Domain Adaptation in Predicting Turbocharger Failures Using Vehicle's Sensor Measurements

Mahmoud Rahat[1], Peyman Sheikholharam Mashhadi[1], Sławomir Nowaczyk[1], Thorsteinn Rögnvaldsson[1], Atabak Taheri[2], and Ataollah Abbasi[2]

[1] *Center for Applied Intelligent Systems Research (CAISR) Halmstad University, Sweden*
{*mahmoud.rahat, peyman.mashhadi, slawomir.nowaczyk, thorsteinn.rognvaldsson*}*@hh.se*

[2] *Volvo Group Trucks Technology Gothenburg, Vastra Gotaland County, Sweden*
{*atabak.taheri, ataollah.abbasi*}*@volvo.com*

ABSTRACT

The discrepancy in the distribution of source and target domains is usually referred to as a domain shift. It is one of the reasons for the inferior performance of machine learning solutions at deployment. We illustrate that the domain shift issue is pertinent to the readings of the vehicles' operational sensors. This is due to the fact that these measurements are collected over a period of time and are susceptible to various changes that happen in the meantime. Examples of these changes are usage pattern variations, aging of the vehicles, seasonal shifts, and driver changes. However, domain adversarial neural networks (DANN) have shown promising results to reduce the negative impact of the domain shift. The present study investigates domain adaptation (DA) in the predictive maintenance field by estimating the remaining useful life (RUL) of turbochargers. The devices are operating on a fleet of VOLVO trucks, and the information about their services is collected over four years between 2016 and 2019. The input features to the model are a set of bi-weekly collected measurements called logged vehicle data (LVD). The contributions of this paper are two-fold. First, we propose a new approach for detecting domain (covariate) shift using an autoencoder. Second, we adapt domain adversarial neural networks to the specific application of predicting turbocharger failures. Finally, we deploy a recurrent feature extraction layer in the DANN architecture to incorporate temporal aspect of the data. The experimental results demonstrate the superiority of the proposed method over the traditional approach.

1. INTRODUCTION

The main goal of predictive maintenance (PdM) is to monitor equipment behavior and suggest a proper time for performing maintenance. This provides multiple benefits for businesses by avoiding unexpected breakdowns and improving the quality of customer service. When it comes to heavy-duty vehicles such as trucks, PdM's importance becomes more evident as they usually carry out important tasks around the clock. Any potential breakdown could lead to catastrophic situations in terms of cost, time, and pledges.

Numerous papers in the literature use data-driven approaches and machine learning for predictive maintenance of industrial equipment (W. Zhang, Yang, & Wang, 2019). Based on how they formulate the problem, these methods could coarsely be divided into two groups of regression (Ding, Jia, Miao, & Huang, 2021; Y. Zhang, Hutchinson, Lieven, & Nunez-Yanez, 2020) and classification approaches (Prytz, Nowaczyk, Rögnvaldsson, & Byttner, 2015; Rahat, Pashami, Nowaczyk, & Kharazian, 2020). In the regression approach, the methods estimate a component's RUL as a value of a continuous quantity (unbounded real number). In this scenario, the RUL estimates the amount of time a piece of equipment can perform its intended functionality. However, one could argue that estimation of the exact remaining useful life is too complicated and unnecessary as the only important question that we need to answer each time a truck visits a workshop is binary, either to substitute the component or not (Prytz et al., 2015; Rahat et al., 2020).

By looking at the literature, one can observe that methods based on neural networks and ensemble frameworks are becoming more popular in the field (Mashhadi, Nowaczyk, & Pashami, 2020; Revanur, Ayibiowu, Rahat, & Khoshkangini, 2020; Xia, Song, Zheng, Pan, & Xi, 2020; Rahat et al., 2020; Uddagiri, Ramalingam, Rahat, & Mashhadi, 2021) and generally, the predictive maintenance systems are becoming more

sophisticated over time. Although most of the initial works on PdM were focused on basic components such as industrial bearings (Wang, Liang, Zheng, Gao, & Zhang, 2020), it is clear that the trend is going toward applying PdM on more complex equipment such as batteries (Altarabichi, Fan, Pashami, Mashhadi, & Nowaczyk, 2021), compressors (Fan, Nowaczyk, & Rögnvaldsson, 2015a, 2015b), and turbochargers (Rahat et al., 2020).

