Task Conditional Adversarial Domain Adaptation for Regression

Anonymous authors
Paper under double-blind review

Abstract

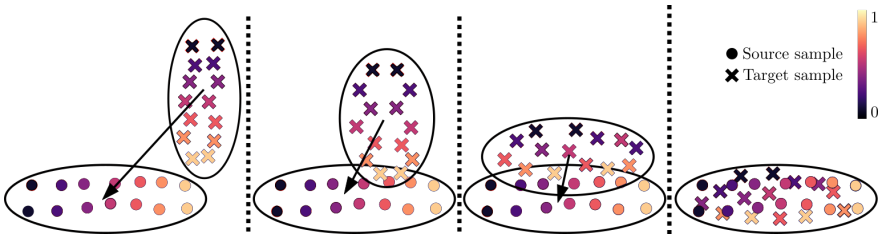
Accurate training of deep neural networks often requires large labeled datasets, which can be a significant limitation. Unsupervised domain adaptation (UDA) addresses this issue by transferring knowledge from a labeled source domain to a similar but distinct target domain for which only unlabeled samples are available. While UDA has been extensively studied for classification, it remains underexplored for regression tasks. In this work, we present a conditional adversarial adaptation strategy for regression that leverages a multilinear map between features and position-encoded regression predictions. The proposed position encoding keeps the discriminator input scale independent of the predicted regression value, enabling stable conditional alignment across the continuous output space. We show that conditioning the feature space on regressor predictions allows us to minimize the distance between domains in a way that also reduces the joint error. Furthermore, unlike in classification, we show that it is essential to incorporate the regressor into the minimax optimization of adversarial training to enhance the regressor's expressiveness for the target domain. We further introduce a target-focused batch-normalization layer to prevent a behavior mismatch between training and inference. We evaluate our approach through extensive experiments and ablation studies on four benchmark datasets: dSprites, MPI3D, Biwi Kinect, and our proposed Syn2Biwi adaptation benchmark. Our method outperforms state-of-the-art techniques overall, achieving an average mean prediction error reduction of up to 25%.

1 Introduction

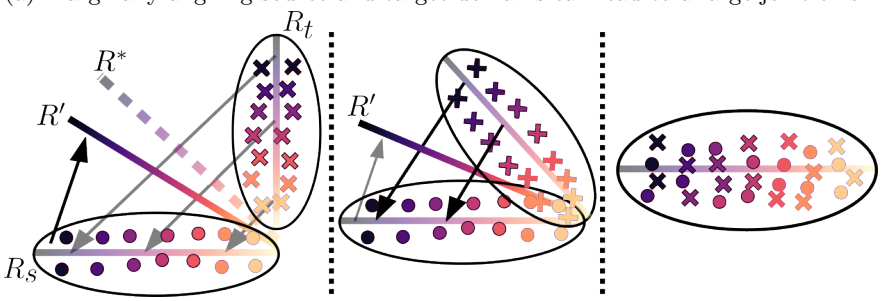
Deep learning has achieved remarkable success across many computer vision tasks. However, this success often relies on the assumption that training and test data are drawn from the same distribution. In practice, this assumption frequently breaks down: models encounter a domain shift or domain gap between training and test data, which can substantially degrade performance. Unsupervised domain adaptation (UDA) addresses this issue by transferring knowledge from a labeled source domain to an unlabeled target domain. Although UDA has been extensively studied for classification, it remains comparatively underexplored for regression.

A fundamental challenge in adapting regression models is the absence of clear decision boundaries. Many state-of-the-art UDA methods for classification rely on pseudo-labeling (Westfechtel et al., 2024; Prabhu et al., 2021), class anchors (Xie et al., 2018; Kang et al., 2019), and/or clustering techniques (Tang et al., 2020; Kang et al., 2019; Liang et al., 2021). These strategies are not directly applicable to regression because the output space is continuous. As a result, effectively aligning feature distributions between source and target domains is more difficult for regression tasks.

One of the most widely used approaches for UDA is adversarial adaptation, which performs minimax optimization between a feature extractor and a domain discriminator. The discriminator is trained to distinguish source from target samples, while the feature extractor is trained to deceive the discriminator and produce domain-invariant features. The basic adversarial domain adaptation method (Ganin et al., 2016) can be applied directly to regression, but in this setting the regression network and domain discriminator are trained



(a) Marginally aligning source and target domains can lead to a large joint error.



(b) Conditional alignment better preserves the relation between features and regression labels.

Figure 1: Motivation for conditional adversarial adaptation in regression. Markers indicate source and target samples, while color indicates the continuous regression label. Simply minimizing the distance between source and target distributions can result in a large joint error. Instead, aligning the conditional distribution ensures a more effective adaptation. Our method offers two key advantages: (1) The regressor weights are incorporated into the minimax optimization of adversarial training, thereby training a regressor R' closer to an optimal regressor R^* that minimizes the error for both source and target. (2) Conditioning the feature space on the regressor predictions aligns both feature spaces along the regressor, leading to a more precise adaptation and a minimized joint error.

independently. For classification, previous works (Long et al., 2018; Westfechtel et al., 2023) have shown that conditioning the domain discriminator on classifier outputs significantly improves adaptation accuracy. These methods, however, do not directly transfer to regression because regression outputs are continuous and lack clear class boundaries.

In this work, we present a conditional adversarial adaptation strategy for regression. Whereas conventional adversarial training aligns the marginal feature distributions of source $P(z^s)$ and target $Q(z^t)$, our method aligns the conditional distributions $P(z^s, y^s)$ and $Q(z^t, y^t)$. To condition the feature space on the output, our method uses a multilinear map between features and position-encoded (PE) regression predictions. By maintaining a consistent L1 norm across encoded regressor outputs, the PE ensures that all conditional distributions, including those for regressor values near zero, remain equally visible during adversarial alignment, thereby preventing conditioning-induced scale sensitivity.

The conditional adaptation addresses a key limitation of standard adversarial training: minimizing domain discrepancy alone can still lead to large joint errors (Fig. 1a). However, for this alignment to work efficiently, we employ regressor adversarial update (RAU), which updates the regressor's weights as part of the adversarial process. This differs from domain adaptation algorithms for classification, which train the classifier only with the classification loss. Because classification outputs are discrete while regression outputs are continuous, regression adaptation is much more sensitive to misalignment between source and target predictors, and therefore more prone to incorrect conditioning. For example, adaptation may fail if all samples are conditioned on the source regressor line R_s instead of the regressor R' obtained by RAU in Fig. 1b.

