Domain Adaptation for Structural Fault Detection under Model Uncertainty

Ali I. Ozdagli and Xenofon Koutsoukos

Department of Computer Science, Vanderbilt University, Nashville, TN 37212 USA
Ali.I.Ozdagli@vanderbilt.edu
Xenofon.Koutsoukos@vanderbilt.edu

ABSTRACT

In the last decade, the interest in machine learning (ML) has grown significantly within the structural health monitoring (SHM) community. Traditional supervised ML approaches for detecting faults assume that the training and test data come from similar distributions. However, in real-world applications, the statistical properties of these datasets may diverge from each other in time, leading to a deterioration in the prediction performance of the ML. This paper proposes a domain adaptation approach for ML-based damage detection and localization problems to treat such issues where the classifier has access to the labeled training (source) and unlabeled test (target) data, but the source and target domains are statistically different. The proposed domain adaptation method seeks to form a feature space that minimizes the discrepancy between the source and target domain implementing a domain-adversarial neural network. To evaluate the performance, we present two case studies where we design a neural network model for classifying the health condition of a variety of systems. The effectiveness of the domain adaptation is shown by computing the classification accuracy of the unlabeled target data with and without domain adaptation. Furthermore, the performance gain of the domain adaptation over well-known transfer knowledge approaches such as Transfer Component Analysis and Joint Distribution Adaptation is also demonstrated. Overall, the results demonstrate that domain adaption is a valid approach for damage detection applications where access to labeled experimental data is limited.

1. INTRODUCTION

United States (US) has one of the most sophisticated infrastructures in the world (World Bank, 2019). However, according to a recent study conducted by the American Society of Civil Engineers (ASCE), the US infrastructure is aging and failure on maintaining it may cost an economical loss in GDP as big as $3.1 trillion (American Society of Civil Engineers, 2013, 2017). The condition of infrastructure for other modern societies is also under stress (Zachariadis, 2018). Overall, it is economically not viable to replace all deteriorating infrastructure due to limited resources, and the operations of maintenance, repair, and replacement should be prioritized accordingly. Acting proactively when a critical infrastructure requires care and preventing catastrophic damages call for novel and innovative approaches.

In the last few decades, structural health monitoring (SHM) has gained a lot of momentum as a means of detecting and localizing damages (Sohn, Farrar, Hemez, & Czarnecki, 2002). The introduction of machine learning (ML) into SHM enabled further refinement as mature pattern recognition techniques provide higher accuracy in recognizing structural damages compared to traditional methods (Farrar & Worden, 2012). Among many ML applications, supervised methods are particularly useful (Kiranyaz et al., 2019). Especially, when coupled with artificial neural networks, supervised learning offers promising results for damage detection and localization (Park, Kim, Hong, Ho, & Yi, 2009; Dackermann, Li, & Samali, 2013; Nick, Asamene, Bullock, Esterline, & Sundaresan, 2015).

A majority of supervised SHM applications assume that the data used for training the damage condition classifier has the same distribution as the testing data. However, this assumption is problematic. First, it is unrealistic that one can obtain data belonging to a particular damage condition without actually harming the integrity of the structure before its service (Lu et al., 2016; Gardner, Liu, & Worden, 2020). In other words, creating labeled data based on the original state of the structure is not practical for supervised learning models. On the other hand, we can generate a labeled data set using a representative finite-element model or a similar scaled structure where introducing damages is a more cost-effective approach. The collection of labeled normal and damaged state data from this representative structure is called *source domain* and could

https://doi.org/10.36001/IJPHM.2021.v12i2.2948

be used for training a robust damage condition classifier. The second problem with the supervised ML applications is that a model trained with labeled source domain data may fail to predict the condition of the original structure during testing time. The features for the original structure establish the *target domain*. Both source and target domains are distinct in a way that they have probability distributions that diverge from each other. To summarize, source domain is the model trained on labeled data derived from a representation of the original structure. The model trained on the unlabeled data directly sought from the original structure is the target domain. Both domains have different statistics, which is known as domain shift. The objective of domain adaptation is to design a new learning architecture that generalizes the prediction over both domains (Goodfellow, Bengio, & Courville, 2016). This generalization is achieved by finding a mapping that can extract domain-invariant features. Eventually, this mapping is expected to improve the prediction accuracy for the target domain compared to an architecture that does not implement domain adaptation. In brief, transfer of knowledge gained from source domain to target domain is conceptualized as domain adaptation (see Figure 1).

First attempt for domain adaptation started by addressing the distribution shift between labeled training and unlabeled test data. For example, Kernel Mean Matching (KMM) aims to minimize the covariate distribution between two datasets in a higher feature space called Reproducing Kernel Hilbert Space (RKHS) by reweighing the sample data. As a result, KMM is capable of producing a mapping that can match the test data distribution in RKHS (Gretton et al., 2009). While KMM outperforms ordinary classifiers and regressors, the improvement is limited to covariate shift such that the conditional distribution remains same ($P_{train}(y|x) = P_{test}(y|x)$) but input distribution shifts ($P_{train}(x) \neq P_{test}(x)$) across both domains (Bouvier, Very, Hudelot, & Chastagnol, 2019).

Many domain adaptation problems are susceptible to dataset shift where $P(Y|X)$ is not conserved between source and target domains to its highest degree (Wang & Deng, 2018; Wilson & Cook, 2020). Thus, reweighting algorithms are not always effective in such cases. Modern domain adaptation techniques focus on finding a common latent space (also known as domain-invariant feature space) that represents both source and target domains. For example, as an improvement to KMM, maximum mean discrepancy (MMD) metric is introduced to measure the divergence between distributions and to compute a function in RKHS to maximize the difference in expectations between two probability distributions (Borgwardt et al., 2006). A well-known transfer learning method, transfer component analysis (TCA) uses this MMD metric to minimize the maximum expected distribution shift between source and target domain (Pan, Tsang, Kwok, & Yang, 2010). Similarly, joint distribution adaptation (JDA) utilizes MMD to measure the statistical difference in marginal and conditional distributions (Long, Wang, Ding, Sun, & Yu, 2013). Within the SHM community, (Lu et al., 2016), (X. Li, Zhang, & Ding, 2018), and (X. Li, Zhang, Ding, & Sun, 2019) utilized MMD metric as a loss function for the training of neural networks to improve the prediction over target data using both source and target data during training for gear fault diagnosis. Similarly, (Xie, Zhang, Duan, & Wang, 2016) and (Gardner et al., 2020) applied TCA to classify the damage on gears and structures, respectively. (Xie et al., 2016).

The new generation domain approaches exploit adversarial training to find domain-invariant features (Wilson & Cook, 2020). These approaches adopt the zero-sum game where a label classifier (the network that predicts the correct label of input whether it is coming from source or target domain) is trained to deceive a domain classifier (another network in parallel that predicts whether the input is source or target domain data). For instance, Domain Adversarial Neural Network (DANN) uses gradient reversal layer during back-propagation to reverse the domain classifier weight derivatives to maximize the domain confusion (Ganin et al., 2016). Adversarial Discriminative Domain Adaptation (ADDA) uses a two-step approach where the network is first pre-trained on source data and then a domain classifier is trained to learn target domain features. As an alternative to DANN-type of domain adaptation, domain mapping approach uses GANs to translate a sample data from target domain to source domain (Benaim & Wolf, 2017; Zhu, Park, Isola, & Efros, 2017). However, these applications are limited to image-like domains.

