HierMUD: Hierarchical multi-task unsupervised domain adaptation between bridges for drive-by damage diagnosis

Jingxiao Liu¹, Susu Xu², Mario Bergés³ and Hae Young Noh¹

Abstract

Monitoring bridges through vibration responses of drive-by vehicles enables efficient and low-cost bridge maintenance by allowing each vehicle to inspect multiple bridges and eliminating the needs for installing and maintaining sensors on every bridge. However, many existing drive-by monitoring approaches are based on supervised learning models that require massive labeled data from every bridge. It is expensive and time-consuming, if not impossible, to obtain these labeled data. Furthermore, directly applying a supervised learning model trained on one bridge to new bridges would result in low accuracy due to the shift between different bridges’ data distributions. Moreover, when we have multiple tasks (e.g., damage detection, localization, and quantification), the distribution shifts become more challenging than having only one task because different tasks have distinct distribution shifts and varying task difficulties. To this end, we introduce HierMUD, the first Hierarchical Multi-task Unsupervised Domain adaptation framework that transfers the damage diagnosis model learned from one bridge to a new bridge without requiring any labels from the new bridge. Specifically, our framework learns a hierarchical neural network model in an adversarial way to extract features that are informative to multiple tasks and invariant across multiple bridges. To match distributions over multiple tasks, we design a new loss function based on a newly derived generalization risk bound to adaptively assign higher weights to tasks with more shifted distributions. To learn multiple tasks with varying task difficulties, we split them into easy-to-learn and hard-to-learn tasks based on their distributions. Then, we formulate a feature hierarchy to utilize more learning resources to improve the hard-to-learn tasks’ performance. We evaluate our framework with experimental data from 2 bridges and 3 vehicles. We achieve up to 2X better performance than baseline methods, including average accuracy of 95% for damage detection, 93% for localization, and 0.38 lbs mean absolute error for quantification.

Keywords

Bridge health monitoring, indirect structural health monitoring, vehicle scanning method, vehicle-bridge interaction, transfer Learning, unsupervised domain adaptation, domain adversarial learning, multi-task learning

Introduction

Bridges are key components of transportation infrastructure. Aging bridges all over the world pose challenges to the economy and public safety. According to the 2016 National Bridge Inventory of the Federal Highway Administration, 139,466 of 614,387 bridges in the U.S. are structurally deficient or functionally obsolete.¹ The state of aging bridges demands researchers to develop efficient and scalable approaches for monitoring a large stock of bridges.

Currently, bridge maintenance is based on manual inspection,² which is inefficient, incurs high labor costs, and fails to detect damages in a timely manner. To address these challenges, structural health monitoring techniques,³ where structures are instrumented using sensors (e.g., strain gauge, accelerometers, cameras, etc.) to collect structural performance data, have been developed to achieve continuous and autonomous bridge health monitoring (BHM). Yet, such sensing methods are hard to scale up as they require on-site installation and maintenance of sensors on every bridge and...
cause interruptions to regular traffic for running tests and maintaining instruments.

To address the drawbacks of current BHM, drive-by BHM approaches were proposed to use vibration data of a vehicle passing over the bridge for diagnosing bridge damage. Drive-by BHM is also referred to as the vehicle scanning method or indirect structural health monitoring. Vehicle vibrations contain information about the vehicle-bridge interaction (VBI) and thus can indirectly inform us of the dynamic characteristics of the bridge for damage diagnosis. This is a scalable sensing approach with low-cost and low-maintenance requirements because each instrumented vehicle can efficiently monitor multiple bridges. Also, there is no need for direct installations and on-site maintenance of sensors on every bridge.

Previous drive-by BHM focuses on estimating bridge modal parameters (e.g., fundamental frequencies and damping coefficients) that can be used for detecting and localizing bridge damage. However, since the VBI system is a complex physical system involving many types of noise and uncertainties (e.g., environmental noise, vehicle operational uncertainties, etc.), such modal analysis methods for drive-by BHM are susceptible to them. This makes the modal parameters’ estimation inaccurate and limits the ability of drive-by BHM to diagnose bridge damage (e.g., localize and quantify damage severity).

More recently, data-driven approaches use signal processing and machine learning techniques to extract information from the vehicle acceleration signals. The extracted features are more robust to noise, enabling more sophisticated diagnoses such as damage localization and quantification. However, such data-driven approaches generally use supervised learning models developed on available labeled data (i.e., a set of bridges with known damage labels) that is expensive, if not impossible, to obtain. This labeled data requirement is further exacerbated by having multiple diagnostic tasks (e.g., damage detection, localization, and quantification). Furthermore, the standard supervised learning-based approaches learned using vehicle vibration data collected from one bridge are inaccurate for monitoring other bridges because data distributions of the vehicle passing over different bridges are shifted. Having to re-train for each new bridge in multiple bridge monitoring is time-consuming and costly. Therefore, for multiple bridge monitoring with multiple diagnostic tasks, one needs to transfer or generalize the multi-task damage diagnostic model learned from one bridge to other bridges, in order to eliminate the need for requiring training labeled data from every bridge in multiple tasks.

To this end, we introduce HierMUD, a new Hierarchical Multi-task Unsupervised Domain adaptation framework that transfers a model learned from one bridge data to predict multiple diagnostic tasks (including damage detection, localization, and quantification) in another bridge in an unsupervised way. Specifically, HierMUD makes a prediction for the target bridge, without any labels from the target bridge in any of the tasks. We achieve this goal by extracting features that are 1) informative to multiple tasks (i.e., task-informative) and 2) invariant across the source and target domains (i.e., domain-invariant).

Our framework is inspired by an unsupervised domain adaptation (UDA) approach that has been developed in the machine learning community to address the data distribution shift between two different domains, namely source and target domains. In our work, we denote the bridge with labeled data as “Source Domain,” and the new bridge of interest without any labels as “Target Domain.” UDA focuses on the unsupervised learning tasks in the target domain, which transfers the model learned using source domain data and labels (e.g., vehicle vibration data with the corresponding bridge damage labels) to predict tasks in the target domain without labels (e.g., vehicle vibration data from other bridges without knowing damage labels). In particular, domain adversarial learning algorithm extracts a feature that simultaneously maximizes the damage diagnosis performance for the source domain based on source domain labeled data while minimizing the performance of classifying which domain the feature came from using both source and target domain data. This algorithm can better match complex data distributions between the source and target domains by learning feature representations through neural networks, when compared to other conventional UDA approaches.

However, directly matching different domains’ data distributions over each diagnostic task separately through domain adversarial learning limits the overall performance of drive-by BHM, which has multiple diagnostic tasks. This is because these tasks are coupled/related with each other and share damage-sensitive information. The prediction performance of each task depends on that of other coupled tasks.

Therefore, our algorithm integrates multi-task learning (MTL) with domain-adversarial learning to fuse information from different bridges and multiple diagnostic tasks for effectively improving diagnostic accuracy and scalability of drive-by BHM. MTL is helpful because it simultaneously solves multiple learning tasks to improve the generalization performance of all the tasks, when compared to training the models for each task independently or sequentially (i.e., independent or sequential task learning).

Yet, when integrating MTL and domain-adversarial learning, two particular research challenges exist:

1) **Distinct distribution shift:** When we have multiple damage diagnostic tasks, the distribution shift problem becomes more challenging than having...
only one task because different tasks have distinct shifted distributions between the source and target domains. Therefore, it is important to develop efficient optimization strategies to find an optimal trade-off for matching different domains’ distributions over multiple tasks.

2) **Varying task difficulty:** The learning difficulties of different tasks vary with the complexity of mappings from the vehicle vibration data distribution to the damage label distributions. Some tasks (e.g., damage quantification task) would have highly non-linear mappings between data and label distributions and, therefore, be more difficult to learn than other tasks (i.e., hard-to-learn). Therefore, we need to distribute learning resources (e.g., representation capacities) according to the learning difficulty of tasks.

To address the distinct distribution shift challenge, we introduce a new loss function that prioritizes domain adaptations for tasks having more severe distribution shifts between different domains through a soft-max objective function. To formulate this new loss function, we first derive a new generalization risk bound for multi-task UDA, by optimizing which the new loss function is designed. Specifically, we minimize our loss function to jointly optimize three components: 1) feature extractors, 2) task predictors, and 3) domain classifiers. The parameters of task predictors are optimized to predict task labels in the source domain training set, which ensures that the extracted features are task-informative. The parameters of feature extractors are optimized with the domain classifiers in an adversarial way, such that the best trained domain classifier cannot distinguish which domains the extracted features come from. During the optimization, our new loss function adaptively weighs more on minimizing the distribution divergence of tasks having more shifted distributions between different domains. In this way, the model is optimized to automatically find a trade-off for matching different domains’ distributions over multiple tasks.

Further, to address the varying task difficulty challenge, we develop hierarchical feature extractors to allocate more learning resources to hard-to-learn tasks. We first split the multiple tasks into easy-to-learn tasks (e.g., damage detection and localization) and hard-to-learn tasks (e.g., damage quantification) based on their degrees of learning difficulty. Specifically, we model the learning difficulty of each task to be inversely proportional to performance in the source domain (e.g., damage localization and quantification accuracy in supervised learning settings). Then, the hierarchical feature extractors learn two-level features: task-shared and task-specific features. To achieve high prediction accuracy for multiple tasks without creating a complex model, we extract task-shared features from input data for easy-to-learn tasks and then extract task-specific features from the task-shared features for only the hard-to-learn tasks. In this way, we allocate more learning resources (learning deeper feature representations) to learn hard-to-learn tasks, which improves the overall performance for all the tasks.

We evaluate our framework on the drive-by BHM application using lab-scale experiments with two structurally different bridges and three vehicles of different weights. We introduce damage by attaching a mass to the bridge as a damage proxy. By modifying the location and size of the attached mass, we can non-destructively assess the sensitivity of a diagnostic algorithm to changes in the bridge. This damage proxy has been used by researchers for validating damage diagnostic algorithms prior to their use in actual systems. In the evaluation, we train our framework using labeled data collected from a vehicle passing over one bridge to diagnose damage in another bridge with unlabeled vehicle vibration data. Our framework outperforms five baselines without UDA, MTL, the new loss function, or hierarchical structure.

In summary, this paper has three main contributions:

1) We introduce HierMUD, a new multi-task UDA framework that transfers the model learned from one bridge to achieve multiple damage diagnostic tasks in another bridge without any labels from the target bridge in any of the tasks. To the best of our knowledge, this new framework is the first domain adaptation framework for multi-task bridge monitoring. We have since released a PyTorch implementation of HierMUD at https://github.com/jingxiaoliu/HierMUD.

2) We derive a generalization risk bound that provides a theoretical guarantee to achieve domain adaptation on multiple learning tasks. This work bridges the gaps between the theories and algorithms for multi-task UDA. Based on this bound, we design a new loss function to find a trade-off for matching different domains’ distributions over multiple tasks, which addresses the distinct distribution shift challenge.

3) We develop a hierarchical architecture for our multi-task and domain-adversarial learning algorithm. This hierarchical architecture ensures that the framework accurately and efficiently transfers the model for predicting multiple tasks in the target domain, which addresses the varying task difficulty challenge.

The remainder of this paper is divided into six sections. In section 2, we study the MTL and data distribution shift challenges in the drive-by BHM application. Section 3 derives the generalization risk bound for multi-task UDA. Section 4 presents our HierMUD framework, which
includes the description of our framework, loss function, and algorithm design. Section 5 describes the evaluation of our framework on the drive-by BHM application, following by Section 6 that presents the evaluation results. Section 7 concludes our work and provides discussions about future work.

