Unsupervised Domain Adaptation for Regression Using Dictionary Learning

Mohamad Dhaini, Maxime Berar, Paul Honeine, Antonin van Exem

Unsupervised Domain Adaptation for Regression Using Dictionary Learning

Mohamad Dhaini[a,b,*], Maxime Berar[a], Paul Honeine[a], Antonin Van Exem[b]

[a] *LITIS, University of Rouen Normandy, Rouen, France*
[b] *Tellux, Rouen, France*

Abstract

Unsupervised domain adaptation aims to generalize the knowledge learned on a labeled source domain across an unlabeled target domain. Most of existing unsupervised approaches are feature-based methods that seek to find domain invariant features. Despite their wide applications, these approaches proved to have some limitations especially in regression tasks. In this paper, we study the problem of unsupervised domain adaptation for regression tasks. We highlight the obstacles faced in regression compared to a classification task in terms of sensitivity to the scattering of data in feature space. We take this issue and propose a new unsupervised domain adaptation approach based on dictionary learning. We seek to learn a dictionary on source data and follow an optimal direction trajectory to minimize the residue of the reconstruction of the target data with the same dictionary. For stable training of a neural network, we provide a robust implementation of a projected gradient descent dictionary learning framework, which allows to have a backpropagation friendly end-to-end method. Experimental results show that the proposed method outperforms significantly most of state-of-the-art methods on several well-known benchmark datasets, especially when transferring knowledge from synthetic to real domains.

Keywords: Domain Adaptation, Transfer Learning, Regression, Deep Learning, Dictionary Learning, Sparse Coding

*Corresponding author:
Email addresses: mohamad.dhaini@tellux.fr (Mohamad Dhaini), maxime.berar@univ-rouen.fr (Maxime Berar), paul.honeine@univ-rouen.fr (Paul Honeine), antonin.vanexem@tellux.fr (Antonin Van Exem)

Preprint accepted in Knowledge-Based Systems

1. Introduction

Deep learning has achieved remarkable success in knowledge engineering, such as classification, regression and clustering. However, there still exist several problems that users encounter during implementation. First, deep learning models behavior depends on large-scale labeled datasets, which can be expensive and time consuming in real-world applications. Another main issue, as imposed by the assumptions underlying the learning theory, both training and test samples should be drawn from the same features space and from the same distribution. A solution to these two problems relies on using labeled data from a relevant domain (referred to as source domain) to learn statistical models that work well on the new domain (usually referred to as target domain), by undertaking an adaptation that minimizes the shift and bias between these two domains. This framework in machine learning is called Domain Adaptation.

Domain adaptation problems can be divided into two main groups based on the learning settings. When dealing with labeled target data in the learning process, we refer to the problem as supervised domain adaptation, while in the other case it is considered unsupervised. Here, we are interested in the latter, which is more difficult, due the lack of any label from the target domain in the alignment process of the two domains. In the same manner, unsupervised domain adaptation methods can be classified into four main classes [42]. The first gathers the self-labeling approaches that are based on guessing labels for target domains and then adjusting them during training [18]. For example, the authors of [5] introduced a pseudo-labeling curriculum inside a domain adaptation framework for semantic segmentation. After the first epoch, the pseudo-labels are generated for the target using a dynamic thresholding strategy with a linear decay; Then, starting from the second epoch, these labels are used to train the model. The second class groups cluster-based approaches that give the same label to instances belonging to the same dense regions such as in [41] and [1]. In [38], a discriminative clustering is used to classify the features extracted from

target data. The clustering objectives are based on entropy minimization for cluster separation, a soft-Fisher criterion for inter-cluster isolation and intra-cluster purity. In addition to the clustering objectives, the network is trained with a supervised learning objective using source labels. In the third class, the approaches are based on instance weighting, where matched instances between two domains are being re-weighted [22]. The last class consists of the feature representation learning approaches that seek to find a common space with invariant components between the two domains [34], [9] and [21]. In recent years, the machine learning community has been focusing on the last class of approaches that proved to be effective in the unsupervised problem. Existing approaches such as [35] make use of Maximum Mean Discrepancy to minimize the distance between source and target distributions. Other approaches, like [37], aim to align the second-order statistics of these distributions via a linear transformation. Adversarial training can also be used such as in [46], [13] and [31] with a domain discriminative feature module. In [19], the authors propose a network composed of three parts: a feature extractor, a domain classifier; and a domain discriminator. The network is trained through a minimax objective function that maximizes the adversarial loss and minimizes the classification loss. To align the two domains, the authors introduce a manifold alignment loss based on Grassmann distance between the orthogonal bases extracted via singular value decomposition.

The feature-based methods achieved good performance on classification tasks while having major limitations on regression tasks. To highlight this difference, we illustrate a domain adaptation problem for classification in Figure 1, and for regression in Figure 2. Figure 1 shows a scatter plot for two sets of data in a two-dimensional space, before (left figure) and after domain adaptation (right figure). Prior to any adaptation, the decision boundary learned on labeled source data cannot perfectly separate target data. Distribution matching techniques aim to transform both source and target data to a new feature space where each class of the target samples lies on the right side of the decision boundary. This means that the accuracy of the matching techniques is robust to the scattering

of the samples in the new feature space (U_1, U_2) unless they are in the correct classification zone. And here lies the difference with a regression task. Figure 2 shows an example of the domain adaptation process for regression. In the input space (X_1, X_2), a relation between the distribution of the source data and their labels (i.e., regression values) can be seen with color and level sets. However, this information is missing on the target samples. Therefore, to apply an accurate distribution matching, the source and target data are transformed into a new feature space where the two distributions match with the level sets applying to the unlabeled target data. This means that performance of the matching techniques is highly sensitive to the scattering of the samples in the feature space (U_1, U_2). Moreover, as illustrated with level sets, one can see the regression task as a continuum of boundaries, compared to a single boundary for a classification task.

These problems with domain adaptation for regression were also highlighted in [7]. The authors conducted an experiment where they show the robustness of a classification task with respect to the Frobenius norm of the input while, for a regression task, the error changes as a function of the norm. For that, the authors proposed a state-of-the-art method based on a deep neural network for the adaptation process. From the feature matrices extracted via a deep feature extractor (ResNet-18), the method aims to align the orthogonal bases of the matrices using a geometrical distance based on Grassmann manifold and principal angles calculated via a singular value decomposition (SVD). Although the effectiveness of this approach, there exist some setbacks. First, the SVD imposes an orthogonality constraint on the bases vectors. This additional constraint can cause a loss in information and interpretability of the data, which will later affect the alignment of the data. Besides, integrating an SVD module inside a neural network provides instability to the training [47]. This comes from the fact that power iteration's gradients of SVD depend on the singular values of the input data. In case of zero singular values or values close to each other, the gradients will explode.

The initial feature matrices extracted from each domain via a Siamese fea-

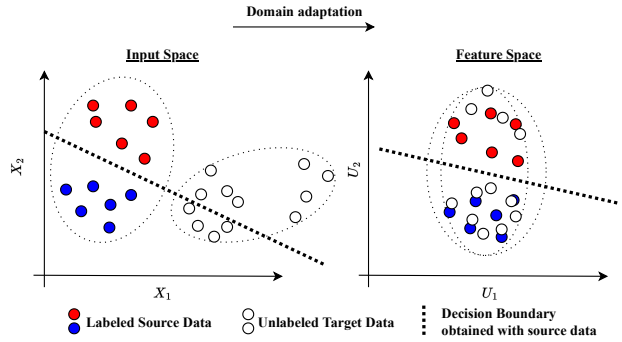


Figure 1: Domain Adaptation in Classification.

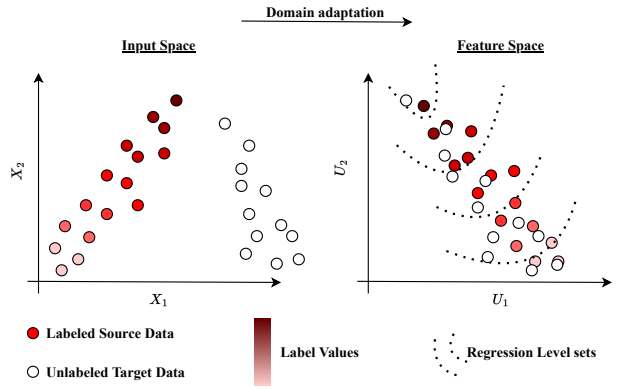


Figure 2: Domain Adaptation in Regression.

ture extractor allow to hold information related to the task at hand as well as the domain it belongs to. To obtain proper adaptation across different domains, the goal is to reduce the domain-specific features presented in these feature matrices while holding the task-specific ones. This can be achieved by extracting a shared domain-invariant subspace from both domains and using the projected features as an input for a regression module. Dictionary learning has proved its ability in extracting representative subspace of the input data, such as in hyperspectral unmixing problems where high dimensional data can be represented by a much lower dimension dictionary as well as a sparse coefficient matrix [10]. Moreover, several coupled dictionary learning approaches were introduced

to fuse two different domains together, such as in [17] where multispectral and hyperspectral images are fused together to perform collaborative clustering.

