A Expanded related works

In this section, we further expand on the related works described in Section 2.

Additional methods that leverage multiple datasets for causal inference. While several other methods propose leveraging multiple datasets to improve CATE estimation, these focus on combining observational data with randomized data from clinical trials with the aim of handling the bias from hidden confounders \[55 \text{–} 57\]. Moreover, all of these methods learn potential outcome functions that are shared between the different domains. Conversely, we assume that there are no hidden confounders and propose an approach that can share information from a source domain to improve CATE estimation on a target domain of interest.

A different strand of work aims to address the problem of identifiability of causal effects for an observational dataset of interest by leveraging data from other datasets. In particular, \[58\] investigates how to transfer average treatment (causal) effects obtained from experimental data (such as randomized clinical trials) to an observed population that may have different distribution of covariates, treatments and outcomes and where the causal relationship of interest cannot be identified using only the observational data. The paper assesses under which conditions such average causal effects can be transported according to the differences between the randomized and observational data. The authors also provide a brief discussion of how to transfer these average causal estimates between observational datasets. Alternatively, \[59\] aims to help identify the average effects of interventions on a target population of interest by integrating multiple types of auxiliary data: data from a randomized study on the same population, data from an observational study on the same population, selection biased data from the same population and data from a randomized study from a different population. All of the auxiliary datasets have the same set of features as the target dataset. Both \[58\], \[59\] consider this transportability problem of average treatment effects in the context of causal diagrams. This is different from our set-up for CATE estimation where we assume that the potential outcomes are identifiable in both the source and target domains. Moreover, we also handle the case of heterogeneous feature spaces and only assume access to a source domain larger than a target domain, without putting any restrictions on whether this data is experimental or observational.

Transfer learning and domains adaptation in the predictive setting. Another method loosely related to ours is the one of \[60\], which considers the case of having both heterogeneous feature and label spaces in the context of natural language processing and proposes a method that learns a common embedding for the features and labels and then builds a mapping between them. In this paper, we consider a shared label space. In addition, \[61\] uses a causal approach to address the problem of domain adaptation for the predictive setting, where it considers labelled data in one or more source domains, unlabelled data in the target domain, same feature spaces between the source and target domains and aims to learn predictive functions that are invariant to the changes between domains to be able to reliably estimate the outcomes in the target domain. \[61\] proposes tackling this unsupervised domain adaptation by modelling the different distributions between the source and target domains as different contexts of a single underlying system (where the context represents interventions causing the distribution shifts between domains, such that the source and target domains come from different interventions).

Table 2 provides a comparison of our method with the most relevant related works. Note that our heterogeneous transfer learning approach is designed to share information between one source domain and a target domains. Moreover, we assume access to labels on the target domain and build target-specific outcome functions that share a common structure with the source domain.
where and

Similarly to computing the corresponding loss function. Refer to Section 5.1 and Appendix B for architectural

C Pseudo-code for training our proposed HTCE-TARNet. Note that a similar training

procedure can be used for the HTCE-S-Learner, HTCE-propensity estimator and in the second stage

provides the pseudo-code for training our proposed HTCE-TARNet. Note that a similar training

B Heterogeneous transfer propensity estimator

For our HTCE-propensity estimator that is used as part of the two-step meta-learners (e.g. HTCE-

DR-learner) we build a model that shares information between the treatment assignment mech-

anisms in the source and target datasets as illustrated in Figure 8. In particular, our HTCE-

DR-learner) we build a model that shares information between the treatment assignment mech-

anisms in the source and target datasets as illustrated in Figure 8. In particular, our HTCE-

HTCE-learners

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Heterogeneous domains</th>
<th>Heterogeneous feature spaces</th>
<th>Access to labels on target domain outcome functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATE learners [10][12][15]</td>
<td>1</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Johansson et al. [36]</td>
<td>2</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Shi et al. [27]</td>
<td>N</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Kunzel et al. [35]</td>
<td>2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Learning across domains</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ganin et al. [33]</td>
<td>2</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Magliacane et al. [61]</td>
<td>N</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rudner et al. [41]</td>
<td>N</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Yoon et al. [24]</td>
<td>N</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Moon et al. [72]</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Ours) HTCE-learners</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2: Comparison of our proposed method with related works.

C Pseudo-code for the HTCE-learners

Algorithm[1] provides the pseudo-code for training our proposed HTCE-T-learner and Algorithm[2]

provides the pseudo-code for training our proposed HTCE-TARNet. Note that a similar training

procedure can be used for the HTCE-S-Learner, HTCE-propensity estimator and in the second stage

of the two-step learners (e.g. HTCE-DR-learner). The main differences are in the encoder used, in
terms of sharing layers between PO functions within each domain and the outcome used for
computing the corresponding loss function. Refer to Section[5][1] and Appendix[3] for architectural
details. Similarly to [18], we do not use cross-fitting for the two-step learners (HTCE-DR-learner).
Refer to [13] for more details. For simplicity, we set the mini-batch size to be the same in both the
source and target datasets: \(B_T = B_R = B\). Moreover, note that in practice, we multiply \(\mathcal{L}_{orth_x}\)
and \(\mathcal{L}_{orth_y}\) by 0.01 to ensure that all losses have a similar scale and we use early stopping on the
validation dataset from the target domain to check for convergence.

