NORMALIZING FLOWS FOR INTERVENTIONAL DENSITY ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing machine learning methods for causal inference usually estimate quantities expressed via the mean of potential outcomes (e.g., average treatment effect). However, such quantities do not capture the full information about the distribution of potential outcomes. In this work, we estimate the density of potential outcomes after interventions from observational data. For this, we propose a novel, fully-parametric deep learning method called *Interventional Normalizing Flows*. Specifically, we combine two normalizing flows, namely (i) a teacher flow for estimating nuisance parameters and (ii) a student flow for a parametric estimation of the density of potential outcomes. We further develop a tractable optimization objective via a one-step bias correction for an efficient and doubly robust estimation of the student flow parameters. As a result our *Interventional Normalizing Flows* offer a properly normalized density estimator. Across various experiments, we demonstrate that our *Interventional Normalizing Flows* are expressive and highly effective, and scale well with both sample size and high-dimensional confounding. To the best of our knowledge, our *Interventional Normalizing Flows* are the first fully-parametric, deep learning method for density estimation of potential outcomes.

1 INTRODUCTION

Causal inference increasingly makes use of machine learning methods to estimate treatment effects from observational data (e.g., van der Laan et al., 2011; Künzel et al., 2019; Curth & van der Schaar, 2021; Kennedy, 2022). This is relevant for various fields including medicine (e.g., Bica et al., 2021), marketing (e.g., Yang et al., 2020), and policy-making (e.g., Hünermund et al., 2021). Here, causal inference from observational data promises great value, especially when experiments for determining treatment effects are costly or even unethical.

The vast majority of the machine learning methods for causal inference estimate *averaged* quantities expressed by the (conditional) mean of potential outcomes. Examples of such quantities are the average treatment effect (ATE) (e.g., Shi et al., 2019; Hatt & Feuerriegel, 2021), the individual treatment effect (ITE) (e.g., Shalit et al., 2017; Hassanpour & Greiner, 2019; Zhang et al., 2020), and treatment-response curves (e.g., Bica et al., 2020; Nie et al., 2021). Importantly, these estimates only describe averages *without* distributional properties.

However, making decisions based on averaged causal quantities can be misleading and, in some applications, even dangerous (Spiegelhalter, 2017; van der Bles et al., 2019). On the one hand, if potential outcomes have different variances or number of modes, relying on the average quantities provides incomplete information about potential outcomes, and may inadvertently lead to local – and not global – optima during decision-making. On the other hand, distributional knowledge is needed to account for uncertainty in potential outcomes, and thus informs how likely a certain outcome is. For example, in medicine, knowing the distribution of potential outcomes is highly important (Gische & Voelkle, 2021): it gives the probability that the potential outcome lies in a desired range, and thus defines the probability of treatment success or failure. Motivated by this, we aim to estimate the *density* of potential outcomes.

An example highlighting the need for estimating the density of potential outcomes is shown in Fig. 1. Here, we simulated outcomes according to a given structural causal model (SCM). We observe that the potential outcomes have the same mean (i.e., $\mathbb{E}(Y[0]) = \mathbb{E}(Y[1])$) and the same variance

(i.e.,$\mathbb{V}(Y[0]) = \mathbb{V}(Y[1])$). Hence, the ground-truth ATE equals zero. Nevertheless, the distributions of potential outcomes (i. e., $\mathbb{P}(Y[a])$) are clearly different. Hence, in medical practice, acting upon the ATE without knowledge of the distributions of potential outcomes could have severe, negative effects. To show this, let us consider a "do nothing" treatment ($a = 0$) and some medical treatment ($a = 1$). Further, let us consider an outcome to be successful if some risk score Y is below the threshold of five. Then, the probability of treatment success (i. e., $\mathbb{P}\{Y[1] < 5.0\} \approx 0.63$) is much larger than the probability of success after the "do nothing" treatment (i. e., $\mathbb{P}\{Y[0] < 5.0\} \approx 0.51$), highlighting the importance of treatment.

In this paper, we aim to estimate the *density* of potential outcomes after intervention a, i. e., $\mathbb{P}(Y[a] = y)$. From this point on, we refer to this task as **interventional density estimation** (IDE). Estimating the density of interventions has several crucial advantages: it allows to identify multi-modalities in the distribution of potential outcomes; it allows to estimate quantiles of the distribution; and it allows to compute the probability with which a potential outcome lies in a certain range. Importantly, traditional density estimation methods are **not** applicable for IDE due to the fundamental problem of causal inference: that is, the counterfactual outcomes are typically never observed, and, hence, the sample from ground-truth interventional distribution is also inaccessible.

In prior literature, Kennedy et al. (2021) introduced a theory for efficient semi-parametric IDE estimation, but without a flexible algorithmic instantiation in form of a method. Existing literature also offers some specific methods for IDE, which are either semi- or non-parametric.[1] Examples are kernel density estimation (Kim et al., 2018) and kernel mean embeddings of distributions (Muandet et al., 2021). However, both methods neither scale well with the sample size nor with the dimensionality of covariates. Furthermore, both methods have an additional, crucial limitation: estimated densities could be unnormalized or even return negative values (which, by definition, is not possible). Fully-parametric methods, on the other hand, have several practical advantages: they automatically provide properly normalized density estimators, they allow one to sample from the estimated density and typically scale well with large and high-dimensional datasets. However, to the best of our knowledge, there is no fully-parametric, deep learning method for IDE.

In this paper, we develop a novel, fully-parametric deep learning method: ***Interventional Normalizing Flows*** (INFs). Our INFs build upon normalizing flows (NFs) (Rezende & Mohamed, 2015), but which we carefully adapt for causal inference. This requires several non-trivial adaptations. Specifically, we combine two NFs: a (i) teacher flow for estimating nuisance parameters, and a (ii) student flow for a parametric estimation of the density of potential outcomes. Further, we develop a training procedure based on a two-step knowledge transfer where we further integrate a one-step bias correction to allow for an efficient and doubly robust estimation of the student flow parameters.

Overall, our **main contributions** are following:[2]

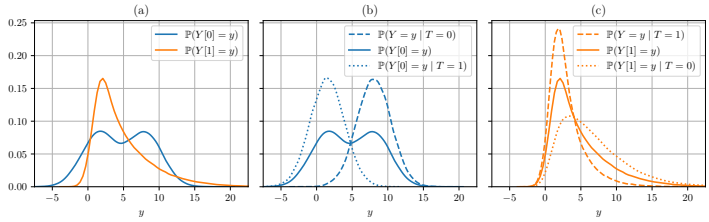
1. We introduce the first fully-parametric, deep learning method for interventional density estimation, called *Interventional Normalizing Flows* (INFs). Our INFs provide a properly normalized density estimator.
2. We propose a two-step training procedure with a one-step bias correction for efficient and doubly robust estimation.
3. We demonstrate in various experiments that our INFs are highly expressive and effective. We further show that they scale well to both large and high-dimensional datasets.

2 Related work

Recently, there has been a great interest in using machine learning and, specifically, deep learning for estimating causal quantities. Examples are machine learning for estimating ATEs (e.g., Shi et al., 2019; Hatt & Feuerriegel, 2021), ITEs (e.g., Johansson et al., 2016; Alaa & van der Schaar, 2018; Wager & Athey, 2018; Curth & van der Schaar, 2021), and treatment-response curves (e.g., Bica et al., 2020; Schwab et al., 2020; Nie et al., 2021). In this regard, some papers proposed uncertainty-aware

[1] We follow Pearl's hierarchy on causal inference (Bareinboim et al., 2022): we distinguish the interventional distribution (i.e., $\mathbb{P}(Y[a])$) and the counterfactual distribution (i.e., $\mathbb{P}(Y[a] \mid A = a')$). In general, the interventional and the counterfactual distribution are different. This can be seen by comparing plots (a) vs. (b) and (c) in Fig. 1. We acknowledge that other papers oftentimes call both counterfactual distributions for simplicity.

[2] Code is available at `https://anonymous.4open.science/r/AnonymousInterFlow-E2F3`.



$$X \sim \text{Mixture}\big(0.5N(0;1) + 0.5N(3;1)\big)$$
$$A \sim \text{Bern}\left(\frac{N(X;0,1)}{N(X;0,1) + N(X;b,1)}\right)$$
$$Y \sim N\big(A\,(X^2 - 1.82X + 2.0) + (1-A)\,(2.18\,X + 1.5);\,1\big)$$

Figure 1: Motivating example showing the densities of observational, interventional, and counterfactual distributions of outcome Y. These are simulated via the structural causal model on the right (here: $N(x;\mu,\sigma^2)$ are densities of the normal distribution; and $b = 3$ is a covariates shift, which regulates the probability of treatment assignment). Potential outcomes have different distributions but the same mean $\mathbb{E}(Y[0]) = \mathbb{E}(Y[1]) \approx 4.77$ and the same variance $\mathbb{V}(Y[0]) = \mathbb{V}(Y[1]) \approx 4.06$. (a) Interventional distributions. (b) and (c) Observational and counterfactual distributions for the same outcomes. As shown here, the observational, interventional, and counterfactual distributions can be vastly different.

methods, e. g., by using the variance of potential outcomes (Alaa & van der Schaar, 2017; Jesson et al., 2020). However, the aforementioned works are all concerned with estimating *averaged* causal quantities expressed via the mean of potential outcomes. In contrast, there are only a few papers that estimate the *density* of outcomes after intervention.

2.1 Interventional density estimation

Kennedy et al. (2021) introduced a theory for efficient semi-parametric estimation. The theory also lends to a hypothetical estimator as a solution to an integral equation, namely a bias-corrected moment condition. However, the theory comes without an algorithmic instantiation in form of a method. We later adopt the theoretical framework and convert the bias-corrected moment condition into a tractable optimization objective, which we can then solve very effectively with deep learning.

Table 1 lists existing methods for IDE. Importantly, these are either non-parametric or semi-parametric. Kim et al. (2018) developed a doubly robust kernel density estimation (KDE) via an efficient estimation of density functionals. Muandet et al. (2021) proposed kernel mean embeddings of distributions (DKME), which provides a non-parametric plug-in estimator. However, both methods (Kim et al., 2018; Muandet et al., 2021) have limitations. First, they do not provide a properly normalized density estimator. Hence, the estimated densities can be unnormalized or even negative, yet which, by definition, is not possible. Second, they do not offer direct sampling, which would allow one to sample from the estimated density without an additional algorithm. This may complicate computations of the test log-probability or empirical Wasserstein distance during evaluation. Third, another limitation of both non-parametric and semi-parametric methods is that they typically scale not well. This is unlike fully-parametric methods, which scale well to both large and high-dimensional datasets. However, so far, there is no full-parametric, deep learning method for IDE.

