Generalizability of Deep Domain Adaptation in case of Sample Selection Bias

Emiel Witting[1†]

Supervisor(s): Joana de Pinho Gonçalves[1], **Yasin Tepeli**[1]

[1]EEMCS, Delft University of Technology, The Netherlands
†

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Emiel Witting
Final project course: CSE3000 Research Project
Thesis committee: Joana de Pinho Gonçalves, Julián Urbano Merino, Yasin Tepeli

Abstract

Domain adaptation allows machine learning models to perform well in a domain that is different from the available train data. This non-trivial task is approached in many ways and often relies on assumptions about the source (train) and target (test) domains. Unsupervised domain adaptation uses unlabeled target data to mitigate a shift or bias difference between the domains. Deep domain adaptation (DDA) is a powerful class of these methods, which utilizes deep learning to extract high-level features that are common across the domains and robust against the shift. These algorithms adapt to a specific target domain. This has two possible downsides. Firstly, the model might not generalize and thus require retraining for each new domain. Secondly, obtaining data for the target domain(s) might be difficult. There can be situations where both source and target domains originate from a "global" domain, from which the samples are selected in a biased way. We explore a new application of existing DDA methods and answer the question: *How effective is deep domain adaptation when adapting with the global domain, instead of the target domain, in case of sample selection bias?*. Results with synthetic data show that where target adaptation works, global adaptation also improves accuracy compared to supervised learning, although to a lesser extent.

1 Introduction

Supervised machine learning models optimize performance on a set of "source" (train) samples, intending to generalize to any sample during deployment or testing. This assumes that the training samples are independent and identically distributed random variables of the "target" (test) domain. In practice, however, this might not be the case. The shift between the domains can result in decreased performance when a model is deployed.

Domain shift has been observed in multiple fields. For example, a widely used biomedical sepsis prediction model used a biased method of labelling training data. This caused accuracy when deployed to be lower than the developers reported [1][2]. Data sets used in computer vision also contain biases. Models trained on one data set perform noticeably worse on others when trying to classify the same object types [3]. Furthermore, simulated environments can be used to train robots, but the virtual environment may not be a perfect representation of the real world [4].

Various so-called domain adaptation[1] (DA) methods have been successfully used to mitigate the shift between the source and target domains [6][7][8]. Unsupervised domain adaptation (UDA) methods use unlabeled target data to identify the shift [9]. A subclass of those, called deep domain adaptation (DDA), uses artificial neural networks to extract input features that are robust against the domain shift. The deep learning component makes it suitable for computer vision and other high-dimensional applications [5].

Shift between domains can be caused by selection bias [10]. In the context of domain adaptation, this is usually interpreted as the source domain being a biased selection of the target domain. However, source and target data could both be (differently) biased selections of another domain, which we refer to as the "global" domain. In the aforementioned example of bias in computer vision data sets, each individual data set can be seen as a biased sample of all existing images on the internet. The latter forms the global domain.

To the best of our knowledge, current (deep) domain adaptation methods adapt to specific target domains and do not use the global domain. Adapting to the global domain could be beneficial for two reasons. Firstly, it might promote generalization that carries over to any unseen target, alleviating the need for data collection and retraining for each new target domain. It might also be costly, or not possible at all, to collect representative target data. We evaluate the applicability of existing DDA methods to this novel use case. The main research question is thus:

> *How effective is deep domain adaptation when adapting to the global domain, instead of the target domain, in case of sample selection bias?*

This is divided into two sub-questions:

1. How does adapting to global data affect DDA accuracy in case of varying amounts of selection bias on features only?
2. How does adapting to global data affect DDA accuracy in case of selection bias that causes varying amounts of shift in both features and labels?

The proposed global adaptation is compared against baselines on synthetic data sets, with performance measured in terms of accuracy on binary classification problems. To limit the scope, two DDA methods are considered. The domain adversarial neural network (DANN) [11] encodes the input features such that labels can be classified, but a second classifier network cannot distinguish the source from the target data. The other method pre-trains a contractive autoencoder [12] to extract latent features, on which a classifier is trained. This classifier mitigates domain shift by minimizing a domain discrepancy measure for one of the hidden layers [13][6].

The remainder of this paper is structured as follows. In the next section, 2, background information on the algorithms and their assumptions on bias type are discussed. Section 3 explains the experimental setup. The results are listed and discussed in section 4. The main conclusions and recommendations for future work are summarized in section 5. Lastly, a critical reflection on this research and its reproducibility is included in section 6

2 Background

It should be emphasized that there are many approaches to (unsupervised) domain adaptation. These rely on different

[1] In literature, domain adaptation is sometimes referred to by the umbrella term 'Transfer Learning'. The formal relation between these two is explained well by Csurka [5].

assumptions about the shift between the source and target domains. This is relevant, as it guided the choice of data sets for the experiments in this research, and helps explain certain results. The following sections will therefore explain the principles behind the methods and types of domain shift.

2.1 Deep Domain Adaptation

Unsupervised domain adaptation methods, which combine labelled source data with unlabeled target data, can be categorized as either feature-based, sample-based or inference-based [9]. Feature-based methods transform the input features such that the same classifier trained with source data also performs well on target data (Fig. 1). Under certain assumptions, a transformation that aligns the distributions of the source and target features has this property [14].

This research focuses on deep domain adaptation (DDA), a class of feature-based methods utilizing deep learning. Deep learning is especially suitable for high-dimensional data such as images as it can leverage large (convolutional) networks [5]. Due to this, the feature transformation is often from high to low dimensional data. The process of aligning distributions through transformation can then also be interpreted as extracting high-level features that are domain-invariant.

One of the two methods we chose to represent DDA is the domain-adversarial neural network (DANN) [11]. This architecture consists of an encoder network that feeds into both a label classification[2] head and a domain classification head (Fig. 2). The encoder minimizes label classification loss $\mathcal{L}_c$ while maximizing domain classification loss $\mathcal{L}_d$, weighted by trade-off parameter λ:

$$\mathcal{L} = \mathcal{L}_c - \lambda \mathcal{L}_d \tag{1}$$

Maximizing domain confusion implicitly aligns the distributions of encoded source and target features. This method was chosen as it is the most straightforward and interpretable model using the domain-adversarial network, but a range of more advanced algorithms have been based on it [7].

