DNet: Distributional Network for Distributional Individualized Treatment Effects

Guojun Wu
wuguojun.wu@bytedance.com
Bytedance
Beijing, China

Ge Song
songge.cindy@bytedance.com
Bytedance
Beijing, China

Xiaoxiang Lv
lvxiaoxiang.misaka@bytedance.com
Bytedance
Beijing, China

Shikai Luo*
shadow.luo@bytedance.com
Bytedance
Beijing, China

Chengchun Shi*
c.shi7@lse.ac.uk
London School of Economics and Political Science
London, United Kingdom

Hongtu Zhu*
htzhu@email.unc.edu
University of North Carolina at Chapel Hill
Chapel Hill, United States

ABSTRACT

There is a growing interest in developing methods to estimate individualized treatment effects (ITEs) for various real-world applications, such as e-commerce and public health. This paper presents a novel architecture, called DNet, to infer distributional ITEs. DNet can learn the entire outcome distribution for each treatment, whereas most existing methods primarily focus on the conditional average treatment effect and ignore the conditional variance around its expectation. Additionally, our method excels in settings with heavy-tailed outcomes and outperforms state-of-the-art methods in extensive experiments on benchmark and real-world datasets. DNet has also been successfully deployed in a widely used mobile app with millions of daily active users.

CCS CONCEPTS

• **Computing methodologies → Machine learning algorithms**.

KEYWORDS

uplift modeling, causal inference, quantile regression

ACM Reference Format:
Guojun Wu, Ge Song, Xiaoxiang Lv, Shikai Luo, Chengchun Shi, and Hongtu Zhu. 2023. DNet: Distributional Network for Distributional Individualized Treatment Effects. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599809

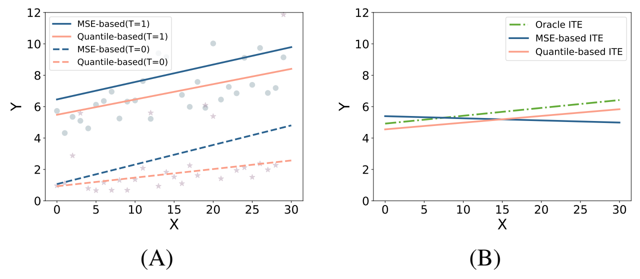


Figure 1: A toy simulation example to visualize the disadvantage of standard ITE estimators with heavy-tailed outcomes. Panel A plots the data distribution for treatments 0 and 1 with circles and stars, respectively. The blue and orange lines are the conditional mean estimators obtained by minimizing the MSE and quantile losses, respectively. Panel B displays the corresponding ITE estimators, computed by subtracting the two mean curves in Panel A. The green dashed line depicts the oracle ITE values. The outcomes are generated from a heavy-tailed distribution.

1 INTRODUCTION

The individualized treatment effect (ITE) has been widely studied in various fields, including medical science, psychology, sociology, economics, e-commerce, and education. Recently, technology companies have been using ITE estimation to optimize their marketing efforts [10, 39, 40, 42, 43]. There is a growing trend towards using statistical and machine learning methods to estimate ITEs from observational data [1, 6, 11, 21, 25, 27, 32–34]. However, most existing research has focused on the conditional average treatment effect (CATE), which is the average individual treatment effect given certain baseline characteristics. This approach is not robust to outliers and may not provide accurate estimators when the outcome distribution is skewed. See Figure 1 for an illustration. Circles and stars indicate heavy-tailed outcomes under two treatments. Panel A depicts the conditional mean estimators trained by minimizing the mean squared error (MSE) and quantile loss. Panel B displays the corresponding ITE estimators obtained by subtracting the two conditional mean curves. The MSE-based ITE estimator is biased, highlighting the danger of ignoring heavy tailedness. The slope of the quantile-based ITE estimator is consistent, demonstrating its robustness against outliers.

This paper aims to learn distributional ITEs in order to capture the intrinsic uncertainty of the counterfactual outcome. A key observation is that, a given distribution function can be approximated

*These authors are senior authors of this paper.

by using a set of quantiles [23, 36]. This motivates us to consider quantile ITEs at a given set of quantile levels. However, a major challenge in this process is that many existing quantile estimators may not necessarily satisfy the monotonicity constraint, i.e., the estimator at the q_1th quantile is strictly larger than that at the q_2th quantile for some $q_1 < q_2$, referred to as "crossing." This occurs when quantile regression is conducted at each level without imposing a non-crossing constraint. Crossing leads to instability in treatment selections, reduces the estimator's statistical efficiency and harms the model's interpretability. Although it has been extensively studied in the statistics and economics literature [5, 7, 9, 12, 13, 20, 22, 24], most existing works focus on linear quantile regression models. Non-crossing and non-linear quantile regression has been less studied in the literature. This paper is to fill in this gap by imposing explicit non-crossing constraints on the quantile regression while employing existing state-of-the-art deep learning techniques to capture the nonlinear causal relationships among the variables.

Our major contributions are summarized as follows:

(a) This paper is pioneering in studying non-crossing quantile regression for ITE estimation. We propose a distributional network (DNet) architecture, based on non-crossing quantile regression, to handle heavy-tailed outcomes. This architecture can be utilized as a standalone component and incorporated into other neural network models to solve other distributional learning tasks.

(b) We further develop two novel adaptations of DNet to address some common challenges in practical applications, including monotone treatment effects and zero-inflated outcomes.

(c) We conduct extensive empirical studies by comparing our proposal against existing state-of-the-art on a number of benchmark and real-world datasets. The proposed DNet provides more accurate treatment effects estimation in our applications, and generate better personalized intervention policies in the production environment of a well-known tech company, serving millions of users each day.

The rest of this paper is structured as follows: Section 2 reviews related works on treatment effects estimation and quantile regression. Section 3 introduces the background and some notation. Section 4 describes the DNet architecture as well as its training and inference procedures. In Section 5, we discuss how DNet can be adapted to handle zero-inflated outcomes and monotone treatment effects. Section 6 presents extensive experimental evaluations of DNet on benchmark and real-world datasets, as well as the online deployment results. Finally, we conclude our findings in Section 7.

2 RELATED WORK

The proposed methods are closely related to three different areas in causal inference and machine learning, including conditional average treatment effect estimation, conditional distributional treatment effect estimation and quantile regression.

2.1 Conditional Average Treatment Effect Estimation

In recent years, there is a growing interest in developing machine learning methods for conditional average treatment effect (CATE) estimation. For instance, [21, 27] developed meta-learners which effectively combine several base-learners computed by existing supervised learning or regression method in order to estimate the CATE function. Another line of research focused on developing flexible non-parametric heterogeneous treatment effect estimation based on random forest [1, 28, 34]. These tree-based methods can produce consistent estimators with fast convergence rate, and valid confidence intervals. However, they can hardly be adapted to incremental training and tend to be ineffective with sparse categorical features. Finally, [18, 32, 33, 41] proposed original neural network architectures to simultaneously model multiple potential outcomes. In particular, DragonNet [33] provides an end-to-end procedure based neural networks for simultaneously estimating the propensity score and the expected outcomes given the covariates and the treatment. [41] developed a collaborating causal network to learn the full potential outcome distribution.

