NeurIPS Reproducibility Challenge Report: Adapting Neural Networks for the Estimation of Treatment Effects

Xiaoting Wang, Hao Li and Bozhong Lu

McGill University
`xiaoting.wang@mail.mcgill.ca`

Abstract

Causal inference is a fundamental problem in many fields. In this report, we consider the problem of causal effects estimation based on a semi-synthetic dataset. To be specific, this estimation task was completed by using different neural network architectures including Dragonnet, TARNET and Nednet. The performance of these network architectures is compared. Specially, we focus on the effect of Dragonnet and Dragonnet with target regularization models on treatment estimation, which are two adaptive approaches implemented by original authors. To study the robustness of Dragonnet architecture, we explored different hyperparameters. It can be found that most of the hyperparameters we have tried have a slight impact on the Dragonnet architecture. Also, we tried to modify the number of hidden layers in Dragonnet for the outcome models. We found that a minor improvement has been shown by adding more hidden layers. We further compared the effect of two different plug-in treatment effect estimators. The results have shown that Dragonnet has decent performance even with a conditional-outcome-only estimator.

1 Introduction

Applying machine learning methods to problems of causal inference has gained an intense interest in various fields like healthcare, economics and education. With the availability of large datasets in these fields, an increasing interest has been gained in developing methods for learning causal effects from observational data [1, 2]. In this project, we focus on the problem of estimation of causal effects based on observational data. Specially, we work under the assumption that all the factors determining which actions were taken are observed ("no-hidden confounding assumption").

In this project, we aimed to estimate the effect of treatment W on an outcome Y through covariate X adjustment or propensity score re-weighting [3]. For example, estimate whether the patients recover (outcome) after received drugs (treatment) adjusting on their illness severity(covariate). To better describe our task, we will denote the conditional outcome as $Q(w, x) = \mathbb{E}[Y|W = w, X = x]$ and propensity score as $ps(x) = \Pr(W = 1|X = x)$, where w and x are the treatment and covariates from the observational data, respectively. Followed by the work of original authors, we consider the average effect of binary treatment (see section 4 for detail setup).

Our estimation task contains two parts: fit the model to obtain $Q(w, x)$ and $ps(x)$ by using neural networks; plug the fitted model into a downstream estimator. For this project, most of our work is identical to work by the original authors. The detail information of what we have done in this project can be listed as follows.

Task 1. Applied different network architectures on a semi-synthetic dataset (IHDP) [4] to estimate the treatment effects.

Task 2. Further tried a regularization procedure called target regularization to fit a model with a suitable downstream estimator.

Task 3. Modified the hyperparameters to explore how the Dragonnet architecture and target regularization procedure perform on the treatment estimation. Dragonnet architecture and target regularization procedure are two main contributions of the paper [5] we tried to reproduce.

Task 4. Attempted to increase the number of hidden layers of Dragonnet model to study the performance of treatment estimation for a deeper neural network.

Task 5. Compared different downstream estimators and reported the results.

The rest of this report is organized as follows. Section 2 summarizes the related work of causal inference and estimation theory. Section 3 gives a brief introduction of dataset and setup. Section 4 introduces the setup of average treatment effect and proposed approaches. Section 5 shows the experiment and results. Section 6 states the discussion and conclusion. Section 7 presents the statement of contribution.

2 Related work

Adapting machine learning methods for causal effect inference has gained much interest recently. Beck et al. earlier exemplified that neural nets can help detect treatment effects when some data from randomized experiments is available [6]. Later, Jason et al. applied a deep neural net to solve for causal effects in the presence of instrumental variables [7]. Much recently, these methods have been focused on learning a covariate representation that has a balanced distribution across treatment and outcome. For example, Fredrik et al. developed a new algorithmic framework for counterfactual inference. In their work, a modification of the standard feed-forward architecture with fully connected layers (see Fig. 2) was proposed.