This paper introduces an effective domain adaptation approach to address the distribution shift between source and target domain for supervised machine-learning-based SHM applications. More specifically, we utilize a domain adversarial neural network (DANN) approach to predict the damage condition of a structure operating under a target domain using both labeled source and unlabeled target domain data during training time. The main purpose of the DANN architecture is learning features that represent both source and target domains. To achieve this goal, DANN implements a multi-task topology that combines a regular feed-forward neural network (NN) based damage classifier using source data with a domain discriminator NN which utilizes source and target domain data. The domain discrimination component enables the feed-forward NN to extract latent features underlying both domains by minimizing H-divergence between domains.

To demonstrate the suitability of the DANN for SHM applications, the paper investigates two case studies. The first case study focuses on a gearbox system with a set of damage conditions operating under low- and high-load. A DANN model is trained with labeled low-load and unlabeled high-load data to predict the damage condition for the high-load operation of the gearbox. Additionally, for this case, DANN

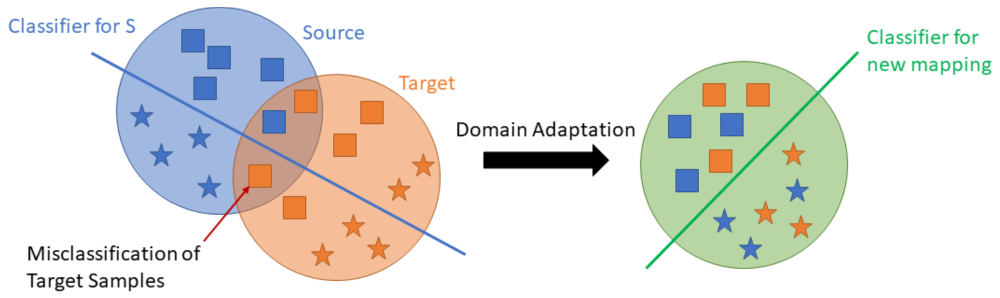

Classifier for S
Source
Target
Domain Adaptation
Classifier for new mapping
Misclassification of Target Samples

Figure 1. Concept of Domain Adaptation

is compared to two well-known transfer knowledge methods, Transfer Component Analysis (TCA) and Joint Distribution Adaptation (JDA) to show the performance gain. In the second case, the effectiveness of the domain adaptation from the numerical model to experimental data is studied for a small-scale three-story structure. The numerical model of the structure is used to simulate various damage conditions for the source domain whereas the experimental data constitutes the target domain. Results from both case studies indicate that domain adaptation is a viable method for SHM applications, and it increases the accuracy of damage condition prediction considerably. Additionally, the DANN can be considered as a potential ML architecture enabling appropriate knowledge transfer across the source and target domains.

For many machine-learning-based SHM applications focusing on damage detection and localization, a shift from source to the target domain is expected. Domain adaptation is a viable methodology for minimizing the distribution shift between source and target domains. This paper demonstrates that DANN is a suitable approach for learning latent features that underline both source and target domains. The case studies examined in this paper show that DANN improves the prediction accuracy of supervised damage detection and localization algorithms.

Overall, the major findings and contributions of this paper can be summarized as below:

- For many machine learning-based SHM applications focusing on damage detection and localization, a shift from source to the target domain is expected. Domain adaptation is a viable methodology for minimizing the distribution shift between source and target domains.
- This paper demonstrates that DANN (Ganin et al., 2016) is a suitable approach for learning latent features that underline both source and target domains for SHM applications. The case studies examined in this paper show that DANN improves the prediction accuracy of supervised damage detection and localization algorithms.
- The effectiveness of DANN is compared to the black-box approach and traditional knowledge transfer methods called TCA and JDA. Our results show that DANN outperforms all three architectures.

The rest of the paper is outlined as follows. First, Section 2 discusses condition monitoring briefly and formulates the domain shift problem. This section introduces the DANN model for SHM applications as well. Section 3 presents case studies and the evaluation results. Lastly, Section 4 summarizes the paper and draws conclusions.

The code to generate the results in this paper can be accessed from `https://github.com/aliirmak/DASHM`.

2. DOMAIN ADAPTATION IN SHM

In traditional SHM applications, vibration data is captured from various locations of the structure in the form of accelerations (Abdeljaber, Avci, Kiranyaz, Gabbouj, & Inman, 2017; Ozdagli & Koutsoukos, 2019). Meaningful features extracted from these measurements through time or frequency domain analysis establish the input space for a supervised learning model. Each data in the input space can be associated with a label describing the structural condition in terms of the location of the damage and its intensity to form $\{X, Y\}$. Supervised learning algorithms require access to those labeled data for proper training. While the no-damage/normal data is often available when the structure is first erected, it is impractical to abuse the structure just to obtain the data relevant to various damage conditions.

As a solution to the main fallback of the supervised learning methods, model-based SHM approaches exploit numerical models to establish a baseline for damage detection and damage localization (Mirzaee, Abbasnia, & Shayanfar, 2015; Figueiredo, Moldovan, Santos, Campos, & Costa, 2019). Numerical models can be useful for generating labeled source domain data. However, an ML model trained with source domain data may suffer from the uncertainty gap between the numerical model and the experimental structure (Catbas, Gokce, & Frangopol, 2013). Consequently, the learning model may not yield correct labels for the unlabeled target domain and may diagnose the damage improperly for the target structure. From the domain adaptation perspective, the distribution shift between source and target domain should be addressed (Singh, Azamfar, Ainapure, & Lee, 2020; J. Li, Li, He, & Qu, 2020). Accordingly, the problem for supervised SHM applications is finding domain-invariant features that represent both labeled source and unlabeled target domain.

In this paper, the source domain D_S consists of labeled data derived either from numerical simulations or from a particular state of the structure (for example, low wind, low traffic load, low-load, etc. corresponding to the normal operation). The target domain D_T is either the data captured from the experimental structure or an operational state of the structure that is not relevant to source domain (such as high wind, high traffic, high-load, etc. corresponding to stressing operations) and it is unlabeled. Then, the typical domain adaptation task for supervised SHM application is predicting the class for unlabeled target domain data using the knowledge gained from both source and target data.