2.2 Conditional Distributional Treatment Effect Estimation

Recently, a few methods have been developed in the literature for distributional treatment effects estimation or policy evaluation [2, 4, 8, 15, 26, 29, 37, 38]. In particular, [8] proposed to discretize Y and to employ regression to estimate the resulting conditional cumulative distribution function (CCDF). However, the estimated CCDF is not guaranteed to satisfied the monotonicity constraint. [4, 15] proposed to apply generalized additive models for location, scale, and shape [GAMLSS, 30] for learning conditional distributional treatment effects [4, 15]. However, it relies crucially on the additive assumption and suffers from model misspecification. [26, 29] consider learning distributional treatment effect via kernel conditional mean embeddings which embeds the counterfactual outcome distribution into a reproducing kernel Hilbert space (RKHS). However, empirical evidence suggests that kernel-based methods deteriorate significantly when the covariate space is high-dimensional.

2.3 Quantile Regression

Commonly used in statistics and econometrics, quantile regression estimates the conditional quantiles of a given outcome, as opposed to linear regressions which focus on the mean outcome. There are two main advantages of quantile regression over linear regression. First, it is more robust against outliers. Second, different from linear regression which only computes a conditional mean function, quantile regression produces conditional estimators at a range of quantile levels. In that sense, quantile regression provides more information. A number of quantile regression methods have been developed in the literature [see e.g., 7, 9, 12, 13, 19, 20, 22, 23, 35]. However, it is common to encounter the crossing problem, where the estimated quantiles violate the monotonicity property. To address this issue, a few methods have been proposed to impose the non-crossing constraints [3, 9, 13, 23]. However, these methods are very computationally intensive and are mostly designed for parametric quantile models. As such, it requires the development of flexible and efficient optimization algorithm to solve the non-crossing issue.

3 BACKGROUND AND PROBLEM FORMULATION

In this section, we apply the Neyman-Rubin potential outcomes framework [see e.g., 17, 31] for problem formulation. We begin with some notations and assumptions. The underlying probability space can be represented by $(\Omega, \mathcal{F}, \mathcal{P})$, where $\Omega = \mathcal{X} \times \mathcal{T} \times \mathcal{Y}$ with the input space $\mathcal{X} \subseteq R^p$, treatment space $\mathcal{T} = 0, 1, 2, \ldots, M-1$ where M denotes the number of treatment options (0 denotes the control by convention), and output space $\mathcal{Y} \subseteq R$. Random variables $X\colon \Omega \to \mathcal{X}$, $T\colon \Omega \to \mathcal{T}$, and $Y^*(t)\colon \Omega \to \mathcal{Y}$ represent the covariates, treatment assignment, and the counterfactual outcome that would have been observed if treatment t were assigned. In practice, it is impossible to observe all the potential outcomes. Instead, we only have access to the observed outcome $Y\colon \Omega \to \mathcal{Y}$, depending on the treatment we receive (see the consistency assumption in (A1) below). This missing data issue is known as the fundamental problem of causal inference [16], prevents us from directly computing the difference in outcomes under treatments and control for each unit. The observed data correspond to i.i.d. copies of the triplet (X, T, Y), given by $(x_i, t_i, y_i)_{i=1}^N$.

Accurate estimation of the ITE is essential in many practical applications. In particular, this paper is motivated by the applications in a world-leading video creation and sharing platform. The company aims to implement a 'pop-up' notification policy to inform users of the latest updates and new features in their updated application, while maintaining a seamless user experience. To successfully implement this policy, we must accurately assess the impact of pop-up windows on user engagement, i.e. the treatment effect with and without pop-ups. In this case, T is binary and indicates whether the pop-up window appears or not when a user launches the app. X and Y correspond to historical behavior of the user in regards to their usage of the app and their daily active time, respectively.

Throughout this paper, we assume that the following three assumptions hold.

(A1) $Y = Y^*(T)$.

(A2) T is independent of $(Y^*(0), Y^*(1), \ldots, Y^*(M-1))$ given X.

(A3) $b(t|x)\colon = P(T = t|X = x) > 0$ for $\forall x, t$.

Assumption (A1) is known as the consistency assumption. Assumption (A2) asserts that there are no unmeasured confounding variables, which is automatically satisfied in randomized studies where the treatment assignment mechanism depends only upon the feature X. Assumption (A3) is referred to as the positivity or overlap assumption that is commonly imposed in the literature.

These assumptions ensure that the causal effects can be inferred from the observed dataset. In particular, they imply that $P_{Y(t)|X} = P_{Y(t)|X,T=t} = P_{Y|X,T=t}$ for all t where $P_{U|V}$ denotes the conditional distribution of U given V for any random variables U and V. We use $b(t|x)$ to denote the propensity score, which is known in a randomized controlled trial, and equals $1/M$ in a randomized experiment.

The primary interest in the causal inference literature lies in inferring the average treatment effect (ATE), $\tau_t = E[Y^*(t) - Y^*(0)]$, and the conditional average treatment effect (CATE), $\tau_t(x) = E[Y^*(t) - Y^*(0)|X = x]$. As commented earlier, these causal estimands describe the averaged (heterogeneous) treatment effects, but do not capture the uncertainty around the mean (e.g., variance) or other higher order moments. This motivates us to consider the distributional treatment effect. We also remark that distributional treatment effects play an important role in making fair and explainable decisions, since they provide a more comprehensive understanding of the treatment effect.

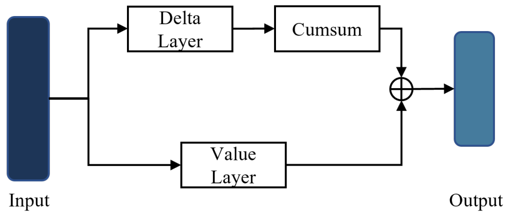


Figure 2: A graphical illustration of the Non-Crossing Quantile Layer. The Value Layer aims to learn the averge of all quantiles and the Delta Layer aims to learn the differences between any two adjacent quantiles.

4 METHODOLOGY

In this section, we present our method for estimating distributional treatment effects using non-crossing deep quantile regression. We begin by introducing a non-crossing quantile layer. Next, we formally present the proposed distributional network architecture, discussing some practical implementation details.

