Application of Joint Distributional Adaptation to Bridge Damage Detection

Irene Chang[1] Alva Couch[2]

[12] Tufts University

May 2023

Abstract

Bridges in real life vary in structures and materials. Therefore, it is expensive and challenging to develop a single damage detection model that produces great results on every bridge. The inherent differences on structures have given rise to many studies on transfer learning models that can be trained on a given bridge to analyze the sensor signals of a new bridge. As part of the ongoing expansion of this line of work, this study seeks to evaluate a transfer learning technique called Joint Distribution Adaptation on a dataset of acceleration measurements in a binary damage detection classification task. This technique has previously been showed to produced improved precision on a similar task from Gardner et al [1], and we want to examine its performance on our own data. The dataset in the study was provided by University of Hampshire Civil Engineering department. We analyze the strengths and drawbacks of our model, with a view to guiding the next steps of this study in the direction of developing a generalizable, trustworthy classifier for damage detection on bridges.

1 Introduction

1.1 Motivation

Damage detection on bridges has always been studied extensively for real-life implications. Bridges vary greatly in structures and materials and their damages also differ in severity as well as in structures, which makes training a generalized model that can produce good results on every bridge very expensive. Understanding and formulating these disparities across different structures prove to be a significant component towards building a robust model that is less vulnerable to non-uniformity. This study focuses on applying a previously developed method for general structural damage detection on our bridge dataset and examining its effectiveness and limitations in our specific case. At the end of the project, we hope to shed light on an optimal or near optimal model for our damage detection task.

1.2 Transfer Learning

Transfer learning is a growing area of studies whose aim is to apply existing knowledge learned from some past task to make decisions on a related task. Transfer learning is useful when there is not enough training data to build a comprehensive model, and when expensive efforts are required to retrain the model from scratch for new incoming data.

Here we introduce major terminologies for transfer learning:

- Domain D: composed of 2 parts, a feature space $\mathcal{X} \subseteq \mathbb{R}^n$ and and marginal distribution $P(X)$ where $X \in \mathcal{X}$. Domain is often observed by a number of instances with or without the class label information. Thus, we can write $D = \{\mathcal{X}, P(X)\}$
- Task T: composed of label space $\mathcal{Y}$ and function $f(\cdot)$ to predict class labels, which is not observed but can be learned from the labeled training data $\{(x_i, y_i)|i \in \{1, \ldots, n\}, x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$. From a probabilistic viewpoint, $f(x_i)$ can be written as $P(y_i|x_i)$. Thus, we can write $T = \{\mathcal{Y}, P(Y|X)\}$
- Transfer Learning Problem: improve the outcome of target objective function for target task T_t in the target domain D_t, using knowledge from source task T_s in source domain D_s, in which either $D_s \neq D_t$ or $T_s \neq T_t$.

The most commonly studied transfer learning is when the features of the source and target domains are different in nature, referred to as **domain shift**, denoted as $D_s \neq D_t$, which constitutes a subclass called *domain adaptation*. Domain shift comes in two forms:

- Feature space mismatch $\mathcal{X}_s \neq \mathcal{X}_t$. For instance, if our task is to perform spam vs non-spam classification of emails, then our source might be emails in English while target domain might be in Spanish.
- Marginal probability mismatch $P(X_s) \neq P(X_t), X_s \in \mathcal{X}_s, X_t \in \mathcal{X}_t$. For example, even though the emails are in the same language, the source might be an official/corporate email dataset (which typically has shorter emails with documents, presentations as attachments) while target may be a personal email dataset.

In our task, since the discrepancy emerges from the features, namely in the joint distribution, in our dataset, we employ a domain adaptation technique, called *joint distribution adaptation* on our dataset.

1.3 Joint Distribution Adaptation (JDA)

Mingsheng Long et al [2] devised JDA, an unsupervised domain adaptation technique that aims to conduct distribution alignment for two datasets along with potential dimension reduction. JDA makes the following assumptions:

- D_s is labeled and D_t is unlabeled.
- $\mathcal{X}_s = \mathcal{X}_t, \mathcal{Y}_s = \mathcal{Y}_t$
- $P_s(X_s) \neq P_t(X_t)$, i.e the marginal distributions are different.
- $Q_s(Y_s|X_s) \neq Q_t(Y_t|X_t)$, i.e the class-conditional distributions are different.