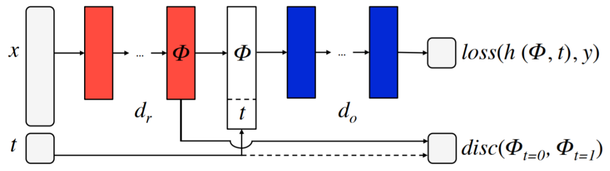

Figure 1: Neural network architecture. The first d_r hidden layers are used to learn a representation $\Phi(x)$ of the input x. The output of the d_r layer is used to calculate the discrepancy $disc(\Phi_{t=0}, \Phi_{t=1})$. The d_o layers following the first d_r layers take as additional input the treatment assignment t_i and generate a prediction $h([\Phi(x_i), t_i])$ of the outcome [2].

Based on Fredrik's work, Shalit et al. evaluated the proposed framework CFRNet and its variant without balancing regularization (TARNET) in causal inference, where TARNET is used as a baseline in [8]. Also, there are several studies combining deep generative models [10] and GANs [9] with standard causal identification results. Ahmed et al. proposed more sophisticated estimators based on pre-trained propensity score [11].

Recently, Claudia et al. proposed a new Dragonnet architecture, which serves as a complementary of other approaches [5] (paper we tried to reproduce). To guarantee desirable asymptotic properties of the downstream estimator, they further go to the target regularization procedure which was inspired by targeted minimum loss estimation (TMLE) [12].

There is another recent paper based on the Dragonnet-type model in causal inference combining the black-box embedding methods [13]. However, they make an extension of robust estimation results to (non-iid) network data, which differs from the assumption in Claudia et al.'s work (iid network data).

Based on the above literature review, in addition to replicating the work of authors in the original paper, we tried to implement some of these techniques for our project.

3 Dataset and setup

The observation data used in the paper is based on two semi-synthetic datasets, IHDP and ACIC 2018. Both of them are the benchmark for causal effects.

IHDP Dataset. The Infant Health and Development Program (IHDP) provided professional child care and home visits for targeted infants [15]. IHDP dataset contains the causal effects of the randomized experiment on future cognitive test scores for us to explore. Claudia et al. provided 1000 realizations where each realization contains 747 observation units[5]. In addition, they provided a portion of the dataset (50 out of 1000) in their code for us to test their models. Due to time limits, we use this small dataset to examine the importance and effects of different parameters of the models. See Appendix A Remark 2 for the detail of ACIC dataset and GPU time of this experiment.

Data Partitioning. To run the models with the IHDP dataset, following [5], the data is split into train, validation, and test set with proportion 63/27/10.[1] Under this procedure, we can obtain the in sample and out of sample estimation errors (see Table 1). For setup of calculating estimation error for all dataset, see Appendix A Remark 3.

4 Background and proposed approaches

4.1 Average treatment effect

In this section, some notation and necessary ideas about the statistical estimation of causal effects are recalled. We first give the following assumption:

Assumption 1 *It is assumed that the data is generated independently and identically, where* $(Y_I, W_i, X_i) \overset{\text{iid}}{\sim} P$*.*

The average treatment effect (ATE) Φ of a binary outcome is defined as:

$$\Phi = \mathbb{E}[Y|\text{do}(W=1)] - \mathbb{E}[Y|\text{do}(W=0)] \tag{1}$$

where W and Y are defined in section 1. Based on Pearl's do notation, the effect of interest is causal: what will be the expected outcome if we intervene by assigning the treatment to a given unit? If X_i contains all common confounders of Y_i and W_i then the causal effect is identifiable as a parameter of the observational distribution [13]:

$$\Phi = \mathbb{E}[\mathbb{E}[Y|X, W=1] - \mathbb{E}[Y|X, W=0]] \tag{2}$$

Considering estimation of Φ from a finite sample generated from P, a natural estimator based on (2) is provided in [5].

$$\hat{\Phi}^{\mathrm{Q}} = \frac{1}{n}\sum_i [\hat{Q}(1, x_i) - \hat{Q}(0, x_i)] \tag{3}$$

where $\hat{Q}$ denotes the estimate of the conditional outcome $Q(w, x)$ (defined in section 1). Estimators based on this method are called conditional-outcome-only estimator.But $\hat{Q}$ is not the only possible choice of estimator. In principle, it is possible to do better by incorporating estimates of propensity score $ps(x)$ (see section 5.3 for detail).