4.1 Non-Crossing Quantile Layer

The non-crossing quantile layer is used to estimate the outcome distribution under a given treatment t. To simplify the notation, we omit all dependencies on t throughout this section. Let $\gamma = (\gamma_1, \ldots, \gamma_K)$ denote a set of K pre-specified non-decreasing quantile levels, typically chosen as $\gamma_k = k/(K+1)$ for $k = 1, \cdots, K$. A standard quantile layer is defined as follows:

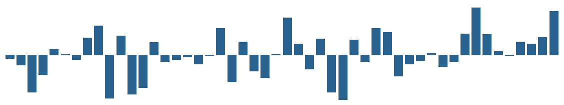
$$\mathbf{q} := f(x; \theta_q),$$

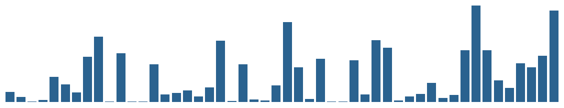
where x denotes the input data, θ_q is the set of parameters in the quantile layer, $\mathbf{q}$ denotes the output quantiles in ascending order, i.e. $\mathbf{q} = (q_{\gamma_1}, \ldots, q_{\gamma_K})$ where q_{γ_k} represents the γ_kth conditional quantile of Y given $X = x$ and $T = t$, and f denotes certain neural network model.

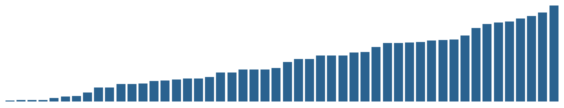
As commented earlier, the learned quantile curves may not satisfy the monotonicity constraint, reducing the estimation accuracy and hurting the model interpretability. To address this issue, we propose a Non-Crossing Quantile (NCQ) Layer. See Figure 2 for a graphical illustration. The NCQ Layer essentially replaces the output layer in the standard quantile layer with two independent dense layers: the *Value Layer* and the *Delta Layer*, which share the same inputs. It is formally defined as:

$$NCQ(x) = v(x; \theta_v) \oplus d(x; \theta_\delta), \tag{1}$$

where $v(\cdot; \theta_v)$ and $d(\cdot; \theta_\delta)$ denote the *Value Layer* and *Delta Layer*, respectively, and the $\oplus$ symbol denotes a broadcast operator to perform a broadcast add between the scalar-valued $v(\cdot; \theta_v)$ and the K-dimensional output $d(\cdot; \theta_\delta)$. The *Value Layer* is a dense layer and corresponds to the average of the predicted quantiles. The *Delta Layer* outputs a K-dimensional vector $\mathbf{c} = g(\cdot; \theta_\delta)$. Both layers are


(A) Output of a base dense layer.


(B) Apply the activation function σ to create non-negative outputs.


(C) Apply the cumsum function to generate non-crossing quantiles.

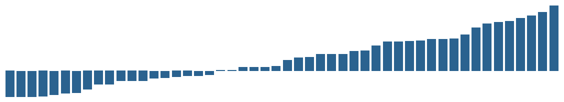

(D) Center the outputs. Perform a broadcast sum between these centered outputs and the value layer.

Figure 3: Visualizing the implementation of non-crossing quantile estimation with 50 quantiles given an arbitrary input.

used to produce the final quantiles $\{q_{\gamma_k}\}_k$ as follows:

$$\begin{aligned} d_{\gamma_k}(\cdot;\theta_\delta) &= \sum_{j=1}^{k}\sigma(c_j) - K^{-1}\sum_{j=1}^{K}(K+1-j)\sigma(c_j), \\ q_{\gamma_k} &= d_{\gamma_k}(\cdot;\theta_\delta) + v(\cdot;\theta_v), \end{aligned} \tag{2}$$

where d_{γ_k} measures the difference between the γ_k-quantile and the average of all the quantiles, and the activation function $\sigma(x) = ELU(x) + 1$ with $ELU(x)$ being an exponential linear unit function, defined as,

$$ELU(x) = \begin{cases} x, & x \geq 0; \\ \alpha(\exp(x) - 1), & x < 0, \end{cases}$$

for some $\alpha > 0$. By definition, the ELU-function rectifies the input to $(-1, +\infty)$. As such, σ is a strictly positive function. It follows from (2) that

$$q_{\gamma_k} - q_{\gamma_{k-1}} = \sigma(c_k) > 0.$$

Consequently, the proposed neural network architecture guarantees that the quantile estimators are monotonically increasing, making the resulting model more interpretable and reliable. A graphical illustration of the NCQ Layer is given in Figure 3.

Finally, it is worth mentioning that the Value Layer and the Delta Layer are allowed to have different neural network architectures. This makes the proposed structure more flexible. For instance, it allows us to impose the monotone treatment constraint on the Value Layer only, as illustrated in Section 5.

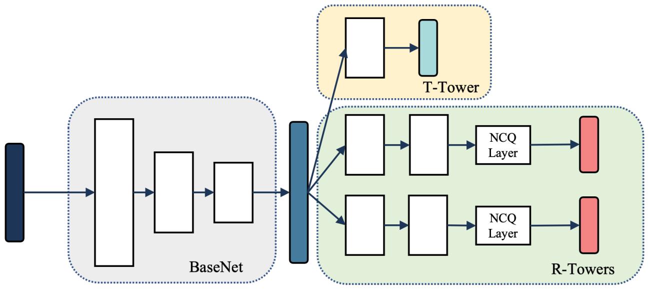


Figure 4: Illustration of DNet architecture with two treatments. The base network learns shared feature representation and the towers produce final predictions.

4.2 Distributional Network

The proposed Distributional Network (DNet) is composed of multiple NCQ Layers. As shown in Figure 4, it consists of three major components:

- A *BaseNet* that learns a shared representation for all treatments.
- A *T-Tower*, a simple softmax layer that estimates the propensity vector, $\pi(x;\theta_\pi) = \{P(T = t|X = x, \theta_\pi)\}_{t=0}^{M-1}$.
- A *R-Tower* associated with each individual treatment t, represented by $R(\cdot, t; \theta_r)$ with the last layer being an NCQ layer,

where θ_π and θ_r denote the network parameters of the *T-Tower* and *R-Tower*, respectively. The *BaseNet* is a bottom network used for feature extraction. Its output will be used as the inputs for the *T-Tower* and *R-Tower*. The *T-Tower* forces the shared representation to be closely tied to the estimated propensity scores. As discussed later, these propensity scores will not be used to produce ITE estimators. Nonetheless, the inclusion of the *T-Tower* helps improve the prediction accuracy [33]. Finally, the *R-Tower* is used to produce the quantile estimators. We next detail the training and inference procedures.