The last two conditions are essentially referring to the difference in the joint distributions. The method seeks a transformation $\phi : \mathcal{X} \to \mathcal{H}$ (Gardner et al [1]) from the source and target feature spaces to a common space (that possibly has fewer dimensions than that of the original dataset) where $P(\phi(X_s)) \approx P(\phi(X_t)), Q_s(Y_s|\phi(X_s)) \approx Q_t(Y_t|\phi(X_t))$.

Not requiring labels in the target set, JDA employs a statistical classifier as the pseudo-labeling technique to obtain an estimate $\hat{Y}_t$ of Y_t. In order to bridge the gap between the source and the target distributions, JDA minimizes the distance between the class-conditional sample means of the source and the target data. This is done through the distance matrices called Maximum Mean Discrepancy (MMD), which is computed separately for each of the label in the label space following the schema below:

$$(M_c)_{ij} = \begin{cases} \frac{1}{\left(n_s^{(c)}\right)^2}, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_s^{(c)} \\ \frac{1}{\left(n_t^{(c)}\right)^2}, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_t^{(c)} \\ \frac{-1}{n_s^{(c)} n_t^{(c)}}, & \begin{cases} \mathbf{x}_i \in \mathcal{D}_s^{(c)}, \mathbf{x}_j \in \mathcal{D}_t^{(c)} \\ \mathbf{x}_j \in \mathcal{D}_s^{(c)}, \mathbf{x}_i \in \mathcal{D}_t^{(c)} \end{cases} \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{D}_s^{(c)} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathcal{D}_s \wedge y(\mathbf{x}_j) = c\}$ is the set of instances of class c in the source data, $y(\mathbf{x}_j)$ is the true label of $\mathbf{x}_i$; $\mathcal{D}_t^{(c)} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathcal{D}_t \wedge \hat{y}(\mathbf{x}_j) = c\}$ is the set of instances of class c in the target data, $\hat{y}(\mathbf{x}_j)$ is the pseudo label of $\mathbf{x}_i$; and $n_s^{(c)}$, $n_t^{(c)}$ is the number of source and target data belonging to class c, respectively.

The optimization problem is then formulated as:

$$\min_{\mathrm{A}^T \mathrm{KHK}^T \mathrm{A} = I} \sum_{c=0}^{C} \mathrm{tr}(A^T K (M_c) K^T A) + \lambda ||A||_F^2 \tag{1}$$

where λ is a regularization parameter, $K = \psi(X)^T \psi(X) \in \mathbb{R}^{n \times n}$ is the kernel matrix from the kernel mapping ψ, $H = I - \frac{1}{n}\mathbf{1}$ is the centering matrix, $||\cdot||_F$ is the Frobenius norm; $A \in \mathbb{R}^{n \times k}$ is the adaptation matrix in question and can be solved in the same way as an eigenvalue problem for the k smallest eigenvectors with the equation below [2]:

$$\left(K \sum_{c=0}^{C} M_c K^T + \lambda I \right) A = K H K^T A \Phi$$

1.4 Dataset

Since the real-life data on our bridges is limited and imbalanced (data for the damaged class is rare compared to the healthy class), we employ simulated sensor data to train our model. Our dataset was provided by the University of New Hampshire's Civil Engineering department. There are 2 different simulated beam structures, coded A and B. The two beam structures represent structurally different types of bridges. For each structure, we have datasets of a number of beams that differ in load history, location of the damage and damage level. The damage level is indicated through their flexural rigidity coefficient, called EI. The EI value (%) represents the severity of the damage of a specific location a beam, with lower value indicating a more damaged status, and EI= 100% indicates an undamaged, or healthy, state. Load history (LH), or structural load, is a force, deformation, or acceleration applied to structural elements [3]. Different LH values signify the different amounts of stress, deformation, and displacement to different beams. Each beam has 9 sensors along its length, each of which is either damaged with $\text{EI} \in \{10, 20, \ldots, 90\}$, or healthy. We denote the damage state of each sensor i as $d_i, i \in \{1, \ldots, 9\}$, where $d_i = \text{EI} \in \{10, \ldots, 100\}$. The beams are created such that only 1 sensor is damaged at a time. In other words, if sensor i is damaged, $d_i \in \{10, \ldots, 90\}$, then $d_1 = \ldots = d_{i-1} = d_{i+1} = \ldots = d_9 = 100$ for that beam.

