Unsupervised domain adaptation with non-stochastic missing data

Matthieu Kirchmeyer, Patrick Gallinari, Alain Rakotomamonjy, Amin Mantrach

Unsupervised domain adaptation with non-stochastic missing data

Matthieu Kirchmeyer[1,2], Patrick Gallinari[1,2], Alain Rakotomamonjy[2,3], and Amin Mantrach[4]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[2] Criteo AI Lab, Paris, France
[3] Université de Rouen, LITIS, France
[4] Amazon, Luxembourg

Abstract. We consider unsupervised domain adaptation (UDA) for classification problems in the presence of missing data in the unlabelled target domain. More precisely, motivated by practical applications, we analyze situations where distribution shift exists between domains and where some components are systematically absent on the target domain without available supervision for imputing the missing target components. We propose a generative approach for imputation. Imputation is performed in a domain-invariant latent space and leverages indirect supervision from a complete source domain. We introduce a single model performing joint adaptation, imputation and classification which, under our assumptions, minimizes an upper bound of its target generalization error and performs well under various representative divergence families ($\mathscr{H}$-divergence, Optimal Transport). Moreover, we compare the target error of our Adaptation-imputation framework and the "ideal" target error of a UDA classifier without missing target components. Our model is further improved with self-training, to bring the learned source and target class posterior distributions closer. We perform experiments on three families of datasets of different modalities: a classical digit classification benchmark, the Amazon product reviews dataset both commonly used in UDA and real-world digital advertising datasets. We show the benefits of jointly performing adaptation, classification and imputation on these datasets.

1 Introduction

Motivated by real applications, we consider a classification problem where: (1) a source and target domain are available with observed source labels and missing target labels, (2) a distribution shift exists between source and target on joint distributions in the input and label space, (3) source input data are fully available while target data have missing input components, which cannot be measured on this domain and (4) there is no possible supervision in the target domain for imputation, thus requiring indirect supervision from the source domain. Furthermore, unobserved features contain complementary information not present in the observed ones so that the former cannot be inferred directly from the latter. (1) and (2) correspond to the classical setting of unsupervised domain adaptation, (3) corresponds to a missing data imputation problem on the target with the difficulty (4). [27, 36] distinguish three categories of missing data problems

based on a missingness mechanism denoted ϕ. Let $\mathbf{m}$ define a pattern of missing data, ϕ defines the conditional distribution $p_\phi(\mathbf{m}|\mathbf{x})$ where $\mathbf{x}$ represents a sample. Missing Completely at Random (MCAR) problems verify $\forall \mathbf{x}, p_\phi(\mathbf{m}|\mathbf{x}) = p_\phi(\mathbf{m})$, Missing At Random, $\forall \mathbf{x}, p_\phi(\mathbf{m}|\mathbf{x}) = p_\phi\left(\mathbf{m}|\mathbf{x}^{\text{obs}}\right)$ with $\mathbf{x}^{\text{obs}}$ the observed feature and Missing Not At Random covers all the other cases. The key idea behind Rubin's theory is that $\mathbf{m}$ is a random variable with a probability distribution and specific imputation approaches were developed for each missingness setting. We consider the setting where target data have systematically missing input components. This corresponds to MCAR with the additional difficulty that $\mathbf{m}$ is deterministic, not stochastic. This problem is more difficult than classical MCAR as neither classical maximum likelihood solutions nor stochasticity in missing features can be used to reconstruct the missing information. While general adaptation and imputation problems were considered independently, there are several instances where they occur simultaneously. This has seldom been analyzed and only for specific cases. We propose a principled solution to this problem under non-stochastic missingness and present practical situations where this occurs.

There are many problems where specific features in collected data may be systematically absent on a domain. In the literature, this setting is mostly considered when dealing with data with multiple modalities. For example, in disease diagnosis in medical imaging [8], for some collected dataset, several modalities are present while they are absent on other datasets for which the corresponding equipment was unavailable. In multi-lingual text classification [16] some collections may be available only for a limited set of languages. Similar considerations hold for recommendation in advertising [42] and object recognition with multi-sensor data [39]. The situation which initially motivated our investigation, is the *prospecting* setting in computational advertising. The classical framework for ads on the internet is *retargeting*: users have already interacted with a set of merchant sites and they are targeted when they come back on one of these sites. Retargeting makes use of global user statistics collected on the whole set of merchant sites and of statistics from the specific site the user is browsing. Prospecting aims at targeting a user that visits a site for the first time [1]; while for such a user, features from his general behavior are available, there is no user information for the targeted site and the corresponding features are absent. The second issue considered is the distribution shift between domains. For instance, data may be collected on different devices as in medical imaging [9] or background noise may affect each domain differently. This issue has given rise to the literature of Domain Adaptation when aiming at transferring knowledge from one domain to the other [33]. The ads case described above is subject to both missing data for prospecting users and distribution shift between retargeting and prospecting users as detailed in Section 6.3.

We propose a model addressing the Adaptation-imputation problem defined by (1) to (4), which learns to perform imputation for the target domain with a conditional generative model. Imputation makes use of indirect supervision from the complete source domain. This allows us to handle non-stochastic missing data, while satisfying the constraints related to adaptation in a latent space and to classification. The imputation process plays an important role, providing us with information about the missing target data while contributing to the alignment and the reconstruction losses. Extensive empirical evidence on handwritten digits, Amazon product reviews and Click-Through-Rate

(CTR) prediction domain adaptation problems illustrate the benefit of our model. The original contributions are the following:

- We propose a new end-to-end model for handling non-stochastic missing data with domain adaptation. It generates relevant missing information in the latent space conditionally on available information while aligning latent source and target marginals and classifying labelled instances. The joint missing-data and adaptation problem has been seldom considered and never in our context.
- We derive an adaptation and an imputation upper bounds. The first one upper bounds our model's target generalization error and is minimized explicitly by our training objective. The second one upper bounds an ideal target error corresponding to an UDA problem without missing features in the target domain.
- We improve this model by bringing the source and target class posteriors closer to one another with self-training; this is a useful heuristic when class posteriors mismatch.
- We evaluate the model on academic benchmarks and on challenging real-world advertising data and illustrate on these datasets that conditional generative models improve regression-based approaches seen in the literature.

2 Related work

Our problem is related to generic ML topics usually addressed separately e.g. domain adaptation and imputation and in an extend other secondary topics. We provide a brief overview of related contributions in the main topics below and in other minor topics in Appendix A.

Unsupervised Domain Adaptation A number of learning methods approach UDA by weighting individual observations during training [11, 26]. Recent deep learning methods align the source and target distributions by embedding them in a joint latent space. There are two main directions for learning joint embeddings. One is based on adversarial training, making use of GAN extensions; the seminal work of [18] learns to map source and target domains onto a common latent space by optimizing jointly 1) an approximation of the $\mathscr{H}$-divergence between the source and target embeddings via adversarial training, 2) a classification term on source data embeddings. This work has been extended in several papers [28, 40]. The other direction directly exploits explicit distance measures between source and target representations using Integral Probability Metrics such Maximum Mean Discrepancy [29] or Wasserstein distance [14, 38]. These work consider full input data on both domains.

Imputation Data imputation is addressed by several methods [27, 41]. Most approaches consider a supervised setting where (1) paired or unpaired complete and incomplete data are available, (2) missingness corresponds to a stochastic process (e.g. a mask distribution for tabular data) and (3) imputation is performed in the original feature space. This is different from our setting when one considers (1) reconstruction in a latent space, (2) imputation for a classification task, (3) no direct supervision and (4) fixed missingness which prevents us from exploiting the statistics from different incomplete

samples leading to a much more complex problem. Recently, generative models were adapted for data imputation, e.g. [46] and [30] for GANs and VAEs respectively. The general approach with generative models is to learn a distribution over imputed data which is similar to the one of plain data. This comes in many different instances and usually, generative training alone is not sufficient; additional loss terms are often used. In paired problems where each missing datum is associated to a plain version, a reconstruction term imposed by a MSE contraint is added [21]; in unpaired problems a cycle-consistency loss is imposed [50]. [25, 32] are among the very few approaches addressing unsupervised imputation in which full instances are never directly used. Both extend AmbientGAN [7] and consider stochastic missingness. Our imputation problem is closer to the one addressed in some forms of inpainting [34], missing view imputation [16] or multi-modality missing data [8]. These approaches are fully supervised. The latter considers, as we do, imputation when one modality is systematically absent, but on one domain only, i.e. without adaptation. [15, 44, 45] are the only papers we are aware of that consider imputation as we do. [15] considers low-rank constraints and dictionary learning to guide transfer and was not used here as a baseline due to a high complexity that prevents large-scale experiments. [44, 45] are close to our work but assume that missing data can be reconstructed from the observed one through regression. In our setting, this is not possible: given the observed features, there are multiple possible imputations for the missing features; regression is thus meaningless and one has to learn their distribution or at least some modes. This motivates learning a generative model. Moreover, in [44,45] classification occurs as a downstream task whereas our approach is end-to-end for classification, adaptation and imputation. Finally, our method is theoretically justified and addresses a challenging large size application motivated by a concrete real-world problem never handled before.

Cold-start Cold-start occurs when making predictions or recommendations when data from the item or user of interest is not available or was not observed in the training set. The standard hypothesis is i.i.d. data coming from the same domain. In recommender systems, several papers address cold-start and leverage auxiliary information about users or items e.g. user attributes, profile, social context or cross-domain information [4, 37]. Cold-start is related to zero-shot learning with unobserved data where usual solutions learn a representation space using auxiliary knowledge e.g. grounded word embeddings with visual context [48]. As for our problem, cold-start deals with non-stochastic missing data, but usually considers only one domain while we deal with distribution shift as well through adaptation.

3 Problem definition

Notations $\mathscr{X}, \mathscr{Y}$ denote the input and label space. We use X, Y to denote random variables with values in $\mathscr{X}, \mathscr{Y}$. A domain D is defined by a distribution $p_D(X)$ on $\mathscr{X}$ and a deterministic labeling function $f_D : \mathscr{X} \to \{1, ..., K\}$ where K is the number of classes. D will refer to either the source S or target T domain. Data from domain D is $(\mathbf{x}_D, y_D) \in \mathbb{R}^n \times \{1, ..., K\}$ where n is the dimension of the input space, sampled from the domain's joint distribution $p_D(X, Y)$. In the UDA setting, target labels are unknown.

We consider that $\mathbf{x}_D$ has two components, $\mathbf{x}_D = (\mathbf{x_{D_1}}, \mathbf{x_{D_2}})$; X_1, X_2 refer to each component with values in $\mathscr{X}_1, \mathscr{X}_2$. Given input $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{m} \in \{0,1\}^n$ is a binary mask indicating which entries of $\mathbf{x}$ are missing (1 for missing and 0 for observed). We define $\mathscr{Z} = \mathscr{Z}_1 \times \mathscr{Z}_2$ as the representation space built with a feature extractor. Assuming both components $(\mathbf{x_{D_1}}, \mathbf{x_{D_2}})$ are observed, we define g as

$$\begin{aligned} g : \mathscr{X}_1 \times \mathscr{X}_2 &\to \mathscr{Z}_1 \times \mathscr{Z}_2 \\ (\mathbf{x_{D_1}}, \mathbf{x_{D_2}}) &\mapsto (g_1(\mathbf{x_{D_1}}), g_2(\mathbf{x_{D_2}})) \end{aligned} \tag{1}$$

where $\mathbf{z_{D_1}} = g_1(\mathbf{x_{D_1}})$, $\mathbf{z_{D_2}} = g_2(\mathbf{x_{D_2}})$ with Z_1, Z_2, the corresponding random variables. This is illustrated in Figure 1 (b) using examples from the `digits` dataset. While X_2 is available on the source domain, it is absent on the target domain. As detailed in Section 4.2, we will learn to perform imputation in the latent $\mathscr{Z}$ space via a generative network r operating on $\mathscr{Z}$. For this we will introduce a mapping $\widehat{g}$ as follows:

$$\begin{aligned} \widehat{g} : \mathscr{X}_1 &\to \mathscr{Z}_1 \times \mathscr{Z}_2 \\ \mathbf{x_{D_1}} &\mapsto (g_1(\mathbf{x_{D_1}}), r \circ g_1(\mathbf{x_{D_1}})) \end{aligned} \tag{2}$$

where $g_1 : \mathscr{X}_1 \to \mathscr{Z}_1$, $r : \mathscr{Z}_1 \to \mathscr{Z}_2$ and $\widehat{\mathbf{z}}_{\mathbf{D_2}} = r \circ g_1(\mathbf{x_{D_1}})$. $\widehat{Z}_2$ is the corresponding random variable built from X_1 with $r \circ g_1$ with values in $\mathscr{Z}_2$ and $\widehat{Z} = (Z_1, \widehat{Z}_2)$. This is illustrated in Figure 1 (a). For reasons detailed later, this mapping $r \circ g_1(\cdot)$ will be used on both S and T.

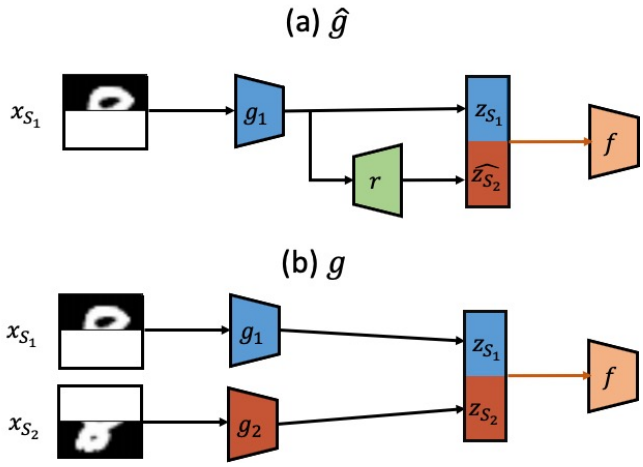


Fig. 1: Encoding of an input source digit $\mathbf{x_S} = (\mathbf{x_{S_1}}, \mathbf{x_{S_2}})$ with $\widehat{g}$ (a) and g (b). g_1 encodes the first part of the input $\mathbf{x_{S_1}}$ into $\mathbf{z_{S_1}}$. The second latent component is either built by encoding $\mathbf{x_{S_2}}$ with g_2 as $\mathbf{z_{S_2}}$ (b) or with reconstruction via $r \circ g_1$ as $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ (a). These latent components $\widehat{\mathbf{z}}_{\mathbf{S}}$ (a), $\mathbf{z_S}$ (b) are fed directly into a classifier f.