For SHM, it is natural to consider a classification task where $X = \{x_i\}_{i=1}^{N}$ is the input space of features and $Y = \{y_i\}_{i=1}^{N}$ is the output space corresponding to the labels. Suppose that we have two different distributions over the $\{X, Y\}$: *i)* D_S is the source domain which contains the labeled source samples with $S = \{(x_i, y_i)\}_{i=1}^{n} \sim D_S$; and *ii)* D_T is the target domain which consists of the unlabeled target samples with $T = \{x_j\}_{j=1}^{n'} \sim D_T$. We assume that the distributions for both domains are different such that $D_S \neq D_T$. This implies that the distributions for the input space from S and T are not identical, namely $p(X_S) \neq p(X_T)$. Similarly, the conditional distributions that are used for inference may not match, that is $p(Y_S|X_S) \neq p(Y_T|X_T)$. Given D_S and D_T, the task for the domain adaptation is to build a classification model $h(x)$ which can predict correct labels for samples from D_T using the knowledge learned from D_S and D_T.

2.1. Domain Adversarial Neural Network

A common domain adaptation approach is finding a mapping function that can minimize a probabilistic discrepancy metric between the two domains. The majority of these metrics focus on computing the divergence, i.e., the distance between two probability distributions. For example, the kernel mean matching (KMM) algorithm minimizes the mean distance in a kernel space by re-weighting the target domain with respect to source domain(Huang, Gretton, Borgwardt, Schölkopf, & Smola, 2007). The approach in (Sugiyama, Nakajima, Kashima, Buenau, & Kawanabe, 2008) proposes to minimize the Kullback-Leibler (KL) divergence for minimizing domain shifts. A well-known transfer learning algorithm called transfer component analysis (TCA) utilizes Maximum Mean Discrepancy (MMD) to minimize the distance between two domains in Hilbert space (Sejdinovic, Sriperumbudur, Gretton, & Fukumizu, 2013; Pan et al., 2010). Lastly, (Ben-David et al., 2010) hypothesize that a classifier-induced divergence, namely H-divergence is sufficient for domain adaptation.

H-divergence relies on distinguishing the examples of D_S and D_T and computing the domain divergence from the data in both domains. Accordingly, we label the data from D_S and D_T as *0* and *1*, respectively. Then, we have a new dataset that can be described as:

$$U = \{x_i, 0\}_{i=1}^{n} \cup \{x_j, 1\}_{j=1}^{n'} \tag{1}$$

Then, the objective is to develop a function that predicts the class of the sample input χ correctly, i.e., $f : \chi \rightarrow [0, 1]$. Similarly, $h'(x)$ is the learned model $h' : \chi \rightarrow [0, 1]$. Then, the generalized error is:

$$\epsilon = E[|h'(x) - f(x)|] \tag{2}$$

Given ϵ, the H-divergence is approximately:

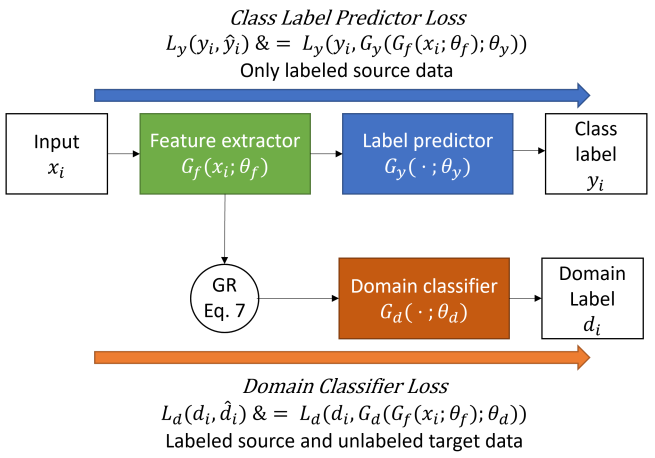
$$d = 2(1 - \epsilon) \tag{3}$$

One purpose of the domain adaptation is minimizing the H-divergence d. More details on the derivation of the divergence can be found in (Ben-David et al., 2010; Ganin et al., 2016).

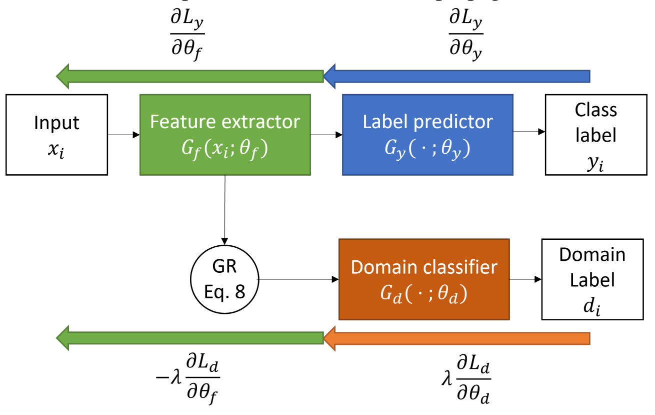
There are inherently two tasks for implementing H-divergence based domain adaptation. First, we want to train a domain classifier $h'(x)$ that can discriminate between source and target domains. At the same time, we want to design a class predictor $h(x)$ to correctly predict the class for the source domain data during training. It should be noted that the class predictor cannot be trained on target domain data as they are unlabeled. The ultimate aim of the domain adaptation is finding features that underline both the source and the target domain. Such a representation is expected to minimize the H-divergence and the domain predictor $h'(x)$ should not be able to distinguish between the source and target domains.

The domain-adversarial neural network (DANN) approach introduced in (Ganin et al., 2016) exploits this objective by proposing a multi-task learning approach. The DANN is composed of three components: feature extractor, label predictor, and domain classifier). Figure 2 illustrates the training (forward and backward propagation) and testing phases. Here, the feature extractor (green colored) is a set of neural network layers that extracts domain-invariant features for a given input. The label predictor (blue colored) layers predict the label of a given input based on the domain-invariant features computed by feature extractor layers. The domain classifier (red colored) is tasked with discriminating the domain of the input whether it is originating from source or target domain. Finally, a gradient reversal layer denoted as GR connects feature extractor to domain classifier. GR changes its behavior based on the type of propagation.

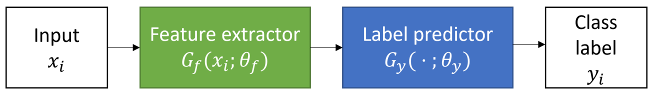
During the forward propagation (see Fig. 2 a), the class label


Class Label Predictor Loss
$L_y(y_i, \hat{y}_i)$ & = $L_y(y_i, G_y(G_f(x_i; \theta_f); \theta_y))$
Only labeled source data
Input x_i
Feature extractor $G_f(x_i; \theta_f)$
Label predictor $G_y(\cdot\,; \theta_y)$
Class label y_i
GR Eq. 7
Domain classifier $G_d(\cdot\,; \theta_d)$
Domain Label d_i
Domain Classifier Loss
$L_d(d_i, \hat{d}_i)$ & = $L_d(d_i, G_d(G_f(x_i; \theta_f); \theta_d))$
Labeled source and unlabeled target data

(a) Computation of loss - Forward propagation


$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
Input x_i
Feature extractor $G_f(x_i; \theta_f)$
Label predictor $G_y(\cdot\,; \theta_y)$
Class label y_i
GR Eq. 8
Domain classifier $G_d(\cdot\,; \theta_d)$
Domain Label d_i
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
$\lambda \frac{\partial L_d}{\partial \theta_d}$

(b) Computation of gradients - Backward propagation


Input x_i
Feature extractor $G_f(x_i; \theta_f)$
Label predictor $G_y(\cdot\,; \theta_y)$
Class label y_i

(c) Testing phase

Figure 2. Simplified DANN architecture

loss is computed only over the labeled source domain data, whereas the domain classifier loss is calculated using both labeled source and unlabeled target domain data. In this phase, GR acts as a linear function and does not modify the propagation of loss. During the backward propagation phase (see Fig. 2 b), the gradients of the losses are computed. In this phase, GR reverses the gradient by multiplying the propagation with a negative small constant. This negative gradient maximizes the domain confusion by enforcing latent features extracted from both source and target domain to be indistinguishable. After training is complete, one can test the network only using feature extractor and label predictor layers.