**Data distribution shift and multi-task learning challenges for the drive-by bridge health monitoring**

In this section, we describe the physical insights that enable our drive-by BHM and explain the associated challenges in achieving this scalable BHM approach. We first characterize the structural dynamics of the VBI system. Next, we study the multi-task learning and data distribution shift challenges, respectively, by proving the error propagation between multiple damage diagnostic tasks and characterizing the shifting of the joint distribution of vehicle vibration and damage labels.

**Characterizing the structural dynamics of the VBI system**

To provide physical insights of drive-by BHM, we model the VBI system, as shown in Figure 1, as a sprung mass (representing the vehicle) traveling with a constant speed on a simply supported beam (representing the bridge). We assume the beam is of the Euler-Bernoulli type with a constant cross section. We also assume that there is no friction force between the ‘wheel’ and the beam. The damage is simulated by attaching a mass (magnitude/severity level: \( q \)) at location \( l \) on the beam. The added mass changes the mass of the bridge and its dynamic characteristics.\(^{45} \)

Modifying the weight of the attached mass is a non-destructive way of creating physical changes to the VBI system to mimic structural damage.\(^ {33-45} \)

In our prior works,\(^ {10,42} \) we have derived the theoretical formulation of the VBI system in the frequency domain, which is summarized in the following paragraphs.

The equations of motion for the vehicle and bridge in the time domain are first derived as

\[
\begin{align*}
\dot{u}_d(t) + k_v [u_d(t) - y_d(x = vt,t) - y_{st}(x = vt)] + c_v \left[ \dot{y}_d(t) - \dot{y}_s(x = vt,t) - \dot{y}_{st}(x = vt) \right] &= 0 \\
\rho A \ddot{y}_d(x,t) + q \dot{y}_d(x,t) \delta(x - l) + E I \phi''_d(x,t) &= \left\{ -m_s g + k_v [u_d(t) - y_d(x = vt,t) - y_{st}(x = vt)] + c_v \left[ \ddot{u}_d(t) - \dot{y}_d(x = vt,t) - \dot{y}_{st}(x = vt) \right] \right\} \delta(x - vt)
\end{align*}
\]

where \( m_s, k_v, c_v, \) and \( u \) are the mass, stiffness, damping coefficient, and total displacement of the vehicle, respectively; \( \rho, A, E, I, \) and \( y_{st} \) are the density, sectional area, Young’s modulus, moment of inertia, and the static displacement of the bridge, respectively; \( \delta(x - vt) \) is the Dirac delta function; and \( u_d(t) \) and \( y_d(t) \) are the dynamic displacements of vehicle and bridge, respectively.

Then the \( n \)-th mode frequency response of the vehicle’s acceleration is

\[
\ddot{U}_{d,n}(\omega) = \ddot{U}_{d,n}\left(\omega - \frac{2n\pi v}{L}\right) - \frac{\omega^4 E I - i \left( \omega - \frac{n\pi v}{L} \right)^2} {2\pi m_s L^3 \left( \omega - \frac{n\pi v}{L} \right)^2 \sin \left( \frac{n\pi}{2} \right)} \left( \omega - \frac{n\pi v}{L} \right) \\
- \sqrt{2g} \left[ \Delta \left( \omega - \frac{2n\pi v}{L} \right) - \Delta(\omega) \right]
\]

where \( i \) is the imaginary number; \( \ddot{Y}_{d,n}(x, \omega) \) is the \( n \)-th mode frequency response function (FRF) of the bridge element acceleration at the location \( x \); and \( U_{d,n}(\omega) \) and \( \Delta(\omega) \)

![Figure 1](image-url)  
**Figure 1.** The vehicle-bridge interaction system with a surrogate damage simulated by attaching a mass having magnitude \( q \) at location \( l \).
are the $n$-th mode FRF of the vehicle acceleration and the FRF of the Dirac delta function, respectively.

From Equation (3), we obtain the following important physical understandings of the drive-by vehicle vibration:

1) **Non-linear property**: Vehicle acceleration $(\ddot{U}_{\omega,\nu}(\omega))$ is a high-dimensional signal that has a complex non-linear relationship with bridge properties ($\rho$, $A$, $E$, $l$) and damage parameters ($q$, $l$). Thus, it is difficult to infer damage states for different bridges by directly analyzing the raw vehicle signals. It is important to model features that can represent the non-linearity of the VBI system.

2) **Coupled diagnostic tasks**: Different damage locations ($l$) and severity levels ($q$) only vary the term $q\sin^2\left(\frac{\pi l}{2L}\right)$, representing structural dynamic characteristic changes due to damage. The damage localization and quantification tasks are coupled with each other through the same damage information $d$, and thus the estimation of them depends on each other’s estimation.

We incorporate these physical insights of the VBI system to develop our multi-task UDA framework. The non-linear property instructs us to use non-linear models or extract non-linear features from the vehicle vibrations for estimating bridge damage. In this work, we use a neural network-based model to non-linearly extract task-informative features from vehicle vibration data.

Moreover, the coupled tasks property of the VBI system informs us to use or extract the shared damage information (e.g., task-shared features) instead of independently learning multiple diagnostic tasks. This is further discussed in the following subsection.

**Error propagation between multiple tasks for a VBI system**

The shared information among multiple tasks can be learned simultaneously from multiple tasks or learned sequentially from one task to the next. In this section, we illustrate that simultaneous learning (i.e., MTL) is more accurate than sequential learning through a theoretical study of the VBI system. The study shows that the sequential learning method results in a significant error propagation from the previous task to the next.

For instance, if we localize and quantify the bridge damage sequentially (localize the damage first and then quantify the severity of the damage at the obtained damage location), the estimation of damage severity $q$ is $d/\sin^2\left(\frac{\pi \hat{l}}{2L}\right)$, where $\hat{l}$ is the estimated damage location. Then, the propagation of error from the damage location estimation to the severity estimation is

$$
\sigma_q = \pm \sqrt{\frac{1}{\sin^4\left(n\pi \hat{l}/L\right)}} + \frac{4n^2\pi^2d^2L^2\cos^2\left(n\pi \hat{l}/L\right)}{L^2\sin^6\left(n\pi \hat{l}/L\right)}
$$

(4)

where $\sigma_q$, $\sigma_l$, $\sigma_d$ are errors of severity, location, and damage information estimations, respectively. $\sin^4\left(n\pi \hat{l}/L\right)$ and $\sin^6\left(n\pi \hat{l}/L\right)$ in the denominators are smaller than 1, which makes the estimation error of $q$ very large as their values decrease. Especially, when the damage is close to the bridge supports (i.e., $|\hat{l} - L/2| \to L/2$), it leads to $\sin\left(n\pi \hat{l}/L\right) \to 0$ and $\sigma_q \to \infty$. Thus, damage location estimation error propagates, which results in a very inaccurate estimation of damage severity level.

To this end, we solve multiple tasks simultaneously, which can improve the overall accuracy by minimizing error propagation and learning the shared information (e.g., task-shared feature representations) from the coupled tasks.

Besides simultaneously learning multiple tasks, a scalable drive-by BHM approach needs to work for multiple domains (i.e., bridges). The following subsection discusses the data distribution shift challenge for drive-by monitoring of multiple bridges.

**Data distribution shift for VBI systems**

The joint distributions of vehicle vibrations and damage labels are shifted as the vehicle passes by different bridges. If we consider the process of the VBI system as a stochastic process, according to Equation (3), the joint distributions of the vehicle accelerations $(\ddot{U}_{\omega,\nu}(\omega))$ and damage labels ($q$ or $l$) changes non-linearly with bridge properties (e.g., $\rho$, $A$, $E$, $l$). An example of the shifting of these joint data distributions for different bridges is visualized in a low-dimensional space in Figure 2. It shows a two-dimensional t-Distributed Stochastic Neighbor Embedding (tSNE)$^{48}$ visualization of vehicle vibration data distributions for Bridges#1 and #2. Each vibration signal is collected from a vehicle passing over an 8-feet bridge model with a damage at the location ($l$) of 2 feet, 4 feet, or 6 feet. The data for the two structurally different bridges (Bridge#1 and #2) are represented by filled and unfilled markers, respectively. More details of this experiment and dataset are in Evaluation Section. We can observe from Figure 2 that directly applying the model learned from one bridge’s dataset (e.g., Bridge#1) to
A generalization risk bound for multi-task unsupervised domain adaptation

In this section, we derive the upper bound of the generalization risk for multi-task UDA problems to investigate the theoretical guarantee of its performance on target domain unseen data. The generalization risk (or error) of a model is the difference between the empirical loss on the training set and the expected loss on a test set, as defined in statistical learning theory. In other words, it represents the ability of the trained model to generalize from the training dataset to a new unseen dataset. In our problem, the generalization risk is defined to represent how accurately a classifier trained using source domain labeled data and target domain unlabeled data predicts class labels in the target domain. Therefore, deriving the upper bound of the generalization risk provides insights on how to develop learning algorithms to efficiently optimize it.

We first derive a generalization risk bound for UDA and then integrate it with the risk bound for MTL. Next, we characterize the newly derived generalization risk bound to provide insights to our multi-task UDA problem.

A generalization risk bound for unsupervised domain adaptation

We first derive a new generalization risk bound for UDA by representing the original data distribution in an intermediate feature space, which has been ignored by the existing risk bounds for UDA. Having a feature space enables the modeling of task-shared feature representation when we have multiple tasks. This results in a tighter generalization risk bound for multi-task UDA than independently estimating each task’s generalization risk bound. Yet, this feature space requires us to estimate the discrepancy between the marginal feature distributions of the source and target domains for obtaining the generalization risk bound, which is introduced in this section.

We consider a classification task that labels input $X$ as belonging to different classes $Y$. We also consider mappings $X \xrightarrow{w} Z \xrightarrow{h} Y$, where $X$, $Z$, and $Y$ are random variables of input, feature representation, and class label, which are taken from the input, feature, and output space $\mathcal{X}$, $\mathcal{Z}$, and $\mathcal{Y}$, respectively. The function $w: \mathcal{X} \rightarrow \mathcal{Z}$ is a $k$-dimension feature transformation, and the function $h: \mathcal{Z} \rightarrow \mathcal{Y}$ is a hypothesis on the feature space (i.e., a labeling function). Then, we have a predictor $h \circ w$, that is, $(h \circ w) (x) = h (w(x))$, for every $x \in \mathcal{X}$.
Further, we define a domain as a distribution on the input space $X$ and the output space $Y$. UDA problems involve two domains, a source domain $D_S$ and a target domain $D_T$. We denote $D_S^X$, $D_S^Y$, $D_T^X$, and $D_T^Y$ as the marginal data ($X$) and label ($Y$) distributions in the source ($D_S$) and target ($D_T$) domains, respectively. Note that we also have feature representation distributions, $D_S^X$ and $D_T^X$, in the source and target domains. Mathematically, an unsupervised domain adaptation algorithm has independent and identically distributed (i.i.d.) labeled source samples $<X_S, Y_S>$ drawn from $D_S$ and i.i.d. unlabeled target samples $X_T$ drawn from $D_T$, as shown below

$$<X_S, Y_S> = \{x_i, y_i\}_{i=1}^{n_S} \sim D_S;$$

$$X_T = \{x_i\}_{i=1}^{n_T} \sim D_T^X$$

where $n_S$ and $n_T$ are the number of samples in the source and target domains, respectively. The goal of UDA is to learn $h$ and $w$ with a low target domain risk under distribution $D_T$, which is defined as: $\epsilon_T(h \circ w; f_T) = \Pr_{x \sim D_T^X}(h \circ w(x) \neq f_T(x))$, where $f_T$ is the ground truth labeling function and $y = f_T(x)$ for $(x,y) \sim D_T$.