The methods for IDE above (Kim et al., 2018; Muandet et al., 2021; Kennedy et al., 2021) build upon general assumptions for causal identifiability. We later adopt the *same* assumptions for IDE (see Section 3), and we then develop a fully-parametric, deep learning method called INFs. Our method has three favorable properties: it yields a proper density estimator, it allows for direct sampling, and it scales well.

Table 1: Overview of methods for interventional density estimation from observational data.

Method	Density model	Parametric	Estimator type	Proper density	Direct sampling
Kim et al. (2018)	kernel density estimation (KDE)	semi-parametric	A-IPTW	✗	✗
Muandet et al. (2021)	distributional kernel mean embeddings (DKME)	non-parametric	plug-in	✗	✗
INFs (this paper)	normalizing flows (NFs)	fully-parametric	A-IPTW	✓	✓

A-IPTW: augmented inverse propensity of treatment weighted

2.2 Efficient estimation

In the context of treatment effect estimation, so-called augmented inverse propensity of treatment weighted (A-IPTW) estimators were developed for efficient, semi-parametric estimation of target estimands (parameters) (Robins, 2000). Formally, A-IPTW estimation performs a first-order bias correction of plug-in models (Bickel et al., 1993; Chernozhukov et al., 2018). A-IPTW estimation

also offers the property of being double robust, i. e., fast convergence rates even if one of the nuisance parameter estimators converges slowly Kennedy (2020). Although Kennedy et al. (2021) formulated an integral equation for semi- and fully-parametric efficient IDE estimation (see Eq. 19 therein), no flexible algorithmic instantiations in form of a method have been implemented so far. Later, we reformulate this equation as a tractable optimization problem, and, thereby, turn our INFs into an efficient and doubly robust IDE estimator.

2.3 Normalizing flows

Normalizing flows were introduced for expressive variational approximations in variational autoencoders (Rezende & Mohamed, 2015). One practical benefit of NFs is that they yield universal density approximators (Dinh et al., 2014; 2017; Huang et al., 2018; Durkan et al., 2019). Furthermore, NFs can be leveraged for conditional density estimation (e. g., via so-called hypernetworks (Trippe & Turner, 2018)). Normalizing flows were used for causal inference, but in a different setting from ours (see Appendix A).

Research gap: Existing methods for IDE are either non- or semi-parametric. To the best of our knowledge, our work is the first to propose a fully-parametric, deep learning method for IDE.

3 Setup: Interventional density estimation

Notation. Let $\mathbb{P}(Z)$ be a distribution of a random variable Z, and let $\mathbb{P}(Z = z)$ be its density or probability mass function. Let $\pi_a(x) = \mathbb{P}(A = a \mid X = x)$ denote the propensity score. Further, $\mathbb{1}(\cdot)$ is the indicator function; $\mathbb{P}_n\{f(X)\} = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$ is the sample average of a random $f(X)$; and $\mathbb{P}_b^{\mathcal{B}}\{f(X)\}$ is the average evaluated on a minibatch $\mathcal{B}$ of size b. For readability, we sometimes highlight random variables and the corresponding averaging operator in green color. Furthermore, $\mathbb{P}(Y \mid X, A)$ is the conditional distribution of the outcome Y.

Problem statement. In this work, we aim at estimating the interventional density from observational data, namely $\hat{\mathbb{P}}(Y[a] = y)$. To compare the goodness-of-fit of different estimators, we evaluate the distributional distance between the ground-truth interventional density and the estimated density. Such distributional distances include, e.g., the log-probability (which is equivalent to Kullback–Leibler (KL) divergence) and the empirical Wasserstein distance.

We build upon the standard setting of potential outcomes framework (Rubin, 1974), where $Y[a]$ stands for the potential outcome after intervening on treatment by setting it to a. That is, we consider an observational sample $\mathcal{D}$ with d_X-dimensional covariates $X \in \mathcal{X} \subseteq \mathbb{R}^{d_X}$, a treatment $A \in \{0, 1\}$, and a continuous outcome $Y \in \mathcal{Y} \subseteq \mathbb{R}$, drawn i.i.d. We assume the treatment to be binary, but note that our INFs also work with categorical treatments. We denote $\mathcal{D} = \{X_i, A_i, Y_i\}_{i=1}^{n} \sim \mathbb{P}(X, A, Y)$, where n is the sample size, and i is the index of an observation. For example, in critical care, the patient covariates X are different risk factors (e.g., age, gender, weight, prior diseases), the treatment is whether a ventilator is applied, and the outcome is the probability of patient survival. The covariates X are also called confounders if $\mathbb{P}(Y[a]) \neq \mathbb{P}(Y \mid A = a)$.

Identifiability. To identify the interventional density, we make the following identifiability assumptions with respect to the data-generating mechanism of $\mathcal{D}$: (1) *Positivity:* For some $\epsilon > 0$, $\mathbb{P}\{1 - \epsilon \geq \pi_a(X) \geq \epsilon\} = 1$. (2) *Consistency:* If $A = a$ for some patient, then $Y = Y[a]$. (3) *Exchangeability:* $A \perp\!\!\!\perp Y[a] \mid X$ for all a. Note that these assumptions are standard in the literature (Kim et al., 2018; Kennedy et al., 2021; Muandet et al., 2021). Under assumptions (1)–(3), the density of interventional distribution $\mathbb{P}(Y[a])$ can be expressed in terms of observational distribution with back-door adjustment, i.e.,

$$\mathbb{P}(Y[a] = y) = \int_{x \in \mathcal{X}} \mathbb{P}(Y = y \mid X = x, A = a)\, \mathbb{P}(X = x)\, \mathrm{d}x = \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X)} \big(\mathbb{P}(Y = y \mid X, A = a)\big), \quad (1)$$

where $\mathbb{P}(Y = y \mid X, A)$ is the conditional density of the outcome.

Plug-in estimator. A straightforward approach for IDE (Robins & Rotnitzky, 2001) is the following: first, one estimates the conditional outcome distribution, $\hat{\mathbb{P}}(Y \mid X, A)$ (here, any method for conditional density estimation could be used). Then, one takes a sample average over covariates X:

$$\hat{\mathbb{P}}^{\text{PI}}(Y[a] = y) = \mathbb{P}_n\{\hat{\mathbb{P}}(Y = y \mid X, A = a)\}. \quad (2)$$

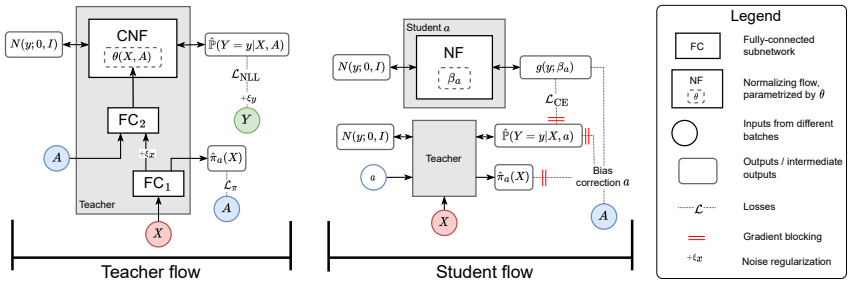


Figure 2: Overview of *Interventional Normalizing Flows*. Our INFs combine two normalizing flows, which we call "teacher flow" and "student flow". The teacher flow estimates the nuisance parameters, i.e., the propensity score $\hat{\pi}_a(X)$ and the conditional outcome distribution $\hat{\mathbb{P}}(Y \mid X, A)$. The student flow utilizes them to estimate the projection parameters $\hat{\beta}_a^{\text{A-IPTW}}$. Both teacher and student flows are trained via a two-step knowledge transfer procedure.

This estimator is an unbiased but inefficient estimator of interventional density, which is known as *semi-parametric plug-in estimator*. Semi-parametric IDE, unlike, e. g., semi-parametric ATE estimation, is highly problematic. For large sample sizes, the semi-parametric estimator requires averaging over the full sample for each evaluation point. Motivated by this, we aim to develop a fully-parametric estimator.

4 Theoretical Background for Fully-Parametric IDE

In this section, we introduce a theory for fully-parametric estimation of interventional density. First, we describe a parametric plug-in estimator as a solution to the moment condition (Kennedy et al., 2021). We call this estimator *covariate-adjusted estimator*. Second, we develop a one-step bias correction for efficient estimation.

We start by defining a parametric model, $\{g(y; \beta_a) \mid \beta_a \in \mathbb{R}^d\}$, where $\beta_a \in \mathbb{R}^d$ are parameters of estimator, and $g(\cdot; \beta_a)$ is a density, i. e., $\int_{y \in \mathcal{Y}} g(y; \beta_a) \, \mathrm{d}y = 1$. For IDE, we approximate the interventional distribution $\mathbb{P}(Y[a])$ with a distribution from our parametric model. That is, we aim at minimizing the distributional distance (specifically KL-divergence) between $\mathbb{P}(Y[a])$ and $g(\cdot; \beta_a)$ via

$$\hat{\beta}_a = \arg\min_{\beta_a} \mathrm{KL}\left(\mathbb{P}(Y[a]) \middle\| g(\cdot; \beta_a)\right) = \arg\min_{\beta_a} \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}(Y[a])} \Big(- \log g(Y^a; \beta_a) \Big), \tag{3}$$

where $\hat{\beta}_a$ are called *projection parameters* as they project the true interventional density onto a class $\{g(\cdot; \beta_a); \beta_a \in \mathbb{R}^d\}$.

4.1 Covariate-adjusted estimator

Let the d-dimensional random variable $T(Y; \beta_a) = -\nabla_{\beta_a} \log g(Y; \beta_a)$ denote the score function. Following Kennedy et al. (2021), the projection parameters can be equivalently expressed as a solution to the *moment condition* $m(\beta_a) \stackrel{!}{=} 0$, where

$$m(\beta_a) = \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}(Y[a])} T(Y^a; \beta_a) = \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X)} \Big(\mathbb{E}\big(T(Y; \beta_a) \mid X, A = a\big) \Big). \tag{4}$$

Here, the moment condition is the expected score function of the potential outcome. Throughout the paper, we assume that the moment condition has a unique solution, and, therefore, the minimization task in Eq. (3) and the root-finding task in Eq. (4) are equivalent.

In practice, we have neither observations from the interventional distribution nor counterfactual outcomes. Therefore, we cannot use the ground-truth $\mathbb{P}(Y[a])$ but, instead, must use the plug-in estimator distribution from Eq. (2). Specifically, we can obtain a plug-in estimator of projection parameters, i. e., $\hat{\beta}_a^{\text{PI}}$, either by minimizing a cross-entropy loss or by solving the moment condition, both of which are equivalent:

$$\hat{\beta}_a^{\text{PI}} = \arg\min_{\beta_a} \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}_n\{\hat{\mathbb{P}}(Y|X, A=a)\}} - \log g(Y^a; \beta_a) \iff \hat{m}^{\text{PI}}(\beta_a) = \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}_n\{\hat{\mathbb{P}}(Y|X, A=a)\}} T(Y^a; \beta_a) \stackrel{!}{=} 0. \tag{5}$$

Then, we can define a *parametric covariate-adjusted* (CA) estimator as $\hat{\mathbb{P}}^{\text{CA}}(Y[a] = y) = g(y; \hat{\beta}_a^{\text{PI}})$.[3] By choosing a sufficiently expressive class of densities for both g and the conditional density estimator $\hat{\mathbb{P}}(Y \mid X, A)$ (e. g., normalizing flows), CA can be shown to consistently estimate the interventional density (see Appendix B.5 in Kennedy et al. (2021)).