Figure 8: HTCE-propensity estimator.
Algorithm 1 Pseudo-code for training the HTCE-T-learner.

1: Input: Dataset with expert demonstrations from source \(D^R\) and target \(D^T\) domains, learning rate \(\lambda\), mini-batch size for source dataset \(B_R\), mini-batch size for target dataset \(B_T\).
2: Initialize: \(\theta\) (all parameters of the HTCE-T-Learner).
3: while not converged do
4: Sample mini-batch of \(B_R\) demonstrations from the source dataset \((x_i^R, w_i, y_i) \sim D^R\) and a minibatch of \(B_T\) demonstrations from the target dataset \((x_i^T, w_i, y_i) \sim D^T\).
5: Handle heterogeneous feature spaces:
6: for \(i = 1 \ldots B_R\) do \(\triangleright\) Process batch from source domain.
7: \[ z_i^R = \phi^s(w_i(x_i)), \] where \(x_i^s\) is part of \(x_i^R\); \(z_i^R = \phi^R(w_i(x_i))\)
8: Set: \(\zeta^s = \left[z_1^R \ldots z_B^R\right]^T\), \(\zeta^P = \left[z_1^P \ldots z_B^P\right]^T\).
9: for \(i = 1 \ldots B_T\) do \(\triangleright\) Process batch from target domain.
10: \[ z_i^T = \phi^s(w_i(x_i)), \] where \(x_i^s\) is part of \(x_i^T\); \(z_i^P = \phi^R(w_i(x_i))\)
11: Set: \(\zeta^s = \left[z_1^T \ldots z_B^T\right]^T\), \(\zeta^P = \left[z_1^P \ldots z_B^P\right]^T\).
12: Compute orthogonal regularization loss for the feature representations:
\[ \mathcal{L}_{\text{orth}} = ||\zeta^s\zeta^s^T||_F^2 + ||\zeta^s\zeta^P^T||_F^2. \]
13: Share information between PO functions across domains:
14: for \(i = 1 \ldots B_R\) do \(\triangleright\) Process batch from source domain.
15: for \(l = 1 \ldots L\) do
16: if \(l \equiv 1\) then
17: \[ h_{w_i,1}^P = [z_1^R \ldots z_B^R]^T\], \(\tilde{h}_{w_i,1}^P = \tilde{h}_{w_i,1}^P\)
18: else
19: \[ h_{w_i,1}^P = [h_{w_i,1-1}^P]^T\], \(\tilde{h}_{w_i,1}^P = [h_{w_i,1-1}^P]^T\).
20: \[ h_{w_i,l}^P = \text{Shared}_l(w_i, (\tilde{h}_{w_i,l}), h_{w_i,l}^P = \text{Private}_l(w_i, (\tilde{h}_{w_i,l}), h_{w_i,l}^P)\]
21: \[ y_i^R = \psi(h_{w_i,L}^P + h_{w_i,L})\]
22: for \(i = 1 \ldots B_T\) do \(\triangleright\) Process batch from target domain.
23: for \(l = 1 \ldots L\) do
24: if \(l \equiv 1\) then
25: \[ h_{w_i,1}^T = [z_1^T \ldots z_B^T]^T\], \(\tilde{h}_{w_i,1}^T = \tilde{h}_{w_i,1}^T\)
26: else
27: \[ h_{w_i,1}^T = [h_{w_i,1-1}^T]^T\], \(\tilde{h}_{w_i,1}^T = [h_{w_i,1-1}^T]^T\).
28: \[ h_{w_i,l}^T = \text{Shared}_l(w_i, (\tilde{h}_{w_i,l}), h_{w_i,l}^T = \text{Private}_l(w_i, (\tilde{h}_{w_i,l}), h_{w_i,l}^T)\]
29: \[ y_i^T = \psi(h_{w_i,L}^P + h_{w_i,L})\]
30: Compute outcome loss:
\[ \mathcal{L}_y = \sum_{i=1}^{B_R} l(y_i, \hat{y}_i^R) + \sum_{i=1}^{B_T} l(y_i, \hat{y}_i^T) \]
31: Compute orthogonal regularization loss for the PO shared and private layers:
\[ \mathcal{L}_{\text{orth}_p} = \sum_{w \in \{0,1\}} \sum_{l=1}^{L} \|\theta_{w,l}^s\theta_{w,l,1:m_{l-1}^T}^{P_R}\|_F^2 + \|\theta_{w,l}^s\theta_{w,l,1:m_{l-1}^T}^{P_P}\|_F^2. \]
32: Parameters update:
33: \(\theta \leftarrow \theta - \lambda \nabla_\theta (\mathcal{L}_y + \mathcal{L}_{\text{orth}_p} + \mathcal{L}_{\text{orth}_p})\)
34: Output: Learnt parameters \(\theta\)
Algorithm 2 Pseudo-code for training the HTCE-TARNet.

