HAL open science

# DISENTANGLEMENT BY CYCLIC RECONSTRUCTION

David Bertoin, Emmanuel Rachelson

**To cite this version:**

David Bertoin, Emmanuel Rachelson. DISENTANGLEMENT BY CYCLIC RECONSTRUCTION. 2021. hal-03501330v1

**HAL Id: hal-03501330**

**https://hal.science/hal-03501330v1**

Preprint submitted on 23 Dec 2021 (v1), last revised 29 Sep 2022 (v2)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# DISENTANGLEMENT BY CYCLIC RECONSTRUCTION

**David Bertoin**
IRT Saint-Exupéry
ISAE-SUPAERO
ANITI
Toulouse, France
david.bertoin@irt-saintexupery.com

**Emmanuel Rachelson**
ISAE-SUPAERO
Université de Toulouse
ANITI
Toulouse, France
emmanuel.rachelson@isae-supaero.fr

**ABSTRACT**

Deep neural networks have demonstrated their ability to automatically extract meaningful features from data. However, in supervised learning, information specific to the dataset used for training, but irrelevant to the task at hand, may remain encoded in the extracted representations. This remaining information introduces a domain-specific bias, weakening the generalization performance. In this work, we propose splitting the information into a task-related representation and its complementary context representation. We propose an original method, combining adversarial feature predictors and cyclic reconstruction, to disentangle these two representations in the single-domain supervised case. We then adapt this method to the unsupervised domain adaptation problem, consisting of training a model capable of performing on both a source and a target domain. In particular, our method promotes disentanglement in the target domain, despite the absence of training labels. This enables the isolation of task-specific information from both domains and a projection into a common representation. The task-specific representation allows efficient transfer of knowledge acquired from the source domain to the target domain. In the single-domain case, we demonstrate the quality of our representations on information retrieval tasks and the generalization benefits induced by sharpened task-specific representations. We then validate the proposed method on several classical domain adaptation benchmarks and illustrate the benefits of disentanglement for domain adaptation.

## 1 Introduction

The wide adoption of Deep Neural Networks in practical supervised learning applications is hindered by their sensitivity to the training data distribution. This problem, known as *domain shift*, can drastically weaken, in real-life operating conditions, the performance of a model that seemed perfectly efficient in simulation. Learning a model with the goal of making it robust to a specific domain shift is called *domain adaptation* (DA). The data available to achieve DA often consist of a labeled training set from a source domain and an unlabeled sample set from a target domain. This yields the problem of *unsupervised domain adaptation* (UDA).

In this work, we take an information disentanglement perspective on UDA. We argue that a key to efficient UDA lies in separating the necessary information to complete the network's task (classification or regression) from a task-orthogonal information which we call context or *style*. Disentanglement in the target domain seems however a difficult endeavor since the available data is unlabeled. Our contribution is two-fold. We propose a formal definition of the disentanglement problem for UDA which, to the best of our knowledge, is new. Then we design a new learning method, called DiCyR (Disentanglement by Cyclic Reconstruction), which relies on cyclic reconstruction of inputs in order to achieve efficient disentanglement, including in the target domain. We derive DiCyR both in the supervised learning and in the UDA cases.

Although this paper is developed mainly around UDA, we emphasize that our contribution lies in the disentanglement between task and context information. Its consequences carry out to information retrieval in the single domain case as well as to UDA.

This paper is organized as follows. Section 2 presents the required background on supervised learning and UDA, and proposes a definition of disentanglement for UDA. Section 3 reviews recent works in the literature that allow for a critical look at our contribution and put it in perspective. Section 4 introduces DiCyR, first for the single-domain case, and then for the UDA problem. Section 5 empirically evaluates DiCyR against state-of-the-art methods and discusses its strengths, weaknesses, and variants. Section 6 summarizes and concludes this paper.

## 2 Problem definition

In this section, we introduce the notations and background upon which we build the contributions of Section 4. Let $\mathcal{X}$ be an input space of descriptors and $\mathcal{Y}$ an output space of labels. A supervised learning problem is defined by a distribution $p_s(x, y)$ over elements of $\mathcal{X} \times \mathcal{Y}$. In what follows, $p_s$ will be called the *source* distribution. One wishes to estimate a mapping $\hat{f}$ that minimizes a risk function of the form $\mathbb{E}_{(x,y)\sim p_s}[l(\hat{f}(x), y)]$. The optimal estimator is denoted $f$ and one often writes the distribution $\mathbb{P}(y|x)$ as $y \sim f(x) + \eta$, where $\eta$ captures the deviations between $y$ and $f(x)$. Hence, one tries to learn $f$. In practice, the risk can only be approximated using a finite set of samples $\{(x_i, y_i)\}_{i=1}^n$ all independently drawn from $p_s$ and $\hat{f}$ is a parametric function (such as a deep neural network) of the form $y = \hat{f}(x; \theta)$.

Domain adaptation (DA) consists in considering a *target* distribution $p_t$ over $\mathcal{X} \times \mathcal{Y}$ that differs from $p_s$, and the transfer of knowledge from learning in the source domain ($p_s$) to the target domain ($p_t$). Specifically, *unsupervised* DA exploits the knowledge of a labelled training set $\{(x_i^s, y_i^s)\}_{i=1}^n$ sampled according to $p_s$, and an unlabelled data set $\{(x_i^t)\}_{i=1}^m$ sampled according to $p_t$. For instance, the source domain data could be a set of labelled photographs of faces, and the target domain data, a set of unlabelled face photographs, taken with a different camera under different exposure conditions. The problem consists in minimizing the target loss $\mathbb{E}_{(x,y)\sim p_t}[l(\hat{f}(x), y)]$.

We suppose that a necessary condition to benefit from the knowledge available in the source domain and transfer it to the target domain is the existence of a common information manifold between domains, where an input's projection is sufficient to predict the labels. We call this useful information *task-specific* or *task-related*. The complementary information should be called *task-orthogonal*; it is composed of information present in the input but irrelevant to the task at hand. For the sake of naming simplicity, we will call this information *style*. However, we insist that this should not be confused with the classical notion of style.

Let $\Pi_\tau : \mathcal{X} \to \mathcal{T}$ and $\Pi_\sigma : \mathcal{X} \to \mathcal{S}$ denote two projection operators, where $\mathcal{T}$ and $\mathcal{S}$ denote respectively the latent task-related information space and the latent style-related information space. Let $\Pi$ be the joint projection $\Pi(x) = (\Pi_\tau(x), \Pi_\sigma(x))$. Conversely, we shall note $\bar{\Pi} : \mathcal{T} \times \mathcal{S} \to \mathcal{X}$ a reconstruction operator. And finally, $c : \mathcal{T} \to \mathcal{Y}$ will denote the labeling operator which only uses information from $\mathcal{T}$. We consider that the information of the elements of $\mathcal{X}$ is correctly disentangled by $\Pi = (\Pi_\tau, \Pi_\sigma)$ if one can find $\bar{\Pi}$ and $c$ such that:

- C1: $c \circ \Pi_\tau$ minimizes the risk (and thus fits $f$ on the appropriate domain),
- C2: $\bar{\Pi} \circ \Pi$ fits the identity operator $id_{\mathcal{X}}$,
- C3: With $X, T, S$ the random variables in $\mathcal{X}, \mathcal{T}, \mathcal{S}$, the mutual information $I(T, S|X)$ between $T$ and $S$ conditioned by $X$, is 0,
- C4: There is no $g : \mathcal{T} \to \mathcal{X}$ such that $g \circ \Pi_\tau = id_{\mathcal{X}}$,

Condition C1 imposes that the projection into $\mathcal{T}$ retains enough information to correctly label samples. Condition C2 imposes that all the information necessary for the reconstruction is preserved by the separation performed by $\Pi$. Condition C3 states that no information is present in both $\mathcal{T}$ and $\mathcal{S}$. Condition C4 imposes that the information contained in $\mathcal{T}$ alone is insufficient to reconstruct an input, and thus the information of $\mathcal{S}$ is necessary. Note that the symmetrical condition is unnecessary, since the combination of C1 and C3 already guarantees that $\mathcal{S}$ cannot contain the task-related information. Overall, solving this disentanglement problem for DA implies finding a quadruplet $\langle \Pi_\tau, \Pi_\sigma, \bar{\Pi}, c \rangle$ that meets the conditions above. In particular, note that conditions C3 and C4 open a perspective to a formulation of disentanglement in the general case. To the best of our knowledge, this is the first explicit set of necessary and sufficient conditions defining disentanglement.