4.2 Dragonnet

In order to obtain a better estimation of Φ, models which can fit both conditional outcome $Q(w, x)$ and propensity score $ps(x)$ should be implemented. To this end, we introduce a state-of-the-art neural network architecture: Dragonnet. This type of neural network is based on the sufficiency of the propensity score. It is inspired by the theorem from a related prior work on propensity score [14] (see Theorem 1 and 2).

In terms of covariates X, some components in X are only useful for the prediction of outcomes, but not the treatment. If these irrelevant covariates are used to estimate the causal effects, they will act

[1]By contacting with the original authors, we find that there is a typo in the original paper [5]. They use train/validation/test with proportion 63/27/10 instead of 10/27/63.

as noise. In short, in order to model the estimate of the conditional outcome $\hat{Q}$, one should predict Y with information from X that is only relevant to the treatment W. Its main idea is to extract the covariates that are relevant to the treatment as features. These features are thus relevant to the propensity score $\hat{ps}$. The model can then make predictions conditioning on the features. However, this approach is difficult to use especially because of its architecture. Moreover, errors generated from predicting $\hat{ps}$ can result in extra errors on predicting $\hat{Q}$.

An improved network named Dragonnet is thus implemented to predict both conditional outcome and propensity score at the same time. See Appendix C for detail architecture of Dragonnet.

According to Claudia et al, the training process of the Dragonnet depends on an objective function $\hat{R}(\theta; X)$ where θ represent the parameters of Dragonnet [5].

$$\hat{R}(\theta; X) = \frac{1}{n}\sum_i \left[Q^{nn}(w_i, x_i; \theta) - y_i)^2 + \alpha \text{CrossEntropy}(ps^{nn}(x_i; \theta), w_i)\right] \tag{4}$$

In Equation (4), $\hat{Q} = Q^{nn}(\cdot, \cdot; \theta)$ and $\hat{ps} = ps^{nn}(\cdot, \cdot, \hat{\theta})$ refer to fitted models. $\alpha \in \mathbb{R}_+$ is a hyperparameter controls the loss function. It indicates that training best parameters $\hat{\theta}$ is equivalent to minimizing $\hat{R}(\theta; X)$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \hat{R}(\theta; X) \tag{5}$$

Thanks to the structure of Dragonnet, all useful information may be kept while training. As Claudia et al mentioned in their paper, Dragonnet is proved to ensure a trade-off prediction quality and propensity score [5]. With a better propensity score, Dragonnet could generate a better estimation of the average treatment affect (ATE).

4.3 Target regularization

In this section, we will introduce the target regularization proposed in [5], which is a modification to the objective function. To this end, an extra model parameter ϵ and a regularization term $\gamma(y, w, x; \theta, \epsilon)$ are first introduced with the following form.

$$\begin{aligned}\widetilde{Q}(w_i, x_i; \theta, \epsilon) &= Q^{nn}(w_i, x_i; \theta) + \epsilon\left[\frac{w_i}{ps^{nn}(x_i; \theta)} - \frac{1 - w_i}{1 - ps^{nn}(x_i; \theta)}\right] \\ \gamma(y, w, x; \theta, \epsilon) &= \left(y_i - \widetilde{Q}(w_i, x_i; \theta, \epsilon)\right)^2\end{aligned} \tag{6}$$

By using the extra introduced term, the new estimator $\hat{\Phi}^{\text{treg}}$ can be designed as:

$$\begin{aligned}\Phi^{\text{treg}} &= \frac{1}{n}\sum_i [\hat{Q}^{treg}(1, x_i) - \hat{Q}^{treg}(0, x_i)] \\ Q^{\text{treg}} &= \widetilde{Q}(\cdot, \cdot; \hat{\theta}, \hat{\epsilon})\end{aligned} \tag{7}$$

where $\hat{\theta}$ and $\hat{\epsilon}$) are obtained by minimizing the modified objective:

$$\hat{\theta}, \hat{\epsilon} = \underset{\theta, \epsilon}{\operatorname{argmin}} \left[hatR(\theta; X) + \beta \frac{1}{n}\sum_i \gamma(y, w, x; \theta, \epsilon)\right] \tag{8}$$

The term Q^{treg}, ps^{treg} and Φ^{treg} can be proven to satisfy the non-parametric estimating equation (see [5] for details). The estimator ϕ^{treg} has shown stable finite-sample performance and strong asymptotic guarantees which has been shown in [5]. Also, we will provide comparison of this estimator with other types in section 5.