4.2 Efficient estimation via one-step bias correction

In the following, we aim to develop an efficient estimator of the projection parameter $\hat{\beta}_a$ from Eq. (3) or, equivalently, the moment condition $\hat{m}(\beta_a)$ at fixed β_a from Eq. (4). For this, we make use of semi-parametric efficiency theory (van der Laan & Robins, 2003; Kennedy et al., 2021), where the efficient influence function $\phi_a(T, \mathbb{P})$ for the functional $\mathbb{E}(\mathbb{E}(T \mid X, A = a))$ equals to

$$\phi_a(T; \mathbb{P}) = \frac{\mathbb{1}(A = a)}{\pi_a(X)} \Big(T - \mathbb{E}(T \mid X, A = a) \Big) + \mathbb{E}(T \mid X, A = a) - \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X)} (\mathbb{E}(T \mid X, A = a)). \quad (6)$$

Here, we use red color to show the nuisance parameters of $\mathbb{P}$ that are influencing the functional. We emphasize that the nuisance parameters (i. e., the propensity score and conditional expectations/probabilities) can be either known or estimated.

The efficient influence function in Eq. (6) allows us to construct an efficient estimator of the moment condition. Following (Kennedy et al., 2021), we transform the plug-in estimator $\hat{m}^{\text{PI}}(\beta_a)$ from Eq. (5) into an efficient estimator with the help of a *one-step bias correction*. In our case, the bias-corrected moment condition has the following form:

$$\hat{m}^{\text{A-IPTW}}(\beta_a) = \hat{m}^{\text{PI}}(\beta_a) + \mathbb{P}_n \big\{ \phi_a(T(Y; \beta_a); \hat{\mathbb{P}}) \big\} \stackrel{!}{=} 0, \quad (7)$$

where $\hat{\mathbb{P}} = \{\hat{\pi}_a(x), \hat{\mathbb{P}}(Y \mid X, A)\}$ are the estimated nuisance parameters of $\mathbb{P}$. The estimated nuisance parameters are simultaneously used for plug-in estimation of the moment condition. We call the solution of the bias-corrected moment equation $\hat{\beta}_a^{\text{A-IPTW}}$ an *augmented inverse propensity of treatment weighted* (A-IPTW) estimator of the projection parameters. Unlike CA estimator, A-IPTW estimator achieves efficiency and possesses a double robustness property.

We now transform the bias-corrected moment condition into the following tractable optimization task (see Appendix B for details):

$$\hat{\beta}_a^{\text{A-IPTW}} = \arg\min_{\beta_a} \Bigg[\underbrace{\mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}_n\{\hat{\mathbb{P}}(Y \mid X, A=a)\}} \Big(-\log g(Y^a; \beta_a) \Big)}_{\text{cross-entropy loss}} + \underbrace{\mathbb{P}_n \Bigg\{ \frac{\mathbb{1}(A = a)}{\hat{\pi}_a(X)} \Big(-\log g(Y; \beta_a) + \mathop{\mathbb{E}}_{Y \sim \hat{\mathbb{P}}(Y \mid X, A=a)} \big(\log g(Y; \beta_a) \big) \Big) \Bigg\}}_{\text{one-step bias correction}} \Bigg]. \quad (8)$$

Previously Kennedy et al. (2021) proposed to directly solve bias-corrected moment condition, yet which is in general not tractable. In contrast, we develop an optimization objective that can be directly incorporated into a loss of a deep learning density estimator.

5 Interventional Normalizing Flows

In the following, we describe our *Interventional Normalizing Flows*: a fully-parametric method for interventional density estimation via deep learning. First, we describe all the components of our architecture and, then, introduce an efficient estimation using one-step bias correction.

5.1 Components

In our INFs, we combine two normalizing flows, which we refer to as (i) *teacher flow* and (ii) *student flow (see Fig. 2)*. The rationale for this is based on our derivations in Section 4, according to which a

[3] The literature has developed a hierarchy of treatment effect methods (Curth & van der Schaar, 2021). This hierarchy distinguishes one-step learners and two-step learners. Examples of one-step learners are plug-in methods. Examples of two-step learners are regression adjustment (RA), propensity weighting (PW), and doubly robust (DR) or A-IPTW learners. Under this hierarchy, our covariate-adjusted estimator corresponds to a regression adjustment learner.

fully-parametric IDE requires two models: (i) one for the estimation of nuisance parameters, and (ii) one for the subsequent optimization of the learning objective with respect to projection parameters. Accordingly, both NFs in our INFs have thus different objectives: (i) the teacher flow estimates the nuisance parameters (i.e., the propensity score and the conditional outcome distribution); and (ii) the student flow uses the estimated nuisance parameters to estimate the projection parameters.

(i) Teacher flow. The teacher flow has three components: two fully-connected (FC) subnetworks and a conditional normalizing flow parameterized by θ. The first FC subnetwork (FC_1) takes the covariates X as input and, then, outputs a representation $R \in \mathbb{R}^r$ together with a propensity score $\hat{\pi}_a(X)$. The second FC subnetwork (FC_2) takes the representation R and the observed treatment A_i as input and, then, outputs the parameters of flow, conditioned on X and A, i. e., $\theta(X, A)$. Together, FC_1 and FC_2 form a so-called hypernetwork (Ha et al., 2017) for the conditional normalizing flow, which allows us to learn the conditional outcome distribution via back-propagation.

Let $\mathcal{L}_t$ be the loss of the teacher flow. Here, we combine a conditional negative log-likelihood ($\mathcal{L}_{\text{NLL}}$) and binary cross-entropy loss for the propensity score ($\mathcal{L}_\pi$), i.e., $\mathcal{L}_t(\hat{\mathbb{P}}, \hat{\pi}_a) = \mathbb{P}_n\{\mathcal{L}_{\text{NLL}} + \alpha\mathcal{L}_\pi\}$ with $\mathcal{L}_{\text{NLL}} = -\log \hat{\mathbb{P}}(Y = Y \mid X, A)$; $\mathcal{L}_\pi = \text{BCE}(\hat{\pi}_A(X), A)$, where $\alpha > 0$ is a hyper-parameter. In general, conditional normalizing flows are prone to overfitting when trained via a conditional negative log-likelihood. To address this, we later employ noise regularization (Rothfuss et al., 2019) in the conditional density estimation.

(ii) Student flow. The student flow uses the outputs of the teacher flow and then learns the interventional distribution. We first describe the naïve variant of the student flow without one-step bias correction (we introduce this later in Section 5.2). Different from the conditional normalizing flow in the teacher flow, the student flow is a non-conditional normalizing flow, parameterized by β_a. Specifically, we consider two separate normalizing flows, that is, one for each potential outcome (i.e., $a = 0$ and $a = 1$, respectively).[4]

To fit the student flow, we must solve the moment condition from Eq. (5) or, equivalently, minimize a cross-entropy loss ($\mathcal{L}_{\text{CE}}$). Here, we use a tractable approximation with numeric integration:[5]

$$\begin{aligned}\mathcal{L}_{\text{CE}}(\beta_a) &= \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}_n\{\hat{\mathbb{P}}(Y \mid X, A=a)\}} -\log g(Y^a; \beta_a) = -\int_{y \in \mathcal{Y}} \log g(y; \beta_a) \mathbb{P}_n\{\hat{\mathbb{P}}(Y = y \mid X, A = a)\}\, \mathrm{d}y \\ &\approx -h \sum_{j=1}^{K} \log g(y_j; \beta_a) \mathbb{P}_n\{\hat{\mathbb{P}}(Y = y_j \mid X, A = a)\}, \end{aligned} \tag{9}$$

where $y_{\min} \leq y_1 < \cdots < y_K \leq y_{\max}$ is an equidistant grid of points on $\mathcal{Y}$ with step size h.

Training. To train both components in our INFs, we make use of a two-step knowledge transfer. Specifically, we first fit the nuisance parameters with the teacher flow. Then, we freeze the parameters of the teacher flow and fit the student flow. We additionally employ exponential moving average (EMA) of the student parameters with a smoothing hyperparameter γ to stabilize the training for small minibatch sizes (Polyak & Juditsky, 1992). We show the full algorithm in Appendix C and further implementation details in Appendix D.

5.2 One-step bias correction

To provide an efficient estimation for the parameters of the student flow, we augment the cross-entropy loss (Eq. (9)) with a one-step bias correction. To evaluate the bias correction term, we need to compute an approximation of the conditional cross-entropy loss ($\mathcal{L}_{\text{CCE}}(X; \beta_a)$) as in Eq. (9). We thus compute

$$\mathcal{L}_{\text{CCE}}(X; \beta_a) = \mathop{\mathbb{E}}_{Y \sim \hat{\mathbb{P}}(Y \mid X, A=a)} -\log g(Y; \beta_a) \approx -h \sum_{j=1}^{K} \log g(y_j; \beta_a) \hat{\mathbb{P}}(Y = y_j \mid X, A = a),$$

where $y_{\min} \leq y_1 < \cdots < y_K \leq y_{\max}$ is an equidistant grid of points on $\mathcal{Y}$ with step size h. Finally, we obtain the loss of the student flow ($\mathcal{L}_s$), which is now suitable for our A-IPTW estimation from Eq. (9). We thus yield

$$\mathcal{L}_s(\beta_a) = \mathcal{L}_{\text{CE}}(\beta_a) + \mathbb{P}_n\left\{ \frac{\mathbb{1}(A = a)}{\hat{\pi}_a(X)} \Big(-\log g(Y; \beta_a) - \mathcal{L}_{\text{CCE}}(X; \beta_a) \Big) \right\}. \tag{10}$$

[4] One can use a single normalizing flow with a hypernetwork for categorical treatments.

[5] For high-dimensional outcomes, one can also use sampling-based methods, such as Monte Carlo integration.