The other DDA approach covered in this research uses auto-encoders to extract the most relevant components of data. This can be done with neural networks that compress [12] and reconstruct the input features, or remove augmented noise [16]. A label classifier can be trained simultaneously on the encoded features to encourage useful representations [6] (Fig. 3), using combined loss:

$$\mathcal{L} = \mathcal{L}_{\text{dec}} + w_{\text{aux}} \mathcal{L}_{\text{aux}} \tag{2}$$

Where $\mathcal{L}_{\text{dec}}$ is the reconstruction loss between the input and the decoder output, which can be computed for every sample regardless of missing labels. The auxiliary classifier loss $\mathcal{L}_{\text{aux}}$ is only nonzero for the labelled source samples, and if available, a small amount of labelled target samples. Parameter w_{aux} determines the balance between the classification and reconstruction power of the encoder.

This kind of encoding reduces irrelevant noise and extracts common features when original inputs are high-dimensional. However, it does not necessarily remove shifts between domains. One method minimizes domain shift by introducing a domain distance loss on one of the hidden layers of the classification network, during a so-called adaptation stage (Fig. 4) [6]. This uses maximum mean discrepancy (MMD) for the loss, which is related to a statistical test to determine whether two samples are from the same distribution [13]. Again, this can be incorporated through a weighted sum of loss functions:

$$\mathcal{L} = \mathcal{L}_{\text{c}} + w_{\text{MMD}} \mathcal{L}_{\text{MMD}} \tag{3}$$

Where $\mathcal{L}_{\text{MMD}}$ is the maximum mean discrepancy between the activations of a hidden layer for source and target samples, and w_{MMD} is a constant weight parameter.

To summarize the two methods. DANN trains an encoder network that both serves the role of extracting features and removing domain shift, where the shift is quantified using a domain classifier network. The second method trains an autoencoder just for extracting features, then removes leftover domain shift in the second stage by minimizing a statistical domain distance measure. Notable differences are, respectively, the one-stage versus two-stage training and the deep learning versus statistics-based domain shift measure.

As mentioned before, for all of the above to work, mitigating shift by aligning the features also has to align corresponding class labels. This is not guaranteed with DANN, which is discussed in more detail in Section 2.3 "A gap between theory and algorithm" by Sicilia et al. [17]. That section also mentions that the base DANN algorithm is not expected to perform well when there is a shift in label balance between domains.

Lastly, we note that DANN has already shown ability to generalize with multiple domains. This was done with a modified architecture that combines labelled data from multiple source domains [17] to improve the accuracy on a single target domain. However, this does not guarantee success for our application, as ours uses one source domain and intends to generalize to many target domains instead. Furthermore, while the global domain might be interpreted as containing multiple target domains, we lack the domain labels to distinguish them from each other.

2.2 Bias

The premise of this research is that the source and target domain are biased samples of a global domain. Sample selection bias is defined by Moreno-Torres et al. [18] as selecting samples based on a random variable s, such that:

$$\begin{aligned} P_{\text{selected}} &= P(s = 1 \mid x, y) \\ P_{\text{biased}}(x, y) &= P_{\text{selected}} \cdot P(x, y) \end{aligned} \tag{4}$$

For features x and labels y, where the original data is distributed according to $P(x, y)$ and the biased to $P_{\text{biased}}(x, y)$. Note that the biased data cannot contain samples that would not exist in the original data (those where $P(x, y)$ is zero). The aforementioned study also defines different types of bias and shifts based on that selection probability. For instance, if the selection s only depends on x, independent of y, this causes a "covariate shift", with:

$$\begin{aligned} P_{\text{biased}}(y \mid x) &= P(y \mid x) \\ P_{\text{biased}}(x) &\neq P(x) \end{aligned} \tag{5}$$

[2] For consistency of this research, this is explained in terms of label classification, but a regression network can also be used for most methods.

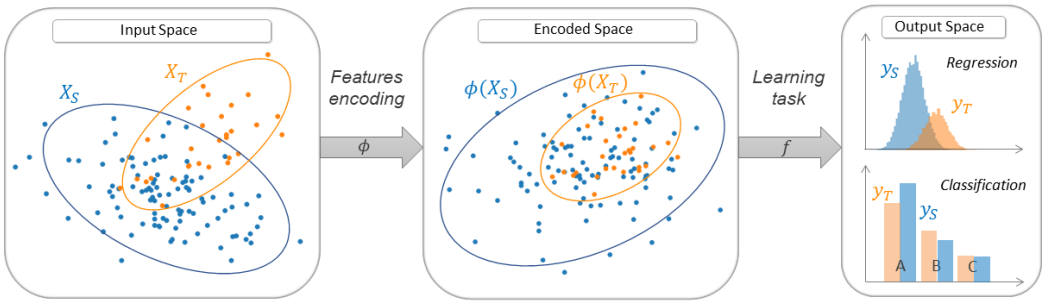

Figure 1: Principle behind feature-based domain adaptation methods. Some transformation or encoding Φ aligns the feature distribution from the source (X_S) and the target domain (X_T), removing shift. A classifier or regression model is trained on the encoded data. [15]

This does not change the conditional probability of labels: the same part of the feature space will be associated with the same label (proportion) for the original and the biased distribution. In the context of classification models, this can also be interpreted as the ideal decision boundary not changing. However, if $P_{\text{biased}}(x)$ is low enough in some regions that no samples are drawn from there, different decision boundaries might be learned that only perform optimally on the limited data.

As discussed before, the two DDA methods assume that the same transformation that aligns features also aligns corresponding labels. While it is not always mentioned explicitly[3], this seems to conversely imply that the features and labels of the biased domain contain a joint transformation from those of the original domain. Hence, "undoing" that transformation based on the known features also undoes the equivalent shift on the unknown labels.

This type of join shift of features and labels can be classified as a type of "concept shift" [18]. Concept shift occurs when the selection is based on a dependent combination of x and y such that:

$$\begin{aligned} P(x \mid y)_{biased} &\neq P(x \mid y) \\ P(y)_{biased} &= P(y) \end{aligned} \tag{6}$$

Besides the reason above, it can also be inferred that the algorithms might be suited better for concept shift, as it guarantees by definition that class balance ($P(y)$) remains equal. It has already been mentioned that at least DANN is sensitive to this.

N.B. that these definitions were given in the context of the source domain being a biased selection of the target domain. The relative shift between two domains that are both biased selections of a third global domain might have different properties.

3 Methodology

Deep domain adaptation methods were evaluated on synthetic data sets, with induced bias. Section 3.2 describes the bias-inducing procedures, DDA method implementation, are discussed in 3.1, and the last section, 3.3, explains the evaluation procedure.