1: Input: Dataset with expert demonstrations from source $\mathcal{D}^R$ and target $\mathcal{D}^T$ domains, learning rate $\lambda$, mini-batch size for source dataset $B_R$, mini-batch size for target dataset $B_T$
2: Initialize: $\theta$ (all parameters of the HTCE-TARNet)
3: while not converged do
4:   Sample mini-batch of $B_R$ demonstrations from the source dataset $(x^R_i, w_i, y_i) \sim \mathcal{D}^R$ and a minibatch of $B_T$ demonstrations from the target dataset $(x^T_i, w_i, y_i) \sim \mathcal{D}^T$.
5:   Handle heterogeneous feature spaces:
6:   for $i = 1 \dots B_R$ do \hspace{1em} \triangleright Process batch from source domain.
7:   $z^R_i = \phi^R(x^R_i)$, where $x^R_i$ is part of $x^R_i$; $z^R_i = \phi^R(x^R_i)$
8:   Set: $\zeta^R = [z^R_1 \cdots z^R_{B_R}]^T$, $\zeta^R = [z^R_1 \cdots z^R_{B_R}]^T$
9:   for $i = 1 \dots B_T$ do \hspace{1em} \triangleright Process batch from target domain.
10: $z^T_i = \phi^T(x^T_i)$, where $x^T_i$ is part of $x^T_i$; $z^T_i = \phi^T(x^T_i)$
11: Set: $\zeta^T = [z^T_1 \cdots z^T_{B_T}]^T$, $\zeta^T = [z^T_1 \cdots z^T_{B_T}]^T$
12: Compute orthogonal regularization loss for the feature representation:
13: $\mathcal{L}_{orth} = ||\zeta^R \zeta^T||_F^2 + ||\zeta^T \zeta^T||_F^2$.

14: Share information between PO functions within each domain:
15: for $i = 1 \dots B_R$ do \hspace{1em} \triangleright Process batch from source domain.
16: $z^R_i = \Omega^R \left( [z^R_i \| z^R_i] \right)$
17: for $i = 1 \dots B_T$ do \hspace{1em} \triangleright Process batch from target domain.
18: $z_i = \Omega^T \left( [z^T_i \| z^T_i] \right)$

19: Share information between PO functions across domains:
20: for $i = 1 \dots B_R$ do \hspace{1em} \triangleright Process batch from source domain.
21: for $l = 1 \dots L$ do
22: if $l == 1$ then
23: $\hat{h}_{w_i,1}^{PR} = \hat{z}^R_i$; $\hat{h}_{w_i,1}^{PR} = \hat{z}^R_i$
24: else
25: $\hat{h}_{w_i,1}^{PR} = [h_{w_i,1-1} \| h_{w_i,1-1}]$; $\hat{h}_{w_i,1}^{PR} = [h_{w_i,1-1}]$
26: $\hat{h}_{w_i,1}^{s} = \text{Shared}_L \hat{h}_{w_i,1}^{PR}$; $\hat{h}_{w_i,1}^{PR} = \text{Private}_L \hat{h}_{w_i,1}^{PR}$
27: $\hat{y}_i = \psi(h_{w_i,L}^{PR} + h_{w_i,L}^{s})$
28: for $i = 1 \dots B_T$ do \hspace{1em} \triangleright Process batch from target domain.
29: for $l = 1 \dots L$ do
30: if $l == 1$ then
31: $\hat{h}_{w_i,1}^{PT} = \hat{z}^T_i$; $\hat{h}_{w_i,1}^{PT} = \hat{z}^T_i$
32: else
33: $\hat{h}_{w_i,1}^{PT} = [h_{w_i,1-1} \| h_{w_i,1-1}]$; $\hat{h}_{w_i,1}^{PT} = [h_{w_i,1-1}]$
34: $\hat{h}_{w_i,1}^{s} = \text{Shared}_L \hat{h}_{w_i,1}^{PT}$; $\hat{h}_{w_i,1}^{PT} = \text{Private}_L \hat{h}_{w_i,1}^{PT}$
35: $\hat{y}_i^T = \psi(h_{w_i,L}^{PT} + h_{w_i,L}^{s})$
36: Compute outcome loss:
37: $\mathcal{L}_y = \sum_{i=1}^{B_R} l(y_i, \hat{y}_i^R) + \sum_{i=1}^{B_T} l(y_i, \hat{y}_i^T)$
38: Compute orthogonal regularization loss for the PO shared and private layers:
39: $\mathcal{L}_{orth} = \sum_{w \in \{0,1\}} \sum_{l=1}^{L} \|\theta_{w,l}^s \theta_{w,l,1:m_{l-1}^s}^{PR} \|_F^2 + \|\theta_{w,l}^T \theta_{w,l,1:m_{l-1}^T}^{PT} \|_F^2$
40: Parameters update:
41: $\theta \leftarrow \theta - \lambda \nabla_{\theta} (\mathcal{L}_y + \mathcal{L}_{orth} + \mathcal{L}_{orth})$
42: Output: Learnt parameters $\theta$
D Experimental details