Finally, we introduce a target-focused batch-normalization layer (TF-BNL). This layer is motivated by the observation that source and target batches are forwarded separately during training, so batch normalization uses domain-specific batch statistics; at inference time, however, it uses running statistics contributed by both

domains. By updating the running statistics exclusively with target-domain data, we avoid a train–inference behavior mismatch in the BNL that artificially increases test error despite a well-aligned network.

We evaluate our method on four datasets: dSprites, MPI3D, Biwi Kinect, and Syn2Biwi, including the Syn2Biwi benchmark introduced in this work. We drastically outperform current state-of-the-art methods, achieving an average mean prediction error reduction of up to 25%. Furthermore, we show the contributions of our proposed PE, RAU, and TF-BNL in various ablation studies.

The key contributions of this paper are:

- We propose a conditional adversarial adaptation strategy for regression that leverages a multilinear map between features and position-encoded (PE) regression predictions.
- We introduce regressor adversarial update (RAU) as part of the adaptation and show that, unlike in classification tasks, it is essential for regression tasks.
- We design a target-focused batch-normalization layer that overcomes behavior mismatch between training and inference.
- We evaluate our method on four different datasets, outperforming current state-of-the-art methods. This includes a new evaluation task that combines two different head position datasets (rendered $\leftrightarrow$ real images), introducing a clearer simulated-to-real domain gap than the gender-based split in Biwi Kinect. We further show the effectiveness of our approach in several ablation studies.

2 Related works

Domain Adaptation for Classification: Unsupervised domain adaptation aims to overcome the domain gap between source and target datasets. Zhuang et al. (2020); Zhao et al. (2020) provide extensive surveys of domain adaptation. Existing approaches range from minimizing domain discrepancy statistics (Kang et al., 2019; Long et al., 2015; Tzeng et al., 2014), to adversarial methods (Ganin et al., 2016; Long et al., 2018; Westfechtel et al., 2023; Tzeng et al., 2017; Zhang et al., 2019), to self-training methods (Prabhu et al., 2021; Liang et al., 2020; Tan et al., 2020).

However, most recent classification methods cannot be directly employed for regression, since they rely on pseudo-labels (Westfechtel et al., 2023; 2024; Prabhu et al., 2021), class anchors (Xie et al., 2018; Kang et al., 2019), and/or clustering methods (Tang et al., 2020; Kang et al., 2019; Liang et al., 2021).

Our work follows the adversarial line of approaches. DANN (Ganin et al., 2016) uses a domain classifier and a gradient reversal layer to learn domain-invariant features, but trains the classifier and domain discriminator independently, which can lead to suboptimal solutions. BIWAA (Westfechtel et al., 2023) extended DANN by weighting features according to their importance to the classification task before feeding them into the discriminator. CDAN (Long et al., 2018) employed multilinear conditioning of the domain classifier on classifier predictions, making it the closest classification counterpart to our conditional adaptation strategy. However, neither of these methods can be directly applied to regression. In this work, we focus on adapting CDAN (Long et al., 2018) to the regression setting.

Domain Adaptation for Regression: While UDA has been extensively studied for classification, it remains relatively underexplored for regression. Mansour et al. (2009); Cortes & Mohri (2011); Zhao et al. (2022) discussed theoretical properties of the task. Most early methods either focused on importance weighting (Geng et al., 2007; Guo et al., 2008; Yamada et al., 2014; De Mathelin et al., 2021) or aimed to learn invariant representations (Cao et al., 2010; Pan et al., 2010; Courty et al., 2017; Nikzad-Langerodi et al., 2020). However, most of these methods address semi-supervised domain adaptation, requiring a small number of labeled target samples. More recently, several methods have been introduced for the unsupervised problem setting (Chen et al., 2021; Xia et al., 2022; Dhaini et al., 2023; Nejjar et al., 2023; Yang et al., 2024). Representation Subspace Distance (Chen et al., 2021) mitigates the domain gap through orthogonal bases of the representation space combined with bases mismatch penalization. Xia et al. (2022) used a bi-regressor architecture in which disagreement between regressors is maximized before a domain discriminator

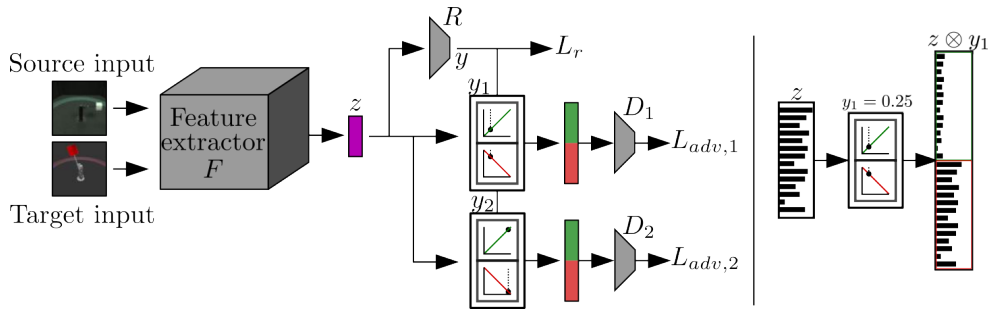


Figure 2: Pipeline of our method. We apply a multilinear map between features and position-encoded regression predictions. For each regression output y_j, a separate discriminator aligns the corresponding PE-conditioned feature distribution. The right side illustrates the encoding for one output, $z \otimes c(y_1)$; each bar represents a feature dimension, and its length indicates the absolute value.

is employed to learn domain-invariant features while the regressors are frozen. Dhaini et al. (2023) employs a dictionary learning approach to extract bases of the feature matrices of both domains and uses a projected gradient descent dictionary learning module to unify the dictionaries. Nejjar et al. (2023) proposed aligning the inverse Gram matrix of the features, based on its presence in the ordinary least squares solution and its ability to capture feature correlations. Conditional Domain Adaptation (COD) (Yang et al., 2024) extends DARE-GRAM by incorporating conditional alignment, matching first- and second-order statistics of the feature distributions. Similar to COD, our approach leverages conditional alignment. However, in contrast to computationally expensive techniques such as kernel matrix inversion or singular value decomposition, our method employs a simple multilinear mapping based on positional encoding of regressor outputs. Furthermore, unlike COD, we also adapt the regressor as part of our adaptation strategy. Our target-focused batch-normalization layer is related to domain-specific batch normalization (Chang et al., 2019); however, rather than using separate BN layers for each domain, we update the running statistics only with target samples.