4.4 Other approaches

Besides the approaches mentioned in section 4.2 and 4.3, we also implemented TARNET and Nednet model, where TARNET is the main baseline in original paper [5].

TARNET. Treatment-Agnostic Representation Network (TARNET) is a two-headed architecture which only predicts the outcome [8]. When the propensity score of Dragonnet is moved, these two models are equivalent. The comparisons of this model and Dragonnet are presented in section 5.1.

Nednet. Nednet is a multi-stage approach, with essentially same architecture as Dragonnet. For training the model, only a pure treatment prediction objective is used at first. Then, the fianl layer (treatment prediction head) is cut off and substituted by an outcome-prediction neural network. Based on this, the representation layers will be frozen and the output of the outcome-prediction neural network will be a pure outcome prediction [5]. This method is compared with the other two methods in section 5.6.

5 Experiments

Evaluation of causal inference approaches is always challenging due to the lack of ground-truth for the causal effects. Generally, evaluation of causal inference methods is based on synthetic or semi-synthetic datasets [10]. Following the work of authors in original paper[5], we assessed the methods empirically using an existing benchmark dataset IHDP [2]. Following provides some basic settings of our experiments, all of which are based on the original paper.

Baseline settings: the main baseline is an implication of TARNET architecture; the target regularization baseline is set by using TARNET as the outcome model and logistic regression as the propensity score model.

Dragonnet and target regularization: by default, the hyperparameters α in (4) and β in (8) are set to be 1, except for the exploration of these two parameters in section 5.2 and 5.3.

For all models settings: we set the hidden layer size to 200 for the shared representation layers, and 100 for condition outcome layers. The batch_size is set as 64. The stochastic gradient decent (SGD) with momentum is used as our default optimizer, except for studying the performance of different optimizer (section 5.4).

For estimators and metrics: we will present the mean absolute difference between the estimate and sample ATE, which can be calculated by $\Delta = |\hat{\Phi} - (\frac{1}{n}\sum_i Q(1, x_i) - Q(0, x_i))|$. For $\hat{\Phi}$, we use naive $\hat{\Phi}$ as our estimator, except for the models with target regularization ($\hat{\Phi}^{\mathrm{treg}}$) and the target minimum loss estimation ($\hat{\Phi}^{\mathrm{tmle}}$).

The results of experiments we will present are listed below.

Results 1. Replicating the work of authors in original paper (see section 5.1).

Results 2. Exploring how Dragonnet performs by modifying the hyperparameters (see section 5.2).

Results 3. Effect of different β of target regularization on causal effects estimation (see section 5.3).

Results 4. Performance of different optimizer with Dragonnet (see section 5.4).

Results 5. Modification of the Dragonnet architecture (see section 5.5).

Results 6. Comparison of different estimators (see section 5.6).

5.1 Evaluation of Treatment Estimation

In this section, we report the IHDP simulation results of original paper to evaluate different treatment estimation methods including the results of the original paper [5]. The results can be seen at Table 1. As we can see, Dragonnet has a comparable results compared with TARNET. However, it is difficult to draw conclusions about the methods due to the small sample size and limited simulation settings of IHDP. From Table 1, we can find fitting model with all data and assessing the estimate can achieve a better performance than data splitting. The above ideas are consistent with the original paper [5]. There is a slight difference between our results and the original results, we will discuss the possible reasons in section 6.