3.1 Data Set Generation

Synthetic classification data sets, for later inducing bias, were generated randomly. The generator[4] creates a given amount of clusters for each class, which are generated as normally distributed points around the vertices of a hypercube. Afterwards, noise and interdependence between features are added. For simplicity, the data sets were always binary classification problems. In early experiments, a feature space of 10 dimensions and 200 samples per domain was chosen arbitrarily. However, this led to strong overfitting, possibly due to the curse of dimensionality [20]. This was resolved for the final experiments after reducing the dimensions to five and using 1000 samples per domain. The DDA methods are often based on extracting features and reducing dimensionality, but reducing dimensions might lose too much information if there are only five features. Therefore, two of the five features were linear combinations of the first three, emulating high-dimensional data that can be reduced. Note that this linear relationship does not necessarily hold after inducing bias.

Covariate shift

A covariate shift was induced for the first research subquestion, regarding selection bias on features only. The initial data set was randomly split into global, source candidate and target candidate samples. The global samples were left unbiased. Source and target data were sampled with replacement from their candidate sets, both proportional to a different (multivariate) normal distribution over the features to induce bias. The standard deviation, or scale, of these bias distributions, is defined by parameter σ and is the same for every feature dimension. The means, or locations, are Euclidian distance b apart, on opposite sides of the global mean. This was done by computing the mean $\overline{x}$ of all data and a random vector v of specified length b, then choosing the distribution means as $\overline{x} + \frac{v}{2}$ and $\overline{x} - \frac{v}{2}$. Before these steps, the initial data was normalized to have a standard deviation of one in each dimension so that values for σ and b had a consistent effect.

[3] For example, by Ben-David et al. [14], in section 3.1, the authors justify the aligning of labels only with "This embodies our domain adaptation assumption"

[4] Using Scikit-learn's `make-classification` [19]

Concept shift

For the second sub-question, a joint shift in features and corresponding labels was simulated. This will be called concept shift, as described in section 2. Given the premise of this research, the concept shift between source and target must be caused by biased sampling from the global domain. We justify how this can occur and base our implementation on the following assumption: There exist distinct pseudo-domains, each being concept-shifted variations of each other. The global domain consists of all these related pseudo-domains, the source and target domains of specific ones. The distinct pseudo-domains could for instance be medical images from different scanners.

Thus, to induce bias, the initial data is first divided randomly into a number of pseudo-domains. For the experiments, this number was arbitrarily chosen to be four. Points receive a different random translation in feature space for each pseudo-domain. The translation in each dimension is sampled uniformly from $U(-\frac{b}{2}, \frac{b}{2})$, using bias parameter b. Again, this was done after normalizing the initial data to have a standard deviation of one. Note that, unlike the covariate shift procedure, this does not guarantee that the shift is exactly of distance b. Because it is random, it is possible for two pseudo-domains to receive the same, or similar, translation and have little relative shift, even when b is large. The random component was used with the purpose of distributing the pseudo-domains uniformly. If the translation length was made constant, like in section 3.1, it would result in all pseudo-domains lying on a hypersphere, with none in the centre.

Finally, global, source and target data were sampled from the pseudo-domains. Global domain samples were drawn randomly from all of them. The source and target domains were each assigned a different pseudo-domain at random, from which their data was sampled. The points that were already drawn for the global domain were excluded from this.

3.2 DDA Methods

The DANN implementation used a library [15] and works as described in section 2. For more detail, we refer the reader to the original paper on the algorithm [11]. The architecture was mostly left at the default configuration, with the only exception being the activation function of the last neuron of the classifier head. By default, this was linear, but a sigmoid function was chosen as it is more suitable for the binary classification task. For the same reason, the loss function corresponding to that label classification neuron was changed from mean-squared error to binary cross-entropy (see appendix A). Figure 2 shows the resulting architecture used for all DANN experiments.

The autoencoder method that was used follows the same structure as the two-stage contractive autoencoder using maximum mean discrepancy by Li et al. [6], also described in section 2. Compared to their architecture, ours (Fig. 3 and 4) is smaller and does not have convolutional layers. This is partially due to our synthetic data sets being of lower dimensions, but also to use a comparable architecture to DANN. For instance, the (auxiliary) classifier network uses the exact same architecture as DANN's classifier. Furthermore, the

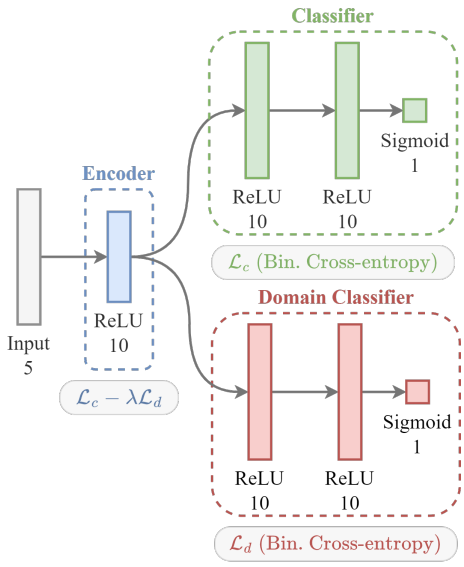

Figure 2: Architecture of the DANN method used in experiments, where "ReLU 10" indicates a network layer with 10 neurons and the rectified linear unit activation function. The grey boxes show the loss function that is minimized by the weights of each part of the network.

transfer network for the autoencoder is the same as the encoder of DANN, which perform partially the same task as discussed in section 2. The encoder network of three neurons is smaller than the input of five, to achieve the contractive autoencoder effect. MMD loss[5] is computed between activations of the transfer layer across the source and target samples. the MMD minimization process was handled automatically by the gradient descent framework that was used for training the network [21]. While most of the problem formulation is similar to that of Li et al., it should be noted that they assume a small number of target labels are available, which is not the case in this research.

3.3 Experimental setup

To evaluate the effectiveness of adapting to the global domain, accuracy on the target domain was measured with four configurations: *source-only, global adaptation, target adaptation, and target-only*. Source-only and target-only form the baselines that indicate the impact of the domain shift, and the effectiveness of the two adaptation methods can be judged by how well they bridge the (expected) gap between target-only and source-only. We refer to this as the adaptation gap. Normally, the DDA methods expect labelled source data and unlabeled target data. Here, the adaptation methods also use labelled source data, but either unlabeled global or target data, respectively. Source-only and target-only represent supervised learning. To change as few variables as possible, the supervised models are emulated by reusing the DDA model architectures and disabling adaptation. This was done

[5]Implemented using https://www.idiap.ch/software/bob/docs/bob/bob.learn.tensorflow/v1.2.0/, see Gretton et al. [13] for the theoretical background.