The underlying assumption of the machine learning models is that the train and test data are sampled independently from a static distribution that does not change between learning and evaluation. This is generally considered as a necessary condition that ensures the likelihoods the model receives match the expectations within the same distribution. However, this assumption usually does not hold for real-world time-series since such a data is collected under different working conditions overtime. The key motivation of the paper is to formulate changes as such in the context of domain shift and obtain versatility using power of DANN.

Because of the natural temporal order in time-series, it is not possible to shuffle the data randomly between train and test, since we must avoid learning from future observations and evaluating on the past. This has also been considered as part of the sample selection bias (Quiñonero-Candela, Sugiyama, Schwaighofer, & Lawrence, 2008). The significance of applying domain adaptation in the context of time-series data becomes more clear once we consider further challenges such as seasonality, and usage pattern shifts. Knowledge transfer between different domains has recently gained a lot of attention in several neural network applications such as natural language processing (Yang et al., 2019), and machine vision (Kharazian, Rahat, Fatemizadeh, & Nasrabadi, 2020). The failure prognosis and fault diagnosis applications are not an exception to the mentioned trend (Che, Deng, Lin, Hu, & Hu, 2021; Li, Tang, Tang, & He, 2021; Mao, Liu, Ding, Safian, & Liang, 2020).

The two central contributions of the paper are 1) proposing a new approach for domain shift detection using an autoencoder 2) adapting DANN to the specific application of predicting turbocharger failures. Moreover, we use a recurrent feature extraction layer to integrate sequential information in the common feature extraction layers. Previous works have shown the effectiveness of using an LSTM (as feature extraction) in the context of DANN for estimating the remaining useful life (da Costa, Akçay, Zhang, & Kaymak, 2020).

We first analyze the existence of domain shift in the data by using an autoencoder. Then, we address the issue by applying unsupervised domain adaptation. The adopted network architecture simultaneously optimizes two heads; a regressor that estimates the RUL (the primary task) and a binary classifier that predicts whether the samples are drawn from source or target distribution (the auxiliary task). This architecture promotes the emergence, in the shared internal representation layer, of features that remain robust over time. In other words, it helps to avoid overfitting to the source-specific features. We further improve the results by considering a sequence of observations using an LSTM network in the feature extraction layers. The experiments demonstrate that the RUL prediction error reduces significantly over different data splits once we add domain adaptation head to the model.

The remainder of the paper is organized as follows. Section 2 introduces the data. Section 3 presents the proposed methodology. Section 4 demonstrates the results, and finally section 5 concludes the paper.

2. DATA

The data used in this study includes bi-weekly measurement readouts of the sensors installed on 415 VOLVO trucks. These measurements are called logged vehicle data or, in short, LVD and are collected over four years between 2016 and 2019. Most of the measurements are collected through telecommunication and during workshop visits. The original data had unequal timestamps and contained missing values. The missing values are imputed using mean, and linear interpolation is used to equalize the reading intervals. The data includes 372 attributes in total.

The date and time of the measurement, average speed, mileage, vehicle identification number (id), time driving in each gear are among the features available in the dataset. The information about the repairs done on the turbochargers of the trucks is stored in a separate table called Vehicle Service Records (VSR). This table includes vehicle id, part code, and repair dates. The vehicle identifier field (id) is common between the repair and LVD tables and can be used to join two tables.

3. METHODOLOGY

In this section we first, in 3.1, introduce the new approach for domain (dataset) shift detection using autoencoder, and then, in 3.2, we present the adapted model for predicting failures of turbocharger in the presence of such domain shift.

3.1. Domain shift detection using autoencoder

One of the most common types of domain (dataset) shifts is called covariate shift where input sample distribution $P(x)$ is subject to changes (Quiñonero-Candela et al., 2008). A traditional approach to detect such a shift is to label source and target domain samples with zero and one respectively and then train a binary classifier on the newly labeled dataset. A higher performance in the trained model indicates a higher shift in the dataset. As an alternative, we propose to use an autoencoder for detecting the shift in the dataset. Autoencoders lend themselves naturally to this purpose since they consider only reconstruction of input features which is exactly the marginal

distribution required in detecting the covariate shift (change in $P(x)$). Another advantage of using autoencoder for detecting covariate shift is that the effect can also be detected based on the discrepancy between the representation of source and target in the bottleneck layer.

