs are used, including multiple health states [23]. As shown in Table I, the dataset consists of four data subsets with different operating conditions and fault modes denoted as FD001, FD002, FD003, and FD004.

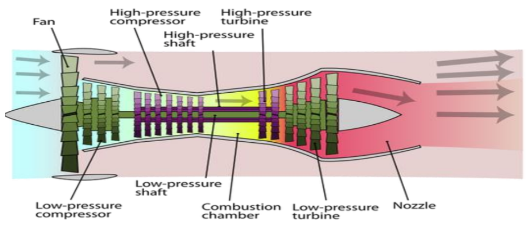

Fig. 3: Diagram of the engines in C-MAPSS data [23].

TABLE I: Properties of C-MAPSS Dataset

Dataset	FD001	FD002	FD003	FD004
# Training engines	100	260	100	249
# Testing engines	100	259	100	248
# Operating conditions	1	6	1	6
# Fault types	1	1	2	2

Particularly, some data subsets have single fault mode (e.g., FD001 and FD002) while other data subsets have double fault modes (e.g., FD003 and FD004). In addition, the data collected under different operating conditions, for instance, FD001 and FD003 have single operating condition while FD002 and FD004 have multiple operating conditions. These working conditions are comprised by the variation of three different parameters: altitude (0-42K ft.), mach number (0-0.84), and throttle resolver angle (TRA) (20-100), as shown in Fig. 4. As a result of these variant operating parameters, the model trained on data subsets such as FD001 fails to generalize to different data subsets (e.g., FD002, FD003, and FD004).

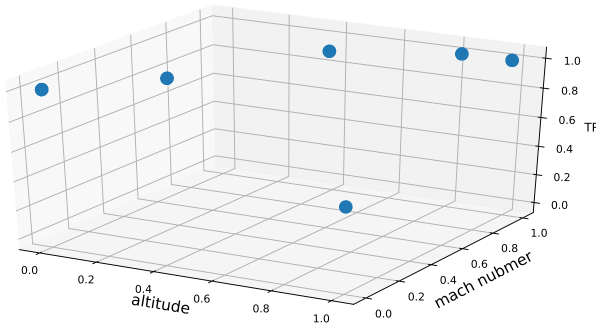

Fig. 4: Six operating conditions in FD002 dataset

B. Data Processing

The provided sensor readings represent full run-to-failure test for each engine, which can be very long sequence. Sliding window has been used to segment the long sequences into shorter samples to facilitate the model training. As shown in Fig. 5, T represents the total number of cycles, W is the window length, s is the shifting size, and n is the total number of sensors. In our case, the window length W is 30 and shifting size is 1.

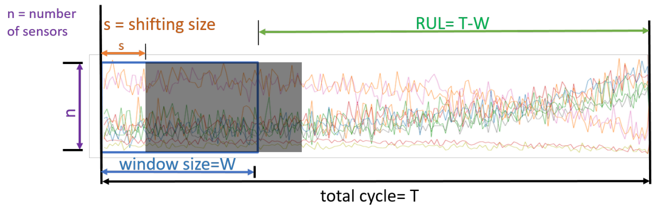

Fig. 5: Sliding window to generate data samples.

In addition, different sensor types are involved in current system (e.g.,speed, pressure, and temperature) which also exposed to different working conditions. As a result, the raw sensor readings are highly varying among each other, and the direct use of this data may obstruct performance of the machine learning models. To address this problem, we normalized the reading of sensor to be between $[0, 1]$ by using Min-Max normalization, as in [12], [13]. Furthermore, to handle the variability of working conditions, we grouped and normalized the sensor reading with respect to their corresponding working condition. Formally, the data points of sensor i under k-th operating condition is represented by vector V_{ik}, we compute the normalized vector $\hat{V}_{ik}$ as follows

$$\hat{V}_{ik} = \frac{V_{ik} - min(Q_{ik})}{max(Q_{ik}) - min(Q_{ik})}. \tag{11}$$

C. Sensor Selection

Different type of sensors are deployed in different locations inside each engine to measure different parameters (e.g., speed, pressure, temperature). To select the most informative sensors, we visualized the sensors of a randomly selected engine from run to failure. We visualized the sensor readings from datasets with single working condition (e.g., FD001) and multiple working conditions (e.g., FD002). To clearly show the trendy pattern of the sensor readings, moving average has been applied to smooth the sensor signals.

