Representation Learning and Knowledge Distillation for Lightweight Domain Adaptation

Sayed Rafay Bin Shah*, Shreyas Subhash Putty† and Andreas Schwung‡
Department of Electrical Power Engineering
South Westphalia University of Applied Sciences
Soest, Germany
Email: *shah.sayedrafaybin@fh-swf.de, †putty.shreyas@gmail.com, ‡schwung.andreas@fh-swf.de

***Abstract*—In industrial machine learning applications, insufficient data, lack of labeling, distribution shift between subsets, varying operational conditions, etc. result in poor generalizing performance by pre-trained neural network models across domains. In contrast to image detection tasks, time series dataset contain critical domain-specific characteristics that must be learned by the corresponding networks. Naively aligning the learned representations during the adaptation process increases the risk of losing these key information, thus resulting in poor performance. This paper proposes a lightweight domain adaptation method involving representation learning and knowledge distillation (RepLKD). A separate network is pre-trained to learn valuable information from the target data in its latent space with the help of a reconstructor. In the adaptation stage, we use maximum mean discrepancy to minimize the difference in distributions between the source and target latent space. Additionally, we implement knowledge distillation to encourage the target network to generate source-like latent embedding and penalize only when an upper-bound condition is not fulfilled to prevent over-regularization and loss of domain-specific features. Finally, we test our proposed method on 12 cross-domain scenarios with the C-MAPSS dataset and compare the efficacy of our method against existing literature methods.**

***Index Terms*—Unsupervised domain adaptation, maximum mean discrepancy, knowledge distillation, representation learning, remaining useful lifetime estimation, C-MAPSS.**

I. Introduction

The transferability of learned representations across multiple tasks is an advantage yielded by deep neural networks. However, inherent dataset bias or domain drift can hinder the performance of a neural network model when generalizing across datasets even originating from similar tasks. In a real-world scenario, maintaining a homogeneous distribution across data from the same system but with slightly different operational conditions is hardly possible. Moreover, supervised learning heavily relies on sufficient labeled data with i.i.d (independent and identically distributed) characteristics, thus becoming infeasible for inadequately-sized dataset without labels. One way to mitigate this is the unsupervised domain adaptation (UDA) approach, where a target domain network is trained in an unsupervised manner with the aid of a pretrained source domain network, already containing valuable information from an almost identical task encoded in its parameters.

Domain adaptation research has made significant progress in the field of computer vision [1] in recent years. In contrast, the progress in industrial application fields with multivariate time series sensor data has been very slow. In an industrial process, operational parameters, machinery conditions, system run-time, external environment, raw material quality, final product requirements etc. keep changing on a frequent basis resulting in distribution shifts across multiple data subsets [2], [3]. Moreover, freshly setting up a prediction model on a new system or a plant can be impractical due to the lack of labeled data from that system. Acquiring enough data points, manually labeling them and learning a new model can be both cost and time expensive. Hence, domain adaptation can play a vital role in bridging the gap across multiple domains within an industrial process. Notable domain adaptation methodologies involve adversarial techniques [4]–[8], representation learning via additional regularizers [9]–[11] or both. Although these frameworks have seen significant success in the recent years, they predominantly focus on image classification tasks and not on time series predictions. Time series datasets contain valuable information regarding degradation patterns, anomalies or failure modes within the raw signals. While optimizing the mapping function between the source and target networks to minimize distribution drift, the target network at the same time must not deviate from its original characteristics and signal patterns which are of high importance in fault predictions or forecasting tasks.

In this paper, we propose an unsupervised domain adaptation framework combining representation learning and knowledge distillation (RepLKD) for domain alignment. Maximum mean discrepancy (MMD) is deployed to reduce the distances between domain representations embedded in a reproducing kernel Hilbert Space [12]. Knowledge distillation is additionally deployed as an upper bound condition-based regularizer to penalize the target network only when deviation from soft source predictions occur beyond a certain margin. Furthermore, to retain inherent degradation patterns in the target domain, we separately perform unsupervised reconstruction using target training data prior to the adaptation phase. The methodology is tested on the C-MAPSS Turbofan engine dataset containing four subsets with varying operational characteristics and the objective is to predict the remaining useful lifetime (RUL) of these engines. Although these datasets are provided with RUL labels, we refrain from using the target labels to imitate an unsupervised adaptation scenario. Finally, we compare our results with existing methodologies and

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00214

demonstrate the effectiveness of our approach in a regression-based industrial application scenario.