An autoencoder consists of two parts, an encoder $f(x)$ that maps the input x into a lower dimensional representation and a decoder $g(x)$ that reproduces back the original input from the learned internal representation. Therefore, the objective of the network is to minimize the loss function formulated in equation (1) w.r.t the parameters of the model (Θ_f and Θ_g).

$$\ell\left(\Theta_f, \Theta_g\right) = \sum_{i=1}^{N} \left\| x_i - g\left(f\left(x_i\right)\right)\right\|^2 \tag{1}$$

We propose to consider the Average Reproduction Error (in short ARE) as a quantitative indicator of the magnitude of domain shift in the dataset, see equation (2). As an example, let's say we have two datasets are called source S and target T. We train autoencoder on S and evaluate it on T, by calculating the ARE. The higher the value of ARE, the higher the shift between S and T distributions. To the best of our knowledge, this has not been practiced before for measuring the shift between two distributions.

$$ARE = \frac{\ell\left(\Theta_f, \Theta_g\right)}{N} \tag{2}$$

3.2. Predicting turbocharger failures

The prediction model used in this paper is inspired from the domain adaptation technique introduced in (Ganin et al., 2016). Figure 1 shows the adapted structure of the DANN model. The layers on the left side of the figure represent feature extraction layers parameterized in Θ_f. The produced internal representation is then mapped into two separate heads. The upper head performs the main task of RUL prediction (parameterized in Θ_r). Thus, it is essentially a regressor mapping the internal representation to a positive unbounded number. The lower head represents the domain adaptation branch of the network parameterized in Θ_d. This branch is connected to the feature extractor through a gradient reversal layer (Ganin et al., 2016).

The input to the network is a sample with a pair of labels $(x_i, [r_i, d_i])$ where x_i is drawn from $Source \bigcup Target$ distribution randomly, and r_i and d_i represent the ground truth for the remaining useful life (y_{rul}) and the ground truth domain (y_{domain}) respectively. If x_i is drawn from $Target$ domain (i.e. $d_i = 1$), then the task branch will be turned off and r_i won't be considered accordingly. Equation (3) represents the loss function of the network. The negative sign in the formula is to make sure the optimization algorithm maximizes the domain classification error.

$$\ell\left(\Theta_f, \Theta_r, \Theta_d\right) = \frac{1}{N} \sum_{i=1}^{N=N_s+N_d} \ell_r^i\left(\Theta_f, \Theta_r, d_i\right) - \alpha\left(\frac{1}{N} \sum_{i=1}^{N=N_s+N_d} \ell_d^i\left(\Theta_f, \Theta_d\right)\right), \tag{3}$$

where α is a parameter to gauge the effect of adversarial branch and is tuned adaptively, and $\ell_r^i\left(\Theta_f, \Theta_r, d_i\right)$ calculates loss on the task branch (RUL prediction) with respect to the target values (r_i) and predicted values ($\hat{r_i}$) and is calculated using formula (4).

$$\left(1 - d_i\right) \cdot \left|r_i - \hat{r_i}\right|^2 \tag{4}$$

and $\ell_d^i\left(\Theta_f, \Theta_d\right)$ applies cross entropy loss for a binary classification task and is calculated using formula (5).

$$d_i \cdot log\left(\hat{d}_i\right) + \left(1 - d_i\right) \cdot log\left(1 - \hat{d}_i\right) \tag{5}$$

where $\hat{d}_i$ represents the predicted domain of i^{th} sample by the domain branch of the network, d_i indicates the ground truth domain label, and N_s and N_d are the number of samples drawn from source and target distributions. The optimization algorithm in the network simultaneously minimizes the loss on the task branch while maximizing the loss on the domain adaptation branch. The intuition behind maximizing the loss on the domain adaptation branch is to promote emergence of the features that are uninformative in distinguishing between source and target samples. This will ensure the network does not overfit to the source-specific features, and keeps its generalization capacity when transferred to the target domain.