## 3 Related work

Disentanglement between the domain-invariant, task-related information and the domain-specific, task-orthogonal, style information is a desirable property to have for DA. In the next paragraphs, we cover important work in representation disentanglement, domain adaptation, and their interplay. For each contribution, we evaluate whether it complies with conditions C1 to C4, and how.

Before deep learning became prevalent, [40] presented a method using bi-linear models able to separate style from content. More recently, methods based on generative models have demonstrated the ability to disentangle factors of variations from elements of a single domain [30, 26, 7, 13, 33]. In a cross-domain setting, [12] use pairs of images with the same labels from different domains to separate representations into a shared information common to both domains and a domain-exclusive information. We note that these approaches do not explicitly aim at respecting all conditions listed in Section 2. Additionally, most require labeled datasets (and in some cases even paired datasets) and thus do not address the *unsupervised* DA problem.

One approach to UDA consists in aligning the source and target distributions statistics, a topic closely related to *batch normalization* [18]. CORAL [37] minimizes the distance between the covariance matrices of the features extracted from the source and target domains. Assuming the domain-specific information is contained inside the batch normalization layers, AdaBN [22] aligns the batch statistics by adopting a specific normalization for each domain. Autodial [6] aims to align source and target feature distributions to a reference one and introduce domain alignment layers to automatically learn the degree of feature alignment needed at different levels of the network. Similarly, DWT [31] replaces batch normalization layers with domain alignment layers implementing a so-called feature whitening. A significant asset of these methods is the possibility to be used jointly with other DA methods (including the one we propose in Section 4). These methods jointly learn a common representation for elements from both domains. Conversely, SHOT [23] freezes the representations learned in the source domain before training a target-specific encoder to align the representations of the target elements by maximizing the mutual information between intermediate feature representations and outputs of the classifier.

Ensemble methods have also been applied to UDA [19, 39]. SEDA [8] combines stochastic data augmentation with self-ensembling to minimize the prediction differences between a student and a teacher network in the target domain.

Another approach involves learning domain-invariant features, that do not allow to discriminate whether a sample belongs to the source or target domain, while still permitting accurate labeling in the source domain. This approach relies on the assumption that such features allow efficient labeling in the target domain. DRCN [11] builds a two-headed network sharing common layers; one head performs classification in the source domain, while the second is a decoder that performs reconstruction for target domain elements. [10] propose the DANN method and introduce Gradient Reversal Layers to connect a domain discriminator and a feature extractor. These layers invert the gradient sign during back-propagation so that the feature extractor is trained to fool the domain discriminator. WDGRL [35] modifies DANN and replaces the domain discriminator with a network that approximates the Wasserstein distance between domains. ADDA [41] optimizes, in an adversarial setting, a generator and a discriminator with an inverted label loss.

Other methods focus on explicitly disentangling an information shared between domains (analogous to the domain-invariant features above) from a domain-specific information. Inspired by InfoGAN [7], CDRD [25] isolate a latent factor, representing the domain information, from the rest of an encoding, by maximizing the mutual information between generated images and this latent factor. Some domain information may still be present in the remaining part of the encoding and thus may not comply with conditions C3 and C4.

UFDN [24] (and also [21]) trains an encoder to produce domain-invariant representations used by an image generator trained to fool a discriminator with cross-domain images. DSN [2] also produces domain-invariant features by training a shared encoder to fool a domain discriminator. It trains two domain-private encoders with a difference loss that encourages orthogonality between the shared and the private representations (similarly to condition C3). DIDA [5] (but also [4] and [28]) combines a domain discriminator with an adversarial classifier to separate the information shared between domains from the domain-specific information.

All these methods build a shared representation that prevents discriminating between source and target domains, while retaining enough information to correctly label samples from the source domain. However, because they rely on an adversarial classifier that requires labeled data, they do not guarantee that the complementary, domain-specific information for samples *in the target domain* does not contain information that overlaps with the shared representation. In other words, they only enforce C3 in the source domain. They rely on the assumption that the disentanglement will still hold when applied to target domain elements, which might not be true.

Another identified weakness in methods that achieve a domain-invariant feature space is that their representations might not allow for accurate labeling in the target domain. Indeed, feature alignment does not necessarily imply a correct mapping between domains. To illustrate this point, consider a binary classification problem (classes $c_1$ and $c_2$) and two domains ($d_1$ and $d_2$). Let $(c_1, d_1)$ denote samples of class $c_1$ in $d_1$. It is possible to construct an encoding that projects $(c_1, d_1)$ and $(c_2, d_2)$ to the same feature values. The same holds for $(c_1, d_2)$ and $(c_2, d_1)$ for different feature values. This encoding allows discriminating between classes in $d_1$. It also fools a domain discriminator since it does not allow predicting the original domain of a projected element. However, applying the classification function learned on $d_1$ to the projected $d_2$ elements leads to catastrophic predictions.

Transforming a sample from one domain to the other, while retaining its label information can be accomplished by image-to-image translation methods. Using an adversarial setting, SBADA-GAN [32], ACAL [16], and CyCADA [15] extend the cycle consistency introduced in CycleGAN [42]. A major drawback of these methods lies in the possible instability during training that is caused by the min-max optimization problem induced by the adversarial training of generators and discriminators.

In the next section, we introduce a method that does not rely on a domain discriminator and an adversarial label predictor, but directly minimizes the information sharing between representations. This allows to guarantee that there is no information redundancy between the task-related and the task-orthogonal style information in both the source and the target domains. Along the way, it provides an efficient mechanism to disentangle the task-related information from the style information in the single domain case. Our method combines information disentanglement, intra-domain and cross-domain cyclic consistency, to enforce a more principled mapping between each domain.

# 4 Disentanglement with Gradient Reversal Layers and cyclic reconstruction

First, we propose an original method to disentangle the task-related information from the style information for a single domain in a supervised learning setting. In a second step, we propose an adaptation of this method to learn these disentangled representations in both domains for UDA. This disentanglement allows, in turn, to efficiently predict labels in the target domain.

## 4.1 Task-style disentanglement in a single domain

Our approach consists in estimating jointly $\Pi$, $\bar{\Pi}$ and $c$ as a deep feed-forward neural network. We shall note $\theta_\Pi$, $\theta_{\bar{\Pi}}$, and $\theta_c$ the parameters of the respective sub-parts of the network. $\Pi \circ \bar{\Pi}$ takes the form of an auto-encoder, while $\Pi \circ c$ is a task-related (classification or regression) network. Figure 1a summarizes the global architecture which we detail in the following paragraphs.

Conditions C1 and C2 are expressed through the definition of a task-specific loss $\mathcal{L}_{task}$ (*e.g.* cross-entropy for classification, L2 loss for regression) and a reconstruction loss $\mathcal{L}_{reco}$. Thus, the update of $\theta_\Pi$ should follow $-\nabla_{\theta_\Pi}(\mathcal{L}_{task} + \mathcal{L}_{reco})$, the update of $\theta_{\bar{\Pi}}$ relies on $-\nabla_{\theta_{\bar{\Pi}}}\mathcal{L}_{reco}$, and that of $\theta_c$ uses $-\nabla_{\theta_c}\mathcal{L}_{task}$.