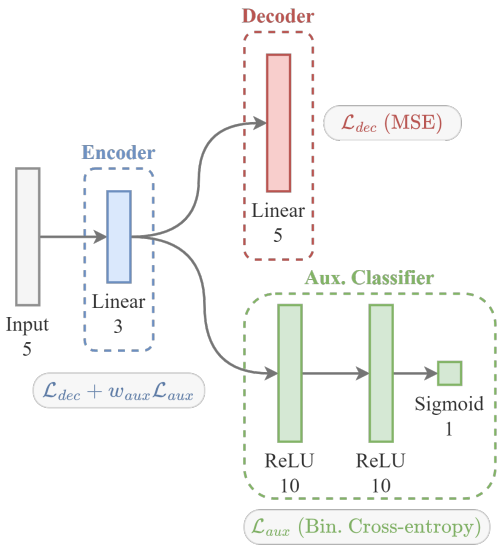

Figure 3: Architecture of the autoencoder method during pre-training, where "Linear 5" indicates a network layer with 5 neurons and the linear activation function. The grey boxes show the loss function that is minimized by the weights of each part of the network, with MSE standing for mean squared error.

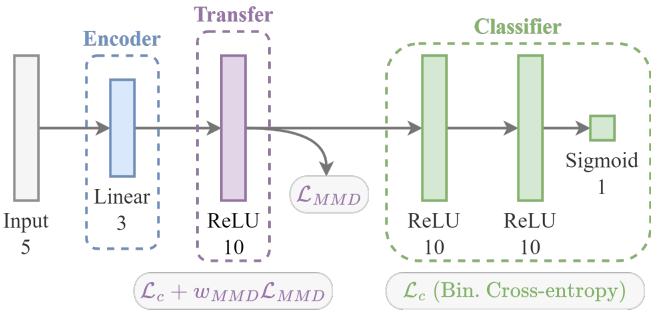

Figure 4: Architecture of the autoencoder method during adaptation, where "Linear 3" indicates a network layer with 3 neurons and the linear activation function. The grey boxes show the loss function that is minimized by the weights of each part of the network. The MMD loss is based on the discrepancy between activations of the transfer layer across the source and target samples. The encoder is fixed and does not optimize any loss function at this stage.

by setting parameters λ and w_{MMD} to zero for DANN and the autoencoder, respectively. Furthermore, in the pre-training stage of the autoencoder, it is given only source data or only target data to reconstruct.

Performance in the case of covariate shift and concept shift, as described in section 3.1, corresponds to the two research sub-questions. To be able to give a better-informed answer, strong and weak versions of both types of bias were tested, totalling four bias configurations. Weak covariate shift used $\sigma = 1, b = 2$, strong used $\sigma = 0.75, b = 3$. Weak concept shift used $b = 2$, whereas strong concept shift used $b = 4$. These values were chosen such that for weak shift the source and target domains were (partially) overlapping, while strong shift caused nearly no overlap. Overlap was expected to be a possibly impactful factor, as both DDA methods are inherently unable to make different predictions for source and target samples if they are in the same position in the feature space.

For both methods, for each bias type and amount, for each of the four configurations, target accuracy was measured. First, parameters λ, w_{MMD}, and w_{aux} were determined for each combination (except when set to zero as discussed) by fifteen trials of hyperparameter search [22], optimizing mean target accuracy on ten data sets. With the best parameter values, target accuracy is measured again, this time on another fifty data sets to aggregate in a final box plot. At all times, a train-test split of 70% per domain was maintained for reliable validation accuracy.

4 Results and Discussion

Before discussing the DDA method results, we quickly inspect some generated data sets, to demonstrate that the bias-inducing procedure works. For covariate shift (Fig. 5a and 5b), samples with similar features have the same label, regardless of domain. However, the source and target samples are clearly concentrated on different sides of the global data. These two properties correspond to the definition in eq. 5. For concept shift (Fig. 5c and 5b) it can be observed that the source and target domains contain a similar pattern, but are offset by a translation. Unlike with covariate shift, the label balance seems to be preserved across domains, satisfying eq. 6. Lastly, it holds for every bias type that source and target samples are always accompanied by similar global samples. This shows that the source and target domains can indeed be interpreted as selections of the global domain.

The experiments, as described previously, were run with four configurations: source-only, global adaptation, target adaptation and target-only. The results over fifty runs are visualized in box plots (Fig. 6). When the following sections mention significance, this refers to a two-sided Welch's t-test [23], which compares means and accounts for unequal variance. The test assumes normally distributed data, so the result should be considered less reliable when the box plots show a skewed distribution. The accuracy of each configuration was tested against source-only, totalling 24 comparisons. Bonferroni correction for multiple testing was applied, yielding the significance level $\alpha/m = 0.05/24 \approx 0.002$. Increases or decreases mentioned as accuracy percentages will always refer to absolute differences, not proportions. Numeric results corresponding to the box plots and hyperparameter values can be found in appendix B and C, respectively.

4.1 Covariate Shift

In the case of weak covariate shift (Fig. 6a and 6b), no configuration performed significantly better or worse than source-only, regardless of algorithm. For stronger covariate shift (Fig. 6c and 6d), the only significant difference is the increase for target-only. This suggests that, depending on the strength of the shift, there is either no significant adaptation gap or there is but the adaptation methods do not improve compared to supervised learning.

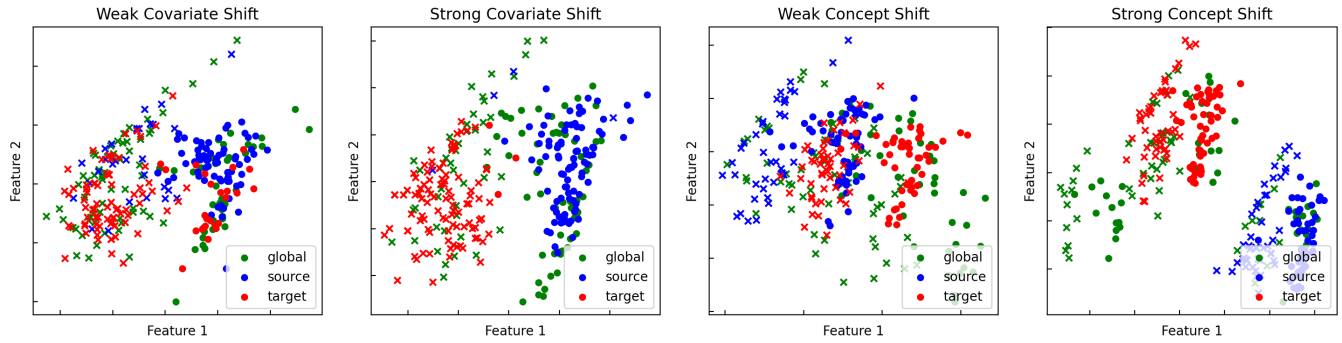

Figure 5: Synthetic data sets that were generated by inducing bias in four different ways, starting with the same initial classification data. The circle or cross-shaped markers denote the label, and the colour shows the domains that points were assigned to. The bias configuration is the same as used in other experiments but the data is of lower dimension and with smaller sample counts, for visualization purposes. Axis scales were set dynamically to fit all data.