3 Method

In unsupervised domain adaptation (UDA) for regression, the goal is to adapt a labeled source domain $\mathcal{D}_s = (x_i^s, y_i^s)_{i=1}^{n_s}$ with n_s samples to an unlabeled target domain $\mathcal{D}_t = (x_i^t)_{i=1}^{n_t}$ with n_t samples. The label vector y consists of J continuous variables, corresponding to the number of regression subtasks. Following previous works (Chen et al., 2021; Nejjar et al., 2023; Yang et al., 2024), we normalize label vectors to the range $[0, 1]$. Source and target samples are drawn from joint distributions $P(x^s, y^s)$ and $Q(x^t, y^t)$, respectively. Due to the domain gap, these distributions differ, i.e., $P \neq Q$. The goal is to reduce the distribution shift between domains and thereby reduce the target risk. In this work, we assume similar output ranges for source and target domains.

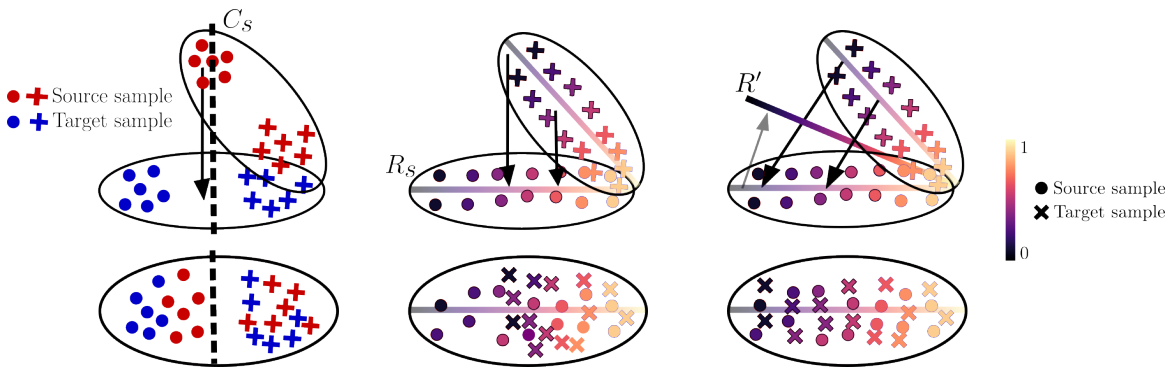
One of the most popular approaches to this problem is adversarial domain adaptation (DANN) (Ganin et al., 2016). Adversarial domain adaptation uses a minimax optimization between a feature extractor $F : X \to Z$ and a domain classifier $G : Z \to D$. The domain classifier is trained to distinguish source from target features, while the feature extractor is trained adversarially to make the two domains indistinguishable:

$$\max_F, \min_G L_{adv}(G_{DANN}(F(x_i)), d_i) \tag{1}$$

where d_i denotes the domain of sample x_i. Conditional Adversarial Domain Adaptation (CDAN) (Long et al., 2018) extends this objective by conditioning the discriminator input on the classifier output through a multilinear mapping:

$$\max_F, \min_G L_{adv}(G(F(x_i) \otimes y_i), d_i) \tag{2}$$

While DANN can be directly applied to regression, CDAN cannot. The reasons are twofold. First, directly conditioning on raw regression predictions makes the scale of the conditioned feature vector dependent on



(a) Classification (b) Regression (c) Regression with RAU

Figure 3: Effect of conditioning in classification and regression. In classification (a), conditional adversarial alignment can use the source classifier C_s because small predictor misalignments often do not change the discrete class assignment. In regression (b), conditioning on a source regressor R_s can propagate target-domain misalignment because the output space is continuous. With regressor adversarial updates (RAU) (c), conditioning uses a regressor R' that is better aligned to the target domain.

the predicted value. Small predicted values, especially near zero, suppress the feature signal passed to the discriminator, making conditional alignment scale-sensitive across the regression range. Second, the regressor is trained only on source data, which biases it toward the source distribution. While this issue is negligible for classification, it is detrimental for regression due to the continuous output space.

In this section, we first introduce our positional encoding (PE) to overcome scale sensitivity and regressor adversarial update (RAU) to overcome the bias toward the source distribution. Lastly, we present our target-focused batch-norm layer (TF-BNL).

3.1 Task conditioned adversarial domain adaptation

Following CDAN (Long et al., 2018), we apply a multilinear mapping between features and predictions, $z \otimes y$, to preserve dependencies between feature representations and regression predictions.

However, directly using the regressor output in the multilinear mapping suffers from scale sensitivity because the L1-norm of the output vector varies with the prediction. In classification, the softmax output has an L1-norm of 1, but regression outputs do not have this property. In the extreme case, an output vector of $y = \mathbf{0}$, which is a valid prediction, would produce a zero vector as input to the domain discriminator and make feature alignment infeasible. To mitigate this problem, we introduce a simple positional encoding that maintains a consistent L1 norm across outputs, keeping the discriminator input scale independent of the regressor output:

$$c(y_i) = [y_i, 1 - y_i] \tag{3}$$

Since the output y_i is normalized to $[0, 1]$, this encoding ensures that the L1-norm is independent of the regressor's prediction.

Thus, the adversarial objective becomes:

$$\max_F, \min_G L_{adv}(G(F(x_i) \otimes c(y_i)), d_i) \tag{4}$$

For conditional adversarial domain adaptation to be effective, the estimated regressor output y_i should be close to the ground truth $\hat{y}_i$. The regressor $y_i = R(z_i)$, with $z_i = F(x_i)$, is trained on source data and is therefore aligned with the source domain. However, as we show in our ablation studies, a regressor trained only on source data can be misaligned with the target domain (Fig. 3b). While a slight misalignment can be neglected in classification (i.e. the predicted class might be unchanged), regression is sensitive to such errors because the output is continuous. Conditioning features on erroneous predictions can propagate this

misalignment into the feature extractor throughout adversarial training. Our ablation studies show that such source-biased conditional alignment can even worsen the error compared to simple adversarial feature alignment (i.e., DANN).

To overcome this problem, we include the regressor R in the adversarial training objective as regressor adversarial update (RAU):

$$\max_{F,R}, \min_{G} L_{adv}(G(F(x_i) \otimes c(R(z_i))), d_i) \tag{5}$$

Here, the regressor weights are updated as part of the minimax optimization. In this way, the regressor is trained not only to minimize the source error but also to align source and target outputs, i.e., to align $P(y^s)$ with $Q(y^t)$. This requires a regressor R' that is expressive for both domains (Fig. 8d). In our ablation study, we show that RAU achieves a regressor that is aligned not only with source data but also with target data.