Fig 6 shows the sensor readings of a selected engine in FD001 dataset, while Fig 7 shows the sensory data of FD002 dataset. Clearly, there are some sensors that are non trendy during the whole run-to-failure test, which can deteriorate the model performance. In addition, for our domain adaptation problem, where we map the model source dataset (e.g., FD001) to target dataset (e.g., FD004). We selected the most common 14 informative sensors among the source and target. The selected sensors are S2, S3, S4, S7, S8, S9, S11, S12, S13, S14, S15, S17, S20 and S21.

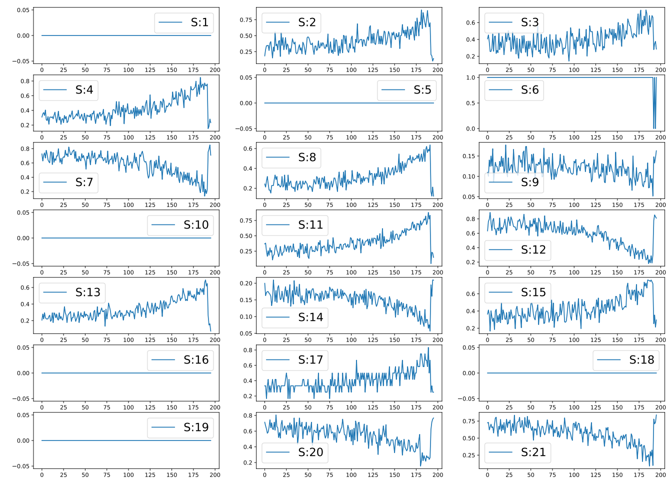

Fig. 6: The readings of 21 sensors for a randomly selected engine in FD001 dataset.

D. Experimental Settings

In this section, we provide detailed explanations of our technical details. Our ADARUL model is composed of three main components, namely LSTM feature extractor, Discriminator and RUL predictor network. The LSTM feature extractor is composed of 3 bidirectional layers, each with 32 hidden neurons. The discriminator network is a 3-layer fully connected network with input layer of 64 neurons, second layer of 32 neurons and last layer of a single neuron. The RUL predictor network has a similar architecture with an additional dropout layer to reduce overfitting (the dropout rate is set as 0.5). Hyper-parameters are so critical for the model performance, in our experiment, the learning rate is equal to $1e^{-4}$, batch size is equal to 10, the length of each training sample is 30, and only 20 epochs of training are required to align the two domains.

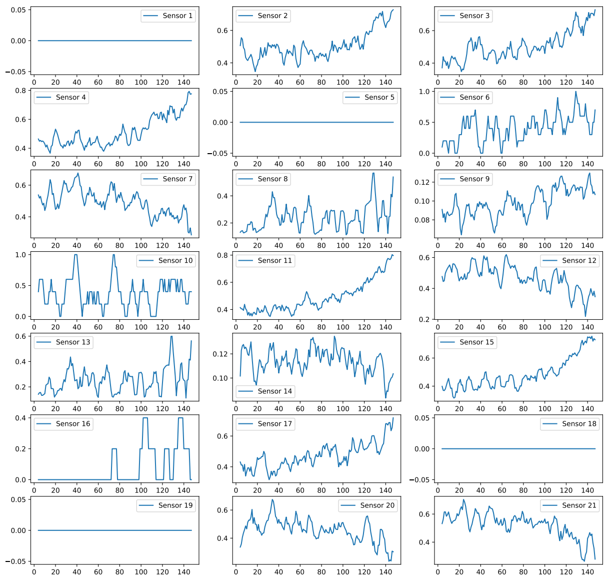

Fig. 7: The readings of 21 sensors for a randomly selected engine in FD002 dataset.