One important aspect of analyzing the time series data is the information available in the sequence of observations. Although the adopted DANN architecture gains boost by introducing the domain adaptation branch, it is still unable to process a sequence of observations as the feature extraction layers only accepts a single readout at a time.

To mitigate this drawback, a recurrent neural network layer known as long short-term memory (LSTM) is used in the feature extraction layer. The input samples to this network are sequences of observations $\left(\left[x_i^{t-k}, ..., x_i^t\right], [r_i, d_i]\right)$ where again $\left[x_i^{t-k}, ..., x_i^t\right]$ is a sequence of observations drawn randomly from $Source \bigcup Target$ distributions. While the DANN model receives a single readout as input, the recurrent archi-

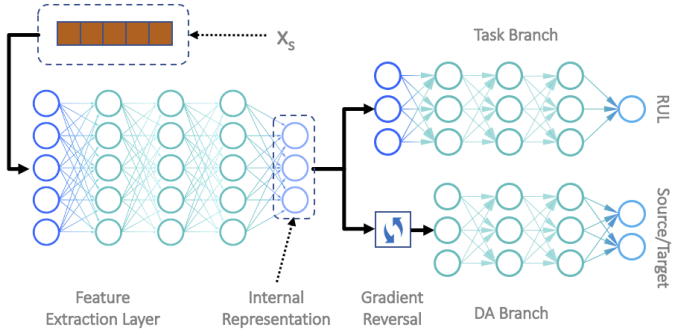

Figure 1. DANN architecture

tecture receives a sequence of observations with length k (in the experiments we consider $k = 8$). Figure 2 shows the structure of the recurrent model. As you can see, the task branch and the domain adaptation branch are similar to the regular DANN.

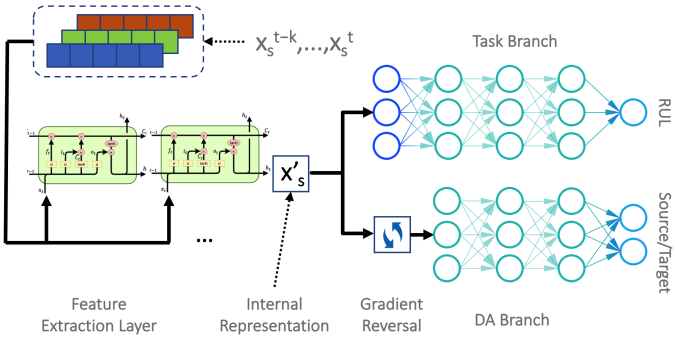

Figure 2. recurrent DANN architecture

4. EXPERIMENTS AND DISCUSSIONS

This section is split into four subsections. 4.1 studies usage of autoencoder for shift detection. 4.2 and 4.3 demonstrate the results achieved by domain adaptation model with single observation and sequential observations. Finally, 4.4 visualizes the effect of domain adaptation in the feature extraction layer by plotting the output of internal representation layer.

4.1. Domain shift in the data

In the first experiment, we conduct an empirical test on the LVD readouts to investigate the existence of distribution change between source and target domains. A three-layer autoencoder neural network is employed in a self-supervised setup to map the sensor readouts into a bottleneck layer (size of 32 neurons) and reproduce the input signals back from the internal representation. Here, we are interested to compare the Average signal reproduction error (see section 3.1) between source and target domains. The Average reproduction error is used as a quantitative empirical indicator of the magnitude of domain shift between different domains.

In order to study the effect of time on the distribution change of input features $P(x)$, we divide the data into four sections using three split dates each six month apart from each other (see figure 3). The first split represents the source domain and is used to train the autoencoder. The three other sections are used as target domains for evaluating the autoencoder where target1 covers six month after the source time span. Target2 covers the time between six months to one year after the source time span. Eventually, target3 starts one year after the source time span.

We first train the autoencoder using source data and then evaluate it using three target distributions each time calculating the mean absolute error between the regenerated signal values and the original values.