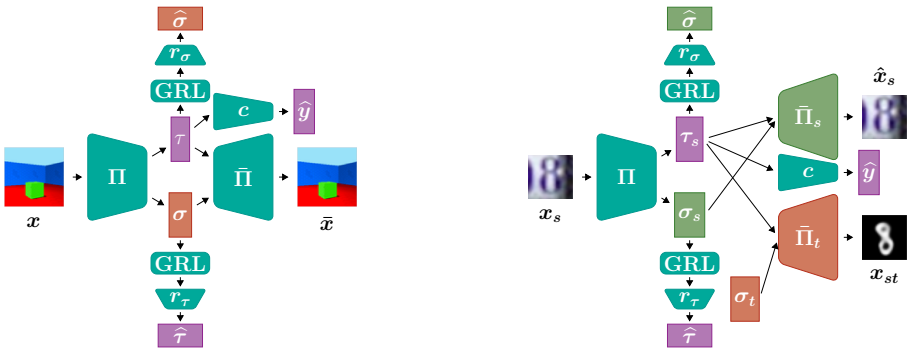
In order to achieve condition C3, we exploit Gradient Reversal Layers [10, GRL]. Note that this choice is arbitrary and alternatives exist (e.g. [33]) to enforce C3. We train two side networks $r_\tau : \mathcal{S} \rightarrow \mathcal{T}$ and $r_\sigma : \mathcal{T} \rightarrow \mathcal{S}$ whose purpose is to attempt to predict $T$ given $S$, and $S$ given $T$ respectively. For a given $x$, let us write $(\tau, \sigma) = \Pi(x)$, $\widehat{\tau} = r_\tau(\sigma)$, and $\widehat{\sigma} = r_\sigma(\tau)$. We train $r_\tau$ and $r_\sigma$ to minimize the losses $\mathcal{L}_{r_\tau} = \|\tau - \widehat{\tau}\|_2$ and $\mathcal{L}_{r_\sigma} = \|\sigma - \widehat{\sigma}\|_2$. Let $\mathcal{L}_{info} = \mathcal{L}_{r_\tau} + \mathcal{L}_{r_\sigma}$ denote the combination of these losses. We connect these two sub-networks to the whole architecture using GRLs. GRLs behave as the identity function during the forward pass and invert the gradient sign during the backward pass, hence pushing the parameters to maximize the output loss. During training, this architecture constrains $\Pi$ to produce features in $\mathcal{T}$ and $\mathcal{S}$ with the least information shared between them. Consequently, the update of $\theta_\Pi$ follows $+\nabla_{\theta_\Pi}\mathcal{L}_{info}$.

This constraint efficiently avoids information redundancy between $\mathcal{T}$ and $\mathcal{S}$. However, it does not avoid all the information being pushed into $\mathcal{T}$. Preventing this undesirable behavior is the purpose of condition C4. To that end, we use a cyclic reconstruction scheme. Consider two elements $x$ and $x'$ from $\mathcal{X}$, and their associated $(\tau, \sigma) = \Pi(x)$ and $(\tau', \sigma') = \Pi(x')$. Let $\tilde{x} = \bar{\Pi}(\tau, \sigma')$ be the reconstruction of $\tau$ that uses the style $\sigma'$ of $x'$. A correct allotment of the information between $\mathcal{T}$ and $\mathcal{S}$ requires that the task and style information be preserved in $(\tilde{\tau}, \tilde{\sigma}) = \Pi(\tilde{x})$. So, we wish to have $\tilde{\tau}$ as close as possible to $\tau$, or, alternatively, to have $c(\tilde{\tau})$ as close as possible to $c(\tau)$. Similarly, we wish to have $\tilde{\sigma}$ as close as possible to $\sigma'$. To enforce C4 and avoid the degenerate case where the encoder predicts a constant style for all samples, we force $\tilde{\sigma}$ to lie sufficiently far from $\sigma$ to avoid style confusion. We achieve this with a triplet loss [34] using $\tilde{\sigma}$ as the anchor, $\sigma'$ and $\sigma$ as, respectively, the positive and negative inputs, and a margin $m$. Thus C4 results in minimizing the cyclic reconstruction loss $\mathcal{L}_{cyclic} = \|\tilde{\tau} - \tau\|_2 + max\{\|\tilde{\sigma} - \sigma'\|_2 - \|\tilde{\sigma} - \sigma\|_2 + m, 0\}$.

Overall, with a learning rate $\alpha$, the gradient-based update procedure of the network parameters boils down to:

$$
\begin{aligned}
\theta_\Pi &\leftarrow \theta_\Pi - \alpha\nabla_{\theta_\Pi}\left(\mathcal{L}_{task} + \mathcal{L}_{reco} - \mathcal{L}_{info} + \mathcal{L}_{cyclic}\right), \\
\theta_{\bar{\Pi}} &\leftarrow \theta_{\bar{\Pi}} - \alpha\nabla_{\theta_{\bar{\Pi}}}(\mathcal{L}_{reco} + \mathcal{L}_{cyclic}), \\
\theta_c &\leftarrow \theta_c - \alpha\nabla_{\theta_c}\mathcal{L}_{task}, \\
\theta_{r_\tau} &\leftarrow \theta_{r_\tau} - \alpha\nabla_{\theta_{r_\tau}}\mathcal{L}_{r_\tau} \\
\theta_{r_\sigma} &\leftarrow \theta_{r_\sigma} - \alpha\nabla_{\theta_{r_\sigma}}\mathcal{L}_{r_\sigma}.
\end{aligned}
$$

We call this method DiCyR for Disentanglement by Cyclic Reconstruction.




(a) Supervised learning

(b) Unsupervised domain adaptation

Figure 1: Network architectures

### 4.2 Task-style disentanglement in the unsupervised domain adaptation case

We propose a variation of DiCyR for UDA, where we replace the decoder $\bar{\Pi}$ by two domain-specific decoders, $\bar{\Pi}_s$ and $\bar{\Pi}_t$. We shall compensate for the lack of labeled data in the target domain by computing cross-domain cyclic reconstructions.

Let $(x_s, y_s)$ be a sample from the source domain and $x_t$ be a sample from the target domain. Let us denote $(\tau_s, \sigma_s) = \Pi(x_s)$ and $(\tau_t, \sigma_t) = \Pi(x_t)$, the corresponding projections in the latent task and style-related information spaces. Then one can define, as in the previous section, $\mathcal{L}_{task}$ as the task-specific loss on the source domain, and $\mathcal{L}_{reco_s}$ and $\mathcal{L}_{reco_t}$ as the reconstruction losses in the source and target domains respectively. As previously, we constrain the task-related representation and the style representation not to share information using two networks $r_\tau$ and $r_\sigma$, connected to the main architecture by GRL layers (Figure 1b), allowing the definition of the $\mathcal{L}_{r_\tau}$, $\mathcal{L}_{r_\sigma}$ and $\mathcal{L}_{info}$ losses. Lastly, we exploit cyclic reconstructions in both domains to correctly disentangle the information and hence define the same $\mathcal{L}_{cyclic}$ loss as above.

This disentanglement in the target domain separates the global information in two but does not guarantee that what is being pushed into $\tau$ is really the task-related information. This can only be enforced by cross-domain knowledge (since no correct labels are available in the target domain). Thus, finally, we would like to allow projections from one domain into the other while retaining the task-related information, hence allowing domain adaption. Using the notations above, we construct $x_{ts} = \bar{\Pi}_t(\tau_s, \sigma_t)$, the reconstruction of $x_s$'s task-related information, in the style of $x_t$. This creates an artificial sample in the target domain, whose label is $y_s$. Then, with $(\tau_{ts}, \sigma_{ts}) = \Pi(x_{ts})$, one wishes to have $\tau_{ts}$ match closely $\tau_s$ (or, alternatively, $c(\tau_{ts})$ match closely $y_s$) in order to prevent the loss of task information during the cross-domain projection and thus to constrain the task representations to be domain-invariant. Symmetrically, one can construct the artificial sample $x_{st} = \bar{\Pi}_s(\tau_t, \sigma_s)$ and enforce that $\tau_{st}$ closely matches $\tau_t$. Note that the label of $x_{st}$ is unknown and yet it is still possible to enforce the disentanglement by cyclic reconstruction.