The overall loss for the feature extractor and regression network consists of regression loss L_r and adversarial loss L_{adv}:

$$L_{F,R} = L_r(x^s, y^s) - \alpha \cdot \lambda L_{adv}(G(z_i \otimes c(R(z_i))), d_i) \tag{6}$$

where α regularizes the two losses and λ is a warmup function. For the discriminator network, the following loss function is minimized:

$$L_G = \alpha \cdot L_{adv}(G(z_i \otimes c(R(z_i))), d_i) \tag{7}$$

To strengthen the domain discriminator, we align the marginal distributions of each subtask separately, matching source $P(z^s, y^s_j)$ and target $Q(z^t, y^t_j)$ for each j, instead of aligning the full joint distribution over all regression outputs. In practice, this means that we employ a separate domain discriminator for each subtask:

$$L_{adv} = \sum_{j=1}^{J} L_{adv,j}(G_j(z_i \otimes c(y_{i,j})), d_i) \tag{8}$$

The pseudo-code is shown in Alg. 1.

Input: Source and target datasets: $\mathcal{D}_s, \mathcal{D}_t$
for $n = 1; n \leq N$ **do**
  Get samples $x^s, \hat{y}^s, x^t$
  $z^s, z^t \leftarrow F(x)$
  $y^s, y^t \leftarrow R(z)$
  for $j = 1; j \leq J$ **do**
    `/* For each subtask j: */`
    `/* Positional encoding */`
    $c(y_j) = [y_j, 1 - y_j]$
    `/* Domain discriminator */`
    $L_{adv,j} = L_{CE}(G_j(z \otimes c(y_j), d))$
  end
  $L_{adv} = \alpha \sum_{j=1}^{J} L_{adv,j}$
  $L_r = L_{MSE}(y^s, \hat{y}^s)$
  Update F, R, G_j based on L_{adv}
  Update F, R based on L_r
end

Algorithm 1: Proposed algorithm

3.2 Target-focused batch-norm layer

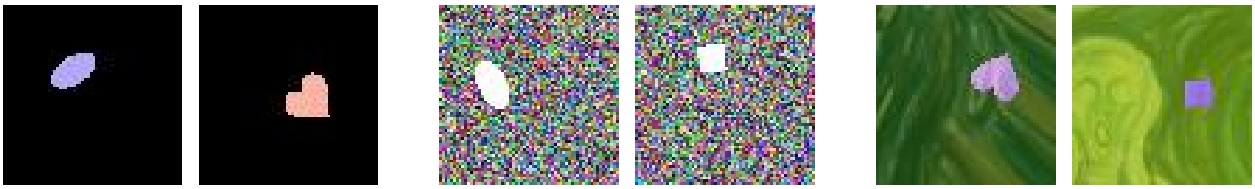
In most domain adaptation implementations, source and target data are forwarded separately through the backbone. Consequently, batch normalization layers compute and apply normalization using domain-specific batch statistics (mean and variance) during training, since each batch contains samples from only one domain. During inference, however, these layers rely on running statistics accumulated during training,

which aggregate information from both source and target domains. If the batch statistics of the source and target data differ significantly (e.g., between the noise and color domains in Fig. 4), this creates a behavior mismatch between training and inference and artificially increases the alignment error. To address this behavior mismatch, we introduce target-focused batch normalization, which updates the running statistics only with target data and performs better than approaches such as domain-specific batch normalization (Chang et al., 2019).

4 Experiments

We evaluate our proposed method on four benchmarks: dSprites, Biwi Kinect, Syn2Biwi, and MPI3D. We achieve strong improvements on three benchmarks and a marginal improvement on Biwi Kinect. We argue that the limited gain on Biwi Kinect is due to its gender-based split, where the domain gap is relatively small compared to the variation within each domain. We further conduct ablation studies to analyze the contributions of the individual components, their effect on the regressor, the effectiveness of the target-focused batch-norm layer, and the sensitivity to hyperparameters.

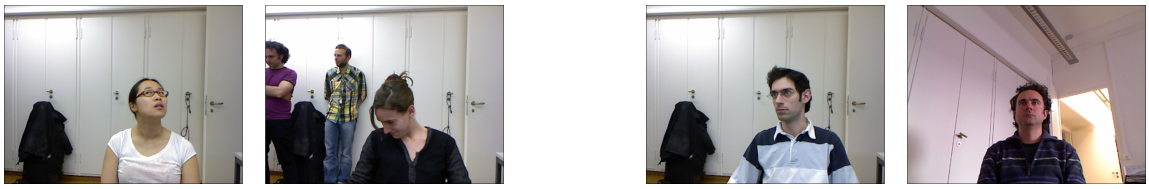
4.1 Setup



(a) Color (b) Noise (c) Scream

Figure 4: Sample images from the dSprites dataset.

dSprites (Matthey et al., 2017) is a synthetic 2D dataset with 737,280 images from three domains: **C**olor, **N**oise, and **S**cream. The images show objects of different sizes at different positions. The regression tasks are the x-position, y-position, and scale of the object. We evaluate the Mean Absolute Error (MAE) for all six possible transfer tasks.

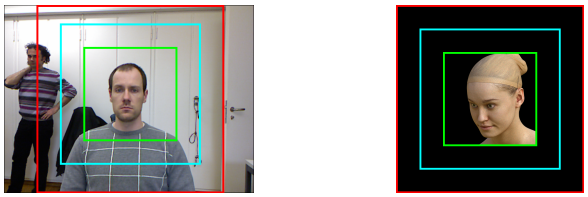


(a) Female (b) Male

Figure 5: Sample images from the Biwi Kinect dataset.

Biwi Kinect (Fanelli et al., 2013) is a real-world dataset of head poses recorded by a Kinect camera. Although depth data is provided alongside RGB data, we do not use it in this work. We split the dataset by gender into two domains: **F**emale, consisting of 5874 images of 6 women, and **M**ale, consisting of 9804 images of 14 men. The regression tasks are the yaw, pitch, and roll angles of the head pose. We evaluate MAE on both possible transfer tasks. We include this dataset because it was used in previous works; however, we believe that it is of limited value for domain adaptation, since the variance within each domain (e.g., lighting, background clutter, and presence or absence of people in the background) is arguably much larger than the domain gap between domains (i.e., male $\leftrightarrow$ female).