D.1 Dataset description

In this section, we provide a description of the three datasets used for evaluation: MAGGIC [45], Twins [43] and TCGA [44]. Note that these datasets have diverse characteristics in terms of the number of features and the proportion of categorical/continuous features. To the best of our knowledge, none of the medical datasets used contain personally identifiable information nor offensive content. Given that Twins and TCGA are datasets from a single domain, we then explain how we used them to obtain source and target datasets with heterogeneous feature spaces.

MAGGIC. MAGGIC [45] consists of datasets with heterogeneous features from 30 domains representing medical studies of patients who have experienced heart failure. Note that this medical dataset is private. Refer to [45] for details on how the dataset was collected and curated. Each medical study consists of a number of patients ranging from 190 to 13279. From these, we sample a dataset with < 500 patients to represent our target domain and a dataset with > 500 patients to represent our source domain. The different datasets in MAGGIC consist of a mixture of categorical and continuous features representing clinical covariates for patients who have experienced heart failure such as age, gender, serum creatinine, diabetes, beta-blocker prescription, lower systolic blood pressure, lower body mass, time since diagnosis, current smoker, chronic obstructive pulmonary disease, etc. The total number of features across the different medical studies in MAGGIC is 216 and the average number of features in a single study is 73. After sampling the source and target dataset from the different medical studies in MAGGIC, we simulate treatment assignments and outcomes as described in Section 6. Note that different source and target datasets are sampled for each of the different 10 random seeds used for all experimental results.

Twins. Twins [43] represents one of the standard benchmark datasets in the causal inference literature for estimating the effects of binary treatments. The dataset consists of data from 11400 pairs of twin births in the USA recorded between 1989-1991 [43]. Refer to [43] for details of how the dataset was collected and curated. We use the publicly available version of the dataset from: https://github.com/AliciaCurth/CATENets For each twin pair, 39 relevant covariates were recorded related to the parents, pregnancy and birth such as marital status, mother’s age, number of previous births, pregnancy risk factors, quality of care during pregnancy, number of gestation weeks, etc. These represent a mixture of continuous and categorical features. We obtain our full dataset with $N_F = 114000$ examples and $D_F = 39$ features by randomly sampling one of the twins to observe. We then sub-sample our source and target datasets as described below and simulate treatments and outcomes as described in Section 6.

TCGA. TCGA is a dataset consisting of cancer patients for which gene expression measurements were recorded [44]. The dataset is publicly available and has been used by previous methods in causal inference [40, 47, 62]. In particular, we use the publicly available version of the TCGA dataset as used by DRNet [47] https://github.com/d909b/drnet Refer to [44] for details of how the dataset has been collected and curated. The dataset consists of $N_F = 9659$ patients (samples), for which we use the measurements from the $D_F = 100$ most variable genes. Note that these are all continuous features. Moreover, we log-normalized the gene expression data and scale each feature in the [0, 1] interval. We then sub-sample our source and target datasets as described below. The treatment assignment and outcomes are simulated as described in Section 6.

Obtaining source and target datasets for Twins and TCGA. Unless the size of the target dataset is fixed, from the full datasets, we sample a target dataset of size $N_T \sim U(100, 500)$ and a source dataset of size $N_R \sim U(1000, N_F - N_T)$, where $N_T$ is the size of the target dataset, $N_R$ is the size of the source dataset and $N_F$ is the size of the full dataset. For the experiments where we vary the size of the target dataset, we set $N_T \in \{100, 200, 300, 500, 1000, 2000, 4000\}$ and we again sample the size of the source dataset from $N_R \sim U(1000, N_F - N_T)$. Moreover, to create heterogeneous feature spaces for the source and target domains, let $D_F$ be the number of features in the full dataset. From these, we randomly sample $D_{pt} \in U(5, D_F/3)$ features that are private for the source dataset, $D_{pt} \in U(5, D_F/3)$ features that are private for the target dataset and $D_p \in U(5, D_F/3)$ features that are shared between the two. Note that different sizes for the source and target datasets and different sets of shared and private features are sampled for each of the different 10 random seeds used for all experimental results.
While RadialGAN was developed to work with any number of source datasets and uses multiple GANs, we use the full source datasets for training the benchmarks, making use of the source dataset. Each validation dataset is used for early stopping.