E. Experimental Results

In this section, we evaluate our ADARUL approach on on C-MAPSS dataset. To evaluate our cross domain performance performance, we train on one labeled dataset (e.g., FD001) and test (i.e., deploy) on a different unlabeled one (e.g., FD002). Note that we only use target domain labels to evaluate the performance on the target domain. In total, we have 12 cross domain scenarios across the different datasets. Different metrics are used to evaluate the model performance such as root mean square error (RMSE) and Score, following the previous studies in the literature [13], [21], [24]. Table II

TABLE II: Comparison between the proposed approach and source only in terms of Score and RMSE

Scenario	Score		RMSE	
	Source-Only	Proposed	Source-Only	Proposed
FD001–>FD002	5066	4426	20.77	19.87
FD001–>FD003	23762	10711	48.52	39.74
FD001–>FD004	12519	12369	34.15	31.78
FD002–>FD001	657	455	15.48	14.33
FD002–>FD003	3504	3449	33.99	32.6
FD002–>FD004	12715	12227	36.08	34.35
FD003–>FD001	34538	1391	33.03	19.97
FD003–>FD002	270078	20764	33.01	23.47
FD003–>FD004	113835	48748	27.11	26.33
FD004–>FD001	161724	121677	39.30	37.89
FD004–>FD002	96527	62215	31.31	28.77
FD004–>FD003	1861	537	19.26	14.13
Mean	61399	24914	31.00	26.94

TABLE III: RMSE comparison against state-of-the-art approaches.

Scenario	NN-TCA [21]	DNN-TCA [21]	NN-CORAL [21]	DNN-CORAL [21]	DNN-DANN [21]	CNN-DANN [21]	LSTM_DANN [21]	Proposed
FD001–>FD002	94.1± 1	90.0 ± 2.9	99.2 ± 3.6	77.5 ± 4.6	67.1 ± 24.1	50.3 ± 6.4	48.6 ± 6.8	**19.87 ± 0.27**
FD001–>FD003	120.0 ± 1.0	116.1 ± 1.0	60.0 ± 0.7	69.6 ± 5.2	64.2 ± 28.7	42.8 ± 3.3	45.9± 3.6	**39.74 ± 0.32**
FD001–>FD004	120.1 ± 1.0	113.8 ± 6.9	107.7 ± 2.8	84.6 ± 7.0	74.8 ± 32.5	87.0 ± 60.8	43.8 ± 4.1	**31.78 ± 0.36**
FD002–>FD001	94.7 ± 1.1	85.6 ± 5.5	77.9 ± 19	80.9 ± 9.4	52.1 ± 27.3	47.4 ± 16.3	28.1 ± 5.0	**14.33 ± 0.14**
FD002–>FD003	107.4 ± 3.7	111.5 ± 7.2	60.9 ± 15.1	79.8 ± 10.1	73.9 ± 46.4	40.8 ± 0.2	37.5 ± 1.5	**32.60 ± 0.39**
FD002–>FD004	93.5 ± 2.8	94.4 ± 6.7	37.5 ± 0.5	43.6 ± 3.6	26.3 ± 0.5	38.0 ± 7.3	**31.8 ± 1.6**	34.35 ± 0.67
FD003–>FD001	98.7± 0.4	90.5 ± 4.6	26.5 ± 0.5	26.5 ± 1.9	28.7 ± 10.1	33.8 ± 6.0	31.7 ± 9.4	**19.97 ± 0.45**
FD003–>FD002	90.5 ± 0.3	80.8 ± 4.3	113.2 ± 4.5	75.6 ± 9.5	77.2 ± 16.5	68.2 ± 15.2	44.6 ± 1.2	**23.47 ± 0.61**
FD003–>FD004	78.9 ± 5.3	102.6 ± 3.2	113.9 ± 5.5	77.2 ± 9.1	92.7 ± 47.2	73.7 ± 67.5	47.9 ± 5.8	**26.33 ± 0.88**
FD004–>FD001	98.5 ± 0.4	85.6 ± 5.0	119.1 ± 16.7	94.0 ± 8.8	71.3 ± 47.4	67.4 ± 24.3	**31.5 ± 2.4**	37.89 ± 0.17
FD004–>FD002	75.3 ± 1.7	80.8 ± 5.8	37.3 ± 0.6	30.9 ± 1.4	24.9 ± 1.8	39.1 ± 2.5	**24.9 ± 1.8**	28.77 ± 0.34
FD004–>FD003	77.2 ± 6.0	102.9 ± 2.7	68.1 ± 11.1	68.6 ± 11.2	47.5 ± 25.1	45.5 ± 11.0	27.8 ± 2.7	**14.13 ± 0.16**
Mean	95.74	96.22	76.78	67.40	58.15	52.83	37.01	**26.94**