The contributions of this paper are as follow:

- We propose a light-weight unsupervised domain adaptation technique by combining MMD and a conditional knowledge distillation function, focused mainly on regression-based remaining useful life estimation.
- We deploy a reconstruction stage for the target domain prior to adaptation to allow the network to learn valuable data information in the latent embedding space.
- Our method outperforms existing literature with similar methods by achieving the best or second best scores in 10 out of 12 cross-domain scenarios with C-MAPSS dataset.

II. Related Work

Domain adaptation has been widely researched in the past decades for reducing target domain drift in both supervised and unsupervised scenarios, with minimization of domain invariant feature space distribution being the primary focus [1], [4], [9], [10], [13]. The gradient reversal layer introduced in [4], [14] optimizes the mapping between the source and target representations by maximizing the loss of a discriminator layer. This work is extended in [15] where the source classifier from the former work is replaced by a regressor layer and LSTM is used as a shared feature extractor. However, such approaches carry a risk of early discriminator convergence and non convergence of the shared feature extractor. The adversarial framework proposed in [5] deploys a secondary network for the target domain while freezing the source domain. This allows independent domain specific mappings and retention of domain-specific features by and restrains premature convergence of the discriminator network. The use of multiple discriminators to alleviate under- or negative-transfer of complex domain invariant knowledge is presented in [6]. Cycle consistency approaches aim to learn mappings that enable forward and backward translations of the original data, thus allowing more domain-specific features to be retained by the translators [7], [8]. Source domain data reweighting for distribution shift reduction and optimizing a classifier on the reweighted data is introduced in [16]. The method also uses Maximum Mean Discrepancy (MMD) to minimize drifts in feature representations from both domains. The same regularizer is used in [11] to minimize the distribution shift between two hidden layers and improve the discriminative capability of deep neural networks. Multi-kernel MMD is proposed in [10] where a part of the network is shared across domains and the representations from subsequent task-specific hidden layers are aligned via maximum mean discrepancy. An additional MMD loss along with task-specific and domain discrepancy losses is implemented in [17] whereas, the framework in [9] exploits MMD as a guiding mechanism to determine the placement and dimension of an adaptation layer while jointly training both the source and target network streams. The progressive adaptation approach in [18] introduces intra- and inter-class domain discrepancy minimization to improve the adaptation capability via class-aware sampling. The deep reconstruction-classification network framework in [19] performs multi-task learning of a shared deep convolutional encoder, a source label classifier and a convolutional reconstructor for the target domain. The approach in [20] includes a reconstructor layer in addition to a domain discriminator and a label classifier to neutralize the effects due to possible outliers in a time series prediction task. To minimize domain drifts in a remaining useful lifetime prediction task, the MDAN method in [21] merges MMD and correlation alignment losses to learn domain invariant features from domain-specific features extractors, which are forwarded to domain regressors penalized by an additional loss. The LAMA-Net architecture in [22] learns domain representations in RUL tasks with MMD loss and performs manifold learning with autoencoders and smoothing constraints. Retraining scale factor and offset parameters of adaptive batch normalization layers in a pretrained deep convolutional network in freeze mode is proposed in [2] for predicting RULs. Our proposed RepLKD approach differs from the existing methods with the inclusion of a pre-reconstruction stage for learning feature representations in the embedding space and conditional knowledge distillation for retention of domain-specific information in the target network along with MMD loss for aligning domain-invariant features between the source and target encoders.

III. Theoretical Background

In this section, we briefly reintroduce the two methods applied in this paper for domain alignment namely, maximum mean discrepancy (MMD) and knowledge distillation (KD).