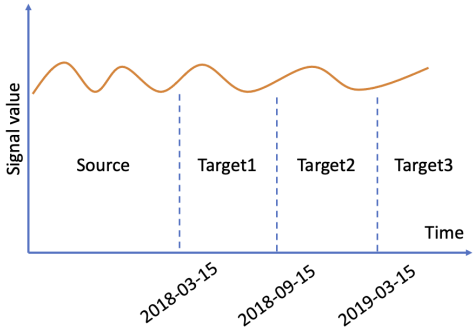

Figure 3. The figure shows how the data is split between source and target domains. The time interval between split dates is six months. The reason for proposing multiple targets is to evaluate the magnitude of the shift in the data compared to source distribution with respect to the time elapse.

Figure 4 shows the learning curves of the autoencoder. The blue curve indicates the training loss on the source domain, while the red, green, and yellow curves illustrate the loss on target1, target2, and target3 respectively. The y axis represents the reconstruction loss, and the x axis shows the training epochs.

The first important observation from the learning curves is that the amount of loss increases comparing target1 to target2 and target3. This illustrates that as we go further away from the source time span, the trained autoencoder quickly losses its generalization capability and is no longer able to accurately regenerate the input signals. Another interesting observation is that the trained autoencoder performs perfectly well on target1 which means the amount of shift is subtle up to six month after the training time. However, the situation changes quickly as we go further away from training period creating a huge contrast comparing the losses of target3 and target1.

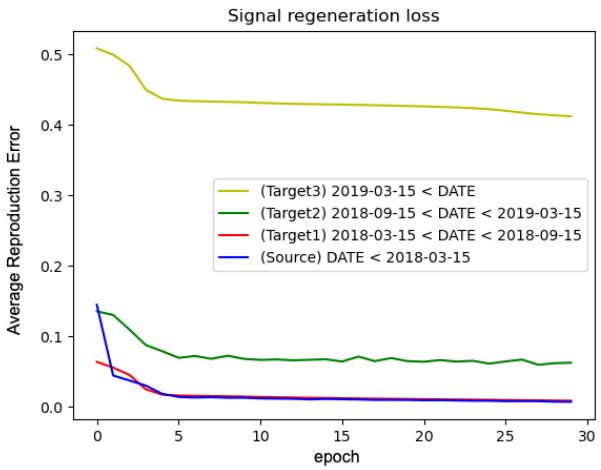

Figure 4. Comparing source and three target losses while generating input signals using an autoencoder. The red curve shows the average loss for a period of six months after source, the green after one year, and the yellow for more than a year. Note how the loss increases over time comparing target1, target2, and target3.

We interpret this variance is as an indication of a distributional shift in the data over time. This shift can dwindle the performance of any data-driven model that tries to predict future by learning from the past values of the signals. Therefore, we suggest the application of domain adaptation.

4.2. Domain adaptation without sequential memory

The previous experiment demonstrated the existence of the distribution shift in the data. In this experiment, we investigate the effect of transductive learning through domain adaptation of source and target features. We do not include the sequence information in the first setup and consider each readout as a stand-alone observation. The effect of adding memory to the model is analyzed afterwards in section 4.3.

A dense feed-forward neural network architecture with two output heads is employed to model the remaining useful life of turbochargers. The network has an input layer with 372 neurons, followed by two feature extraction layers with 100 and 64 neurons. The extracted features are then mapped into two separate heads, the RUL regressor and the domain predictor. The RUL regressor has two layers with corresponding 50 and 1 neurons. In comparison, the domain predictor contains four layers starting with a gradient reversal layer followed by 64, 32, and 2 neurons in the subsequent three layers. All layers use the ReLu activation function except the last layer of the regressor, which uses linear activation function and the last layer of the domain predictor uses softmax binary cross entropy. The Adam optimizer with learning rate equal to 0.001 is used during the training phase. The batch size is 128. Finally, the number of training epochs is tuned using an early stopping mechanism, while 30% of the training samples are set aside as validation.

Table 1. The evaluation results comparing the RUL predictions with or without the domain adaptation respectively presented in "NN" and "DANN" columns. The values show average and standard deviation of five times running the experiment with different random seeds.