Training of DNet. To train the DNet, we define loss functions for each component. We denote all loss functions as ℓ. The *T-Tower* uses cross-entropy loss to estimate the propensity score. For the *R-Tower*'s, we consider either the quantile loss $\rho_\gamma(u) = u(\gamma - \mathbf{1}\{u < 0\})$, or the quantile Huber loss,

$$\rho_\gamma^\kappa(u) = |\gamma - \mathbf{1}\{u < 0\}|\frac{\ell_\kappa(u)}{\kappa},$$

where $\ell_\kappa(u) = 0.5u^2$ if $|u| \leq \kappa$ and $\kappa(|u| - 0.5\kappa)$ otherwise. We then set the loss function for a single R-Tower to

$$\ell(R(x,t;\theta_r), y) = \frac{1}{K}\sum_{k=1}^{K}\ell_\kappa(y - q_{\gamma_k}(x,t)), \tag{3}$$

where $q_{\gamma_k}(x,t)$ is the kth qauntile output of $R(x,t;\theta_r)$ under treatment t. The final loss function of DNet for a single sample (x_i, t_i, y_i) is given by

$$\mathcal{L}(x_i, t_i, y_i; \theta_r, \theta_\pi) = \ell(R(x_i, t_i; \theta_r), y_i) + \omega\ell(\pi(x_i;\theta_\pi), t_i),$$

where $\ell(\pi(x_i;\theta_\pi), t_i)$ is the cross-entropy loss, and ω is a weight parameter that balances the two loss components.

Inference of DNet. When using a trained model to make predictions on new data, we only use the predicted quantile vector under each treatment and ignore the estimated propensity score. We set the final conditional mean outcome prediction for each treatment to the average of the predicted quantiles, that is,

$$\widehat{E}[Y|X=x,T=t] = \frac{1}{K}\sum_{k=1}^{K} q_{Y_k}(x,t). \tag{4}$$

The treatment effects $\tau_t(x)$ are then estimated as

$$\begin{aligned}\widehat{\tau}_t(x) &= \widehat{E}[Y|X=x,T=t] - \widehat{E}[Y|X=x,T=0]\\ &= \frac{1}{K}\sum_{k=1}^{K} q_{Y_k}(x,t) - \frac{1}{K}\sum_{k=1}^{K} q_{Y_k}(x,0).\end{aligned} \tag{5}$$

Next, we illustrate how to conduct policy optimization based on the outputs of DNet. Optimization in real applications often involves multiple objectives, i.e. stay duration, promotion costs, etc. For illustration purposes, consider two outcomes Y^R and Y^C, where Y^R corresponds to the business metric we wish to maximize and Y^C corresponds to cost Y^C we wish to control. We train different DNets for Y^R and Y^C, denoted as $q_Y^R(x,t)$ and $q_Y^C(x,t)$. Let $q_{ijk}^R = q_{Y_k}^R(x_i,t_j)$, $\bar{q}_{ij}^R = K^{-1}\sum_{k=1}^K q_{ijk}^R$, and $\widehat{\tau}_{ij}^R = \widehat{\tau}_{t_j}^R(x_i)$. Define $q_{ijk}^C, \bar{q}_{ij}^C$, and $\widehat{\tau}_{ij}^C$ similarly. We formalize the policy optimization problem as the following linear programming problem,

$$\begin{aligned}\underset{A}{\text{argmax}}\ &\underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=0}^{M-1}\bar{q}_{ij}^R \times a_{ij}}_{(i)} - \lambda_1 \underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=0}^{M-1}(q_{ijK}^R - q_{ij1}^R)\times a_{ij}}_{(ii)}\\ &- \lambda_2 \underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=0}^{M-1}(q_{ijK}^C - q_{ij1}^C)\times a_{ij}}_{(iii)}, \\ s.t. &\underbrace{\sum_{i=1}^{N}\sum_{j=0}^{M-1}\bar{q}_{ij}^C \times a_{ij}}_{(iv)} \le B, \sum_{j=0}^{M-1} a_{ij} = 1, a_{ij}\in\{0,1\}\end{aligned} \tag{6}$$

where $A = (a_{ij})_{i=1,j=0}^{N,M-1} \in \mathbb{R}^{N\times M}$ with a_{ij} being the indicator of whether to assign user x_i to treatment t_j, and B is the total budget. The treatment assignment matrix A is evaluated based on four criteria: average reward and cost represented by (i) and (iv) respectively, and penalization of treatments with high variance in reward and cost represented by (ii) and (iii) respectively. λ_1 and λ_2 are the hyperparameters chosen via cross-validation to balance the trade-off between these criteria. The inclusion of (ii) and (iii) ensures stable and consistent performance, as well as improves compliance with budgeted costs. Furthermore, notice that (i) and (iv) can be represented as

$$\begin{aligned}(i) &= \frac{1}{N}\sum_{i=1}^{N}\bar{q}_{i0}^R + \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M-1}\widehat{\tau}_{ij}^R \times a_{ij},\\ (iv) &= \frac{1}{N}\sum_{i=1}^{N}\bar{q}_{i0}^C + \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M-1}\widehat{\tau}_{ij}^C \times a_{ij},\end{aligned} \tag{7}$$

which depends on the estimated ITE. Equation (7) thus implies that accurate estimation of the treatment effects is critical to policy optimization. The optimization problem is a special multiple-choice knapsack problem whose dual problem can be solved in parallel. It usually converges in several iterations, and takes just a few minutes for tens of millions of samples.

Toy Example. The toy example in Section 1 provides a visual illustration of how non-crossing quantile regression improves the estimation of ITE. In Figure 1, the orange lines are the conditional mean outcome predictions based on DNet with number of quantiles $K = 5$. The blue lines are given by a multi-headed neural network model with MSE loss. It is apparent that the model with MSE loss is affected by "large" outcomes, which results in biased ITE estimation shown in Figure 1 (B). To the contrary, the ITE estimation curve based on DNet performs much better, and is parrallel to the oracle ITE curve.

5 VARIATIONS OF THE DNET ARCHITECTURE

In the previous section, we discussed the standard DNet architecture. In this section, we will introduce two modifications of DNet to accommodate some real-world tasks.

Mono-DNet. In many real-world applications, the expected outcome is often a monotonic function of the treatments applied. For example, in an e-commerce platform, providing more coupons will likely result in higher marketing costs and more customer orders. To borrow this information, we propose a monotonic DNet (Mono-DNet) by imposing the monotonic treatment constraint during the training phase. In this scenario, it is typically assumed that the outcome is an increasing function of the treatments. Similar to the monotonic constraints on quantile values, we apply the transformation function $ELU(x) + 1$ to the original outputs of the Value Layers $v(x,t;\theta_v)$, $t = 0,1,\ldots,M-1$. Let $\tilde{v}(x,t;\theta_v)$ denote the modified monotonic Value Layer for treatment t. We next define the output

$$\tilde{v}(x,t;\theta_v) = \sum_{t'=1}^{t}(ELU(v(x,t';\theta_v)) + 1) + v(x,0;\theta_v), \tag{8}$$

where ELU is not applied on $v(x,0;\theta_v)$ to allow $\tilde{v}(x,t;\theta_v)$ to take negative values. Such a transformation guarantees that the expected outcome is a non-decreasing of the treatments in equation 8.