Assumptions Let us now introduce formally the different assumptions underlying our context and model. We address UDA with non-stochastic missing target features and aim at finding a single hypothesis $h_{\widehat{g}} : \mathscr{X} \to \{0, ..., K\}$ of the form $f \circ \widehat{g}$, with $\widehat{g}$ the feature extractor defined in (2) and f the classifier with low target risk. Since the problem is under-specified, one has to make assumptions to define it properly:

Assumption 1. *Labelled source data* $\mathbf{x_S}$ *are fully observed while unlabelled target data are partially observed with* $\mathbf{x_{T_2}}$ *missing. The missingness mechanism corresponds to Missing Completely At Random [27] on the target with fixed missingness pattern. Thus the distribution statistics of the missing data cannot be leveraged for imputation and we can only resort to indirect supervision with adaptation as in Section 4.2: we consider only statistics from the source to infer the imputation mechanism as later explained.*

Assumption 2. *The distribution of* $X_2|X_1$ *projected in the latent space with* g *from* (1)*,* $p_D(Z_2|Z_1)$*, is multi-modal and* Z_1 *and* Z_2 *are not statistically independent. This allows to impute* $\mathbf{z_{D_2}}$ *given* $\mathbf{z_{D_1}}$*. However, regression on* $\mathbf{z_{D_1}}$ *cannot recover all modes as MSE produces blurry reconstructions by averaging modes. For example, assuming the feature variables* Z_1, Z_2 *encode the contour of the top, respectively bottom of a digit, given the bottom contours of a digit, we can reconstruct several candidates of the top contours (the bottom half of 7 can either be reconstructed into 1 or 7; regression will lead to a blurry digit averaging these two modes). As mentioned, this uncertainty is present for all our datasets.*

Assumption 3. *The distribution of* $X_2|X_1$ *projected in the latent space with* g *in* (1) *is the same across domains i.e.* $p_S(Z_2|Z_1) = p_T(Z_2|Z_1)$*. This allows us to make use of the source domain information (with available supervision) to infer the target conditional distribution and recover the missing latent component useful for classification. For example, assuming the feature variables* Z_1, Z_2 *encode the contour of the top, respectively bottom of a digit,* $p(Z_2|Z_1)$ *is the distribution of the contours of the bottom of the digit given those of the top; it is reasonable to assume that this distribution is the same across domains.*

Assumption 4. *Covariate shift is valid in the latent space obtained with* $\widehat{g}$ *in* (2) *i.e.* $p_S(Y|\widehat{Z}) = p_T(Y|\widehat{Z})$ *while* $p_S(\widehat{Z}) \neq p_T(\widehat{Z})$*. Thus, we can find a classifier* $f \circ \widehat{g}$ *with low source and target error; this is a common assumption for standard UDA methods.*

4 Adaptation-Imputation model

As several generative approaches to UDA, we project source and target data onto a common latent space in which data distributions from the two domains should match and learn a classifier using source labels. Our novelty is to offer a solution to deal with datasets with systematically missing data in the target domain. Our model, denoted *Adaptation-Imputation*, is trained to perform three operations jointly: imputation of missing information, alignment of the distributions of both domains and classification of source instances. The three operations are performed in a joint embedding space and all components are trained together with shared parameters. The term imputation is used here in a specific sense: our goal is to recover information from $\mathbf{x_{T_2}}$ that will

be useful for adaptation and for the target data classification objective and not to reconstruct the whole missing $\mathbf{x}_{\mathbf{T}_2}$. This is achieved via a generative model, which for a given datum in T and conditionally on the available information $\mathbf{x}_{\mathbf{T}_1}$, attempts to generate the missing information. Because $\mathbf{x}_{\mathbf{T}_2}$ is systematically missing for T (Assumption 1), there is no possible supervision with target samples; instead we use indirect supervision from source samples while transferring to the target. We consider two variants of the same model based on different divergence measures between distributions: the $\mathscr{H}$-divergence approximated through adversarial training (`ADV`) and the Wasserstein distance (`OT`) computed through the primal by finding a joint coupling matrix γ with linear programming [35]. Our two models can be seen respectively as extensions of DANN [18] and DeepJDOT [14] to the missing data problem. We only describe the `ADV` version in the main text, the extension to `OT` is detailed in Appendix B. Results for both models are in Section 6.

4.1 Inference

The latent space representations are denoted $\widehat{\mathbf{z}}_{\mathbf{D}} = (\mathbf{z}_{\mathbf{D}_1}, \widehat{\mathbf{z}}_{\mathbf{D}_2})$. $\mathbf{z}_{\mathbf{D}_1} = g_1(\mathbf{x}_{\mathbf{D}_1})$ is the mapping of the observed component $\mathbf{x}_{\mathbf{D}_1}$ onto the latent space and $\widehat{\mathbf{z}}_{\mathbf{D}_2} = r \circ g_1(\mathbf{x}_{\mathbf{D}_1})$ is the second component's latent representation generated conditionally on $\mathbf{x}_{\mathbf{D}_1}$ through generator r, as later described. At inference, given $\mathbf{x}_{\mathbf{T}_1}$, we generate $\widehat{\mathbf{z}}_{\mathbf{T}} = (\mathbf{z}_{\mathbf{T}_1}, \widehat{\mathbf{z}}_{\mathbf{T}_2})$ where $\widehat{\mathbf{z}}_{\mathbf{T}_2}$ encodes part of the missing information $\mathbf{x}_{\mathbf{T}_2}$ in $\mathbf{x}_{\mathbf{T}}$ (Figure 2 (b)). Finally $\widehat{\mathbf{z}}_{\mathbf{T}}$ is fed to the classifier f.

4.2 Training

For simplicity, we describe each component in turn but please note that they all interact and that their parameters are all optimized according to the three objectives mentioned above. The interaction is discussed after the description of each individual module. The model's components are illustrated in Figure 2 (a).

Adaptation Adaptation aligns the distributions of $\widehat{\mathbf{z}}_{\mathbf{S}}$ and $\widehat{\mathbf{z}}_{\mathbf{T}}$ in the latent space. For `ADV`, alignment is performed via an adversarial loss operating on the latent representations

$$L_1 = \mathbb{E}_{\mathbf{x_S} \sim p_S(X)} \log D_1(\widehat{\mathbf{z}}_{\mathbf{S}}) + \mathbb{E}_{\mathbf{x_T} \sim p_T(X)} \log(1 - D_1(\widehat{\mathbf{z}}_{\mathbf{T}})) \tag{3}$$

where $D_1(\widehat{\mathbf{z}})$ represents the probability that $\widehat{\mathbf{z}}$ comes from S rather than T.

Imputation Imputation generates an encoding $\widehat{\mathbf{z}}_{\mathbf{T}_2}$ for the missing information, conditioned on the available $\mathbf{x}_{\mathbf{T}_1}$ thanks to a generative model r. Since we never have access to $\mathbf{x}_{\mathbf{T}_2}$, we develop a distant learning strategy: we learn imputation on S through $\widehat{\mathbf{z}}_{\mathbf{S}_2} = r \circ g_1(\mathbf{x}_{\mathbf{S}_1})$ (Figure 2) and then transfer to the target domain ($\widehat{\mathbf{z}}_{\mathbf{T}_2}$ on the figure) via adaptation. For that we perform two operations in parallel. First, we align the distributions of $\widehat{\mathbf{z}}_{\mathbf{S}_2}$ and $\mathbf{z}_{\mathbf{S}_2} = g_2(\mathbf{x}_{\mathbf{S}_2})$ which is the encoding of $\mathbf{x}_{\mathbf{S}_2}$, using an adversarial loss and discriminator D_2 (L_{ADV} on Figure 2). As alignment acts globally on distributions we have no guarantee that $\widehat{\mathbf{z}}_{\mathbf{S}_2}$ will be associated to the corresponding $\mathbf{z}_{\mathbf{S}_1}$. We then enforce a one-to-one relationship by associating a $\widehat{\mathbf{z}}_{\mathbf{S}_2}$ to its specific $\mathbf{z}_{\mathbf{S}_1}$. For that, we

use a reconstruction term, the MSE distance between $\mathbf{z_{S_2}}$ and $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ (L_{MSE} on Figure 2). This guarantees that the imputed $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ truly represents information in $\mathbf{z_{S_2}}$. The learned mappings are used to perform imputation on the target data $\widehat{\mathbf{z}}_{\mathbf{T_2}} = r \circ g_1(\mathbf{x_{T_1}})$. The imputation loss L_2 has thus two terms: an adversarial term L_{ADV} for aligning $\mathbf{z_{S_2}}$ and $\widehat{\mathbf{z}}_{\mathbf{S_2}}$; and a reconstruction term L_{MSE}:

$$L_2 = L_{ADV} + \lambda_{MSE} \times L_{MSE} \tag{4}$$

$$L_{ADV} = \mathbb{E}_{\mathbf{x_{S_2}} \sim p_S(X_2)} \log D_2(\widehat{\mathbf{z}}_{\mathbf{S_2}}) + \mathbb{E}_{\mathbf{x_{S_1}} \sim p_S(X_1)} \log(1 - D_2(\mathbf{z_{S_2}})) \tag{5}$$

$$L_{MSE} = \mathbb{E}_{\mathbf{x_S} \sim p_S(X)} \left\| \mathbf{z_{S_2}} - \widehat{\mathbf{z}}_{\mathbf{S_2}} \right\|_2^2 \tag{6}$$

where λ_{MSE} weights the regression term over the generative term. Imputation and adaptation influence each other and both are also influenced by classification described below. The latter forces the generated $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ to contain information about $\mathbf{x_{S_2}}$ relevant for the classification task. This information is transferred via adaptation to the target when generating $\widehat{\mathbf{z}}_{\mathbf{T_2}}$.

Classification The last component is a classifier f, trained on source mappings $\widehat{\mathbf{z}}_{\mathbf{S}}$ as done in classic UDA. The corresponding loss, with L_{Disc} a cross-entropy loss, is

$$L_3 = \mathbb{E}_{(\mathbf{x_S}, y_S) \sim p_S(X,Y)} L_{Disc}(f(\widehat{\mathbf{z}}_{\mathbf{S}}), y_S) \tag{7}$$

Overall loss L is the weighted sum of the adaptation, imputation and classification losses

$$L = \lambda_1 \times L_1 + \lambda_2 \times L_2 + \lambda_3 \times L_3 \tag{8}$$

with $\lambda_1, \lambda_2, \lambda_3$ some hyperparameters and we solve

$$\min_{g_1, g_2, r, f} \max_{D_1, D_2} L \tag{9}$$

Interaction between the model's components Mappings g_1, g_2, r appear in the three terms of L, meaning that they should learn to perform the three tasks simultaneously. g_1 maps $\mathbf{x_{S_1}}$ and $\mathbf{x_{T_1}}$ onto the latent space, the embeddings being denoted respectively $\mathbf{z_{S_1}}$ and $\mathbf{z_{T_1}}$. r learns to generate missing information $\widehat{\mathbf{z}}_{\mathbf{D_2}}$ from $\mathbf{z_{D_1}}$. $\widehat{\mathbf{z}}_{\mathbf{D}}$ is generated to fulfill the classification objective. g_2 should fulfill the imputation objective while preserving part of the information present in $\mathbf{x_{S_2}}$. Our model uses a unique mapping g_1 for both S and T; compared to using separate mappings, this reduces the number of parameters and was found to perform as well.

Implementation For adversarial training, discriminators D_1 (adaptation) and D_2 (imputation) are implemented by binary classifiers. D_1 is trained to distinguish $\widehat{\mathbf{z}}_{\mathbf{S}}$ from $\widehat{\mathbf{z}}_{\mathbf{T}}$ mappings while D_2 is trained to separate imputed $\widehat{\mathbf{z}}_{\mathbf{S_2}}$, generated from $\mathbf{x_{S_1}}$, and $\mathbf{z_{S_2}}$, a direct embedding of $\mathbf{x_{S_2}}$. We use gradient reversal layers [18] for implementing the min-max condition on D_1 and D_2. To stabilize adversarial training, we update progressively λ_1, λ_2, respectively the hyperparameter for the adaptation loss L_1 and the imputation loss L_2, from 0 to 1 when updating the feature extractors g_1, g_2. Both λ_1 and λ_2 are set

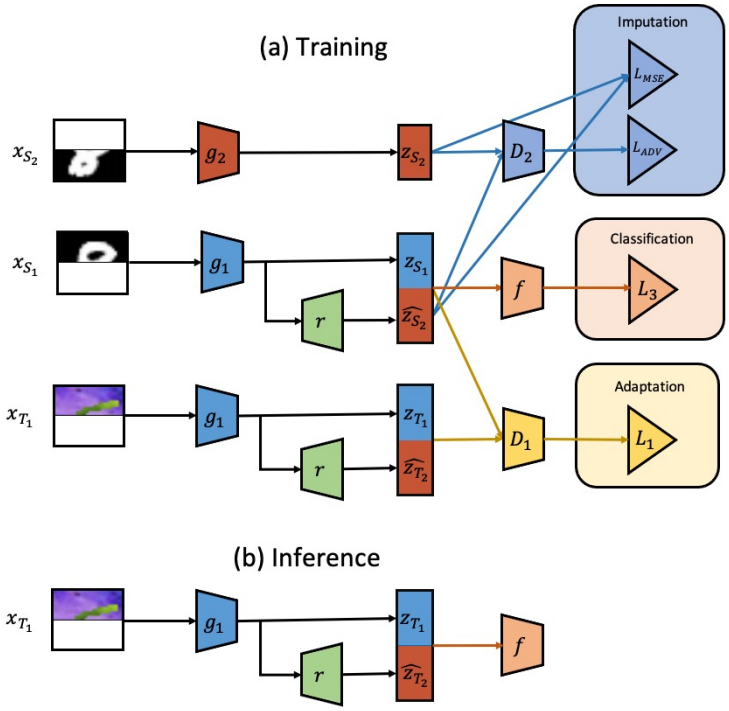


Fig. 2: *Adaptation-Imputation* model. The first column represents examples of raw data with missing and non-missing parts. Trapezoidal boxes represent mapping functions. Triangles in the last column represent loss functions used only for training. At training, the top-row depicts how $\mathbf{x}_{\mathbf{S}_2}$ is mapped into the latent space with g_2. The second and third rows show how $\widehat{\mathbf{z}}_{\mathbf{S}}$ and $\widehat{\mathbf{z}}_{\mathbf{T}}$ are obtained. All these imputed and mapped source and target samples are then used in training losses. At inference, we only need the learned g_1 and r for mapping the target example with missing data into the latent space and f for predicting its class.

to 1 when updating the discriminators D_1, D_2 per [18]. Moreover, we decay all learning rates. We fix $\lambda_3 = 1$ to avoid additional tuning and only tune λ_{MSE} as shown in the ablation study in Table 4 and Figure 5. All components are trained jointly after first initializing the classifier f and feature extractors g_1, g_2 to minimize L_3 replacing $\widehat{\mathbf{z}}_{\mathbf{S}_2}$ with $\mathbf{z}_{\mathbf{S}_2}$ such that discriminitative components are learned before joint adaptation and imputation. Appendix E provides details of all architectures and parameters and our code is available[5].