For a general DANN, the loss function can be formulated as following:

$$\mathcal{L} = \frac{1}{n_s} \sum_{x_i \in D_s} L_y(y_i, \hat{y}_i) - \frac{\lambda}{n_s + n_t} \sum_{x_i \in D_s \cup D_t} L_d(d_i, \hat{d}_i) \tag{4}$$

where n_s is the number of source domain sample; n_t is the number of target domain samples; L_y and L_d are the class label and domain label loss, respectively; λ is the trade-off parameter between class label and domain label loss; x_i is the i_{th} input; y_i and $\hat{y}_i$ are the corresponding true and predicted class labels, respectively; d_i and $\hat{d}_i$ are the corresponding true and predicted domain labels, respectively.

Next, we denote the feature extractor layers as G_f, label predictor as G_y, and domain classifier as G_d. Accordingly, L_y and L_d can be formulated as:

$$L_y(y_i, \hat{y}_i) = L_y(y_i, G_y(G_f(x_i; \theta_f); \theta_y)) \tag{5}$$
$$L_d(d_i, \hat{d}_i) = L_d(d_i, G_d(R(G_f(x_i; \theta_f)); \theta_d)) \tag{6}$$

where θ_f, θ_y, and θ_d are the weights for feature extractor, label predictor, and domain classifier, respectively. Here, $R(x)$ represents the gradient reversal layer as a function. The GR function has distinct behavior for forward (see Eq. 7) and backward propagation (see Eq. 8) as prescribed below:

$$R(x) = x \tag{7}$$
$$\frac{dR}{dx} = -\mathbf{I} \tag{8}$$

The computed loss and gradients based on the GR function, are given in Figures 2a and 2b. Then, the objective of the training is finding the weights, θ_f, θ_y, and θ_d that optimize the joint loss, $\mathcal{L}$ given in Eq 4 by minimizing the label predictor loss and maximizing the domain classifier loss. Accordingly, the weights are updated during gradient descent as given below:

$$\theta_f = \theta_f - \mu \left(\frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f} \right) \tag{9}$$
$$\theta_y = \theta_y - \mu \frac{\partial L_y}{\partial \theta_y} \tag{10}$$
$$\theta_d = \theta_d - \mu \lambda \frac{\partial L_d}{\partial \theta_d} \tag{11}$$

where μ is the learning rate.

While the architecture explained here provides a generic prescription for the implementation of an arbitrary DANN architecture, the choice of hyperparameters such as number and types of layers, activations, and loss functions per component (feature extraction, label prediction, domain discrimination) depends on the task, experience, and expected performance.

3. EVALUATION, RESULTS, AND ANALYSIS

For the evaluation of the proposed domain adaptation approach for SHM, two case studies are analyzed. The first case study investigates the prediction performance for the damage condition of a gearbox system under various torques. In the second case study, a three-story structure with several levels of damage conditions is used.

3.1. Case Study 1: Gearbox Fault Detection

3.1.1. Dataset and Preprocessing

PHM Data Challenge 2009 introduced a dataset simulating various fault types for a generic gearbox system (*PHM data challenge 2009.*, 2009). Acceleration data were collected at the input and output shaft of the gearbox at different shaft speeds (30, 35, 40, 45, and 50 Hz) under two different loading conditions (low- and high-load). For each shaft speed and loading conditions, 6 fault types are simulated (normal, chipped gear tooth, broken gear tooth, bent shaft, imbalanced shaft, broken gear tooth with bent shaft). For each case which is the combination of fault type, shaft speed, and load condition, about 4 seconds of data is collected at a sampling rate of $f_s = 66.67$ kHz twice. *In this paper, only the output shaft vibration data is considered.*

According to the literature on gearbox fault detection (Chen, Li, & Sanchez, 2015; Jing, Zhao, Li, & Xu, 2017), the frequency domain provides a rich feature set for fault detection using vibration data. Thus, before training, all raw data is converted to the frequency domain using sliding-window Fast Fourier Transformation (FFT) also known as Short-Time Fourier Transform (STFT). The parameters for the transformations are selected as prescribed by the length of each window segment which is 1000 samples. The segments overlap by 80 percent and the sample length of FFT is 1200. The frequency resolution is $\Delta f = 111$ Hz. After prepossessing, each damage condition case has about 2700 data points with 601 features per loading condition. Here, each feature represents a STFT value corresponding to a frequency point discretized with Δf (i.e. $f_s/\Delta f = (66.67kHz)/(111Hz) \approx 601$). The dataset is divided into source and target domains according to loading conditions. The source domain corresponds to low loading conditions consisting of all shaft speeds and fault types whereas the target domain is composed of the high-load operation. Since the task is detecting the type of the fault regardless of shaft speed, the data belonging to the same fault type are stacked together. Finally, both domain data are split into training and test data using a 4-to-1 ratio. All data is standardized with respect to the source training data and all labels are one-hot encoded.

3.1.2. Implementation

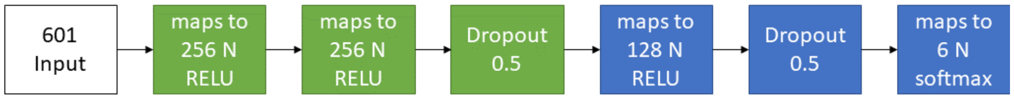
Three different models are developed: *Model 1:* source-only model which is trained only with source domain data; *Model 2:* the multi-tasking DANN model for training which uses both source and target domain data to discriminate the domain and predict the label; and *Model 3:* single-task DANN model for prediction and used only for testing. The architectures are shown in Figure 3.

Here, each box corresponds to a layer of the neural network. *Input* is the input layer that utilizes the features. N is number of neurons. In other words, *256 N* within a box means 256 densely connected neurons are used for the particular layer. *RELU* stands for Rectified Linear Unit, and it is the activation function for the neurons defined within the layer. *Dropout* layers are represented along with the rate of dropout. For example, *Dropout 0.5* means a layer with a dropout rate of 0.5 is used. Finally, *Softmax* is the softmax activation layer that provides the class membership probability for each class.