Summary statistics for the datasets We provide in Table 3 summary statistics of the different characteristics of the full source and target datasets used across our 10 experimental runs. Note that for the experiments where we vary the size of the target dataset, \(N_T\) is fixed as described above. To highlight the diversity of the datasets used for evaluating the benchmarks, we highlight in Table 3 the mean and standard deviations for the size of the source dataset \(N_R\), size of the target dataset \(N_T\), number of features shared between the source and target datasets \(D_s\), number of features private to the source dataset \(D_{pR}\), number of features private to the target dataset \(D_{pT}\), proportion of shared features in the source dataset \(D_s/(D_s + D_{pR})\), proportion of shared features in the target dataset \(D_s/(D_s + D_{pT})\) and the Maximum Mean Discrepancy (MMD) (computed using RBF kernel) between the shared features of the source and target datasets MMD\((X^{sr}, X^{st})\). Note that the MMD between the shared features of the source and target datasets for MAGGIC is much higher than for TCGA and Twins, as the different datasets in MAGGIC represent medical studies with different patient populations.

<table>
<thead>
<tr>
<th>(N_R)</th>
<th>(N_T)</th>
<th>(D_s)</th>
<th>(D_{pR})</th>
<th>(D_{pT})</th>
<th>(D_s/(D_s + D_{pR}))</th>
<th>(D_s/(D_s + D_{pT}))</th>
<th>MMD((X^{sr}, X^{st}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCGA</td>
<td>Twins</td>
<td>MAGGIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4219 ± 2569</td>
<td>5106 ± 2988</td>
<td>2533 ± 3711</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>360 ± 120</td>
<td>360 ± 120</td>
<td>267 ± 75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 ± 8</td>
<td>9 ± 2</td>
<td>27 ± 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 ± 7</td>
<td>9 ± 2</td>
<td>25 ± 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 ± 9</td>
<td>8 ± 3</td>
<td>21 ± 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.57 ± 0.17</td>
<td>0.49 ± 0.10</td>
<td>0.54 ± 0.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.50 ± 0.14</td>
<td>0.52 ± 0.08</td>
<td>0.60 ± 0.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0009 ± 0.0005</td>
<td>0.0011 ± 0.0008</td>
<td>0.056 ± 0.057</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.2 Implementation details and hyperparameter setting

As mentioned in Section 6, we evaluate the following benchmarks (1) training the CATE-learners only on the target dataset, (2) using only the shared features between the source and target datasets and (3) using RadialGAN to ‘translate’ the source dataset into the target domain (4) training the HTCE-learners on the full source and target datasets. We describe here the implementation details for RadialGAN, for the different CATE-learners used (S-Learner, T-Learner, DR-Learner and TARNet) and the corresponding HTCE-learners (HTCE-S-Learner, HTCE-T-Learner, HTCE-DR-Learner and HTCE-TARNet).

RadialGAN. We use the publicly available implementation of RadialGAN, which we augment to the treatment effects setting by considering the treatment separate from the rest of the features. While RadialGAN was developed to work with any number of source datasets and uses multiple GAN architectures to learn to translate from any one of the source datasets to the target domain, we consider here a simplified case with one source dataset. We use the same hyperparameter setting described in and found in the publicly available implementation for the generators, discriminators, encoders and decoders architectures and optimization (optimizer, batch size, learning rate).

CATE-learners. We use the publicly available implementation of the different CATE learners from. This allows us to have a consistent implementation for the S-Learner, T-Learner, DR-Learner and TARNet. Moreover, we use components similar to those in for all networks which ensure that \(\mu(x, w), \hat{\mu}(x)\) and \(\hat{\tau}(x)\) have access to similar number of layers and units in total, thus resulting in equally complex nuisance functions. In particular, for the S-learner, \(\mu(x, w)\) consists of 5 layers of 200 units each, for the T-learner, each of \(\mu_0(x), \mu_1(x)\) consists of 5 layers with 200 units each. For the DR-Learner, we set each of \(\mu_0(x), \mu_1(x), \hat{\mu}(x), \hat{\tau}(x)\) to have 5 layers with 200 units each. Finally, TARNet consists of 3 representation layers of 200 units each, followed by two outcome heads, each with 2 layers of 100 units each. We use ReLU activation function for the hidden layers.

\footnotesize
\textsuperscript{*}https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/RadialGAN
\textsuperscript{22}https://github.com/AliciaCurth/CATENets

\[\text{Table 3: Summary statistics of the datasets sampled for training the benchmarks.}\]
and linear/sigmoid activation for the last layer (for continuous/binary outcomes respectively). The different CATE learners are trained using the Adam optimizer [63], with learning rate set to 0.0001 and batch size 128. We perform early stopping on the validation dataset from the target domain.