Over a period of 1000 timesteps, a simulated vehicle is to run over a given beam, and measurements–Acceleration, Displacement, Rotation and Strain–are recorded for all sensors at every timestep. Let the resulting raw time series dataset for each beam be $\mathbf{R} \in \mathbb{R}^{1000 \times 9}$. In this project, we only experiment on the Acceleration datasets.

Each beam is characterized by $(\mathbf{d}, \text{LH}), \mathbf{d} = \{d_1, \ldots, d_9\}$–distinct combinations of damage location, damage level, and the load history. No two beams share the same characterization.

Hence, in each structure type (A and B), we have:

- For each damage level, the number of damaged beams: 9 (number of sensors) $\times$ 200 (number of LH values) = 1800 (beams). Across all 9 damaged levels, we will have $1800 \times 9 = 16200$ distinct beams.
- Number of healthy beams: 9 (number of sensors) $\times$ 200 (number of LH values) = 1800 distinct beams.
- Total: 18000 distinct beams.

Data preprocessing. In order to preprocess and train our model, we follow the steps in Akintunde et al [4] to extract the damage indicators $\mathbf{x}_j \in \mathbb{R}^9, j \in \{1, \ldots, n_{\text{beam}}\}$ where n_{beam} is the number of beams included in our model. Each entry of a damage indicator vector corresponds to a sensor on a bridge, containing an acceleration value that can demonstrate the health of the beam at that specified location. For each bridge, the procedure involves conducting Singular Value Decomposition (SVD) and obtaining the first eigenvector of the left singular matrix.

Thus, the resulting input features for the model are:

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1^T & — \\ & \vdots & \\ — & \mathbf{x}_{n_{\text{beam}}}^T & — \end{bmatrix} \in \mathbb{R}^{n_{\text{beam}} \times 9}$$

2 Method

In this project, we focus on binary classification tasks to predict whether a beam is damaged, regardless of the location and the severity of the damage. We first examine the performance of binary models M_d^i that is trained on the source beam to predict whether there is damage of severity d at location i on the target beam of the same conditions (single location, single damage level classifiers). Next, we carry out sensitivity analysis on the classification results to see how credible the obtained JDA results are on our task.

2.1 Experiment 1: Single location, single damage level JDA

There are 200 beams—that differ in LH values—for any (i, d) combination. Thus, for any single damage level binary classification model M_d^i, there are 800 beams ($n_s^{(1)} = n_s^{(0)} = n_t^{(1)} = n_t^{(0)} = 200$) involved in the training and evaluation process.

Evaluation. Split the target dataset into a training set (75%) and a test set (25%). Then, train the JDA model on the source dataset and the target training set. Produce the predictions for both the target training set and the target test set (which will henceforth also refer to as the test set), for which the accuracy scores can be compared.

For this task, our main interest lies in detecting any existing damage in the structure. In fact, failing to detect damage incurs more cost than failing to identify a healthy bridge, since the former can cost lives. Thus, beside improving the overall accuracy, minimizing false negative (deciding that the bridge is not damaged while in reality it is) is also of equally high importance. Therefore, we will also look into the recall rate for both the healthy and the damage classes on the test set when analyzing the results.

For each of the models, we carry out parameter tuning and present the classification results for the set of parameters that yield the highest precision and highest recall on the test data.

2.2 Experiment 2: Multi-damage level classification

Building separate models for each damage level at each location means that we require some sort of ensemble model to combine these individual classifiers in order to conclude if our beam contains any type of damage at all. Motivated by such a scenario, we want to find out if our JDA algorithm generalizes well to a scenario where our dataset for the damaged class consists of beams sustaining different damage levels. Our hypothesis is that the damage indicators vary greatly from one damage level to another, and for that reason, the model will not achieve test accuracy scores high as those for single damage level classifiers. Let $M^i_{d_1,\ldots,d_9}$ denote the classifier built on the healthy class and a combination of damage levels d_1 through to d_9 at sensor i.

Using the same experiment setup as in section 2.1 and the same evaluation criteria, we create and present the results of the fine-tuned model for multi-damage level case, and provide a comparison to the single damage level models above.