ZI-DNet. In practice, we often encounter outcomes that are non-negative, heavy-tailed and have a significant fraction of zero values, resulting in a zero-inflated heavy-tailed distribution. Examples of such outcomes include customers' future lifetime value (LTV) and the gross merchandise value (GMV) in a marketing campaign, since many people may not use coupon as tall. Estimating ITEs can be extremely challenging in this case as the outcome is a mixture of discrete and continuous variables.

To address this challenge, we propose a modified version of DNet, called zero-inflated DNet, which involves an auxiliary task for predicting whether the outcome is zero. As shown in Figure 5, we add another head in the NCQ Layer to predict $Z(x,t;\theta_z) = P(Y = 0|X = x, T = t;\theta_z)$, i.e., the conditional probability of Y being 0. The final prediction can be reformulated as,

$$\tilde{R}(x,t;\theta_r) = \text{stop\_gradient}\{1 - Z(x,t;\theta_z)\} \otimes R(x,t;\theta_r).$$

We apply the stop-gradient 'operation in the zero-inflated NCQ Layer to ensure that the Prob Layer only learns from the classification loss.

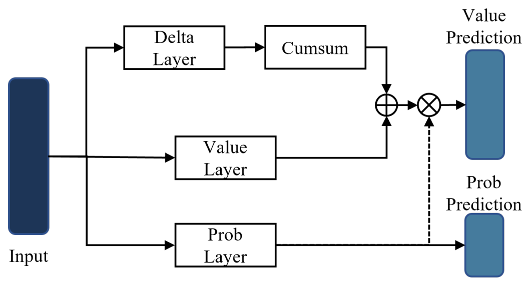


Figure 5: The illustration of Zero-Inflated Non-Crossing Quantile Layer, where we learn a separate probability about whether the outcome is zero.

Finally, we extend the loss function to include the classification loss of a zero-inflated term,

$$\begin{aligned}\mathcal{L}(x_i, t_i, y_i; \theta_r, \theta_\pi, \theta_z) &= \ell(\tilde{R}(x_i, t_i; \theta_r), y_i)\\ &+ \omega_1 \ell(\pi(x_i; \theta_\pi), t_i) + \omega_2 \ell(Z(x_i, t_i; \theta_z), \mathbf{1}(y_i = 0)),\end{aligned}$$

where $\mathbf{1}(Y_i = 0)$ is an indicator function. It is worth noting that the final prediction is $\tilde{R}(x, t; \theta_r)$ and the original output $R(x, t; \theta_r)$ can be viewed as the prediction conditional on the fact that the outcome is non-zero.

6 EVALUATION

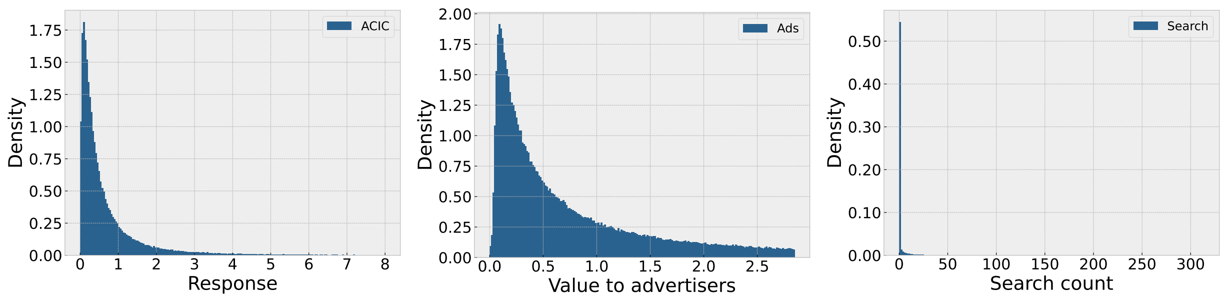
We evaluate the performance of the proposed DNet architecture through a thorough examination of synthetic and real-world datasets. The results show that the proposed architecture effectively captures the outcome distribution under each treatment and provides more accurate estimates of the individualized treatment effects. Furthermore, the model has been deployed in a production setting, resulting in significant improvements in key business metrics, validating its effectiveness in practical situations.

Datasets. Evaluating causal models can be challenging due to the absence of ground truth for causal effects in real-world datasets. To overcome this, researchers often use semi-synthetic datasets. In this study, we use the widely-used public datasets **IHDP** and **ACIC** as benchmarks to compare the performance of DNet with other baseline methods. Additionally, we collect several large-scale real-world datasets to further the usefulnesses of DNet and its variants.

- **IHDP:** The Infant Health and Development Program (IHDP) conducted randomized experiments to investigate the effect of home visits by specialists on infants' cognitive scores. Hill [14] generated thousands of datasets for causal effect estimation based on these experiments. We use 1000 realizations from the NPCI package, as in [32] and [33]. Each realization contains 747 instances. 63% of them are used for training whereas the remaining 27% and 10% are used for validation and testing, respectively.
- **ACIC:** We also use the data from the 2019 Atlantic Causal Inference Conference competition. The data is high-dimensional with continuous features and heterogeneous ITEs. Specifically, it contains 200,000 samples, each with 200 features and a binary treatment indicator. We split the data into 80% for training, 10% for validation and 10% for testing.
- **Real Datasets:** To evaluate the effectiveness of the proposed DNet architecture in real-world scenarios, we conduct online randomized controlled experiments and collect two datasets from a leading technology company to illustrate the usefulness of our proposal. This company operates one of the largest mobile platforms for production, aggregation, and distribution of various types of information. On the platform, users can earn rewards by watching video ads, searching relevant features, etc. We consider an ads dataset and a search dataset, use these datasets to evaluate the performance of DNet and its variants, and demonstrate their effectiveness in practical scenarios.
  - **Ads:** In the first dataset, users earn rewards by watching video ads. The dataset consists of millions of ad requests, with each request containing hundreds of pre-treatment features. The treatment corresponds to the reward we assigned to a give user. Here, we discretize all rewards into five levels, corresponding to $T = 0, 1, 2, 3$, and 4, where larger values lead to more rewards. For each request, a uniformly random reward is assigned and effective cost per mille (eCPM) is recorded as the value to the advertiser which serves as the outcome. The goal is to develop an optimal policy to maximize the total value to the advertisers, subject to a budget constraint.
  - **Search:** In the second dataset, we divide the daytime into three time windows. When a user launches the app for the first time during a time window, they may see a pop-up window that recommends them to experience a new search feature. However, some users find pop-ups annoying. The company is interested in developing an optimal 'pop-up' notification policy that implement this strategy for a subgroup of users at each time window to increase their search frequency. At each time window, the company can decide whether to show the pop-up window or not. This yields $2^3 = 8$ treatment options. The baseline treatment is to always show the pop-up window when the user launches the app for the first time regardless of what time window it is. Thus, we have 9 treatments in total. We randomly sample 9 equal-size groups of users and assign different treatments to different groups. This randomized controlled experiment lasts for two weeks. We collect hundreds of pre-treatment features for each user and are interested in the treatment effects on users' search counts in the two weeks. An optimal policy will maximize the total search counts without hurting user experience, which is reflected by the frequency of using the company's app.