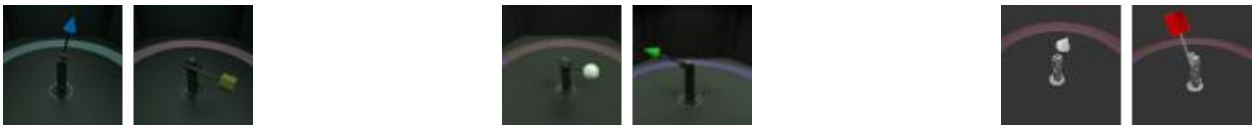
Syn2Biwi combines the Biwi Kinect and SynHead datasets (Gu et al., 2017). We introduce this adaptation benchmark to address the limitations of the Biwi Kinect gender split described above. The benchmark



(a) Biwi (b) SynHead

Figure 6: Examples from the Syn2Biwi benchmark. Colored outlines indicate the sizes used for the Easy (green), Medium (blue), and Hard (red) difficulty levels.

combines the real-world setting of Biwi Kinect with a simulated-to-real domain gap. Specifically, we use the entire Biwi Kinect dataset (both male and female) as one domain and the SynHead dataset as the other. The SynHead dataset contains 500K highly photorealistic rendered images of 10 real head models, each annotated with head-pose labels (yaw, pitch, roll). Early experiments revealed that similar head sizes between the datasets are critical for effective adaptation. To this end, we define three levels of difficulty: Easy, Medium, and Hard. For Biwi Kinect, we crop rectangular regions around the head with sizes of 240×240, 360×360, and 480×480 pixels, respectively. For SynHead, we add a black border of 100 pixels for the Medium task and 200 pixels for the Hard task. Fig. 6 shows examples of these preprocessing steps. We evaluate MAE in both transfer directions across all three difficulty levels.



(a) Real (b) Realistic (c) Toy

Figure 7: Sample images from the MPI3D dataset.

MPI3D (Gondal et al., 2019) is a simulation-to-real 2 DoF (degree of freedom) robotic arm dataset. It consists of three domains, **T**oy, Realistic (**RC**), and Real (**RL**), each containing 1,036,800 images. The regression tasks are the 2 DoF of the robotic arm, corresponding to rotations about the vertical and horizontal axes. We evaluate MAE for all six possible transfer tasks.

Implementation details: We follow the implementations of (Chen et al., 2021; Nejjar et al., 2023; Yang et al., 2024) and use a ResNet-18 backbone with our target-focused batch-norm layer (TF-BNL). For fair comparison, we also add TF-BNL to the main comparison methods (RSD, DARE-GRAM, COD) and report these results alongside the previously reported ones. Since we introduce the Biwi-SynHead benchmark, previous works did not report results on it; therefore, we report only results using TF-BNL for this benchmark. The regression network consists of a single linear layer followed by a sigmoid activation function, and labels are scaled to the range $[0, 1]$. Each domain discriminator consists of two linear layers with a hidden size of 2048 and a ReLU activation in between, followed by a sigmoid output. Since our discriminator uses a slightly different architecture than previous works, we investigate this design choice in the appendix; however, none of the main comparison methods employs a discriminator network. We use a gradient reversal layer with warmup function $\lambda = 2/(1 + e^{(-10 \cdot p)}) - 1$, where p denotes the iteration progress in $[0, 1]$. We set the regularization parameter between the regression and adversarial losses to $\alpha = 0.5$. We use SGD with a learning rate of 0.01, and train the regressor and domain discriminator layers with a ten times higher learning rate. Images are resized to 224×224 without data augmentation. Following previous work, we use a batch size of $b = 36$. While almost all previous work uses task-specific parameters, we use the same parameters for every dataset and adaptation task, omitting task-specific optimizations. We train each task on an NVIDIA A100 GPU; each training task takes slightly less than 30 minutes. For better comparison, we rerun the experiments of the three main comparison methods using target-focused batch normalization.

Table 1: Domain adaptation regression results (MAE) on dSprites (ResNet-18). Best results are displayed in bold and the runner-up results are underlined. [1] denotes results reported in previous works and [2] denotes our reimplemented version. We also report results with target-focused batch-normalization for the three main contenders as the second result after the '/'.

Method	C→N	C→S	N→C	N→S	S→C	S→N	Avg
ResNet-18 (He et al., 2016)	0.94	0.90	0.16	0.65	0.08	0.26	0.498
TCA (Pan et al., 2010)	0.94	0.87	0.19	0.66	0.10	0.23	0.498
MCD (Saito et al., 2018)	0.81	0.81	0.17	0.65	0.07	0.19	0.450
JDOT (Courty et al., 2017)	0.86	0.79	0.19	0.64	0.10	0.23	0.468
AFN (Xu et al., 2019)	1.00	0.96	0.16	0.62	0.08	0.32	0.523
DAN (Long et al., 2015)	0.70	0.77	0.12	0.50	<u>0.06</u>	0.11	0.377
DANN[1] (Ganin et al., 2016)	0.47	0.46	0.16	0.65	**0.05**	0.10	0.315
RSD (Chen et al., 2021)	0.31 / 0.15	0.31 / 0.29	0.12 / 0.10	0.53 / 0.30	0.07 / 0.10	<u>0.08</u> / 0.09	0.237 / 0.171
DARE-GRAM (Nejjar et al., 2023)	0.30 / 0.16	0.20 / <u>0.15</u>	0.11 / 0.09	0.25 / **0.17**	**0.05** / 0.08	**0.07** / <u>0.08</u>	0.164 / <u>0.123</u>
COD (Yang et al., 2024)	<u>0.12</u> / 0.23	0.16 / <u>0.15</u>	0.10 / 0.09	0.23 / **0.17**	0.08 / 0.07	0.12 / **0.07**	0.134 / 0.133
DANN[2] (Ganin et al., 2016)	0.17	0.28	<u>0.07</u>	0.33	0.07	**0.07**	0.166
Ours	**0.07**	**0.13**	**0.06**	<u>0.18</u>	<u>0.06</u>	**0.07**	**0.095**

Table 2: Domain adaptation regression results (MAE) on Biwi Kinect (ResNet-18). Best results are displayed in bold and the runner-up results are underlined. [1,2] denote results reported in previous works and reimplemented by us, respectively.

Method	F→M	M→F	Avg
ResNet-18 (He et al., 2016)	0.38	0.29	0.335
TCA (Pan et al., 2010)	0.39	0.31	0.350
MCD (Saito et al., 2018)	0.37	0.31	0.340
JDOT (Courty et al., 2017)	0.39	0.29	0.340
AFN (Xu et al., 2019)	0.41	0.32	0.365
DAN (Long et al., 2015)	0.37	0.28	0.325
DANN[1] (Ganin et al., 2016)	0.37	0.30	0.335
RSD (Chen et al., 2021)	0.30	0.26	0.280
DARE-GRAM (Nejjar et al., 2023)	0.29	0.23	0.260
COD (Yang et al., 2024)	0.21	0.20	0.205
DANN[2] (Ganin et al., 2016)	**0.16**	<u>0.18</u>	<u>0.170</u>
Ours	<u>0.17</u>	**0.17**	**0.169**

4.2 Results

Results for dSprites: Tab. 1 shows the results on dSprites. Our DANN implementation performs much better than previously reported DANN results and is almost on par with previous state-of-the-art methods. As shown in the appendix, this improvement is due to slight modifications of the discriminator network, namely removing dropout and increasing the hidden size. Adding TF-BNL decreases the error for all three main comparison methods; the decrease is substantial for RSD and DARE-GRAM, but only marginal for COD[1]. By introducing our task conditioning, we reduce the MAE by almost 40% over DANN. Our method significantly outperforms the other methods, reducing the average MAE by 0.028 ($\sim$ 25%) over the next best performing method.