It is straightforward to conclude from this that the adaptation had no impact. However, these results are the best possible ones as found through hyperparameter optimization. Notably, the parameters that were found for the strong covariate shift had low λ and w_{MMD} (Table 2 and 3 in appendix). These control the amount of adaptation, indicating that the best results are achieved when adaptation is decreased or disabled. To confirm this hypothesis, the covariate experiments were run again, using adaptation parameter values that worked well for a strong concept shift. This showed that using adaptation with DANN can indeed decrease accuracy for both weak and strong covariate shifts (appendix B). The autoencoder method also caused a decrease, but only with strong shift and target adaptation.

Ineffective adaptation can be explained by the main assumption of the DDA methods, that aligning features also aligns the labels. In the case of covariate shift, the labels across domains already are aligned (see Fig. 5a), so the optimal decision boundary for source data also works for target data. Additional shifting of data and decision boundaries by adaptation methods would then be unwarranted. The adaptation gap for strong covariate shift could be caused by insufficient source data across the feature space for learning the complete decision boundary.

4.2 Concept Shift

In the cases of concept shift, there was a significant adaptation gap between source-only and target-only. For weak concept shift (Fig. 6e and 6f) this was a difference of 20% or 21% in median accuracy, for DANN and the autoencoder respectively. For strong concept shift (Fig. 6g and 6h) the gap was larger, with 34% and 28% differences. The adaptation gap can be explained by the translated (pseudo-)domains having translated labels, and corresponding optimal decision boundaries. The learned model for the source domain is then not guaranteed to work for the target.

With DANN, target adaptation significantly improved performance and covered a large part of the adaptation gap. This is to be expected, as DANN is intended for target adaptation and assumes this type of joint shift of features and labels. For global adaptation, the approach we test in this research, the accuracy also increased compared to supervised learning, although by a smaller amount. In the case of a strong concept shift, this was statistically significant, and the median accuracy increased by 19%. For weak concept shift, while having an 8.5% increase of median, the difference was not statistically significant. At the risk of sacrificing scientific integrity, however, we will tentatively consider it an improvement. This is because we (1) consider the effects of strong and weak concept shift to be positively dependent on each other, and (2) the box plots show that source-only is skewed towards low median accuracy, while global adaptation isn't. These go against the independence and normality assumptions of the test [23] in favour of considering global adaptation an improvement. Finally, we combine this justification with the p-value of 0.005 already being of a similar scale as the threshold of 0.002. Note that no other significance test results would change if the threshold was increased to this level. Regardless of significance, it should be noted that global adaptation is less consistent than target adaptation, with standard deviation being 2.7 or 2.3 times as large, for strong and weak shifts respectively.

The autoencoder method with target adaptation also significantly increased accuracy in the case of strong concept shift, by a 19% difference of medians. With a weaker concept shift, there was an improvement of 8.6%, although not statistically significant. We do not consider the corresponding p-value of 0.030 close enough to the threshold to use the same justification as above to conclude otherwise. Global adaptation yielded no significant improvements.

Comparing the two methods, adaptation with DANN achieves better accuracy than the autoencoder for both global and target adaptation. There are a few possible explanations for this. Firstly, when the autoencoder pre-train stage compresses five-dimensional features into three dimensions, it is possible that it loses information that would be necessary for adaptation. Furthermore, in the adaptation stage, the DANN's adversarial domain classifier might be a more accurate mea-

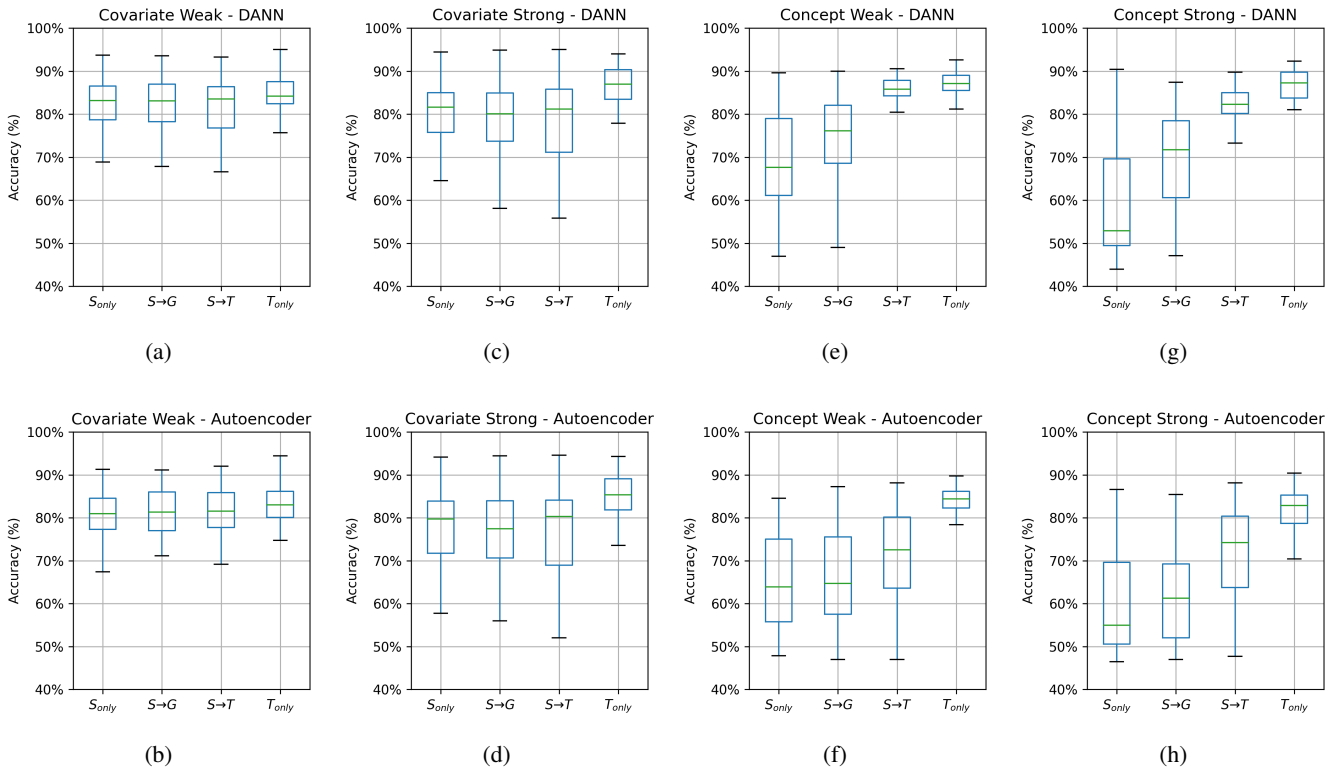

Figure 6: Accuracy on the target domain, after training DANN and the auto-encoder model with different configurations of unlabelled and labelled set. S_{only} and T_{only} use only one labelled domain for supervised learning, source and target, respectively. For $S \rightarrow G$ and $S \rightarrow T$, the left-hand side, source, is labelled, and the right-hand side, global or target, is unlabelled. Results were measured for different types and amounts of bias, as indicated by their title. Each boxplot aggregates results across 50 datasets, outliers are not shown.

sure of domain shift than MMD loss, or produce better gradients for guiding weights to an encoding that minimizes the shift. Lastly, the DANN implementation used a public software framework, unlike the autoencoder, suggesting that its architecture has been tested and optimized based on more experience.