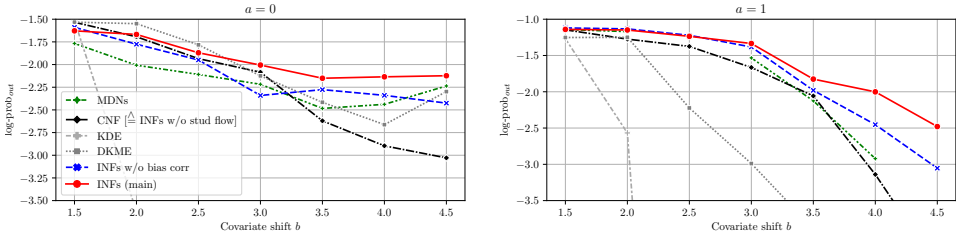


Figure 3: Results for synthetic data based on the SCM from Figure 1. Reported: mean over ten-fold train-test splits. Some runs for MDNs resulted with the log-prob$_{\text{out}} = -\infty$ and, thus, are not shown.

6 Experiments

6.1 Overview

To show the effectiveness of our INFs, we use established (semi-)synthetic datasets that have been previously used for treatment effect estimation (Shi et al., 2019; Curth & van der Schaar, 2021). The benefit of (semi-)synthetic datasets is that both factual and counterfactual outcomes are available (i.e., Y_i^{f} and Y_i^{cf}). Therefore, we can obtain a sample from the ground-truth interventional distribution, i. e., $Y[a]_i = \mathbb{1}(A_i = a)\, Y_i^{\text{f}} + \mathbb{1}(A_i \neq a)\, Y_i^{\text{cf}}$, which we can then use for IDE benchmarking.

Evaluation metric. We use the average log-probability as our standard metric for comparing density estimators. It is given by log-prob$_{\mathcal{D}} = \frac{1}{n} \sum_{i=1}^{n} \log \hat{\mathbb{P}}(Y[a] = Y[a]_i)$, where higher values indicate a better fit. The maximum value of the average log-probability is upper-bounded by the entropy, which, in general, is different for each potential outcome. Therefore, we separately report the results for each potential outcome. Of note, the log-probability is equivalent to the empirical KL-divergence.

Baselines. We use state-of-the-art IDE baselines (see Sec. 2.1): (1) mixture density networks (**MDNs**) (Bishop, 1994)[6]; (2) conditional normalizing flow (**CNF**) (Trippe & Turner, 2018); (3) kernel density estimation (**KDE**) (Kim et al., 2018); and (4) distributional kernel meaning embeddings (**DKME**) (Muandet et al., 2021). Both MDNs and CNF are semi-parametric plug-in estimators (see Eq. (2)). Importantly, both KDE and DKME do not guarantee a proper density estimation (unlike our INFs). We thus performed an additional re-normalization, so that we can use the average log-probability as an evaluation metric. Details on the baselines are in Appendix E, and hyperparameter tuning is reported in Appendix F.

Ablation studies. We compare three variants of our INFs: (1) **INFs** (main): Our INFs as introduced above using A-IPTW estimation. (2) **INFs w/o stud flow**: A simplified variant which uses only the conditional density estimation from the teacher flow as a semi-parametric plug-in estimator, and thus without student flow. This variant is identical to the CNF baseline. (3) **INFs w/o bias corr**: We use the covariate-adjusted fully-parametric estimator, where the student flow only uses the cross-entropy loss from Eq. (9) but without one-step bias correction. The ablations have the same hyperparameters as our main method for better comparability.

6.2 Results

Synthetic data. We generate synthetic data using the SCM ($d_X = 1$) from Fig. 1. Here, we vary the covariate shift b, which controls the overlap between the treated and non-treated population. Notably, low values of b correspond to the case, where both populations are similar or the same, while high values of b result in the violation of the positivity assumption. Further details on the synthetic dataset are provided in Appendix G. Fig. 3 shows the results. Our INFs achieve large performance improvements over the baselines, especially for larger b. Moreover, the ablation studies confirm that our proposed deep learning architecture with one-step bias correction is superior.

IHDP dataset. The Infant Health and Development Program (IHDP) (Hill, 2011) is a semi-synthetic dataset with two synthetic potential outcomes generated from real-world medical covariates ($n = 747, d_X = 25$). Here, we used ten-fold train/test splits (90%/10%) and perform hyperparameter tuning based on the first split. Our INFs outperform all baselines for both potential outcomes (see

[6]MDNs were previously used to estimate the conditional distribution of outcome for quantifying the ignorance regions of ITE estimation (Jesson et al., 2021; 2022). However, this is different from our IDE task.

Table 2). The ablation study again confirms that our main INFs are superior over the other variants without the student flow and without bias correction. In Appendix H, we repeat the evaluation using the empirical Wasserstein distance with similar findings.

Table 2: Results for IHDP dataset.Reported: mean $\pm$ standard deviation over ten-fold train-test splits.

	$a = 0$		$a = 1$	
	log-prob$_{\text{in}}$	log-prob$_{\text{out}}$	log-prob$_{\text{in}}$	log-prob$_{\text{out}}$
MDNs	-0.927 ± 0.024	-0.942 ± 0.080	-0.679 ± 0.048	-0.684 ± 0.077
CNF [$\hat{=}$ INFs w/o stud flow]	-0.943 ± 0.032	-0.970 ± 0.072	-0.679 ± 0.061	-0.674 ± 0.091
KDE (Kim et al., 2018)	-18.415 ± 0.650	-18.230 ± 1.717	-0.869 ± 0.043	-0.868 ± 0.044
DKME (Muandet et al., 2021)	-0.999 ± 0.009	-1.003 ± 0.056	-0.789 ± 0.008	-0.789 ± 0.028
INFs w/o bias corr	-0.932 ± 0.017	-0.943 ± 0.103	-0.667 ± 0.025	-0.682 ± 0.074
INFs (main)	$\mathbf{-0.916 \pm 0.015}$	$\mathbf{-0.931 \pm 0.107}$	$\mathbf{-0.656 \pm 0.029}$	$\mathbf{-0.670 \pm 0.091}$

Higher = better (best in bold)

ACIC 2016 & 2018 datasets. ACIC 2016 & 2018 provide a collection of semi-synthetic datasets with various data-generating mechanisms (Dorie et al., 2019; Shimoni et al., 2018) (see details in Appendix G). We select 15 random datasets from ACIC 2016 and 24 random datasets (4 of each of 6 sizes) from ACIC 2018. We perform five random train/test splits (80%/20%) for each dataset, tune hyperparameters on the first split and evaluate the average in- and out-sample log-probability on every split. Table 3 provides the performance comparison. Again, our INFs have a clear performance improvement over both baselines and other model variants. Compared to MDNs as the second-best method, our INFs scale much better in terms of runtime, especially for large sample sizes (see Appendix I).

Table 3: Results for ACIC 2016 and ACIC 2018. Reported: % of runs with the best performance.

	ACIC 2016 (15 datasets)		ACIC 2018 (24 datasets)	
	log-prob$_{\text{in}}$	log-prob$_{\text{out}}$	log-prob$_{\text{in}}$	log-prob$_{\text{out}}$
MDNs	24.67%	21.33%	23.75%	25.42%
CNF [$\hat{=}$ INFs w/o stud flow]	18.67%	27.33%	19.58%	16.67%
KDE (Kim et al., 2018)	2.67%	2.00%	0.00%	0.00%
DKME (Muandet et al., 2021)	20.00%	17.33%	11.67%	11.25%
INFs w/o bias corr	2.00%	2.67%	10.00%	11.67%
INFs (main)	**32.00%**	**29.33%**	**35.00%**	**35.00%**

Higher = better (best in bold)

HC-MNIST dataset. Hidden confounding MNIST dataset is a semi-synthetic dataset (Jesson et al., 2021), constructed on top of the canonical image dataset of handwritten digits (MNIST) (LeCun, 1998). To satisfy the exchangeability assumption, we add a hidden confounder to the set of all covariates, i. e., 28x28 images ($d_X = 784 + 1$). For dataset details, see Appendix G. For our experiments, we use only the train subset of the original MNIST ($n = 42,000$). Here we used ten random train/test splits (80%/20%) and tune hyperparameters on the first split. Table 4 shows the results of the experiments. Note that the non- and semi-parametric baselines suffer from scalability issues and were thus excluded. Further, our INFs outperform the variant without a bias correction, i. e., the only other available baseline.

Table 4: Results for HC-MNIST. Reported: mean $\pm$ std. dev. over ten random train-test splits.

	$a = 0$		$a = 1$	
	log-prob$_{\text{in}}$	log-prob$_{\text{out}}$	log-prob$_{\text{in}}$	log-prob$_{\text{out}}$
INFs w/o bias corr	-1.430 ± 0.181	-1.429 ± 0.181	-1.400 ± 0.171	-1.402 ± 0.170
INFs (main)	$\mathbf{-1.339 \pm 0.005}$	$\mathbf{-1.338 \pm 0.009}$	$\mathbf{-1.329 \pm 0.002}$	$\mathbf{-1.332 \pm 0.007}$

Higher = better (best in bold)

Scalability. Experiments with ACIC 2018 and HC-MNIST datasets showed high effectiveness of our INFs for datasets with large sample sizes ($n > 25,000$) and with high-dimensional covariates ($d_X > 100$). We provide a runtime comparison in Appendix I. For HC-MNIST, non- and semi-parametric methods even become completely impractical due to memory and time constraints.

7 Conclusion

For decision-making in personalized medicine, it is not only important to know what effect treatments have on patient health but also how likely it is that treatments achieve the desired outcome. To address this, we propose a novel method for estimating the density of potential outcomes. Specifically, we present *Interventional Normalizing Flows* which is the first, fully-parametric, deep learning method for this purpose.

REFERENCES

Ahmed M. Alaa and Mihaela van der Schaar. Bayesian inference of individualized treatment effects using multi-task Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

Ahmed M. Alaa and Mihaela van der Schaar. Bayesian nonparametric causal inference: Information rates and learning algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 12(5): 1031–1046, 2018.

Sourabh Balgi, Jose M. Pena, and Adel Daoud. Personalized public policy analysis in social sciences using causal-graphical normalizing flows. In *Association for the Advancement of Artificial Intelligence*, 2022.

Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. On Pearl's hierarchy and the foundations of causal inference. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pp. 507–556. Association for Computing Machinery, 2022.

Ioana Bica, James Jordon, and Mihaela van der Schaar. Estimating the effects of continuous-valued interventions using generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2020.

Ioana Bica, Ahmed M. Alaa, Craig Lambert, and Mihaela van der Schaar. From real-world patient data to individualized treatment effects using machine learning: current and future methods to address underlying challenges. *Clinical Pharmacology & Therapeutics*, 109:87–100, 2021.

Peter J. Bickel, Chris A.J. Klaassen, Ya'acov Ritov, and Jon A. Wellner. *Efficient and adaptive estimation for semiparametric models*, volume 4. Springer New York, NY, 1993.

Christopher M. Bishop. Mixture density networks. 1994.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, 2020.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters: Double/debiased machine learning. *The Econometrics Journal*, 21(1):C1–C68, 2018.