In this paper, we propose a novel domain adaptation technique based on dictionary learning for the alignment of the feature matrices. Starting from a constraintless dictionary learned on the source domain, we follow an optimal direction path that minimizes the reconstruction residue of target data with the same dictionary [11]. The contributions presented in this paper can be summarized as following:

1. We propose to extract the bases of the feature matrices of both domains using a dictionary learning approach. This offers a relaxation step to the process, where we seek to represent the features in the most complete way without any kind of information loss.
2. We integrate a projected gradient descent dictionary learning module inside a neural network. It is an iterative projection method for solving the sparse decomposition problem. It consists of first order gradient updates for dictionaries and a Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [4]. This process is backpropagation friendly with a stable behavior during training as it is compatible with PyTorch autograd[1], which provides the automatic gradient calculations that empowers neural network training. To our best knowledge, this the first time where this kind of module is integrated within a deep neural network.
3. We present a new domain alignment approach that seeks to unify the dictionaries between source and target feature matrices. We make use of optimal direction's updates [11] for dictionaries to follow a smooth trajectory that goes from a source dictionary to the target ones. For stable gradients, we make use of the Moore-Penrose pseudoinverse [3] to solve the ridge regression problem.

The rest of the paper is organized as follows. Section 2 highlights some of the

[1] https://pytorch.org/docs/stable/autograd.html

related work on unsupervised domain adaptation. In Section 3, we present the core ideas behind our proposed method. After that, the experimental studies with the obtained results are presented in Section 4. The contributions are summarized in Section 5, as well as future work.

2. Related Work

To address unsupervised domain adaptation, several works have been introduced [2, 23, 26, 29]. Most of the existing methods seek for finding domain invariant features for the source and target domains. These feature-based methods were integrated in both shallow and deep regimes. In [28], the shallow-regime Transfer Component Analysis (TCA) method is proposed to learn the transfer components across target and source domains in a Reproducing Kernel Hilbert Space (RKHS) and using the Maximum Mean Discrepancy (MMD); After obtaining a new subspace shared between different domains, a new classifier or regressor is trained using labeled source samples in order to use in the target domain. For deep-regime approaches, [6] introduces a marginalized denoising autoencoder to learn new domain invariant representations. For the Deep Adaptation Network (DAN) introduced in [25], hidden representations of all task-specific layers are embedded in an RKHS, where the mean embeddings of different domain distributions can be explicitly matched; The domain discrepancy is further reduced using an optimal multi-kernel selection method for mean embedding matching. In the same spirit, the Domain Adversarial Neural Network (DANN) [14] uses an adversarial loss to learn hidden features that are discriminative for the regression or classification task on the source domain, while indiscriminate with respect to the shift between the domains. More recently, the Maximum Classifier Discrepancy (MCD) [33] generates target features near the task-specific decision boundary to minimize domain discrepancy while maximizing the discrepancy between two classifier's output. A parameter-free adaptive feature norm approach is presented in [44] with the Adaptive Feature Norm (AFN) method, which progressively adapts the feature norms of the

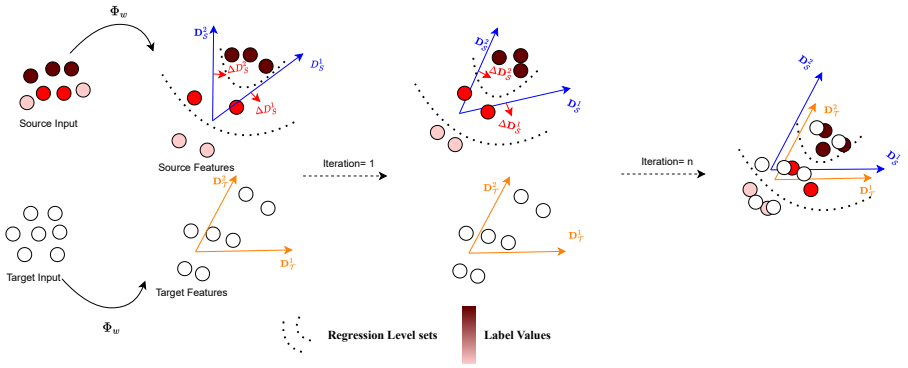


Figure 3: Methodology of the proposed method.

two domains into a larger one, thus having more transfer gains. However, the application of these methods is limited to the classification problem.

As defined in the previous section, the focus of this paper is on the regression problem. Most of the existing methods in this field are based on importance weighting such as [15, 45]. Such methods rely on at least few labeled target data, which makes them not applicable to the unsupervised problem. For feature-based methods, [27] searches for a low-dimensional subspace such that the projections of the source domain samples are informative with respect to the output variable and the projected domain input samples have a small covariance difference. The Joint Distribution Optimal Transport (JDOT) method [8] performs the mapping of a prediction function in a source domain into the target ones in a shallow regime, taking as an assumption that there exists a nonlinear transformation between the joint feature/label space distributions of both domains. For deep feature methods, the Representation Subspace Distance (RSD) method [7] uses a singular value decomposition (SVD) to extract orthogonal bases of the representation spaces. A geometrical distance over representation subspaces is defined within the Riemannian geometry of Grassmann manifold, and deep transferable representations are obtained by minimizing it. In [43], an Adversarial Bi-Regressor Network (ABRNet) is introduced to produce domain-invariant representations. Using a dual-regressor design, the model

detects target samples outside source distribution and serves as a discriminator to generate domain-invariant features. Moreover, to overcome the distribution shift across source and target domains, ABRNet seeks to eliminate the original cross-domain discrepancy by constructing source-similar and target-similar domains and align them with the discriminator.

3. Proposed Method

In this section, we describe the proposed methodology for domain adaptation via subspace mapping using dictionary learning, and derive the corresponding optimization algorithm. For better understanding, the optimization process is illustrated in Figure 3. We first present the underlying idea before describing in detail each part of the model.

Within the unsupervised scenario under study, we have labeled source input data and unlabeled target ones. Starting with these source and target data, we seek to find more representative features for each domain via a deep network denoted as Φ_w. Now the purpose is to align these features so that a regression module trained on the source data shall perform well on the target data. For this purpose, we propose to close the domain through aligning the bases of each domain feature's matrix extracted via a dictionary learning block.

We start the process by extracting the dictionary $\mathbf{D}_{\mathcal{S}}$ of the source features and then reconstruct the target features with the same dictionary. The aim is to unify the dictionaries for both feature sets; In other words, we aim to push $\mathbf{D}_{\mathcal{S}}$ towards $\mathbf{D}_{\mathcal{T}}$. Thus, after obtaining the reconstruction of target data, we follow the method of optimal directions [11] that seeks to obtain adjustment vectors for each atom of the dictionary $\Delta\mathbf{D}_{\mathcal{S}}$, and pushes the source dictionary toward the target one.

As the training is done in batches, we update the network parameters in a way that the calculated adjustment vectors become close to zero, which is equivalent to minimizing their Frobenius norm $\|\Delta\mathbf{D}_{\mathcal{S}}\|_F^2$. The advantage of such a subspace gap approach is the ability to work on all types of dictionaries

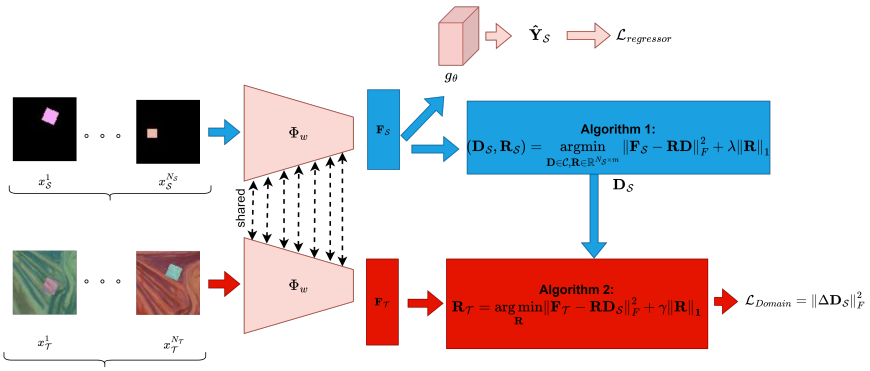


Figure 4: Architecture of the proposed deep model. The blue block applies Algorithm 1. The red block applies Algorithm 2. The overall process is given in Algorithm 3.

compared to existing approaches that are based on principal angles and work only with orthonormal bases. Besides, we are seeking to adapt a regression task on both domains; thus, we shall also add a regression loss while training, by taking advantage of the availability of source labels to train the network with a mean squared error loss.

Figure 4 shows the architecture of the proposed model for domain adaptation for regression. In the following, we present in detail each part of the deep neural model.