Since we do not have labeled data in the target domain, we cannot directly compute the target domain risk. Therefore, the upper bound of the target domain risk is estimated by the source domain risk and the discrepancy between the marginal data distributions of the source and target domains, $D_S^X$ and $D_T^X$. The discrepancy between $D_S^X$ and $D_T^X$ is quantified through the $\mathcal{H}\Delta\mathcal{H}$-divergence $\Delta$ that measures distribution divergence with finite samples of unlabeled data from $D_S^X$ and $D_T^X$. It is defined as

$$\Delta_{\mathcal{H}\Delta\mathcal{H}}(D_S^X, D_T^X) = 2 \sup_{h, h' \in \mathcal{H}} \Pr_{x \sim D_S^X}[h(x) \neq h'(x)] - \Pr_{x \sim D_T^X}[h(x) \neq h'(x)],$$

where $\mathcal{H}$ is a hypothesis space, and $\mathcal{H}\Delta\mathcal{H}$ is the symmetric difference hypothesis; $h'$ and $h$ are two samples of hypotheses randomly drawn from the hypothesis set $\mathcal{H}$.

Then, we can derive the generalization bound for UDA in Theorem 1.

**Theorem 1.** Let $\mathcal{W}$ be a hypothesis space on $X$ with Vapnik-Chervonenkis dimension (VC dimension) $d_\mathcal{W}$ and $\mathcal{H}$ be a hypothesis space on $Z$ with VC dimension $d_H$, where VC dimension is a measure of the complexity of a set of functions that a classification algorithm can learn. If $X_S$ and $X_T$ are samples of size $N$ from $D_S^X$ and $D_T^X$, respectively, and $Z_S$ and $Z_T$ follow distributions $D_S^Z$ and $D_T^Z$, respectively, then for any $\delta \in (0, 1)$ with probability at least $1 - \delta$, for every $h \in \mathcal{H}$ and $w \in \mathcal{W}$

$$\epsilon_T(h \circ w; f_T) \leq \epsilon_S(h \circ w; f_S) + 2\epsilon_S(h \circ w; f_S) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(Z_T, Z_S) + O\left(\sqrt{\frac{2d_H \log(2N) + \log(2/\delta)}{N}}\right)\left(\frac{2d_H \log(2N) + \log(2/\delta)}{N}\right)$$

Proof. See Appendix A.

We prove in Theorem 1 that the upper bound of the target domain risk consists of five components:

1) The source domain risk

$$\epsilon_S(h \circ w; f_S) + 2\epsilon_S(h \circ w; f_S),$$

which quantifies the error for estimating class labels in the source domain.

2) The minimal risk

$$\epsilon_T(h^* \circ w^*; f_T) + \epsilon_S(h^* \circ w^*; f_S),$$

which quantifies the error for estimating class labels using the ideal joint hypothesis over the source and target domains. It is the smallest error we can achieve using the best predictor in the hypothesis set.
3) The empirical symmetric divergence between marginal feature distributions

\[
\frac{1}{2} d_{HY}(Z, Z_S),
\]

which quantifies the distribution difference between the source and target domain marginal feature distributions.

4) The supremum of empirical symmetric divergence set between marginal data distributions

\[
\frac{1}{2} \sup_{h \in \mathcal{H}} \left[ \delta_{kW}(X_S, X_T) \right],
\]

which quantifies the distribution difference between the source and target domain marginal data distributions.

5) The Big-O terms that measure the complexity of the estimation of divergence.

Next, in the following subsection, we derive a generalization risk bound for multi-task UDA problems by considering the feature space \( Z \) being the task-shared feature space for multiple tasks.

**Integrating multi-task learning bound with the unsupervised domain adaptation bound**

We first consider multiple classification tasks that label input \( X_m \) as belonging to different classes \( Y_m \), for \( m = 1, 2, \ldots, M \), where \( M \) is the total number of tasks. The mappings for this multi-task learning problem becomes

\[
X_m \xrightarrow{w} Z_m \xrightarrow{h_m} Y_m, \quad m = 1, 2, \ldots, M,
\]

where \( X_m \), \( Z_m \), and \( Y_m \) are the \( m \)-th task’s random variables of input, feature representation, and class label, respectively, which are taken from the input, feature, and output space \( X, Z, \) and \( Y \), respectively. The function \( w \) is a task-shared \( k \)-dimensional feature transformation, and the function \( h_m \) is a task-specific hypothesis for the \( m \)-th task on the feature space. For each task, we have a hypothesis set \( h_m \in \mathcal{H} \). We define the task-averaged true risk under the joint distribution \( \prod_{m=1}^{M} D_m \) as

\[
\epsilon_{avg}(w, h_1, \ldots, h_M; f_1, \ldots, f_M) = \frac{1}{M} \sum_{m=1}^{M} \Pr_{x \sim D_m^w}(h_m \circ w(x) \neq f_m(x)),
\]

where \( f_m \) is the ground truth labeling function for the \( m \)-th task and \( y = f_m(x) \) for \( (x, y) \sim D_m \). We also define the task-averaged empirical risk as

\[
\epsilon_{avg}(\hat{w}, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{Y}) = \frac{1}{NM} \sum_{m=1}^{M} \sum_{i=1}^{N} I(h_m(\hat{w}(x_{m,i})) \neq y_{m,i}),
\]

where \( N \) is the total number of samples in each task, which is assumed to be the same for each task, and \( I(\cdot) \) is the indicator function. We define that \( (X, Y) = (X_1, \ldots, X_M, Y_1, \ldots, Y_M) \) are i.i.d. samples drawn from the joint distribution \( \prod_{m=1}^{M} D_m \). For the \( m \)-th task

\[
(X_m, Y_m) = \left\{ (x_{m,i}, y_{m,i}) \right\}_{i=1}^{N} \sim D_m.
\]

Further, we consider a multi-task UDA problem, which has labeled samples \( (X_S, Y_S) = (X_{S,1}, \ldots, X_{S,M}, Y_{S,1}, \ldots, Y_{S,M}) \) drawn from a joint source domain \( \prod_{m=1}^{M} D_{S,m} \) and unlabeled samples \( X_T = (X_{T,1}, \ldots, X_{T,M}) \) drawn from a joint target domain \( \prod_{m=1}^{M} D_{T,m} \). The goal of multi-tasks UDA is to learn \( h_1, \ldots, h_M \) and \( w \) with a low target domain task-averaged risk under the joint distribution \( \prod_{m=1}^{M} D_{T,m} \), which is defined as:

\[
\epsilon_{avg,T}(w, h_1, \ldots, h_M; f_1, \ldots, f_M).
\]

Our generalization risk bound for multi-task UDA is built on our Theorem 1 by combining it with the risk bound for multi-task learning. The multi-task learning risk bound was introduced in the work of Maurer et al., which showed that the upper bound of the task-averaged empirical risk, the complexity measure relevant to the estimation of the representation, and the complexity measure of estimating task-specific predictors. Specifically, with probability at least \( 1 - \delta \), where \( \delta \in (0, 1) \), in the draw of \( (X, Y) = (X_1, \ldots, X_M, Y_1, \ldots, Y_M) \sim \prod_{m=1}^{M} D_m \), it holds for every \( w \in W \) and every \( h_1, \ldots, h_M \in \mathcal{H} \)

\[
\epsilon_{avg}(w, h_1, \ldots, h_M; f_1, \ldots, f_M) \leq \epsilon_{avg}(\hat{w}, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{Y}) + c_1 \left( \frac{L \hat{G}(W(X))}{NM} + c_2 \right) + c_2 \frac{9 \log(2/\delta)}{2NM},
\]

where \( L \) is the Lipschitz constant for \( h \in \mathcal{H} \); \( c_1 \) and \( c_2 \) are universal constants; \( \hat{G}(W(X)) \) is the Gaussian average that measures the empirical complexity relevant to the estimation of the feature representation; and

\[
Q = \sup_{x \neq x'} \frac{1}{|x - x'|} \mathbb{E}_{h \in \mathcal{H}} \left[ \sum_{i=1}^{N} \gamma_i(h(x_i) - h(x'_i)) \right], \text{ } \gamma_i \text{ are independent standard Gaussian random variables.}
Now, by integrating the multi-task learning risk bound in Equation (10) with the unsupervised domain adaptation risk bound in Theorem 1, we obtain the following theorem:

**Theorem 2.** Let $D_{S1}, \ldots, D_{SM}$ and $D_{T1}, \ldots, D_{TM}$ be probability measures on $(X, Y)$. Let $W$ be a hypothesis space on $X$ with VC dimension $d_W$ and $H$ be a hypothesis space on $Z$ with VC dimension $d_H$. Let $\delta \in (0, 1)$. With probability at least $1 - \delta$ in the draw of $(X_s, Y_s) \sim D_{SM}$, $X_T \sim \prod_{m=1}^{M} D_{TM}^Z$ (i.e., $(X_{Sm}, Y_{Sm}) \sim D_{Sm}$ and $X_{Tm} \sim D_{TM}^Z$ for $m = 1, \ldots, M$), and $Z_S, Z_T$ that follow distributions $\prod_{m=1}^{M} D_{SM}^Z$ and $\prod_{m=1}^{M} D_{TM}^Z$, it holds for every $w \in W$ and every $h_1, \ldots, h_M \in H$ that

$$
\mathbb{E}_{a, r}(w, h_1, \ldots, h_M; f_{T1}, \ldots, f_{TM}) \leq \mathbb{E}_{a, r}(w, h_1, \ldots, h_M; f_{T1}, \ldots, f_{TM}),
$$

where

$$
w^*, h_{M}^{*} = \arg\min_{w^* \in W, h_{M}^{*} \in H} \{ \mathbb{E}_{a, r}(w^*, h_{1}, \ldots, h_{M}; f_{T1}, \ldots, f_{TM}) \}.
$$

We show in Theorem 2 that the upper bound of task-averaged target domain risk contains seven components:

1. **The source domain empirical risks**

$$
\mathbb{E}_{a, S}(w, h_{1}, \ldots, h_{M}; f_{T1}, \ldots, f_{TM})
$$

2. **the task-averaged minimal risks**

$$
\mathbb{E}_{a, r}(w^*, h_{1}, \ldots, h_{M}^{*}; f_{T1}, \ldots, f_{TM}),
$$

3. **the complexity measure relevant to the estimation of the representation**

$$
\frac{L \mathcal{G}(W(X_S))}{\sqrt{NM}}
$$

4. **the complexity measure of estimating task-specific predictors**

$$
\frac{Q \sup_{w \in W} \|w(X_S)\|}{\sqrt{NM}}
$$

5. **the empirical symmetric divergence between marginal feature distributions**

$$
\frac{1}{2} d_{\text{H}N}(Z_S, Z_T)
$$

6. **the supremum of empirical symmetric divergence set for multiple tasks between marginal data distributions**

$$
\sup_{h_1, \ldots, h_M \in H} \left[ \frac{1}{2} d_{\text{H}N}(X_S, X_T) \right]
$$

7. **the Big-O complexity measures of the estimation of divergence.**

Once we determine the hypothesis sets $H$ and $W$, the task-averaged minimal risk and complexity terms are fixed. Therefore, we can minimize the target domain risk bound (Equation 11) by minimizing the sum of the source domain empirical risks, the empirical divergence between marginal data distributions, and the empirical divergence between marginal feature distributions.
minimize \( \epsilon_{\text{avg},S} \left( \hat{\omega}, \hat{h}_1, \ldots, \hat{h}_M, \mathcal{T}_S \right) \)
+ \( 2\epsilon_{\text{avg},S} \left( w^*, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{T}_S \right) \)
+ \( \frac{1}{2} \mathcal{D}_{\text{HATA}} \left( \mathcal{Z}_S; \mathcal{Z}_T \right) \)
+ \( \sup_{h_1 \cdots h_k \in \mathcal{H}} \left[ \frac{1}{2} \mathcal{D}_{\text{HWAV}} \left( \mathcal{X}_S; \mathcal{X}_T \right) \right] \) \hspace{1cm} (15)

To efficiently solve Equation (15) that minimizes the generalization risk bound for multi-task UDA, we interpret and characterize the bound in the following subsection.