Results for Biwi Kinect: Tab. 2 shows the results on Biwi Kinect. Again, our DANN implementation performs much better than previously reported DANN results, already outperforming all previous methods and almost halving the MAE of the reported DANN baseline. However, task conditioning improves the results only marginally. We believe that one major reason is that the variance within each domain (e.g.,

[1]This is at least partly because of reproducibility problems, despite using the official code. We speculate that the authors might accidentally have used batch statistics instead of running stats during inference, which would mostly explain the only marginal decrease of MAE.

Table 3: Domain adaptation regression results (MAE) on Syn2Biwi (ResNet-18). Best results are displayed in bold and the runner-up results are underlined. We report results for all three difficulty levels. We performed a parameter grid search for the reference methods (see appendix). For fair comparison, we used target-focused batch-normalization for all methods.

	Easy			Medium			Hard			All
Method	B→S	S→B	Avg	B→S	S→B	Avg	B→S	S→B	Avg	Avg
ResNet-18 (He et al., 2016)	0.18	0.29	0.231	0.24	0.35	0.295	0.29	0.38	0.336	0.287
RSD (Chen et al., 2021)	0.15	0.16	0.159	0.17	0.19	0.181	**0.19**	0.29	<u>0.239</u>	0.193
DARE-GRAM (Nejjar et al., 2023)	0.12	0.16	0.141	<u>0.13</u>	<u>0.18</u>	<u>0.153</u>	<u>0.22</u>	0.28	0.249	<u>0.181</u>
COD (Yang et al., 2024)	<u>0.11</u>	0.16	0.134	0.19	<u>0.18</u>	0.186	0.35	<u>0.27</u>	0.314	0.211
DANN[2] (Ganin et al., 2016)	<u>0.11</u>	<u>0.13</u>	<u>0.117</u>	0.14	0.19	0.163	0.23	0.33	0.280	0.187
Ours	**0.09**	**0.12**	**0.103**	**0.11**	**0.15**	**0.131**	**0.19**	**0.23**	**0.208**	**0.148**

Table 4: Domain adaptation regression results (MAE) on MPI3D (ResNet-18). Best results are displayed in bold and the runner-up results are underlined. [1,2] denote results reported in previous works and reimplemented by us, respectively. We also report results with target-focused batch-normalization for the three main contenders as the second result after the '/'.

Method	RL→RC	RL→T	RC→RL	RC→T	T→RL	T→RC	Avg
ResNet-18 (He et al., 2016)	0.17	0.44	0.19	0.45	0.51	0.50	0.377
TCA (Pan et al., 2010)	0.17	0.42	0.19	0.42	0.50	0.50	0.373
MCD (Saito et al., 2018)	0.13	0.40	0.15	0.45	0.52	0.50	0.358
JDOT (Courty et al., 2017)	0.16	0.41	0.16	0.41	0.47	0.47	0.353
AFN (Xu et al., 2019)	0.18	0.45	0.20	0.46	0.53	0.53	0.390
DAN (Long et al., 2015)	0.12	0.35	0.12	0.27	0.40	0.41	0.278
DANN[1] (Ganin et al., 2016)	0.09	0.24	0.11	0.41	0.48	0.37	0.283
RSD (Chen et al., 2021)	0.09 / 0.11	0.19 / <u>0.14</u>	<u>0.08</u> / 0.09	0.15 / <u>0.13</u>	0.36 / 0.37	0.36 / 0.29	0.205 / 0.186
DARE-GRAM (Nejjar et al., 2023)	0.09 / 0.09	0.15 / 0.16	0.10 / 0.13	0.14 / 0.14	<u>0.24</u> / <u>0.24</u>	0.24 / **0.21**	<u>0.160</u> / <u>0.160</u>
COD (Yang et al., 2024)	0.10 / <u>0.08</u>	**0.11** / 0.19	0.13 / 0.14	0.15 / 0.14	0.37 / **0.23**	0.29 / **0.21**	0.192 / 0.164
DANN[2] (Ganin et al., 2016)	**0.06**	0.20	0.09	0.22	0.40	0.30	0.212
Ours	**0.06**	0.16	**0.07**	**0.11**	0.28	<u>0.22</u>	**0.151**

background, lighting, and subject identity) is much larger than the domain gap between domains (female and male). The results on Syn2Biwi support this interpretation.

Results for Syn2Biwi: Tab. 3 shows the results on Syn2Biwi. Task conditioning consistently outperforms state-of-the-art methods across all difficulty levels, reducing the average MAE by 0.033 ($\sim 18\%$) over the next best performing method. Furthermore, as the difficulty increases, the improvement over DANN becomes more pronounced. This is in line with the results on Biwi Kinect, where the domain gap is rather marginal, and highlights the need for conditional alignment as the domain gap grows.

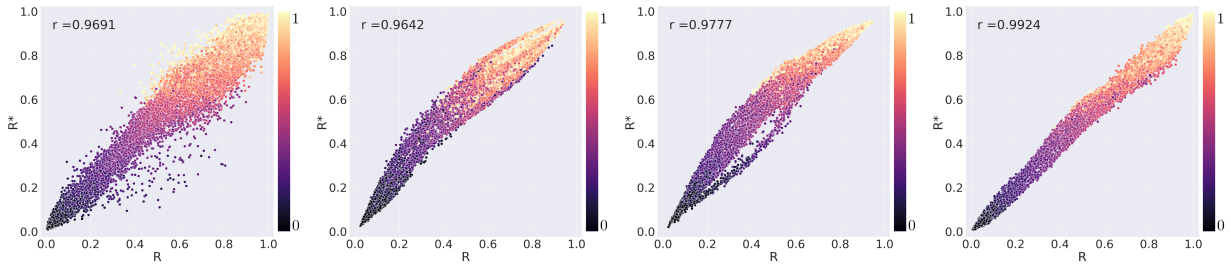
Results for MPI3D: Tab. 4 shows the results on MPI3D. While our DANN implementation significantly improves over previously reported DANN results, it still falls short of previous state-of-the-art methods. With task conditioning, our method significantly outperforms the other methods, achieving an average MAE that is lower by 0.009 than DARE-GRAM ($\sim 6\%$) and 0.013 than COD ($\sim 9\%$).