TABLE IV: Score comparison against state-of-the-art approaches

Model	LSTM_DANN	Proposed
FD001–>FD002	93841	**4426**
FD001–>FD003	27,005	**10711**
FD001–>FD004	57,044	**12369**
FD002–>FD001	8,411	**455**
FD002–>FD003	17,406	**3449**
FD002–>FD004	66,305	**12227**
FD003–>FD001	5,113	**1391**
FD003–>FD002	37,297	**20764**
FD003–>FD004	141,117	**48748**
FD004–>FD001	**7,586**	121677
FD004–>FD002	**17,001**	62215
FD004–>FD003	5,941	**537**
Mean	40339	**24914**

shows the performance of proposed approach with respect to Score and RMSE. Note that "Source-Only" means our ADARUL model without domain alignment, i.e., we use the source feature extractor directly on the target domain without domain adaptation. Overall, ADARUL can achieve significant improvement in terms of both Score and RMSE by implementing the domain adaptation in the framework.

1) Comparison Against State-of-the-art Approaches: In this section, to show the superiority of proposed ADARUL, we compare against state-of-the-art approaches in transfer learning for prognostics. The authors of LSTM-DANN [21] reported the performance of 6 domain adaptation approaches. These approaches used three different architectures to extract features such as shallow neural networks (NN), deep neural networks (DNN), and convolutional neural networks (CNN). These architectures have been combined interchangeably with three domain adaptation approaches as follows.

- Transfer Component Analysis (TCA) [25]: TCA is a conventional subspace learning technique based on maximum mean discrepancy (MMD) that regularized by principle Component Analysis (PCA).
- Correlation Alignment (CORAL) [26]: In contrast to subspace based methods, it minimizes the covariance of source and target domains.
- Domain Adversarial Neural Network (DANN) [27]: This method uses a domain classifier coupled with gradient reverse layer to reduce the discrepancy between source and target features.

In our experiments, for fair comparison, we compared against the six aforementioned stat-of-the-art methods using the same cross-domain scenarios experimental settings. Table III shows the comparison of the proposed approach against all the aforementioned related works in terms of RMSE metric. We achieved the best results among 9 cross-domain scenarios. While some cross domain scenarios can be easily adapted, other scenarios can be challenging, which attributed to the different distributions discrepancies across different domains. For instance, we achieve significant improvement when mapping between datasets with single operating condition (e.g., FD001 and FD003), which seems to be close domains. While mapping between datasets with multiple operating conditions (e.g., FD002 and FD004) can be more challenging. Overall, our model is achieving an average RMSE of 26.94, which is significantly lower than that of LSTM-DANN (37.01).

In terms of score metric, to clearly show our superiority, we compared against the best state-of-the-art method (i.e., LSTM-DANN) as shown in Table IV. While our ADARUL achieves better Score performance under 10 out of 12 cross domain scenarios, our model achieves lower Score when mapping from complex datasets (e.g, FD004 to FD001 and FD002). On the whole, our ADARUL still significantly outperforms DANN-LSTM by achieving an average Score of 24914.

IV. Conclusion

In this work, we handled a more challenging yet practical problem of predicting machine RUL under varying working environments. We developed an end-to-end architecture based on deep LSTM network and adversarial domain adaptation. Different from traditional data-driven methods, the proposed deep LSTM network enabled automatic feature extraction and RUL prediction in an end-to-end manner. To mitigate the domain shift problem from different working conditions, we successfully realized an adversarial domain adaptation to learn new feature representation that can be invariant among different working conditions. Experimental results on real-world datasets showed that our proposed ADARUL significantly outperforms the state-of-the-art methods, demonstrating the effectiveness of ADARUL for prognostics across different domains.