**HTCE-learners.** Each $\phi^{PR}, \phi^{PT}, \phi^s$ for handling the heterogeneous feature spaces consists of one hidden layer with 100 units each. For the HTCE-S-Learner, we consider a building block of 5 layers with 100 hidden units for each shared and private subspace to share information between $\hat{\mu}^R(x^R; w), \hat{\mu}^T(x^T; w)$. Note that this essentially gives us 200 hidden units for the hypothesis function in each domain, thus ensuring that the nuisance functions for the CATE vs. HTCE-learners in each domain have similar complexity. For the HTCE-T-learner, we use separate feature encoders for each treatment $w \in \{0, 1\}$ ($\phi^{PR}_w, \phi^{PT}_w, \phi^s_w$ each consisting of one hidden layer with 100 units), and we use two building blocks of 5 layers with 100 units for each shared and private subspace to share information between $(\hat{\mu}^R_0(x^R), \hat{\mu}^T_0(x^T))$ and $(\hat{\mu}^R_1(x^R), \hat{\mu}^T_1(x^T))$. Similarly, for the HTCE-DRLearner, we use the same architecture as for the HTCE-T-learner to obtain the plug-in estimates of the potential outcomes, which consists of separate feature encoders for each treatment $w$, and two building blocks of 5 layers with 100 units for each shared and private subspace to share information between $(\hat{\pi}^R(x^R), \hat{\pi}^T(x^T))$. Moreover, for the pseudo-outcome regressor we also use separate feature encoders ($\phi^{PR}_y, \phi^{PT}_y, \phi^s_y$ each consisting of one hidden layer with 100 units) and a building blocks of 5 layers with 100 units for each shared and private subspace to share information between $(\hat{\tau}^R(x^R), \hat{\tau}^T(x^T))$.

For HTCE-TARNet, we use feature encoders shared between the two treatments ($\phi^{PR}, \phi^{PT}, \phi^s$ each consisting of one hidden layer with 100 units), followed by 3 representation layers (for each of $\Omega^R, \Omega^T$) and two building blocks of 2 layers with 100 units for each shared and private subspace to share information between $(\hat{\mu}^R_0(x^R), \hat{\mu}^T_0(x^T))$ and $(\hat{\mu}^R_1(x^R), \hat{\mu}^T_1(x^T))$. We use ReLU activation function for the hidden representation layers, SeLU activation for the shared and private subspace layers and linear/sigmoid activation for the last layer (for continuous/binary outcomes respectively). Note that for the building block of sharing information between outcome functions, we build upon the FlexTENet implementation [14]. The different HTCE-learners are trained using the Adam optimizer [63], with learning rate set to 0.0001 and batch size 128. Moreover, we also use the validation dataset from the target domain for early stopping.

All of the experiments were run on a virtual machine with 6CPUs, an Nvidia K80 Tesla GPU and 56GB of RAM.
E Additional experimental results

This section provides results for the different experimental settings considered in Section 6 for the remaining datasets and for all learners. Note that we observe similar trends for the different experimental settings across all datasets.

E.1 Benchmarks comparison and source of gain

Table 4 reports results on the TCGA and MAGGIC datasets for the comparison of the HTCE-learners against the benchmarks and the source of gain analysis. To reiterate, we compare the following benchmarks (1) training the CATE learners only on the target dataset, (2) using only the shared features between the source and target datasets to train the CATE learners (3) using RadialGAN to ‘translate’ the source dataset into the target domain and training the CATE learners on the target dataset augmented in this way and (4) training our HTCE-learners on the full source and target datasets. Moreover, we evaluate the impact of removing the shared and private layers that enable sharing information between the PO functions across domains (HTCE - PO sharing), removing the orthogonal regularization loss from the PO layers in Equation 5 (HTCE - $\mathcal{L}_{\text{orth}}$) and removing orthogonal regularization loss from the PO functions across domains (HTCE - $\mathcal{L}_{\text{orth}(\text{poly})}$). We note that on the TCGA and MAGGIC datasets as well, the HTCE-learners achieve better performance compared to the baselines and that each component we propose brings performance improvements.

Table 4: Benchmarks comparison and source of gain analysis in terms of PEHE on TCGA and MAGGIC datasets.