4.3 Ablation studies

Effect of different parts: Tab. 5 shows the contributions of positional encoding (PE), regressor adversarial update (RAU), and the subtask-specific discriminator (SSD). Without PE or RAU, the results do not improve significantly and often become worse. This highlights that both strategies are indispensable for conditional adaptation in regression. Using both PE and RAU improves over the DANN baseline in all cases, and adding SSD further improves the results.

Table 5: Ablation of positional encoding (PE), regressor adversarial update (RAU), and subtask-specific discriminator (SSD)

PE	RAU	SSD	dSprites	MPI3D	Syn2Biwi	Avg
			0.166	0.212	0.187	0.188
	✓	✓	0.132	0.243	<u>0.169</u>	0.181
✓		✓	<u>0.116</u>	0.259	0.227	0.201
✓	✓		0.125	<u>0.196</u>	0.178	<u>0.166</u>
✓	✓	✓	**0.095**	**0.151**	**0.148**	**0.131**



(a) DANN (b) COD (c) DARE-GRAM (d) Ours

Figure 8: Predictions of the trained regressor R compared to the oracle regressor R^*, which is trained to minimize the joint error. The x-axis shows predictions from R, the y-axis shows predictions from R^*, and color indicates the ground-truth label. Results are shown for the RC→T task of MPI3D after 10k training iterations. The discrepancy between R and R^* is much smaller for our method.

Importance of aligning the regressor: To further show the misalignment of the regressor for target data, we train an oracle linear regressor R^* using the regression loss of both source and target data. We then compare the predictions of the trained regressor R with those of the oracle regressor R^*. We compare our method with DANN (Ganin et al., 2016), COD (Yang et al., 2024), and DARE-GRAM (Nejjar et al., 2023). Fig. 8 shows a large discrepancy between the oracle and trained regressors for all comparison methods, highlighting regressor misalignment. Since our method updates the regressor as part of the minimax optimization (RAU), its regressor R is much closer to the oracle regressor R^*, effectively reducing the joint error and providing better target embeddings for conditional alignment. Fig. 9 shows the discrepancy between the two regressors during training. Our method reduces this discrepancy faster than the other methods, achieving better alignment for both domains even early in training.

Target-focused batch-norm layer: We investigate the contribution of the target-focused **(TF-BNL)** batch-norm layer using several baselines: forwarding source and target data separately while both update the running statistics (the **conv**entional implementation), forwarding source and target data as a **comb**ined batch, using domain-specific batch normalization **(DSBN)** (Chang et al., 2019), and using batch statistics

Table 6: Different strategies for BatchNormLayer

Method	dSprites	MPI3D	Biwi Kinect	Avg
conv	0.271	<u>0.159</u>	**0.165**	0.198
comb	1.165	0.634	0.168	0.656
DSBN	0.107	0.252	0.200	0.170
BSI-36	0.128	0.162	0.170	0.153
BSI-1k	<u>0.103</u>	0.160	<u>0.167</u>	<u>0.143</u>
TF-BNL	**0.095**	**0.151**	0.169	**0.138**

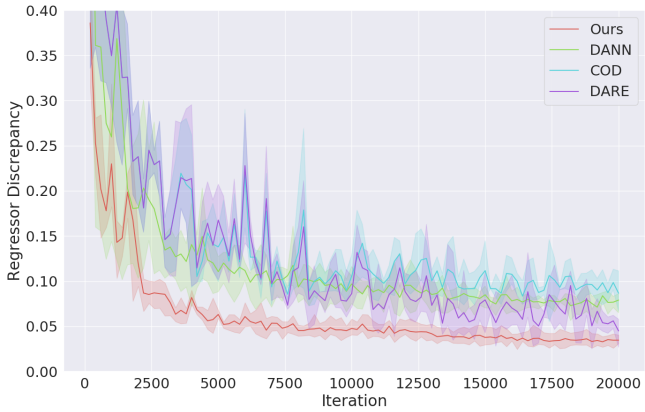


Figure 9: Discrepancy between the trained regressor R and the oracle regressor R^* during training on the RC→T task of MPI3D. The discrepancy is measured by the mean prediction distance; lower values indicate closer agreement.

during inference **BSI** instead of running statistics with batch sizes of 36 and 1000. The results are shown in Tab. 6. Conv works relatively well for MPI3D and Biwi Kinect, but has a greatly inflated error on dSprites despite using the same trained weights as the BSI and target-focused strategies. DSBN improves performance on dSprites, though not as much as target-focused batch normalization, but worsens the results on the other two datasets. BSI-1k performs close to the target-focused strategy but requires much more memory during inference.

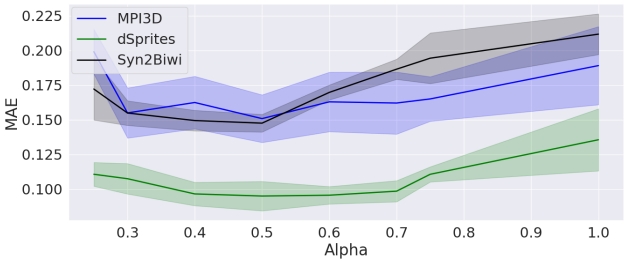


Figure 10: Effect of the regularization parameter α on MAE across datasets.

Parameter ablation: Fig. 10 shows the results for different values of α across datasets. The error is relatively flat around $\alpha = 0.5$, but increases for values close to 1 or much smaller values.

5 Discussion

In this work, we propose a conditional adversarial adaptation strategy for regression. Building on CDAN, we introduce a positional encoding (PE) of the regressor output and show that regressor adversarial updates (RAU) are needed as part of the minimax optimization. By adapting the feature space z and the regressor R jointly, we can find an aligned regressor network R' that more closely resembles an ideal regressor R^* for both domains. This enables more precise conditional alignment and reduces the joint error. Our experiments on dSprites, Biwi Kinect, Syn2Biwi, and MPI3D show strong improvements on three benchmarks and a marginal improvement on Biwi Kinect.

One limitation of our method is that the number of discriminator parameters grows linearly with the number of regression outputs ($\sim$ 4 million parameters per output). This could be addressed by sharing one discriminator across several outputs, although this introduces a trade-off between error and memory usage.

References

Bin Cao, Sinno Jialin Pan, Yu Zhang, Dit-Yan Yeung, and Qiang Yang. Adaptive transfer learning. In *AAAI*, volume 24, pp. 407–412, 2010.

Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *CVPR*, pp. 7354–7362, 2019.

Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation subspace distance for domain adaptation regression. pp. 1749–1759, 2021.