References

[1] M. Pecht and J. Gu, "Physics-of-failure-based prognostics for electronic products," *Transactions of the Institute of Measurement and Control*, vol. 31, no. 3-4, pp. 309–322, 2009.

[2] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation–a review on the statistical data driven approaches," *European journal of operational research*, vol. 213, no. 1, pp. 1–14, 2011.

[3] J. I. Aizpurua, V. M. Catterson, I. F. Abdulhadi, and M. S. Garcia, "A model-based hybrid approach for circuit breaker prognostics encompassing dynamic reliability and uncertainty," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1637–1648, 2018.

[4] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, and N. Zerhouni, "Direct remaining useful life estimation based on support vector regression," *IEEE Transactions on industrial electronics*, vol. 64, no. 3, pp. 2276–2285, 2016.

[5] K. Medjaher, D. A. Tobon-Mejia, and N. Zerhouni, "Remaining useful life estimation of critical components with application to bearings," *IEEE Transactions on Reliability*, vol. 61, no. 2, pp. 292–302, 2012.

[6] J. B. Ali, B. Chebel-Morello, L. Saidi, S. Malinowski, and F. Fnaiech, "Accurate bearing remaining useful life prediction based on weibull distribution and artificial neural network," *Mechanical Systems and Signal Processing*, vol. 56, pp. 150–172, 2015.

[7] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International Conference on Database Systems for Advanced Applications*, 2016, pp. 214–228.

[8] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[9] J. Zhu, N. Chen, and W. Peng, "Estimation of bearing remaining useful life based on multiscale convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3208–3216, 2018.

[15] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

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

[11] 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)*, 2017, pp. 88–95.

[12] H. Miao, B. Li, C. Sun, and J. Liu, "Joint learning of degradation assessment and rul prediction for aeroengines via dual-task deep lstm networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5023–5032, 2019.

[13] C.-G. Huang, H.-Z. Huang, and Y.-F. Li, "A bidirectional lstm prognostics method under multiple operational conditions," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8792–8802, 2019.

[14] A. Al-Dulaimi, S. Zabihi, A. Asif, and A. Mohammadi, "A multimodal and hybrid deep neural network model for remaining useful life estimation," *Computers in Industry*, vol. 108, pp. 186–196, 2019.

[16] M. Ma, C. Sun, and X. Chen, "Deep coupling autoencoder for fault diagnosis with multimodal sensory data," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1137–1145, 2018.

[17] Q. Wang, G. Michau, and O. Fink, "Domain adaptive transfer learning for fault diagnosis," in *2019 Prognostics and System Health Management Conference (PHM-Paris)*, 2019, pp. 279–285.

[18] L. Guo, Y. Lei, S. Xing, T. Yan, and N. Li, "Deep convolutional transfer learning network: A new method for intelligent fault diagnosis of machines with unlabeled data," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 9, pp. 7316–7325, 2019.

[19] Y. Fan, S. Nowaczyk, and T. Rögnvaldsson, "Transfer learning for remaining useful life prediction based on consensus self-organizing models," *arXiv preprint arXiv:1909.07053*, 2019.

[20] R. Yan, F. Shen, C. Sun, and X. Chen, "Knowledge transfer for rotary machine fault diagnosis," *IEEE Sensors Journal*, 2019.

[21] P. R. d. O. da Costa, A. Akçay, Y. Zhang, and U. Kaymak, "Remaining useful lifetime prediction via deep domain adaptation," *Reliability Engineering & System Safety*, vol. 195, p. 106682, 2020.

[22] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7167–7176.

[23] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 international conference on prognostics and health management*. IEEE, 2008, pp. 1–9.

[24] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2306–2318, 2016.

[25] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2010.

[26] B. Sun, J. Feng, and K. Saenko, "Correlation alignment for unsupervised domain adaptation," in *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 153–171.

[27] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.