[5] https://github.com/mkirchmeyer/adaptation-imputation

Algorithm 1 Adversarial Adaptation-Imputation training procedure

N: number of epochs, k: batch size

Initialize f, g_1, g_2 by minimizing L_3 replacing $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ with $\mathbf{z}_{\mathbf{S_2}}$
for $n_{epoch} < N$ **do**
    Sample $\{\mathbf{x}_{\mathbf{S}}^{(\mathbf{i})}, y_S^{(i)}\}_{1 \leq i \leq k}$ from $p_S(X,Y)$
    Sample $\{\mathbf{x}_{\mathbf{T}}^{(\mathbf{j})}\}_{1 \leq j \leq k}$ from $p_T(X)$
    Decay learning rate and update gradient scale at each batch
    Compute $L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3$ performing joint adaptation, imputation, classification
    Update D_1, D_2 by ascending L through Gradient Reversal Layer
    Update f, g_1, g_2, h by descending L

5 Theoretical insights

5.1 Target generalization error

Given the model in Section 4.2, we show in this section that, despite having only unlabelled target samples, we minimize the model's target classification error using source labels with an adaptation upper bound (Theorem 1), under our assumptions. We then show an imputation upper bound of the "ideal" target error obtained with all components observed (classical UDA setting) by our model's target error times a factor (Proposition 1). The analytical expression of this factor highlights the role of two components: imputation on the source and transfer of this imputation from the source to the target. In the optimal case, when the model perfectly recovers these two operations, we show that our model retrieves the ideal target error. These two bounds thus provide an approach with adaptation and imputation to minimize our model's target error and reach the ideal target error, using only missing target data and source supervision for both labels and imputation.

Definitions First, we recall some definitions. $\widehat{g}$ in (2) maps the first component of a sample to its imputed latent representation. $\widehat{g}$ can be applied to both source and target samples. On the other hand, g in (1) maps both input components on the latent space and is thus only applicable to source samples. In practise, $\widehat{g}$ and g share the same encoder for the first component g_1; the second encoder is respectively $r \circ g_1$ for $\widehat{g}$ and g_2 for g. The random variables associated to these projections are denoted respectively Z_2, for the latent missing component built from X_2 with g_2 and $\widehat{Z}_2$, for the reconstruction of Z_2 from X_1 with $r \circ g_1$. X_2 is missing on T but observed on S. Based on these mappings, we define the risk of a hypothesis h on domain $D \in \{S,T\}$, either $h_g \in \mathscr{H}_g = \{f \circ g : f \in \mathscr{F}\}$ or $h_{\widehat{g}} \in \mathscr{H}_{\widehat{g}} = \{f \circ \widehat{g} : f \in \mathscr{F}\}$, as its error under the true labeling function f_D and D, i.e.

$$\varepsilon_D(h) \triangleq \varepsilon_D(h, f_D) \triangleq \mathbb{E}_{\mathbf{x} \sim p_D(X)}[|h(\mathbf{x}) - f_D(\mathbf{x})|]$$

In our case, as h and f_D are binary classification functions, this definition reduces to the probability that h disagrees with f_D under $p_D(X)$

$$\varepsilon_D(h) = \mathbb{E}_{\mathbf{x} \sim p_D(X)}[|h(\mathbf{x}) - f_D(\mathbf{x})|] = \mathbb{E}_{\mathbf{x} \sim p_D(X)}[\mathbb{I}(h(\mathbf{x}) \neq f_D(\mathbf{x}))] = \Pr_{\mathbf{x} \sim p_D(X)}(h(\mathbf{x}) \neq f_D(\mathbf{x}))$$

In the following, we describe the adaptation and imputation bounds.

Adaptation bound As target samples are unlabelled, we cannot directly minimize our model's target error, $\varepsilon_T(f \circ \widehat{g})$. In practise, we upper bound $\varepsilon_T(f \circ \widehat{g})$ in Theorem 1 with adaptation. Adaptation is performed on both components despite target missingness thanks to imputation which reconstructs the missing latent component conditionally on the observed one.

Theorem 1 (Proof in Appendix C). *Given $f \in \mathscr{F}, \widehat{g}$ in* (2) *and $p_S(\widehat{Z})$, $p_T(\widehat{Z})$ the latent marginal distributions obtained with $\widehat{g}$.*

$$\varepsilon_T(f \circ \widehat{g}) \leq \underbrace{\left[\varepsilon_S(f \circ \widehat{g}) + d_{\mathscr{F}\Delta\mathscr{F}}(p_S(\widehat{Z}), p_T(\widehat{Z})) + \lambda_{\mathscr{H}_{\widehat{g}}}\right]}_{\text{Domain Adaptation } \textit{(DA)}} \tag{10}$$

with $\varepsilon_S(\cdot), \varepsilon_T(\cdot)$ the expected error under the labelling function f_S, f_T on S, T respectively; $\mathscr{F}\Delta\mathscr{F}$ the symmetric difference hypothesis space[6]*; $d_{\mathscr{H}}$ the $\mathscr{H}$-divergence for $\mathscr{H} = \mathscr{F}\Delta\mathscr{F}$ and $\lambda_{\mathscr{H}_{\widehat{g}}} = \min_{f' \in \mathscr{F}} [\varepsilon_S(f' \circ \widehat{g}) + \varepsilon_T(f' \circ \widehat{g})]$, the joint risk of the optimal hypothesis.*

The upper bound in (10) consists of $\varepsilon_S(f \circ \widehat{g})$ assessing the discriminative information of source latent components and $d_{\mathscr{F}\Delta\mathscr{F}}(p_S(\widehat{Z}), p_T(\widehat{Z})) + \lambda_{\mathscr{H}_{\widehat{g}}}$, assessing the transfer to the target. Our model minimizes this upper bound (DA); L_3 in (7) corresponds to the first term while L_1 in (3) to the second. Assumption 4 allows us to consider the third term as small. Adaptation affects both components $(Z_1, \widehat{Z}_2)$ as the missing component is imputed with $r \circ g_1$, yet, imputation here is not supervised with fully observed components.

Imputation bound Given $f \in \mathscr{F}$, we compare under our assumptions $\varepsilon_T(f \circ \widehat{g})$ and the ideal target error with full data, $\varepsilon_T(f \circ g)$, with $g = (g_1, g_2)$ and $\widehat{g} = (g_1, r \circ g_1)$. This allows us to measure the loss in performance due to missingness when using $f \circ \widehat{g}$ instead of $f \circ g$. g_1 is shared in g and $\widehat{g}$ while $r \circ g_1$ reconstructs the missing component on both domains. We first derive Lemma 1 used in our upper bound in Proposition 1.

Lemma 1 (Proof in Appendix C). *For any continuous density distributions p, q defined on an input space $\mathscr{X}$, such that $\forall \mathbf{x} \in \mathscr{X}, q(\mathbf{x}) > 0$, the inequality $\sup_{\mathbf{x} \in \mathscr{X}} [p(\mathbf{x})/q(\mathbf{x})] \geq 1$ holds. Moreover, the minimum is reached when $p = q$.*

We derive Proposition 1. Under Assumption 3, given a classifier $f \in \mathscr{F}$ and encoders $g, \widehat{g}$, this proposition upper bounds $\varepsilon_T(f \circ g)$ with $\varepsilon_T(f \circ \widehat{g})$ multiplied by a factor (IT) in (11). Our model minimizes both the Adaptation upper bound and the term (IT).

Proposition 1 (Proof in Appendix C). *Under Assumption 3, let $f \in \mathscr{F}, \widehat{g}$* (2) *and g* (1)*,*

$$\varepsilon_T(f \circ g) \leq \underbrace{\underbrace{\sup_{\mathbf{z} \sim p(Z)} \left[\frac{p_S(Z_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}\right]}_{\text{Imputation error on S } \textit{(IS)}} \times \underbrace{\sup_{\mathbf{z} \sim p(Z)} \left[\frac{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}\right]}_{\text{Transfer error of Imputation } \textit{(TI)}}}_{\text{Imputation error on T } \textit{(IT)}} \times \varepsilon_T(f \circ \widehat{g}) \tag{11}$$

[6] $h \in \mathscr{F}\Delta\mathscr{F} \iff h(\mathbf{x}) = f_1(\mathbf{x}) \oplus f_2(\mathbf{x})$ for some $f_1, f_2 \in \mathscr{F}$ where $\oplus$ is the XOR function.

Under Lemma 1, (IT)=1 is the minimal value reached when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ *and* $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$. *In this case,* $\varepsilon_T(f \circ g) = \varepsilon_T(f \circ \widehat{g})$.

The upper bound in (11) shows that for any $f, \widehat{g}, g$, $\varepsilon_T(f \circ g)$ is upper bounded by $\varepsilon_T(f \circ \widehat{g})$ times the multiplicative factor (IT). The optimal situation, equality, is obtained when (IT) equals 1. (IT) measures how imputation recovers the missing target component and is decomposed into two terms. (IS) quantifies how imputation learns $p_S(Z_2|Z_1)$ with $p_S(\widehat{Z}_2|Z_1)$ i.e. reconstructs the component $Z_2 = g_2(X_2)$ with $\widehat{Z}_2 = r(Z_1)$ and $Z_1 = g_1(X_1)$ on the source. (TI) measures the divergence of $\widehat{Z}_2|Z_1$ across domains; the lower, the better indirect imputation supervision from S transfers to T. The equality case occurs when (IT) is minimal, i.e when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$. Our model minimizes (IT) after first initializing f, g with $\text{argmin}_{f,g}\, \varepsilon_S(f \circ g)$ replacing $\widehat{g}$ with g in L_3, (7) to extract discriminative components $(\mathbf{z_{S_1}}, \mathbf{z_{S_2}})$. It minimizes (IS) with L_2 in (4) while (TI) is minimized with the adaptation loss L_1 in (3). Note that (IT) is minimal when $L_1 = L_2 = 0$ yielding to the equality of $\varepsilon_T(f \circ g)$ and $\varepsilon_T(f \circ \widehat{g})$.

5.2 Self-training refinement $\mathscr{R}$

We now introduce a heuristic based on pseudo-labels useful for settings where Assumption 4 is not verified because $p_S(Y|\widehat{Z}) \neq p_T(Y|\widehat{Z})$. Assumption 4 allows to consider $\lambda_{\mathscr{H}_{\widehat{g}}}$ in (10) as small. Indeed, several authors e.g. [22, 49] recently demonstrated that minimizing the first two terms in (DA) (10) is not sufficient for successful UDA. They show that (1) even when covariate shift is true in the data space, it usually does not hold in the latent space; (2) even when the first two terms in (DA) in (10) are minimized, the third, $\lambda_{\mathscr{H}_{\widehat{g}}}$, might increase so that the bound is not minimized. [49] shows that in addition to the above conditions, one should enforce the posterior class distributions $p_D(Y|X)$ to be close on the two domains. Since T is unlabeled there is no direct way to do that. We instead propose a simple heuristic using pseudo-labels and show how they can be incorporated with a simple adaptation of (10). Pseudo-labels are tentative labels assigned to target unlabelled samples by a classifier, denoted $h_{\widehat{g}}$ below. As $\lambda_{\mathscr{H}_{\widehat{g}}}$ cannot be measured without target labels, we will approximately evaluate and minimize it with pseudo-labels.

Proposition 2 (Proof in Appendix C). *Assume a joint distribution* $p_{\widetilde{T}}(X,Y)$ *where* $p_{\widetilde{T}}(X) = p_T(X)$ *and* $Y = h_{\widehat{g}}(X)$ *where* $h_{\widehat{g}} = f \circ \widehat{g} \in \mathscr{H}_{\widehat{g}}$ *is a candidate hypothesis. Then,*

$$\lambda_{\mathscr{H}_{\widehat{g}}} \leq \min_{h_{\widehat{g}} \in \mathscr{H}_{\widehat{g}}} \left[\varepsilon_S(h_{\widehat{g}}) + \varepsilon_{\widetilde{T}}(h_{\widehat{g}}) + \varepsilon_T(f_{\widetilde{T}}) \right] \tag{12}$$

with $\varepsilon_T(f_{\widetilde{T}}) = \Pr_{\mathbf{x} \sim p_T(X)}(f_{\widetilde{T}}(\mathbf{x}) \neq f_T(\mathbf{x}))$ *the T error of the pseudo-labelling function* $f_{\widetilde{T}}$.

The first two terms on the right hand side of (12) may be controlled as we know source labels and target pseudo-labels; the third term is the error of the pseudo-labeling function, minimal if pseudo-labels are equal to true target labels. We cannot measure the last term but propose self-training as a way to heuristically improve the pseudo-labeling function.

We detail one way to do so in Algorithm 2. We start from an initial set of pseudo-labels, e.g. the pseudo-labels provided by the model in Section 4.2 and then refine them. Many self-training methods have been proposed. We use a combination of two such methods, initially proposed for semi-supervised learning: an adaptation of the semi-supervised discriminant Classification Expectation Maximization (CEM) in [2] and semi-supervised learning by entropy minimization [19]. We found that combining these two approaches performed better than each method used alone.

In the following we assume to have a set $\mathscr{S}, \mathscr{T}$ respectively of labelled S and unlabelled T samples. [2] introduce an iterative method which starts from pseudo-labels provided by an initial classifier and retrains the classifier with these labels. We start with $f(\widehat{\mathbf{z}})$ trained as in Section 4.2 and keep, at each iteration, all samples in $\mathscr{T}$ whose classification score is above a threshold, this set of pseudo-labelled instances is denoted $\mathscr{T}^{pl}$. We then minimize a cross-entropy loss on $\mathscr{S} \cup \mathscr{T}^{pl}$, between the labels for $\mathscr{S}$ or pseudo-labels for $\mathscr{T}^{pl}$ and the predicted scores. [19] optimizes an entropy loss on the distribution of the predicted class posteriors output from f for all unlabelled samples; we apply this loss to $\mathscr{T} \setminus \mathscr{T}^{pl}$. This entropy loss can be considered as a soft version of the discriminant CEM loss.