5.2 Hyperparameters of Dragonnet

For Dragonnet model, we focus on some hyperparameters, which are given as follows: $\boldsymbol{\alpha}$ in equation (4), **min_lr**, **cooldown**, **factor** in ReduceLROnPlateau and **momentum**. Also, the original authors

[2]Data provided by the original authors: `github.com/claudiashi57/dragonnet/dat`

used **EarlyStopping** when fitting the model. We have tried to train the model without **EarlyStopping**. Note that for each experiment, we did not change other hyperparameters except for the one we tried to explore. The results of each experiments are given as follows. [3]

Tuning $\boldsymbol{\alpha}$. We first tried to explore the effect of different $\boldsymbol{\alpha}$ in Dragonnet (see Table 2). $\boldsymbol{\alpha}$ is a hyperparameter weighting the loss components. From Table 2, we can find that $\alpha = 1$ yields a better result over others we have tried.

Tuning min_lr. **min_lr** is the lower bound of learning rate. We tried different **min_lr** to see how it affects the treatment estimation. As can be seen from Table 3, slight change of **min_lr** has minor affect on the estimation.

Table 1: Mean absolute difference between the estimate and sample ATE using different models. Δ_{in} is the mean absolute error (and standard error) when the estimators are computed with the training and validation data. Δ_{out} is the mean absolute error (and standard error) when using heldout data, and Δ_{all} is for all data.

Method	Δ_{in}	Δ_{out}	Δ_{all}
baseline (TARNET) [5]	0.16 ± 0.01	0.21 ± 0.01	0.13 ± 0.01
baseline + t-reg [5]	0.15 ± 0.01	0.20 ± 0.01	0.12 ± 0.01
Dragonnet [5]	0.14 ± 0.01	0.21 ± 0.01	0.12 ± 0.01
Dragonnet + t-reg[5]	0.14 ± 0.01	0.20 ± 0.01	0.11 ± 0.01
baseline (TARNET)	0.15 ± 0.01	0.21 ± 0.01	0.12 ± 0.01
baseline + t-reg	0.15 ± 0.01	0.20 ± 0.01	0.12 ± 0.01
Dragonnet	0.15 ± 0.01	0.19 ± 0.01	0.12 ± 0.01
Dragonnet + t-reg	0.15 ± 0.01	0.21 ± 0.01	0.12 ± 0.01

Table 2: Dragonnet with different α

Method	α	Δ_{all}
Dragonnet	0.5	0.15 ± 0.01
Dragonnet	2	0.17 ± 0.01
Dragonnet	10	0.14 ± 0.01

Table 3: Dragonnet with different min_lr

Method	min_lr	Δ_{all}
Dragonnet	1e-5	0.11 ± 0.01
Dragonnet	1e-6	0.11 ± 0.01
Dragonnet	1e-7	0.12 ± 0.01

Tuning cooldown. **cooldown** is number of epochs to wait before resuming normal operation after lr has been reduced. We tried to increase the value of **cooldown** to see how it affects the estimation. The results can be seen on Table 4. For the values we have tried, cooldown value has minor affect on the estimation.

Tuning momentum. The original authors used the optimizer SGD with momentum 0.9. We tried to change the momentum value to see the performance of our estimator. The results can be seen on Table 5.

Table 4: Dragonnet with different cooldown values.

Method	cooldown	Δ_{all}
Dragonnet	1	0.11 ± 0.01
Dragonnet	3	0.12 ± 0.01
Dragonnet	5	0.12 ± 0.01
Dragonnet	10	0.12 ± 0.01

Table 5: Dragonnet using SGD optimizer with different momentum.

Method	momentum	Δ_{all}
Dragonnet	0.8	0.13 ± 0.01
Dragonnet	0.85	0.13 ± 0.01
Dragonnet	0.90	0.12 ± 0.01
Dragonnet	0.95	0.12 ± 0.01

[3]For the space limit of our report, we only report the results of Dragonnet. Some results of TARNET with different hyperparameters can be seen on `github.com/lhcaleo/NeurIPS_Challenge/blob/master/README.md`.

Tuning ReduceLROnPlateau factor. The learning rate will be reduced by this factor value (e.g. new_lr = lr $\times$ factor). The original factor is 0.5. Detail results are shown in Table 6.