3.1. Feature Extraction

Let $\mathbb{R}^k$ denote the input space. The model takes as an input a batch of $N_\mathcal{S}$ samples from the source dataset and $N_\mathcal{T}$ samples from the target datasets, denoted respectively

$$\begin{cases} \mathbf{X}_\mathcal{S} = [x_\mathcal{S}^1, x_\mathcal{S}^2, \ldots, x_\mathcal{S}^{N_\mathcal{S}}]^\top \in \mathbb{R}^{N_\mathcal{S}\times k} \\ \mathbf{X}_\mathcal{T} = [x_\mathcal{T}^1, x_\mathcal{T}^2, \ldots, x_\mathcal{T}^{N_\mathcal{T}}]^\top \in \mathbb{R}^{N_\mathcal{T}\times k} \end{cases} \tag{1}$$

Let $y_\mathcal{S}^i$ denote the label of the i-th sample $x_\mathcal{S}^i$ of the source batch with $\mathbf{Y}_\mathcal{S} = [y_\mathcal{S}^1, y_\mathcal{S}^2, \ldots, y_\mathcal{S}^{N_\mathcal{S}}]^\top$.

The first part of the neural network consists of a feature extractor Φ_w, with w being the parameters of the feature extractor that maps the input from the

input space $\mathbb{R}^k$ to a feature space $\mathbb{R}^b$. The features extracted from the source batch $\mathbf{X}_\mathcal{S}$ and target batch $\mathbf{X}_\mathcal{T}$ are denoted respectively

$$\begin{cases} \mathbf{F}_\mathcal{S} = \Phi_w(\mathbf{X}_\mathcal{S}) \in \mathbb{R}^{N_\mathcal{S} \times b} \\ \mathbf{F}_\mathcal{T} = \Phi_w(\mathbf{X}_\mathcal{T}) \in \mathbb{R}^{N_\mathcal{T} \times b} \end{cases} \tag{2}$$

Let $f^i_\mathcal{S}$ and $f^i_\mathcal{T}$ be the two features obtained from samples $x^i_\mathcal{S}$ and $x^i_\mathcal{T}$, respectively.

Any feature extractor can be used in our model. We restrict ourselves to deep neural networks that are backpropagation friendly. Without loss of generality, ResNet-18 is used for this purpose since it has been largely investigated in the literature of domain adaptation; see Section 4 for more details.

3.2. Generation of Domain Dictionaries

In this section, we seek to learn a domain dictionary on the source features $\mathbf{F}_\mathcal{S}$. Dictionary learning aims to extract a sparse representation of the input by re-describing it as a linear combination of a few essential elements that form a dictionary. These atoms are considered more flexible than an orthogonal basis because they provide richer data representations. Moreover, the proposed dictionary learning algorithm (described in the following) is based on a gradient descent which provides a stable gradient calculations that allows to overcome the issue of instability that other approaches suffer inside a neural network, such as singular value decomposition.

The (sparse) dictionary learning problem seeks to find a dictionary $\mathbf{D}_\mathcal{S}$ of m atoms, namely $\mathbf{D}_\mathcal{S} = [d_1, ..., d_m]^\top \in \mathbb{R}^{m \times b}$, and a representation (called sparse codes) $\mathbf{R}_\mathcal{S} = [r_1, ..., r_{N_\mathcal{S}}]^\top \in \mathbb{R}^{N_\mathcal{S} \times m}$, according to the following optimization problem:

$$\begin{cases} (\mathbf{D}_\mathcal{S}, \mathbf{R}_\mathcal{S}) = \underset{\mathbf{D} \in \mathcal{C}, \mathbf{R} \in \mathbb{R}^{N_\mathcal{S} \times m}}{\operatorname{argmin}} \|\mathbf{F}_\mathcal{S} - \mathbf{R}\mathbf{D}\|_F^2 + \lambda \|\mathbf{R}\|_1 \\ \text{where } \mathcal{C} = \left\{\mathbf{D} \in \mathbb{R}^{m \times b} : \|d_i\|_2 \leq 1, \forall i = 1, \ldots, m\right\} \end{cases} \tag{3}$$

for some positive hyperparameter λ that controls the sparsity level and where $\mathcal{C}$ is a constraint that forces the columns of $\mathbf{D}$ to have a constrained ℓ_2-norm so

that we do not have arbitrary low values of r_i. A well-known strategy to solve this problem is alternating minimization, where the cost function is minimized over one variable while keeping the second fixed [24].

To find the optimal sparse codes for a given dictionary $\mathbf{D}$, several techniques improved to be efficient, such as orthogonal matching pursuit [40] and LASSO [39]. While the former is a greedy approach that has setbacks in high dimensional data. However, the LASSO solvers such as FISTA proved to be fast and efficient [4]. The FISTA optimization problem is defined as following:

$$\mathbf{R}_{\mathcal{S}} = \arg\min_{\mathbf{R}} \|\mathbf{F}_{\mathcal{S}} - \mathbf{R}\mathbf{D}\|_F^2 + \lambda \|\mathbf{R}\|_{\mathbf{1}} \tag{4}$$

Using the iterative shrinkage-thresholding algorithm (ISTA), we obtain the following iterative solution

$$\mathbf{R}^{t+1} = \mathcal{O}_{\lambda\mu}\left(\mathbf{R}^t - 2\mu\nabla_{\mathbf{R}}\left(\left\|\mathbf{F}_{\mathcal{S}} - \mathbf{R}^t\mathbf{D}\right\|_F^2\right)\right) \tag{5}$$

where μ is an appropriate stepsize, $\nabla.$ denotes the gradient operator, and $\mathcal{O}_{\lambda\mu}$ is the shrinkage operator defined by

$$\mathcal{O}_{\lambda\mu}(\mathbf{R}) = \max\{0, |\mathbf{R}| - \lambda\mu\}\,\mathrm{sgn}\,(\mathbf{R}) \tag{6}$$

with $\mathrm{sgn}(\cdot)$ being the sign function. FISTA uses Nesterov's Accelerated Gradient Descent [30] to solve the descent step in Equation 5.

With a fixed sparse code $\mathbf{R}$, the dictionary is updated. Several update techniques have been proposed in the literature, the most-known being the Method of Optimal Directions [11]. This method introduces the update step as $\mathbf{D} = \mathbf{R}^{+}\mathbf{F}$, where $\mathbf{R}^{+}$ denotes the Moore-Penrose pseudoinverse of $\mathbf{R}$, and then re-normalizes the columns of $\mathbf{D}$ to fit the constraints. However, the matrix-inversion becomes intractable for high dimensional data. To overcome this issue, we consider a stochastic gradient descent approach, where the update is done iteratively according to following equation:

$$\mathbf{D}^{t+1} = \mathrm{proj}_{\mathcal{C}}\left\{\mathbf{D}^t - \nabla_{\mathbf{D}}\left(\left\|\mathbf{F}_{\mathcal{S}} - \mathbf{R}\mathbf{D}^t\right\|_F^2\right)\right\} \tag{7}$$

where $\mathrm{proj}_{\mathcal{C}}$ is a projection operator into the constraint set $\mathcal{C}$ usually done via normalization of dictionaries. In this expression, the gradient operation

can be easily computed, yielding $\mathbf{R}^\top(\mathbf{F}_\mathcal{S} - \mathbf{R}\mathbf{D}^t)$. The resolution of the above optimization problem can be done in two steps, the update $\mathbf{D}^{t+1} = \mathbf{D}^t - \mathbf{R}^\top(\mathbf{F}_\mathcal{S} - \mathbf{R}\mathbf{D}^t)$ followed by a unit-norm normalization of its columns.

The pseudocode of the overall projected gradient descent algorithm for dictionary learning is presented in Algorithm 1. The advantages of the proposed method can be seen as following:

- The algorithm is based on the projected gradient descent technique, which offers stable and bounded gradients. This property is essential because this algorithm will be added to the training of a neural network, thus gradient explosions would cause computational and convergence troubles.
- The use of FISTA method is not affected with the curse of dimensionality.

Algorithm 1 Projected Gradient Descent Dictionary Learning.

Require: $\mathbf{F}_\mathcal{S} \in \mathbb{R}^{N_\mathcal{S} \times b}$, Rank m, Number of iterations $\mathbb{T}$, λ

Initialize $\mathbf{D}^0 \in \mathbb{R}^{m \times b}$ with random values from $\mathcal{N}(0, 1)$
for $t = 0$ to $\mathbb{T}$ **do**
  $\mathbf{R}^t = \textbf{FISTA}(\mathbf{F}_\mathcal{S}, \mathbf{D}^t, \lambda)$ {Update $\mathbf{R}^t$ via (5)}
  $\mathbf{D}^{t+1} = \mathbf{D}^t - \mathbf{R}^{t\top}(\mathbf{F}_\mathcal{S} - \mathbf{R}^t\mathbf{D}^t)$ {Update $\mathbf{D}^t$ via (7)}
  for $i = 1$ to m **do**
    $\mathbf{D}_i^{t+1} = \mathbf{D}_i^{t+1} / \left\|\mathbf{D}_i^{t+1}\right\|_2$ {Normalize $\mathbf{D}^t$}
  end for
end for
return $\mathbf{D}_\mathcal{S} = \mathbf{D}^{\mathbb{T}+1}$

3.3. Minimizing Subspace Gap Between Source and Target

Subspace Modeling is a common way of tackling the problem of domain adaptation. Having a shared subspace between both source and target data will allow us to close the domain gap. One approach can be done by extracting two dictionaries, one for each domain, and then deriving a mutual description between them. This would require two dictionary learning procedures, one for

the source and one for the target domain, which would be cumbersome. In the following, we provide a more efficient strategy, which takes one domain as a reference and then, by modifying the other domain, we create a trajectory to reach this reference.