In conclusion, we first train the model without pseudo-labels minimizing L (Section 4.2). We then use the learned classifier to provide initial pseudo-labels and minimize jointly discriminant CEM and entropy loss to refine them. Given $h_{\widehat{g}} = f \circ \widehat{g} \in \mathscr{H}_{\widehat{g}}$ a hypothesis with $\forall k \in [1,K], h_{\widehat{g}_k}(\mathbf{x})$ the probability of predicting instance $\mathbf{x}$ to class k, L_{Disc} a cross-entropy loss and λ a weight for entropy, the objective function of our refinement method is:

$$L_{\mathscr{R}} = \underbrace{\sum_{(\mathbf{x},y)\in\mathscr{S}\cup\mathscr{T}^{pl}} L_{Disc}(h_{\widehat{g}}(\mathbf{x}),y)}_{\text{Discriminant CEM (CEM)}} + \lambda \underbrace{\sum_{\mathbf{x}\in\mathscr{T}\setminus\mathscr{T}^{pl}} \sum_{k=1}^{K} h_{\widehat{g}_k}(\mathbf{x}) \log h_{\widehat{g}_k}(\mathbf{x})}_{\text{Entropy (E)}} \tag{13}$$

The first term in (13), (CEM), controls $\varepsilon_S(h_{\widehat{g}}) + \varepsilon_{\widetilde{T}}(h_{\widehat{g}})$ while the second term, (E), heuristically controls $\varepsilon_T(f_{\widetilde{T}})$ by encouraging separation between classes. We found that this heuristically brings pseudo-labels closer to the target labels on our datasets. In practise, we minimize $L_{\mathscr{R}}$ with respect to $f, \widehat{g}$.

Algorithm 2 Self-training procedure for Adaptation-Imputation

Input $\mathscr{S} = \{(\mathbf{x}_{\mathbf{S}}^{(\mathbf{i})}, y_S^{(i)})\}_{i=1}^{N_S}$, $\mathscr{T} = \{(\mathbf{x}_{\mathbf{T}}^{(\mathbf{i})}\}_{i=1}^{N_T}$, Adaptation-Imputation method $\mathscr{A}$ in Section 4.2
Output Classifier f; Feature extractor $\widehat{g}$ defined in (2)

1: $f, \widehat{g} = \mathscr{A}(\mathscr{S}, \mathscr{T})$ ▷ Initialize $f, \widehat{g}$ with *Adaptation-Imputation* (8)
2: $f, \widehat{g} = \arg\min_{f,\widehat{g}} L_{\mathscr{R}}$ ▷ Semi-supervised refinement of $f, \widehat{g}$ by optimizing (13)

6 Experiments

6.1 Datasets and experimental setting

Datasets Experiments are performed on three types of datasets. The first one, `digits`, is a classical multi-class classification benchmark used in many UDA studies and adapted to fit our missing data setting. The second one, which initially motivated our framework, consists of advertising datasets where we aim at transferring knowledge from retargeting users with full browsing information to prospecting users with missing information. The task is binary classification as measured by Click-Through-Rate (CTR) or Conversion Rate (CR)[7] given user browsing traces. We use two such datasets: `ads-kaggle` is a public kaggle dataset[8], while `ads-real` was gathered internally. Both correspond to real advertising traffic. Finally, we performed tests on a text dataset, Amazon reviews, denoted `amazon`. The initial problem is transformed into binary classification and to a non-stochastic missing data problem. For both `digits` and `amazon`, a subset of the components are set to 0 to mimic missing data while on `ads`, data is missing structurally (more details in Appendix D).

Baselines We report results for the following models:

(a) *Source-Full* trained without adaptation on $\mathbf{x_S}$ and tested on full $\mathbf{x_T}$; adaptation is added in *Adaptation-Full*. Note that this model is only applicable for our academic benchmark where we have access to full data.

(b) *Source-ZeroImputation* and *Adaptation-ZeroImputation* do the same but considering full $\mathbf{x_S}$ while $\mathbf{x_T}$ is incomplete. Missing data $\mathbf{x_{T_2}}$ is set to $\mathbf{0}$, $\mathbf{x_T} = (\mathbf{x_{T_1}}, \mathbf{0})$.

(c) *Source-IgnoreComponent* and *Adaptation-IgnoreComponent* are a variant of the above where only $\mathbf{x_{D_1}}$ is considered while $\mathbf{x_{D_2}}$ is ignored for both S and T.

(d) *Adaptation-Imputation*, our model, considers full $\mathbf{x_S}$ and $\mathbf{x_T} = (\mathbf{x_{T_1}}, \mathbf{0})$ adding imputation with a conditional generative model.

(e) We add self-training to *Adaptation-Imputation* and when applicable to *Adaptation-Full*.

Note that *Adaptation-Full* is an upper bound of our imputation model since it uses full information while $\mathbf{x_{T_2}}$ is not available in practice. *Adaptation-ZeroImputation* and *Adaptation-IgnoreComponent* are lower bounds for our model since they only perform adaptation and do not impute non-zero values.

Hyperparameters Parameters are chosen using the DEV estimator [47]. For `digits`, NN architectures are adapted from [18]; we use Adam optimizer with $lr = 10^{-2}$ decayed; batch size of 128 and 100 epochs. For `ads` and `amazon`, three-layered NN with 128 neurons per layer are used as feature extractors; the classifier and discriminators are single-layered with 128 neurons; $lr = 10^{-6}$ and is decayed; batch size is 500 with 50 epochs. Reported results are mean value and standard deviation over five runs and best results are indicated in **bold**. Further details are given in the Appendix E.2.

6.2 Digits

Description We consider UDA problems between several datasets: MNIST [23], USPS [20], SVHN [31] and MNIST-M [18] as illustrated in Figure 3 (a). MNIST $\rightarrow$ SVHN

[7] CTR is the number of clicks made on ads divided by the number of shown ads. CR replaces clicks with purchases.

[8] http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/

is not considered as it is difficult for traditional UDA [18]. All tasks are 10-class classification problems. From complete digits datasets, we build datasets with missing input values by setting corresponding pixel values to zero for horizontal patches of different sizes as illustrated on Figure 3 (b) for MNIST-M digits. It is clear that there is domain shift on these datasets as the pixel values have different mean and variance across domains.

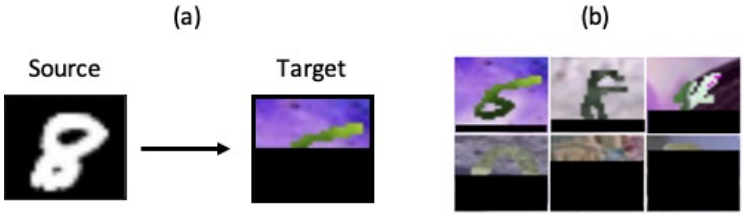


Fig. 3: (a) MNIST→MNIST-M adaptation; (b) Digits with missing horizontal patches of increasing size

Results with half of the digit missing We first removed half of each target digit, the horizontal bottom part. We report target accuracy in Table 1 for both `ADV` and `OT` models. Removing half of the digit leads to a strong performance decrease for *Source-IgnoreComponent* and *Source-ZeroImputation* compared to the upper-bounds of *Source-Full*; the performance is partially recovered with adaptation. *Adaptation-Imputation* clearly improves on *Adaptation-IgnoreComponent* and *Adaptation-ZeroImputation* in all cases which validates the importance of imputation. However, it does not reach the upper bound performance of *Adaptation-Full*. Both `ADV` and `OT` versions exhibit the same behavior. In the results in Table 1, `ADV` performance is higher than `OT`. This is because performance is highly dependent on the NN architectures and we tuned our NNs for `ADV`. `OT` models may reach performance similar to `ADV` but require an order of magnitude more parameters. To keep the comparison fair, we use the same NN models for both `ADV` and `OT`. Imputation models achieve their highest performance when adaptation between domains is complex (MNIST → MNIST-M, SVHN → MNIST) illustrating the importance of imputation when transfer is difficult. We show in Appendix F the learned latent representations $\widehat{\mathbf{z}}_{\mathbf{S}}, \widehat{\mathbf{z}}_{\mathbf{T}}$ for various `digits` adaptation problems.

Varying missing patch size We analyze the impact of the size of the missing patch by removing a percentage $p \in \{30\%, 40\%, 50\%, 60\%, 70\%\}$ of MNIST digits when adapting SVHN → MNIST, with the same hyperparameters. Mean values over five runs are reported in Figure 4 for `ADV` models. We notice that our model constantly beats the other baselines regardless of the missing patch size. The figure exhibits borderline cases when the size of the missing patch becomes very small ($< 30\%$) or very large ($> 65\%$). When the missing patch is small there is enough information for predicting the label thus simple models perform well; when it becomes big, there is not enough information for efficient reconstructions.

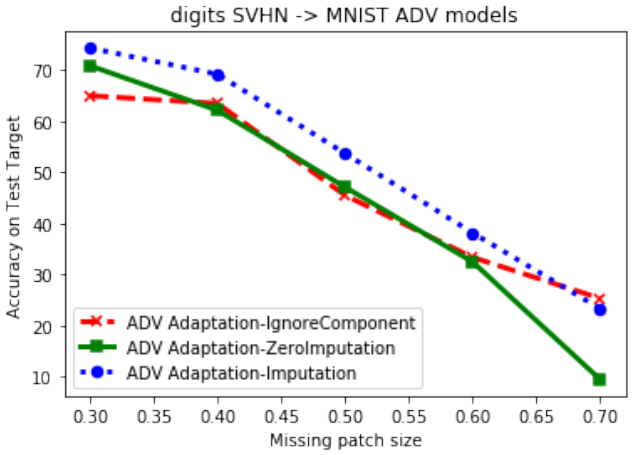


Fig. 4: ADV target accuracy (↑) on SVHN → MNIST with missing patch size

6.3 Ads

Description The ads datasets are used for solving the binary classification problem of predicting if a **user** exposed to an ad from a **partner** (e.g. Booking.com) clicks given his browsing history. A row in this dataset is a vector $\mathbf{x} = (\mathbf{x_{D_1}}, \mathbf{x_{D_2}})$ specific to a (user-partner) pair where $\mathbf{x_{D_1}}$ gathers mean statistics for this user on all visited partners summarizing the user's display and click statistics and $\mathbf{x_{D_2}}$ corresponds to the user-partner specific traces. The label is the response to an ad for this (user, partner) pair, a click for ads-kaggle or a purchase for ads-real. We transfer knowledge from the labelled source domain composed of all user-partner pairs for which the user has already interacted with the partner (**retargeting** users) to the unlabelled target domain composed of all the user-partner pairs for which the user has never interacted with this partner (**prospecting** users). $\mathbf{x_{S_2}}$ is known but $\mathbf{x_{T_2}}$ is **unknown**. There are several partners and users per domain. These datasets are large scale as seen in Table 6 (1M and 24M source displays respectively for ads-kaggle, ads-real) with some specificities: there is class imbalance and five times less data on the target than on the source. For both datasets besides missingness, there is also an adaptation problem: prospecting users tend to be less active and their statistics are usually different from those of retargeting users, with a higher overall activity (e.g. in terms of frequency of a partner's website visits); this translates into distribution shifts on $\mathbf{x_{D_1}}$ across domains. We visualize in the Appendix in Table 7 and Figure 6 the domain shift in ads-kaggle which comprises 13 features. Table 7 reports mean and standard deviation on each feature's value over a domain and Figure 6 plots the histogram of the distribution of each feature where the y-axis is unnormalized and corresponds to real counts. Feature 5 is naturally missing on T and distributions are different in shape, mean and variance across domains. To show the benefit of modelling additional missing features, we artificially set features 1, 6, 7, 11 and 12 to zero on T such that in total 6 features are missing while 7 are present. On ads-real, 12 features are missing while 17 are present and we ob-

serve the same domain shift trend; however missing features are naturally missing and we do not have access to their value.

Results We report results in Table 1 only for ADV models as we observed that the trend is similar for both ADV and OT. Missing features are structurally missing in the datasets, so we cannot report results for models using full inputs. The classes being imbalanced, accuracy is not relevant here so we report the log cross-entropy (CE) between the predicted values and the true labels. CE is considered to be the most reliable metric to estimate revenue for the ads problem and for large user bases small CE improvements can lead to a large revenue increase. For ads-kaggle, an improvement of 0.001 in CE is considered as significant [43]. A first observation is that the imputation model is substantially better than the baselines on both datasets. For ads-kaggle it improves by 2.3% the best adaptation model i.e. the adaptation model with zero imputation while for ads-real the improvement reaches 6.3% over the second-best *Source-IgnoreComponent*. A second observation is that for any model, adaptation consistently improves over the model without adaptation. The only exception is the setting ignoring the missing component in ads-real. A third observation is that there is a benefit of imputing the missing component for classification: source CE (not reported) shows that *Source-ZeroImputation* which exploits $\mathbf{x}_{\mathbf{D_2}}$ is consistently higher than *Source-IgnoreComponent* which does not, leading to relative gains of 5.6% on ads-kaggle and 8.2% on ads-real. The imputation model is able to generate and exploit this information.

Dataset	MNIST → USPS		USPS → MNIST		SVHN → MNIST		MNIST → MNIST-M		ads-kaggle	ads-real
Model w/o $\mathscr{R}$	ADV	OT	ADV	OT	ADV	OT	ADV	OT	ADV	ADV
Source-Full	71.5±2.7		74.2±2.7		58.1±1.1		28.3±1.4		NA	
Adaptation-Full	85.8±3.2	92.6±1.7	94.6±2.1	93.9±0.6	78.0±3.4	76.1±1.4	60.8±3.8	46.9±3.9	NA	
Source-ZeroImputation	25.7±3.7		39.2±2.6		31.5±2.		14.4±1.1		0.545±0.019	0.663±0.011
Adaptation-ZeroImputation	48.4±4.8	60.9±6.3	67.5±2.2	65.3±5.2	47.1±5.7	37.5±6.2	34.7±2.5	20.2±2.5	0.397±0.0057	0.660±0.025
Source-IgnoreComponent	52.9±9.7		54.3±1.6		44.6±1.9		19.1±2.6		0.406±0.00046	0.622±0.0048
Adaptation-IgnoreComponent	71.5±3.2	64.0±5.0	80.0±1.4	72.0±1.8	45.5±1.9	47.9±1.8	29.4±1.6	26.8±4.4	0.403±0.0030	0.634±0.0082
Adaptation-Imputation	**74.2±2.3**	**66.8±1.3**	**81.4±0.8**	**72.5±2.7**	**53.8±1.4**	**49.2±1.5**	**57.9±2.3**	**29.2±1.4**	**0.389±0.014**	**0.583±0.013**

Table 1: Best target accuracy (↑) on digits and CE (↓) on ads without $\mathscr{R}$

6.4 Amazon reviews

Description Besides dealing with images and interaction features in the digits and ads datasets, we also performed experiments on an additional modality, text. amazon is the Amazon product review dataset [6] with four domains (Books, DVDs, Electronics, and Kitchen) transformed to binary classification with positives referring to reviews with rating above 3 stars and negatives to reviews with rating below 3 stars. Additional details on data processing can be found in Appendix D. We consider four adaptation problems and simulate missing features by setting the first half of the features to zero.