<table>
<thead>
<tr>
<th>Learner</th>
<th>S-Learner</th>
<th>T-Learner</th>
<th>DR-Learner</th>
<th>TARNet</th>
<th>S-Learner</th>
<th>T-Learner</th>
<th>DR-Learner</th>
<th>TARNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>0.15 ± 0.02</td>
<td>0.12 ± 0.01</td>
<td>0.10 ± 0.01</td>
<td>0.11 ± 0.01</td>
<td>0.39 ± 0.04</td>
<td>0.21 ± 0.01</td>
<td>0.19 ± 0.01</td>
<td>0.19 ± 0.01</td>
</tr>
<tr>
<td>Shared features</td>
<td>0.30 ± 0.07</td>
<td>0.30 ± 0.07</td>
<td>0.29 ± 0.07</td>
<td>0.30 ± 0.07</td>
<td>0.32 ± 0.03</td>
<td>0.28 ± 0.03</td>
<td>0.27 ± 0.03</td>
<td>0.27 ± 0.03</td>
</tr>
<tr>
<td>RadialGAN</td>
<td>0.12 ± 0.01</td>
<td>0.11 ± 0.01</td>
<td>0.09 ± 0.01</td>
<td>0.13 ± 0.03</td>
<td>0.31 ± 0.01</td>
<td>0.19 ± 0.01</td>
<td>0.19 ± 0.02</td>
<td>0.19 ± 0.01</td>
</tr>
<tr>
<td>HTCE - PO sharing</td>
<td>0.15 ± 0.01</td>
<td>0.10 ± 0.01</td>
<td>0.09 ± 0.01</td>
<td>0.10 ± 0.01</td>
<td>0.29 ± 0.02</td>
<td>0.15 ± 0.01</td>
<td>0.14 ± 0.01</td>
<td>0.17 ± 0.01</td>
</tr>
<tr>
<td>HTCE - $\mathcal{L}_{\text{orth}}$</td>
<td>0.10 ± 0.01</td>
<td>0.07 ± 0.01</td>
<td>0.06 ± 0.01</td>
<td>0.07 ± 0.01</td>
<td>0.25 ± 0.02</td>
<td>0.10 ± 0.01</td>
<td>0.09 ± 0.01</td>
<td>0.12 ± 0.01</td>
</tr>
<tr>
<td>HTCE - $\mathcal{L}_{\text{orth}(\text{poly})}$</td>
<td>0.11 ± 0.01</td>
<td>0.08 ± 0.01</td>
<td>0.07 ± 0.01</td>
<td>0.05 ± 0.01</td>
<td>0.26 ± 0.02</td>
<td>0.10 ± 0.01</td>
<td>0.10 ± 0.01</td>
<td>0.12 ± 0.01</td>
</tr>
<tr>
<td>HTCE (ours)</td>
<td>0.07 ± 0.01</td>
<td>0.06 ± 0.01</td>
<td>0.04 ± 0.01</td>
<td>0.06 ± 0.01</td>
<td>0.24 ± 0.02</td>
<td>0.08 ± 0.01</td>
<td>0.08 ± 0.01</td>
<td>0.10 ± 0.01</td>
</tr>
</tbody>
</table>

When looking at the performance of the benchmarks, using only the shared features between the two datasets generally has poor performance. This is due to the fact that this introduces hidden confounders in the CATE estimation since all of the patient features in both domains are used to obtain the potential outcomes and the treatment assignment in our data simulation (see Section 6). Moreover, note that RadialGAN only slightly improves performance over only training the CATE learners on the target dataset. We hypothesise that this may also be due to the small size of the target dataset which hinders the generator networks in RadialGAN to learn how to map examples from the source domain to the target one.

E.2 Varying the information sharing between domains

Figure 10 reports the results for varying the parameter $\alpha$, which controls the amount of information shared between the PO in the source and target domains, on the TCGA (top) and MAGGIC (bottom) datasets. We again notice that our HTCE-learners, which use a flexible approach for information sharing between PO functions across domains achieve good performance both when the PO functions are significantly different between the source and target dataset ($\alpha = 0.1$) and also when they have the same functional form and only depend on the shared features ($\alpha = 1.0$). In addition, for high values of $\alpha$ and especially for $\alpha = 1.0$ we again notice the degradation in performance due to the fact that the learners need to perform the implicit task of feature selection.

E.3 Varying the target dataset size.

Figure 11 reports the results on when varying the size of the target dataset $N_T$ on the TCGA dataset. Note that for the MAGGIC dataset, the size of the target domain is not simulated and it represents the size of the various studies selected as target domains. Thus, we only report here results on TCGA where we directly simulate the size of the target dataset. We notice that the benefits of doing transfer learning degrade as we increase the size of the target dataset.
Figure 9: Performance comparison on TCGA (top) and MAGGIC (bottom) when varying the information sharing between PO functions across domains through $\alpha$.

Figure 10: Performance comparison on TCGA when varying the size of the target dataset ($N_T$).

E.4 Effect of selection bias

Finally, we report results when varying the selection bias in the source and target domains for all learners on all three datasets: Twins (Figure 11), TCGA (Figure 12) and MAGGIC (Figure 13). We consider three settings for the selection bias in the source dataset $\kappa_R = 0$ for random treatment assignment, $\kappa_R = 2$ for moderate and $\kappa_R = 10$ for strong selection bias. For each setting of $\kappa_R$, we vary the selection bias in the target dataset $\kappa_T$ from $\kappa_T = 0$ to $\kappa_T = 10$. Our HTCE-learners have consistent performance when increasing the selection bias in both datasets and when there are significant discrepancies in the treatment assignment mechanism between the source and target datasets. While the impact of increasing the selection bias differs among the learners (e.g. TARNet vs. DRLearner), note that this is due to the intrinsic characteristics of each learner on a single domain, as the DRLearner uses propensity weighting to adjust for the selection bias, while TARNet does not.
Figure 11: Performance comparison on Twins for all learners when varying the selection bias $\kappa_R$ and $\kappa_T$ in the source and target datasets.