Alicia Curth and Mihaela van der Schaar. Nonparametric estimation of heterogeneous treatment effects: From theory to learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Vincent Dorie, Jennifer Hill, Uri Shalit, Marc Scott, and Dan Cervone. Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition. *Statistical Science*, 34(1):43–68, 2019.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, 2019.

Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

Christian Gische and Manuel C. Voelkle. Beyond the mean: A flexible framework for studying causal effects using linear models. *Psychometrika*, 87:868–901, 2021.

Steffen Grünewälder, Guy Lever, Arthur Gretton, Luca Baldassarre, Sam Patterson, and Massimiliano Pontil. Conditional mean embeddings as regressors. In *International Conference on Machine Learning*, 2012.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations*, 2017.

Negar Hassanpour and Russell Greiner. Learning disentangled representations for counterfactual regression. In *International Conference on Learning Representations*, 2019.

Tobias Hatt and Stefan Feuerriegel. Estimating average treatment effects via orthogonal regularization. In *CIKM*, 2021.

Jennifer L. Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, 2018.

Paul Hünermund, Jermain Christopher Kaminski, and Carla Schmitt. Causal machine learning and business decision making. In *Academy of Management Proceedings*, 2021.

Maximilian Ilse, Patrick Forré, Max Welling, and Joris M. Mooij. Combining interventional and observational data using causal reductions. *arXiv preprint arXiv:2103.04786*, 2021.

Andrew Jesson, Sören Mindermann, Uri Shalit, and Yarin Gal. Identifying causal-effect inference failure with uncertainty-aware models. In *Advances in Neural Information Processing Systems*, 2020.

Andrew Jesson, Sören Mindermann, Yarin Gal, and Uri Shalit. Quantifying ignorance in individual-level causal-effect estimates under hidden confounding. In *International Conference on Machine Learning*, 2021.

Andrew Jesson, Alyson Douglas, Peter Manshausen, Nicolai Meinshausen, Philip Stier, Yarin Gal, and Uri Shalit. Scalable sensitivity and uncertainty analysis for causal-effect estimates of continuous-valued interventions. *arXiv preprint arXiv:2204.10022*, 2022.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International Conference on Machine Learning*, 2016.

Edward H. Kennedy. Optimal doubly robust estimation of heterogeneous causal effects. *arXiv preprint arXiv:2004.14497*, 2020.

Edward H. Kennedy. Semiparametric doubly robust targeted double machine learning: A review. *arXiv preprint arXiv:2203.06469*, 2022.

Edward H. Kennedy, Sivaraman Balakrishnan, and Larry Wasserman. Semiparametric counterfactual density estimation. *arXiv preprint arXiv:2102.12034*, 2021.

Ilyes Khemakhem, Ricardo Monti, Robert Leech, and Aapo Hyvarinen. Causal autoregressive flows. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Kwangho Kim, Jisu Kim, and Edward H. Kennedy. Causal effects based on distributional distances. *arXiv preprint arXiv:1806.02935*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Sören R Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019.

Yann LeCun. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

M.F. MacDorman and J.O. Atkinson. Infant mortality statistics from the linked birth/infant death data set–1995 period data. *Monthly Vital Statistics Report*, 46(6 Suppl 2):1–22, 1998.

Krikamol Muandet, Motonobu Kanagawa, Sorawit Saengkyongam, and Sanparith Marukatat. Counterfactual mean embeddings. *Journal of Machine Learning Research*, 22:1–71, 2021.

Jens Müller, Robert Schmier, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Learning robust models using the principle of independent causal mechanisms. In *DAGM German Conference on Pattern Recognition*, 2021.

Lizhen Nie, Mao Ye, Qiang Liu, and Dan Nicolae. VCNet and functional targeted regularization for learning causal effects of continuous treatments. In *International Conference on Learning Representations*, 2021.

Kenneth R. Niswander. The collaborative perinatal study of the National Institute of Neurological Diseases and Stroke. *The Woman and Their Pregnancies*, 1972.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

James M. Robins. Robust estimation in sequentially ignorable missing data and causal inference models. In *Proceedings of the American Statistical Association*, 2000.

James M. Robins and Andrea Rotnitzky. Comment on the Bickel and Kwon article, "Inference for semiparametric models: Some questions and an answer". *Statistica Sinica*, 11(4):920–936, 2001.

Jonas Rothfuss, Fabio Ferreira, Simon Boehm, Simon Walther, Maxim Ulrich, Tamim Asfour, and Andreas Krause. Noise regularization for conditional density estimation. *arXiv preprint arXiv:1907.08982*, 2019.

Donald B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688, 1974.

Patrick Schwab, Lorenz Linhardt, Stefan Bauer, Joachim M. Buhmann, and Walter Karlen. Learning counterfactual representations for estimating individual dose-response curves. In *AAAI Conference on Artificial Intelligence*, 2020.

Uri Shalit, Fredrik D. Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, 2017.

Claudia Shi, David Blei, and Victor Veitch. Adapting neural networks for the estimation of treatment effects. In *Advances in Neural Information Processing Systems*, 2019.

Yishai Shimoni, Chen Yanover, Ehud Karavani, and Yaara Goldschmnidt. Benchmarking framework for performance-evaluation of causal inference analysis. *arXiv preprint arXiv:1802.05046*, 2018.

David Spiegelhalter. Risk and uncertainty communication. *Annual Review of Statistics and Its Application*, 4(1):31–60, 2017.

D. Michael Titterington, Smith Afm, Adrian F.M. Smith, U.E. Makov, et al. *Statistical analysis of finite mixture distributions*, volume 198. John Wiley & Sons Incorporated, 1985.

Brian L. Trippe and Richard E. Turner. Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.

Anne Marthe van der Bles, Sander van der Linden, Alexandra L.J. Freeman, James Mitchell, Ana B. Galvao, Lisa Zaval, and David J. Spiegelhalter. Communicating uncertainty about facts, numbers and science. *Royal Society Open Science*, 6(5):181870, 2019.

Mark J. van der Laan and James M. Robins. *Unified methods for censored longitudinal data and causality*, volume 5. Springer New York, NY, 2003.

Mark J. van der Laan, Sherri Rose, et al. *Targeted learning: Causal inference for observational and experimental data*, volume 10. Springer, 2011.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Rongguang Wang, Pratik Chaudhari, and Christos Davatzikos. Harmonization with flow-based causal inference. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2021.

Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Jeremy Yang, Dean Eckles, Paramveer Dhillon, and Sinan Aral. Targeting for long-term outcomes. *arXiv preprint arXiv:2010.15835*, 2020.

Yao Zhang, Alexis Bellot, and Mihaela van der Schaar. Learning overlapping representations for the estimation of individualized treatment effects. In *International Conference on Artificial Intelligence and Statistics*, 2020.

A RELATED WORK: NORMALIZING FLOWS FOR CAUSAL INFERENCE

NFs have been used in the wider area of causal inference, yet in vastly *different* tasks than ours. Examples include, e. g., robust prediction by employing causal mechanisms (Müller et al., 2021); combining interventional and observational datasets (Ilse et al., 2021); and causal discovery (Brouillard et al., 2020). Further, several works aim to model Bayesian networks or structural causal models (SCMs) with known or unknown causal diagrams. For example, NFs were used as a probabilistic model for Bayesian networks aimed at causal discovery, as well as downstream interventional and counterfactual inference (Khemakhem et al., 2021; Wang et al., 2021; Wehenkel & Louppe, 2021). Balgi et al. (2022) build upon a temporal SCM with exogenous noise, where NFs are used for interventional and counterfactual queries. Importantly, all the aforementioned methods assume continuous variables in SCMs and *independence of exogenous noise*.[7] Hence, these methods are **not** applicable in our case, which considers semi-Markovian SCMs and which is thus a different inference task.[8] In sum, NFs have not yet been adapted to IDE, which is our novelty.

[7]This is commonly known as a causal Markov condition.

[8]This is stated in our identifiability assumptions: there is no limitation on the exogenous noise independence between outcome and covariates. Hence, our setting is more general.

B BIAS-CORRECTED MOMENT CONDITION AS AN OPTIMIZATION TASK

We aim to transform the bias-corrected moment condition into an optimization objective:

$$\hat{m}^{\text{A-IPTW}}(\beta_a) = \hat{m}^{\text{PI}}(\beta_a) + \mathbb{P}_n\big\{\phi_a(T(Y;\beta_a);\hat{\mathbb{P}})\big\} \stackrel{!}{=} 0. \tag{11}$$

We first note that the plug-in estimator of moment condition $\hat{m}^{\text{PI}}(\beta_a)$ can be rewritten as

$$\hat{m}^{\text{PI}}(\beta_a) = \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}_n\{\hat{\mathbb{P}}(Y|X,A=a)\}} T(Y^a;\beta_a) = \int_{\mathcal{Y}} T(y;\beta_a)\, \mathbb{P}_n\{\hat{\mathbb{P}}(Y=y \mid X, A=a)\}\, \mathrm{d}y \tag{12}$$

$$= \mathbb{P}_n\Big\{\int_{\mathcal{Y}} T(y;\beta_a)\, \hat{\mathbb{P}}(Y=y \mid X, A=a)\, \mathrm{d}y\Big\} = \mathbb{P}_n\Big\{\hat{\mathbb{E}}\big(T(Y;\beta_a) \mid X, A=a\big)\Big\}, \tag{13}$$

where the last equality follows from the definition of the conditional expectation. Notably, we see that the moment condition could be equivalently solved with either the conditional distribution, $\mathbb{P}(Y \mid X, A=a)$, or with the functional regression, $\mathbb{E}\big(T(Y;\beta_a) \mid X, A=a\big)$.

Let us unroll the bias correction term of Eq. (7):

$$\mathbb{P}_n\{\phi_a(T;\hat{\mathbb{P}})\} = \mathbb{P}_n\Big\{\frac{\mathbb{1}(A=a)}{\hat{\pi}_a(X)}\Big(T - \hat{\mathbb{E}}(T \mid X, A=a)\Big) + \hat{\mathbb{E}}(T \mid X, A=a) - \mathbb{P}_n\big\{\hat{\mathbb{E}}(T \mid X, A=a))\big\}\Big\} \tag{14}$$

$$= \mathbb{P}_n\Big\{\frac{\mathbb{1}(A=a)}{\hat{\pi}_a(X)}\Big(T - \hat{\mathbb{E}}(T \mid X, A=a)\Big) + \hat{\mathbb{E}}(T \mid X, A=a)\Big\} - \mathbb{P}_n\big\{\hat{\mathbb{E}}(T \mid X, A=a))\big\}, \tag{15}$$

where nuisance parameters are marked with red color. Here, the last term is in fact the plug-in estimator of the moment condition, i. e., $-\hat{m}^{\text{PI}}(\beta_a)$. Therefore, we can simplify one-step bias corrected moment condition via

$$\hat{m}^{\text{A-IPTW}}(\beta_a) = \mathbb{P}_n\Big\{\frac{\mathbb{1}(A=a)}{\hat{\pi}_a(X)}\Big(T - \hat{\mathbb{E}}(T \mid X, A=a)\Big) + \hat{\mathbb{E}}(T \mid X, A=a)\Big\} \tag{16}$$

$$= \mathop{\mathbb{E}}_{Y^a \sim \mathbb{P}_n\{\hat{\mathbb{P}}(Y|X,A=a)\}} T(Y^a;\beta_a) + \mathbb{P}_n\Big\{\frac{\mathbb{1}(A=a)}{\hat{\pi}_a(X)}\Big(T(Y;\beta_a) - \mathop{\mathbb{E}}_{Y \sim \hat{\mathbb{P}}(Y|X,A=a)} T(Y;\beta_a)\Big)\Big\}, \tag{17}$$

where we use the conditional density estimator but not an estimator for the functional regression. This allows us to transform the A-IPTW moment condition into an optimization objective (Eq. (8)) by taking antiderivative with respect to β_a.