Having the source dictionary $\mathbf{D}_{\mathcal{S}}$ obtained via the projected gradient descent dictionary learning, we first reconstruct the target features $\mathbf{F}_{\mathcal{T}}$ with $\mathbf{D}_{\mathcal{S}}$ via a sparse coding model as following:

$$\mathbf{R}_{\mathcal{T}} = \arg\min_{\mathbf{R}} \|\mathbf{F}_{\mathcal{T}} - \mathbf{R}\mathbf{D}_{\mathcal{S}}\|_F^2 + \gamma \|\mathbf{R}\|_{\mathbf{1}} \tag{8}$$

where $\mathbf{R}_{\mathcal{T}} = [r_1, \ldots, r_{N_{\mathcal{T}}}]^\top \in \mathbb{R}^{N_{\mathcal{T}} \times m}$ denotes the sparse coefficients matrix of the target features and γ is the sparsity level. The resolution of this optimization problem is done using the aforementioned FISTA algorithm. The residue of this reconstruction can be expressed as following:

$$\mathbf{J}_{res} = \mathbf{F}_{\mathcal{T}} - \mathbf{R}_{\mathcal{T}}\mathbf{D}_{\mathcal{S}} \tag{9}$$

Now $\mathbf{J}_{res}$ achieves its minimum whenever $\mathbf{D}_{\mathcal{S}}$ is as close as possible to $\mathbf{D}_{\mathcal{T}}$, since $\mathbf{D}_{\mathcal{T}} = \arg\min_{\mathbf{D}} \|\mathbf{F}_{\mathcal{T}} - \mathbf{R}_{\mathcal{T}}\mathbf{D}\|_F^2$ (it is worth noting that the dictionary of the target data needs not to be computed in our method). According to [11], the optimal adjustment $\Delta\mathbf{D}_{\mathcal{S}}$ (a path) that minimizes $\mathbf{J}_{res}$ can be obtained as follows:

$$\Delta\mathbf{D}_{\mathcal{S}} = \min_{\Delta\mathbf{D}} \|\mathbf{J}_{res} - \mathbf{R}_{\mathcal{T}}\Delta\mathbf{D}\|_F^2 \tag{10}$$

For a smooth adjustment at each step, a penalization can be applied on $\|\Delta\mathbf{D}_{\mathcal{S}}\|_F^2$:

$$\Delta\mathbf{D}_{\mathcal{S}} = \min_{\Delta\mathbf{D}} \|\mathbf{J}_{res} - \mathbf{R}_{\mathcal{T}}\Delta\mathbf{D}\|_F^2 + \alpha \|\Delta\mathbf{D}\|_F^2 \tag{11}$$

where α is a trade-off parameter. This can be seen as a ridge regression problem. Setting the first order derivatives to zero, the solution becomes:

$$\Delta\mathbf{D}_{\mathcal{S}} = \left(\mathbf{R}_{\mathcal{T}}^\top\mathbf{R}_{\mathcal{T}} + \alpha\mathbf{I}\right)^{-1}\mathbf{R}_{\mathcal{T}}^\top\mathbf{J}_{res} \tag{12}$$

where $\mathbf{I}$ is the identity matrix.

Algorithm 2 shows the steps of the domain gap estimation algorithm that computes the subspace gap between source and target bases. Algorithm 2 offers the same kind of stability as that of Algorithm 1 with a controlled smooth adjustment criterion to the source dictionaries.

Algorithm 2 Domain Gap Estimation Module.

Require: $\mathbf{F}_{\mathcal{T}} \in \mathbb{R}^{N_{\mathcal{S}} \times b}$, $\mathbf{D}_{\mathcal{S}} \in \mathbb{R}^{m \times b}$, γ, α

1: $\mathbf{R}_{\mathcal{T}} = \arg\min_{\mathbf{R}} \|\mathbf{F}_{\mathcal{T}} - \mathbf{R}\mathbf{D}_{\mathcal{S}}\|_F^2 + \gamma \|\mathbf{R}\|_{\mathbf{1}}$ {Calculate using (5)}

2: $\mathbf{J}_{res} = \|\mathbf{F}_{\mathcal{T}} - \mathbf{R}_{\mathcal{T}}\mathbf{D}_{\mathcal{S}}\|_F^2$

3: $\Delta\mathbf{D}_{\mathcal{S}} = \left(\mathbf{R}_{\mathcal{T}}^{\top}\mathbf{R}_{\mathcal{T}} + \alpha\mathbf{I}\right)^{-1} \mathbf{R}_{\mathcal{T}}^{\top}\mathbf{J}_{res}$ {Adjustments of source dictionary}

4: **return** $\Delta\mathbf{D}_{\mathcal{S}}$

3.4. Domain Adaptation in Deep Neural Networks

The architecture of the proposed method, as illustrated in **??**, shows the two investigated objective functions that will be backpropagated within the deep neural network for the domain adaptation for regression: the domain loss and the regression loss. The overall loss is composed of these two losses, namely it takes the form

$$\mathcal{L}_{total} = \mathcal{L}_{regressor} + \beta\, \mathcal{L}_{Domain} \tag{13}$$

where β controls the trade-off between the two losses. In the following, we describe in detail these two losses.

For the regression part, the features $\mathbf{F}_{\mathcal{S}}$ and $\mathbf{F}_{\mathcal{T}}$ are used as an input for a regression network g_θ that gives the label prediction as an output. To train this network, only the source data are used, since target data are unlabeled. The regression loss is defined as the mean squared error between true source labels and the predicted ones, for each source batch, namely

$$\mathcal{L}_{regressor} = \frac{1}{N_{\mathcal{S}}} \sum_{i=1}^{N_{\mathcal{S}}} \left\| y_{\mathcal{S}}^i - g_\theta\left(f_{\mathcal{S}}^i\right) \right\|^2 \tag{14}$$

where $g_\theta(f_{\mathcal{S}}^i)$ is the predicted label of sample $x_{\mathcal{S}}^i$ of the source batch.

Besides, the source features $\mathbf{F}_{\mathcal{S}}$ are also passed to a dictionary learning module in order to learn the corresponding dictionary $\mathbf{D}_{\mathcal{S}}$. Then $\mathbf{D}_{\mathcal{S}}$ is passed into

the sparse coding module, as well as the target features $\mathbf{F}_{\mathcal{T}}$ in order to estimate the gap $\Delta\mathbf{D}_{\mathcal{S}}$, as given in Algorithm 2. When both domains become more and more aligned, $\|\Delta\mathbf{D}_{\mathcal{S}}\|_F^2$ should converge to zero, meaning no further adjustment for source dictionaries is necessary to match the target data. Therefore, the domain loss is defined as following:

$$\mathcal{L}_{Domain} = \|\Delta\mathbf{D}_{\mathcal{S}}\|_F^2 \tag{15}$$

During backpropagation of $\mathcal{L}_{Domain}$, the parameters of the feature extractor ϕ_w are updated in order to minimize the gap between both domains. Algorithm 3 shows the recap of the proposed method for domain adaptation using dictionary learning.

Algorithm 3 Domain Adaptation Using Dictionary Learning.

Require: $\mathbf{X}_{\mathcal{S}} \in \mathbb{R}^{N_{\mathcal{S}} \times k}$, $\mathbf{X}_{\mathcal{T}} \in \mathbb{R}^{N_{\mathcal{T}} \times k}$, $\lambda, \gamma, \alpha, \beta$

for each epoch **do**
  for each batch **do**
    $\mathbf{F}_{\mathcal{S}} = \Phi_w(\mathbf{X}_{\mathcal{S}})$
    $\mathbf{F}_{\mathcal{T}} = \Phi_w(\mathbf{X}_{\mathcal{T}})$
    $\mathbf{D}_{\mathcal{S}}$ = Algorithm 1 ($\mathbf{F}_{\mathcal{S}}$, m,$\mathbb{T}$, λ)
    $\Delta\mathbf{D}_{\mathcal{S}}$ = Algorithm 2 ($\mathbf{F}_{\mathcal{T}}$, $\mathbf{D}_{\mathcal{S}}$, γ, α)
    $\mathcal{L}_{total} = \mathcal{L}_{regressor} + \beta\ \mathcal{L}_{Domain}$ {Calculate using (14) and (15)}
    Backward step {Calculate gradients using PyTorch autograd}
    Update Neural Network {Update using SGD Optimizer}
  end for
end for

3.5. Model Interpretability

For a regression module to work perfectly on both domains, the input to this module should contain the task-specific features only without any domain-specific ones. The purpose of the proposed method is to eliminate domain-specific information from $\mathbf{F}_{\mathcal{S}}$ and $\mathbf{F}_{\mathcal{T}}$ that are later on passed to the regression module g_θ to perform task prediction.