**Characterizing the derived risk bound**

We can learn two main insights of the new generalization bound for multi-task UDA from Equation (15):

1) **Feature divergence minimization:** Some UDA methods\(^{30, 32}\) minimize the empirical symmetric divergences between marginal feature distributions (i.e., \( \frac{1}{2} \mathcal{D}_{\text{HATA}} \left( \mathcal{Z}_S; \mathcal{Z}_T \right) \)) to make the task-specific classifiers (i.e., \( \hat{h}_1, \ldots, \hat{h}_M \)) invariant across domains. However, these methods are not scalable as the number of tasks grows because they require every task-specific classifier to be domain-invariant. Therefore, our multi-task UDA approach avoids directly minimizing the feature divergence, which requires adapting classifiers of each task separately.

2) **Data divergence minimization:** Some UDA methods\(^{31, 34}\) minimize the empirical symmetric divergence between marginal data distributions (i.e., \( \sup_{h_1 \cdots h_k \in \mathcal{H}} \left[ \frac{1}{2} \mathcal{D}_{\text{HWAV}} \left( \mathcal{X}_S; \mathcal{X}_T \right) \right] \)) to extract domain-invariant features. Such methods would be successful and scalable if the feature distributions \( \mathcal{D}_S^z \) and \( \mathcal{D}_T^z \) are matched because in this case the empirical symmetric divergence between marginal feature distributions (i.e., \( \frac{1}{2} \mathcal{D}_{\text{HATA}} \left( \mathcal{Z}_S; \mathcal{Z}_T \right) \)) would be very small or even be zero. However, directly minimizing this supremum divergence is difficult and data inefficient under the MTL setting because different tasks have distinct distribution shifts between the source and target domains. Therefore, we introduce a new efficient optimization strategy to find an optimal trade-off for minimizing the data divergence over multiple tasks.

In summary, to develop an algorithm that is scalable to the number of tasks, we need to minimize the empirical divergence between marginal data distributions to learn a feature mapping that matches feature distributions between different domains. Therefore, we can rewrite Equation (15) as

\[
\min_{\hat{\omega}, \hat{h}_1, \ldots, \hat{h}_M} \left( \epsilon_{\text{avg},S} \left( \hat{\omega}, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{T}_S \right) \right) + 2\epsilon_{\text{avg},S} \left( w^*, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{T}_S \right) + \frac{1}{2} \mathcal{D}_{\text{HATA}} \left( \mathcal{Z}_S; \mathcal{Z}_T \right) + \sup_{h_1 \cdots h_k \in \mathcal{H}} \left[ \frac{1}{2} \mathcal{D}_{\text{HWAV}} \left( \mathcal{X}_S; \mathcal{X}_T \right) \right]
\]

For a classification problem, minimizing the first two terms in Equation (16) can be achieved by minimizing the cross-entropy loss between predicted labels and ground truth labels in source domain: \( \hat{\epsilon}_{\text{avg},S} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{(x,y) \in \mathcal{S}} \sum_{c=1}^{C_m} \mathbb{I}(y = c) \log \hat{h}_m \left( \hat{\omega}(x) \right) \),

where \( C_m \) is the number of classes for the \( m \)-th tasks.

Further, if we consider \( \hat{h}_1, \ldots, \hat{h}_M \) are hypotheses independently drawn from the hypothesis class \( \mathcal{H} \), we can write the last term in Equation (16) as

\[
\max_{h \in \mathcal{H}} \sup_{m \in [M]} \left[ \frac{1}{2} \mathcal{D}_{w \mathcal{H} \mathcal{A} \mathcal{W}} \left( \mathcal{X}_{S,m}; \mathcal{X}_{T,m} \right) \right] \quad (17)
\]

without loss of generality. This means that we can minimize the maximum divergence between marginal feature distributions over all the tasks to achieve the minimization of the divergence term in Equation (16). An approximation of the empirical symmetric divergence between distributions is computed by learning a domain discriminator \( (h_m^{\hat{w}}) \) that distinguishes samples from different domains\(^{31}\)

\[
d_{w \mathcal{H} \mathcal{A} \mathcal{W}} \left( \mathcal{X}_{S,m}; \mathcal{X}_{T,m} \right) = 2 \left( 1 - \min_{w \in \mathcal{H} \mathcal{A} \mathcal{W}} \left( \frac{1}{N} \sum_{x \sim S,m} \mathbb{I}(h_m(w(x) = 1) \right) + \frac{1}{N} \sum_{x \sim T,m} \mathbb{I}(h_m(w(x) = 0) \right) \right).
\]

To this end, there are three ways to minimize Equation (16):

1) **Hard-max objective:** directly minimizing the maximum divergence over all \( M \) tasks

\[
\min_{\hat{\omega}, \hat{h}_1, \ldots, \hat{h}_M} \left( \epsilon_{\text{avg},S} \left( \hat{\omega}, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{T}_S \right) \right) + 2\epsilon_{\text{avg},S} \left( w^*, \hat{h}_1, \ldots, \hat{h}_M; \mathcal{T}_S \right) + \frac{1}{2} \mathcal{D}_{\text{HATA}} \left( \mathcal{Z}_S; \mathcal{Z}_T \right) + \max_{h \in \mathcal{H}} \sup_{m \in [M]} \left[ \frac{1}{2} \mathcal{D}_{w \mathcal{H} \mathcal{A} \mathcal{W}} \left( \mathcal{X}_{S,m}; \mathcal{X}_{T,m} \right) \right]
\]
2) **Average objective**: minimizing the average divergence:

\[
\begin{align*}
\text{minimize} & \quad \left( \hat{\epsilon}_{\text{avg}, S} + \frac{1}{2M} \sum_{m=1}^{M} \sup_{h_m \in \mathcal{H}} d_{-h_m, \mathcal{Y} \mathcal{M} \mathcal{V}}(X_{S,m}, X_{T,m}) \right) ; \\
\end{align*}
\]

(20)

3) **Soft-max objective**: minimizing a soft maximum, that is, a LogSumExp, of Equation (18)

\[
\begin{align*}
\text{minimize} & \quad \left( \hat{\epsilon}_{\text{avg}, S} + \frac{1}{2} \log \sum_{m=1}^{M} \exp \left( \sup_{h_m \in \mathcal{H}} d_{h_m, \mathcal{Y} \mathcal{M} \mathcal{V}}(X_{S,m}, X_{T,m}) \right) \right) . \\
\end{align*}
\]

(21)

The hard-max objective is data inefficient because the gradient of the max function is only non-zero for \( h_m \) with the maximum divergence, and the algorithm only updates its parameters based on the gradient from one of the \( M \) tasks. The average objective updates algorithm parameters based on the average gradient from all \( M \) tasks. However, this objective considers each task as equally contributing to updating the algorithm parameters, which may not allocate enough computational and learning resources to optimize tasks with larger divergence. The soft-max objective combines the gradients from all the tasks and adaptively assigns the loss of task that have a larger divergence with a heavier weight. In this way, the model automatically applies larger gradient magnitudes to tasks having more shifted distributions between different domains. As a result, we propose to use the soft-max objective (Equation 21) to optimize the multi-task UDA problem.

**HierMUD: A hierarchical multi-task unsupervised domain adaptation framework**

We now proceed to introduce our HierMUD framework that transfers the model learned from a source domain to predict multiple tasks on a target domain without any labels from the target domain in any of the tasks. In our drive-by BHM application, the two domains are vibration data and damage information collected from a vehicle passing over two structurally different bridges, and the multiple learning tasks are damage diagnostic tasks, such as damage detection, localization, and quantification. The overview flowchart of our framework is shown in Figure 3. The framework contains three modules: 1) a data pre-processing module, 2) a multi-task UDA module, and 3) a target domain prediction module. In the following subsections, we present each module in detail.

**Data pre-processing module**

The data pre-processing module contains two steps: data augmentation and initial processing. In the first step, to avoid overfitting and data biases while providing sufficient information of each class, we expand the size of the dataset and introduce data variability by conducting data augmentation, which improves the robustness of the learned multi-task UDA model. Some widely utilized data augmentation procedures include adding white noise, randomly cropping matrix size, or randomly erasing samples from the original data for each batch of training data. The procedure is selected based on the prediction performance on the validation set (e.g., for our drive-by BHM application, we add white noise to augment the vibration data).

In the second step, we create the input to our multi-task UDA module, including the source domain data with the corresponding labels and the target domain data. Feature transforms are applied to the raw input data to provide information in other feature space. For example, a Fast Fourier Transform can be used to convert the signal from its original domain (time or space) to the frequency domain. Short-Time Fourier Transform (STFT) or wavelet transform can be used to convert the time or space domain signal to the time–frequency domain. Specifically, in our drive-by BHM application, we conduct data augmentation by adding white noise to vehicle vibration signals. Then, we compute the STFT of each vertical acceleration record of the vehicle traveling over the bridge to preserve the time–frequency domain information.

**Multi-task unsupervised domain adaptation module**

In this module, we introduce our hierarchical multi-task and domain adversarial learning algorithm (as shown in Figure 4) that exploits the derived generalization risk bound for multi-task UDA based on the theoretical study in the previous section. This algorithm integrates domain adversarial learning and hierarchical multi-task learning to achieve an optimal trade-off between domain-invariance and task-informativeness.

Our algorithm consists of three components: hierarchical feature extractors (orange blocks), task predictors (blue blocks), and domain classifiers (red blocks). Domain adversarial learning utilizes the domain classifier to minimize the domain discrepancy through an adversarial objective (e.g., Equation 18) for training against the feature extractors, which encourages the extracted feature to be domain-invariant. The architectures of each component are presented in the following paragraphs.
Figure 3. The flowchart for our HierMUD framework. The input to the framework is source domain data with the corresponding labels and the unlabeled target domain data, and the framework outputs the predicted target domain labels. Black dash lines indicate source domain data stream, solid red lines indicate target domain data stream, and the blue dash line indicates that we copy the multi-task predictors pre-trained using source domain data to predict target domain tasks.

Figure 4. The architecture of our hierarchical multi-task and domain-adversarial learning algorithm. The red and black arrows between blocks represent source and target domain data stream, respectively. Orange blocks are feature extractors, blue blocks are task predictors, and red blocks are domain classifiers.

**Architecture of hierarchical feature extractors, task predictors, and domain classifiers.** Hierarchical feature extractors. The hierarchical feature extractors extract domain-invariant and task-informative features. To ensure domain-invariance, the parameters of the extractors are optimized with domain classifiers in an adversarial way to extract features that cannot be differentiated by the domain classifiers (i.e., domain-invariant) while the domain classifiers are optimized to best distinguish which domain the extracted features come from.