With or without EarlyStopping. Table 7 shows the results obtained by training with and without EarlyStopping. It can be seen that when fitting the model with **EarlyStopping**, the performance of the estimator is much better than the one trained without **EarlyStopping**.

Table 6: ReduceLROnPlateau with different factor, for factor = 0.5, $\Delta_{all} = 0.12 \pm 0.01$.

Method	factor	Δ_{all}
Dragonnet	0.4	0.11 ± 0.01
Dragonnet	0.6	0.12 ± 0.01

Table 7: Fitting Dragonnet model with and without EarlyStopping.

Method	EarlyStopping	Δ_{all}
Dragonnet	with	0.12 ± 0.01
Dragonnet	without	0.24 ± 0.01

5.3 Hyperparameter of Target Regularization

In this section, we report the mean and standard errors for the metrics obtained by different β, which is the weighting of For different β, different performance of the estimator will be obtained. An very serious case is that when we choose a too large β, the result will go to nan. But for just a slight change of β, the regularization term does not affect the estimation noticeably.

Table 8: Target regularization with different β. Note that for $\beta = 1$, $\Delta_{all} = 0.12 \pm 0.01$.

Method	β	Δ_{all}
Dragonnet + t-reg	0.5	0.13 ± 0.01
Dragonnet + t-reg	10	0.18 ± 0.01
Dragonnet + t-reg	20	0.18 ± 0.01

Table 9: Dragonnet with different optimizer. "SGD + M" denotes with momentum.

Method	Optimizer	Δ_{all}
Dragonnet+t-reg	SGD + M	0.12 ± 0.01
Dragonnet+t-reg	Adam	0.12 ± 0.01
Dragonnet+t-reg	Adammax	0.10 ± 0.01

5.4 Performance of Different Optimizer with Dragonnet

Table 9 shows the performance of Dragonnet + target regularization with SGD, Adam and Adammax. Although the original authors in [5] used SGD with momentum in their work, we find that Adammax provides us with a better results for Dragonnet with target regularization.

5.5 Modified Dragonnet

In this section, we modified the hidden layer in Dragonnet for the outcome models $Q(1, \cdot)$ and $Q(0, \cdot)$. Note that the original authors used 2-hidden layer networks for each of the outcome models. From Table 10, we can find that slight affect has been made on the the estimates by adding more hidden layers. Note that due to randomness, the estimates will change even with same parameter.

Table 10: Different hidden-layer Dragonnet for outcome models.

Method	Hidden-layer	Δ_{all}
Dragonnet	3	0.11 ± 0.01
Dragonnet	4	0.11 ± 0.01
Dragonnet	5	0.13 ± 0.01

Table 11: Comparison of different estimators. Table entries are Δ_{all}.

Method	Φ^{Q}	Φ^{TMLE}
TARNET	0.12 ± 0.01	0.12 ± 0.01
Dragonnet	0.12 ± 0.01	0.12 ± 0.01
Nednet	0.18 ± 0.01	0.13 ± 0.01

5.6 Different Choice of Estimator

We report comparisons of downstream estimators in Table 11. We consider 2 options for the plug-in treatment effect estimator. One is the conditional-outcome-only estimator and the other is targeted minimum loss based estimator (TMLE) [12]. The original authors provided the comparison of Dragonnet and Nednet using these two estimators. Additionally, we also present the results of

TARNET with conditional-outcome-only estimator and TMLE. It can seen from Table 11, Dragonnet has a better performance than Nednet under the conditional-outcome-only estimator. Performance of Nednet has improved under TMLE.

Remark 1 *From Table 11, we can see a slight difference has been shown for Nednet and Dragonnet under TMLE. The results shown in original authors' work for IHDP experiments also provide the same idea. However, they have demonstrated that Dragonnet shows a better performance even under TMLE by the ACIC experiments* [5].[4]

6 Discussion and Conclusion

To conclude, computing estimates from the model trained by reusing same data works better than training model with data splitting. Also, Dragonnet actually has strong robust and better performance compared with Nednet from our experiments. As for TARNET and Dragonnet, it is difficult to draw conclusion about them due to the small sample size and limited simulation setting of IHDP.