2.3 Robustness Analysis

2.3.1 Experiment 3: Noise injections

In order to determine how trustworthy the results of the fine-tuned JDA model are, we carry out the sensitivity analysis by introducing noise into the raw dataset and observe how much the predictions on the test set vary due to these small disturbances. Let ϵ be a pre-selected amount of noise and let $t = \{1, \ldots, 1000\}$ represent the 1000 timesteps where we record the acceleration. At every t, we E^i_t be a random noise associated with sensor i that follows the discrete uniform distribution over the set $\{-\epsilon, \epsilon\}$. Let $\mathbf{E} = \{\mathbf{E}^1_t, \ldots, \mathbf{E}^9_t\} \in \mathbb{R}^{1000\times 9}$ be the noise matrix that we apply to the raw time series $\mathbf{R} + \mathbf{E} = \mathbf{R}'$. The data is then processed and the model is then built in the same way as before on $\mathbf{R}'$. Note that in this experiment, we build all models on the same set of parameters that yields the best result on the original model.

Evaluation. We look into the number of true negatives (true damage), false negatives (false damage), true positives (true healthy), false positives (false healthy) as we increase ϵ. If the four statistics stay constant against the small noises, then we can conclude that the model is robust against that amount of noise. Geometrically, this means the decision boundary is well-defined. Conversely, if the labels of a lot of instances are flipped even with a very small noise injection, that implies weak reliability of the algorithm to be used in practice to predict damages.

2.3.2 Experiment 4: Amount of target data

In practice, there is plenty of data for the healthy state, but an extremely limited amount of data for the damaged state. At the same time, conducting experiments to obtain damage data is very costly and impractical. Thus, from an engineering perspective, it is also important to evaluate how much damage data in the target set is needed in order to achieve desired accuracy of the model.

We ran the experiment on the full source dataset as before, combined with $f\%$ of the target training set, where $f = \{0, 10, 20, 30, \ldots, 100\}$.

Evaluation. We plot the bootstrapped accuracy scores, as well as the recall for both classes evaluated on the test set.

3 Experiments

In a similar process as in Huang and Wang (2021 [5]), to capture the uncertainty, for each of the experiment below, we draw 1000 bootstrap resamples of the evaluation set with replacement, each of size 500. We then calculate the mean and standard deviation of the statistics in question from all samples. The statistics computed tells us how the result for a given model might vary across evaluation sets drawn from the same empirical distribution.

For experiment 1, we run and analyze the results of the fine-tuned models on the transfer from beam A to beam B to identify the damages at sensors (locations) $i \in \{1, 3, 5, 7\}$ for damage severity levels $d_i \in \{10, 30, 50, 70, 90\}$ (against the "healthy" class). Thus, we build a total of 20 models. The result is displayed in Table 1

Severity / Loc	10		30		50		70		90	
	train	test	train	test	train	test	train	test	train	test
1	0.57±0.03	0.45±0.03	0.68±0.02	0.66±0.03	0.58±0.02	0.52±0.03	0.53±0.03	0.55±0.04	0.50±0.03	0.47±0.03
3	1.00±0.00	1.00±0.00	1.00±0.00	0.99±0.02	0.97±0.01	0.99±0.01	0.83±0.02	0.77±0.02	0.66±0.02	0.6±0.02
5	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	0.93±0.02	0.96±0.01
7	1.00±0.00	0.99±0.04	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	0.75±0.02	0.74±0.02

Table 1: Accuracy (higher means better) after 1000 bootstrap resamples (mean ± std) for JDA model evaluated on target train and test set for the transfer from beam A to beam B across different damage locations and damage levels.

Severity / Loc	10		30		50		70		90	
	damage	healthy	damage	healthy	damage	healthy	damage	healthy	damage	healthy
1	0.45±0.03	0.08±0.27	0.60±0.04	0.72±0.04	0.47±0.02	0.64±0.06	0.48±0.03	0.59±0.05	0.44±0.03	0.53±0.05
3	1.00±0.01	1.00±0.00	1.00±0.01	1.00±0.00	1.00±0.01	0.98±0.02	0.71±0.05	0.86±0.06	0.54±0.03	0.70±0.04
5	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	0.89±0.03	0.97±0.02
7	1.00±0.00	0.99±0.04	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.01	0.77±0.03	0.72±0.03

Table 2: Recall for the healthy class (class 0) and the damage class (class 1) (higher means better) after 1000 bootstrap resamples (mean ± std), evaluated on the test set across different damage locations and damage levels

For experiment 2, to mirror the setup in experiment 1, we combine the data from damage levels 10, 30, 50, 70, 90 and build the binary classifiers for sensors 1, 3, 5, and 7, separately. For each of the settings, the training target data has 156 instances from the healthy class and 744 instances from the damage class combined across 5 levels. In total, we have 4 models.