To ensure task-informativeness, we implement hierarchical feature extractors that learn task-shared and task-specific feature representations for tasks with different learning difficulties. Inspired by human learning and the work of Guo et al., we separate the total of $M$ tasks into $M_1$ easy-to-learn and $M_2$ hard-to-learn tasks based on the
task difficulty, which is inversely proportional to the learning performance (e.g., prediction accuracy) in the source domain. In particular, we train, respectively, $M$ classifiers with the same model complexity using source domain data of the $M$ tasks, and obtain testing accuracy values, $\{p_1, p_2, \ldots, p_M\}$, for the $M$ tasks. A threshold $p_i$ is determined based on our domain knowledge and empirical observations of the problem. We then consider tasks with $p_m \geq p_i$ as easy-to-learn tasks, and tasks with $p_m \leq p_i$ as hard-to-learn tasks. The choice of $p_i$ is application different and relies on domain knowledge. A smaller $p_i$ would categorize more tasks as hard-to-learn ones and distribute more learning resources to these tasks, increasing the model complexity, and vice versa. For example, in our drive-by BHM application, the value of $p_i$ is 0.9, damage detection and localization are considered as easy-to-learn tasks, and damage quantification is considered as a hard-to-learn task.

Furthermore, we implement two types of feature extractors: one task-shared feature extractor and $M_2$ task-specific feature extractors. We denote $W_0 (\cdot)$ as the task-shared feature extractor with parameters $\theta_{W_0}$, and $W_m (\cdot)$ as the task-specific feature extractor for the $m$-th hard-to-learn task with parameters $\theta_{W_m}$, where $m \in \{1, 2, \ldots, M_2\}$. The source and target domain data after being processed in the data pre-processing module, $(\mathcal{X}_s, \mathcal{X}_t)$, are input to the task-shared feature extractor to extract task-shared features, $Z_s$ and $Z_t$, for the source and target domain, respectively. For each hard-to-learn task, task-specific features, $Z_{s,m}$ and $Z_{t,m}$, are extracted from the task-shared features using the corresponding task-specific feature extractor.

The task-shared feature extractor is implemented as a deep convolutional neural network (CNN) that combines convolutional layers and pooling layers to extract feature representations. We utilize CNN to extract features because it has an excellent performance in understanding spatial hierarchies and structures of features in various resolutions. Further, task-specific feature extractors are implemented as deep fully connected neural networks, consisting of multiple fully connected layers that map task-shared features to task-specific features.

Task predictors. Task predictors are trained to ensure that the extracted features from the hierarchical feature extractors are task-informative. They are implemented as deep fully connected neural networks that map features to task labels. There are $M$ task predictors for all the $M$ learning tasks. We denote $F_m (\cdot)$ as the task predictor for the $m$-th task with parameters $\theta_{F_m}$, where $m \in \{1, 2, \ldots, M\}$. In the training phase, the input to the task predictor of each easy-to-learn task is the task-shared feature in the source domain. The input to the task predictor of each hard-to-learn task is the corresponding task-specific feature in the source domain. Each predictor outputs the predicted source domain labels, $\hat{YS}_m$, in each task.

Domain classifiers. Domain classifiers are trained to distinguish which domain the extracted features are from. We also have two types of domain classifiers: one task-shared domain classifier and $M_2$ task-specific domain classifiers. We denote $D_0 (\cdot)$ as the task-shared domain classifier with parameters $\theta_{D_0}$, and $D_m (\cdot)$ as the task-specific domain classifier for the $m$-th hard-to-learn task with parameters $\theta_{D_m}$, where $m \in \{1, 2, \ldots, M_2\}$. The task-shared domain classifier takes the task-shared features in all $M$ tasks from the source task or the target domain as input and predicts if the feature sample comes from the source domain or not (i.e., a binary classification). Each task-specific domain classifier takes the task-specific features for each task from the source domain or the target domain as input and also classifies the feature sample into two classes (as the source or the target domain).

We implement the domain adversarial learning by back-propagation, inspired by Ganin et al., using the gradient reversal layer (GRL). We use GRL because it can be easily incorporated into any existing neural network architecture that can handle high-dimensional signals and multiple learning tasks. In particular, each domain classifier is connected to the corresponding feature extractor via a GRL that multiplies the gradient by a negative constant during back-propagation updating. With GRL, feature extractors and domain classifiers are trained in an adversarial way, such that the extracted features are as indistinguishable as possible for even well-trained domain classifiers.

Domain classifiers are implemented as deep fully connected neural networks that map feature to domain labels. One should note that the architecture of domain classifiers is simpler than that of task predictors to avoid overusing learning resources to train domain classifiers over task predictors, which could reduce task-informativeness of the extracted features.

Loss function for hierarchical multi-task and domain-adversarial learning algorithm. In this subsection, we present the loss function for our hierarchical multi-task and domain-adversarial learning algorithm, which minimizes the objective function in Equation (21).

After considering the hierarchical structure, we can rewrite the objective function of the optimization in Equation (21) as
where \( L_c(m; \theta_w, \theta_f) = -\mathbb{E}_{(x,y) \sim (X_m, Y_m)} \sum_{c=1}^{C_m} \mathbb{I}(y = c) \log p(y|x; \theta_w, \theta_f) \) is the cross-entropy loss for the \( m \)-th easy-to-learn task; \( C_m \) is the number of classes for the \( m \)-th task; \( L_{h,m}(\theta_w, \theta_w, \theta_w) = -\mathbb{E}_{(x,y) \sim (X_m, Y_m)} \sum_{c=1}^{C_m} \mathbb{I}(y = c) \log p(y|x; \theta_w, \theta_w, \theta_w) \) is the cross-entropy loss for the \( m \)-th hard-to-learn task; \( \mathcal{L}_{D_{h,m}}(\theta_w, \theta_f) = -\mathbb{E}_{x \sim X_m} \log p(y|x; \theta_w, \theta_f) - \mathbb{E}_{x \sim X_m} \log (1 - p(y|x; \theta_w, \theta_f)) \) is the task-shared domain classifier loss for the \( m \)-th tasks \( m \in \{1, 2, \ldots, M\} \); \( \mathcal{L}_{D_{m}}(\theta_w, \theta_f) = -\mathbb{E}_{x \sim X_m} \log p(y|x; \theta_w, \theta_f) - \mathbb{E}_{x \sim X_m} \log (1 - p(y|x; \theta_w, \theta_f)) \) is the task-specific domain classifier loss; \( \lambda_1, \ldots, \lambda_M \) are hyper-parameters to represent the trade-off between easy-to-learn tasks and hard-to-learn tasks weights; and \( \lambda_D, \lambda_D, \ldots, \lambda_D \) are hyper-parameters to represent the trade-off between domain-invariance and task-informativeness of features.

The minimax optimization problem in Equation (22) is solved by finding the saddle point

\[
\hat{\theta}_{w_1}, \hat{\theta}_{w_1}, \ldots, \hat{\theta}_{w_M}, \hat{\theta}_{f_1}, \ldots, \hat{\theta}_{f_M},
\]

such that

\[
\begin{align*}
\theta_{w_1}, \theta_{w_1}, \ldots, \theta_{w_M}, \theta_{f_1}, \ldots, \theta_{f_M} &= \argmin_{\theta_{w_1}, \ldots, \theta_{w_M}, \theta_{f_1}, \ldots, \theta_{f_M}} \left[ \frac{1}{M_1} \sum_{m=1}^{M_1} \lambda_m L_{c,m}(\theta_{w_1}, \theta_{f_1}) \right. \\
&+ \lambda_1 \sum_{m=1}^{M_1} \lambda_m L_{h,m}(\theta_{w_1}, \theta_{w_1}, \theta_{f_1}) \left. \\
&+ \lambda_D \log \sum_{m=1}^{M} \exp \left( -\min_{\theta_{w_1}, \ldots, \theta_{w_M}, \ldots, \theta_{f_1}, \ldots, \theta_{f_M}} L_{D_{m}}(\theta_{w_1}, \theta_{w_1}, \theta_{f_1}) \right) \right] \\
&+ \lambda_{D,1} \log \sum_{m=1}^{M} \exp \left( -\min_{\theta_{w_1}, \ldots, \theta_{w_M}, \ldots, \theta_{f_1}, \ldots, \theta_{f_M}} L_{D_{m}}(\theta_{w_1}, \theta_{w_1}, \theta_{f_1}) \right),
\end{align*}
\]

(23)

(24)

(25)

A saddle point defined by Equations (23)–(25) can be found as a stationary point of the following gradient updates: for updating hierarchical feature extractors,

\[
\theta_{w_1} \leftarrow \theta_{w_1} - \mu \left( \frac{1}{M_1} \sum_{m=1}^{M_1} \lambda_m \frac{\partial L_{c,m}}{\partial \theta_{w_1}} + \frac{1}{M_2} \sum_{m=1}^{M_2} \lambda_{m+M} \frac{\partial L_{h,m}}{\partial \theta_{w_1}} - \lambda_D \sum_{m=1}^{M} \frac{\partial L_{D_{m}}}{\partial \theta_{w_1}} \right)
\]

(26)

(27)

for updating the \( m \)-th easy-to-learn task predictor, where \( m \in \{1, \ldots, M_1\} \)

\[
\theta_{f_1} \leftarrow \theta_{f_1} - \mu \left( \frac{\lambda_{m+M}}{M_1} \frac{\partial L_{c,m}}{\partial \theta_{f_1}} - \lambda_D \frac{\partial L_{D_{m}}}{\partial \theta_{w_1}} \right)
\]

(28)

for updating the \( m \)-th hard-to-learn task predictor, where \( m \in \{1, \ldots, M_2\} \)

\[
\theta_{f_1} \leftarrow \theta_{f_1} - \mu \left( \frac{\lambda_{m+M}}{M_2} \frac{\partial L_{c,m}}{\partial \theta_{f_1}} - \lambda_D \frac{\partial L_{D_{m}}}{\partial \theta_{f_1}} \right)
\]

(29)

for updating domain classifiers

\[
\theta_{D_1} \leftarrow \theta_{D_1} - \mu \lambda_D \sum_{m=1}^{M} \frac{\partial L_{D_{m}}}{\partial \theta_{D_1}}
\]

(30)

(31)

where \( \mu \) is the learning rate;

\[
\omega_m = \frac{\exp \left( -\min_{\theta_{w_1}, \ldots, \theta_{w_M}, \ldots, \theta_{f_1}, \ldots, \theta_{f_M}} L_{D_{m}}(\theta_{w_1}, \theta_{f_1}) \right)}{\sum_{m=1}^{M} \exp \left( -\min_{\theta_{w_1}, \ldots, \theta_{w_M}, \ldots, \theta_{f_1}, \ldots, \theta_{f_M}} L_{D_{m}}(\theta_{w_1}, \theta_{f_1}) \right)}
\]

is the adaptive weight for the \( m \)-th task. Task with larger distribution divergence has larger weight. The updates of Equation (28)−(31) are similar to those of deep neural network models using stochastic gradient descent (SGD). For Equation (26) and (27), the difference between the updates of them and SGD updates is that the gradients from the domain classifiers are subtracted. This subtracted gradient is accomplished by inserting the aforementioned GRL between feature extractors and domain classifiers. Specifically, the forward propagation of the GRL is the same as an identity transformation, and during back-propagation, the GRL changes the sign of the gradient (i.e., multiply it by a negative constant) before passing it to the preceding layer. The optimization of our hierarchical multi-task and domain-adversarial learning algorithm is summarized in Algorithm 1.
Algorithm 1. Hierarchical multi-task and domain-adversarial learning algorithm

**Input:**
- Training iterations: \( P \),
- Batch size: \( K \),
- Number of tasks: \( M \),
- The first \( M_1 \) tasks are easy-to-learn tasks, the last \( M_2 \) tasks are hard-to-learn tasks, and \( M = M_1 + M_2 \),
- Number of classes for the tasks: \( C_m \) for \( m = 1, \ldots, M \),
- Hyper-parameters: \( \lambda_1, \ldots, \lambda_M, \lambda_{D_1}, \ldots, \lambda_{D_M} \).