For the simulation results shown in Table 1, there is a slight difference between our results with the original ones. By contacting with the original authors, we summarized a few possible reasons here:

Cause 1 We implemented 50 replications in the experiments instead of 1000. (most likely reason)

Cause 2 The difference may derived from "randomness", which may caused by:

- Randomness in the train-test-split function:
  - Each call to the train-test-split function would result in a different split. Some entries are easier to predict than others. Therefore this could potentially result in differences in error.
  - The authors do not sure if they have specially set the random_state seed when they ran the IHDP experiment.
  - When using only 10% of the data, the randomness would dominate.
- Randomness in the seed:
  - Random initialization of TensorFlow, Keras, and Numpy would give us a different result.
  - The authors may use certain seeds for the experiment of in sample and out of sample testing.

During the experiments, we found that the Dragonnet sometimes has worse performance than the baseline. By checking Table 3 in the original paper, we know that Dragonnet and target regularization help half of the time. The key idea is just that in typical bad case, the degradation is small. While in the typical good case, the help is large. This has been confirmed with the original authors.

In the future, we plan to learn how other people implement neural networks that have similar architecture as Dragonnet. Furthermore, it is still an important open question: how do we find the best parameter settings for Dragonnet? Higher accuracy might be obtained by a thorough process of parameter tuning. Also, we intend to investigate the unsolved problem why the data splitting technique abnormally lowers the performance of these experiments. We believe the research on causal effects will become more successful by combining the current Dragonnet with other complicated models in the same field of study.

7 Statement of Contributions

Xiaoting Wang: model setup, report editing, parameter tuning, and communicating with authors.
Hao Li: parameter tuning, report editing, and communicating with authors.
Bozhong Lu: parameter tuning and report editing.

[4] See Table 4 in [5] for the comparison of Dragonnet and Nednet

Acknowledgements

We would like to thank the authors especially Claudia Shi and Victor Veitch for their assistance with the datasets, the code implementation and our confusions in the paper. We gratefully thank them for their consistent and extremely helpful responses for our confusions.

Appendices

A Additional remark for dataset

ACIC 2018. The 2018 Atlantic Causal Inference Conference (ACIC) competition dataset is another causal inference tool used. It is another collection of semi-synthetic datasets collected from the Linked Births and Infant Deaths Database (LBIDD) [16] which is based on a real-world medical measurements. It is a comparatively large datasets since it involves 63 unique data generating process settings. Claudia mentioned that they randomly chose 3 datasets of size either 5k or 10k for each data generating setting.[5] When we contacted with Claudia, she provided us with the exact full ACIC data with the size of around 20GB. Since the approximately training time for this dataset is around 17 days GPU time, we directly use the trained data she provided us for reproducing their experiment results for simplicity.

Remark 2 *Two pre-established causal benchmark datasets were used in the original paper: IHDP and ACIC 2018. By communicating with the original authors, we found that the entire process for ACIC experiments took about 101 (dataset) $\times$ 25 (runs each)$\times$ 5(methods) $\times$ 2(mins each) = 17.5 days of GPU time. Considering the time limits, we only reproduce the IHDP experiments in the original paper.*

Remark 3 *The original authors set the "test_size" to be 0 when calculating the estimation error for all dataset. However, a compile error will occur when we use the same code on Colab.* [5] *To avoid this error, we set the variable "test_size" in the code as 0.0001.* [6]

B Theorems

Theorem 1 *If treatment assignment is strongly ignorable given x, then it is strongly ignorable given any balancing score $b(x)$.*

Claudia et al. [5] stated the following theorem of sufficiency of propensity score according to Theorem 1.