Baseline Models. We compare the performance of our DNet architecture with several commonly used neural network baselines for ITE estimation.

- **TARNet** [32]: TARNet is a widely used deep learning causal model. It is a multi-headed neural network and each head corresponds to the expected outcome of each treatment. This shared bottom architecture effectively addresses the challenge of imbalance samples.
- **CFR** [32]: CFR is an extension of TARNet and addresses the challenge of imbalance samples by incorporating an additional loss function. This loss function forces the learned covariate distributions to be more similar across different treatments. We report the CFR performance with two different distribution distance measurement metrics, corresponding



(A) ACIC. (B) Ads. (C) Search.

Figure 6: Histograms of outcomes in ACIC/Ads/Search datasets. All distributions are heavy-tailed and the outcome in the Search dataset is also zero-inflated.

to Wasserstein and MMD, and denote the resulting two models as CFR_{Wass} and CFR_{MMD} respectively.

- **DragonNet** [33]: DragonNet is designed to learn the average treatment effect (ATE). However, it can also be used to estimate the conditional average treatment effect (CATE) as well. It extends TARNet by adding a regularization term (also known as the targeted regularization) to the loss function. This targeted regularization provides an end-to-end training procedure, yielding ATE estimators with desirable asymptotic properties and excellent finite-sample performance.

Evaluation Metrics. For the synthetic datasets including IHDP and ACIC, we use the Precision in Estimation of Heterogeneous Effect (PEHE) as the evaluation metric, since we have access to the ground truth of individual treatment effect $\tau_t(x)$. The PEHE is defined as,

$$\epsilon^{(t)}_{PEHE} = E[\hat{\tau}_t(x) - \tau_t(x)]^2.$$

For real-world datasets, we use the Area Under Uplift Curve (AUUC) as the evaluation metric due to that the ground truth is unknown to us. To plot the uplift curve for a given treatment t, we first sort all the samples according to their estimated ITEs $\hat{\tau}_t(X_i), i = 1, \cdots, N$ in decreasing order. Let D_n be the first n elements after sorting, and ATE_n be the average treatment effect computed using D_n. The uplift curve is drawn by connecting points $((n/N), (n/N) \times ATE_n)$ for $n = 0, 1, 2, \ldots, N$.

6.1 Hyperparameters Settings

For all the baselines, we use the same network structure and hyperparameter settings in their original papers such as network depth, layer dimension and learning rate. For the proposed DNet, the base net contains two layers with size of 200 and 100 for the normal dense layers with L-2 regularization in the action tower, which is the same as TARNET and DragonNet. For all experiments, we set the number of quantiles to 50 and we discuss how the number of quantiles affects the performance of DNet in Section 6.4.

	IDHP		ACIC	
	$\sqrt{\epsilon PEHE_{in}}$	$\sqrt{\epsilon PEHE_{out}}$	$\sqrt{\epsilon PEHE_{in}}$	$\sqrt{\epsilon PEHE_{out}}$
TARNET	0.88	0.95	4.35	4.69
CFR Wass	0.71	0.76	3.10	3.42
CFR MMD	0.73	0.77	3.08	3.38
DragonNet	0.68	0.77	4.04	4.35
DNet	**0.49±0.02**	**0.56±0.03**	**1.87± 0.18**	**2.34± 0.15**

Table 1: Performance summary of IHDP and ACIC datasets. *in* **stands for train and validation datasets while** *out* **stands for test set.**

6.2 Performance of DNet

In this section, we conduct extensive comparisons between our model with other baselines on IHDP, ACIC, and two real data sets.

IHDP. We report the mean rooted PEHE across 1000 realizations for all models in Table 1. The results show that the DNet architecture outperforms all baselines on both the training and testing sets. DragonNet works reasonably well on both the training and testing sets, with PEHEs equal to 0.68 and 0.77. The best baseline on the testing set is the CFR Wass with a PEHE of 0.76. However, our architecture can achieve more than 15% reduction in the rooted PEHE for both the training and testing sets. Additionally, we tried to add targeted regularization [33] to DNet, but it did not improve the overall performance on either the training or testing dataset. Similarly, we found that DragonNet achieves almost the same performance with or without the targeted regularization term in terms of rooted PEHE. We believe that adding targeted regularization may help the estimation of ATE, as shown in [33], but may not be helpful for the estimation of ITE. In conclusion, our results demonstrate that the proposed DNet architecture can outperform all baselines.

ACIC. On the ACIC dataset, we also use rooted PEHE as the evaluation metric. The results in Table 1 show significant improvement compared to the baselines. In contrast to the IHDP dataset, the CFR model outperforms DragonNet partially due to the non-random treatment assignment. Additionally, since the outcome of the ACIC dataset is heavy tailed as shown in Figure 6, DNet performs significantly better than other baselines.

Real Data. We also evaluate DNet on two real-world datasets based on AUUC. Unlike PEHE, a higher AUUC indicates better performance as it demonstrates the ability to more accurately rank samples

	IDHP		ACIC	
	$\sqrt{\epsilon PEHE_{in}}$	$\sqrt{\epsilon PEHE_{out}}$	$\sqrt{\epsilon PEHE_{in}}$	$\sqrt{\epsilon PEHE_{out}}$
TARNET	0.88	0.95	4.35	4.69
CFR Wass	0.71	0.76	3.10	3.42
CFR MMD	0.73	0.77	3.08	3.38
DragonNet	0.68	0.77	4.04	4.35
DNet	**0.49±0.02**	**0.56±0.03**	**1.87± 0.18**	**2.34± 0.15**

Table 2: Performance summary of IHDP and ACIC datasets. in **stands for train and validation datasets while** out **stands for test set.**

	T=1	T=2	T=3	T=4	Mean
DNet	0.53	0.58	0.68	0.58	0.59
Mono-DNet	**0.70**	**0.70**	**0.84**	**0.79**	**0.76**

Table 3: AUUCs of DNet and Monotonic-DNet models on value to advertiser in the ads dataset.

based on their estimated treatment effects. We report the average AUUC of all treatments in Table 2. It can be seen from the results that DNet outperforms all baselines on both tasks. In particular, DNet yields a larger improvement on complex tasks where all models have relatively lower AUUC. On the other hand, it only lead to a marginal improvement on the Search dataset which is relatively easy to learn for all baseline methods. It is also important to note is that we have multiple treatments in both the Ads and Search datasets, and the CFR model performs poorly under a multi-treatment setting. We believe this is due to the difficulty of aligning the feature embedding distributions across different treatments.