Overall, these terms boil down to a cross-domain cyclic reconstruction loss for UDA $\mathcal{L}_{domain_cyclic} = \|\tau_s - \tau_{ts}\|_2 + \|\tau_t - \tau_{st}\|_2$.

Finally, the network parameters are updated according to:

$$
\begin{aligned}
\theta_\Pi &\leftarrow \theta_\Pi - \alpha \nabla_{\theta_\Pi} (\mathcal{L}_{task} + \mathcal{L}_{reco_s} + \mathcal{L}_{reco_t} - \mathcal{L}_{info} + \mathcal{L}_{cyclic} + \mathcal{L}_{domain_cyclic}) \\
\theta_{\bar{\Pi}_s} &\leftarrow \theta_{\bar{\Pi}_s} - \alpha \nabla_{\theta_{\bar{\Pi}_s}} (\mathcal{L}_{reco_s} + \mathcal{L}_{cyclic}) \\
\theta_{\bar{\Pi}_t} &\leftarrow \theta_{\bar{\Pi}_t} - \alpha \nabla_{\theta_{\bar{\Pi}_t}} (\mathcal{L}_{reco_t} + \mathcal{L}_{cyclic}), \\
\theta_c &\leftarrow \theta_c - \alpha \nabla_{\theta_c} \mathcal{L}_{task}, \\
\theta_{r_\tau} &\leftarrow \theta_{r_\tau} - \alpha \nabla_{\theta_{r_\tau}} \mathcal{L}_{r_\tau} \\
\theta_{r_\sigma} &\leftarrow \theta_{r_\sigma} - \alpha \nabla_{\theta_{r_\sigma}} \mathcal{L}_{r_\sigma}.
\end{aligned}
$$

## 5 Experimental results and discussion

We first evaluate DiCyR's ability to disentangle the task-related information from the style information in the supervised context. Then we demonstrate DiCyR's efficiency on UDA[1]. Hyperparameters, network architectures and implementation choices are summarized in Appendix B

### 5.1 Single-domain disentanglement

We evaluate the disentanglement performance of DiCyR by following the protocol introduced by [26]. Since we do not use generative models, we only focus on their two first items: *swapping* and *retrieval*. We evaluate DiCyR on the SVHN [27], and 3D Shapes [3] disentanglement benchmarks. The task is predicting the central digit in the image for the SVHN dataset, and the shape of the central object in the scene for the 3D Shapes dataset.

*Swapping* involves swapping styles between samples and visually assessing the realism of the generated image. It combines the task-related information $\tau_i$ of a sample $x_i$ with the style $\sigma_j$ of another sample $x_j$. We use the decoder to produce an output $\tilde{x}_{ij}$. Figure 2 shows randomly generated outputs on the two datasets. DiCyR produces visually realistic artificial images with the desired styles.

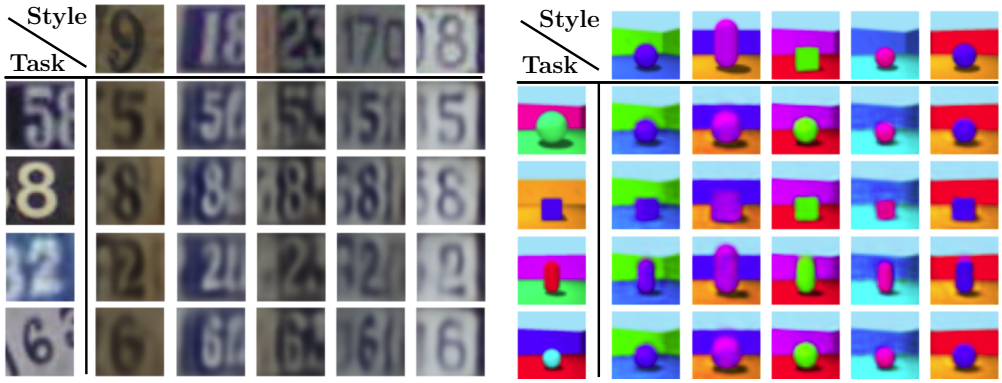



Figure 2: Swapping styles on SVHN and 3D Shapes

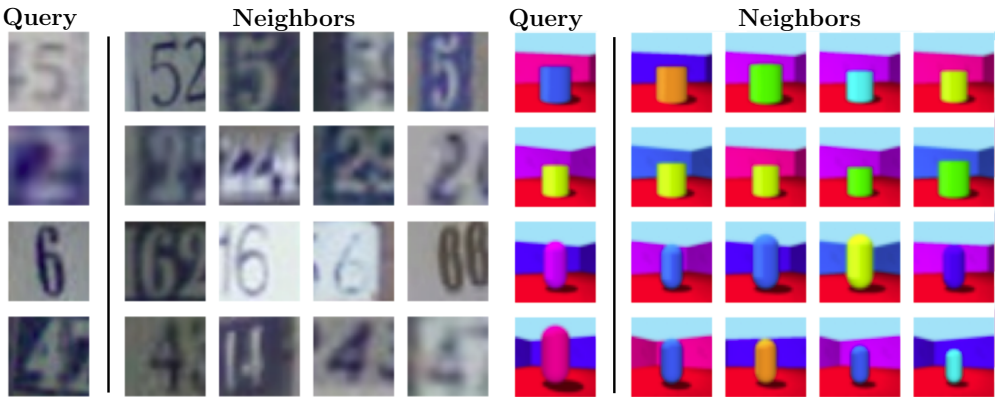
| Method | SVHN | 3D Shape | floor hue | wall hue | object hue | scale | orientation |
|---|---|---|---|---|---|---|---|
| Full features | 0.98 | 1 | 0.94 | 0.94 | 0.89 | 0.6 | 0.5 |
| Task-related features | 0.98 | 1 | 0.11 | 0.12 | 0.13 | 0.15 | 0.10 |
| Style features only | 0.17 | 0.26 | 0.89 | 0.95 | 0.88 | 0.59 | 0.42 |
| Random guess | 0.10 | 0.25 | 0.10 | 0.10 | 0.10 | 0.125 | 0.067 |

Table 1: Classification accuracy using task, style or full information, on SVHN and 3D shapes.

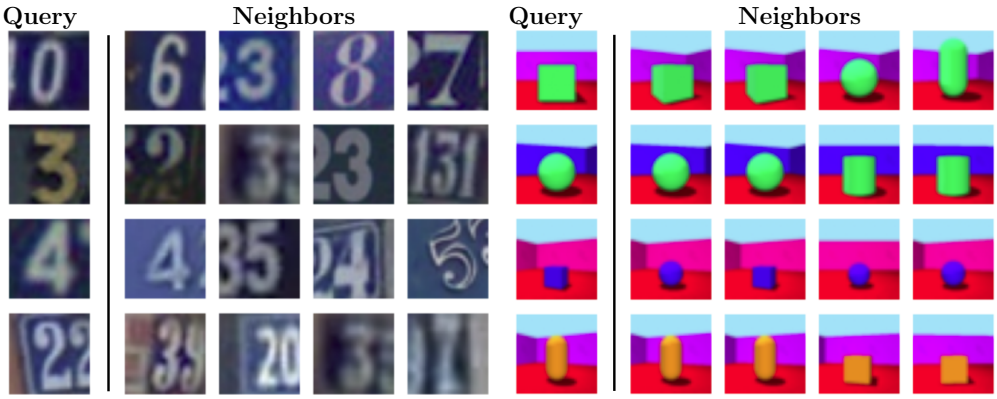
*Retrieval* concerns finding, in the dataset, the nearest neighbors in the embedding space for an image query. We carry out this search for nearest neighbors using the Euclidean distance on both the task-related and the style representations. A good indicator of the effectiveness of the information disentanglement would be to observe neighbors with the same labels as the query when computing distances on the task-related information space, and neighbors with similar style when using the style information. Figure 3 demonstrate that the neighbors found when using the task-related information are samples with the same label as the query's label and that the neighbors found using the style representation share many characteristics with the query but not necessarily the same labels.