In order to understand the underlying mechanism of the proposed method using dictionary learning, we analyze the model when convergence is reached. As previously stated, the proposed domain loss consists in minimizing $\|\Delta \mathbf{D}_{\mathcal{S}}\|_F^2$. Therefore, the following properties are achieved at convergence:

- $\|\Delta \mathbf{D}_{\mathcal{S}}\|_F^2 \to 0$
- $\mathbf{D}_{\mathcal{S}} \approx \mathbf{D}_{\mathcal{T}}$
- $\mathbf{F}_{\mathcal{T}} \approx \mathbf{R}_{\mathcal{T}} \mathbf{D}_{\mathcal{S}}$

This means that both feature matrices $\mathbf{F}_{\mathcal{S}}$ and $\mathbf{F}_{\mathcal{T}}$ share the same basis vectors. In other words, the process is able to find a domain-invariant subspace common between both domains and project these feature matrices into it, thus resulting in the compact representations $\mathbf{R}_{\mathcal{S}}$ and $\mathbf{R}_{\mathcal{T}}$. These representations serve as the inputs to a shared module g'_θ for regression, which is composed of the two parts. The first one is a linear layer with $\mathbf{D}_{\mathcal{S}}$ as its weights, and the second one is the regression module g_θ. The architecture of the method can be seen as in Figure 5.

To highlight the significance of the extracted dictionaries, one can make an analogy with the unmixing scheme presented in [10]. We can see that in the case of hyperspectral data as input for our model, $\mathbf{R}$ represents the abundances (output of the encoder part) while $\mathbf{D}$ represents the endmembers (decoder part). Therefore, for domain adaptation in hyperspectral data, it is sufficient to follow the optimal path presented in this paper to adapt the endmembers from two different domains.

4. Experiments

In this section, we evaluate the performance of the proposed method and compare it with state-of-the-art techniques. Three benchmark datasets are used for this purpose: dSprites, MPI3D and Biwi Kinect. Regression performance, representation transferability and time complexity are analyzed.

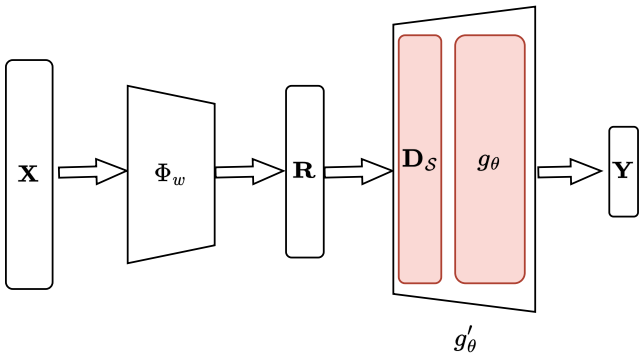


Figure 5: Interpretation at convergence of the proposed method based on dictionary learning.

Figure 6: Examples of dSprites Images.

4.1. Datasets

4.1.1. dSprites

dSprites[2] is a 2D synthetic dataset used for domain adaptation. It consists of three domains each with 737,280 images. The three domains are Color (**C**), Noisy (**N**) and Scream (**S**). An example is shown in Figure 6 and the labels of each image are shown in Table 1. For the experiments, following the same settings as in [7], we predict three regression parameters: scale, Position X and Position Y. The model will be then evaluated on six transfer tasks: $\mathbf{C} \rightarrow \mathbf{N}, \mathbf{C} \rightarrow \mathbf{S}, \mathbf{N} \rightarrow \mathbf{C}, \mathbf{N} \rightarrow \mathbf{S}, \mathbf{S} \rightarrow \mathbf{C}$, and $\mathbf{S} \rightarrow \mathbf{N}$

[2] https://github.com/deepmind/dsprites-dataset

Table 1: Parameters of dSprites Dataset

Factor	Parameters	Task
Shape	square, ellipse, heart	recognition
Scale	$\in [0.5, 1]$	regression
Orientation	$\in [0, 2\pi]$	regression
Position X	$\in [0, 1]$	regression
Position Y	$\in [0, 1]$	regression

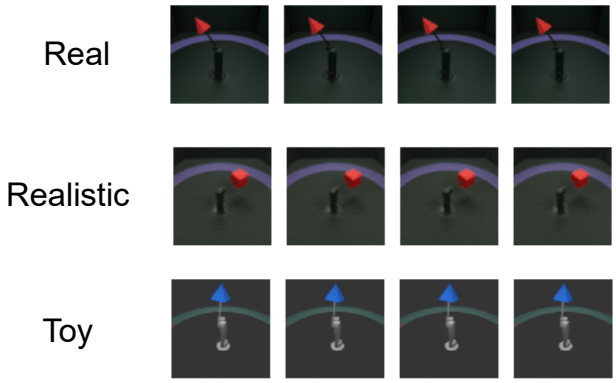


Figure 7: Examples of MPI3D Images.

4.1.2. MPI3D

MPI3D[3] is a simulation-to-real dataset with 3D objects. It consists of three domains: Toy (**T**), Realistic (**RC**) and Real (**RL**), each of 1,036,800 images. Examples of the images are shown in Figure 7. The labels are reported in Table 2 with two regression tasks, which are the rotations over the horizontal and vertical axis. The model will be then evaluated on six transfer tasks: **RL** $\rightarrow$ **RC**, **RL** $\rightarrow$ **T**, **RC** $\rightarrow$ **RL**, **RC** $\rightarrow$ **T**, **T** $\rightarrow$ **RL**, and **T** $\rightarrow$ **RC**.

[3] https://github.com/rr-learning/disentanglement_dataset

Table 2: Parameters of MPI3D Dataset

Factor	Parameters	Task
Object Color	5 values	recognition
Object Shape	6 values	recognition
Object Size	2 values	recognition
Camera Height	3 values	recognition
Background Color	3 values	recognition
Horizontal Axis	40 values $\in [0, 1]$	regression
Vertical Axis	40 values $\in [0, 1]$	regression

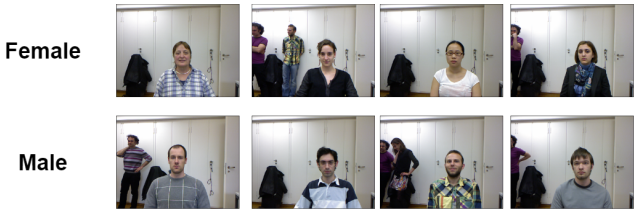


Figure 8: Examples of Biwi kinect Images.

4.1.3. Biwi Kinect

Biwi Kinect[4] is a challenging real world dataset used for head pose estimation. It consists of 5874 female images ($\mathbf{F}$) and 9804 male ones ($\mathbf{M}$). Each image is labeled with 3 regression factors, pitch $\in [-92.044, 231.352]$, Yaw $\in [-87.7066, 246.684]$ and Roll $\in [754.182, 1297.45]$. Examples of the Biwi images can be seen in Figure 8. For domain adaptation, the dataset is divided into two domains based on gender. Thus, this results in two transfer tasks: $\mathbf{F} \rightarrow \mathbf{M}$ and $\mathbf{M} \rightarrow \mathbf{F}$.

[4]https://www.kaggle.com/datasets/kmader/biwi-kinect-head-pose-database

4.2. Implementation

The model was implemented using PyTorch and trained with Google Colab P100 GPU. We used a ResNet-18 [20] pretrained on the ImageNet [32] dataset. We then train our network on the three datasets mentioned above to predict all regression factors in each dataset. For each dataset, every label is normalized to [0,1] to eliminate the diversity in scales. For input images, we resize them to 224×224 and we normalize them to meet the ImageNet images for the ResNet-18 to work perfectly. Therefore, we subtract the mean and divide by the standard deviation of each channel of ImageNet images according to $x_{\text{normalized}} = (x - \mu)/\sigma$, where x is the input image, $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$ are the mean and the standard deviation of the images in ImageNet dataset. The regressor g_θ is trained from scratch, so its learning rate is set to be 10 times greater than that of Φ_w. The model was trained using mini-batch stochastic gradient descent (SGD) with a learning rate of 0.1. To keep it fast and efficient, a small batch size of 36 was selected. We compare it to several state-of-the-art techniques for unsupervised domain adaptation. All these methods are reported in Table 3. For each method, we run the process 5 times and then we give the mean and standard deviations of these trials.