6.3 Performance of DNet's Variants

In this section, we demonstrate how Monotonic and Zero-inflated DNets can be applied to real-world problems.

Mono-DNet. For the ads dataset, we can reasonably assume that the probability of users watching the video ads is monotonous with respect to rewards, i.e. treatments. We apply monotonic DNet on this dataset, and present AUUC scores of treatments $T = 1, 2, 3, 4$ over baseline treatment $T = 0$ in Table 3. The results demonstrate that imposing the monotonicity constraint can significantly improve the AUUC scores for all four treatments.

ZI-DNet. In the search dataset, Figure 6 shows that around 75% of the search counts are zero. It is not surprising since users tend to search only when they need specific information. In Table 4, we compare the zero-inflated DNet to the original DNet on this dataset. We present the AUUC scores of non-baseline treatments over the baseline one with search counts as the outcome of interest, and it shows that adding an auxiliary task to predict whether the outcome is zero or not can significantly improve the AUUC score.

	T=1	T=2	T=3	T=4	
DNet	0.84	1.02	0.96	1.05	
ZI-DNet	**0.90**	**1.12**	**1.04**	**1.11**	
	T=5	T=6	T=7	T=8	Mean
DNet	1.33	2.13	0.96	0.98	1.16
ZI-DNet	**1.52**	**2.26**	**1.13**	**0.96**	**1.26**

Table 4: AUUCs of DNet and ZI-DNet models on search counts in the search dataset.

6.4 Ablation Study

In this section, we run extensive experiments to explore why DNet can outperform baseline models and how to train a better model.

Training stability. Visualize the training process in Figure 7. The first 80 epochs of rooted PEHE on validation set are plotted. The curves show instability at early stages with higher numbers of quantiles. The reason is that methods without extreme value theory usually break down for learning extreme quantiles, especially when the sample size is small. We leave the study of neural networks for extreme quantiles as future work. For practical applications, we recommend choosing the number of quantiles via cross validation based on AUUC scores.

Impact of number of quantiles. In DNet, non-crossing quantile regression captures outcome distributions across treatments. Figure 8 evaluates DNet's rooted PEHE with varying numbers of quantiles. Results show that 50 quantiles yield the best performance. The rooted PEHEs form a concave curve, indicating that accurate treatment effect estimation requires a specific number of quantiles. If set to 1, DNet performs worse as a model with MAE loss.

Results of 1000 IHDP tasks. See Figure 9 for the relative improvement of DNet over the strong baseline DragonNet on rooted PEHE for all 1,000 tasks. The red line shows where DNet underperforms DragonNet. Results show DNet outperforms DragonNet for 904 out of 1,000 tasks, confirming the DNet architecture is universally effective across multiple tasks, not just a selected few.

6.5 Online Deployment

All the DNet based models introduced in the real datasets have been successfully deployed in the production environment on a widely used mobile app with millions of daily active users. As introduced in Section 4.2, we solve the corresponding optimization problem based on the estimated treatment effects of these DNet models, and obtain an optimal policy in each application. The deployment decisions were based on two-week online A/B tests which showed that the new policy was able to bring statistically significant improvements in key business metrics. In the rewarded ads example, the optimal policy based on DNet architecture was able to achieve 2.8% significant increases in value to advertisers, while not negatively impacting user experience, as reflected by the average number of days and stay duration on the app. In the search example, ZI-DNet was able to improve the number of search counts by more than 13%. Additionally, the DNet model has been adopted by the monetization department to improve user experience, resulting in a significant 0.1% increase in user activity with minimal loss in value to advertisers.

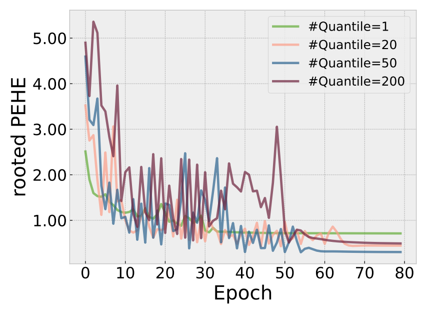


Figure 7: Validation PEHE versus training epochs.

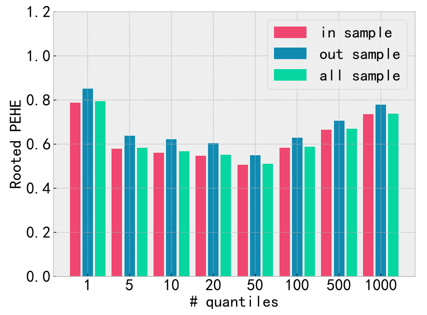


Figure 8: Rooted PEHE on IHDP dataset of models with different number of quantiles in NCQ Layer.

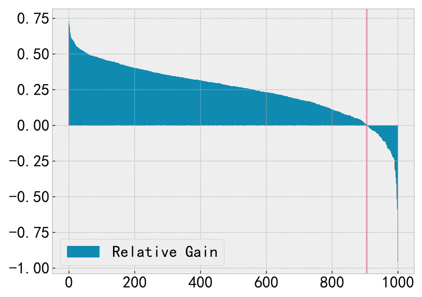


Figure 9: Relative differences of rooted PEHE on various tasks.

7 CONCLUSIONS

We presented DNet, a non-crossing neural network architecture for quantile ITE estimation with heavy-tailed outcomes. DNet captures the entire outcome distribution across treatments and is robust to outliers. We also develop two variants of DNet, and find that the lead to improved AUUC scores in real-world applications. We notice that the AUUC score measures the accuracy of the ranking instead of that of the estimated ITE. Future research is warranted to develop better evaluation methods on real datasets.