A. Maximum Mean Discrepancy

Maximum mean discrepancy (MMD) is a non-parametric criterion used to determine the marginal difference in the distribution between two datasets. Given labeled source dataset s and unlabeled target dataset t, x_i^s and y_i^s are the i-th source input sample and corresponding labels and x_j^t is the j-th target input sample respectively. The total number of samples for s and t are given by M and N respectively. The MMD between these two domain datasets is shown as follows,

$$\mathrm{MMD}(s,t;\mathcal{F}) = \sup_{\|f\|_{\mathcal{H}}\leq 1} \left\| \mathbb{E}_{\mathrm{x^s}\sim\mathrm{s}}[f(x^{\mathrm{s}})] - \mathbb{E}_{\mathrm{x^t}\sim\mathrm{t}}[f(x^{\mathrm{t}})] \right\|_{\mathcal{H}} \tag{1}$$

where, a set of functions in the unit ball of a reproducing kernel Hilbert space is given by $\|f\|_{\mathcal{H}} \leq 1$. According to the two sample statistical MMD test, $MMD(s,t) = 0$ if and only if there exists no distribution shift between the two datasets i.e. $s = t$. The empirical estimation of MMD for the two samples can also be stated as the representation loss between latent embedding of two domains $\mathcal{L}_{MMD}(s,t)$ as follows,

$$\mathcal{L}_{MMD} = \widehat{MMD}^2(X^s, X^t)$$
$$= \left\| \frac{1}{N}\sum_{i=1}^{N} f(X_i^s) - \frac{1}{M}\sum_{j=1}^{M} f(X_j^t) \right\|_{\mathcal{H}}^2$$
$$= \sum_{i,j}^{N} \frac{k(X_i^s, X_j^s)}{N^2} - 2\sum_{i,j}^{N,M} \frac{k(X_i^s, X_j^t)}{N, M} + \sum_{i,j}^{M} \frac{k(X_i^t, X_j^t)}{M^2} \tag{2}$$

B. Knowledge Distillation

One of the effective ways of transferring knowledge from a pretrained neural network model to a secondary model with low computational footprint is knowledge distillation. This technique can be applied in domain adaptation whereby a source or the teacher network transfers earned representations or "knowledge" to a target or the student network to mimic the behaviour of the former model, eventually guiding the student to perform better generalizations and improved predictions in an unlabeled dataset. The student network is trained to generate the same predictions as the teacher model predictions. However, naively replicating the teacher outputs can mislead the target network since, the time series datasets may contain degradation patterns that are unique across multiple domains and cannot be forcefully aligned. These patterns are significant in identifying the points of deterioration in an engine's life cycle. To tackle this, we deploy the teacher bounded regression loss as proposed in [23], where the teacher predictions are used as an upper bound for the student to achieve and the student network is only penalized by it when the student prediction loss is higher than the teacher loss by a minimum margin. For predictions $\hat{X}_s^s$ and $\hat{X}_t^s$ by the source and target networks for the source input sample X^s, the conditions apply as below,

$$\mathcal{L}_b = \begin{cases} \mathcal{L}_{MSE}(\hat{X}_t^s, X^s), & \text{if } \mathcal{L}_{MSE}(\hat{X}_t^s, X^s) + m > \\ & \quad \mathcal{L}_{MSE}(\hat{X}_s^s, X^s) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The final regression loss $\mathcal{L}_{reg}$ is the sum of the bounded loss $\mathcal{L}_b$ and the source reconstruction loss by the target network $\mathcal{L}_{rec}^t$ as follows,

$$\mathcal{L}_{reg} = \mathcal{L}_{rec}^t(\hat{X}_t^s, X^s) + v\mathcal{L}_b \tag{4}$$

where, $\mathcal{L}_{MSE}$ denotes the mean squared error between the predicted and ground truth signals and m is the error margin. $\mathcal{L}_{rec}$ is the smooth mean absolute error similar to [23] and v is the weight for the bounded loss.

IV. Methodology

In this section, we present the proposed methodology of representation learning and knowledge distillation (RepLKD) for domain alignment in a remaining useful lifetime estimation task. The steps involved are discussed as follows.

A. Pre-training

The source domain network is designed in an encoder-decoder format where, the encoder compresses the input data into learned latent representations (see Fig. 1(a)). The RUL predictor S_{rul} decodes these embedding into remaining life cycles via supervised learning. In the next steps, the learned knowledge from the source encoder S_{enc} will be distilled into a target encoder T_{enc}. After successful adaptation, the target encoder will be connected to the pretrained RUL predictor to predict RULs for the target test samples. The objective function for the pretraining stage is given as follows,

$$\min_{S_{enc}, S_{rul}} \mathcal{L}_{pretrain} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i^s - \hat{Y}_i^s)} \tag{5}$$

where, $\mathcal{L}_{pretrain}$ is the root mean squared error (RMSE), Y_i^s and $\hat{Y}_i^s$ are the source ground truth and predicted RUL labels respectively for the i-th sample and N is the total number of samples in the source domain.