4.3. Hyperparameter Tuning

The proposed framework consists of three main hyperparameters: the sparsity levels λ and γ, and the smoothing penalization parameter α. Now to get the most out of the proposed method, a proper hyperparameter tuning should be performed. As we are working in a domain adaptation problem, training and testing data follow different distributions while the conditional probability of output given the input $P(Y|X)$ is unchanged. This is referred to as covariate shift where traditional model selection methods that are based on cross-validation of training data are not valid anymore.

To address this issue, methods based on the importance weighted cross-validation (IWCV) can be used [36, 12]. The IWCV gives an unbiased estimate

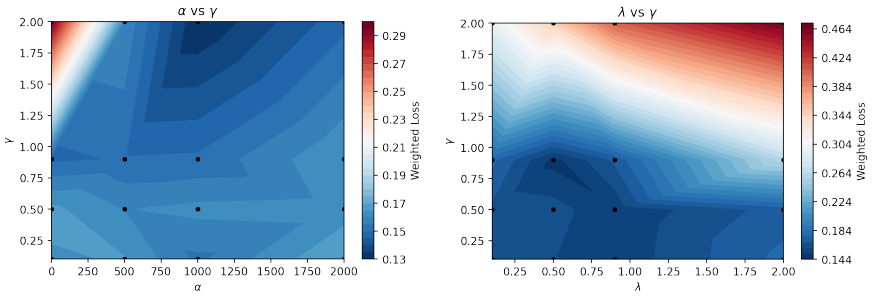
Table 3: Benchmark Methods Used for Comparison.

Method	Description
ResNet-18 [20]	Baseline Model with no domain adaptation
TCA [28]	Transfer Component Analysis
DAN [25]	Deep Adaptation Network
DANN [14]	Domain Adversarial Neural Network
JDOT [8]	Joint Distribution Optimal Transport
MCD [33]	Maximum Classifier Discrepancy
AFN [44]	Adaptive Feature Norm
RSD [7]	Representation Subspace Distance for Domain Adaptation Regression
ABRNet [43]	Adversarial Bi-Regressor Network for Domain Adaptive Regression

of the loss under the covariate shift where the weighted loss function is defined as following:

$$\begin{cases} \widehat{R}\left(g_\theta\right) = \dfrac{1}{N_{\mathcal{S}}} \displaystyle\sum_{i=1}^{N_{\mathcal{S}}} k^*(f_{\mathcal{S}}^i) \left\| y_{\mathcal{S}}^i - g_\theta\left(f_{\mathcal{S}}^i\right) \right\|^2 \\ \text{with } k^*(f_{\mathcal{S}}^i) = p_{\mathcal{T}}(f_{\mathcal{S}}^i)/p_{\mathcal{S}}(f_{\mathcal{S}}^i) \end{cases} \tag{16}$$

where $k^*(\cdot)$ is a weight function applied to training samples to make the empirical loss unbiased w.r.t. testing data, and $p_{\mathcal{S}}$ and $p_{\mathcal{T}}$ are the probability distributions of the source and target feature data respectively. Kernel Mean Matching [16] was used for estimating these weights. It accounts for the difference between the two distribution probabilities by re-weighting the training points such that the mean of the training samples is close to that of test samples.



(a) α vs γ contour plot. (b) λ vs γ contour plot.

Figure 9: Contour plots showing the weighted loss as a function of the model hyperparameters.

4.4. Results

*4.4.1. **Hyperparameter Tuning***

The optimal hyperparameters are estimated using the IWCV described in subsection 4.3. For that, we performed a grid search with discrete values within the grid intervals $\alpha \in [0, 500, 100, 2000]$, $\lambda \in [0.1, 0.5, 0.9, 2]$ and $\gamma \in [0.1, 0.5, 0.9, 2]$. Thus, the total number of trials is 64; However, each trial was repeated 3 times and the average was recorded. At each trial, we select a value from each grid interval and we record the weighted loss. The contour plots obtained on the dSprites dataset are shown in Figure 9. The results show that the model exhibits a stable performance for a wide range of moderate values (not high) of λ and γ (the sparse coding regularization power). However, we notice that the loss increases as these two terms increase. This can be interpreted as high sparse coding regularization can lead to a loss in significant features of both source and target feature matrices. For the smoothness parameter α, we can see that the model is robust to a wide range of values. In first place, α controls the term $\|\Delta \mathbf{D}_{\mathcal{S}}\|_F^2$ in the forward step of the training (where the dictionaries are being computed) and helps in achieving stable gradients in the backward step to minimize the domain loss. Therefore, this robustness around the values of α means that the model has stable gradients in every trial. The optimal values that were used are $\gamma = 0.9$, $\lambda = 0.5$ and $\alpha = 1000$.

4.4.2. ***dSprites***

The MAE results on the dSprites dataset are reported in Table 4. As we can see from the table, the proposed method achieved better scores in almost every task. The improvement in the results can be seen by looking at tasks where the source domain is **N** or **S**. This demonstrates the ability of the proposed model to transfer knowledge about the shapes in the images across domains without confusion with the background, such as noise and scream backgrounds that did not affect the performance compared to other state-of-the-art techniques. This is related to the fact that the proposed method seeks to make the features extracted from a noisy domain and the features from a non-noisy one share the same dictionaries. These dictionaries are task-related and therefore will eliminate the ones related to the noisy background. For tasks with source domain **C**, the method still achieves good results but close to the RSD method. Shallow methods, such as TCA and AFN, achieved the worst average scores, demonstrating the advantage of using deep neural-network based methods over traditional ones. This shows the importance of using a feature extractor (in our case a ResNet-18) to extract high-level representations from the input data. Besides, from these results, we can see that the proposed method achieves a close performance to the RSD method on some of the winning tasks. For that, we conducted a Kruskal-Wallis H-test to examine the null hypothesis whether the median of all of the several runs of each method is equal. For task N $\rightarrow$ C, the test showed a p-value of 0.008; Since this value is below 0.05, we reject the null hypothesis and therefore consider that the difference between the two sets of results is significant.

4.4.3. ***MPI3D***

The MAE results on the MPI3D dataset are reported in Table 5. As we can see from the table, the proposed method achieved better scores in every single task. The improvement that our model brings to the results can be seen by looking at tasks where the source domain is **T**. The massive improvement in the MAE scores highlights the power of the proposed model to learn in a

Table 4: MAE of the three regression tasks on dSprites (best scores are highlighted in red).

Method	C → N	C → S	N → C	N → S	S → C	S → N	Avg
ResNet-18 [20]	0.94 ± 0.06	0.90 ± 0.08	0.16 ± 0.02	0.65 ± 0.02	0.08 ± 0.01	0.26 ± 0.03	0.498
TCA [28]	0.94 ± 0.03	0.87 ± 0.02	0.19 ± 0.02	0.66 ± 0.05	0.10 ± 0.02	0.23 ± 0.04	0.498
DAN [25]	0.70 ± 0.05	0.77 ± 0.09	0.12 ± 0.03	0.50 ± 0.05	0.06 ± 0.02	0.11 ± 0.04	0.377
DANN [14]	0.47 ± 0.07	0.46 ± 0.07	0.16 ± 0.02	0.65 ± 0.05	0.05 ± 0.00	0.10 ± 0.01	0.315
JDOT [8]	0.86 ± 0.03	0.79 ± 0.02	0.19 ± 0.02	0.64 ± 0.05	0.10 ± 0.02	0.23 ± 0.04	0.468
MCD [33]	0.81 ± 0.09	0.81 ± 0.12	0.17 ± 0.12	0.65 ± 0.03	0.07 ± 0.02	0.19 ± 0.04	0.450
AFN [44]	1.00 ± 0.04	0.96 ± 0.05	0.16 ± 0.03	0.62 ± 0.04	0.08 ± 0.01	0.32 ± 0.06	0.523
RSD [7]	0.31 ± 0.03	0.31 ± 0.03	0.12 ± 0.02	0.53 ± 0.01	0.07 ± 0.00	0.08 ± 0.01	0.237
ABRNet [43]	**0.20 ± 0.02**	**0.27 ± 0.01**	0.14 ± 0.01	0.56 ± 0.03	0.05 ± 0.01	0.08 ± 0.02	**0.217**
Proposed Method	0.30 ± 0.03	0.39 ± 0.03	**0.11 ± 0.02**	**0.44 ± 0.02**	**0.04 ± 0.00**	**0.05 ± 0.00**	0.221

Table 5: MAE of 2 regression tasks on MPI3D (best scores are highlighted in red).