REFERENCES

[1] Susan Athey, Julie Tibshirani, and Stefan Wager. 2019. Generalized random forests. *The Annals of Statistics* 47, 2 (2019), 1148–1178.
[2] Marianne P Bitler, Jonah B Gelbach, and Hilary W Hoynes. 2017. Can variation in subgroups' average treatment effects explain treatment effect heterogeneity? Evidence from a social experiment. *Review of Economics and Statistics* 99, 4 (2017), 683–697.
[3] Howard D Bondell, Brian J Reich, and Huixia Wang. 2010. Noncrossing quantile regression curve estimation. *Biometrika* 97, 4 (2010), 825–838.
[4] Guillermo Briseño Sanchez, Maike Hohberg, Andreas Groll, and Thomas Kneib. 2020. Flexible instrumental variable distributional regression. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 183, 4 (2020), 1553–1574.
[5] Dong Chen, Ma Shujie, Zhu Liping, and Feng Xingdong. 2020. Estimation and inference for non-crossing multiple-index quantile regression. *SCIENTIA SINICA Mathematica* 51, 4 (2020), 631.
[6] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal* 21, 1 (2018), C1–C68.
[7] Victor Chernozhukov, Ivan Fernandez-Val, and Alfred Galichon. 2009. Improving point and interval estimators of monotone functions by rearrangement. *Biometrika* 96, 3 (2009), 559–575.
[8] Victor Chernozhukov, Iván Fernández-Val, and Blaise Melly. 2013. Inference on counterfactual distributions. *Econometrica* 81, 6 (2013), 2205–2268.
[9] Holger Dette and Stanislav Volgushev. 2008. Non-crossing non-parametric estimates of quantile curves. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70, 3 (2008), 609–627.
[10] Shuyang Du, James Lee, and Farzin Ghaffarizadeh. 2019. Improve User Retention with Causal Learning. In *The 2019 ACM SIGKDD Workshop on Causal Discovery*. PMLR, 34–49.
[11] Mehrdad Farajtabar, Andrew Lee, Yuanjian Feng, Vishal Gupta, Peter Dolan, Harish Chandran, and Martin Szummer. 2020. Balance regularized neural network models for causal effect estimation. *arXiv preprint arXiv:2011.11199* (2020).
[12] Peter Hall, Rodney CL Wolff, and Qiwei Yao. 1999. Methods for estimating a conditional distribution function. *Journal of the American Statistical association* 94, 445 (1999), 154–163.
[13] Xuming He. 1997. Quantile curves without crossing. *The American Statistician* 51, 2 (1997), 186–192.
[14] Jennifer L Hill. 2011. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics* 20, 1 (2011), 217–240.
[15] Maike Hohberg, Peter Pütz, and Thomas Kneib. 2020. Treatment effects beyond the mean using distributional regression: Methods and guidance. *PloS one* 15, 2 (2020), e0226514.
[16] Paul W Holland. 1986. Statistics and causal inference. *Journal of the American statistical Association* 81, 396 (1986), 945–960.
[17] Guido W Imbens and Donald B Rubin. 2010. Rubin causal model. *Microeconometrics* (2010), 229–241.
[18] Fredrik Johansson, Uri Shalit, and David Sontag. 2016. Learning representations for counterfactual inference. In *International conference on machine learning*. PMLR, 3020–3029.
[19] Roger Koenker, Victor Chernozhukov, Xuming He, and Limin Peng. 2017. Handbook of quantile regression. (2017).
[20] Roger Koenker, Pin Ng, and Stephen Portnoy. 1994. Quantile smoothing splines. *Biometrika* 81, 4 (1994), 673–680.
[21] Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. 2019. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences* 116, 10 (2019), 4156–4165.
[22] Yufeng Liu and Yichao Wu. 2009. Stepwise multiple quantile regression estimation using non-crossing constraints. *Statistics and its Interface* 2, 3 (2009), 299–310.
[23] Nicolai Meinshausen and Greg Ridgeway. 2006. Quantile regression forests. *Journal of Machine Learning Research* 7, 6 (2006).
[24] Sang Jun Moon, Jong-June Jeon, Jason Sang Hun Lee, and Yongdai Kim. 2021. Learning multiple quantiles with neural networks. *Journal of Computational and Graphical Statistics* 30, 4 (2021), 1238–1248.
[25] Belbahri Mouloud, Gandouet Olivier, and Kazma Ghaith. 2020. Adapting Neural Networks for Uplift Models. *arXiv preprint arXiv:2011.00041* (2020).
[26] Krikamol Muandet, Motonobu Kanagawa, Sorawit Saengkyongam, and Sanparith Marukatat. 2021. Counterfactual Mean Embeddings. *J. Mach. Learn. Res.* 22 (2021), 162–1.
[27] Xinkun Nie and Stefan Wager. 2021. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika* 108, 2 (2021), 299–319.
[28] Miruna Oprescu, Vasilis Syrgkanis, and Zhiwei Steven Wu. 2019. Orthogonal random forest for causal inference. In *International Conference on Machine Learning*. PMLR, 4932–4941.
[29] Junhyung Park, Uri Shalit, Bernhard Schölkopf, and Krikamol Muandet. 2021. Conditional distributional treatment effect with kernel conditional mean embeddings and U-statistic regression. In *International Conference on Machine Learning*. PMLR, 8401–8412.
[30] Robert A Rigby and D Mikis Stasinopoulos. 2005. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 54, 3 (2005), 507–554.
[31] Donald B Rubin. 2005. Causal inference using potential outcomes: Design, modeling, decisions. *J. Amer. Statist. Assoc.* 100, 469 (2005), 322–331.
[32] Uri Shalit, Fredrik D Johansson, and David Sontag. 2017. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*. PMLR, 3076–3085.
[33] Claudia Shi, David Blei, and Victor Veitch. 2019. Adapting Neural Networks for the Estimation of Treatment Effects. *Advances in Neural Information Processing Systems* 32 (2019), 2507–2517.
[34] Stefan Wager and Susan Athey. 2018. Estimation and inference of heterogeneous treatment effects using random forests. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1228–1242.
[35] Huixia Judy Wang and Lan Wang. 2009. Locally weighted censored quantile regression. *J. Amer. Statist. Assoc.* 104, 487 (2009), 1117–1128.

[36] Lan Wang, Yu Zhou, Rui Song, and Ben Sherwood. 2018. Quantile-optimal treatment regimes. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1243–1254.

[37] Yang Xu, Chengchun Shi, Shikai Luo, Lan Wang, and Rui Song. 2022. Quantile Off-Policy Evaluation via Deep Conditional Generative Learning. *arXiv preprint arXiv:2212.14466* (2022).

[38] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. 2018. GANITE: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*.

[39] Kui Zhao, Junhao Hua, Ling Yan, Qi Zhang, Huan Xu, and Cheng Yang. 2019. A Unified Framework for Marketing Budget Allocation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1820–1830.

[40] Zhenyu Zhao and Totte Harinen. 2019. Uplift modeling for multiple treatments with cost optimization. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 422–431.

[41] Tianhui Zhou and David Carlson. 2021. Estimating Potential Outcome Distributions with Collaborating Causal Networks. *arXiv preprint arXiv:2110.01664* (2021).

[42] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*. Springer, 566–576.

[43] Will Y Zou, Shuyang Du, James Lee, and Jan Pedersen. 2020. Heterogeneous causal learning for effectiveness optimization in user marketing. *arXiv preprint arXiv:2004.09702* (2020).