5 Conclusions and Future Work

To summarize, the main goal of this research was to assess the effectiveness of existing deep domain adaptation (DDA) methods when adapting to the global domain, of which both source and target are biased selections. Results with synthetic data sets showed that, given a suitable type of bias, global adaptation can offer a significant improvement over supervised learning. However, global adaptation was less consistent and effective than the usual target adaptation, so it should be considered an inferior alternative. It might be used if target adaptation is not feasible, due to lacking target data or having too many target domains.

To elaborate on the suitable type of bias. This refers to a collection of distinct domains existing, each containing a joint shift in both features and corresponding labels relative to each other (concept shift). The source and target were assumed to be two specific domains, and the global domain contains data from all of them. For selection bias on features only (covariate shift), adaptation had either no impact or a negative one. The premise of unsupervised domain adaptation is that there is a lack of labelled target data. So it can be difficult to estimate if a data set indeed has a suitable type of bias. This is not necessarily unique to global adaptation, however, the same issue holds for target adaptation. If a data set is known to work well with target adaptation, this might indicate that global adaptation could also work, although possibly to a lesser extent.

Of the two methods used, the domain adversarial neural network [11] was more effective. The other method was based on an autoencoder and minimizing maximum mean discrepancy [6], this was less effective at target adaptation and yielded no significant adaptation when using the global domain.

Since this novel approach is beneficial in some cases, further research is warranted. This research was limited to synthetic data sets and only two algorithms. The experiment can be reproduced for a larger range of algorithms. This should include unsupervised domain adaptation methods other than DDA. Different data sets should be covered as well. For instance, concept shift was represented by a translation on the data, while any transformation that satisfies equation 6 is also concept shift. Furthermore, a major benefit of deep domain adaptation is allowing complex (convolutional) neural networks on high-dimensional data such as images. The data used in this research was much simpler and of low dimen-

sion, being less representative of realistic DDA use cases. Lastly, since none of the algorithms were designed for this approach, some modifications might produce much better results. It could also be interesting to consider the case where the global domain has domain labels. This is a realistic scenario when the global domain consists of data sets from multiple sources, which are easily labelled during collection. A modification to the DANN approach for multiple source domains [17] might be used.

6 Responsible Research

We will briefly reflect on the reproducibility of this research and other relevant ethical aspects. One important requirement is that the implementation of the DDA methods and the bias-inducing procedures are reproducible. While briefly explored, the option of including a lengthy formal description in pseudo-code was abandoned. This was because of the counteractive effect it had on readability. For the exact implementation, readers could always refer to the source code that is supplied with the text. In hindsight, however, this is not as accessible. A better alternative could have been to include an intuitive explanation in the main text, pseudo-code in the appendix, and source code separately.

Several decisions have been made throughout the research process that favoured simplicity. This includes using low-dimensional data compared to i.e. images, applying only translation for concept shift, or using DANN instead of more advanced state-of-the-art alternatives. This was not only out of necessity. Simpler data and algorithms allowed analysis and interpretability to an extent that would not have been possible with complex and time-consuming alternatives. We do acknowledge, as mentioned in section 5, that the lack of realistic tests diminishes the value of our conclusion.

Lastly, an obvious criticism of this research could be the rejection of the outcome of a significance test in section 4. As explained in that section, we do justify the reasoning for assuming that this was indeed a false negative. This practice is risky however and can be a slippery slope. Allowing yourself to use a justification for ignoring a statistical test result when it goes against your beliefs, is prone to confirmation bias. In hindsight, the disagreement was mostly caused by a difference in valuing false positives versus false negatives. The Bonferroni correction drastically lowered the significance threshold to ensure a low risk of false positives. A better option would have been to use a less conservative multiple-test correction method, that better reflected our values. Again, this can also be a slippery slope, if the significance threshold is changed until it confirms the expectations of the authors.

7 Code availability

Source code is available on GitHub: https://github.com/EWitting/global-domain-adapation.

References

[1] Karandeep Singh, John Donnelly, and Jeffrey McCullough. *IHPI News*. June 2021. URL: https://ihpi.umich.edu / news / popular - sepsis - prediction - tool - less - accurate-claimed.

[2] Andrew Wong et al. "External validation of a widely implemented proprietary sepsis prediction model in hospitalized patients". In: *JAMA Internal Medicine* 181 (8 Aug. 2021), pp. 1065–1070. ISSN: 21686114. DOI: 10.1001/jamainternmed.2021.2626.

[3] Antonio Torralba and Alexei A. Efros. "Unbiased look at dataset bias". In: *CVPR 2011*. 2011, pp. 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[4] Lei Tai, Giuseppe Paolo, and Ming Liu. "Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation". In: *CoRR* abs/1703.00420 (2017). arXiv: 1703 . 00420. URL: http://arxiv.org/abs/1703.00420.

[5] Gabriela Csurka. "A Comprehensive Survey on Domain Adaptation for Visual Applications". In: *Domain Adaptation in Computer Vision Applications*. Ed. by Gabriela Csurka. Cham: Springer International Publishing, 2017, pp. 1–35. ISBN: 978-3-319-58347-1. DOI: 10.1007/978-3-319-58347-1_1. URL: https://doi.org/10.1007/978-3-319-58347-1_1.

[6] Xiang Li et al. "Intelligent cross-machine fault diagnosis approach with deep auto-encoder and domain adaptation". In: *Neurocomputing* 383 (Mar. 2020), pp. 235–247. ISSN: 18728286. DOI: 10.1016/j.neucom.2019.12.033.

[7] Yuchen Zhang et al. "Bridging Theory and Algorithm for Domain Adaptation". In: *International Conference on Machine Learning*. Apr. 2019.