We ran a quantitative evaluation of disentanglement by training a neural network classifier with a single hidden layer of 32 units to predict labels, using either the task-related information alone, or the style information alone. If the information is correctly separated, we expect the classifier trained with task-related information only to get similar performance to a classifier trained with full information. Conversely, the classifier trained with the style information only should reach similar performance to a random guess (10% accuracy on SVHN, 25% on 3D Shapes). Table 1 (first two columns) reports the obtained testing accuracies.

[1]Code and pre-trained networks available at `https://github.com/SuReLI/DiCyR_code`.




(a) Query on the task-related representation




(b) Query on the style representation

Figure 3: Nearest neighbors according to each representation

Figure 4: Train and test data

It appears that the task-related representation contains enough information to correctly predict labels. We also observe that full disentanglement is closely but not perfectly achieved, as the classifier trained only with style information behaves slightly better than random choice. To quantify how much style information is being unduly encoded in the task-related representation, we ran a similar experiment to predict the five other style variation factors in 3D Shapes (floor hue, wall hue, object hue, scale and orientation). The trained classifier reaches accuracies (Table 1 rightmost columns) that are very close to a random guess, thus validating the disentanglement quality.

Without proper disentanglement, the features extracted by neural networks may contain context information that is specific to the training data distribution and unrelated to the task at hand. This context information may comprehend bias, introduced during the collection of the data for example, which can strongly affect generalization. To illustrate this phenomenon, we train a network to classify zeros and ones extracted from the MNIST dataset. During training, we modify the dataset so that all ones have a yellow color and all zeros have a blue color while varying the intensity of the

| Method | Train accuracy | Test accuracy |
|---|---|---|
| Supervised learning | 1.0 | 0.067 |
| DiCyR | 0.984 | 0.957 |

Table 2: Vanilla classifier vs. DiCyR on a biased dataset

color. We then measure the accuracy of this network on a test dataset in which the colors of the ones and zeros are inverted (Figure 4). The accuracies, reported in Table 2, show that the network is incapable of generalizing on the test set. Conversely, by training DiCyR on the same training set, we observe that its accuracies remain similar despite the change of color. While the first network's predictions exploit the bias present in the training set's context information, DiCyR is insensitive to it. Its ability to disentangle the task information from the context information allows DiCyR to retain its performance on data sampled from a different input distribution. Hence DiCyR helps generalize more robustly on unseen test distributions, which opens perspectives for domain adaptation.

### 5.2 Unsupervised domain adaptation problem

We evaluate DiCyR by performing domain adaptation between the MNIST [20], SVHN, and USPS [17] datasets, and between the Syn-Signs [9] and the GTSRB [36] datasets. Following common practice in the literature, we trained our network on four different settings: MNIST→USPS, USPS→MNIST, SVHN→MNIST, and Syn-Signs→GTSRB. We measure the classification performance in the target domain and compare it with state-of-the-art methods (Table 3). We also compare with a baseline classifier that is only trained on the source domain data. Values reported in Table 3 are quoted from their original papers.[2]

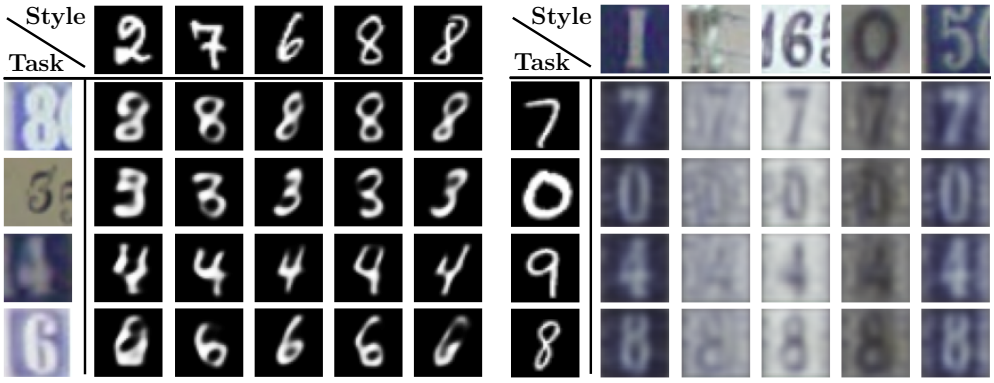
| Method | Source<br>Target | MNIST<br>USPS | USPS<br>MNIST | SVHN<br>MNIST | Syn-Signs<br>GTSRB |
|---|---|---|---|---|---|
| Baseline | | 78.1 | 58.0 | 60.2 | 79.0 |
| DSN [2] | | 91.3 | - | 82.7 | 93.1 |
| DiDA [5] | | 92.5 | - | 83.6 | - |
| SBADA-GAN [32] | | 97.6 | 95.0 | 76.1 | 96.7 |
| CyCADA [15] | | 95.6 | 96.5 | 90.4 | - |
| ACAL [16] | | 98.3 | 97.2 | 96.5 | - |
| DiCyR (ours) | | **98.7** | **98.3** | **97.7** | **97.4** |
| DANN [10] | | 85.1 | 73.0 | 73.9 | 88.6 |
| ADDA [41] | | 89.4 | 90.1 | 76.0 | - |
| DRCN [11] | | 91.8 | 73.7 | 82.0 | - |
| DWT [31] | | 99.1 | 98.8 | 97.7 | - |
| SEDA [8] | | 98.2 | 99.5 | 99.3 | 99.3 |
| SHOT [23] | | 98.4 | 98.0 | 98.9 | - |

Table 3: Target domain accuracy, reported as percentages

Our method, without extensive hyperparameter tuning, appears to be on par with the best state-of-the-art methods. We separate the methods that perform domain adaptation between those that aim at disentanglement or image-to-image translation (DSN, DiDA, SBADA-GAN, CyCADA, ACAL), and those that rely on other principles such as features or domain statistics alignment (DANN, ADDA, DRCN, DWT, SEDA, SHOT). DiCyR fundamentally belongs to the first group and outperforms all methods therein, on all benchmarks. We underline that this is probably the key result of this contribution since DiCyR is primarily a disentanglement method (which we apply to domain adaptation, among other tasks).

DiCyR is only slightly outmatched by DWT and SEDA on the MNIST↔USPS and by SEDA and SHOT in the SVHN→MNIST benchmarks. The variation on batch normalization introduced by DWT and the mean teacher semi-supervised learning model [39] used by SEDA are orthogonal to our contribution and could be combined to DiCyR in order to improve its performance.

[2]Comparisons might be inexact due to reproducibility concerns [29] and these figures mostly indicate which are the top competing methods.




(a) Swapping between SVHN (source) and MNIST (target)

(b) Swapping between Syn-Signs (source) and GTSRB (target)

Figure 5: Cross-domain swapping

DiCyR uses GRLs to ensure that no information is shared between $\mathcal{T}$ and $\mathcal{S}$. One might object that condition C3 was expressed in terms of mutual information. Thus, DiCyR only indirectly implements this condition using GRLs. An alternative could be to use an estimator of the mutual information, such as proposed by [1], to directly minimize it (and thus avoid the adversarial setting altogether). Such an approach was explored in the work of [33] to disentangle representations between pairs of images, and constitutes a promising perspective of research.

As in Section 5.1, we evaluate qualitatively the effectiveness of disentanglement, especially in the target domain, and produce visualizations of cross-domain style and task swapping. Here, we combine one domain's task information with the other domain's styles to reconstruct the images of Figure 5. The most important finding is that the style information was correctly disentangled from the task-related information in the target domain without the use of any label. Specifically, the rows in these figures show that the class information is preserved when a new style is applied, while the columns illustrate the efficient style transfer allowed by disentanglement.

A desirable property of the task-related encoding is its domain invariance. To evaluate this aspect, we built a t-SNE representation [14] of the task-related features, in order to verify their alignment between domains. Figure 6 demonstrate this property.