Results Results are reported in Table 2 and confirm our prior findings i.e. that jointly performing adaptation and imputation improves our baselines. We also notice that our

model achieves similar performance to models using full data showing that imputation successfully recovered the missing component.

Dataset	DVD → Electronics	Books → Kitchen	Kitchen → Electronics	DVD → Books
Source-Full	69.57	73.04	77.88	71.95
Adaptation-Full	73.62	74.09	79.63	72.65
Source-ZeroImputation	58.51	60.52	66.27	61.15
Adaptation-ZeroImputation	64.51	61.08	68.02	62.80
Source-IgnoreComponent	60.21	62.03	67.62	64.35
Adaptation-IgnoreComponent	61.02	64.08	68.47	66.00
Adaptation-Imputation	**72.57**	**72.69**	**78.18**	**72.61**

Table 2: Best target accuracy (↑) on `amazon` without $\mathscr{R}$

6.5 Refinement $\mathscr{R}$

Results with pseudo-labels are reported in Table 3 on `digits` and `ads-kaggle` for *Adaptation-Full* and *Adaptation-Imputation*. We set the threshold score selection for the discriminative CEM component to 95% i.e. the pseudo labels of all target instances $\mathbf{x_T}$ s.t. $\max_k h_{\widehat{g}_k}(\mathbf{x_T}) \geq 0.95$ are considered to be true and set the entropy weight to $\lambda = 0.1$ on `digits` and $\lambda = 1$ on `ads-kaggle`. Learning rates used for solving (9) are divided by 10 and 10 epochs of successive refinement steps are applied. We observe a clear global improvement on both datasets showing that our refinement model is a good heuristic on real-world datasets for which we usually have $p_S(Y|\widehat{Z}) \neq p_T(Y|\widehat{Z})$. For standard UDA methods such as *Adaptation-Full*, performance is significantly improved everywhere with small change on MNIST → MNIST-M; *Adaptation-Full* is not measurable for `ads-kaggle`. Our imputation with refinement model follows the same trend with a considerable relative gain of $+18.5\%$ on `ads-kaggle`.

`ADV` Model	MNIST → USPS	USPS → MNIST	SVHN → MNIST	MNIST → MNIST-M	`ads-kaggle`
Adaptation-Full w/ $\mathscr{R}$	**95.9±0.6 (+12%)**	**96.8±0.6 (+2.3%)**	**83.3±3.9 (+6.8%)**	**60.9±3.7 (+0.2%)**	NA
Adaptation-Imputation w/ $\mathscr{R}$	**78.5±1.6 (+5.8%)**	**82.5±0.5 (+1.4%)**	**58.6±1.8 (+8.9%)**	**58.2±2.3 (+0.5%)**	**0.317±0.0023 (+18.5%)**

Table 3: $\mathscr{R}$ with relative gain over Table 1; target accuracy (↑) on `digits` and CE (↓) on `ads`

6.6 Ablation analysis

We analyze the importance of each component of our model on the public datasets (`digits`, `amazon` and `ads-kaggle`) and report results in Table 4 (bottom) and Figure 5.

Adaptation We measure the effect of adaptation term L_1 (3) in L in Table 4 (first row). When removing adaptation, inference is performed as before by feeding $\widehat{\mathbf{z}}_\mathbf{T}$ to f. This means that we only rely on the imputation and classification losses to learn the parameters of the model. For all datasets, adding L_1 considerably increases performance.

Imputation Imputation $\widehat{\mathbf{z}}_{\mathbf{S_2}} = h \circ g_1(\mathbf{x}_{\mathbf{S_1}})$, combines adversarial training (ADV) and conditioning on the input datum via MSE (MSE) in L_2 (4). ADV aligns the distributions of $\mathbf{z}_{\mathbf{S_2}}$ and $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ while MSE can be thought as performing regression. For a given $\mathbf{x}_{\mathbf{S_1}}$, there are possibly several potential $\mathbf{x}_{\mathbf{S_2}}$ and thus $\mathbf{z}_{\mathbf{S_2}}$. ADV allows us to focus on a specific mode of $\mathbf{z}_{\mathbf{S_2}}$, while MSE will favour a mean value of the distribution. Results in Table 4 (second row), show that for our datasets, combining MSE and ADV leads to improved results compared to using separately each loss. MSE alone already provides good performance, while using only ADV is clearly uncompetitive. Note that reconstruction is an ill-posed problem since the task is inherently ambiguous (different digits may be reconstructed from a half image). We performed tests with a stochastic input component to recover different modes, but the performance was broadly similar. We investigate in Figure 5 several weighted combinations of MSE and ADV: for digits and amazon, equal weights were found to be a good choice, while for ads-kaggle performance is improved with other weightings. On Figure 5, ADV induces a high variance in the results (left part of x-axis) while MSE stabilizes the performance (right part of x-axis). ADV allows for better performance at the expense of high variance; a small contribution from MSE, $\lambda_{MSE} = 0.005$, stabilizes the results.

Ablation study	ADV Model	MNIST → USPS	USPS → MNIST	SVHN → MNIST	MNIST → MNIST-M	ads-kaggle
L_2+L_3 vs. $L_1+L_2+L_3$	$L=\lambda_2 L_2+\lambda_3 L_3$	64.2±1.8 (-13%)	51.3±2.5 (-37%)	44.5±1.4 (-17%)	24.1±2.6 (-58%)	0.410±0.0020 (-5.4%)
ADV-MSE weighting in L_2	$L_2 = L_{MSE}$	71.9±3.7 (-3.1%)	**81.4±1.2 (0%)**	52.5±3.7 (-2.4%)	56.5±2.8 (-2.4%)	0.400±0.0014 (-2.8%)
	$L_2 = L_{ADV}$	28.6±3.2 (-61%)	39.4±5.2 (-52%)	28.8±3.8 (-46%)	30.0±3.7 (-48%)	0.469±0.13 (-21%)
	$L_2 = L_{ADV} + 0.005 \times L_{MSE}$	47.8±3.7 (-36%)	49.6±5.8 (-39%)	46.0±2.6 (-15%)	50.6±2.2 (-13%)	**0.389±0.014 (0%)**
	$L_2 = L_{ADV} + L_{MSE}$	**74.2±2.3 (0%)**	**81.4±0.8 (0%)**	**53.8±1.4 (0%)**	**57.9±2.3 (0%)**	0.401±0.0014 (-3.1%)

Ablation study	ADV Model	DVD → Electronics	Books → Kitchen	Kitchen → Electronics	DVD → Books
ADV-MSE weighting in L_2	$L_2 = L_{MSE}$	71.47 (-1.5%)	71.39 (-1.8%)	77.58 (-0.77%)	72.02 (-0.81%)
	$L_2 = L_{ADV} + L_{MSE}$	**72.57 (0%)**	**72.69 (0%)**	**78.18 (0%)**	**72.61 (0%)**

Table 4: Ablation with relative gain over Table 1; accuracy (↑) on digits, amazon and CE (↓) on ads

6.7 Discussion

Relationship between theoretical and experimental results We comment on our experimental results in light of our adaptation (10) and imputation (11) upper-bounds. Let us first consider (10). The first term in (10), $\varepsilon_S(f \circ \widehat{g})$, is the classification loss L_3 in (7). The second term in (10) $d_{\mathscr{F}\Delta\mathscr{F}}(p_S(\widehat{Z}), p_T(\widehat{Z}))$ is approximated by a proxy L_1 (3) and accounts for alignment. L_1 leads to substantial gains in Table 4 (first row) when added to the loss. The third term $\lambda_{\mathscr{H}_{\widehat{g}}}$ in (10) is the optimal joint error heuristically controlled with self-training as justified by upper-bound (12), with gains shown in Table 3. Second, we consider (11). It is the product of two terms, the target imputation error (I_T) and the error on the target $\varepsilon_T(f \circ \widehat{g})$ which is exactly the left hand side term in bound (10). $(I_T) = (I_S) \times (T_I)$, (I_S) is the source imputation error and is optimized when term L_2 (4) is zero. (T_I) is the transfer error, optimized when L_1 (3) is zero. Adding L_1 to the loss improves the performance (Table 4). L_2 (4) explains the gains of *Adaptation-Imputation* over *Adaptation-ZeroImputation* in Table 1 as *Adaptation-ZeroImputation* does not attempt to impute missing components. To summarize, minimizing our global

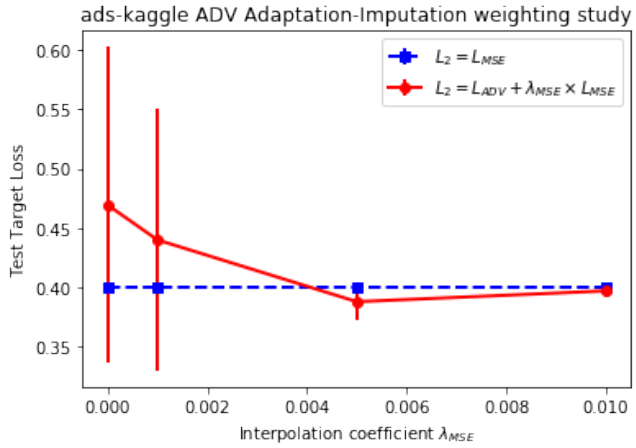


Fig. 5: *Adaptation-Imputation* target CE (↓) with standard deviations on `ads-kaggle` w.r.t. λ_{MSE}

error function L in (8) minimizes, according to the approximations just described, the two upper bounds in (10) and (11).

Limitations Our results are obtained under some assumptions which we are the first to introduce to our knowledge for our problem. First, if the missing and the observed components are statistically independent, Assumption 2 is not valid, and then there is no way to impute this missing data. Second, if $p_S(Z_2|Z_1) \neq p_T(Z_2|Z_1)$ i.e. Assumption 3 is not valid, then we cannot transfer imputation from source to target. Yet, these assumptions are most often met in applications and allow to build a well-defined model with good empirical results.

7 Conclusion

We proposed a new model for UDA with non-stochastic target missingness with indirect supervision from a complete source. This method uses only labelled source instances imputing the missing target values in a latent space. Under our assumptions, it minimizes an adaptation upper-bound of its target error and an imputation upper-bound of the ideal target error with full data and leads to important gains for two representative families of divergences (`OT`, `ADV`) on our benchmarks (digits, amazon) and on real-world advertising datasets, which are a complex task with missing features. We show that approaches using a pure regressive generator underperform compared to our approach on our real-world applications for which distributions are multi-modal. Finally, we introduced a heuristic refinement method based on self-training to deal with settings where posterior distributions mismatch. As follow-up, we plan to further investigate how to generate diverse outputs in our imputation network.

Acknowledgements We would like to thank Keerthi Selvaraj for useful discussions. Alain Rakotomamonjy is funded by RAIMO ANR-20- CHIA-0021-01 and OATMIL ANR-17-CE23-0012 Projects of the French National Research Agency (ANR).

References

1. Aggarwal, K., Yadav, P., Selvaraj, K.S.: Domain adaptation in display advertising: An application for partner cold-start. Proceedings of the 13th ACM Conference on Recommender Systems p. 178–186 (2019)
2. Amini, M.R., Gallinari, P.: Semi-supervised learning with an imperfect supervisor. Knowl. Inf. Syst. **8**, 385–413 (11 2005)
3. Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. ArXiv **arxiv:1907.02893** (2019)
4. Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A., Radha, H.: Cold-start item and user recommendation with decoupled completion and transduction. In: Proceedings of the 9th ACM Conference on Recommender Systems. p. 91–98 (2015)
5. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine Learning **79**(1), 151–175 (2010)
6. Blitzer, J., McDonald, R., Pereira, F.: Domain adaptation with structural correspondence learning. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. pp. 120–128 (2006)
7. Bora, A., Price, E., Dimakis, A.G.: AmbientGAN: Generative models from lossy measurements. In: International Conference on Learning Representations (2018)
8. Cai, L., Wang, Z., Gao, H., Shen, D., Ji, S.: Deep adversarial learning for multi-modality missing data completion. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 1158–1166 (2018)
9. Chen, C., Dou, Q., Chen, H., Qin, J., Heng, P.: Synergistic image and feature adaptation: Towards cross-modality domain adaptation for medical image segmentation. In: Proceedings of the 33rd Conference on Artificial Intelligence (AAAI). pp. 865–872 (2019)
10. Chen, M., Xu, Z., Weinberger, K.Q., Sha, F.: Marginalized denoising autoencoders for domain adaptation. In: Proceedings of the 29th International Conference on International Conference on Machine Learning. p. 1627–1634 (2012)
11. Cortes, C., Mohri, M.: Domain adaptation and sample bias correction theory and algorithm for regression. Theoretical Computer Science **519**, 103–126 (2014)
12. Courty, N., Flamary, R., Amaury, H., Rakotomamonjy, A.: Joint distribution optimal transportation for domain adaptation. In: Advances in Neural Information Processing Systems (2017)
13. Crammer, K., Kearns, M., Wortman, J.: Learning from multiple sources. In: Journal of Machine Learning Research. vol. 9, pp. 1757–1774 (2008)
14. Damodaran, B.B., Kellenberger, B., Flamary, R., Tuia, D., Courty, N.: DeepJDOT : Deep Joint Distribution Optimal Transport for Unsupervised Domain Adaptation. In: European Conference in Computer Visions. pp. 467–483 (2018)
15. Ding, Z., Shao, M., Fu, Y.: Latent low-rank transfer subspace learning for missing modality recognition. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence. pp. 1192–1198 (2014)
16. Doinychko, A., Amini, M.R.: Biconditional gans for multiview learning with missing views. In: Advances in Information Retrieval. pp. 807–820 (2020)
17. Gama, J.a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (Mar 2014)