**Output:** Neural network: \( \{W_0, D_0, W_1, \ldots, W_{M_2}, F_1, \ldots, D_M, \ldots, D_M\} \)

1: Randomly initialize network parameters: \( \theta_{W_1}, \theta_{D_1}, \theta_{W_2}, \ldots, \theta_{W_{M_2}}, \theta_{F_1}, \ldots, \theta_{F_M}, \theta_{D_1}, \ldots, \theta_{D_M} \).
2: **for** \( p \) from 1 to \( P \) **do**
3:   **#** Forward pass
4:   **for** \( m \) from 1 to \( M \) **do**
5:     Sample \( K \) data points and label from the source domain \( D_{S,m} \) for each task: \( (X_{S,m}, Y_{S,m}) \).
6:     Sample \( K \) data points from the target domain \( D_t \) for each task: \( X_{T,m} \).
7:     Compute the task-shared feature for the source domain data: \( Z_{S,m} = W_0(X_{S,m}) \).
8:     Compute the task-shared feature for the target domain data: \( Z_{T,m} = W_0(X_{T,m}) \).
9:   **end for**
10:  **for** \( m \) from 1 to \( M_2 \) **do**
11:     Compute the task-specific feature for the source domain hard-to-learn tasks: \( Z_{S,m} = W_m(Z_{S,m}+\lambda_{D_m}) \).
12:     Compute the task-specific feature for the target domain hard-to-learn tasks: \( Z_{T,m} = W_m(Z_{T,m}+\lambda_{D_m}) \).
13:  **end for**
14:  **#** Backward pass
15:  Update the feature extractors and task predictors using equation (22):

\[
\begin{align*}
&\left(\theta_{W_1}, \theta_{W_2}, \ldots, \theta_{W_{M_2}}, \theta_{F_1}, \ldots, \theta_{F_M}\right) \\
= &\arg\min_{\theta_{W_1}, \theta_{W_2}, \ldots, \theta_{W_{M_2}}, \theta_{F_1}, \ldots, \theta_{F_M}} \left[ \frac{1}{M_1} \sum_{m=1}^{M_1} \lambda_m C_m(\theta_{W_1}, \theta_{F_1}) + \frac{1}{M_2} \sum_{m=1}^{M_2} \lambda_m C_m(\theta_{W_2}, \theta_{F_2}) + \ldots \right. \\
&\left. + \lambda_{D_m} \log \sum_{m=1}^{M} \exp \left( \min_{\theta_{W_1}} \sum_{m=1}^{M} \lambda_{D_m} C_m(\theta_{W_1}, \theta_{F_1}, \theta_{D_1}) \right) \right],
\end{align*}
\]

(32)

16: Update the task-shared domain classifier: \( \hat{D}_{D_0} = \arg\max_{\theta_{D_0}} \log \sum_{m=1}^{M} \exp (-\lambda_{D_m} C_m(\theta_{W_1}, \theta_{F_1}, \theta_{D_1})) \).
17: **for** \( m \) in 1: \( M_2 \) **do**
18:     Update the task-specific domain classifier: \( \hat{D}_{D_m} = \arg\min_{\theta_{D_m}} \lambda_{D_m} C_m(\theta_{W_1}, \theta_{F_1}, \theta_{D_m}) \).
19: **end for**
20: **end for**
Target domain prediction module

The architecture of the target domain prediction module is shown in Figure 5. In this module, the extracted target domain task-shared features are input to the pre-trained easy-to-learn predictors to predict target domain labels in the easy-to-learn tasks, and the extracted target domain task-specific features are input to the pre-trained hard-to-learn predictors to predict target domain labels in the hard-to-learn tasks.

Evaluation

In this section, we evaluate our HierMUD framework for drive-by BHM using data collected from lab-scale experiments. The experiments are conducted on two structurally different bridges using three vehicles of different weights.

Experimental setup and data description

A lab-scale VBI system, as shown in Figure 6, was employed to create the dataset. The collected dataset is subjected to environmental noise (e.g., different temperatures and humidity), electrical noise from the sensing system, and the noise caused by the friction between vehicle wheels and bridge surface. The experiments involved two 8-foot bridges (B1 and B2) with different weights of 34.2 lb and 43.0 lb, different dominant frequencies of 5.9 Hz and 7.7 Hz, and damping ratios of 0.13 and 0.07, respectively. The data are collected from three small-scale vehicles (V1, V2, and V3) with different weights of 10.6 lb, 11.6 lb, and 12.6 lb, respectively, that were driven over the bridge. Vertical acceleration signals were collected from four accelerometers mounted on each vehicle (front chassis, back chassis, front wheel, and back wheel) while they moved.
We consider damage detection and localization as easy-to-learn tasks and damage quantification as a hard-to-learn task. This is because we obtained higher supervised prediction accuracy for damage detection and localization tasks than that for the quantification task, and we observed that the data distributions for different damage locations are more separable than that for different damage severity levels. We then develop the hierarchical architecture that extracts task-shared features for all the tasks and further task-specific features for the quantification task. Moreover, the existence of damage (for damage detection task) is represented by introducing an additional label within the location predictor, instead of creating an additional binary damage detection classifier that increases the model complexity. After all, the overall architecture of the neural network modules used in our HierMUD model are shown in Figure 8 and Table 1. We optimize our HierMUD model using a stochastic gradient descent optimizer with weight decay, which encourages smaller weights of the networks by adding an L2-regularization term to the loss to avoid overfitting. The network architecture and hyper-parameters are determined using the unsupervised hyper-parameter selection method, which is described below. Based on our experiments, the architecture shown in Table 1 provides us with the best damage diagnosis results.

**Baseline methods**

We compare the performance of HierMUD with five baseline methods described below:

1) MCNN is a multi-task convolutional neural network model that directly applies the model trained using source domain data to the target bridge without applying any domain adaptation. The architecture of MCNN lacks the red boxes (domain classifiers) in Figure 8.

2) iUD is an independent task learning model with UDA. It predicts damage location and severity levels using two independent UDA models. The architecture of iUD does not have the task-specific feature extractor, and instead it uses two independent domain adversarial neural networks (DANN) for damage localization and quantification.

3) sUD is a sequential task learning model with UDA. It predicts damage location and severity level step-by-step instead of using multi-task learning. Specifically, sUD uses one DANN for damage localization, and for each predicted damage location, it uses one DANN for damage quantification.

4) MUD directly combines MTL and UDA. It predicts damage location and severity level simultaneously, but does not use the hierarchical structure and the

**Setup of our HierMUD framework**

In this subsection, we describe our data pre-processing procedure and the setup of our HierMUD framework. We first pre-processed the input data by conducting data augmentation. Data augmentation adds white noise to vehicle acceleration signals. The white noise has zero mean and variance of mean squared magnitude of the vehicle acceleration signal. Then, we compute the short-time Fourier transform (STFT) representation of each signal to preserve the time–frequency domain information. We use the STFT representation because the frequency spectrum is sensitive to bridge characteristics and STFT allows us to preserve the location information of the moving vehicle and the bridge damage through its time–frequency domain representation. The size of each input data is $C \times W \times H$, where $C$ is the number of sensor channels on the vehicle; $W$ and $H$ are respectively the number of time segments and the sample frequencies of the STFT representation. Particularly, in our system, the original time-domain acceleration signal’s dimension is $4 \times 4000$, and the processed STFT data’s dimension is $4 \times 64 \times 64$. Figure 7 shows an example of the vehicle acceleration signal and its corresponding STFT representation.
Table 1. The architecture of our multi-task and domain adversarial learning algorithm applied on the drive-by BHM.

| Network                          | Layer                   | Patch size | Input size      | Activation     |
|---------------------------------|-------------------------|------------|-----------------|----------------|
| Task-shared feature extractor   | Convolution (2D)        | 64 × 5 × 5 | 4 × 64 × 64     | LeakyReLU       |
|                                 | Max pooling             | 2 × 2      | 64 × 60 × 60    |                |
|                                 | Convolution (2D)        | 50 × 5 × 5 | 64 × 30 × 30    | LeakyReLU       |
|                                 | Max pooling             | 2 × 2      | 64 × 26 × 26    | LeakyReLU       |
|                                 | Convolution (2D)        | 50 × 3 × 3 | 50 × 13 × 13    | LeakyReLU       |
|                                 | Max pooling             | 2 × 2      | 50 × 11 × 11    | LeakyReLU       |
| Task-specific feature extractor | Flatten                 | 50 × 5 × 5 |                 |                |
|                                 | Full connection         | 1250 × 1250| 1250 × 1        | ReLU           |
| Location predictors             | Flatten                 | 50 × 5 × 5 |                 |                |
|                                 | Full connection         | 1250 × 100 | 100 × 1         | ReLU           |
|                                 | Full connection         | 100 × 4    | 100 × 1         | Soft-max       |
| Severity predictors             | Flatten                 | 50 × 5 × 5 |                 |                |
|                                 | Full connection         | 1250 × 100 | 100 × 1         | ReLU           |
|                                 | Full connection         | 100 × 5    | 100 × 1         | Soft-max       |
| Task-shared and task-specific   | Flatten                 | 50 × 5 × 5 |                 |                |
| domain classifiers              | Full connection         | 1250 × 2   | 1250 × 1        | Soft-max       |

Figure 7. An example of the raw acceleration signal collected from a vehicle passing over a lab-scale bridge and its corresponding STFT representation.

Figure 8. The architecture of our multi-task and domain adversarial learning algorithm applied on the drive-by BHM.

Structural Health Monitoring 0(0)
Table 2. The comparison of our method and baseline methods. MTL, ITL, STL, and UDA stand for multi-task learning, independent task learning, sequential task learning, and unsupervised domain adaptation, respectively.

| Method     | MTL | ITL | STL | UDA | Hierarchical structure | Soft-max objective |
|------------|-----|-----|-----|-----|------------------------|-------------------|
| MCNN       | ✓   |     |     |     |                        |                   |
| iUD        | ✓   | ✓   |     |     |                        |                   |
| sUD        |     | ✓   |     |     |                        |                   |
| MUD        | ✓   |     | ✓   |     |                        |                   |
| HierMUD-a  | ✓   |     |     | ✓   |                        |                   |
| HierMUD (ours) | ✓ |     |     | ✓   |                        | ✓                 |

5) HierMUD-a further adds the hierarchical structure to MUD, but it optimizes the average objective—Equation 20.

The differences of these methods are also summarized in Table 2. Note that to have a fair comparison, components (feature extractors, task predictors, and domain classifiers) of each method have the same architecture as HierMUD method. We also use the same strategy to train these methods, which is described in the following section.