Finally, directly computing the distances on the task-related features in $\mathcal{L}_{domain_cyclic}$ often leads to unstable results. As hinted in Section 4, using instead a task oriented loss $\mathcal{L}_{domain_cyclic} = \|c(\tau_s) - y\|_2 + \|c(\tau_t) - c(\tau_{st})\|_2$ stabilizes training and improves the target domain accuracy. Training $c$ with cross-domain projections from the source domain and the corresponding labels improves its generalization to the target domain and forces the encoder to produce task-related features common to both domains. To illustrate this property, consider the following example. In one domain, the digit "7" is written with a middle bar, while in the other it has none. This domain-specific middle bar feature should not be expressed in $\mathcal{T}$; it should be considered as a task-orthogonal style feature. Thus using $c$'s predictions within the domain cyclic loss, instead of distances in $\mathcal{T}$, prevents the encoder from representing the domain-specific features in $\mathcal{T}$ and encourages their embedding in $\mathcal{S}$.

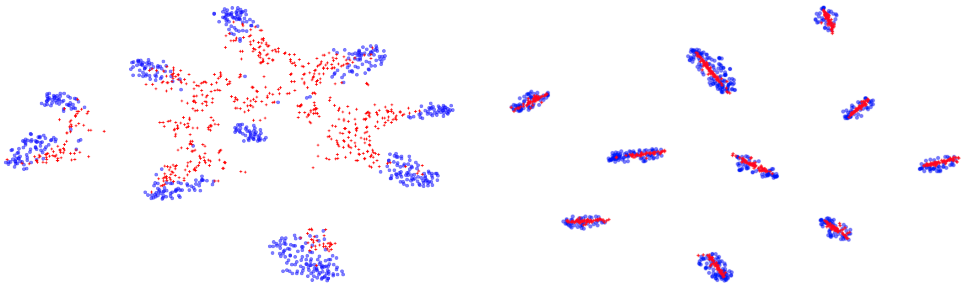


Figure 6: t-SNE on task-related features. SVHN (blue) → MNIST (red). Left: source only. Right: DiCyR

## 6 Conclusion

In this work, we introduced a new disentanglement method, called DiCyR, to separate task-related and task-orthogonal style information into different representations in the context of unsupervised domain adaptation. This method also provides a simple and efficient way to obtain disentangled representations for supervised learning problems. Its main features are its overall simplicity, the use of intra-domain and cross-domain cyclic reconstruction, and information separation through Gradient Reversal Layers. The design of this method stems from a formal definition of disentanglement for domain adaptation which, to the best of our knowledge, is new. Empirical evaluation shows that DiCyR allows for efficient disentanglement, as demonstrated on both information retrieval and domain adaptation tasks where it is competitive with state-of-the-art methods. Moreover, it is the only method that explicitly aims at disentanglement in the target domain, where no label information is available.

## Acknowledgment

The authors acknowledge the support of the DEEL project, the funding of the AI Interdisciplinary Institute ANITI funding, through the French "Investing for the Future – PIA3" program under grant agreement ANR-19-PI3A-0004. They would also like to thank Dennis Wilson and Thomas Oberlin for fruitful discussions and comments on the paper.

## References

[1] Belghazi, M.I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., Hjelm, D.: Mutual information neural estimation. In: 35th International Conference on Machine Learning. pp. 531–540 (2018)

[2] Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: Advances in neural information processing systems. pp. 343–351 (2016)

[3] Burgess, C., Kim, H.: 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/ (2018)

[4] Cai, R., Li, Z., Wei, P., Qiao, J., Zhang, K., Hao, Z.: Learning disentangled semantic representation for domain adaptation. In: International Joint Conference on Artificial Intelligence. p. 2060 (2019)

[5] Cao, J., Katzir, O., Jiang, P., Lischinski, D., Cohen-Or, D., Tu, C., Li, Y.: Dida: Disentangled synthesis for domain adaptation. arXiv pp. arXiv–1805 (2018)

[6] Cariucci, F.M., Porzi, L., Caputo, B., Ricci, E., Bulo, S.R.: Autodial: Automatic domain alignment layers. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 5077–5085. IEEE (2017)

[7] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems (2016)

[8] French, G., Mackiewicz, M., Fisher, M.: Self-ensembling for visual domain adaptation. In: International Conference on Learning Representations (2018)

[9] Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International conference on machine learning (2015)

[10] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research **17**(1), 2096–2030 (2016)

[11] Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: European Conference on Computer Vision. pp. 597–613 (2016)

[12] Gonzalez-Garcia, A., Van De Weijer, J., Bengio, Y.: Image-to-image translation for cross-domain disentanglement. In: Advances in neural information processing systems. pp. 1287–1298 (2018)

[13] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: Beta-VAE: Learning basic visual concepts with a constrained variational framework. In: Int. Conference on Learning Representations (2017)

[14] Hinton, G.E., Roweis, S.T.: Stochastic neighbor embedding. In: Advances in neural information processing systems. pp. 857–864 (2003)

[15] Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: Cycada: Cycle-consistent adversarial domain adaptation. In: International Conference on Machine Learning (2018)

[16] Hosseini-Asl, E., Zhou, Y., Xiong, C., Socher, R.: Augmented cyclic adversarial learning for low resource domain adaptation. In: International Conference on Learning Representations (2019)

[17] Hull, J.J.: A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(5), 550–554 (1994)

[18] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)

[19] Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: 5th International Conference on Learning Representations (2017)

[20] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[21] Li, H., Wan, R., Wang, S., Kot, A.C.: Unsupervised domain adaptation in the wild via disentangling representation learning. International Journal of Computer Vision pp. 1–17 (2020)

[22] Li, Y., Wang, N., Shi, J., Liu, J., Hou, X.: Revisiting batch normalization for practical domain adaptation. In: 5th International Conference on Learning Representations (2017)

[23] Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. arXiv preprint arXiv:2002.08546 (2020)

[24] Liu, A.H., Liu, Y.C., Yeh, Y.Y., Wang, Y.C.F.: A unified feature disentangler for multi-domain image translation and manipulation. In: Advances in neural information processing systems. pp. 2590–2599 (2018)

[25] Liu, Y.C., Yeh, Y.Y., Fu, T.C., Wang, S.D., Chiu, W.C., Wang, Y.C.F.: Detach and adapt: Learning cross-domain disentangled deep representation. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (2018)

[26] Mathieu, M.F., Zhao, J.J., Zhao, J., Ramesh, A., Sprechmann, P., LeCun, Y.: Disentangling factors of variation in deep representation using adversarial training. In: Advances in neural information processing systems. pp. 5040–5048 (2016)

[27] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011 (2011)

[28] Peng, X., Huang, Z., Sun, X., Saenko, K.: Domain agnostic learning with disentangled representations. In: International Conference on Machine Learning. pp. 5102–5112 (2019)

[29] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., Larochelle, H.: Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). arXiv preprint arXiv:2003.12206 (2020)

[30] Rifai, S., Bengio, Y., Courville, A., Vincent, P., Mirza, M.: Disentangling factors of variation for facial expression recognition. In: European Conference on Computer Vision. pp. 808–822 (2012)

[31] Roy, S., Siarohin, A., Sangineto, E., Bulo, S.R., Sebe, N., Ricci, E.: Unsupervised domain adaptation using feature-whitening and consensus loss. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 9471–9480 (2019)

[32] Russo, P., Carlucci, F.M., Tommasi, T., Caputo, B.: From source to target and back: symmetric bi-directional adaptive GAN. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 8099–8108 (2018)

[33] Sanchez, E.H., Serrurier, M., Ortner, M.: Learning disentangled representations via mutual information estimation. arXiv preprint arXiv:1912.03915 (2019)

[34] Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)

[35] Shen, J., Qu, Y., Zhang, W., Yu, Y.: Wasserstein distance guided representation learning for domain adaptation. In: AAAI (2018)

[36] Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: a multi-class classification competition. In: International Joint Conference on Neural Networks. pp. 1453–1460 (2011)

[37] Sun, B., Feng, J., Saenko, K.: Correlation alignment for unsupervised domain adaptation. In: Domain Adaptation in Computer Vision Applications, pp. 153–171. Springer (2017)

[38] Takahashi, R., Hashimoto, A., Sonogashira, M., Iiyama, M.: Partially-shared variational auto-encoders for unsupervised domain adaptation with target shift. arXiv preprint arXiv:2001.07895 (2020)

[39] Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Advances in neural information processing systems. pp. 1195–1204 (2017)

[40] Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. Neural computation **12**(6), 1247–1283 (2000)

[41] Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: IEEE conference on computer vision and pattern recognition (2017)

[42] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE international conference on computer vision. pp. 2223–2232 (2017)

[43] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)

[44] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)

# A Cross-domain disentanglement visualizations

Figures 7 and 8 report extra cross-domain visualizations similar to those in Figure 5.