Split	Split date	NN (MAE)	DANN (MAE)
1	2017-12-15	159.68±2.59	122.50±2.25
2	2018-01-15	148.71±4.49	116.73±2.50
3	2018-02-15	155.15±4.80	119.46±3.33
4	2018-03-15	151.69±6.44	116.74±2.08

Since the data is in time-series format, it is essential to ensure the natural order of the data is kept during training, i.e., one typical error is to train on the future observations and predict the past. We split the observations into two folds as source and target using a split date. All the observations before the split date are source domain, while the observations after the split date form the target domain. Time splitting is also aligned with the development operation of the company. For example, splitting samples based on the time enables the company to take full advantage of the entire history (sensor measurements and repairs information) for each truck up to the point of deployment. We experimented on several different split dates and presented the results in Table 1. The first column gives the split dates. The second column shows the mean absolute error the network achieves, excluding the domain predictor head, i.e., only a feed-forward regressor is employed. The third column provides the results achieved by engaging both heads (domain predictor and regressor). The suggested model estimates the remaining useful life of a turbocharger. Thus, the model essentially solves a regression task. Therefore, we evaluated the model and reported the results using Mean Absolute Error metric.

To analyze the results, we can compare the mean absolute error values across each row between the second column (Source) as a baseline and the third column (Domain Adaptation). As you can see, the value of error has been reduced significantly by adding the domain adaptation head to the network. This is, of course, attained due to promoting the emergence of invariant features between source and target domains in the feature extraction layers by utilizing the domain adaptation head. Furthermore, the Domain Adaptation shows a slightly lower variance on the error rate (calculated over running the experiments five times with random network initialization).

4.3. Domain adaptation with sequential memory

The collected readouts from vehicles are time-series. Thus there is a natural sequential order available in the dataset. The method utilized in section 4.2 lacks any kind of memory and ignores this sequential information. The recurrent architecture, on the other hand, tries to address this drawback by em-

Table 2. The evaluation results comparing the RUL predictions using recurrent DANN model with two baselines.

Split	LSTM (MAE)	DANN (MAE)	DANN LSTM (MAE)
1	145.77±2.37	121±9.71	111.67±1.69
2	150.27±10.85	154.73±9.71	127.27±5.4
3	173.13±12.69	148.19±11.18	111.09±4.94
4	173.45±9.23	138.57±4.63	108.84±8.77

ploying an LSTM architecture within the feature extraction layers.

The input to the model is a sequence of 8 consecutive readouts, each containing 372 features. The number of units in the LSTM layer is 50, and it is followed by a dropout layer with a dropping frequency of 0.2. The rest of the network architecture, as well as training parameters, are the same as described in section 4.2.

Table 2 compares the results obtained by the recurrent model along with two baselines. The first baseline is called "LSTM" and uses the same architecture as recurrent, but the gradients from the domain adaptation layer are neutralized, i.e. the domain adaptation head is ignored. This baseline helps to study the effect of domain adaptation head. The second baseline is called "DANN" and uses a similar architecture as the one described previously in section 4.2. The only difference here is that the input to the network is a flattened sequence of readouts. This is done to make sure the comparison between two architectures are fair as they both receive similar input signals. This baseline helps to analyze the effect of adding sequential information (memory) to the model.

As you can see, the recurrent model has been able to improve the results one step further. The performance improvement is steady throughout the different splits. In the 2018-03-15 split as an example, the recurrent model has been able to reduce the error by approximately 21% compared to the memory-less DANN model, and 37% compared to the traditional LSTM.

4.4. Visualizing the distribution of source and target features

The whole purpose of adding domain adaptation branch is to motivate the extraction of features that are invariant in terms of source and target domains. The gradients that feature extraction layers receive from domain adaptation branch changes the internal representation of the input features in a way that makes it hard for the network to discern domains.

An interesting way of validating such an effect is to visualize the internal representation of the network. In this experiment, we visualize the internal representation produced by recurrent model. Since the dimension of the internal representation is higher than 2, we applied t-SNE (Van der Maaten & Hinton, 2008) to reduce the number of dimensions. For the parameters of t-SNE, we set number of components equal to 2, perplexity equal to 20, and number of iterations equal to 300.

In order to study the effect of domain adaptation, we run this experiment twice. Figure 5 presents the source and target features without domain adaptation. Figure 6 represents the same features once domain adaptation is added to the model. You can see the distribution of source and target features (the red and blue points) are much more intertwined after applying domain adaptation. This indicates the network has focused more on extracting features that are invariant between two domains.