B. Learning Latent Embedding via Reconstruction

In this step, we introduce two reconstructor networks as shown in Fig. 1(b)- one for the source and the other for the target domain. This stage is divided into two sub-steps with separate objectives. In the source domain, S_{enc} is kept frozen while the reconstructor S_{rec} trains with the aim to recreating the input data using the latent representations from the pretraining stage. S_{rec} is further used in the adaptation stage for knowledge distillation. Hence, this step acts as a warmup training phase where the network is allowed to be optimized on the source domain knowledge. In the target domain, the encoder T_{enc} is initialized with the learnable parameters of the pretrained encoder S_{enc}, and consequently trained with a reconstructor network T_{rec} to recreate the target training samples. By mapping the input data into learned representations by T_{enc} for further reconstruction, we attempt to preserve the degradation patterns or domain-specific features in the network since, these will be vital to predict the critical failure points in a system's life cycle. The optimization functions for this step is given as,

$$\min_{S_{rec}} \mathcal{L}_{S_{reconstruction}} = \frac{1}{N}\sum_{i=1}^{N}(X_i^s - \hat{X}_i^s) \tag{6}$$

$$\min_{T_{enc}, T_{rec}} \mathcal{L}_{T_{reconstruction}} = \frac{1}{N}\sum_{i=1}^{N}(X_i^t - \hat{X}_i^t) \tag{7}$$

where, $\mathcal{L}_{D_{reconstruction}}$ is the MSE loss, X_i^D and $\hat{X}_i^D$ are the i-th input and reconstructed samples for any domain D where, $D \in \{S, T\}$.

C. RepLKD - Domain Adaptation by Representation Learning ad Knowledge Distillation

In this stage, we incorporate the pretrained source encoder S_{enc}, the target encoder T_{enc} and the source reconstructor S_{rec} networks and utilize knowledge distillation with representation

learning for domain adaptation from the source to the target domain. The process is illustrated in Fig. 1(c). Firstly, we freeze the networks S_{enc} and S_{rec} from further training and only train the target encoder T_{enc}. The encoders S_{enc} and T_{enc} are allowed to generate latent space mappings from their respective domain data independently, which are then compared using the MMD loss $\mathcal{L}_{MMD}$. The latent embedding from both domains are consecutively passed forward to the reconstructor S_{rec}, to reconstruct the source input data. $\hat{X}_t^s$ and $\hat{X}_s^s$ represent the reconstructed data when the mapped encoding originate from the target encoder T_{enc} and the source encoder S_{enc} respectively. We perform knowledge distillation (KD) with these reconstructed outputs and encourage the target encoder to approximate its latent space embeddings as close to that of the source encoder as possible. The reconstruction errors are calculated for each domain encoder independently, and T_{enc} is only penalized by this loss when the upper bound conditions are not met, as shown previously in (3). This is done to prevent over-penalization of T_{enc} so that it eventually retains some, if not all of its inherent domain-specific characteristics. The optimization function corresponding to our proposed method can thus be written as follows,

$$\begin{aligned}\min_{T_{enc}} \mathcal{L}_{adaptation} =& \lambda_{rep}\mathcal{L}_{MMD}(E^s, E^t) \\ &+ \lambda_{kd}(\mathcal{L}_{rec}^t(\hat{X}_t^s, X^s) + v\mathcal{L}_b)\end{aligned} \tag{8}$$

where, E^s and E^t denote mapped embeddings from the source and target encoders respectively, λ_{rep} and λ_{kd} are controlling weights for the representation and distillation losses respectively. These weights are tuned as hyperparameters during adaptation training.