[8] Annegreet van Opbroek et al. "Transfer Learning Improves Supervised Image Segmentation Across Imaging Protocols". In: *IEEE Transactions on Medical Imaging* 34.5 (2015), pp. 1018–1030. DOI: 10.1109/TMI.2014.2366792.

[9] Wouter M. Kouw and Marco Loog. "A Review of Domain Adaptation without Target Labels". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.3 (2021), pp. 766–785. DOI: 10 . 1109 / TPAMI.2019.2945942.

[10] Palak, Harshita Mangotra, and Nidhi Goel. "Effect of selection bias on Automatic Colonoscopy Polyp Detection". In: *Biomedical Signal Processing and Control* 85 (Aug. 2023). ISSN: 17468108. DOI: 10.1016/j.bspc.2023.104915.

[11] Yaroslav Ganin et al. "Domain-adversarial training of neural networks". In: *Advances in Computer Vision and Pattern Recognition* (9783319583464 2017), pp. 189–209. ISSN: 21916594. DOI: 10.1007/978-3-319-58347-1_10/FIGURES/8. URL: https://link.springer.com/chapter/10.1007/978-3-319-58347-1_10.

[12] Salah Rifai et al. “Contractive Auto-Encoders: Explicit Invariance During Feature Extraction”. In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011* (Jan. 2011).

[13] Arthur Gretton et al. “A Kernel Two-Sample Test”. In: *Journal of Machine Learning Research* 13.25 (2012), pp. 723–773. URL: http://jmlr.org/papers/v13/gretton12a.html.

[14] Shai Ben-David et al. “Analysis of representations for domain adaptation”. In: *Advances in Neural Information Processing Systems* (2007), pp. 137–144. ISSN: 10495258. DOI: 10.7551/MITPRESS/7503.003.0022.

[15] Antoine de Mathelin et al. “ADAPT: Awesome Domain Adaptation Python Toolbox”. In: *arXiv preprint arXiv:2107.03049* (2021).

[16] Chen Lu et al. “Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification”. In: *Signal Processing* 130 (Jan. 2017), pp. 377–388. ISSN: 01651684. DOI: 10.1016/j.sigpro.2016.07.028.

[17] Anthony Sicilia, Xingchen Zhao, and Seong Jae Hwang. “Domain Adversarial Neural Networks for Domain Generalization: When It Works and How to Improve”. In: *Machine Learning* (Feb. 2021). ISSN: 15730565. DOI: 10.1007/s10994-023-06324-x. URL: http://arxiv.org/abs/2102.03924.

[18] Jose G. Moreno-Torres et al. “A unifying view on dataset shift in classification”. In: *Pattern Recognition* 45 (1 2012), pp. 521–530. ISSN: 00313203. DOI: 10.1016/j.patcog.2011.06.019.

[19] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] Tomaso Poggio et al. “Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review”. In: *International Journal of Automation and Computing* 14 (5 Oct. 2017), pp. 503–519. ISSN: 17518520. DOI: 10.1007/s11633-017-1054-2.

[21] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[22] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: (July 2019). URL: http://arxiv.org/abs/1907.10902.

[23] B. L. Welch. “The Generalization of ‘Student’s’ Problem when Several Different Population Variances are Involved”. In: *Biometrika* 34.1/2 (1947), pp. 28–35. ISSN: 00063444. URL: http://www.jstor.org/stable/2332510 (visited on 06/25/2023).

A Loss functions

Two loss functions are mentioned in this research: mean squared error and binary cross-entropy. Mean squared error computes the loss between two vectors and is equivalent to the square of their Euclidian distance:

$$\mathcal{L}_{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

Where y is a true value and $\hat{y}$ is a prediction, for a single n-dimensional sample. Binary cross-entropy is related to classification probabilities, and computed as follows:

$$\mathcal{L}_{BCE}(y, \hat{y}) = y \ln(\hat{y}) + (1 - y) \ln(1 - \hat{y}) \tag{8}$$

Where y is the binary label (exclusively 0 or 1) of a sample, and $\hat{y}$ is a prediction between 0 and 1.

B Supplementary Results

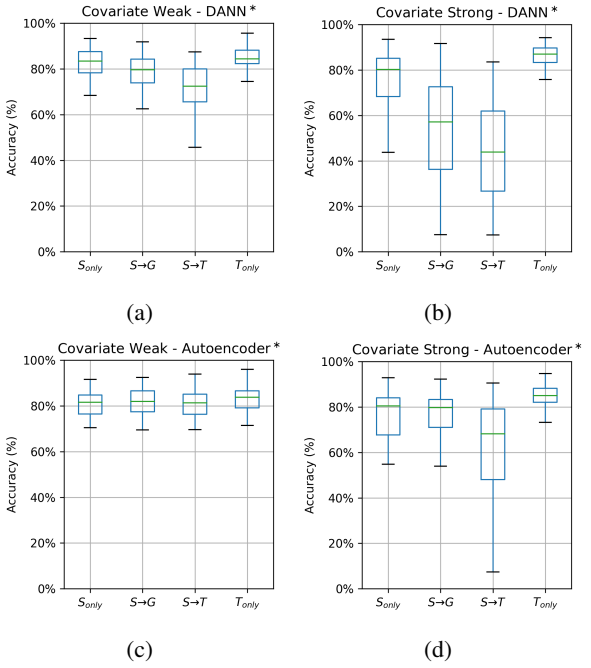

Figure 7: Accuracy on the target domain, after training DANN and the auto-encoder model with different configurations of unlabelled and labelled set. For this experiment, the adaptation parameters λ and w_{MMD} were set to those that hyperparameter optimization found for strong concept shift. S_{only} and T_{only} use only one labelled domain for supervised learning, source and target, respectively. For $S \rightarrow G$ and $S \rightarrow T$, the left-hand side, source, is labelled, and the right-hand side, global or target, is unlabelled. Results were measured for different types and amounts of bias, as indicated by their title. Each boxplot aggregates results across 50 datasets, outliers are not shown.