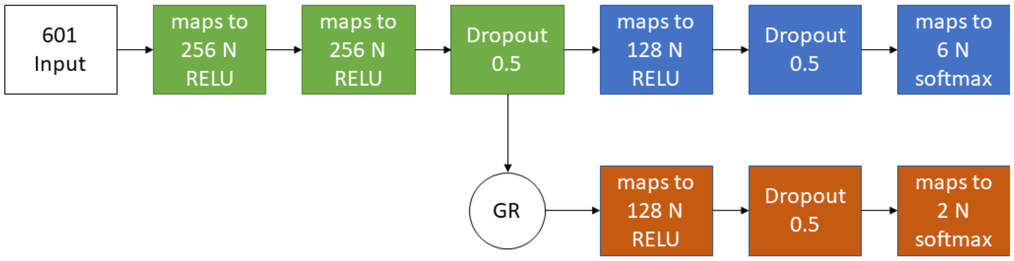
The source-only model is a shallow network consisting of feature extraction (FE, colored in green) and class prediction (CP, colored in blue) layers. In addition to FE and CP layers, the multi-tasking DANN model includes the domain discriminator (DD, colored in red) layers and the gradient reversal (GR) layer. The single-task DANN model has the same structure as the source-only model but with updated weights where the FE contains the latent features that represent both source and target domains after training. Model 1 and Model 2 are trained using stochastic gradient descent. All the losses are chosen as categorical cross-entropy.

The low loading condition data represents the source domain whereas the high loading condition data corresponds to the target domain. During training, the DANN utilizes 128 data points (64 from source and 64 from target) per batch. We assume we have access to the source data labels but not to the target domain labels. The source (input, label) tuples are used explicitly for the class prediction task. For domain prediction, the source data is labeled as *0* and target data as *1*, and then the labels are one-hot encoded. The domain predictor uses both domain data for training and creating domain invariant features. The source-only model is trained with 75 epochs whereas the DANN is trained for 200 epochs.

In addition to DANN, TCA and JDA are used for comparison. TCA utilizes training data from both source and target domain to realize dimension reduction using radial basis function as the kernel. After dimension reduction, a support vector machine (SVM) classifier is trained on the labeled source data as prescribed by (Pan et al., 2010). This classifier is also used to predict labels on unlabeled target data. Like TCA, dimensionality reduction procedure is applied on the training data from both domains for JDA. After feature transformation, a 1-Nearest Neighbor classifier (kNN classifier with



(a) Source-only model



(b) DANN model

Figure 3. Source-only and DANN architectures for numerical example

$k = 1$) is trained on the labeled source data as prescribed in (Long et al., 2013). This classifier is tested on the unlabeled target data used for training. Due to the way JDA is implemented, the algorithm cannot be tested on a new dataset other than the one it is trained with. Testing JDA on labeled source data will result to 100% accuracy. Results for source data indicate the perfect accuracy as N/A (not available).

Since TCA and JDA are essentially a set of matrix multiplications and eigenvalue decompositions, the complete training dataset does not fit into the memory. Due to this limitation, only a quarter of training samples are used from both domains. Both TCA and JDA methods are only applied to the first case study and then discarded for the second case due to their low performance.

3.1.3. Results

Table 1 shows the accuracy for source and target domain test data on the source-only model and the DANN model. The accuracy for predicting the source data is about 97 percent for both models. The accuracy for both source-only and DANN models on the training data is above 99% (not reported in the table). However, the generalization gap between training and testing is small which is an indicator for minimal overfitting.

Without domain adaptation, the accuracy of the target data for the source-only model is 64 percent. The DANN improves the prediction on target data and increases the accuracy to 71 percent. TCA method produces lower accuracy for source and target domain data ranging between 42 to 63 percent. Due to its limitation, JDA is tested only on training source and target data. JDA produces 100% accuracy for source data, and it is donated as N/A. JDA's domain adaptation performance for target data is around 42% and it is marginally lower than TCA.

The accuracy per class is visualized in Fig. 4. Accuracy plot shows that DANN performance improves significantly for some classes over source-only model. For class 3, the performance of both models is similar. A negative transfer learning is observed for class 4 and 5. As a result of this, the prediction performance of DANN is decreased. For all most all classes, TCA and JDA do not exhibit a good domain adaptation. Additionally, Table 2 illustrates the precision, recall, and F1 scores for all the models presented. Among all the models, DANN has the highest precision, recall, and F1 scores despite the slight negative transfer learning.

Overall, results indicate that DANN outperforms TCA and JDA significantly and this finding implies that TCA and JDA may not be used as a reliable domain adaptation method.

Model	Input	Accuracy
Source-only Model	Source	97.46%
	Target	64.43%
DANN Model	Source	97.55%
	Target	71.79%
TCA Model	Source	61.57%
	Target	45.25%
JDA Model	Source	N/A
	Target	42.08%

Table 1. Domain adaption performance for gearbox fault detection

Model	Precision	Recall	F1
Source-only	65.81	64.47	64.59
DANN	72.06	71.80	71.90
TCA	45.24	45.25	44.10
JDA	41.82	42.09	41.06

Table 2. Additional performance scores for gearbox fault detection

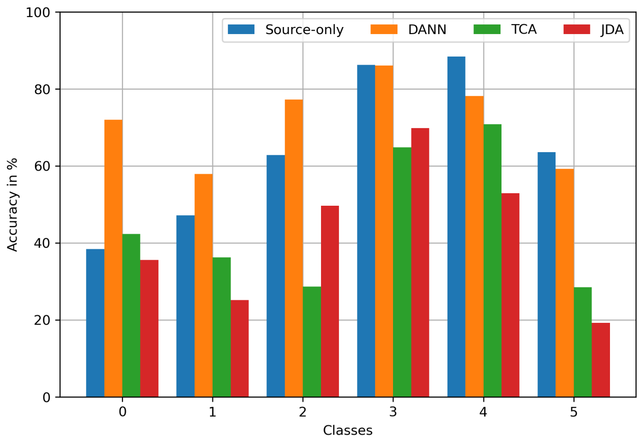


Figure 4. Accuracy per class

In addition to the domain adaptation performance, we examined the empirical computational cost of each approach. Table 3 illustrates the average runtime for each model when they are trained 10 times. Among all, source-only model takes about 75 seconds to train over the entire dataset. The training time for DANN is longer (455 sec) as expected, since it utilizes a larger data set (source and target), and it is a multi-task learning environment. Despite TCA and JDA use only a quarter of the data for training, it takes considerably a lot longer to train them. DANN employs stochastic gradient descent (SGD) for the optimization of the model weights. Thus, there is no need to load all the training samples in the memory at once. However, TCA and JDA require all the training data in the memory. In addition, the eigenvalue decomposition that TCA and JDA rely on is a set of matrix multiplications that are known to be computationally expensive for large dimensions.

Model	Average Training Time (sec)
Source-only	75
DANN	455
TCA	6470
JDA	1520

Table 3. Empirical computational costs for domain adaptation methods

3.2. Case Study 2: Structural Damage Detection

3.2.1. Structure and Numerical Model

This case studies the performance of domain adaptation when the training data are generated using a numerical model but the testing data are from an experimental structure. A small-scale three-story structure is tested by (Figueiredo, Park, Figueiras, Farrar, & Worden, 2009) at the Los Alamos National Laboratory (see Figure 5). The structure is excited with an electromagnetic shaker attached to its base. The accelerations at each floor including the base are recorded at a sampling rate of 320 Hz for about 25 seconds. Seven damage conditions are considered where the stiffness of one or two out of four columns at different stories are reduced. Table 4 summarizes the damage conditions.