Method	RL → RC	RL → T	RC → RL	RC → T	T → RL	T → RC	Avg
ResNet-18 [20]	0.17 ± 0.02	0.44 ± 0.04	0.19 ± 0.02	0.45 ± 0.03	0.51 ± 0.01	0.50 ± 0.03	0.377
TCA [28]	0.17 ± 0.02	0.42 ± 0.01	0.19 ± 0.02	0.42 ± 0.02	0.50 ± 0.02	0.50 ± 0.02	0.373
DAN [25]	0.12 ± 0.03	0.35 ± 0.02	0.12 ± 0.02	0.27 ± 0.02	0.40 ± 0.02	0.41 ± 0.04	0.278
DANN [14]	**0.09** ± 0.01	0.24 ± 0.04	0.11 ± 0.03	0.41 ± 0.03	0.48 ± 0.02	0.37 ± 0.04	0.283
JDOT [8]	0.16 ± 0.02	0.41 ± 0.01	0.16 ± 0.02	0.41 ± 0.02	0.47 ± 0.02	0.47 ± 0.02	0.353
MCD [33]	0.13 ± 0.02	0.40 ± 0.04	0.15 ± 0.02	0.45 ± 0.01	0.52 ± 0.02	0.50 ± 0.03	0.358
AFN [44]	0.18 ± 0.03	0.45 ± 0.02	0.20 ± 0.03	0.46 ± 0.03	0.53 ± 0.02	0.52 ± 0.04	0.390
RSD [7]	**0.09 ± 0.01**	0.19 ± 0.02	**0.08 ± 0.00**	0.15 ± 0.03	**0.36 ± 0.01**	0.36 ± 0.02	0.205
Proposed Method	**0.09 ± 0.01**	**0.18 ± 0.02**	**0.08 ± 0.00**	**0.13 ± 0.02**	**0.36 ± 0.01**	**0.26 ± 0.02**	**0.185**

synthetic (i.e., toyish) environment and then generalize this knowledge to real domains, as opposed to most of the compared state-of-the-art techniques that failed to do so. As said before, the task-related dictionaries make the process robust to the type of environment whether real or toyish. Once again, shallow methods such as JDOT obtained the worst MAE scores. For tasks RL → T and RC → T, the Kruskal-Wallis H-test showed a p-value of 0.01 and 0.008, respectively; Since these p-values are below 0.05, we therefore consider that there is a significant difference between the results.

Table 6: MAE across three regression tasks on Biwi Kinect (best scores are highlighted in red).

Method	F → M	M → F	Avg
ResNet-18 [20]	0.38 ± 0.02	0.29 ± 0.01	0.335
TCA [28]	0.39 ± 0.01	0.31 ± 0.01	0.35
DAN [25]	0.37 ± 0.01	0.28 ± 0.01	0.325
DANN [14]	0.37 ± 0.02	0.30 ± 0.01	0.335
JDOT [8]	0.39 ± 0.01	0.29 ± 0.02	0.34
MCD [33]	0.37 ± 0.02	0.31 ± 0.02	0.34
AFN [44]	0.41 ± 0.02	0.32 ± 0.02	0.365
RSD [7]	0.30 ± 0.02	$\mathbf{0.26 \pm 0.01}$	**0.28**
Proposed Method	$\mathbf{0.29 \pm 0.012}$	0.27 ± 0.01	**0.28**

4.4.4. ***Biwi Kinect***

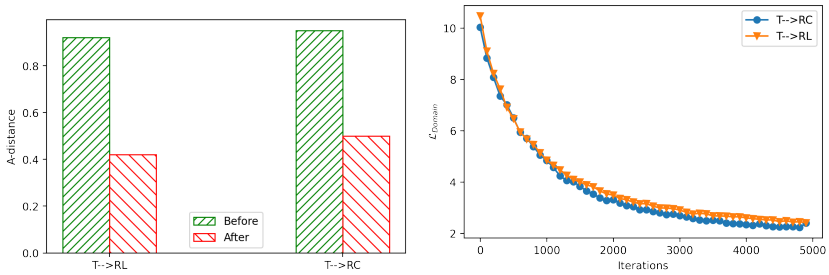
The MAE results on the Biwi Kinect dataset are reported in Table 6. As we can see from the table, the proposed method achieved the best average MAE score as well as the RSD method. These results show the ability of the model to perform adaptation in real-life images that often suffer from a lot of confusion between the object and its background.

4.4.5. ***Time Complexity***

The proposed domain adaptation algorithm contains a dictionary learning module that relies on an iterative strategy, and a sparse coding module. In the following, we provide a numerical analysis for the dSprites dataset, the other datasets having similar results. For a small batch size (e.g. 36), both modules achieve convergence in less than 50 iterations. Training the regression module only takes about 0.203 second for one batch. Adding the domain adaptation process increases the training time to 0.56 second, which remains acceptable compared to the gained training stability.

4.4.6. ***Representation Transferability***

To evaluate the performance of the model in transferring the representation across domains, we start by comparing the evolution of the distance between subspaces before and after the adaptation. To measure this distance, a common



(a) $\mathcal{A}$-distance (b) $\mathcal{L}_{Domain}$

Figure 10: Model performance on $\mathbf{T} \rightarrow \mathbf{RL}$ and $\mathbf{T} \rightarrow \mathbf{RC}$

metric is the $\mathcal{A}$-distance defined as following:

$$\mathcal{A}\text{-distance} = 1 - 2\ \epsilon \tag{17}$$

where ϵ is the generalization error of a binary classifier trained to distinguish between input samples from source and target domains. Figure 10a shows the $\mathcal{A}$-distance on two tasks of the MPI3D dataset: $\mathbf{T} \rightarrow \mathbf{RL}$ and $\mathbf{T} \rightarrow \mathbf{RC}$. We can see the decrease in the subspace distance on both tasks after adaptation, which highlights the ability of the model to adapt from toy to realistic or real domains. This can also be seen as the domain-related information in the feature matrices decreased after the adaptation process. Besides, Figure 10b shows the trends of the domain loss $\mathcal{L}_{Domain}$, which illustrates its smooth and stable decrease as a function of the number of iterations. In the same manner, Figure 11 shows the domain gap between the source and target dictionary extracted from the feature matrices before and after the adaptation. This illustration shows that after the adaptation, dSprites and MPI3D feature matrices hold information related to the scale and position of the object in the images and more head-pose information in Biwi feature matrices.

4.4.7. ***Sensitivity to*** β ***and*** m

Figure 12 shows the surface plot recording the MAE score as a function of loss trade-off parameter β and the rank m. The model shows a robust performance

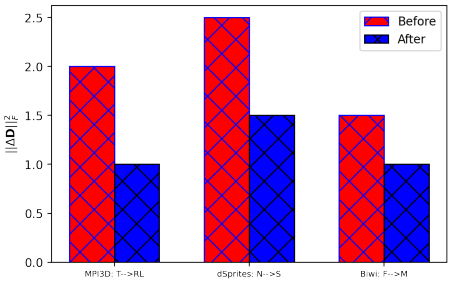


Figure 11: $\|\Delta \mathbf{D}_{\mathcal{S}}\|_F^2$

on a wide range of values of these two parameters. This shows that a dictionary learning block with a rank of factorization m is easy to manipulate as it does not require much user intervention. In addition, we can see the ability of the model to perform well with overcomplete dictionaries, namely when $m > b$. This sheds light on the effectiveness of the relaxation of the orthogonality carried out in our method, which provides a better representation of the domain under study. Besides, the robustness of the model to the values of β highlights the smoothness of training where both regression and domain losses can be minimized with different trade-off values.

4.4.8. ***Ablation Study***

In Table 4, Table 5 and Table 6, the first row represents the baseline model (ResNet-18) with no domain adaptation applied. We can see that applying the dictionary learning module improved the regression results in every scenario.

5. Conclusion

In this paper, we investigated unsupervised domain adaptation for regression tasks. We shed light on the difference between classification and regression tasks in domain adaptation with respect to robustness of these methods to the scattering of samples in feature space. For this purpose, we presented a new method for unsupervised domain adaptation based on dictionary learning. We introduced a new domain adaptation loss to minimize the subspace gap of source and target

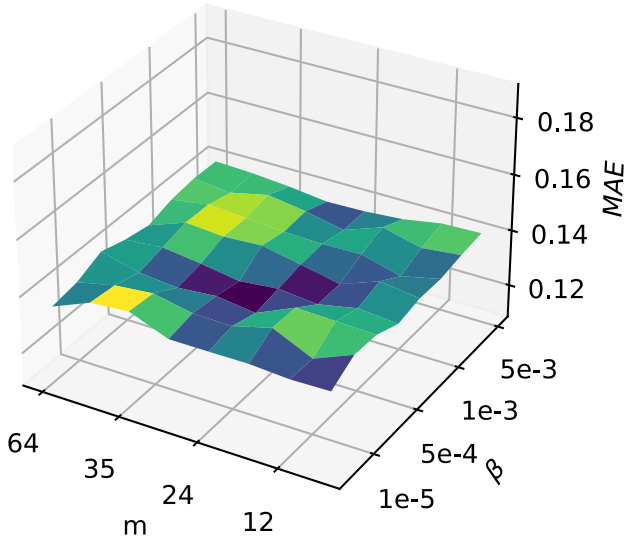


Figure 12: Hyperparameter sensitivity on task **RC** → **T**

representations extracted via deep neural networks. We extracted representations with common domain dictionaries so that a regression model trained on source data would provide good results on target data. Besides, we proposed a stable implementation of a dictionary learning module integrated inside a neural network with a friendly backpropagation mechanism and acceptable time complexity. For evaluation, we tested our model on several regression benchmarks where results showed better performance than state-of-the-art techniques. Results also highlighted the big gain in knowledge transfer of the proposed method while dealing with a synthetic source domain and a real target one. Future work will consist of extending the approach to embrace heterogeneous domains where the source and target data lie from different spaces.