V. Experiments

A. Dataset

The C-MAPSS turbofan engine dataset [3] is used for evaluating the performance of our proposed framework. The C-MAPSS dataset is a multivariate time series regression dataset consisting of four individual subsets. Each subset comprises of simulated degradation data of multiple engines. The sensor readings for each engine start from a healthy condition, slowly progressing to the end-of-life cycle. It is evident from Table I that each of the four sub-datasets operate with dissimilar fault modes and operational conditions. This provides us with the ideal scenario of time series datasets originating from the task but with possible domain drift. We randomly choose sensor channels and validate this shift with the help of Kernel Density Estimation (KDE) plots, which allow us to understand the inherent features of a data by estimating the probability density function and can be used to compare feature distribution of two datasets. The bandwidth for distribution smoothing is kept constant at 0.05. Between subsets FD001 and FD002, it is observable in Fig. 2(a) that FD002 engines contain more feature variance than FD001 for the sensor *Total Temperature at LPT outlet*. A similar pattern can be observed between subsets FD003 and FD004 for the sensor *Total Temperature at LPT Outlet.* in Fig. 2(b). The architectures are optimized to predict the RUL of the engines and analyze the performace based on the scoring functions provided in the original literature [3] as follows,

TABLE I
Operational Information of C-MAPSS Dataset [3]

Subset	Operational Settings	Operating Conditions	Degradation Fault Modes	Train Engines
FD001	Sea Level	1	HPC	100
FD002	Altitude Mach Throttle Resolver Angle	6	HPC	260
FD003	Sea Level	1	HPC, Fan	100
FD004	Altitude Mach Throttle Resolver Angle	6	HPC, Fan	249

$$s_{\mathrm{i}} = \begin{cases} exp(-d_{\mathrm{i}}/a_1) - 1, & d_{\mathrm{i}} < 0 \\ exp(d_{\mathrm{i}}/a_2) - 1, & d_{\mathrm{i}} \geq 0 \end{cases} \tag{9}$$

$$S = \sum_{i=1}^{N} s_{\mathrm{i}} \tag{10}$$

where, a_1 = 13, a_2 = 10, d_i = *Predicted RUL - Actual RUL* for the i-th testing engine, s_i is the score of the i-th engine, N is the total number of engines and S is the sum of scores of all engines in a subset. The scoring function is designed to penalize delayed fault predictions.

B. Experiments and Results

Our experiments begin with the supervised training of the source domain network. We use the encoder layer from the vanilla Transformer architecture proposed in [24] due to its self-attention mechanism, coupled with a projection layer as our encoder. The RUL predictor is a multi-layered fully connected network with LeakyReLU activation after each layer. The projection layer involves an adaptive average pooling layer followed by a flatten and a fully connected layer. This layer prevents the shape mismatch due to varying window length in time series data across multiple domains. The hyperparameters (see Table II) are optimized using the Optuna package with early pruning activated. A total of 150 trials are run with 125 epochs each. The dataset was scaled with min-max scaling with (-1, 1) range. The mini-batch size was fixed at 1024 and ADAM optimizer function was used. Furthermore, we implement a new learning rate scheduler as a function of the encoder hidden size d_{model} and the number of warmup steps $steps_{warmup}$ from the total iterations, which is shown below,

$$\begin{aligned}lr =& 0.125 \cdot d_{model}^{-0.5} \cdot min(step\_num \cdot steps_{warmup}^{-1.5}, \\ & steps_{warmup} \cdot steps_{warmup}^{-1.5})\end{aligned} \tag{11}$$

The supervised training results from the source domain w.r.t scoring function and RMSE loss are presented in Table III.

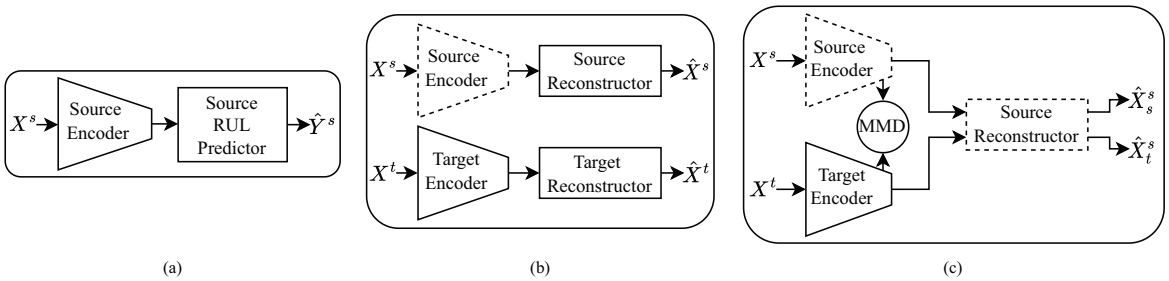

Fig. 1. An illustration of the proposed RepLKD approach *(Dashed line represents non-training mode of a network)*. (a) Pretraining of source domain network with an RUL task. (b) Learning latent embeddings via reconstruction in both source and target domains. (c) Domain adaptation by comparing latent mappings from source and target encoders with MMD loss and reconstructing source input data from domain encoder mappings for further knowledge distillation.