Theorem 2 *If the average treatment effect (ATE) Φ is identifiable from observational data by adjusting for X where $\Phi = \mathbb{E}[\mathbb{E}[Y|X, W = 1] - \mathbb{E}[Y|X, W = 0]]$, it then suffices to adjust for the propensity score:*

$$\Phi = \mathbb{E}[\mathbb{E}[Y|ps(X), W = 1] - \mathbb{E}[Y|ps(x), W = 0]] \tag{9}$$

C Architecture of Dragonnet

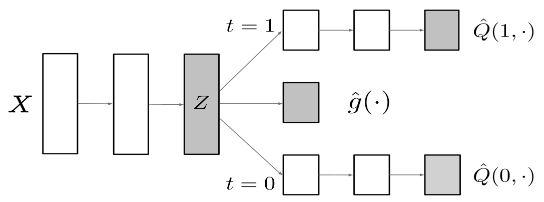

Figure 2: Architecture of Dragonnet [5].

As illustrated in Figure 2, taking covariates X as the network input, there is a special Z-layer $Z(X) \in \mathbb{R}^p$ which is shared by three output models respectively $\hat{Q}(0, \cdot)$:$\mathbb{R}^P \to \mathbb{R}$, $\hat{Q}(1, \cdot)$:$\mathbb{R}^P \to \mathbb{R}$, and $\hat{ps}$. The propensity score model $\hat{ps}$ is the product of a sigmoid function given a linear mapping from Z-layer. Similarly, the conditional outcome models are finally generated by two hidden layers connected with the Z-layer.

[5]This is caused by different Tensorflow version we used. They use Tensorflow 1.13, which we did not find on Colab.

[6]By communicating with the original authors, it is agreed that set "test_size" to be 0.0001 is reasonable.

References

[1] Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., & Blei, D. M. (2016). Edward: A library for probabilistic modeling, inference, and criticism. arXiv preprint arXiv:1610.09787.

[2] Johansson, F., Shalit, U., & Sontag, D. (2016, June). Learning representations for counterfactual inference. In International conference on machine learning (pp. 3020-3029).

[3] Morgan, S. L., & Winship, C. (2015). Counterfactuals and causal inference. Cambridge University Press.

[4] Hill, J. L. (2011). Bayesian nonparametric modeling for causal inference. Journal of Computational and Graphical Statistics, 20(1), 217-240.

[5] Shi, C., Blei, D. M., & Veitch, V. (2019). Adapting Neural Networks for the Estimation of Treatment Effects. In: *arXiv preprint arXiv:1906.02120.*

[6] Beck, N., King, G., & Zeng, L. (2000). Improving quantitative studies of international conflict: A conjecture. American Political Science Review, 94(1), 21-35.

[7] Hartford, J., Lewis, G., Leyton-Brown, K., & Taddy, M. (2016). Counterfactual prediction with deep instrumental variables networks. arXiv preprint arXiv:1612.09596.

[8] Shalit, U., Johansson, F. D., & Sontag, D. (2016). Estimating individual treatment effect generalization bounds and algorithms. arXiv preprint arXiv:1606.03976.

[9] Yoon, J., Jordon, J., & van der Schaar, M. (2018). GANITE: Estimation of individualized treatment effects using generative adversarial nets.

[10] Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R., & Welling, M. (2017). Causal effect inference with deep latent-variable models. In Advances in Neural Information Processing Systems (pp. 6446-6456).

[11] Alaa, A. M., Weisz, M., & Van Der Schaar, M. (2017). Deep counterfactual networks with propensity-dropout. arXiv preprint arXiv:1706.05966.

[12] Van der Laan, M. J., & Rose, S. (2011). Targeted learning: causal inference for observational and experimental data. Springer Science & Business Media.

[13] Veitch, V., Wang, Y., & Blei, D. (2019). Using embeddings to correct for unobserved confounding in networks. In Advances in Neural Information Processing Systems (pp. 13769-13779).

[14] Rosenbaum, P. R., & Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. Biometrika, 70(1), 41-55.

[15] Gross, R. T. (1993). Infant Health and Development Program (IHDP): Enhancing the Outcomes of Low Birth Weight, Premature Infants in the United States, 1985-1988. Inter-university Consortium for Political and Social Research.

[16] Mathews, T. J., & Atkinson, J. O. (1998). Infant mortality statistics from the linked birth/infant death data set—1995 period data. Monthly Vital Statistics Reports, 46(6).