C TWO-STEP LEARNING ALGORITHM

Our INFs are trained with a two-step knowledge transfer procedure. The procedure is shown in Algorithm 1. Recall that we use noise regularization as the main regularization technique for the teacher flow, and exponential moving average (EMA) for the student flow to stabilize training. A-IPTW estimation is also known to become unstable in a finite sample setting (Shi et al., 2019), so that inverse values of propensity score become too large. Thus, we manually discard observations with too small propensity score ($\hat{\pi}_a(X) < 0.05$) from bias correction.

Algorithm 1 Knowledge transfer between the teacher and student flows

Input: number of iterations $n_{\text{iter},t}, n_{\text{iter},s}$; minibatch sizes b_t, b_s; learning rates η_t, η_s; intensities of the noise regularization σ_x^2, σ_y^2; EMA smoothing γ; grid size K.

Init: parameters of the teacher flow: $\text{FC}_1^{(0)}, \text{FC}_2^{(0)}$ ▷ Fitting the teacher flow
for i = 0 **to** $n_{\text{iter, t}}$ **do**
  $\mathcal{B} = \{X, A, Y\} \leftarrow$ minibatch of size b_t
  $R, \hat{\pi}_a(X) \leftarrow \text{FC}_1^{(i)}(X)$
  $\xi_x \sim N(0, \sigma_x^2); \xi_y \sim N(0, \sigma_y^2); \tilde{R} \leftarrow R + \xi_x; \tilde{Y} \leftarrow Y + \xi_y$ ▷ Noise regularization
  $\theta(X, A) \leftarrow \text{FC}_2^{(i)}(A, \tilde{R})$
  $\hat{\mathbb{P}}(Y \mid X, A) \leftarrow$ normalizing flow with parameters $\theta(X, A)$
  $\mathcal{L}_{\text{NLL}} \leftarrow -\log \hat{\mathbb{P}}(Y = \tilde{Y} \mid X, A)$
  $\mathcal{L}_\pi \leftarrow \text{BCE}(\hat{\pi}_A(X), A)$
  $\mathcal{L}_t(\hat{\mathbb{P}}, \hat{\pi}_a) \leftarrow \mathbb{P}_{b_t}^{\mathcal{B}}\{\mathcal{L}_{\text{NLL}} + \alpha \mathcal{L}_\pi\}$
  $\text{FC}_1^{(i+1)}, \text{FC}_2^{(i+1)} \leftarrow$ optimization step wrt. $\mathcal{L}_t(\hat{\mathbb{P}}, \hat{\pi}_a)$ with learning rate η_t
end for
Output: nuisance parameters: $\hat{\mathbb{P}}(Y \mid X, A), \hat{\pi}_a(X)$

Init: parameters of the student flows: $\beta_a^{(0)}$, $\beta_{a,\text{EMA}}^{(0)} \leftarrow \beta_a^{(0)}$ ▷ Fitting the student flows
for i = 0 **to** $n_{\text{iter, s}}$ **do**
  $\mathcal{B} = \{X, A, Y\} \leftarrow$ minibatch of size b_s
  for $a \in \{0, 1\}$ **do**
    $\mathcal{L}_{\text{CE}}(\beta_a^{(i)}) \leftarrow -h \sum_{j=1}^{K} \log g(y_j; \beta_a^{(i)}) \mathbb{P}_{b_s}^{\mathcal{B}}\{\hat{\mathbb{P}}(Y = y_j \mid X, A = a)\}$
    $\mathcal{L}_{\text{CCE}}(X; \beta_a^{(i)}) \leftarrow -h \sum_{j=1}^{K} \log g(y_j; \beta_a^{(i)}) \hat{\mathbb{P}}(Y = y_j \mid X, A = a)$
    bias correction$(\beta_a^{(i)}) \leftarrow \mathbb{P}_{b_s}^{\mathcal{B}} \left\{ \frac{\mathbb{1}(A=a \& \hat{\pi}_a(X) \geq 0.05)}{\hat{\pi}_a(X)} \Big(-\log g(Y; \beta_a^{(i)}) - \mathcal{L}_{\text{CCE}}(X; \beta_a^{(i)}) \Big) \right\}$
    $\mathcal{L}_s(\beta_a^{(i)}) \leftarrow \mathcal{L}_{\text{CE}}(\beta_a^{(i)})$+bias correction$(\beta_a^{(i)})$
    $\beta_a^{(i+1)} \leftarrow$ optimization step wrt. $\mathcal{L}_s(\beta_a^{(i)})$ with learning rate η_s
    $\beta_{a,\text{EMA}}^{(i+1)} \leftarrow \gamma \beta_{a,\text{EMA}}^{(i)} + (1 - \gamma) \beta_a^{(i+1)}$ ▷ EMA update
  end for
end for
Output: $\hat{\beta}_a^{\text{A-IPTW}} \leftarrow \beta_{a,\text{EMA}}^{(n_{\text{iter, s}})}$

D INFS IMPLEMENTATION DETAILS

Implementation. We implemented our INFs using PyTorch and Pyro. For both teacher and student flow, we employ neural spline flows (Durkan et al., 2019) with standard normal ($N(0; 1)$) as a base distribution. Neural spline flows construct an invertible transformation of the base distribution with the help of monotonic rational-quadratic splines. They are characterized by two main hyperparameters: a number of knots n_{knots} and a span of the transformation interval $[-B; B]$. n_{knots} controls the smoothness of estimated density and B affects the support of the transformation. In our experiments, we heuristically set $B = y_{\max} - y_{\min} + 5$.

For the teacher flow, we use fully-connected subnetworks each with one hidden layer (with $h = 10$ hidden units), and the dimensionality of representation is set to $r = 10$.

Training. During training (see full algorithm in Appendix C), we adopt noise regularization (Rothfuss et al., 2019) and add an independent Gaussian noise $\xi_x \sim N(0, \sigma_x^2), \xi_y \sim N(0, \sigma_y^2)$ to the representation and output of the teacher flow, i. e., $\tilde{R} = R + \xi_x; \tilde{Y} = Y + \xi_y$. For faster learning, we approximate a full-sample average $\mathbb{P}_n\{\cdot\}$ with a minibatch average $\mathbb{P}_b^{\mathcal{B}}\{\cdot\}$ for all the losses, where b is the minibatch size. We use stochastic gradient descent (SGD) for fitting the parameters of the teacher flow, and Adam optimizer (Kingma & Ba, 2015) for the student flow with learning rates η_t and η_s, respectively. We fix the weighting hyperparameters of the loss to $\alpha = 1$ and the EMA smoothing hyperparameter to $\gamma = 0.995$. The grid size for approximating the cross-entropy loss is set to $K = 100$. Both $y_{\min}$ and $y_{\max}$ are set to the empirical minimum and maximum of the train sub-sample.

Hyperparameter tuning. We perform extensive hyperparameter tuning only for the teacher flow. Hyperparameters for tuning include, e. g., number of knots of neural spline flows $n_{\text{knots},t}$, the minibatch size b_t, the learning rate η_t, and the intensities of the noise regularization σ_x^2, σ_y^2. On the other hand, we discovered, that the student flow works well with the same plain set of hyperparameters in almost all the experiments. Those include the minibatch size $b_s = 64$ and the learning rate $\eta_t = 0.005$. The number of knots $n_{\text{knots},s}$ is chosen at hand for each dataset. Further details on hyperparameter tuning are provided in Appendix F.

E BASELINES

In the following, we describe the baseline methods in detail. These are two naïve semi-parametric plug-in estimators: mixture density networks (**MDNs**) (Bishop, 1994) and conditional normalizing flow (**CNF**) (Trippe & Turner, 2018). Further, we use two state-of-the-art IDE baselines: kernel density estimation (**KDE**) (Kim et al., 2018) and distributional kernel mean embeddings (**DKME**) (Muandet et al., 2021).

E.1 NAÏVE SEMI-PARAMETRIC PLUG-IN ESTIMATORS

Semi-parametric plug-in estimators estimate the conditional outcome distribution and perform averaging over covariates during evaluation, as introduced in Eq. (2).

Both MDNs and CNF make use of hypernetworks (Ha et al., 2017), which take covariates X and treatment A as an input and output parameters, i. e., $\theta(X, A)$ of the estimated conditional distribution $\hat{\mathbb{P}}(Y \mid X, A)$. For comparability, we use the same network structure of the teacher flow in our INFs as the hypernetwork for the conditional distribution parameters. This gives two fully-connected subnetworks stacked on each other, i. e. FC_1 and FC_2, as introduced in Section 5.1. To regularize both conditional distribution estimators, we use noise regularization (Rothfuss et al., 2019).

MDNs. Mixture density networks (Bishop, 1994) are built on top of mixture of normal distributions, and can approximate any density arbitrarily well (Titterington et al., 1985), i.e.,

$$\hat{\mathbb{P}}(Y = y \mid X, A) = \sum_{j=1}^{n_C} w_j(X, A)\, N(y; \mu_j(X, A), \sigma_j^2(X, A)) \tag{18}$$

where n_C is a number of mixture components, $w_j \geq 0, \sum_{j=1}^{n_C} w_j = 1$ are mixture weights, and $N(y; \mu_j, \sigma_j^2)$ is a density of the normal distribution. In the case of MDNs, the hypernetwork outputs logits of mixture weights and parameters of the normal distribution (i.e., mean and logarithm of the standard deviation), i. e., $\theta = \{\text{logits}(w_j), \mu_j, \log \sigma_j\}$. Here, the number of mixture components n_C controls the smoothness of the estimator and represents the main hyperparameter for tuning.