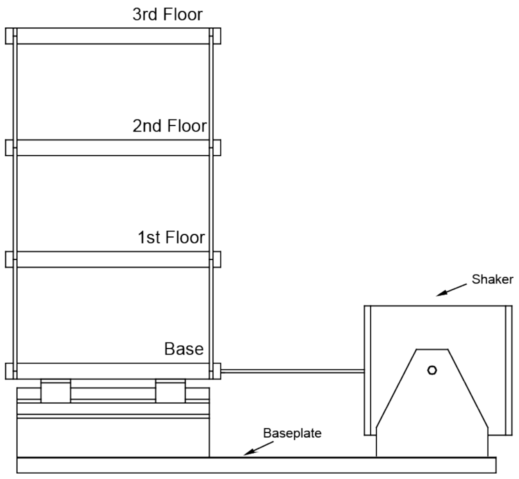


Figure 5. Three-story laboratory structure (Figueiredo et al., 2009)

Label	Damage Type
State #1	Baseline condition - Undamaged
State #2	87.5% stiff. red. in one column at first floor
State #3	87.5% stiff. red. in two columns at first floor
State #4	87.5% stiff. red. in one column at second floor
State #5	87.5% stiff. red. in two columns at second floor
State #6	87.5% stiff. red. in one column at third floor
State #7	87.5% stiff. red. in two columns at third floor

Table 4. Damage types for three-story structure

Our purpose for this case is developing a domain adaptation architecture that can transfer knowledge from the numerical model to experimental structure and predict the correct damage case. For this reason, we considered two sets of lumped-mass numerical models of the structure (see Table 6). First set is named as *low-fidelity model*, and it consists of two sub-model. The first sub-model, also called as *Simple*, establishes a 3-DOF system with the structural and geometric parameters provided by (Figueiredo et al., 2009). A comparison of natural frequencies identified by (Sun & Betti, 2015) and extracted from *Simple* model is given in Table 5. The average error of the natural frequencies for the given model is about 33 percent, and the largest error is 50 percent occurring for the first mode. The differences in the parameters imply that the *Simple* model does not represent the actual structure.

The second low-fidelity model is named *Updated*. The model updating procedure prescribed by (Giraldo, Yoshida, Dyke, & Giacosa, 2004) is performed on the *Simple* model to obtain this model. The method imposes a new stiffness matrix by utilizing the identified natural frequencies while the mass matrix remains the same. The *Updated* model has the same natural frequencies as the experimental structure, however, it is still 3-DOF. Thus, in this research, this model is still considered *low-fidelity*.

	Identified (Hz)	Simple (Hz)	Error (%)
1st Mode	31.09	14.32	53.94
2nd Mode	55.05	40.12	27.11
3rd Mode	72.23	57.98	19.73
		Average Error (%):	33.59

Table 5. A comparison of identified natural frequencies and Simple Model

The second model category is called *high-fidelity*. This model is 4-DOF and its design parameters such as mass and stiffness values are taken from (Sun & Betti, 2015) such that the natural frequencies and mode shapes of the model match closely to the experimental structure. To evaluate the sensitivity of modeling errors on the quality of damage prediction, two additional *high-fidelity* models are considered. The stiffness matrix of these models is perturbed by 5% and 10%, respectively, to simulate modeling errors.

All numerical models are simulated at 320 Hz in MATLAB. For these simulations, the base of the numerical model is excited with acceleration that matches the dynamic characteristics of the data captured at the base level from the experimental structure. For each damage case, the numerical model is damaged by the values given in Table 4. The three floor acceleration responses from both numerical and experimental data are sliced into 1-second bins. The numerical and experimental data have a dimension of $[42350 \times 320 \times 3]$ and $[8750 \times 320 \times 3]$ (# of data instances $\times$ # of samples $\times$ # of channel), respectively. After the data is split into training and testing, it is standardized, and the associated labels are one-hot encoded.

3.2.2. Implementation

For this case study, a slightly modified version of convolutional neural network architecture proposed by the (Lin, Nie, & Ma, 2017) for structural damage detection is used as the source-only model. In addition to 1D convolutional and max-pooling layers, this network utilizes batch normalization for stable learning and dropout for regularization. For DANN implementation, the domain classifier is added just before the label predictor, which is a set of densely connected layers (see Figure 6). The output shape of each layer set is provided at the corner of those layers. The gradient reversal layer is not shown to keep the illustration simple.

For each numerical model, both architectures are trained from scratch. Source-only architecture is trained with the labeled numerical model data (source), and DANN utilizes both labeled model data (source) and unlabeled experimental data (target). For each case, the architectures are trained three times, and the average performance is computed. The source-only and DANN models are trained for 200 and 50 epochs, respectively. It is observed that more epochs do not improve the performance of the DANN model.

3.2.3. Results

Table 6 presents the damage classification performance of source-only and DANN architectures for the five numerical models. Additionally, for each model, the improvement in accuracy from source-only to DANN architecture is given as the difference between their target prediction performance.

First, we look at the performance of low-fidelity models. The accuracy of both architectures is high ($> 90\%$) for the Simple Model against source data. However, the accuracy drops to 14% when tested against target data.For Simple Model, the domain adaptation is ineffective when the modeling error is significant and the numerical model is semantically very different than the experimental structure. As for the Updated Model, the accuracy of source-only model marginally increases against target data (17%). The improvement is more prominent for DANN (40%) which is above 100% increase in the performance compared to source-only architecture.

For the high-fidelity model category, three models are considered with various modeling errors: 0%, 5%, and 10%. Both source-only and DANN architectures perform excellently against source data independent of the model ($> 98\%$ accuracy). The overall results for source-only architecture and no modeling error show that while the high-fidelity model is not perfect, it can predict the correct classes for the experimental data with an accuracy reaching up to 90%. When tested against target data, the classification performance of source-only architecture decreases progressively with respect to the modeling error. The largest performance degradation is observed (81% accuracy) when the modeling error is 10%. When there is no modeling error, domain adaptation improves the damage detection marginally compared to source-only model (88% vs 89%). For small modeling error (5%), the classification accuracy increases slightly (86% vs $\sim$ 90%). Lastly, the most dramatic increase in performance is observed for large modeling error where the improvement is higher than 10%.