For experiment 3, the model we analyze is M_{10}^{1}, i.e the binary classification task for damage level 10 at location 1. We inject 10 exponentially increasing noise values into our data, from as small as 0.001 to 10. The results are displayed in Table 3

For experiment 4, we analyze the performance of the models $M_{10}^{i}, i \in \{1, 3, 5, 7\}$, i.e on the beams with damage level 10 at sensors 1, 3, 5, and 7 separately. We have a total of 4 models. Result is shown in Fig. 1

ϵ	True damage	True healthy	False damage	False healthy
0	186	200	0	14
0.001	186	200	0	14
0.0028	186	200	0	14
0.0077	186	200	0	14
0.0215	186	200	0	14
0.0599	186	200	0	14
0.1668	186	200	0	14
0.4641	186	200	0	14
1.2915	185	200	0	15
3.5938	147	200	0	53
10	152	200	0	48

Table 3: Number of true negatives, true positives, false negatives, false positives of the predictions on the target test set upon the introduction of different noise levels.

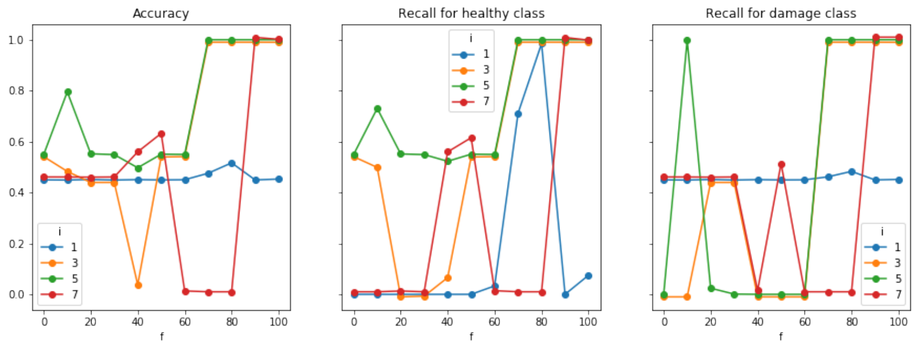

Figure 1: Accuracy, recall rates for both the healthy class (class 0) and the damage class (class 1) after 1000 bootstrap resamples (mean), of a JDA model to identify damage at sensor i as we only include f% of target data of the damage class.

Results

Below we highlight some takeaways from our experiments.

Finding 1: The less severe the damage, the less distinguishable it is from the undamaged beam According to Table 1 and 2, across all sensors i's, there is a general decreasing trend in accuracy score and recall rates as we move to a higher EI value (less damaged). This suggests that the features of less damaged beams are closer in space to the those of the healthy beams, and thus the algorithm fails to classify them.

Finding 2: The algorithm performs poorly on the multi-damage level setting. In this experiment, we found that all the models predicting for four distinct damage locations only produce labels for the damage class (class 1). On the evaluation set of size 300, the train and test accuracy scores are 0.83 and 0.85, respectively, for all four models. They also share the same number of true damage (256), true healthy (0), false damage (44), false healthy (0) on the test set. In this way, the model performs in the same way as a baseline classifier that always predicts the most frequent label in the training set. This indicates that JDA is vulnerable to label imbalance issue, which is not discussed in this report and can be one of the next steps for this project.

Finding 3: JDA algorithm is robust against noises. According to Table 3, given the same setting, i.e the same set of parameters, the breakdown of the predictions stays constant up to the 8th noise injection ($\epsilon = 1.2915$). Provided that the measurements of acceleration on these beams are relatively small, this shows that the decision boundary that JDA constructs on the new space is well-defined, for which the label of any instances are not easily flipped in the presence of small disturbance in the dataset.

Finding 4: Changing the characteristics of the training data has a great impact on the results. According to Fig 1, accuracy and recall rates fluctuate greatly as we vary the proportion of the target training data. Since the models behave inconsistently, we cannot conclude that having fewer target training data correlates with worse performance. We can only make an observation that, across all settings, the models with 90% of the target training set performs most similar to the one with full target training set. We provide a further discussion of this problem in the next section.