The t-SNE algorithm follows a stochastic process, therefore, the resulted figure changes with different seed points. However, we observed that the discussed effect is pertinent in all the experiments with different seeds.

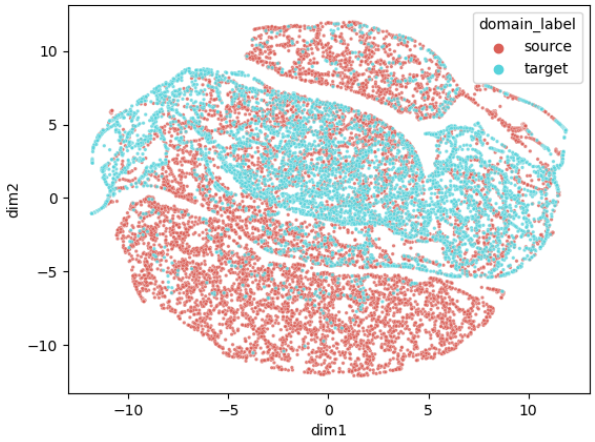

Figure 5. Internal representation of the transformed features WITHOUT domain adaptation

5. Conclusion

In this study, we investigated the occurrence of domain shift in the time series data from vehicle sensor readouts. The aim was to estimate the remaining useful life of turbocharger devices installed on heavy-duty trucks.

We started by showing how one can detect domain shift in the data: by fitting an autoencoder to the measurements from source and evaluating on the target distributions. Then, we employed two neural network architectures inspired from the DANN architecture. One model used a memory-less feature extractor, while the other model employed a Long Short-Term Memory unit.

Finally, we visualized how the domain adaptation changes the internal representation of input features from both source and target domains. This highlights how the adaptation is mak-

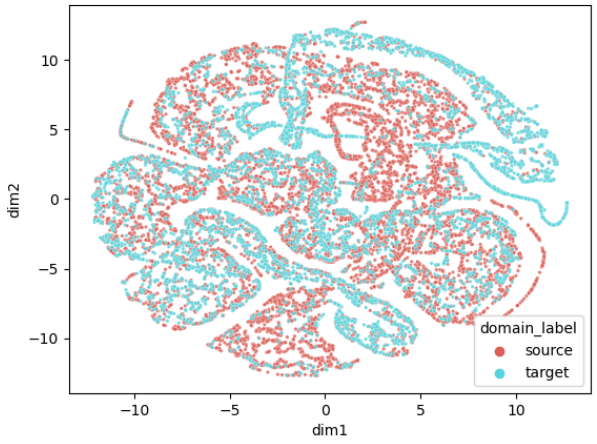

Figure 6. Internal representation of the transformed features WITH domain adaptation

ing it difficult to distinguish between the two domains. The experiments demonstrated that the domain adaptation significantly outperforms the regular architecture. We were able to improve the results even further by incorporating sequential information into the model.

The presented network architecture optimizes for two heads with different characteristics. One head is a regressor and the other one is a classifier. During the experiments, we realized the feature extraction layer receives loss values with different scales from these two heads. Tuning the scale of the losses might be helpful for balancing the effect of domain adaptation. We suggest the task's weight optimization as a future work. The current work estimates RUL of a turbocharger. However, based on each specific application, the RUL values should later be interpreted by the user and considered for decision making. This could also be affected for example by the scheduled workshop visits for each truck (if we want to do the maintenance in the pre-booked regular maintenance visits). The decision making process based on the RUL values is left as a future work.

Acknowledgment

The authors would like to thank VOLVO Trucks Corporation for providing access to the data used in this study. The work was supported by grants from KK-Foundation and Vinnova.

References

Altarabichi, M. G., Fan, Y., Pashami, S., Mashhadi, P. S., & Nowaczyk, S. (2021). Extracting invariant features for predicting state of health of batteries in hybrid energy buses. In *2021 ieee 8th international conference on data science and advanced analytics (dsaa)* (pp. 1–6).