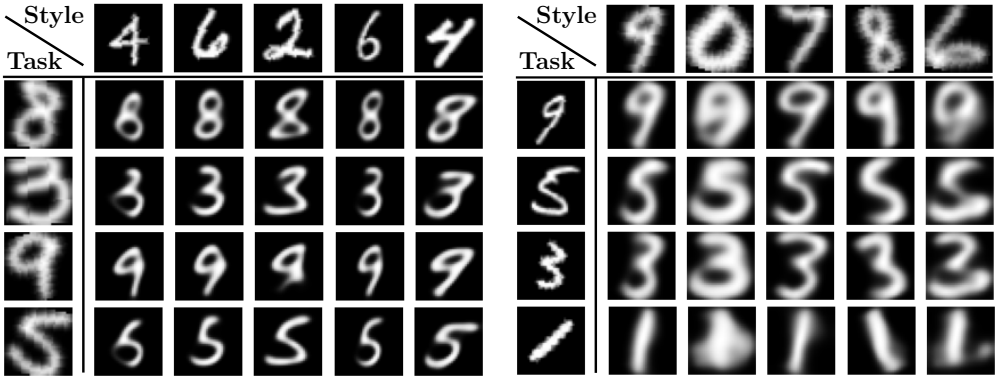



Figure 7: Cross-domain swapping between USPS (source) and MNIST (target)

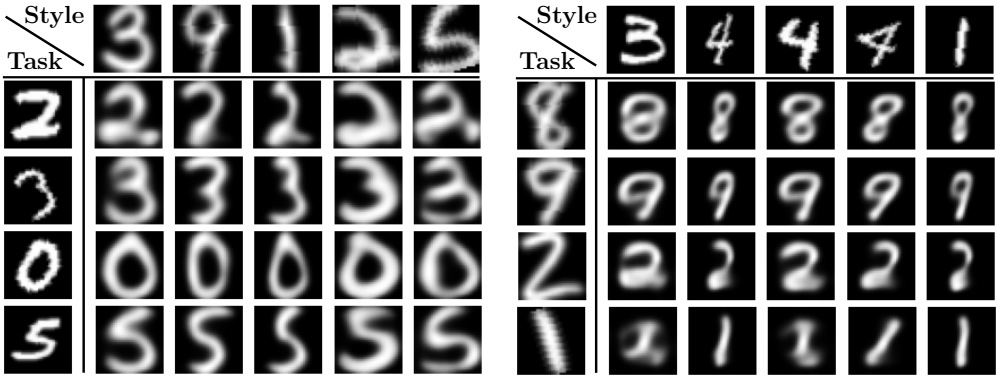



Figure 8: Cross-domain swapping between MNIST (source) and USPS (target)

# B Network architecture and training hyper-parameters

In line with the Machine Learning Reproducibility Checklist [29], we provide all details about network architectures, training hyperparameters and computing resources to permit easy reproduction of our results. The next paragraphs detail the network architectures used in the experiments of Section 5. It should be noted that neither these architectures, nor the associated hyper-parameters have been extensively and finely tuned to their respective tasks, as the goal of this contribution was to provide a generic, robust method. Thus, it is likely that performance gains can still be obtained on this front.

## B.1 Single domain supervised disentanglement experiments

This section and Table 5 describe the network architecture and the hyper-parameters used in the experiments of Section 5.1. The encoder $\Pi$ is composed of shared layers, followed by the specific single-layer task-related and style encodings $\Pi_\tau$ and $\Pi_\sigma$. We used the exact same network architectures for both the 3D shapes and SVHN datasets, the only difference being the dimension of the embeddings $\mathcal{T}$ and $\mathcal{S}$.

In all experiments, we applied a coefficient $\beta_{reco} = 5$ to $\mathcal{L}_{reco}$ and $\beta_{cyclic} = 0.1$ to $\mathcal{L}_{cyclic}$ in the global loss. We also use a $\beta_{info}$ on $\mathcal{L}_{info}$; this coefficient increases linearly from $10^{-2}$ to 10 over the first 10 epochs and remains at 10 afterwards (see Appendix C for a discussion on this coefficient). Convergence was reached within 50 epochs. We used Adam [43] as an optimizer with a learning rate $lr = 5 \cdot 10^{-4}$ for the first 30 epochs and $lr = 5 \cdot 10^{-5}$ for the last 20 epochs.

| Experiment | Batch size | Epochs | Repetitions |
|---|---|---|---|
| 5.1 SVHN | 64 | 50 (35s/epoch) | 50 |
| 5.1 3D Shapes | 64 | 50 (15s/epoch) | 5 |
| 5.2 MNIST→USPS | 128 | 150 (11s/epoch) | 20 |
| 5.2 USPS→MNIST | 128 | 150 (11s/epoch) | 20 |
| 5.2 MNIST→SVHN | 128 | 50 (40s/epoch) | 50 |
| 5.2 SVHN→MNIST | 64 | 50 (45s/epoch) | 50 |
| 5.2 Syn-Signs→GTSRB | 64 | 150 (65s/epoch) | 10 |

Table 4: Experimental setup

### B.2 Unsupervised domain adaptation experiments

This section and Tables 6, 7, and 8 describe the network architecture and the hyper-parameters used in the experiments of Section 5.2. The encoder $\Pi$ is composed of shared layers, followed by the specific task-related and style encodings. Those final layers are denoted $\Pi_\tau$ and $\Pi_\sigma$ in the tables below. For the sake of implementation simplicity, we chose to project samples from the source domain and samples from the target domain into two separate style embeddings (one for each domain). Thus $\Pi_\sigma$ is actually duplicated in two heads $\Pi_{\sigma,s}$ and $\Pi_{\sigma,t}$ with the same structure and output space. In all experiments, in the global loss, we applied a coefficient $\beta_{cyclic} = 0.1$ to $\mathcal{L}_{cyclic}$ and $\beta_{domain_cyclic}$ to $\mathcal{L}_{domain_cyclic}$, with $\beta_{domain_cyclic}$ increasing linearly from 0 to 10 during the 10 first epochs and remaining at 10 afterwards (see Section C for a discussion). Convergence was reached within 50 epochs (generally within 30 epochs). We used Adam [43] as an optimizer with a learning rate $lr = 5 \cdot 10^{-4}$ for the first 30 epochs and $lr = 5 \cdot 10^{-5}$ for the last 20 epochs.

### B.3 Computing resources and code release

All the experiments from section 5 were run on a Google Cloud Platform n1-standard-8 virtual machine (8 virtual cores, 30Go RAM, Nvidia P100 GPU). The code corresponding to the experiments, a list of dependencies, and pre-trained models are available at `https://github.com/SuReLI/DiCyR_code`. Details about each experiment are reported in Table 4.

## C Discussion on the $\beta_{info}$ and $\beta_{domain_cyclic}$ coefficients schedule

Although the schedule on $\beta_{info}$ (single domain case) and $\beta_{domain_cyclic}$ (domain adaptation case) is not absolutely necessary, we found out it helped the overall convergence. These coefficients gradually increase the weight of the information disentanglement objective and the cross-domain reconstruction objective. This assigns more importance to learning a good predictor $c \circ \Pi$ during early stages. From this perspective, gradually increasing $\beta_{info}$ can be seen as gradually removing task-useless information from $\mathcal{T}$ and transferring it to $\mathcal{S}$. Similarly, increasing $\beta_{domain_cyclic}$ corresponds to letting the network discover disentangled representations before aligning them across domains.