3.3. Discussion

To demonstrate the applicability of DANN, we consider two case studies. In the first case study, we predict the condition of a gearbox system running under high-load using the knowledge gained from low-load and high-load operation data. The second study focuses on transferring inference from labeled simulation data to unlabeled experimental data for a three-story structure. For the first case, the improve-

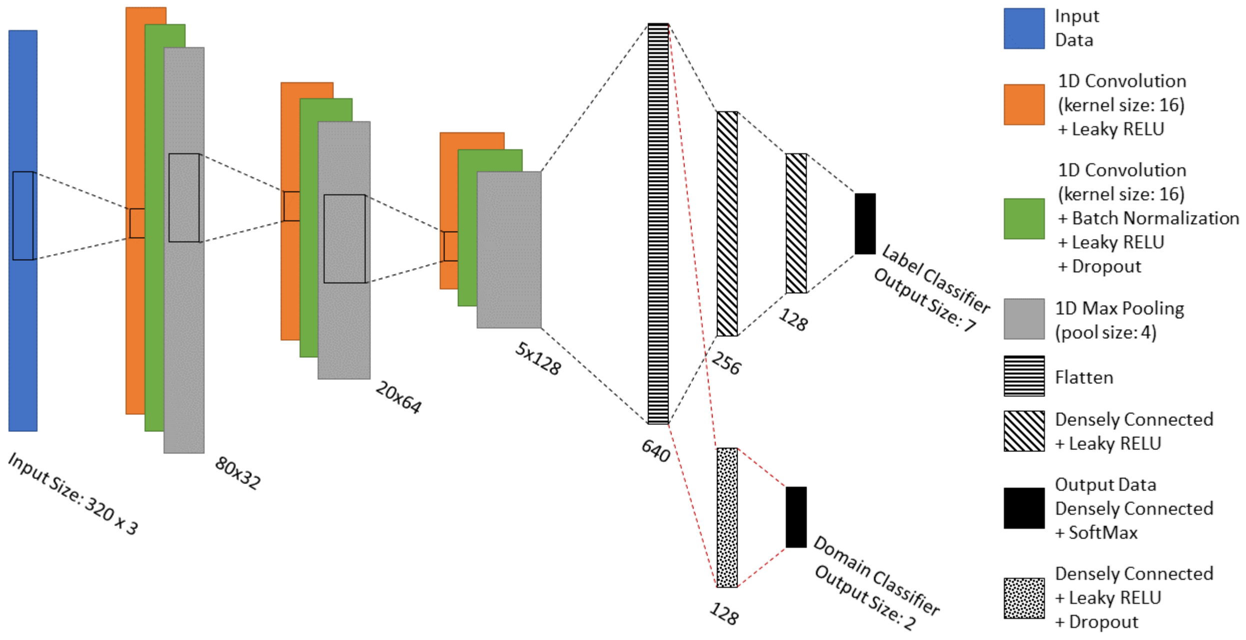

Input Size: 320 x 3
80x32
20x64
5x128
640
256
128
Label Classifier
Output Size: 7
128
Domain Classifier
Output Size: 2
Input Data
1D Convolution (kernel size: 16) + Leaky RELU
1D Convolution (kernel size: 16) + Batch Normalization + Leaky RELU + Dropout
1D Max Pooling (pool size: 4)
Flatten
Densely Connected + Leaky RELU
Output Data Densely Connected + SoftMax
Densely Connected + Leaky RELU + Dropout

Figure 6. DANN architecture for experimental example

		Low-Fidelity Model		High-Fidelity Model		
Architecture	Data	Simple	Updated	0% Error	5% Error	10% Error
Source-only	Source	96.33	94.01	98.99	99.22	99.21
	Target	14.40	17.08	88.35	86.49	80.88
DANN	Source	92.60	99.62	99.98	99.98	99.81
	Target	14.37	41.27	89.70	89.96	92.18
Improvement		-0.02	24.19	1.36	3.47	11.30

Table 6. Accuracy for domain adaption from numerical to experimental data (all values in percentage)

ment DANN provides is around 7%. Here, TCA produced low accuracy both for source and target domain data. Similarly, JDA did not exhibit a successful domain adaptation performance either. This could be attributed to the fact that only a quarter of the total data set (generated with stratified random sampling) is used for the training. Similar to Principal Component Analysis, TCA and JDA are taxing on the memory for large number of samples. Due to this limitation, the generalization over both source and target data set may not be well defined. Additionally, TCA and JDA use SVM and k-NN on dimension-reduced source domain datasets, respectively. SVM and k-NN may not be the most suitable classifier for this application.

For the second case, we observe an increase in the target accuracy varying between 1% and 24%. The results show that if there is a significant divergence between source and target domains (Simple Model), domain adaptation is not very successful. For semantically similar systems (Updated Model), the learning model produced with the numerical data fails to predict correct labels for the target data without proper domain adaptation. On the other hand, the DANN is able to improve the accuracy of the target data by 24%. For high-fidelity models, the performance of source-only model decrease with increasing modeling error. DANN improves the target prediction performance compared to the source-only model, especially when the modeling error is more prominent. Due to the poor performance of TCA and JDA in Case 1, we dismissed them for Case 2.

4. CONCLUSION

For many SHM methods based on supervised learning, experimental target data is often not available. For such cases, a classification model trained with simulation data may not generate correct predictions for real data. Without addressing the data shift between the source and target domain, it is challenging to learn a model that can be used for SHM. This paper shows that domain adaptation is a viable approach to damage classification problems. Specifically, we show the applicability of adversarial domain adaptation using two case studies. In the first case, we study the fault detection performance for a gearbox system between low-load (source) and high-load (target) domains and we observed that the prediction accuracy improves using domain adaptation. Additionally, we compared DANN to TCA and JDA to demonstrate the performance gain from DANN over the traditional domain adaptation methods. The second case focuses on detecting and locating damage for a three-story structure. Here, we utilized a numerical model of the structure for generating labeled source domain data and the experimental data for unlabeled target domain data. The results show that DANN increases classification performance.

The current approach processes source and target data separately during training. In reality, for the majority of structural health monitoring applications, the structure is expected to be in healthy condition right after the construction. Thus, target domain data labeled as *normal/undamaged* is accessible for training to some extent. For future research, novel domain adaptation methods should exploit this limited target domain data during training to extract more generalized latent features and to improve the adaptation. Lastly, other domain adaptation strategies such as GAN-based discriminate approaches (Tzeng, Hoffman, Saenko, & Darrell, 2017) should be also explored.

ACKNOWLEDGMENT

This work is supported in part by DARPA through contract number FA8750-18-C-0089. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor.

REFERENCES

Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M., & Inman, D. J. (2017). Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *Journal of Sound and Vibration*, *388*, 154–170.

American Society of Civil Engineers. (2013). *Failure to act.* `https://www.asce.org/uploadedFiles/Issues_and_Advocacy/Our_Initiatives/Infrastructure/Content_Pieces/failure-to-act-economic-impact-summary-report.pdf`. (Accessed: 2020-05-15)

American Society of Civil Engineers. (2017). *2017 infrastructure report card.* `https://www.infrastructurereportcard.org/wp-content/uploads/2019/02/Full-2017-Report-Card-FINAL.pdf`. (Accessed: 2020-05-15)

Benaim, S., & Wolf, L. (2017). One-sided unsupervised domain mapping. In *Advances in neural information processing systems* (pp. 752–762).

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, *79*(1-2), 151–175.

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., & Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, *22*(14), e49–e57.