References

[1] Baffour, A.A., Qin, Z., Geng, J., Ding, Y., Deng, F., Qin, Z., 2022. Generic network for domain adaptation based on self-supervised learning

and deep clustering. Neurocomputing 476, 126–136. URL: https://www.sciencedirect.com/science/article/pii/S092523122101955X, doi:https://doi.org/10.1016/j.neucom.2021.12.099.

[2] Baktashmotlagh, M., Harandi, M.T., Lovell, B.C., Salzmann, M., 2013. Unsupervised domain adaptation by domain invariant projection, in: Proceedings of the IEEE international conference on computer vision, pp. 769–776.

[3] Barata, J.C.A., Hussein, M.S., 2012. The Moore–Penrose pseudoinverse: A tutorial review of the theory. Brazilian Journal of Physics 42, 146–165.

[4] Beck, A., Teboulle, M., 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences 2, 183–202.

[5] Bi, X., Zhang, X., Wang, S., Zhang, H., 2022. Entropy-weighted reconstruction adversary and curriculum pseudo labeling for domain adaptation in semantic segmentation. Neurocomputing 506, 277–289. URL: https://www.sciencedirect.com/science/article/pii/S0925231222009511, doi:https://doi.org/10.1016/j.neucom.2022.07.073.

[6] Chen, M., Xu, Z., Weinberger, K., Sha, F., 2012. Marginalized denoising autoencoders for domain adaptation. arXiv preprint arXiv:1206.4683 .

[7] Chen, X., Wang, S., Wang, J., Long, M., 2021. Representation subspace distance for domain adaptation regression, in: International Conference on Machine Learning, PMLR. pp. 1749–1759.

[8] Courty, N., Flamary, R., Habrard, A., Rakotomamonjy, A., 2017. Joint distribution optimal transportation for domain adaptation. Advances in Neural Information Processing Systems 30.

[9] Dan, J., Jin, T., Chi, H., Shen, Y., Yu, J., Zhou, J., 2022. Homda: High-order moment-based domain alignment for unsupervised do-

main adaptation. Knowledge-Based Systems , 110205URL: https://www.sciencedirect.com/science/article/pii/S0950705122013016, doi:https://doi.org/10.1016/j.knosys.2022.110205.

[10] Dhaini, M., Berar, M., Honeine, P., Van Exem, A., 2022. End-to-end convolutional autoencoder for nonlinear hyperspectral unmixing. Remote Sensing 14, 3341.

[11] Engan, K., Aase, S.O., Husoy, J.H., 1999. Method of optimal directions for frame design, in: 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258), IEEE. pp. 2443–2446.

[12] Fang, T., Lu, N., Niu, G., Sugiyama, M., 2020. Rethinking importance weighting for deep learning under distribution shift. Advances in Neural Information Processing Systems 33, 11996–12007.

[13] Fu, S., Chen, J., Lei, L., 2022. Cooperative attention generative adversarial network for unsupervised domain adaptation. Knowledge-Based Systems , 110196.

[14] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V., 2016. Domain-adversarial training of neural networks. The journal of machine learning research 17, 2096–2030.

[15] Geng, X., Zhou, Z.H., Smith-Miles, K., 2007. Automatic age estimation based on facial aging patterns. IEEE Transactions on pattern analysis and machine intelligence 29, 2234–2240.

[16] Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., Schölkopf, B., 2009. Covariate shift by kernel mean matching. Dataset shift in machine learning 3, 5.

[17] Guo, B., Liu, T., Gu, Y., 2022. Integrating coupled dictionary learning and distance preserved probability distribution adaptation for multispectral–

hyperspectral image collaborative classification. IEEE Transactions on Geoscience and Remote Sensing 60, 1–13.

[18] Habrard, A., Peyrache, J.P., Sebban, M., 2013. Iterative self-labeling domain adaptation for linear structured image classification. International Journal on Artificial Intelligence Tools 22, 1360005.

[19] He, C., Zheng, L., Tan, T., Fan, X., Ye, Z., 2022. Manifold discrimination partial adversarial domain adaptation. Knowledge-Based Systems 252, 109320.

[20] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[21] Huang, F., Yates, A., 2010. Exploring representation-learning approaches to domain adaptation, in: Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing, pp. 23–30.

[22] Jiang, J., Zhai, C., 2007. Instance weighting for domain adaptation in nlp, ACL.

[23] Kang, G., Jiang, L., Yang, Y., Hauptmann, A.G., 2019. Contrastive adaptation network for unsupervised domain adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4893–4902.

[24] Lee, H., Battle, A., Raina, R., Ng, A., 2006. Efficient sparse coding algorithms. Advances in neural information processing systems 19.

[25] Long, M., Cao, Y., Wang, J., Jordan, M., 2015. Learning transferable features with deep adaptation networks, in: International conference on machine learning, PMLR. pp. 97–105.

[26] Long, M., Zhu, H., Wang, J., Jordan, M.I., 2016. Unsupervised domain adaptation with residual transfer networks. Advances in neural information processing systems 29.

[27] Nikzad-Langerodi, R., Zellinger, W., Saminger-Platz, S., Moser, B.A., 2020. Domain adaptation for regression under beer–lambert's law. Knowledge-Based Systems 210, 106447.

[28] Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q., 2010. Domain adaptation via transfer component analysis. IEEE transactions on neural networks 22, 199–210.

[29] Pinheiro, P.O., 2018. Unsupervised domain adaptation with similarity learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8004–8013.

[30] Qu, G., Li, N., 2019. Accelerated distributed nesterov gradient descent. IEEE Transactions on Automatic Control 65, 2566–2581.

[31] Raab, C., Röder, M., Schleif, F.M., 2022. Domain adversarial tangent subspace alignment for explainable domain adaptation. Neurocomputing 506, 418–429. URL: https://www.sciencedirect.com/science/article/pii/S0925231222009377, doi:https://doi.org/10.1016/j.neucom.2022.07.074.

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., 2015. Imagenet large scale visual recognition challenge. International journal of computer vision 115, 211–252.

[33] Saito, K., Watanabe, K., Ushiku, Y., Harada, T., 2018. Maximum classifier discrepancy for unsupervised domain adaptation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3723–3732.

[34] Sanodiya, R.K., Mishra, S., Arun, P., et al., 2022. Manifold embedded joint geometrical and statistical alignment for visual domain adaptation. Knowledge-Based Systems 257, 109886.

[35] Singh, A., Chakraborty, S., 2020. Deep domain adaptation for regression, in: Development and Analysis of Deep Learning Architectures. Springer, pp. 91–115.

[36] Sugiyama, M., Krauledat, M., Müller, K.R., 2007. Covariate shift adaptation by importance weighted cross validation. Journal of Machine Learning Research 8.

[37] Sun, B., Saenko, K., 2016. Deep coral: Correlation alignment for deep domain adaptation, in: European conference on computer vision, Springer. pp. 443–450.

[38] Tang, H., Wang, Y., Jia, K., 2022. Unsupervised domain adaptation via distilled discriminative clustering. Pattern Recognition 127, 108638.

[39] Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) 58, 267–288.

[40] Tropp, J.A., Gilbert, A.C., 2007. Signal recovery from random measurements via orthogonal matching pursuit. IEEE Transactions on information theory 53, 4655–4666.

[41] Wang, M., Deng, W., 2020. Deep face recognition with clustering based domain adaptation. Neurocomputing 393, 1–14.

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

[43] Xia, H., Wang, P., Koike-Akino, T., Wang, Y., Orlik, P., Ding, Z., . Adversarial bi-regressor network for domain adaptive regression. IJCAI 2022 .

[44] Xu, R., Li, G., Yang, J., Lin, L., 2019. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation,

in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1426–1435.

[45] Yamada, M., Sigal, L., Chang, Y., 2014. Domain adaptation for structured regression. International journal of computer vision 109, 126–145.

[46] Yang, Y., Zhang, T., Li, G., Kim, T., Wang, G., 2022. An unsupervised domain adaptation model based on dual-module adversarial training. Neurocomputing 475, 102–111. URL: https://www.sciencedirect.com/science/article/pii/S0925231221019160, doi:https://doi.org/10.1016/j.neucom.2021.12.060.

[47] Zhang, J., Lei, Q., Dhillon, I., 2018. Stabilizing gradients for deep neural networks via efficient svd parameterization, in: International Conference on Machine Learning, PMLR. pp. 5806–5814.