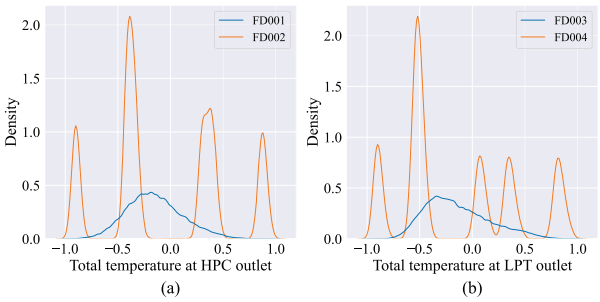

Fig. 2. Kernel Density Estimation Plots for inspecting dataset drift: (a) FD001 vs FD002 for sensor *Total Temperature at HPC Outlet*. (b) FD003 vs FD004 for sensor *Total Temperature at LPT Outlet.*

TABLE II
HYPERPARAMETER SEARCH SPACE FOR SOURCE PRETRAINING

Stage	**Hyperparameter**	**[Categorical Values] / Range (Low, High, Step)**
Training	LR Warmup Ratio	(0.1, 0.4, 0.05)
Encoder	Layers	(1, 3, 1)
	d_{model}	[64, 128, 256]
	Attention Heads	[2, 4, 8, 16]
	Dropout	(0.0, 0.4, 0.1)
Decoder	Hidden Size	[32, 64, 128]

In the warmup reconstruction stage, we deploy two randomly initialized Transformer encoder layers as the source and target reconstructors. The number of layers for these networks are fixed at 1 however, the number of attention heads and the d_{model} size are kept similar to the fine-tuned source encoder network. Using the transformer encoder layer as a reconstructor allows us to use the internal residual connections and self-attention technique to decode the learned embeddings and recreate the original input data. In this stage, we also introduced the target feature extractor initialized with the trainable parameters of the source encoder.

TABLE III
HYPERPARAMETER SEARCH SPACE FOR SOURCE PRETRAINING

	Subdataset			
	FD001	**FD002**	**FD003**	**FD004**
Score	184	1497	240	1946
RMSE	13.96	15.91	12.84	17.81

The networks in the adaptation phase are the source encoder and reconstructor in frozen state and the target encoder in training mode as shown in Fig. 1(c). The network hyperparameters are fixed by this point and the tunable hyperparameters are presented in Table IV. Cosine annealing learning rate scheduler is used in both the warmup and adaptation stages along with ADAM optimizer. This scheduler allows to gradually increase the learning rate and subsequently cool down to zero, thus facilitating a more stable gradient descent. We performed hyperparameter optimization study for 150 trials with 150 epochs each. The first 50 epochs were set as warmup and the rest as adaptation training epochs. Optuna median pruning was activated after the first 75 epochs to prune non-converging trials w.r.t the scores.

The results obtained are presented in Table V. We compare our proposed approach with four existing methodologies namely, LSTM-DANN [15], DDC [9], AdaBN-DCNN [2] and LAMA [22]. With FD001 as the source domain, we achieved the best score with FD004 and second best with FD003 as targets. With FD002 as target, DDC performs best while our approach is closer to the second best score by AdaBN-DCNN. This is significant due to the fact that, FD002 and FD004 are relatively more complex with higher fault modes, more engines and varying feature distributions than FD001 (see Table I). This can also be said with FD003 as source, since RepLKD performed best with FD001 and FD002 and second best with FD004, falling short to AdaBN-DCNN by a small margin. With FD002 as source, RepLKD performed best with FD003 overall and second best by a very small margin with FD004 against AdaBN-DCNN. Our approach lags behind both DDC and AdaBN-DCNN for the FD002-FD001 scenario. We obtained second best results when using FD004 as source for all three targets. Only FD001 provided a slightly declining score whereas, the difference in both FD002 and FD003 as targets are high. To summarize, out of twelve cross-domain