4 Discussion and Future Direction

JDA involves two processes: dimension reduction and bringing the source and target data closer together with the use of MMD matrix. The algorithm does not formulaically seek to maximize the separation between the healthy and the damage class. As the performance of the classifier depends greatly on the separability of the classes, we observe that the result fluctuates greatly even after the source and target distributions are aligned. This also matches our observation that the models detecting more damaged beams from the healthy beams perform better than those detecting damages on less damaged, e.g EI=90%, beams.

From experiment 3, we see that JDA is sensitive to changes in hyperparameters. With a specific set of parameters, the classification results are robust against disturbances in the data. However, relatively small changes in the parameters can produce very different outcomes. Moreover, the performance of JDA does not show any linear correlations to the relevant parameters. Our current experiment has not conducted an extensive grid search of the optimal set of parameters. So far, we only tune each parameter individually. Since it is suggested that JDA is very sensitive to changes in hyperparameters, a more systematic way to explore various combinations will not only increase the likelihood of finding the parameter that yield the optimal results, but might also shed light on performance patterns of JDA that we have yet to see.

Our current study analyzes the acceleration data from the sensor. We can try applying this method to other measurements (displacement, rotation, strain) and see how different the precision and the stability of JDA are compared to our current models.

From experiment 4, we have suggested that JDA also has high sensitivity to the nature of the dataset. A slightly different set of data requires a different set of optimal parameters. A more extensive analysis to learn about the compatibility of a given data characteristics and a specific set of hyperparameters can be a focus of future study.

There is a lot of literature in the field of civil engineering concerning transfer learning using both statistical machine learning and deep learning. In expanding our research in the direction of reducing the current model's weaknesses, we can find a more precise and interpretable model for our task. For example, Wang et al [6] introduces Balanced Distribution Adaptation, which improves JDA by formularizing the mismatch in the joint distribution between the source and target data, as well as addressing class imbalance. Gardner et al [1] also suggested pairing a simple neural network with JDA as the main classifier in the algorithm. Currently, kNN is being used as the main classifier. We can also explore different statistical classifier choices to see if they can tackle the sensitivity issues that the current algorithm is having.

5 Conclusion

In this project, we explore the performance of JDA on our bridge beam data to identify damages of different levels and at different locations. We observed that JDA classification results are robust against noises in the data, but is sensitive to variations in hyperparameters and in the composition of the dataset in training. We make suggestions for future ad-hoc study, as well as other approaches that have potential to better address this task.

6 Acknowledgement

This project would not have been possible but for the assistance from many knowledgeable experts on the team. I would like to extend my sincere thanks to Professor Alva Couch from Tufts Computer Science Department and Professor Abani Patra from Tufts Mathematics Department for guiding me throughout both semesters on this project. Special thanks to Professor Yashar Eftekhar Azam and Burak at University of New Hampshire for providing me with the dataset and sharing a lot of valuable expertise on the subject matter in our meetings.

References

[1] P. Gardner, X. Liu, and K. Worden. "On the application of domain adaptation in structural health monitoring". In: *Mechanical Systems and Signal Processing* 138 (2020), p. 106550. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2019.106550. URL: https://www.sciencedirect.com/science/article/pii/S088832701930771X.

[2] Mingsheng Long et al. "Transfer Feature Learning with Joint Distribution Adaptation". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 2200–2207. DOI: 10.1109/ICCV.2013.274.

[3] *Structural load*. Jan. 2023. URL: https://en.wikipedia.org/wiki/Structural_load.

[4] Emmanuel Akintunde et al. "Unsupervised Machine Learning for Robust Bridge Damage Detection: Full-Scale Experimental Validation". In: *Engineering Structures* 249 (2021), p. 113250. ISSN: 0141-0296. DOI: https://doi.org/10.1016/j.engstruct.2021.113250. URL: https://www.sciencedirect.com/science/article/pii/S0141029621013742.

[5] Zhe Huang and Liang Wang. *The Tufts fNIRS to Mental Workload Dataset: Toward Brain-Computer Interfaces that Generalize*. 2021. URL: https://openreview.net/forum?id=QzNHE7QHhut.

[6] Jindong Wang et al. *Balanced Distribution Adaptation for Transfer Learning*. 2018. arXiv: 1807.00516 [cs.LG].