Che, Y., Deng, Z., Lin, X., Hu, L., & Hu, X. (2021). Predictive battery health management with transfer learning and online model correction. *IEEE Transactions on Vehicular Technology*, *70*(2), 1269–1277.

da Costa, P. R. d. O., Akçay, A., Zhang, Y., & Kaymak, U. (2020). Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering & System Safety*, *195*, 106682.

Ding, Y., Jia, M., Miao, Q., & Huang, P. (2021). Remaining useful life estimation using deep metric transfer learning for kernel regression. *Reliability Engineering & System Safety*, *212*, 107583.

Fan, Y., Nowaczyk, S., & Rögnvaldsson, T. (2015a). Evaluation of self-organized approach for predicting compressor faults in a city bus fleet. *Procedia Computer Science*, *53*, 447–456.

Fan, Y., Nowaczyk, S., & Rögnvaldsson, T. S. (2015b). Incorporating expert knowledge into a self-organized approach for predicting compressor faults in a city bus fleet. In *Scai* (pp. 58–67).

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., ... Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, *17*(1), 2096–2030.

Kharazian, Z., Rahat, M., Fatemizadeh, E., & Nasrabadi, A. M. (2020). Increasing safety at smart elderly homes by human fall detection from video using transfer learning approaches. In *30th european safety and reliability conference (esrel2020) & 15th probabilistic safety assessment and management conference (psam15), venice, italy, 1-5 november, 2020.*

Li, F., Tang, T., Tang, B., & He, Q. (2021). Deep convolution domain-adversarial transfer learning for fault diagnosis of rolling bearings. *Measurement*, *169*, 108339.

Mao, W., Liu, Y., Ding, L., Safian, A., & Liang, X. (2020). A new structured domain adversarial neural network for transfer fault diagnosis of rolling bearings under different working conditions. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–13.

Mashhadi, P. S., Nowaczyk, S., & Pashami, S. (2020). Stacked ensemble of recurrent neural networks for predicting turbocharger remaining useful life. *Applied Sciences*, *10*(1), 69.

Prytz, Nowaczyk, S., Rögnvaldsson, T., & Byttner, S. (2015). Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering applications of artificial intelligence*, *41*, 139–150.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2008). *Dataset shift in machine learning*. Mit Press.

Rahat, M., Pashami, S., Nowaczyk, S., & Kharazian, Z. (2020). Modeling turbocharger failures using markov process for predictive maintenance. In *30th european*

safety and reliability conference (esrel2020) & 15th probabilistic safety assessment and management conference (psam15), venice, italy, 1-5 november, 2020.

Revanur, V., Ayibiowu, A., Rahat, M., & Khoshkangini, R. (2020). Embeddings based parallel stacked autoencoder approach for dimensionality reduction and predictive maintenance of vehicles. In *Iot streams for data-driven predictive maintenance and iot, edge, and mobile for embedded machine learning* (pp. 127–141). Springer.

Uddagiri, V. S. V., Ramalingam, S. N. B., Rahat, M., & Mashhadi, P. S. (2021). Predicting hybrid vehicles' fuel and electric consumption using multitask learning. In *2021 ieee 8th international conference on data science and advanced analytics (dsaa)* (pp. 1–6).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11).

Wang, J., Liang, Y., Zheng, Y., Gao, R. X., & Zhang, F. (2020). An integrated fault diagnosis and prognosis approach for predictive maintenance of wind turbine bearing with limited samples. *Renewable Energy*, *145*, 642–650.

Xia, T., Song, Y., Zheng, Y., Pan, E., & Xi, L. (2020). An ensemble framework based on convolutional bi-directional lstm with multiple time windows for remaining useful life estimation. *Computers in Industry*, *115*, 103182.

Yang, M., Zhao, W., Chen, L., Qu, Q., Zhao, Z., & Shen, Y. (2019). Investigating the transferring capability of capsule networks for text classification. *Neural Networks*, *118*, 247–261.

Zhang, W., Yang, D., & Wang, H. (2019). Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, *13*(3), 2213–2227.

Zhang, Y., Hutchinson, P., Lieven, N. A., & Nunez-Yanez, J. (2020). Remaining useful life estimation using long short-term memory neural networks and deep fusion. *IEEE Access*, *8*, 19033–19045.