CNF. We implement conditional normalizing flow (Trippe & Turner, 2018) with the help of neural spline flows (Durkan et al., 2019). Neural spline flows construct an invertible function parameterized by θ, i. e., $f(\cdot; \theta) : \mathbb{R} \to \mathbb{R}$, which is a monotonic rational-quadratic spline with n_{knots} knots. This spline transforms the density of a base distribution on the interval $[-B; B]$. Outside of the interval, $f(\cdot)$ equals to the identity function. This allows us to perform flexible parametric density estimation with the help of the change of variables formula, i.e.,

$$\hat{\mathbb{P}}(Y = y \mid X, A) = N\Big(f^{-1}\big(y; \theta(X, A)\big); 0, 1\Big) \left| \big(f^{-1}(y; \theta(X, A))\big)' \right| \tag{19}$$

where $f^{-1}(\cdot; \theta)$ is the inverse transformation, and the density of standard normal distribution $N(y; 0, 1)$ serves as a base distribution. As already discussed in Appendix D, B affects the support of transformation, and the number of knots n_{knots} controls the smoothness of the estimator and represents the main hyperparameter for tuning.

E.2 KERNEL DENSITY ESTIMATION (KDE)

Kernel density estimation (KDE) is a semi-parametric method for IDE (Kim et al., 2018). It builds upon the idea of a density functional, namely $T_y(Y; h)$, to transform a random variable Y into a proper density via

$$T_y(Y; h) = \frac{1}{h} K\left(\frac{\|Y - y\|_2}{h}\right) = \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{\|Y - y\|_2^2}{2h^2}\right), \tag{20}$$

where $K(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$ is a radial basis function (RBF) with a smoothing parameter h called bandwidth, and $\|\cdot\|_2$ is the L_2-norm.

Robins & Rotnitzky (2001) proposed a semi-parametric plug-in estimator of interventional density

$$\hat{\mathbb{P}}^{\text{PI}}(Y[a] = y) = \mathbb{P}_n\Big\{\hat{\mathbb{E}}\big(T_y(Y; h) \mid X, A = a\big)\Big\}, \tag{21}$$

where $\hat{\mu}_{a,y}(X) = \hat{\mathbb{E}}\big(T_y(Y;h) \mid X, A=a\big)$ is a functional regression of X and A on $T_y(Y;h)$. Kim et al. (2018) further extended this estimator to an efficient, A-IPTW-style semi-parametric estimator

$$\hat{\mathbb{P}}^{\text{A-IPTW}}(Y[a]=y) = \mathbb{P}_n\left\{\frac{\mathbb{1}(A=a)}{\hat{\pi}_a(X)}\Big(T_y(Y;h) - \hat{\mu}_{a,y}(X)\Big) + \hat{\mu}_{a,y}(X)\right\}, \tag{22}$$

where $\hat{\pi}_a(X)$ is an estimator of the propensity score.

The main challenge here is building a functional regression $\hat{\mu}_{a,y}(X)$. Unfortunately, the work by Kim et al. (2018) does not provide effective, practical solutions. Even more so, Eq. (22) does not guarantee that the estimated density is proper, i. e., integrates to 1 and is positive, especially in a small sample regime or when the propensity score has extremely low values.

To estimate the nuisance parameters, namely, the propensity score and the functional regression, we use the same network structure as for the teacher flow of our INFs (see Section 5.1). In this way, we estimate the propensity score and perform a functional regression with two joined, fully-connected subnetworks (i.e., FC_1 and FC_2). The first subnetwork, FC_1, outputs a representation R and estimates the propensity score. The second subnetwork, FC_2, then takes the representation R and the treatment A, and performs an outcome regression: $\hat{Y} = \hat{\mathbb{E}}(Y \mid X, A)$. The functional expression, i. e., Eq. (20), is predicted via $\hat{\mu}_{a,y}(X) = T_y(\hat{\mathbb{E}}(Y \mid X, A); h)$. Although, this is a biased estimator of $\mu_{a,y}(X)$, it ensures a proper normalization, i.e., $\int_{\mathcal{Y}} \hat{\mu}_{a,y}(X)\,\mathrm{d}y = 1$.

To fit FC_1 and FC_2, we use the sum of mean-squared error ($\mathcal{L}_{\text{MSE}}$) and binary cross-entropy ($\mathcal{L}_\pi$) losses via

$$\mathcal{L}_{\text{KDE}}(\hat{\mathbb{E}}, \hat{\pi}_a) = \mathbb{P}_n\{\mathcal{L}_{\text{MSE}} + \alpha\mathcal{L}_\pi\} \text{ with } \mathcal{L}_{\text{MSE}} = (\hat{Y} - Y)^2; \quad \mathcal{L}_\pi = \text{BCE}(\hat{\pi}_A(X), A), \tag{23}$$

where α is a hyperparameter. In our experiments, we set $\alpha = 1$ and fit the nuisance parameters (i.e., $\hat{\pi}_a$ and $\hat{\mathbb{E}}(Y \mid X, A)$) using the Adam optimizer with $n_{\text{iter}} = 10000$ iterations. Both learning rate η and minibatch size b are subject to hyperparameter tuning.

We employ a median heuristic (Garreau et al., 2017) for choosing the bandwidth h, i.e.,

$$h_{\text{med}} = \sqrt{\frac{1}{2}\,\text{Median}\left(\|Y_i - Y_j\|_2^2\right)}, \quad 1 \le i < j \le n, \tag{24}$$

where $\|\cdot\|_2$ is the L_2-norm, and where Y_i, Y_j are observations from the train subset. To address the numeric instability of the A-IPTW estimator, we clip the propensity score to the 0.1-quantile of the empirical distribution based on the train subset, i. e., $\{\hat{\pi}_a(X_i)\}$.

E.3 DKME

Distributional kernel mean embeddings (DKME) (Muandet et al., 2021) is a non-parametric plug-in estimator of interventional densities. This methods builds a kernel mean embedding (KME), namely, $\mu_{Y|X,A=a}$, for the conditional distribution $\mathbb{P}(Y \mid X, A=a)$ via

$$\mu_{Y|X,A=a}(y) \coloneqq \mathop{\mathbb{E}}_{Y\sim\mathbb{P}(Y|X,A=a)} l(y, Y), \tag{25}$$

where $l(\cdot,\cdot)$ is a measurable positive definite kernel associated with a reproducing kernel Hilbert space $\mathcal{H}$, so that $\mu_{Y|X,A=a}$ provides a mapping from the space of conditional distributions to the space of functions $\mathcal{H}$. If $l(\cdot,\cdot)$ is properly normalized, then $\mu_{Y|X,A=a}(y)$ is in fact a conditional density estimator.

To estimate the KME of the conditional outcome distribution (conditional mean embedding), we use the i.i.d. sample $\mathcal{D} = \{X_i, A_i, Y_i\}_{i=1}^n$, and split it into control and treated subsamples: $\mathcal{D} = \{X_i^0, Y_i^0\}_{i=1}^{n_0} \cup \{X_i^1, Y_i^1\}_{i=1}^{n_1}$. Then, $\mu_{Y|X,A=a}$ can be estimated via

$$\hat{\mu}_{Y|X,A=a}(y) = \sum_{i=1}^{n_a} w_i^a(X)\, l(y, Y_i^a), \tag{26}$$

$$\big(w_1^a(X), \ldots, w_{n_a}^a(X)\big)^\intercal = \left(\mathbf{K}^a + n_a\varepsilon\mathbf{I}\right)^{-1}\mathbf{k}^a(X) \in \mathbb{R}^{n_a}, \tag{27}$$

$$\mathbf{k}^a(X) = \big(k(X, X_1^a), \ldots, k(X, X_{n_a}^a)\big)^\intercal \in \mathbb{R}^{n_a}, \tag{28}$$

where $\mathbf{I} \in \mathbb{R}^{n_a \times n_a}$ is an identity matrix, $\varepsilon > 0$ is a regularization hyperparameter, $\mathbf{K}^a \in \mathbb{R}^{n_a \times n_a}$ is a kernel matrix with elements $\mathbf{K}^a_{ij} = k(X^a_i, X^a_j)$, and $k(\cdot, \cdot)$ is a second kernel representing conditional dependencies between X and Y (Grünewälder et al., 2012).

Muandet et al. (2021) further developed a KME for interventional distribution, i. e., $\mu_{Y[a]}$, and its empirical estimate, $\hat{\mu}_{Y[a]}$:

$$\mu_{Y[a]}(y) = \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X)} \mu_{Y|X,A=a}(y) \tag{29}$$

$$\hat{\mu}_{Y[a]}(y) = \mathbb{P}_n\{\hat{\mu}_{Y|X,A=a}(y)\} = \sum_{i=1}^{n_a} \beta^a_i \, l(y, Y^a_i), \tag{30}$$

$$(\beta^a_1, \ldots, \beta^a_{n_a})^\intercal = (\mathbf{K}^a + n_a \varepsilon \mathbf{I})^{-1} \, \tilde{\mathbf{K}}^a \, \mathbf{1}_m \in \mathbb{R}^{n_a}, \tag{31}$$

where $\tilde{\mathbf{K}}^a \in \mathbb{R}^{n_a \times n}$ is a kernel matrix with elements $\tilde{\mathbf{K}}^a_{ij} = k(X^a_i, X_j)$, and $\mathbf{1}_m = (1/n, \ldots, 1/n)^\intercal$.

For our experiments, we choose both kernels, i. e., outcome kernel, $l(\cdot, \cdot)$, and conditional kernel, $k(\cdot, \cdot)$, to be RBF kernels with bandwidth parameters h_l and h_k, respectively. Therefore, $\hat{\mu}_{Y[a]}(y)$ represents a valid interventional density estimator. Nevertheless, due to small sample sizes, some β^a_i could be negative and the estimated density ends up having negative values.

We set the bandwidth of the outcome kernel according to the median heuristic from Eq. (24). The bandwidth of the conditional kernel h_k and the regularization hyperparameter ε are subjects to the hyperparameter tuning. Motivated by the interpretation of conditional mean embedding as kernel ridge regression (Grünewälder et al., 2012), we use out-sample MSE of the ridge regression with parameters h_k and ε as a tuning criterion.

F HYPERPARAMETER TUNING

We performed hyperparameters tuning for all the baselines based on five-fold cross-validation using the train subset. For each baseline, we performed a grid search with respect to different tuning criteria, evaluated on the validation subsets. Table 5 shows grids for hyperparameter tuning and other parameters, such as tuning criteria, number of training iterations, and optimizers. We aimed for a fair comparison and thus kept the number of parameters, network structures, and grid size similar across models. For the sake of reproducibility, we make the chosen hyperparameters for all the experiments public (see YAML files in our GitHub[9]).

Importantly, to facilitate the convergence of baseline methods, we additionally perform a standard normalization of both factual and counterfactual outcomes for all the datasets.

Table 5: Hyperparameter tuning for baselines.