18. Ganin, Y., Lempitsky, V.: Unsupervised Domain Adaptation by Backpropagation. In: Proceedings of the 32nd International Conference on Machine Learning. pp. 1180–1189 (2015)
19. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. In: Proceedings of the 17th International Conference on Neural Information Processing Systems. p. 529–536 (2005)
20. Hull, J.J.: A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. **16**(5) (1994)
21. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5967–5976 (2017)
22. Johansson, F.D., Sontag, D., Ranganath, R.: Support and invertibility in domain-invariant representations. In: Proceedings of the 32th International Conference on Artificial Intelligence and Statistics. pp. 527–536 (2019)
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
24. Leek, J.T., et al.: Tackling the widespread and critical impact of batch effects in high-throughput data. Nat Rev Genet **11**(10), 733–739 (Oct 2010)
25. Li, S., B., J., Marlin, B.: MisGAN: Learning From Incomplete Data with generative Adversarial Networks. In: International Conference on Learning Representations (2019)
26. Lipton, Z., Wang, Y.X., Smola, A.: Detecting and correcting for label shift with black box predictors. In: Proceedings of the 35th International Conference on Machine Learning. pp. 3122–3130 (2018)
27. Little, R., Rubin, D.: Statistical analysis with missing data. John Wiley & Sons, Inc. (1986)
28. Long, M., Cao, Z., Wang, J., Jordan, M.I.: Conditional adversarial domain adaptation. In: Advances in Neural Information Processing Systems. vol. 31 (2018)
29. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning. vol. 37, pp. 97–105 (2015)
30. Mattei, P.A., Frellsen, J.: MIWAE: Deep Generative Modelling and Imputation of Incomplete Data. In: Proceedings of the 36th International Conference on Machine Learning. vol. 97, pp. 4413–4423 (2019)
31. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011. Proceedings of the IEEE (2011)
32. Pajot, A., de Bezenac, E., Gallinari, P.: Unsupervised Adversarial Image Reconstruction. In: International Conference on Learning Representations (2019)
33. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. p. 1345–1359 (2010)
34. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A.: Context encoders: Feature learning by inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2536–2544 (2016)
35. Peyré, G., Cuturi, M., et al.: Computational optimal transport. Foundations and Trends in Machine Learning **11**(5-6), 355–607 (2019)
36. Rubin, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (12 1976)
37. Sahebi, S., Brusilovsky, P.: Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In: User Modeling, Adaptation, and Personalization. pp. 289–295. Springer Berlin Heidelberg (2013)
38. Shen, J., Qu, Y., Zhang, W., Yu, Y.: Wasserstein distance guided representation learning for domain adaptation. In: 32nd AAAI Conference on Artificial Intelligence (2018)
39. Tran, L., Liu, X., Zhou, J., Jin, R.: Missing modalities imputation via cascaded residual autoencoder. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4971–4980 (2017)

40. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. IEEE Conference on Computer Vision and Pattern Recognition pp. 2962–2971 (2017)
41. Van Buuren, S.: Flexible imputation of missing data. 2nd ed., Chapman and Hall/CRC (2018)
42. Wang, C., Niepert, M., Li, H.: LRMM: Learning to recommend with missing modalities. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 3360–3370 (2018)
43. Wang, R., Fu, B., Fu, G., Wang, M.: Deep cross network for ad click predictions. Proceedings of the ADKDD'17 (2017)
44. Wei, P., Ke, Y., Goh, C.K.: Domain specific feature transfer for hybrid domain adaptation. In: 2017 IEEE International Conference on Data Mining. pp. 1027–1032 (2017)
45. Wei, P., Ke, Y., Goh, C.K.: A general domain specific feature transfer framework for hybrid domain adaptation. IEEE Transactions on Knowledge and Data Engineering **31**(8), 1440–1451 (2019)
46. Yoon, J., Jordon, J., Van Der Schaar, M.: GAIN: Missing data imputation using generative adversarial nets. In: Proceedings of the 35th International Conference on Machine Learning. pp. 5689–5698 (2018)
47. You, K., Wang, X., Long, M., Jordan, M.: Towards accurate model selection in deep unsupervised domain adaptation. Proceedings of the 36th International Conference on Machine Learning pp. 7124–7133 (2019)
48. Zablocki, E., Bordes, P., Soulier, L., Piwowarski, B., Gallinari, P.: Context-aware zero-shot learning for object recognition. In: Proceedings of the 36th International Conference on Machine Learning. vol. 97, pp. 7292–7303 (09–15 Jun 2019)
49. Zhao, H., des Combes, R.T., Zhang, K., Gordon, G.J.: On learning invariant representation for domain adaptation. In: Proceedings of the 36th International Conference on Machine Learning. vol. 97, pp. 7523–7532 (2019)
50. Zhu, J.Y., Park, T., Isola, P., Efros, A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. IEEE International Conference on Computer Vision pp. 2242–2251 (2017)

A Additional related work

We present in this section some other secondary topics related to our problem in complement to Section 2.

Concept drift in data streams Adapting to non i.i.d. data is also considered in evolving data streams where concept drift may occur [17]. The hypotheses are different from the ones in our setting where adaptation is performed between static domains.

Batch effect and multiple environments Data may come from different environments with different distributions. Classical learning frameworks like ERM consider shuffled data without making the distinction between environments which may lead to erroneous conclusions. In biology, this is known as the batch effect [24]. In ML, recent papers learn domain invariant representations from different environments [3]. This is different from the situation considered here where one explicitly adapts from a source to a target environment.

B `OT` Adaptation-Imputation formulation

We present here in more details our model using Optimal Transport (OT) as a divergence metric. The formulation is slightly different compared to `ADV` models. We replace the $\mathscr{H}$-divergence approximation given by the discriminators D_1 and D_2 by the Wasserstein distance between source and target instances (D_1) and true and imputed feature representations (D_2), following the original ideas in [14, 38]. In practice, we compute the Wasserstein distance using its primal form by finding a joint coupling matrix γ, using a linear programming approach [35]. In [12, 14], the OT problem is formulated on the joint $p(X,Y)$ distributions. Similarly to [38], in our case, we focus on a plan that acts only on the feature space without taking care of the labels. This leads to:

$$L_1 = \sum_{ij} \left(||\mathbf{z}_{\mathbf{S_1}}^{(\mathbf{i})} - \mathbf{z}_{\mathbf{T_1}}^{(\mathbf{j})}||^2 + ||\widehat{\mathbf{z}}_{\mathbf{S_2}}^{(i)} - \widehat{\mathbf{z}}_{\mathbf{T_2}}^{(j)}||^2 \right) \boldsymbol{\gamma}_{\mathbf{1}ij} \tag{14}$$

where $\boldsymbol{\gamma}_{\mathbf{1}ij}$ is the alignment value between source instance i and target instance j.

For the imputation part, we keep the reconstruction MSE component in Equation 6 and derive the distribution matching loss as:

$$L_{OT} = \sum_{ij} ||\mathbf{z}_{\mathbf{S_2}}^{(\mathbf{i})} - \widehat{\mathbf{z}}_{\mathbf{S_2}}^{(j)}||^2 \boldsymbol{\gamma}_{\mathbf{2}ij} \tag{15}$$

where $\boldsymbol{\gamma}_{\mathbf{2}ij}$ is the alignment value between source instance i and j. The final imputation loss is:

$$L_2 = \lambda_{OT} \times L_{OT} + \lambda_{MSE} \times L_{MSE} \tag{16}$$

The classification term in Equation 7 is unchanged.

The optimization problem in Equation 9 is solved in two stages following an alternate optimization strategy:

- We fix all parameters but $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_2}$ and find the joint coupling matrices $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_2}$ using EMD $\min_{\boldsymbol{\gamma_1},\boldsymbol{\gamma_2}} L$
- We fix $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_2}$ and solve $\min_{g_1,g_2,r,f} L$

In practice, we first minimize L_3 for a couple of epochs (taken to be 10 for `digits`) then minimize $\lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3$ in the remaining epochs. Learning rate and parameters are detailed further in Section E.

C Proofs

Theorem (1). *Given $f \in \mathscr{F}, \widehat{g}$ in (2) and $p_S(\widehat{Z})$, $p_T(\widehat{Z})$ the latent marginal distributions obtained with g.*

$$\varepsilon_T(f \circ \widehat{g}) \leq \underbrace{\left[\varepsilon_S(f \circ \widehat{g}) + d_{\mathscr{F}\Delta\mathscr{F}}(p_S(\widehat{Z}), p_T(\widehat{Z})) + \lambda_{\mathscr{H}_{\widehat{g}}}\right]}_{\text{Domain Adaptation } (DA)}$$

with $\varepsilon_S(\cdot), \varepsilon_T(\cdot)$ the expected error w.r.t to the labelling function f_S, f_T on S, T respectively; $\mathscr{F}\Delta\mathscr{F}$ the symmetric difference hypothesis space; $d_{\mathscr{H}}$ the $\mathscr{H}$-divergence for $\mathscr{H} = \mathscr{F}\Delta\mathscr{F}$ and
$\lambda_{\mathscr{H}_{\widehat{g}}} = \min_{f' \in \mathscr{F}} [\varepsilon_S(f' \circ \widehat{g}) + \varepsilon_T(f' \circ \widehat{g})]$, the joint risk of the optimal hypothesis.

Proof. We apply [5] to form the bound in $\mathscr{Z}$ using $\widehat{g}$. □

Lemma (1). *For any continuous density distribution p, q defined on an input space $\mathscr{X}$, such that $\forall \mathbf{x} \in \mathscr{X}, q(\mathbf{x}) > 0$, the inequality $\sup_{\mathbf{x} \in \mathscr{X}} [p(\mathbf{x})/q(\mathbf{x})] \geq 1$ holds. Moreover, the minimum is reached when $p = q$.*

Proof. Suppose that $\nexists \mathbf{x} \in \mathscr{X}\, s.t. \sup_{\mathbf{x}} p(\mathbf{x})/q(\mathbf{x}) \geq 1$. This means that $\forall \mathbf{x}, p(\mathbf{x}) < q(\mathbf{x})$. By integrating those positive and continuous functions on their domains, we are lead to the contradiction that the integral of one of them is not equal to 1. Thus, $\exists \mathbf{x} \in \mathscr{X}\, s.t. p(\mathbf{x})/q(\mathbf{x}) \geq 1$. Thus, $\sup_{\mathbf{x} \in \mathscr{X}} [p(\mathbf{x})/q(\mathbf{x})] \geq 1$, with equality trivially when $p = q$. □

Proposition (1). *Under Assumption 3, given $f \in \mathscr{F}, \widehat{g}$ in (2) and g in (1),*

$$\varepsilon_T(f \circ g) \leq \underbrace{\underbrace{\sup_{\mathbf{z} \sim p(Z)} [\frac{p_S(Z_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}]}_{\text{Imputation error on S } (IS)} \times \underbrace{\sup_{\mathbf{z} \sim p(Z)} [\frac{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}]}_{\text{Transfer error of Imputation } (TI)}}_{\text{Imputation error on T } (IT)} \times \varepsilon_T(f \circ \widehat{g}) \tag{11}$$

Under Lemma 1, (IT)=1 is the minimal value reached when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$. In this case, $\varepsilon_T(f \circ g) = \varepsilon_T(f \circ \widehat{g})$.

Proof. We denote $f_{\widehat{T}}^z$, the latent target labeling function. Moreover, for simplicity, we write $h_{\widehat{g}} = f \circ \widehat{g}$, $h_g = f \circ g$ and $\forall \mathbf{z} \sim p(Z), S_D(\mathbf{z}) = \dfrac{p_D(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_D(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}$

$$\begin{aligned}
\varepsilon_T(h_g) &= \mathbb{E}_{\mathbf{x}_T \sim p_T(X)}[\mathbb{I}(h_g(\mathbf{x}_T) \neq f_T(\mathbf{x}_T))] \\
&= \mathbb{E}_{\mathbf{z}_{\mathbf{T_1}} \sim p_T(Z_1), \mathbf{z}_{\mathbf{T_2}} \sim p_T(Z_2|Z_1)}[\mathbb{I}(f(\mathbf{z}_{\mathbf{T_1}}, \mathbf{z}_{\mathbf{T_2}}) \neq f_T^z(\mathbf{z}_{\mathbf{T_1}}, \mathbf{z}_{\mathbf{T_2}}))] \\
&= \mathbb{E}_{\mathbf{z}_{\mathbf{T_1}} \sim p_T(Z_1), \widehat{\mathbf{z}}_{\mathbf{T_2}} \sim p_T(\widehat{Z}_2|Z_1)}[\frac{p_T(Z_2 = \widehat{\mathbf{z}}_{\mathbf{T_2}}|\mathbf{z}_{\mathbf{T_1}})}{p_T(\widehat{Z}_2 = \widehat{\mathbf{z}}_{\mathbf{T_2}}|\mathbf{z}_{\mathbf{T_1}})} \mathbb{I}(f(\mathbf{z}_{\mathbf{T_1}}, \widehat{\mathbf{z}}_{\mathbf{T_2}}) \neq f_{\widehat{T}}^z(\mathbf{z}_{\mathbf{T_1}}, \widehat{\mathbf{z}}_{\mathbf{T_2}}))] \\
&\leq \sup_{\mathbf{z} \sim p(Z)} [S_T(\mathbf{z})] \mathbb{E}_{\mathbf{x}_T \sim p_T(X)}[\mathbb{I}(h_{\widehat{g}}(\mathbf{x}_T) \neq f_T(\mathbf{x}_T))] \\
&= \sup_{\mathbf{z} \sim p(Z)} [S_T(\mathbf{z})] \varepsilon_T(h_{\widehat{g}})
\end{aligned}$$

However, $\forall \mathbf{z} \in \mathscr{Z}, S_T(\mathbf{z})$ cannot be computed as there is not supervision possible on T. We will instead apply Assumption 3 and use source data for which we can compute $S_S(\mathbf{z})$.

$$\begin{aligned}
\forall \mathbf{z} \in \mathscr{Z} \; S_T(\mathbf{z}) &= \frac{p_T(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \\
&= \frac{p_S(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} && \text{Assumption 3} \\
&= \frac{p_S(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \times \frac{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \\
&= S_S(\mathbf{z}) \times \frac{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}
\end{aligned}$$

Thus by applying sup,

$$\sup_{\mathbf{z} \sim p(Z)} [S_T(\mathbf{z})] = \sup_{\mathbf{z} \sim p(Z)} [S_S(\mathbf{z})] \times \sup_{\mathbf{z} \sim p(Z)} [\frac{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}]$$

This yields (11).