Bouvier, V., Very, P., Hudelot, C., & Chastagnol, C. (2019). Hidden covariate shift: A minimal assumption for domain adaptation. *arXiv preprint arXiv:1907.12299*.

Catbas, N., Gokce, H. B., & Frangopol, D. M. (2013). Predictive analysis by incorporating uncertainty through a family of models calibrated with structural health-monitoring data. *Journal of Engineering Mechanics*, *139*(6), 712–723.

Chen, Z., Li, C., & Sanchez, R.-V. (2015). Gearbox fault identification and classification with convolutional neural networks. *Shock and Vibration*, *2015*.

Dackermann, U., Li, J., & Samali, B. (2013). Identification of member connectivity and mass changes on a two-storey framed structure using frequency response functions and artificial neural networks. *Journal of Sound and Vibration*, *332*(16), 3636–3653.

Farrar, C. R., & Worden, K. (2012). *Structural health monitoring: a machine learning perspective*. John Wiley & Sons.

Figueiredo, E., Moldovan, I., Santos, A., Campos, P., & Costa, J. C. (2019). Finite element–based machine-learning approach to detect damage in bridges under operational and environmental variations. *Journal of Bridge Engineering*, *24*(7), 04019061.

Figueiredo, E., Park, G., Figueiras, J., Farrar, C., & Worden, K. (2009). Structural health monitoring algorithm comparisons using standard data sets. *Los Alamos National Laboratory, Los Alamos, NM, Report No. LA-14393*.

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.

Gardner, P., Liu, X., & Worden, K. (2020). On the application of domain adaptation in structural health monitoring. *Mechanical Systems and Signal Processing*, *138*, 106550.

Giraldo, D., Yoshida, O., Dyke, S. J., & Giacosa, L. (2004). Control-oriented system identification using era. *Structural Control and Health Monitoring*, *11*(4), 311–326.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., & Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, *3*(4), 5.

Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., & Smola, A. J. (2007). Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems* (pp. 601–608).

Jing, L., Zhao, M., Li, P., & Xu, X. (2017). A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement*, *111*, 1–10.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2019). 1d convolutional neural networks and applications: A survey. *arXiv preprint arXiv:1905.03554*.

Li, J., Li, X., He, D., & Qu, Y. (2020). A domain adaptation model for early gear pitting fault diagnosis based on deep transfer learning network. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *234*(1), 168–182.

Li, X., Zhang, W., & Ding, Q. (2018). Cross-domain fault diagnosis of rolling element bearings using deep generative neural networks. *IEEE Transactions on Industrial Electronics*, *66*(7), 5525–5534.

Li, X., Zhang, W., Ding, Q., & Sun, J.-Q. (2019). Multi-layer domain adaptation method for rolling bearing fault diagnosis. *Signal processing*, *157*, 180–197.

Lin, Y.-z., Nie, Z.-h., & Ma, H.-w. (2017). Structural damage detection with automatic feature-extraction through deep learning. *Computer-Aided Civil and Infrastructure Engineering*, *32*(12), 1025–1046.

Long, M., Wang, J., Ding, G., Sun, J., & Yu, P. S. (2013). Transfer feature learning with joint distribution adaptation. In *Proceedings of the ieee international conference on computer vision* (pp. 2200–2207).

Lu, W., Liang, B., Cheng, Y., Meng, D., Yang, J., & Zhang, T. (2016). Deep model based domain adaptation for fault diagnosis. *IEEE Transactions on Industrial Electronics*, *64*(3), 2296–2305.

Mirzaee, A., Abbasnia, R., & Shayanfar, M. (2015). A comparative study on sensitivity-based damage detection methods in bridges. *Shock and Vibration*, *2015*.

Nick, W., Asamene, K., Bullock, G., Esterline, A., & Sundaresan, M. (2015). A study of machine learning techniques for detecting and classifying structural damage. *International Journal of Machine Learning and Computing*, *5*(4), 313.

Ozdagli, A. I., & Koutsoukos, X. (2019). Machine learning based novelty detection using modal analysis. *Computer-Aided Civil and Infrastructure Engineering*, *34*(12), 1119–1140.

Pan, S. J., Tsang, I. W., Kwok, J. T., & Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, *22*(2), 199–210.

Park, J.-H., Kim, J.-T., Hong, D.-S., Ho, D.-D., & Yi, J.-H. (2009). Sequential damage detection approaches for beams using time-modal features and artificial neural networks. *Journal of Sound and Vibration*, *323*(1-2), 451–474.

Phm data challenge 2009. (2009). `http://www.phmsociety.org/competition/09`. (Accessed: 2009-09-28)

Sejdinovic, D., Sriperumbudur, B., Gretton, A., & Fukumizu, K. (2013). Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, 2263–2291.

Singh, J., Azamfar, M., Ainapure, A., & Lee, J. (2020). Deep learning-based cross-domain adaptation for gearbox fault diagnosis under variable speed conditions.

Measurement Science and Technology, *31*(5), 055601.

Sohn, H., Farrar, C. R., Hemez, F. M., & Czarnecki, J. J. (2002). *A review of structural health review of structural health monitoring literature 1996-2001* (Tech. Rep.). Los Alamos, New Mexico: Los Alamos National Laboratory.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P. V., & Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems* (pp. 1433–1440).

Sun, H., & Betti, R. (2015). A hybrid optimization algorithm with bayesian inference for probabilistic model updating. *Computer-Aided Civil and Infrastructure Engineering*, *30*(8), 602–619.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7167–7176).

Wang, M., & Deng, W. (2018). Deep visual domain adaptation: A survey. *Neurocomputing*, *312*, 135–153.

Wilson, G., & Cook, D. J. (2020). A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *11*(5), 1–46.

World Bank. (2019). *2017 infrastructure report card.* `https://lpi.worldbank.org/international/global`. (Accessed: 2020-05-15)

Xie, J., Zhang, L., Duan, L., & Wang, J. (2016). On cross-domain feature fusion in gearbox fault diagnosis under various operating conditions based on transfer component analysis. In *2016 ieee international conference on prognostics and health management (icphm)* (pp. 1–6).

Zachariadis, I. A. (2018). *Investment in infrastructure in the eu: Gaps, challenges, and opportunities.* `https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_BRI(2018)628245`. (Accessed: 2020-05-15)

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the ieee international conference on computer vision* (pp. 2223–2232).

BIOGRAPHIES

Ali I. Ozdagli is a graduate student in the Department of Computer Science at Vanderbilt University. Prior to this, he got his M.S. in civil engineering from the University of Notre Dame and his Ph.D. from Purdue University. His research focuses on adapting machine learning approaches to structural health monitoring applications.

Xenofon Koutsoukos is a professor with the Department of Computer Science and a senior research scientist with the Institute for Software Integrated Systems, Vanderbilt University. His research work is in the area of cyber-physical systems with an emphasis on security and resilience, control, diagnosis and fault tolerance, formal methods, and adaptive resource management. He received the Ph.D. degree in electrical engineering from the University of Notre Dame. He is a Fellow of the IEEE.