Model	Sub-model	Hyperparameter	Range / Value
MDNs	—	Number of mixture components (n_C)	5, 10, 20
		Intensity of noise regularization (σ_x^2)	0.0, 0.01, 0.05, 0.1
		Intensity of noise regularization (σ_y^2)	0.0, 0.01, 0.05, 0.1
		Learning rate (η)	0.001, 0.005
		Minibatch size (b)	32, 64
		Tuning strategy	random grid search with 50 runs
		Tuning criterion	$\mathcal{L}_{\text{NLL}}$
		Number of train iterations (n_{iter})	5000
		Optimizer	SGD (momentum = 0.9)
KDE	—	Learning rate (η)	0.001, 0.005, 0.1
		Minibatch size (b)	32, 64, 128
		Tuning strategy	full grid search
		Tuning criterion	$\mathcal{L}_{\text{MSE}} + \alpha\mathcal{L}_{\pi}$
		Number of train iterations (n_{iter})	10000
		Optimizer	Adam (betas=(0.9, 0.999))
DKME	—	Kernel smoothness ($\sigma_k = 2h_k^2$)	0.0001, 0.001, 0.01, 0.1, 1, 10, 20
		Regularization parameter (ε)	0.0001, 0.001, 0.01, 0.1, 1, 10
		Tuning strategy	full grid search
		Tuning criterion	MSE of ridge regression
INFs	Teacher flow	Number of knots ($n_{\text{knots},t}$)	5, 10, 20
		Intensity of noise regularization (σ_x^2)	0.0, 0.01, 0.05, 0.1
		Intensity of noise regularization (σ_y^2)	0.0, 0.01, 0.05, 0.1
		Learning rate (η_t)	0.001, 0.005
		Minibatch size (b_t)	32, 64
		Tuning strategy	random grid search with 50 runs
		Tuning criterion	$\mathcal{L}_{\text{NLL}}$
		Number of train iterations ($n_{\text{iter},t}$)	5000
		Optimizer	SGD (momentum = 0.9)
	Student flow	Number of knots ($n_{\text{knots},s}$)	dataset specific*
		Learning rate (η_s)	0.005
		Minibatch size (b_s)	64
		Tuning strategy	w/o tuning
		Number of train iterations ($n_{\text{iter},s}$)	2000
		Optimizer	Adam (betas=(0.9, 0.999))

* $n_{\text{knots},s} = 5$ (synthetic data), $= 10$ (IHDP, HC-MNIST datasets), $= n_{\text{knots},t}$ (ACIC 2016 & 2018 datasets)

[9] https://anonymous.4open.science/r/AnonymousInterFlow-E2F3

G DATASET DETAILS

G.1 SYNTHETIC DATASET

We sample $n = 1000$ observations from the SCM (Fig. 1) and use a ten-fold split for train/test samples (90%/10%). We separately perform hyperparameter tuning based on the first split for each baseline and each level b. We then report an average out-sample log-likelihood over ten folds.

G.2 ACIC 2016 & 2018 DATASETS

Covariates of ACIC 2016 are taken from a large study of developmental disorders (Niswander, 1972), and covariates of ACIC 2018 are derived from the linked birth and infant death data (MacDorman & Atkinson, 1998). ACIC 2016 and ACIC 2018 differ in the number of true confounders, the varying level of overlap, and the form of conditional outcome distributions. ACIC 2016 has 77 different data-generating mechanisms with 100 equal-sized samples for each mechanism ($n = 4802$, $d_X = 82$).[10] ACIC 2018 provides 63 distinct data-generating mechanisms with around 40 non-equal-sized samples for each mechanism (n ranges from $1,000$ to $50,000$, $d_X = 177$). Notably, ACIC 2018 has a constant ITE for most of the datasets, but heterogeneous propensity scores. During experiments, we set a time budget of 10 minutes per run and discard models, which exceed it (for exact runtimes, see Appendix I).

G.3 HC-MNIST

Jesson et al. (2021) introduced a complex high-dimensional, semi-synthetic dataset based on the MNIST image dataset LeCun (1998), namely HC-MNIST. This dataset maps high-dimensional images onto a one-dimensional manifold, where potential outcomes depend in a complex way on the average intensity of light and the label of an image. The treatment also uses this one-dimensional summary together with an additional (hidden) synthetic confounder. The parameter Γ^* defines what factor influences the treatment assignment to a larger extent, i.e., the additional confounder or the one-dimensional summary. We set $\Gamma^* = \exp(1)$. For further details, we refer to (Jesson et al., 2021).

For the experiments with HC-MNIST, we use a larger network size for our INFs (compared to other benchmarking experiments) to allow for more flexibility. We set the number of hidden units in fully-connected subnetworks to $h = 30$, and the dimensionality of representation $r = 30$. We also increase the number of training iterations to $n_{\text{iter},t} = 15,000$ and $n_{\text{iter},s} = 5000$.

Fig. 4 shows both ground-truth interventional ($\mathbb{P}(Y[a])$) and observational ($\mathbb{P}(Y \mid A = a)$) distributions together with our INFs A-IPTW estimator ($\hat{\mathbb{P}}^{\text{A-IPTW}}(Y[a])$). Remarkably, the interventional distributions in HC-MNIST are multi-modal and differ a lot from observational distributions.

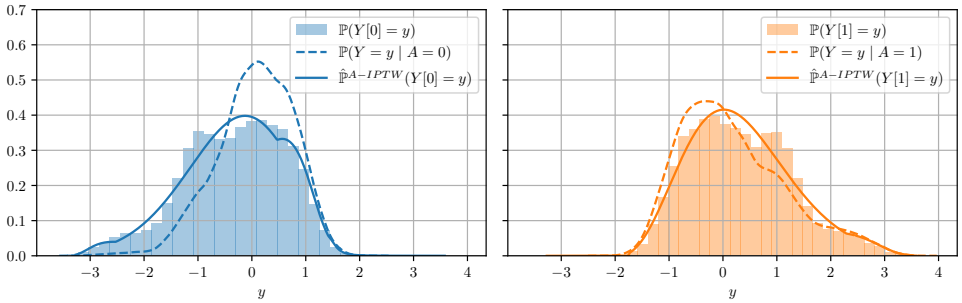


Figure 4: Ground-truth interventional and conditional distributions of the HC-MNIST synthetic outcome. We also plot our INFs density estimator, i. e., $\hat{\mathbb{P}}^{\text{A-IPTW}}(Y[a])$.

[10] After one-hot-encoding of categorical covariates.

H ADDITIONAL RESULTS FOR IHDP

Here, we provide additional results for the IHDP dataset with an alternative evaluation metric, that is, the empirical Wasserstein distance.

Evaluation metric. For one-dimensional outcomes, the Wasserstein distance between two distributions can be simply expressed via quantile functions

$$W^p(\mathbb{P}_1, \mathbb{P}_2) = \left(\int_0^1 |\mathbb{F}_1^{-1}(q) - \mathbb{F}_2^{-1}(q)|^p \, \mathrm{d}q \right)^{1/p}, \tag{32}$$

where $\mathbb{F}_1^{-1}(q)$ and $\mathbb{F}_2^{-1}(q)$ are quantile functions of $\mathbb{P}_1$ and $\mathbb{P}_2$, respectively. The Wasserstein distance is not upper-bounded and equals zero if and only if both distributions are the same. Here, we compute the empirical Wasserstein distance, i. e., $\hat{W}^1$, based on empirical quantile functions. This requires two samples: one from the ground-truth interventional distribution and another from the estimated density. Therefore, methods which do not provide direct sampling (e. g., KDE and DKME) cannot be used for evaluation.

Table 6 shows the results. Here, our INFs perform best almost in every setting. It is superior over the two naïve plug-in estimators and the variant of INFs without bias correction. In sum, the results corroborate our findings from the main paper and add to the effectiveness of our INFs.

Table 6: Additional results for semi-synthetic experiments using the IHDP dataset. The performance is benchmarked using the empirical in-sample / out-sample Wasserstein distance (i.e., $\hat{W}^1_{\text{in}}$ and $\hat{W}^1_{\text{out}}$) for the two potential outcomes (i.e., $a = 0$ and $a = 1$). Reported: mean $\pm$ standard deviation over ten-fold train-test splits.

	$a = 0$		$a = 1$	
	$\hat{W}^1_{\text{in}}$	$\hat{W}^1_{\text{out}}$	$\hat{W}^1_{\text{in}}$	$\hat{W}^1_{\text{out}}$
MDNs	0.067 ± 0.053	0.156 ± 0.054	0.121 ± 0.076	0.183 ± 0.071
CNF [$\hat{=}$ INFs w/o stud flow]	0.118 ± 0.048	0.192 ± 0.069	0.111 ± 0.087	$\mathbf{0.146} \pm \mathbf{0.082}$
INFs w/o bias corr	0.081 ± 0.025	0.182 ± 0.054	0.109 ± 0.049	0.147 ± 0.067
INFs (main)	$\mathbf{0.043} \pm \mathbf{0.008}$	$\mathbf{0.130} \pm \mathbf{0.031}$	$\mathbf{0.089} \pm \mathbf{0.035}$	0.162 ± 0.049

Lower = better (best in bold)

I RUNTIME COMPARISON

INFs are a fully-parametric model and, therefore, provide a decent speed up at the inference time. This is particularly important for scalability, that is, for datasets with large sample size and high-dimensional covariates. In Table 7, we report the total runtime of the baselines and the different variants of our INFs. We see that the runtime of both full INFs and INFs (CA) stay relatively constant, but for the other baselines, it grows polynomially. This demonstrates the benefits of our INFs for scalability.

Table 7: Total runtime (in minutes) of the experiments using ACIC 2018 datasets with different sample sizes. Reported: mean $\pm$ standard deviation over four datasets and five runs for each size (lower is better). Experiments are carried out on Intel(R) Xeon(R) Silver 4316 CPU @ 2.30GHz.

Sample size	1000	2500	5000	10000	25000	50000
MDNs	0.29 ± 0.05	0.34 ± 0.03	0.54 ± 0.05	1.02 ± 0.05	5.06 ± 0.71	$\gg 10.0$
CNF [$\hat{=}$ INFs w/o stud flow]	0.58 ± 0.07	0.82 ± 0.11	1.14 ± 0.17	2.33 ± 0.23	8.95 ± 1.71	$\gg 10.0$
KDE (Kim et al., 2018)	0.41 ± 0.05	0.45 ± 0.01	0.47 ± 0.05	0.67 ± 0.06	1.35 ± 0.09	6.05 ± 1.28
DKME (Muandet et al., 2021)	0.03 ± 0.01	0.04 ± 0.01	0.12 ± 0.07	0.27 ± 0.02	2.07 ± 0.38	$\gg 10.0$
INFs w/o bias corr	0.92 ± 0.06	0.92 ± 0.07	0.92 ± 0.05	0.95 ± 0.06	1.03 ± 0.06	1.08 ± 0.02
INFs (main)	1.44 ± 0.05	1.44 ± 0.06	1.44 ± 0.06	1.49 ± 0.06	1.55 ± 0.07	1.56 ± 0.03