Unsupervised hyper-parameter selection

In this section, we describe an approach to fine-tune hyper-parameters, including the learning rate, network architecture, and trade-off hyper-parameters (as shown in Equation 22), for performing our multi-task and domain-adversarial learning algorithm. Because there is no labeled data in the target domain during the training phase, we select hyper-parameters by conducting a reverse validation.31 61 In short, the labeled source samples $X_S$ and unlabeled target samples $X_T$ are split into training sets ($X'_S$ and $X'_T$) which contain 90% of $X_S$ and $X_T$ (i.e., 8424 labeled data from the source domain and 8424 unlabeled data from the target domain) and the validation sets ($X''_S$ and $X''_T$), where $X' \cup X'' = X$ and they are mutually exclusive. The training sets $X'_S$ and $X'_T$ are used to learn a target domain prediction model $\eta(\cdot)$. We then learn another prediction model $\eta_r(\cdot)$ using unlabeled target domain data, $X''_T$, with the predicted labels ($\eta(x)$ for $x \in X''_T$). The reverse classifier $\eta_r(\cdot)$ is evaluated on the source domain validation set $X''_S$. This evaluation process is conducted for several times with different values of hyper-parameters.

Tuning the trade-off hyper-parameters ($\lambda_1, \ldots, \lambda_M, \lambda_{D_0}, \lambda_{D_1}, \ldots, \lambda_{D_m}$) is important because they affect the convergence of HierMUD’s training and damage diagnosis accuracy by changing the magnitude of gradient from each component of HierMUD. In particular, we observe that using relatively large task-informativeness trade-off hyper-parameters ($\lambda_m$, for $m \in 1, \ldots, M$) and relatively small domain-invariance trade-off hyper-parameters ($\lambda_{D_0}$ and $\lambda_{D_m}$, for $m \in 1, \ldots, M_2$) speeds up the convergence of HierMUD’s training but reduces the target domain prediction accuracy, and vice versa.

In our drive-by BHM problem, we set the hyper-parameter for quantification loss $\lambda_2$ to be one and tune the hyper-parameter for localization loss $\lambda_1$ by searching its optimal value in the set $\{0.01, 0.1, 0.5, 1\}$ that provides the best reverse validation accuracy. For $\lambda_{D_0}$ and $\lambda_{D_1}$, we tune them by searching the values of them in the same set $\{0.01, 0.1, 0.5, 1\}$. Thus, in total, we have $\lambda \times \lambda \times \lambda = 64$ possible combinations of hyper-parameters. We also conduct 10-fold cross-validation with each set of hyper-parameters and choose the hyper-parameters that give the highest cross-validation accuracy for the reverse classifier on the source domain validation set. For our experiment, the best choices of $\lambda_1, \lambda_2, \lambda_{D_0}$, and $\lambda_{D_1}$ are 0.5, 1, 0.1, and 0.5, respectively.

Results and discussion

In this section, we present our evaluation results and discuss our findings. Table 3 presents the results for knowledge transfer from B1 to B2 and from B2 to B1 using each of the three vehicles’ data. For the binary damage detection, we report F1-scores as the performance metric; For the damage quantification task, we report classification accuracy (the number of correct predictions divided by the total number of predictions made) as the performance metric; For the damage quantification task, we report the mean absolute error (MAE) between predictions and ground truth damage severity. Note that we first obtain damage severity estimation through the 4-class classification and then compute MAE by taking the absolute difference between the predicted label and the ground truth. MAE is chosen because it shows the degree of error rather than classification accuracy (i.e., only showing correct vs incorrect). There are six evaluations in total (for 2 bridges and 3 vehicles), and each evaluation was conducted 10 times with different random seeds for splitting train and validation sets and initializing model parameters. Therefore, we have total 60 tests. Each column in Table 3 shows the performance for each of the six evaluations. The last column of the table provides the overall performance of all six
Table 3. The performance of baseline methods and our method on the lab-scale VBI dataset. “A → B” indicates that we transfer knowledge from the source domain A to the target domain B. The numbers in each cell are: average performance (± 95% confidence interval). We bold the number that has the best result in each task.

| Task (Metric) | Method       | Vehicle 1          | Vehicle 2          | Vehicle 3          | Overall          |
|--------------|--------------|---------------------|---------------------|---------------------|-----------------|
|              |              | B1 → B2 | B2 → B1 | B1 → B2 | B2 → B1 | B1 → B2 | B2 → B1 | Overall |
| Damage       | MCNN         | 0.85 (±0.12) | 0.78 (±0.11) | 0.76 (±0.11) | 0.71 (±0.07) | 0.73 (±0.06) | 0.77 (±0.11) | 0.74 (±0.04) |
|              | iUD          | 0.65 (±0.07) | 0.63 (±0.06) | 0.56 (±0.06) | 0.57 (±0.08) | 0.52 (±0.05) | 0.58 (±0.06) | 0.58 (±0.03) |
|              | sUD          | 0.85 (±0.12) | 0.86 (±0.04) | 0.84 (±0.10) | 0.87 (±0.07) | 0.95 (±0.04) | 0.95 (±0.02) | 0.89 (±0.03) |
|              | MUD          | 0.80 (±0.05) | 0.88 (±0.05) | 0.82 (±0.11) | 0.91 (±0.04) | 0.97 (±0.03) | 0.90 (±0.09) | 0.90 (±0.03) |
|              | HierMUD-a    | 0.92 (±0.03) | 0.87 (±0.05) | 0.92 (±0.06) | 0.93 (±0.04) | 0.98 (±0.01) | 0.95 (±0.04) | 0.93 (±0.02) |
|              | HierMUD      | 0.93 (±0.01) | 0.98 (±0.01) | 0.99 (±0.01) | 0.98 (±0.01) | 0.96 (±0.02) | 0.96 (±0.03) | 0.95 (±0.01) |

evaluations. The numbers in each cell of the table are: average performance (± 95% confidence interval). To show the relationship between true damage values and predictions, we also present normalized confusion matrices of our approach for the damage quantification and localization tasks in Figures 15 and 16 (in Appendix B), respectively.

Performance comparison with baseline methods

Except for the damage detection task for the evaluation using V1 response to transfer model from B2 to B1, our method (HierMUD) outperforms other baselines. Overall, HierMUD has the best damage localization accuracy, which is 93% on average. HierMUD also achieves the smallest damage quantification error, which is 0.38 lbs on average that is within one damage severity level of 0.5 lbs. The average objective (HierMUD-a) has the best damage detection F1-score (96%), followed by HierMUD (95%). HierMUD is about twice as good as the baseline without domain adaptation (MCNN) in all three diagnostic tasks and 1.5 times as good as other baselines (iUD, sUD, and MUD), in the hard-to-learn quantification task. HierMUD achieves the best 99% average F1-score (up to 100% in the best test) in the damage detection task using signals collected from V2 for model transfer from B1 to B2, the best 98% average accuracy (up to 100% in the best test) in the damage localization task, and the smallest 0.29 lbs average MAE (0.21 lbs in the best test) in the damage quantification task using signals collected from V3 for model transfer from B1 to B2.

Figures 9, 10, and 11 show the boxplots of performance metric in damage detection, localization, and quantification tasks, respectively. In each figure set, boxplot (a) shows the overall 60 test results of our method and baselines. Boxplots (b) and (c) present 30 test results for model transfer from B1 to B2 and from B2 to B1, respectively. Each box in the boxplot shows five values, including minimum, maximum, median, the first quartile, and the third quartile. In addition to the results from Table 3, we observe that the results of our method have smaller variance (i.e., smaller box in boxplots) than that of baseline methods, which indicates a more stable performance with different random initialization. Furthermore, for the damage quantification task using our method, results for model transfer from B1 to B2 are better than those for model transfer from B2 to B1. This difference is coincidental with or due to the fact that different model transfer directions have distinct model generalizability (i.e., the model learned from B1 data is more generalizable and easy to transfer than that learned from B2).

In summary, our method outperforms baselines in comprehensive laboratory evaluations, which shows the effectiveness of combining domain adversarial, multi-task learning with the hierarchical architecture and the soft-max objective.
Figure 9. The performance ((a) overall performance; (b) B1 to B2; (c) B2 to B1) of baseline methods and our method for damage detection task on the lab-scale VBI dataset.

Figure 10. The performance ((a) overall performance; (b) B1 to B2; (c) B2 to B1) of baseline methods and our method for damage localization task on the lab-scale VBI dataset.

Figure 11. The performance ((a) overall performance; (b) B1 to B2; (c) B2 to B1) of baseline methods and our method for damage quantification task on the lab-scale VBI dataset. Note that our damage severity levels are 0.5, 1.0, 1.5, and 2.0 lbs.

Characterizing the extracted feature distributions

In this section, we characterize the extracted features to show the effect of domain adaptation on the distribution of the extracted features. Figure 12 (a) shows the tSNE projection of the task-shared features ($Z_s$ and $Z_T$) after domain adaptation for the damage localization task; Figure 12 (b) shows the tSNE projection of the source and target domains’ task-specific features ($Z_s^T$ and $Z_T^T$) after domain adaptation for the damage quantification task. Different colors indicate different
The losses of damage localization and quantification in the source domain, and the accuracy/F1-score of damage detection, localization, and quantification in the target domain.
damage labels. Filled and unfilled markers represent features of source and target domains, respectively. Comparing the distributions of features before and after domain adaptation in Figure 2 and 12, we observe that our domain adaptation method successfully transformed the feature distributions of the two domains to be much more similar to each other. Also, comparing Figures 12 (a) and (b), the distributions of the features are more similar and thus better matched for the easy-to-learn damage localization task than that for the hard-to-learn damage quantification task.

Further, Figure 13 shows the same tSNE projection of the task-shared features ($Z_S$ and $Z_T$) but color-coded with different damage severity levels. The distributions of task-shared features having the same damage severity label in Figure 13 (i.e., the markers with the same color) are more spread out than those in Figure 12 (b), making features in Figure 13 more difficult to be correctly classified. Therefore, task-specific features are more task-informative than task-shared features for the quantification task, which corresponds to our results that using the task-specific hierarchical features outperforms using the task-shared features for damage quantification.

**Characterizing training process**

To characterize the training process, we show the training losses and testing accuracy for model transfer from B1 to B2 using V2 data in Figure 14. Blue curves are average losses of the two task predictors in the source domain, and red curves are average accuracy/F1-score of damage detection, localization, and quantification in the target domain. The envelope covers loss or accuracy for the 10 tests with different random initialization. The figure shows that the two losses fluctuate and even increase during early training steps (i.e., when the number of epochs is small) due to domain adversarial learning. They converge to small loss values after around 200 epochs. We also observe that the quantification loss and accuracy have a large area of the envelope because it is difficult to find optimized saddle points of model parameters for this hard-to-learn quantification task.

**Conclusion and future work**

In this work, we introduce HierMUD, a new multi-task unsupervised domain adaptation framework that uses drive-by vehicle acceleration responses to diagnose structural damage of multiple bridges without requiring labeled data from every bridge. Our framework jointly optimizes hierarchical feature extractors, damage predictors, and domain classifiers in an adversarial way to extract features that are task-informative and domain-invariant. To match distinct data distributions over multiple tasks, we introduce a novel loss function based on a new provable generalization risk bound to adaptively provide a larger gradient magnitude on matching tasks having more shifted distribution between the source and target domains. To learn multiple tasks with varying task difficulties, the feature extractor is designed to formulate a feature hierarchy, which learns two-level features: task-shared features are used for learning easy-to-learn tasks, and task-specific features are further extracted from task-shared features to learn hard-to-learn tasks. To the best of our knowledge, this is the first framework to transfer the model learned from one bridge to detect, localize, and quantify damage to another (target) bridge without any labels of the target bridge. We evaluate our framework on experimental data collected from three vehicles passing over two bridges individually. Our framework is up to 2X better than baseline methods and has smaller prediction variance. It achieves average accuracy values of 95% for damage detection, 93% for damage localization, and MAE 0.38 lbs for damage quantification.