Shift type	Model	Config.	Median	Mean (stdev.)	P-value
Con. Weak	Autoencoder	S_{only}	0.639	0.655 (0.111)	
Con. Weak	Autoencoder	$S \rightarrow G$	0.647	0.666 (0.112)	0.6424
Con. Weak	Autoencoder	$S \rightarrow T$	0.725	0.705 (0.112)	0.0297
Con. Weak	Autoencoder	T_{only}	0.844	0.842 (0.034)	**0.0000**
Con. Weak	DANN	S_{only}	0.676	0.688 (0.117)	
Con. Weak	DANN	$S \rightarrow G$	0.761	0.749 (0.093)	0.0050
Con. Weak	DANN	$S \rightarrow T$	0.858	0.853 (0.035)	**0.0000**
Con. Weak	DANN	T_{only}	0.871	0.870 (0.027)	**0.0000**
Con. Strong	Autoencoder	S_{only}	0.549	0.603 (0.123)	
Con. Strong	Autoencoder	$S \rightarrow G$	0.613	0.617 (0.107)	0.5674
Con. Strong	Autoencoder	$S \rightarrow T$	0.742	0.708 (0.120)	**0.0000**
Con. Strong	Autoencoder	T_{only}	0.829	0.816 (0.061)	**0.0000**
Con. Strong	DANN	S_{only}	0.529	0.601 (0.133)	
Con. Strong	DANN	$S \rightarrow G$	0.717	0.697 (0.113)	**0.0002**
Con. Strong	DANN	$S \rightarrow T$	0.823	0.817 (0.055)	**0.0000**
Con. Strong	DANN	T_{only}	0.873	0.869 (0.032)	**0.0000**
Cov. Weak	Autoencoder	S_{only}	0.809	0.804 (0.063)	
Cov. Weak	Autoencoder	$S \rightarrow G$	0.814	0.813 (0.055)	0.4630
Cov. Weak	Autoencoder	$S \rightarrow T$	0.816	0.817 (0.055)	0.2821
Cov. Weak	Autoencoder	T_{only}	0.830	0.831 (0.042)	0.0140
Cov. Weak	Autoencoder*	S_{only}	0.816	0.809 (0.055)	
Cov. Weak	Autoencoder*	$S \rightarrow G$	0.819	0.814 (0.055)	0.6519
Cov. Weak	Autoencoder*	$S \rightarrow T$	0.813	0.807 (0.071)	0.9105
Cov. Weak	Autoencoder*	T_{only}	0.837	0.829 (0.056)	0.0734
Cov. Weak	DANN	S_{only}	0.831	0.823 (0.062)	
Cov. Weak	DANN	$S \rightarrow G$	0.831	0.824 (0.057)	0.9118
Cov. Weak	DANN	$S \rightarrow T$	0.835	0.819 (0.063)	0.7936
Cov. Weak	DANN	T_{only}	0.842	0.846 (0.046)	0.0316
Cov. Weak	DANN*	S_{only}	0.834	0.822 (0.063)	
Cov. Weak	DANN*	$S \rightarrow G$	0.796	0.787 (0.078)	0.0149
Cov. Weak	DANN*	$S \rightarrow T$	0.724	0.698 (0.144)	**0.0000**
Cov. Weak	DANN*	T_{only}	0.844	0.847 (0.045)	0.0235
Cov. Strong	Autoencoder	S_{only}	0.797	0.749 (0.161)	
Cov. Strong	Autoencoder	$S \rightarrow G$	0.774	0.730 (0.179)	0.5720
Cov. Strong	Autoencoder	$S \rightarrow T$	0.803	0.741 (0.174)	0.8056
Cov. Strong	Autoencoder	T_{only}	0.854	0.849 (0.059)	**0.0001**
Cov. Strong	Autoencoder*	S_{only}	0.805	0.727 (0.185)	
Cov. Strong	Autoencoder*	$S \rightarrow G$	0.798	0.735 (0.164)	0.8089
Cov. Strong	Autoencoder*	$S \rightarrow T$	0.682	0.608 (0.242)	0.0068
Cov. Strong	Autoencoder*	T_{only}	0.850	0.846 (0.055)	**0.0001**
Cov. Strong	DANN	S_{only}	0.816	0.802 (0.072)	
Cov. Strong	DANN	$S \rightarrow G$	0.801	0.786 (0.092)	0.3309
Cov. Strong	DANN	$S \rightarrow T$	0.811	0.773 (0.111)	0.1195
Cov. Strong	DANN	T_{only}	0.869	0.866 (0.050)	**0.0000**
Cov. Strong	DANN*	S_{only}	0.803	0.767 (0.108)	
Cov. Strong	DANN*	$S \rightarrow G$	0.571	0.547 (0.222)	**0.0000**
Cov. Strong	DANN*	$S \rightarrow T$	0.439	0.440 (0.216)	**0.0000**
Cov. Strong	DANN*	T_{only}	0.870	0.862 (0.052)	**0.0000**

Table 1: Numeric results corresponding to the accuries reported in figure 6 and 7. P-value corresponds to a two-sided Welch's t-test comparing to the S_{only} configuration and is in bold if it is below the significance threshold of 0.002. $*$ indicates that hyperparameters found for strong concept shift were used.

C Parameters

Shift type	Config.	λ
Concept Weak	S_{only}	**0**
Concept Weak	$S \to T$	1.687
Concept Weak	$S \to G$	2.171
Concept Weak	T_{only}	**0**
Concept Strong	S_{only}	**0**
Concept Strong	$S \to T$	4.216
Concept Strong	$S \to G$	5.219
Concept Strong	T_{only}	**0**
Covariate Weak	S_{only}	**0**
Covariate Weak	$S \to T$	0.021
Covariate Weak	$S \to G$	0.004
Covariate Weak	T_{only}	**0**
Covariate Strong	S_{only}	**0**
Covariate Strong	$S \to T$	0.011
Covariate Strong	$S \to G$	0.071
Covariate Strong	T_{only}	**0**

Table 2: Parameter values used for DANN. Found through hyperparameter optimization, unless manually set, indicated by bold text

Shift type	Config.	w_{aux}	w_{MMD}
Concept Weak	S_{only}	8.853	**0**
Concept Weak	$S \to T$	8.004	6.055
Concept Weak	$S \to G$	9.683	3.967
Concept Weak	T_{only}	8.853	**0**
Concept Strong	S_{only}	6.499	**0**
Concept Strong	$S \to T$	7.253	9.561
Concept Strong	$S \to G$	5.409	3.407
Concept Strong	T_{only}	6.499	**0**
Covariate Weak	S_{only}	2.771	**0**
Covariate Weak	$S \to T$	9.865	2.339
Covariate Weak	$S \to G$	8.717	8.512
Covariate Weak	T_{only}	2.771	**0**
Covariate Strong	S_{only}	9.699	**0**
Covariate Strong	$S \to T$	9.750	0.151
Covariate Strong	$S \to G$	9.993	0.723
Covariate Strong	T_{only}	9.699	**0**

Table 3: Parameter values used for the autoencoder. Found through hyperparameter optimization, unless manually set, indicated by bold text

Parameter	Value
Data sets per combination (optimization stage)	10
Data sets per combination (validation stage)	50
Epochs (optimization stage)	20
Epochs (validation stage)	30
Optimization trial count	15
Batch size	16

Table 4: remaining hyperparameter and configuration values that were shared across the two models.