TABLE IV
HYPERPARAMETER SEARCH SPACE FOR ADAPTATION STAGE

Hyperparameter	[Categorical Values] / Range (Low, High, Step)
Learning Rate	(1e-3, 1e-2, $log\_uniform$)
ν	(0.1, 0.9, 0.05)
λ_{kd}	(0.5, 1, 0.05)
λ_{dist}	(0.5, 1, 0.05)

TABLE V
ADAPTATION SCORES COMPARISON

Source	Target	LSTM -DANN [15]	DDC [9]	AdaBN -DCNN [2]	LAMA [22]	RepL KD
FD001	FD002	93841	**5958**	13400	923000	15823
	FD003	27005	288061	**1680**	22400	2368
	FD004	57044	156224	100000	999000	**14875**
FD002	FD001	8411	**640**	1010	20700	1305
	FD003	17406	62823	7000	23000	**2158**
	FD004	66305	44872	**14100**	151000	14618
FD003	FD001	5113	25826	1920	151000	**1336**
	FD002	37297	1012978	78500	8620000	**16690**
	FD004	141117	275665	**13400**	5900000	14721
FD004	FD001	7586	162100	**1580**	31700	1638
	FD002	17001	179243	**4760**	61400	16084
	FD003	5941	**1623**	17700	31400	2556

scenarios in the C-MAPSS dataset, our proposed RepLKD method achieved the best scores in four and second best scores in six scenarios, generated scores almost as good as the top two methods in five and under-performed in only three scenarios.

VI. CONCLUSION

In this paper, we propose a light-weight domain adaptation method RepLKD, designed specifically for time series prediction purposes. Our framework utilizes MMD function to align the domain invariant features in the source and target latent space. As an additional regularizer, we use knowledge distillation with an upper-bound condition with the help of a pretrained reconstructor layer. Furthermore, the target network is pretrained prior to the adaptation stage to learn critical domain-specific features in the encoded embedding. We compare our results with existing methods from the literature and present the efficacy of our approach.

REFERENCES

[1] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 5, pp. 1–46, 2020.

[2] J. Li, X. Li, and D. He, "Domain adaptation remaining useful life prediction method based on adabn-dcnn," in *2019 Prognostics and System Health Management Conference (PHM-Qingdao)*, 2019, pp. 1–6.

[3] 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*, 2008, pp. 1–9.

[4] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[5] 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.

[6] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[8] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*. Pmlr, 2018, pp. 1989–1998.

[9] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[10] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[11] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," in *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings 13*. Springer, 2014, pp. 898–904.

[12] A. J. Smola, A. Gretton, and K. Borgwardt, "Maximum mean discrepancy," in *13th international conference, ICONIP*, 2006, pp. 3–6.

[13] O. Sener, H. O. Song, A. Saxena, and S. Savarese, "Learning transferrable representations for unsupervised domain adaptation," *Advances in neural information processing systems*, vol. 29, 2016.

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of machine learning research*, vol. 17, no. 59, pp. 1–35, 2016.

[15] P. R. de Oliveira 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.

[16] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1410–1417.

[17] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 4, pp. 801–814, 2018.

[18] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4893–4902.

[19] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 597–613.

[20] L. Guo, Y. Yu, Y. Liu, H. Gao, and T. Chen, "Reconstruction domain adaptation transfer network for partial transfer learning of machinery fault diagnostics," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.

[21] Y. Ding, P. Ding, X. Zhao, Y. Cao, and M. Jia, "Transfer learning for remaining useful life prediction across operating conditions based on multisource domain adaptation," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 4143–4152, 2022.

[22] M. Joseph and V. Lalwani, "Lama-net: Unsupervised domain adaptation via latent alignment and manifold learning for rul prediction," *arXiv preprint arXiv:2208.08388*, 2022.

[23] M. R. U. Saputra, P. P. De Gusmao, Y. Almalioglu, A. Markham, and N. Trigoni, "Distilling knowledge from a deep pose regressor network," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 263–272.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.