If (IT)=1 when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$ per Lemma 1, then $S_T(\mathbf{z}) = 1$ and $\varepsilon_T(f \circ g) = \varepsilon_T(f \circ \widehat{g})$. □

Proposition (2). *Assume a joint distribution $p_{\widetilde{T}}(X,Y)$ where $p_{\widetilde{T}}(X) = p_T(X)$ and $Y = h_{\widehat{g}}(X)$ where $h_{\widehat{g}} = f \circ \widehat{g} \in \mathscr{H}_{\widehat{g}}$ is a candidate hypothesis. Then,*

$$\lambda_{\mathscr{H}_{\widehat{g}}} \leq \min_{h_{\widehat{g}} \in \mathscr{H}_{\widehat{g}}} \left[\varepsilon_S(h_{\widehat{g}}) + \varepsilon_{\widetilde{T}}(h_{\widehat{g}}) + \varepsilon_T(f_{\widetilde{T}})\right]$$

with $\varepsilon_T(f_{\widetilde{T}}) = \Pr_{\mathbf{x} \sim p_T(X)}(f_{\widetilde{T}}(\mathbf{x}) \neq f_T(\mathbf{x}))$ the error of the pseudo-labelling function $f_{\widetilde{T}}$ on T.

Proof. We know that $p_{\widetilde{T}}(X) = p_T(X)$ as instances are not changed by applying the pseudo-labelling function. Thus, given $h_{\widehat{g}} \in \mathscr{H}_{\widehat{g}}$

$$\varepsilon_T(h_{\widehat{g}}) = \varepsilon_T(h_{\widehat{g}}, f_T) = \varepsilon_{\widetilde{T}}(h_{\widehat{g}}, f_T)$$

Applying the triangle inequality for classification error [13],

$$\varepsilon_{\widetilde{T}}(h_{\widehat{g}}, f_T) \leq \varepsilon_{\widetilde{T}}(h_{\widehat{g}}, f_{\widetilde{T}}) + \varepsilon_{\widetilde{T}}(f_{\widetilde{T}}, f_T)$$

Finally, we can rewrite $\varepsilon_{\widetilde{T}}(h_{\widehat{g}}, f_{\widetilde{T}}) = \varepsilon_{\widetilde{T}}(h_{\widehat{g}})$ and $\varepsilon_{\widetilde{T}}(f_{\widetilde{T}}, f_T) = \varepsilon_T(f_{\widetilde{T}}, f_T) = \varepsilon_T(f_{\widetilde{T}})$. □

D Dataset description

D.1 Digits

We scale all images to 32×32 and normalize the input in $[-1, 1]$. When adaptation involves a domain with three channels (SVHN or MNIST-M) and a domain with a single channel, we simply triplicate the channel of the latter domain. As in [14] we use balanced source batches which proves to increase performance especially when the source dataset is imbalanced (e.g. SVHN and USPS datasets) while the target dataset (usually MNIST derived) is balanced. Scaling the input images enables us to use the same architecture across datasets. In practise the embedding size is 2048 after preprocessing. For missing versions, we set pixel values to zero in a given patch as shown in Figure 3. The `digits` datasets are provided with a predefined train / test split. We report accuracy results on the target test set and use the source test set as validation set (Section E.2). The number of instances in each dataset is reported in Table 5. We run each model five times.

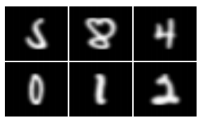
	USPS	MNIST	SVHN	MNIST-M
Train	7438	60k	73257	60k
Test	1860	10k	26032	10k
Size	28×28	28×28	32×32	28×28
Channels	1	1	3	3
				

Table 5: Statistics on `digits` datasets

D.2 Amazon

Each domain has around 2000 samples and we use features freely available at https://github.com/jindongwang/transferlearning/tree/master/data#amazon-review which follows

the data processing pipeline in [10]. Each review is preprocessed as a feature vector of unigrams and bigrams keeping only the 5000 most frequent features. In practise, we consider the dense version of these features after projection onto a low-dimensional sub-space of dimension 400 with PCA as in [10]. Datasets with missing features are built by setting the first half of the features to 0.

D.3 Ads

Table 6 lists statistics on the traffic for the two `ads` datasets; we now describe how they are preprocessed. On both datasets the train and test sets are fixed. We run each model five times.

Dataset	`ads-kaggle`				`ads-real`			
Domain	Source		Target		Source		Target	
Split	`Train`	`Test`	`Train`	`Test`	`Train`	`Test`	`Train`	`Test`
Positive	246.872	61.841	92.333	22.943	X	X	X	X
Negative	699.621	174.783	854.160	213.681	X	X	X	X
Total	946.493	236.624	946.493	236.624	24.465.756	3.760.233	819.073	147.358
$p(Y=1)$	0,2608	0,2613	0,0976	0,0970	X	X	X	X

Table 6: Statistics on `ads` datasets

`ads-kaggle` The Criteo Kaggle dataset is a reference dataset for CTR prediction and groups one week of log data. The objective is to model the probability that a user will click on a given ad given his browsing behaviour. Positives refer to clicked ads and negatives to non-clicked ads. For each datum, there are 13 continuous and 26 categorical features. We divide the traffic into two domains using feature number 30 corresponding to an engagement feature; for a given value for this categorical feature, all instances have a single missing numeric feature (feature number 5). We then construct an artificial dataset simulating transfer between known and new users. We process the original Criteo Kaggle dataset to have an equal number of source and target data. We then perform train / test split on this dataset keeping 20% of data for testing. We used in our experiments only continuous features; to show the benefit of modelling additional missing features, we extend the missing features list to features 1, 5, 6, 7, 11 and 12 by setting them to zero on the target domain. After these operations, 6 features are missing and 7 are non-missing. Preprocessing consists in normalizing continuous features using a log transform.

`ads-real` This private dataset is similar to `ads-kaggle`. We filter out non-clicks and the final task is to model the sale probability for a clicked ad given an user's browsing history. Positives refer to clicked ads which lead to a sale; negatives to clicked ads which did not lead to a sale. We use one week of sampled logs as a training set and use the following day data as the test set. This train / test definition is used so to better

correlate with the performance of a production model. Features are aggregated across user timelines and describe the clicking and purchase behavior of a user. In comparison to `ads-kaggle` more continuous features are used. The count features can be User-centric i.e. describe the global activity of the user (number of clicks, displays, sales done globally across partners) or User-partner features i.e. describing the history of the user on the given partner (number of clicks, sales... on the partner). The latter are missing for prospecting users. Counts are aggregated across varying windows of time and categories of partner catalog. We bucketize these count features using log transforms and project the features into an embedding space of size 596 with 29 features. 12 features are missing and 17 are non-missing.

Domain	Source	Target
feature 1	0.80±2.21	4.4×10^{-4}±0.041
feature 2	9.16±13.04	9.01±13.42
feature 3	4.40±6.32	3.44±6.19
feature 4	2.58±3.27	0.94±2.31
feature 5	61.09±37.67	0.0±0.0
feature 6	11.26±12.24	0.090±1.69
feature 7	4.10±6.23	0.0034±0.13
feature 8	5.12±4.50	1.91±4.26
feature 9	14.32±11.57	3.273±5.36
feature 10	0.046±0.22	1.35×10^{-5}±0.0037
feature 11	1.08±2.11	4.25×10^{-4}±0.029
feature 12	0.083±0.78	6.68×10^{-5}±0.018
feature 13	2.74±3.59	1.21±3.36

Table 7: Feature mean and standard deviation on `ads-kaggle`. We set features 1, 6, 7, 11, 12 to zero on T.

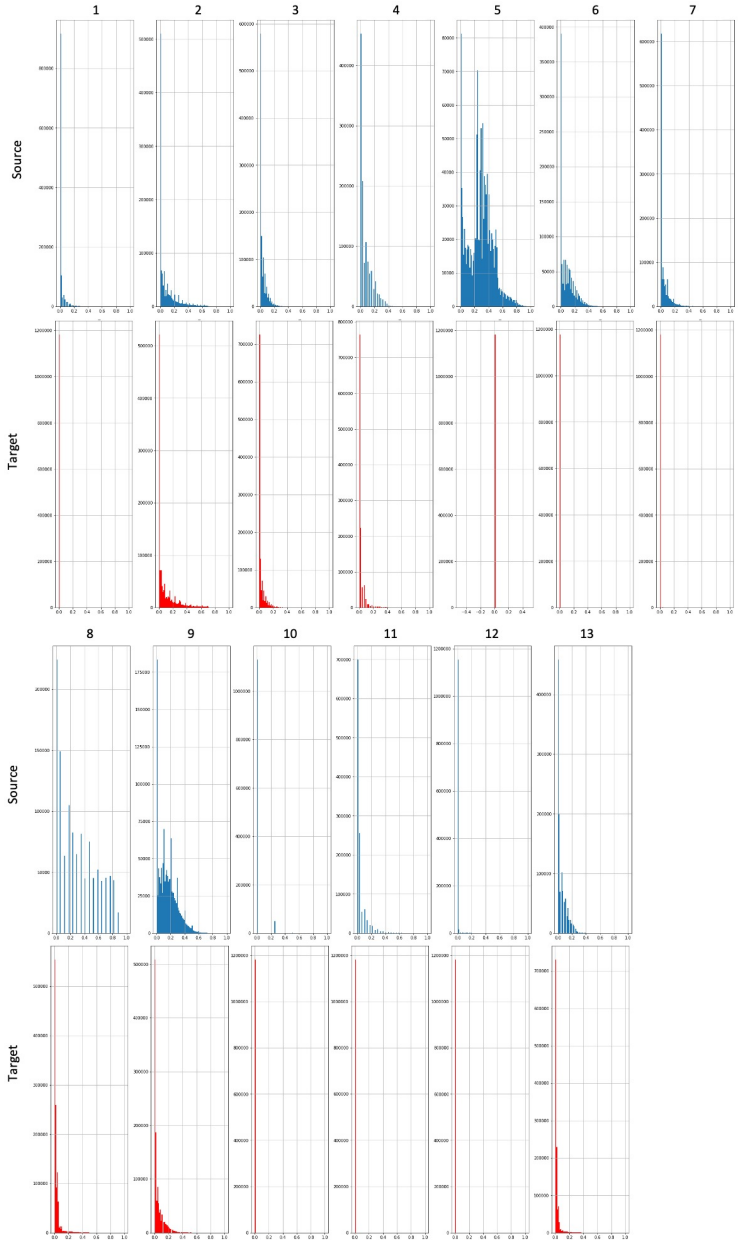


Fig. 6: Source (blue) and Target (red) distributions on `ads-kaggle` for each feature (1 to 13)

E Implementation details

E.1 Neural Net architecture

digits We use the ADV and OT versions of our imputation model. For ADV models, we use the DANN model description in [18]; for OT we use the DeepJDOT model description in [14]. Both models can be considered as simplified instances of our corresponding ADV and OT imputation models when no imputation is performed. Performance of the adaptation models is highly dependent on the NN architectures used for adaptation and classification. In order to perform fair comparisons and since our goal is to evaluate the potential of joint Adaptation-imputation-classification, we selected these architectures through preliminary tests and use them for both the ADV and OT models. The two models are described below and illustrated in Figure 7.

- Feature extractors g_1 and g_2 consists of three convolutional layers with 5×5 kernel and 64 filters interleaved with max pooling layers with a stride of 2 and 2×2 kernel. The final layer has 128 filters. We use batch norm on convolutional layers and ReLU as an activation after max pooling. As in [14] we find that adding a sigmoid activation layer as final activation is helpful.
- Classifier f consists of two fully connected layers with 100 neurons with batch norm and ReLU activation followed by the final softmax layer. We add Dropout as an activation for the first layer of the classifier.
- Discriminator D_1 and D_2 is a single layer NN with 100 neurons, batch norm and ReLU followed by the final softmax layer. On USPS $\rightarrow$ MNIST and MNIST $\rightarrow$ USPS dataset we use a stronger discriminator network which consists of two fully connected layers with 512 neurons.
- Generator r consists of two fully connected layers with 512 neurons, batch norm and ReLU activation. This architecture is used for ADV and OT imputation models. In practice using wider and deeper networks increases classification performance with the more complicated classification tasks (SVHN $\rightarrow$ MNIST, MNIST $\rightarrow$ MNIST-M); in these cases we add an additional fully connected network with 512 neurons. The final activation function is a sigmoid.

We use the same architecture described above for all our models to guarantee fair comparison. As a side note, the input to the imputation model's classifier is twice bigger as in the standard adaptation models.

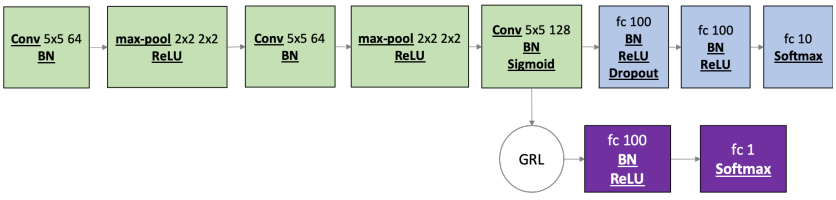


Fig. 7: Base architecture for the ADV DANN model

ads-kaggle and amazon We experiment with ADV models only. As input data is numeric and low dimensional, architectures are simpler than in digits. Our feature extractor is a three layered NN with 128 neurons and with a final sigmoid activation. The classifier is taken to be a single layered NN with 128 neurons and a final softmax layer. Activations are taken to be ReLUs. The domain discriminator is taken to be a two layered NN with 128 neurons and a final softmax layer. Finally the reconstructor is taken to be a two-layered NN with 256 neurons and final sigmoid activation.

ads-real We experiment with ADV models only. Input features after processing are fed directly into the feature extractors g_1, g_2 consisting of two fully connected layers with 128 neurons. The classifier and discriminator is taken to be single-layered NN with 25 neurons. The reconstructor is taken to be a two-layered NN with 128 neurons. Inner activations are taken to be ReLUs and the final activation of the feature extractor is taken to be a sigmoid.