In future work, we plan to investigate how the model transfer direction affects the domain adaptation performance. For instance, we will investigate what characteristics of source domain bridges lead to a better performance when the model is transferred to a target domain. In addition, we intend to study how to transfer our multi-task damage diagnosis model between bridges with different structural forms (e.g., truss bridge, suspension bridge, etc.). Moreover, we believe this framework can be generalized to other multi-task UDA problems beyond structural health monitoring. Thus, we plan to apply our framework in other applications (e.g., visual recognition, machine monitoring, etc.) to further evaluate its robustness and effectiveness.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported in part by the Leavell Fellowship on Sustainable Built Environment and UPS endowment fund at Stanford University, and Mobility 21 at Carnegie Mellon University.

**ORCID iD**

Jingxiao Liu https://orcid.org/0000-0001-5559-1627

**References**

1. ASCE. Bridges- infrastructure report card, https://www.infrastructurereportcard.org/wp-content/uploads/2017/01/Bridges-Final.pdf. (2017).
2. Hartle RA, Ryan TW, Mann E, et al. Bridge inspector’s reference manual: Volume 1, volume 2. United States: Federal Highway Administration, 2002. Technical report.
3. Sun L, Shang Z, Xia Y, et al. Review of bridge structural health monitoring aided by big data and artificial intelligence:
from condition assessment to damage detection. *J Struct Eng* 2020; 146(5): 04020073.

4. Yang Y, Wang ZL, Shi K, et al. State-of-the-art of the vehicle-based methods for detecting the various properties of highway bridges and railway tracks. *Int J Struct Stab Dyn* 2020a: 2041004.

5. Yang YB, Yang JP, Zhang B and Wu Y. *Vehicle Scanning Method for Bridges*. Wiley Online Library, 2020b.

6. Cerda F, Chen S, Bielak J, et al. Indirect structural health monitoring of a simplified laboratory-scale bridge model. *Smart Structures Syst* 2014; 13(5): 849–868.

7. Lederman G, Wang Z, Bielak J, et al (2014) Damage quantification and localization algorithms for indirect shm of bridges. In: *Proc. Int. Conf. Bridge Maint., Safety Manag.*, Shanghai, China.

8. Xu S, Zhang L, Zhang P, et al (2016) An indirect traffic monitoring approach using building vibration sensing system. In: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM. pp. 374–375.

9. Xu S, Zhang L, Zhang P, et al. An information-theoretic approach for indirect train traffic monitoring using building vibration. *Front Built Environ* 2017; 3: 22.

10. Liu J, Chen S, Bergès M, et al. Diagnosis algorithms for indirect structural health monitoring of a bridge model via dimensionality reduction. *Mech Syst Signal Process* 2020c; 136: 106454.

11. Yang YB, Lin C and Yau J. Extracting bridge frequencies from the dynamic response of a passing vehicle. *Journal of Sound and Vibration* 2004; 272(3-5): 471–493.

12. Lin C and Yang Y. Use of a passing vehicle to scan the fundamental bridge frequencies: An experimental verification. *Eng Structures* 2005; 27(13): 1865–1878.

13. Liu J, Xu S, Bergès M, et al. An Expectation-Maximization Algorithm-Based Framework for Vehicle-Vibration-Based Indirect Structural Health Monitoring of Bridges. *Structural Health Monitoring*, 2019b.

14. Yang Y, Li Y and Chang KC. Constructing the mode shapes of a bridge from a passing vehicle: a theoretical study. *Smart Structures Syst* 2014; 13(5): 797–819.

15. Malekjafarian A and OBrien EJ. Identification of bridge mode shapes using short time frequency domain decomposition of the responses measured in a passing vehicle. *Eng Structures* 2014; 81: 386–397.

16. González A, OBrien EJ and McGetrick P. Identification of damping in a bridge using a moving instrumented vehicle. *J Sound Vibration* 2012; 331(18): 4115–4131.

17. Nguyen KV and Tran HT. Multi-cracks detection of a beam-like structure based on the on-vehicle vibration signal and wavelet analysis. *J Sound Vibration* 2010; 329(21): 4455–4465.

18. McGetrick PJ and Kim CW. A parametric study of a drive by bridge inspection system based on the morlet wavelet. in: *Key Engineering Materials*, volume 569. Trans Tech Publ, 2013, pp. 262–269.

19. Liu J. Scalable bridge health monitoring using drive-by vehicles: Phd forum abstract. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems 2020: pp. 817–818.

20. Eshkevari SS and Pakzad SN. Signal reconstruction from mobile sensors network using matrix completion approach. In: *Topics in Modal Analysis & Testing*, Volume 8. Springer; 2020, pp. 61–75.

21. Liu J, Chen B, Chen S, et al. Damage-sensitive and domain-invariant feature extraction for vehicle-vibration-based bridge health monitoring. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020b, pp. 3007–3011.

22. Mei Q, Gül M and Boay M. Indirect health monitoring of bridges using mel-frequency cepstral coefficients and principal component analysis. *Mech Syst Signal Process* 2019; 119: 523–546.

23. Sadeghi Eshkevari S, Pakzad SN, Takač M, et al. Modal identification of bridges using mobile sensors with sparse vibration data. *J Eng Mech* 2020; 146(4): 04020011.

24. Zhang L and Gao X (2019) Transfer adaptation learning: A decade survey. *arXiv preprint arXiv:1903.04687*.

25. Jiang J and Zhai C. *Instance Weighting for Domain Adaptation in Nlp*. ACL, 2007.

26. Pan SJ, Tsang IW, Kwok JT, et al. Domain adaptation via transfer component analysis. *IEEE Trans Neuronal Networks* 2010; 22(2): 199–210.

27. Cao Y, Long M and Wang J. Unsupervised domain adaptation with distribution matching machines. In: Proceedings of the AAAI Conference on Artificial Intelligence 2018; volume 32.

28. Lao YW, Ren CX, Dao-Qing D, et al. Unsupervised Domain Adaptation via Discriminative Manifold Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

29. Xu S and Noh HY. Phymdan: Physics-informed knowledge transfer between buildings for seismic damage diagnosis through adversarial learning. *Mech Syst Signal Process* 2021; 151: 107374.

30. Saito K, Watanabe K, Ushiku Y, et al. Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2018. pp. 3723–3732.

31. Ganin Y, Ustinova E, Ajakan H, et al. Domain-adversarial training of neural networks. *J Machine Learn Res* 2016; 17(1): 2096–2030.

32. Zhang Y, Liu T, Long M, et al (2019) Bridging theory and algorithm for domain adaptation. *arXiv preprint arXiv:1904.05801*.

33. Caruana R. Multi-task learning. *Machine Learning* 1997; 28(1): 41–75.

34. Luong MT, Le QV, Sutskever I, et al (2015) Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

35. Augenstein I, Ruder S and Søgaard A (2018) Multi-task learning of pairwise sequence classification tasks over disparate label spaces. *arXiv preprint arXiv:1802.09913*. 

---

**Note:** The above text is a sample from a larger document and does not represent the full content of the original page. It includes a variety of technical references and citations related to structural health monitoring and vehicle vibration studies.
Appendix A

Theorem 3. Let $W$ be a hypothesis space on $X$ with VC dimension $d_W$ and $H$ be a hypothesis space on $Z$ with VC dimension $d_H$. If $X_S$ and $X_T$ are samples of size $N$ from $D^X_S$ and $D^X_T$, respectively, and $Z_S$ and $Z_T$ follow distributions $D^Z_S$ and $D^Z_T$, respectively, then for any $\delta \in (0, 1)$ with probability at least $1 - \delta$, for every $h \in H$ and $w \in W$
Figures 15 and 16 show normalized confusion matrices of our HierMUD approach for the damage localization and quantification tasks, respectively.

Appendix B

Proof. This proof relies on the triangle inequality for classification error

\[ e_T(h \circ w; f_T) \leq e_S(h \circ w) + 2e_S(h \circ w; f_S) \]

\[ + \frac{1}{2}d_{\text{HANN}}(Z_T, Z_S) \]

\[ + O \left( \frac{\sqrt{2dh \log(2N) + \log(2/\delta)}}{N} \right) \]

\[ + \frac{1}{2} \sup_{h \in H} \left[ d_{\text{HANN}}(X_S, X_T) \right] \]

\[ + O \left( \frac{\sqrt{2dh \log(2N) + \log(2/\delta)}}{N} \right) \]

\[ + e_T(h^* \circ w^*; f_T) + e_S(h^* \circ w^*; f_S). \]

\[ (33) \]

\[ \varepsilon_T(h \circ w; f_T) \leq \varepsilon_T(h \circ w; h^* \circ w^*) + \varepsilon_T(h^* \circ w^*; f_T) \]

\[ \leq \varepsilon_T(h \circ w; h^* \circ w^*) + \varepsilon_T(h \circ w; h^* \circ w^*) + \varepsilon_T(h^* \circ w^*; f_T) \]

\[ \leq \varepsilon_T(h^* \circ w^*; f_T) + \varepsilon_T(h \circ w^*; h^* \circ w^*) + \varepsilon_T(h \circ w; h^* \circ w^*) \]

\[ \leq \varepsilon_T(h^* \circ w^*; f_T) + \varepsilon_T(h \circ w^*; h^* \circ w^*) + \varepsilon_T(h \circ w^*; h^* \circ w^*) + 2\varepsilon_T(h \circ w^*; f_S) + \frac{1}{2}d_{\text{HANN}}(D_{fT}^X, D_{fT}^X) \]

\[ + \varepsilon_S(h \circ w; h^* \circ w^*) + 2\varepsilon_T(h \circ w; f_S) + 2\varepsilon_T(h \circ w^*; f_T) + \frac{1}{2}d_{\text{HANN}}(D_{fT}^X, D_{fT}^X) \]

\[ \leq \varepsilon_T(h^* \circ w^*; f_T) + \varepsilon_S(h \circ w^*; f_S) + 2\varepsilon_T(h \circ w; f_S) + \frac{1}{2}d_{\text{HANN}}(D_{fT}^X, D_{fT}^X) + \sup_{h \in H} \left[ \frac{1}{2}d_{\text{HANN}}(D_{fT}^X, D_{fT}^X) \right] \]

\[ \leq \varepsilon_S(h \circ w^*; f_S) + 2\varepsilon_T(h \circ w^*; f_S) + \frac{1}{2}d_{\text{HANN}}(Z_T, Z_S) + 4\sqrt{\frac{2dh \log(2N) + \log(2/\delta)}{N}} + \sup_{h \in H} \left[ d_{\text{HANN}}(X_S, X_T) \right] \]

\[ + 4\sqrt{\frac{2dh \log(2N) + \log(2/\delta)}{N}} + e_T(h^* \circ w^*; f_T) + e_S(h^* \circ w^*; f_S). \]
Figure 15. Normalized confusion matrices of our HierMUD approach for the damage localization task. (a) V1→B1→B2, (b) V1→B2→B1, (c) V2→B1→B2, (d) V2→B2→B1, (e) V3→B1→B2, (f) V3→B2→B1.
Figure 16. Normalized confusion matrices of our HierMUD approach for the damage quantification task. (a) V1, B1→B2, (b) V1, B2→B1, (c) V2, B1→B2, (d) V2, B2→B1, (e) V3, B1→B2, (f) V3, B2→B1.