Corinna Cortes and Mehryar Mohri. Domain adaptation in regression. In *International Conference on Algorithmic Learning Theory*, pp. 308–323. Springer, 2011.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *NeurIPS*, 30, 2017.

Antoine De Mathelin, Guillaume Richard, François Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Adversarial weighting for domain adaptation in regression. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 49–56. IEEE, 2021.

Mohamad Dhaini, Maxime Berar, Paul Honeine, and Antonin Van Exem. Unsupervised domain adaptation for regression using dictionary learning. *Knowledge-Based Systems*, 267:110439, 2023.

Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random forests for real time 3d face analysis. *IJCV*, 101:437–458, 2013.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

Xin Geng, Zhi-Hua Zhou, and Kate Smith-Miles. Automatic age estimation based on facial aging patterns. *IEEE TPAMI*, 29(12):2234–2240, 2007.

Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. *NeurIPS*, 32, 2019.

Jinwei Gu, Xiaodong Yang, Shalini De Mello, and Jan Kautz. Dynamic facial analysis: From bayesian filtering to recurrent neural network. In *CVPR*, pp. 1548–1557, 2017.

Guodong Guo, Yun Fu, Charles R Dyer, and Thomas S Huang. Image-based human age estimation by manifold learning and locally adjusted robust regression. *IEEE TIP*, 17(7):1178–1188, 2008.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *CVPR*, pp. 4893–4902, 2019.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. pp. 6028–6039. PMLR, 2020.

Jian Liang, Dapeng Hu, and Jiashi Feng. Domain adaptation with auxiliary target domain-oriented classifier. In *CVPR*, pp. 16632–16642, 2021.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. pp. 97–105. PMLR, 2015.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *NeurIPS*, 31, 2018.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Ismail Nejjar, Qin Wang, and Olga Fink. Dare-gram: Unsupervised domain adaptation regression by aligning inverse gram matrices. In *CVPR*, pp. 11744–11754, 2023.

Ramin Nikzad-Langerodi, Werner Zellinger, Susanne Saminger-Platz, and Bernhard A Moser. Domain adaptation for regression under beer–lambert's law. *Knowledge-Based Systems*, 210:106447, 2020.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010.

Viraj Prabhu, Shivam Khare, Deeksha Kartik, and Judy Hoffman. Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In *CVPR*, pp. 8558–8567, 2021.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pp. 3723–3732, 2018.

Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: An empirical odyssey. In *ECCV*, pp. 585–602. Springer, 2020.

Hui Tang, Ke Chen, and Kui Jia. Unsupervised domain adaptation via structurally regularized deep clustering. In *CVPR*, pp. 8725–8735, 2020.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pp. 7167–7176, 2017.

Thomas Westfechtel, Hao-Wei Yeh, Qier Meng, Yusuke Mukuta, and Tatsuya Harada. Backprop induced feature weighting for adversarial domain adaptation with iterative label distribution alignment. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 392–401, 2023.

Thomas Westfechtel, Hao-Wei Yeh, Dexuan Zhang, and Tatsuya Harada. Gradual source domain expansion for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1946–1955, 2024.

Haifeng Xia, Pu Perry Wang, Toshiaki Koike-Akino, Ye Wang, Philip Orlik, and Zhengming Ding. Adversarial bi-regressor network for domain adaptive regression. *arXiv preprint arXiv:2209.09943*, 2022.

Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. pp. 5423–5432. PMLR, 2018.

Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*, pp. 1426–1435, 2019.

Makoto Yamada, Leonid Sigal, and Yi Chang. Domain adaptation for structured regression. *IJCV*, 109: 126–145, 2014.

Hao-Ran Yang, Chuan-Xian Ren, and You-Wei Luo. Cod: Learning conditional invariant representation for domain adaptation regression. In *ECCV*, pp. 108–125. Springer, 2024.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. pp. 7404–7413. PMLR, 2019.

Han Zhao, Chen Dan, Bryon Aragam, Tommi S Jaakkola, Geoffrey J Gordon, and Pradeep Ravikumar. Fundamental limits and tradeoffs in invariant representation learning. *Journal of machine learning research*, 23(340):1–49, 2022.

Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E Gonzalez, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, et al. A review of single-source deep unsupervised visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2): 473–493, 2020.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A Appendix

A.1 Discriminator network

In the paper, we report two results for DANN. One result is copied from previous papers, while the other shows the results that we achieved. This difference in results can be explained by the discriminator network. Since none of the previous papers stated the structure of their discriminator network or provided an implementation of it, we believe that the official implementation used for the classification datasets (Office31, OfficeHome, VisDA) was employed. The structure is listed under Reference in Tab. 8. In our experiments, we removed dropout, reduced the number of hidden layers, and increased the hidden size to 2048. The results for these changes are presented in Tab. 7. These results show that the changes to the discriminator are responsible for the different results. We did not adjust any other parameters.

Table 7: Results for different discriminator networks.

Method	dSprites	MPI3D
Reported	0.315	0.283
Reference	0.316	0.295
No dropout	0.272	0.251
1 hidden layer	0.191	0.245
2048 hidden size	0.166	0.212

A.2 Parameterization for Syn2Biwi

Since we propose the Syn2Biwi evaluation, we also had to evaluate existing methods in our new setting. To achieve a fair comparison, we performed a parameter grid search for each method on the Easy difficulty setting. We examined the official implementations of each method and used all parameterizations that were

Table 8: Discriminator network.

Reference	Remove dropout	Remove 1 hidden layer	Double hidden size
Linear (512 →1024)	Linear (512 →1024)	Linear (512 →1024)	Linear (512 →2048)
ReLU	ReLU	ReLU	ReLU
Dropout (0.5)	Linear (1024 →1024)	Linear (1024 →1)	Linear (2048 →1)
Linear (1024 →1024)	ReLU	Sigmoid	Sigmoid
ReLU	Linear (1024 →1)		
Dropout (0.5)	Sigmoid		
Linear (1024 →1)			
Sigmoid			

used for any task. We then used the best-performing parameters for the other two difficulty settings. We selected the parameters for the two tasks B→S and S→B independently. The parameters for the grid search were:

RSD:	
tradeoff	[0.001; 0.003]
tradeoff2	[0.1; 0.03; 0.01]
DARE-GRAM:	
tradeoff_angle	[0.1; 0.05; 0.01]
tradeoff_scale	[0.001; 0.00001]
threshold	[0.999; 0.95; 0.9]
COD:	
tradeoff_angle	[0.1; 0.05; 0.01]
tradeoff_scale	[0.001; 0.00001]
threshold	[0.999; 0.95; 0.9]
tradeoff_cod	[1e-2; 1e-3; 1e-4]