E.2 Network parameters

Hyperparameter tuning Tuning hyperparameters for UDA is tricky as we do not have access to target labels and thus cannot choose parameters minimizing the target risk on a validation set. Several papers set hyperparameters through reverse cross-validation [18]. Other approaches developed for model selection are based on risk surrogates obtained by estimating an approximation of the risk value on the source based on the similarity of source and target distributions (without the labels). In the experiments, we used a recent estimator, Deep Embedded Validation (DEV) [47] for tuning the initial learning rate and for the OT imputation model, tuning λ_1 and λ_{OT}. For other parameters, we used heuristics and typical hyperparameter values from UDA papers (such as batch size) without further tuning. We use a cross entropy link function on the source validation set; this value provides a proxy for the target test risk. Using parameters from the original paper, this estimator helps select parameter ranges which perform reasonably well. We keep the estimator unchanged for our baseline models. In the imputation case, the discriminator used for computing importance sampling weights discriminates between $\widehat{\mathbf{z}}_\mathbf{S}$ and $\widehat{\mathbf{z}}_\mathbf{T}$ i.e. D_1 (Figure 2).

Digits We find that the results are highly dependent on the NN architecture and the training parameter setting. In order to evaluate the gain obtained with *Adaptation-Imputation*, we use the same NN architecture for all models (ADV and OT) but fine tune the learning rates for each model using the DEV estimator (other parameters do not have a significant impact on the classification performance).

ADV We use an adaptive approach as in [18] for decaying the learning rate lr and updating the gradient's scale s between 0 and 1 for the domain discriminators. We choose the decay values used in [18] ie. $s = \dfrac{2}{1+\exp(-10\times p)} - 1$ and $lr = \dfrac{lr_i}{(1+10\times p)^{0,75}}$ where p is ratio of current batches processed over the total number of batches to be processed without further tuning. We tune the initial learning rate lr_i, chosen in the range

$\{10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5}, 10^{-4}\}$ following Section E.2. In practise we take $lr_i = 10^{-2}$ for `ADV` *Adaptation-Imputation*, *Adaptation-Full*, *Adaptation-IgnoreComponent* and $lr_i = 10^{-2.5}$ for `ADV` *Adaptation-ZeroImputation*. We use Adam as the optimizer with momentum parameters $\beta_1 = 0.8$ and $\beta_2 = 0.999$ and use the same decay strategy and initial learning rate for all components (feature extractor, classifier, reconstructor). Batch size is chosen to be 128; we see in practise that initializing the adaptation models with a source model with smaller batch size (such as 32) can be beneficial.

`OT` We choose parameter $\lambda_{OT} = 0.1$ in Equation 16 after tuning in the range $\{10^{-1}, 10^{-2}, 10^{-3}\}$ using DEV. We weight L_1 in Equation 8 by $\lambda_1 = 0.1$. Following [14], batch size is taken to be 500 and we use EMD a.k.a. Wasserstein-2 distance. We initialize adaptation models with a source model in the first 10 epochs and divide the initial learning rate by two as adaptation starts for non-imputation models. For *Adaptation-Imputation* we follow a decaying strategy on the learning rate and on the adaptation weight as explained in the next item. We choose lr_i in the range $\{10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5}, 10^{-4}\}$. In practise we fix $lr_i = 10^{-2}$ for all models.

Imputation parameters Ablation studies are conducted in Section 6.6 on weights in Equation 4; in `digits` experiments we choose $L_2 = L_{MSE} + L_{ADV}$ for `ADV` and `OT` to reduce the burden of additional feature tuning. For `ADV` model, we fix $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in Equation 8. In the `OT` model, we vary λ_1 between 0 and 0.1 and λ_2 between 0 and 1 following the same schedule as the gradient scale update for `ADV` models to reduce variance.

Ads We use an adaptive strategy for updating the gradient scale and the learning rate with the same parameters as in the `digits` dataset. Optimizer is taken to be Adam. Batch size is taken to be big so that target batches include sufficient positive instances.

`ads-kaggle` The initial learning rate is chosen in the range $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$ using DEV and fixed to be 10^{-6} for all models. Batch size is taken to be 500 and we initialize models with a simple classification loss for five epochs. We run models for 50 epochs after which we notice that models reach a plateau. We find that adding a weighted MSE term allows to achieve higher stability (as measured by variance) as further studied in Section 6.6. In a similar fashion to [34], we tune this weight in the range $\{1, 10^{-1}, 10^{-2}, 7.5 \times 10^{-3}, 5 \times 10^{-3}, 10^{-3}\}$. We find that 0.005 offers the best compromise between mean loss and variance. Moreover on this dataset we use a faster decaying strategy for the discriminator's D_2 and the reconstructor's r learning rate, $lr = \dfrac{lr_i}{(1 + 30 \times p)^{0,75}}$ to achieve higher stability in the training curves while the feature extractor g_1, g_2 and D_1's learning rate are unchanged.

`ads-real` The initial learning rate is chosen in the range $\{10^{-4}, 10^{-5}, 10^{-6}\}$ and fixed to be 10^{-6} for all models. The learning rate is decayed with the same parameters as `digits` for all models. We run models for ten epochs which provides a good trade-off between learning time and classification performance. Batch size is taken to be 500. We choose $L_2 = L_{MSE} + L_{ADV}$ without further tuning; this achieves already good results.

E.3 Amazon

We use the same hyperparameters as `ads-kaggle`. λ_{MSE} is set to 1 without further tuning.

F Latent space visualization on `digits`

In this section we visualize the embeddings $\widehat{\mathbf{z}} = \widehat{g}(\mathbf{z})$ learned by the various models on `digits` by projecting the embeddings in a 2D space using $\widehat{g}$ with t-SNE (the original embedding size being 2048). Figure 8 represents the embeddings learned for `ADV` models on MNIST $\rightarrow$ MNIST-M. Figures 9 and 10 represent these embeddings for `OT` models respectively on MNIST $\rightarrow$ MNIST-M and MNIST $\rightarrow$ USPS. On these figures, we see that *Adaptation-Imputation* generates feature representations that overlap better between source and target examples per class than the adaptation counterparts (although *Adaptation-IgnoreComponent* does a good job at overlapping feature representations). This correlates with the accuracy performance on the target test set. Moreover we notice, as expected, that *Adaptation-IgnoreComponent* and *Adaptation-ZeroImputation* perform badly compared to *Adaptation-Full* which justifies the use of *Adaptation-Imputation* when confronted to missing non-stochastic data.

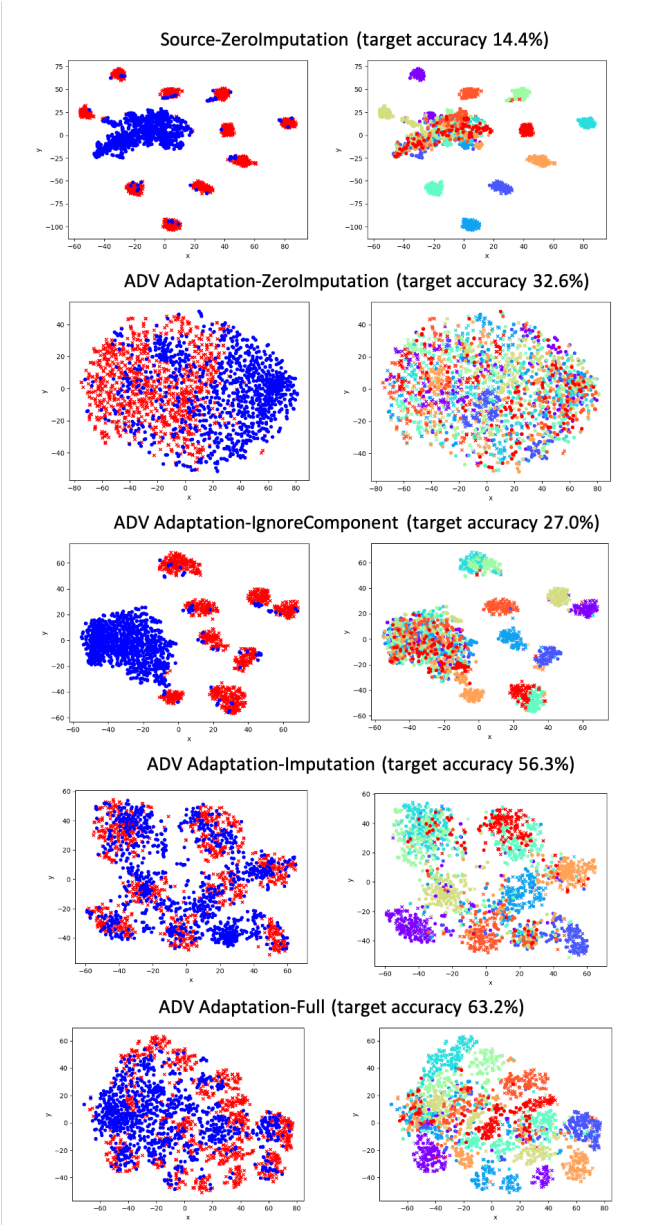


Fig. 8: Embeddings for MNIST → MNIST-M dataset for ADV models on a batch. Figures on the left represent the source (red) and target (blue) clusters; Figures on the right represent the classes on source and target.

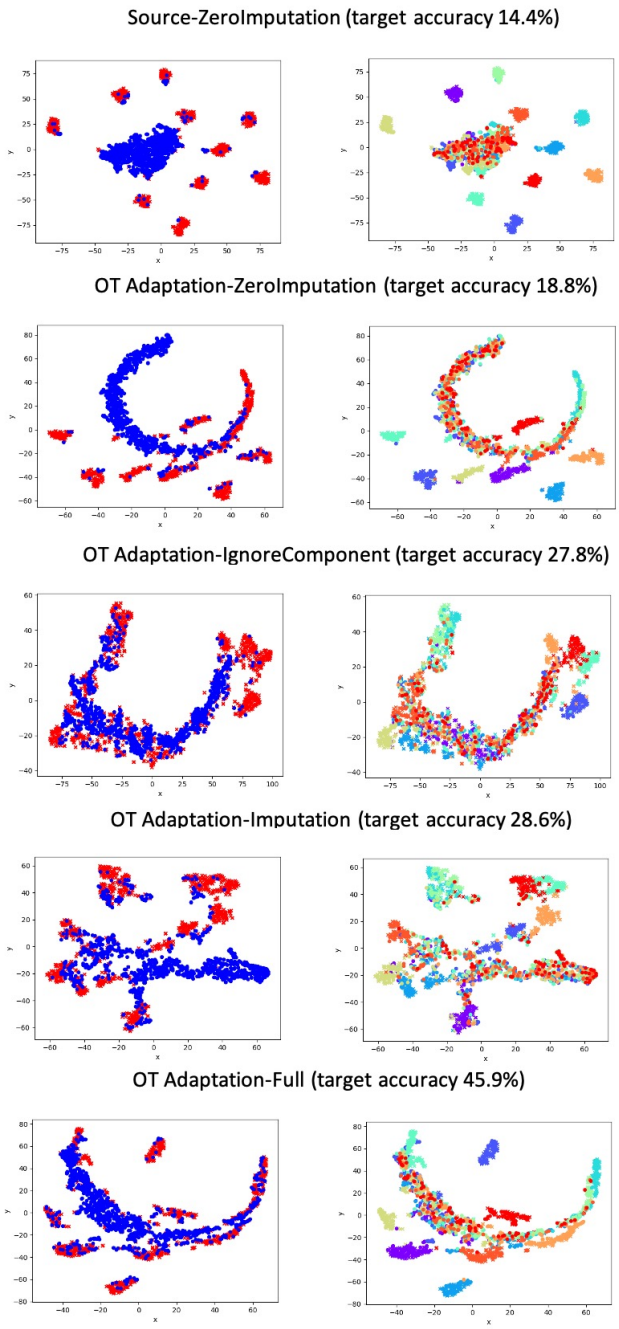


Fig. 9: Embeddings for MNIST → MNIST-M dataset for `OT` models on a batch. Figures on the left represent the source (red) and target (blue) clusters; Figures on the right represent the classes on source and target.

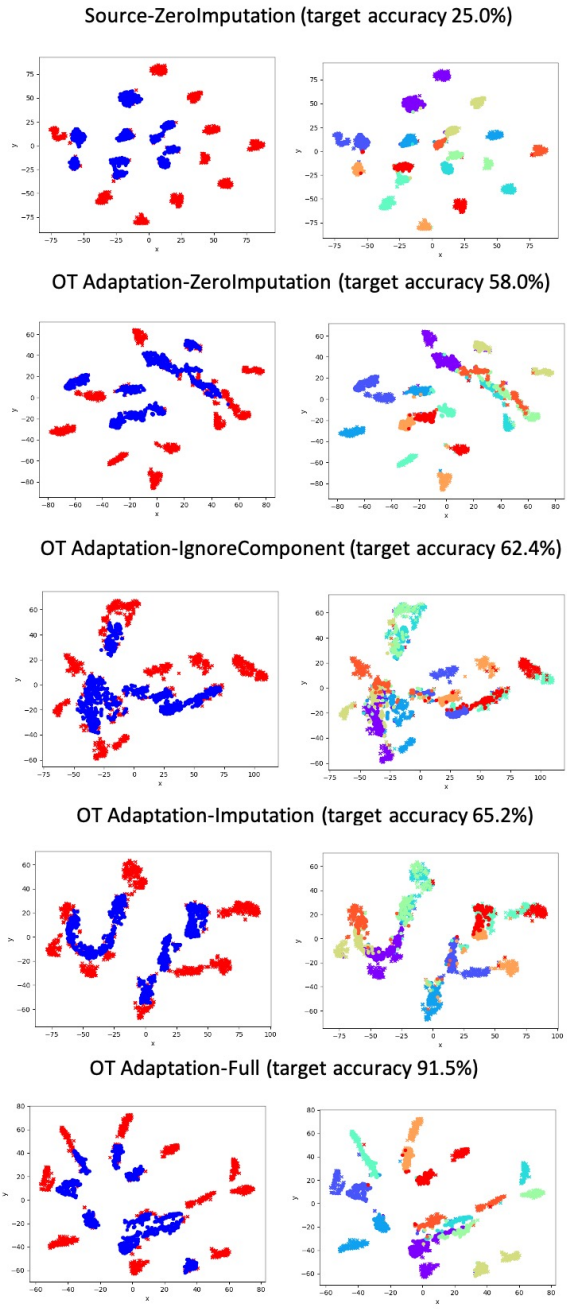


Fig. 10: Embeddings for MNIST → USPS dataset for `OT` models on a batch. Figures on the left represent the source (red) and target (blue) clusters; Figures on the right represent the classes on source and target.