As previously mentioned, our goal in this study was to provide a robust disentanglement method that permits domain adaptation. Therefore, no complete hyper-parameter study and tuning was performed and these findings are thus reported as such and might be incomplete. Refining the understanding of the influence of the different $\beta$ coefficients is closer to the problem of meta-learning and is beyond the scope of this paper.

## D Influence of batch normalization and dropout on DiCyR

Batch normalization [18] is an efficient way to reduce the discrepancy between the source and target distributions statistics. We noticed that, for the specific SVHN $\rightarrow$ MNIST setting, using instance normalization [44] slightly improves the target domain accuracy. Normalizing across chanels, the instance normalization layers helps the networks to be agnostic to the image contrast which is particularly strong in MNIST. We also noticed that using a large dropout in the sub-network $c$, and small embedding dimensions for $\Pi$'s ouputs improves both the disentanglement quality and the target domain accuracy. We conjecture that the information bottleneck induced forces the task-related representation to be as concise as possible and thus encourages disentanglement.

| Network | Architecture |
|---|---|
| $\Pi$ | • Conv2D(filters=32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=64 for SVHN, kernel=$3 \times 3$, stride=1, padding=1, activation=ReLU)<br>• Dense(nb_neurons=1024, activation=ReLU) |
| $\Pi_\tau$ | • Dense(nb_neurons=150, activation=ReLU) |
| $\Pi_\sigma$ | • Dense(nb_neurons=150, activation=ReLU) |
| $\bar{\Pi}$ | • Dense(nb_neurons=1024, activation=ReLU)<br>• Dense(nb_neurons=8192, activation=ReLU)<br>• Conv2D(filters=64, kernel=$3 \times 3$, stride=1, padding=1, activation=ReLU)<br>• Upsample (scale_factor=2)<br>• Conv2D(filters=32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Upsample(scale_factor=2)<br>• Conv2D (filters=3, kernel=$5 \times 5$, stride=1, padding=2, activation=Sigmoid) |
| $c$ | • Dropout(p=0.2(3DShapes) or 0.55(SVHN))<br>• Dense(nb_neurons=nb_labels, activation=Softmax) |
| $r_\tau$ and $r_\sigma$ | • Gradient Reversal Layer<br>• Dense(nb_neurons=100, activation=ReLU)<br>• Dense(nb_neurons=100, activation=ReLU)<br>• Dense(nb_neurons=20(Shapes) or 150(SVHN), activation=Linear) |

Table 5: Supervised experiments architectures

| Network | Architecture |
|---|---|
| $\Pi$ | • Conv2D(filters=32, kernel=5×5, stride=1, padding=2, activation=Linear)<br>• Instance Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=32(SVHN→MNIST) or 64(MNIST→SVHN), kernel=$5 \times 5$, stride=1, padding=2, activation=Linear)<br>• Instance Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=32(SVHN→MNIST) or 128(MNIST→SVHN), kernel=$3 \times 3$, stride=1, padding=2, activation=Linear)<br>• Instance Normalization<br>• Dense(nb_neurons=1024, activation=ReLU) |
| $\Pi_\tau$ | • Dense(nb_neurons=75(SVHN→MNIST) or 200(MNIST→SVHN), activation=ReLU) |
| $\Pi_\sigma$ | • Dense(nb_neurons=75(SVHN→MNIST) or 200(MNIST→SVHN), activation=ReLU) |
| $\bar{\Pi}_s$ and $\bar{\Pi}_t$ | • Dense(nb_neurons=1024, activation=ReLU)<br>• Dense(nb_neurons=2048, activation=ReLU)<br>• Conv2D(filters=32, kernel=$3 \times 3$, stride=1, padding=1, activation=ReLU)<br>• Upsample (scale_factor=2)<br>• Conv2D(filters=32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Upsample(scale_factor=2)<br>• Conv2D (filters=3, kernel=$5 \times 5$, stride=1, padding=2, activation=Sigmoid) |
| $c$ | • Dropout(p=0.55)<br>• Dense(nb_neurons=10, activation=Softmax) |
| $r_\tau$ and $r_\sigma$ | • Gradient Reversal Layer<br>• Dense(nb_neurons=100, activation=ReLU)<br>• Dense(nb_neurons=75(SVHN→MNIST) or 200(MNIST→SVHN), activation=Linear) |

Table 6: SVHN↔MNIST networks architectures

| Network | Architecture |
|---|---|
| $\Pi$ | • Conv2D(filters=50, kernel=5×5, stride=1, padding=2, activation=ReLU)<br>• Batch Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=75, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Batch Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=100, kernel=$3 \times 3$, stride=1, padding=2, activation=Linear)<br>• Batch Normalization<br>• Dense(nb_neurons=1024, activation=ReLU) |
| $\Pi_\tau$ | • Dense(nb_neurons=150, activation=ReLU) |
| $\Pi_\sigma$ | • Dense(nb_neurons=150, activation=ReLU) |
| $\bar{\Pi}_s$ and $\bar{\Pi}_t$ | • Dense(nb_neurons=1024, activation=ReLU)<br>• Dense(nb_neurons=6400, activation=ReLU)<br>• Conv2D(filters=100, kernel=$3 \times 3$, stride=1, padding=1, activation=ReLU)<br>• Upsample (scale_factor=2)<br>• Conv2D(filters=50, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Upsample(scale_factor=2)<br>• Conv2D (filters=3, kernel=$5 \times 5$, stride=1, padding=2, activation=Sigmoid) |
| $c$ | • Dropout(p=0.55)<br>• Dense(nb_neurons=10, activation=Softmax) |
| $r_\tau$ and $r_\sigma$ | • Gradient Reversal Layer<br>• Dense(nb_neurons=100, activation=ReLU)<br>• Dense(nb_neurons=150, activation=Linear) |

Table 7: MNIST↔USPS networks architectures

| Network | Architecture |
|---|---|
| $\Pi$ | • Conv2D(filters=32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Instance Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Instance Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=32, kernel=$3 \times 3$, stride=1, padding=2, activation=Linear)<br>• Instance Normalization<br>• Max Pooling(filters=$2 \times 2$, stride=2)<br>• Conv2D(filters=32, kernel=$3 \times 3$, stride=1, padding=2, activation=Linear)<br>• Instance Normalization<br>• Dense(nb_neurons=1024, activation=ReLU) |
| $\Pi_\tau$ | • Dense(nb_neurons=150, activation=ReLU) |
| $\Pi_\sigma$ | • Dense(nb_neurons=150, activation=ReLU) |
| $\bar{\Pi}_s$ and $\bar{\Pi}_t$ | • Dense(nb_neurons=1024, activation=ReLU)<br>• Dense(nb_neurons=1024, activation=ReLU)<br>• Conv2D(filters=32, kernel=$3 \times 3$, stride=1, padding=1, activation=ReLU)<br>• Upsample (scale_factor=2)<br>• Conv2D(filters=32, kernel=$3 \times 3$, stride=1, padding=1, activation=ReLU)<br>• Upsample (scale_factor=2)<br>• Conv2D(filters=32, kernel=$5 \times 5$, stride=1, padding=2, activation=ReLU)<br>• Upsample(scale_factor=2)<br>• Conv2D (filters=3, kernel=$5 \times 5$, stride=1, padding=2, activation=Sigmoid) |
| $c$ | • Dropout(p=0.55)<br>• Dense(nb_neurons=43, activation=Softmax) |
| $r_\tau$ and $r_\sigma$ | • Gradient Reversal Layer<br>• Dense(nb_neurons=100, activation=ReLU)<br>• Dense(nb_neurons=150, activation=Linear) |

Table 8: Syn-Signs→GTSRB networks architectures