Figure 12: Performance comparison on TCGA for all learners when varying the selection bias $\kappa_R$ and $\kappa_T$ in the source and target datasets.

Figure 13: Performance comparison on MAGGIC for all learners when varying the selection bias $\kappa_R$ and $\kappa_T$ in the source and target datasets.
F Additional discussion about extensions and limitations

In this section, we describe how our method could be extended to additional scenarios than the ones considered in the paper and further discuss limitations.

**Multiple source datasets.** While in this paper we mainly consider the standard and most common transfer learning setting of leveraging a source dataset to improve the estimation of outcomes on a target dataset of interest, our proposed approach could be easily extended to having multiple source domains, and in fact, would scale linearly to incorporating additional domains.

More precisely, consider access to $M$ source domains, $\{D^{R_m}\}_{m=1}^M$, where each source domain consists of $D^{R_m} = \{(X_i^{R_m}, W_i, Y_i)\}_{i=1}^{N_{R_m}}$ and a target domain as described in Section 3. The objective would still be the one from Section 3 of improving the estimation of CATE for the target domain, but in this case, we need to leverage data from all $M$ source domains. Our proposed building blocks could be extended as follows to handle the additional source domains. The building block for handling heterogeneous feature spaces between source and target domains could be extended by having encoders $\phi^{R_m}$ for the private features $X^{P_{n,m}}$ of each source domain (part of $X^{R_m}$). Moreover, we can build an approach that shares information between the potential outcomes of each source domain with the target domain by having $L$ layers shared between all source domains and the target domains and $L$ private layers for each source domain. For source domain $m$, the input to the layer $(l+1)$-th can be obtained using $h_{w,l+1}^{R_m} = [h_{w,l}^{s}]^T h_{w,l}^{R_m}$ where $h_{w,l}^{R_m}$ is the output of the $l$-th private layer for source domain $m$ and $h_{w,l}^{s}$ is the output of the $l$-th shared layer across all domains.

While from a model development perspective this extension can be easily done, one also needs to consider whether the multiple source domains satisfy the underlying implicit assumptions for which such an architecture would be appropriate. In particular, it would be important to consider whether the PO functions across all source domains share information with the PO functions in the target domain, as using source domains that are significantly different from the target domain could harm performance.

**Streaming datasets.** Another important setting to consider is the one of having streaming datasets. In this scenario, one option could be the case where we already have a source domain and a target domain and we have incoming data streaming from the target domain. One such example in healthcare would be the case where in a single hospital we start collecting more or different clinical covariates for the patients which we now want to use for CATE estimation. The source domain would be the patients with only the original set of features, while the target domain would be the patients with the new set of features. However, as we start collecting these additional features, the initial target dataset will be small but it can start increasing with time as we observe more patients. In this setting, we can train an HTCE-learner with the initial data available from the source and target domains. However, instead of retraining the full HTCE-learner as we obtain more patients from the target domain, one option would be to fine-tune the weights using the incoming examples. While this is outside the scope of our paper, we believe that it would be important to investigate appropriate ways for performing such fine-tuning.

Another option could be the case of having full source datasets streaming, while the target dataset remains fixed. This would happen in the setting where for instance we gradually get access to data from multiple locations and we want to use these datasets as source datasets. In this scenario, one possibility would be to use the approach described above for having multiple source datasets and retraining a new model that incorporates all of the available source datasets as we get access to them. Another possibility would be to, instead of retraining a full model as we go from $M$ to $M+1$ source domains, we can add the needed private encoder and layers for the $(M+1)$-th domain and train only the new parameters and fine-tune the shared ones with the data from the new domain. While this is again outside our scope, it can provide interesting avenues for future work.

**Unknown domains.** In this paper, we assume that the source and target domains are known. However, given that we handle the setting of transfer learning for heterogeneous feature spaces, if the domains are unknown, one way to split available data into different domains, in this case, would be by using the same features to denote a single domain. For instance, if different patients in the dataset have recorded different sets of features, then the patients can be grouped according to having the same information collected for them and these can denote the different domains. Then, one needs to decide which represents the target domain, while keeping in mind that doing transfer learning is most
beneficial when the target dataset is small (as highlighted by our experimental results in Section 6.2 and Figure 6 (bottom)).

**Personalized feature spaces.** In the case where different features are available for different patients and the source and target domains are unknown, then the patients can be grouped according to their feature spaces as described in the paragraph above. On the other hand, if the source and target datasets are pre-defined and the patients within each dataset have different features available for them, one possible option would be to consider the super-set of their features as the different feature spaces and consider the features that are not available for each patient as missing. However, as our HTCE-learners has not been designed for this particular setting of having missing data, one would also need to investigate if additional assumptions (in addition to ensuring that the no hidden confounders and overlap assumptions are still satisfied) are needed to be able to obtain valid estimates of causal effects.
References


